

# **Discrete Optimization in Chemical Space Reference Manual**

**by B. C. Rinderspacher**

**ARL-TR-6202**

**October 2012**

## **NOTICES**

### **Disclaimers**

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

# **Army Research Laboratory**

Aberdeen Proving Ground, MD 21005

---

---

**ARL-TR-6202**

**October 2012**

---

## **Discrete Optimization in Chemical Space Reference Manual**

**B. C. Rinderspacher**

**Weapons and Materials Research Directorate, ARL**

| REPORT DOCUMENTATION PAGE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                             |                              |                                         | Form Approved<br>OMB No. 0704-0188                         |                                                             |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|------------------------------|-----------------------------------------|------------------------------------------------------------|-------------------------------------------------------------|
| <p>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p><b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b></p> |                             |                              |                                         |                                                            |                                                             |
| 1. REPORT DATE (DD-MM-YYYY)<br>October 2012                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                             | 2. REPORT TYPE<br>Interim    |                                         | 3. DATES COVERED (From - To)<br>2009–2011                  |                                                             |
| 4. TITLE AND SUBTITLE<br>Discrete Optimization in Chemical Space Reference Manual                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                             |                              |                                         | 5a. CONTRACT NUMBER                                        |                                                             |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                             |                              |                                         | 5b. GRANT NUMBER                                           |                                                             |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                             |                              |                                         | 5c. PROGRAM ELEMENT NUMBER                                 |                                                             |
| 6. AUTHOR(S)<br>B. C. Rinderspacher                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                             |                              |                                         | 5d. PROJECT NUMBER                                         |                                                             |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                             |                              |                                         | 5e. TASK NUMBER                                            |                                                             |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                             |                              |                                         | 5f. WORK UNIT NUMBER                                       |                                                             |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>U.S. Army Research Laboratory<br>ATTN: RDRL-WMM-G<br>Aberdeen Proving Ground, MD 21005                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                             |                              |                                         | 8. PERFORMING ORGANIZATION<br>REPORT NUMBER<br>ARL-TR-6202 |                                                             |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                             |                              |                                         | 10. SPONSOR/MONITOR'S ACRONYM(S)                           |                                                             |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                             |                              |                                         | 11. SPONSOR/MONITOR'S REPORT<br>NUMBER(S)                  |                                                             |
| 12. DISTRIBUTION/AVAILABILITY STATEMENT<br>Approved for public release; distribution unlimited.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                             |                              |                                         |                                                            |                                                             |
| 13. SUPPLEMENTARY NOTES                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                             |                              |                                         |                                                            |                                                             |
| 14. ABSTRACT<br>We present the manual for using the constrained inverse molecular design program. The program uses an object-oriented approach implemented in C++. The manual includes instructions on setting up constrained optimizations of substitutional frameworks and the full application programming interface (API) necessary for extending and developing new capabilities.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                             |                              |                                         |                                                            |                                                             |
| 15. SUBJECT TERMS<br>Molecular optimization, software documentation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                             |                              |                                         |                                                            |                                                             |
| 16. SECURITY CLASSIFICATION OF:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                             |                              | 17. LIMITATION<br>OF ABSTRACT<br><br>UU | 18. NUMBER<br>OF PAGES<br><br>413                          | 19a. NAME OF RESPONSIBLE PERSON<br>B. C. Rinderspacher      |
| a. REPORT<br>UNCLASSIFIED                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | b. ABSTRACT<br>UNCLASSIFIED | c. THIS PAGE<br>UNCLASSIFIED |                                         |                                                            | 19b. TELEPHONE NUMBER (Include area code)<br>(410) 306-2811 |

# Contents

|                                                                           |           |
|---------------------------------------------------------------------------|-----------|
| <b>1. Introduction</b>                                                    | <b>1</b>  |
| 1.1 Usage . . . . .                                                       | 1         |
| 1.1.1 Command Line Options . . . . .                                      | 2         |
| 1.2 Installation . . . . .                                                | 2         |
| <b>2. Discrete Optimization in Chemical Space Module Documentation</b>    | <b>3</b>  |
| 2.1 Discrete Optimization of Chemical Space . . . . .                     | 3         |
| 2.1.1 Function Documentation . . . . .                                    | 3         |
| 2.2 Linear Algebra Package with Sparse Implementations . . . . .          | 4         |
| 2.2.1 Function Documentation . . . . .                                    | 7         |
| <b>3. Discrete Optimization in Chemical Space Namespace Documentation</b> | <b>8</b>  |
| 3.1 Boost Namespace Reference . . . . .                                   | 8         |
| 3.2 Concepts Namespace Reference . . . . .                                | 8         |
| 3.2.1 Function Documentation . . . . .                                    | 9         |
| 3.3 linear_algebra Namespace Reference . . . . .                          | 9         |
| 3.3.1 Function Documentation . . . . .                                    | 17        |
| 3.4 std Namespace Reference . . . . .                                     | 37        |
| <b>4. Discrete Optimization in Chemical Space Class Documentation</b>     | <b>37</b> |
| 4.1 linear_algebra::asymmetric< TN > Class Template Reference . . . . .   | 37        |
| 4.1.1 Member Function Documentation . . . . .                             | 38        |
| 4.2 Concepts::base_iterator< X > Class Template Reference . . . . .       | 38        |
| 4.2.1 Member Function Documentation . . . . .                             | 39        |

|       |                                                                 |    |
|-------|-----------------------------------------------------------------|----|
| 4.2.2 | Member Data Documentation . . . . .                             | 39 |
| 4.3   | binary_entropic< C > Class Template Reference . . . . .         | 40 |
| 4.3.1 | Detailed Description . . . . .                                  | 41 |
| 4.3.2 | Constructor & Destructor Documentation . . . . .                | 41 |
| 4.3.3 | Member Function Documentation . . . . .                         | 41 |
| 4.3.4 | Member Data Documentation . . . . .                             | 42 |
| 4.4   | binary_gdmc< C > Class Template Reference . . . . .             | 42 |
| 4.4.1 | Detailed Description . . . . .                                  | 44 |
| 4.4.2 | Constructor & Destructor Documentation . . . . .                | 44 |
| 4.4.3 | Member Function Documentation . . . . .                         | 44 |
| 4.4.4 | Member Data Documentation . . . . .                             | 45 |
| 4.5   | binary_line_search< C > Class Template Reference . . . . .      | 45 |
| 4.5.1 | Detailed Description . . . . .                                  | 46 |
| 4.5.2 | Member Function Documentation . . . . .                         | 46 |
| 4.6   | binary_steepest_descent< C > Class Template Reference . . . . . | 47 |
| 4.6.1 | Detailed Description . . . . .                                  | 48 |
| 4.6.2 | Member Function Documentation . . . . .                         | 49 |
| 4.7   | chem_opt Class Reference . . . . .                              | 50 |
| 4.7.1 | Detailed Description . . . . .                                  | 51 |
| 4.7.2 | Constructor & Destructor Documentation . . . . .                | 51 |
| 4.7.3 | Member Function Documentation . . . . .                         | 51 |
| 4.8   | ChemGroup Class Reference . . . . .                             | 52 |
| 4.8.1 | Detailed Description . . . . .                                  | 54 |

|        |                                                             |    |
|--------|-------------------------------------------------------------|----|
| 4.8.2  | Constructor & Destructor Documentation . . . . .            | 54 |
| 4.8.3  | Member Function Documentation . . . . .                     | 55 |
| 4.8.4  | Member Data Documentation . . . . .                         | 56 |
| 4.9    | ChemIdent Class Reference . . . . .                         | 57 |
| 4.9.1  | Detailed Description . . . . .                              | 60 |
| 4.9.2  | Constructor & Destructor Documentation . . . . .            | 60 |
| 4.9.3  | Member Function Documentation . . . . .                     | 61 |
| 4.9.4  | Member Data Documentation . . . . .                         | 62 |
| 4.10   | double_truct Reference . . . . .                            | 64 |
| 4.10.1 | Member Data Documentation . . . . .                         | 64 |
| 4.11   | gen_base_entropic< C > Class Template Reference . . . . .   | 64 |
| 4.11.1 | Constructor & Destructor Documentation . . . . .            | 66 |
| 4.11.2 | Member Function Documentation . . . . .                     | 66 |
| 4.11.3 | Member Data Documentation . . . . .                         | 66 |
| 4.12   | gen_base_gdmc< C > Class Template Reference . . . . .       | 67 |
| 4.12.1 | Constructor & Destructor Documentation . . . . .            | 68 |
| 4.12.2 | Member Function Documentation . . . . .                     | 68 |
| 4.12.3 | Member Data Documentation . . . . .                         | 69 |
| 4.13   | gen_base_grad_LS< C, B > Class Template Reference . . . . . | 69 |
| 4.13.1 | Constructor & Destructor Documentation . . . . .            | 71 |
| 4.13.2 | Member Function Documentation . . . . .                     | 71 |
| 4.13.3 | Member Data Documentation . . . . .                         | 72 |
| 4.14   | gen_base_LS< C, B > Class Template Reference . . . . .      | 72 |

|        |                                                                 |    |
|--------|-----------------------------------------------------------------|----|
| 4.14.1 | Constructor & Destructor Documentation . . . . .                | 74 |
| 4.14.2 | Member Function Documentation . . . . .                         | 74 |
| 4.14.3 | Member Data Documentation . . . . .                             | 74 |
| 4.15   | general_base_iterator< X > Class Template Reference . . . . .   | 75 |
| 4.15.1 | Constructor & Destructor Documentation . . . . .                | 77 |
| 4.15.2 | Member Function Documentation . . . . .                         | 78 |
| 4.15.3 | Member Data Documentation . . . . .                             | 79 |
| 4.16   | Concepts::has_gradients< X > Class Template Reference . . . . . | 79 |
| 4.16.1 | Member Function Documentation . . . . .                         | 80 |
| 4.16.2 | Member Data Documentation . . . . .                             | 81 |
| 4.17   | has_gradients_data< X > Class Template Reference . . . . .      | 81 |
| 4.17.1 | Constructor & Destructor Documentation . . . . .                | 82 |
| 4.17.2 | Member Function Documentation . . . . .                         | 82 |
| 4.18   | Concepts::has_hessians< X > Class Template Reference . . . . .  | 83 |
| 4.18.1 | Member Function Documentation . . . . .                         | 84 |
| 4.18.2 | Member Data Documentation . . . . .                             | 84 |
| 4.19   | has_hessians_data< X > Class Template Reference . . . . .       | 85 |
| 4.19.1 | Constructor & Destructor Documentation . . . . .                | 85 |
| 4.19.2 | Member Function Documentation . . . . .                         | 86 |
| 4.19.3 | Member Data Documentation . . . . .                             | 86 |
| 4.20   | Concepts::has_Name< X > Class Template Reference . . . . .      | 86 |
| 4.20.1 | Member Function Documentation . . . . .                         | 87 |
| 4.20.2 | Member Data Documentation . . . . .                             | 88 |



|        |                                                                        |     |
|--------|------------------------------------------------------------------------|-----|
| 4.21   | Concepts::has_optimize< X > Class Template Reference . . . . .         | 88  |
| 4.21.1 | Member Function Documentation . . . . .                                | 89  |
| 4.21.2 | Member Data Documentation . . . . .                                    | 89  |
| 4.22   | Concepts::has_stacksize< X > Class Template Reference . . . . .        | 89  |
| 4.22.1 | Member Function Documentation . . . . .                                | 90  |
| 4.22.2 | Member Data Documentation . . . . .                                    | 90  |
| 4.23   | Concepts::is_threadable< X > Class Template Reference . . . . .        | 90  |
| 4.23.1 | Member Function Documentation . . . . .                                | 91  |
| 4.23.2 | Member Data Documentation . . . . .                                    | 91  |
| 4.24   | is_threadable_abstract Class Reference . . . . .                       | 92  |
| 4.24.1 | Constructor & Destructor Documentation . . . . .                       | 93  |
| 4.24.2 | Member Function Documentation . . . . .                                | 93  |
| 4.24.3 | Member Data Documentation . . . . .                                    | 93  |
| 4.25   | Concepts::Library< X > Class Template Reference . . . . .              | 93  |
| 4.25.1 | Member Function Documentation . . . . .                                | 94  |
| 4.25.2 | Member Data Documentation . . . . .                                    | 95  |
| 4.26   | Library_data Class Reference . . . . .                                 | 95  |
| 4.26.1 | Detailed Description . . . . .                                         | 97  |
| 4.26.2 | Constructor & Destructor Documentation . . . . .                       | 97  |
| 4.26.3 | Member Function Documentation . . . . .                                | 97  |
| 4.26.4 | Member Data Documentation . . . . .                                    | 98  |
| 4.27   | linear_algebra::mat_asym_full< TN > Class Template Reference . . . . . | 98  |
| 4.27.1 | Detailed Description . . . . .                                         | 103 |

|        |                                                                                             |     |
|--------|---------------------------------------------------------------------------------------------|-----|
| 4.27.2 | Constructor & Destructor Documentation . . . . .                                            | 103 |
| 4.27.3 | Member Function Documentation . . . . .                                                     | 103 |
| 4.27.4 | Member Data Documentation . . . . .                                                         | 112 |
| 4.28   | <code>linear_algebra::mat_asym_sparse&lt; TN &gt;</code> Class Template Reference . . . . . | 112 |
| 4.28.1 | Detailed Description . . . . .                                                              | 117 |
| 4.28.2 | Constructor & Destructor Documentation . . . . .                                            | 118 |
| 4.28.3 | Member Function Documentation . . . . .                                                     | 119 |
| 4.28.4 | Friends and Related Function Documentation . . . . .                                        | 129 |
| 4.28.5 | Member Data Documentation . . . . .                                                         | 129 |
| 4.29   | <code>linear_algebra::mat_full&lt; TN &gt;</code> Class Template Reference . . . . .        | 129 |
| 4.29.1 | Detailed Description . . . . .                                                              | 135 |
| 4.29.2 | Constructor & Destructor Documentation . . . . .                                            | 135 |
| 4.29.3 | Member Function Documentation . . . . .                                                     | 136 |
| 4.29.4 | Member Data Documentation . . . . .                                                         | 149 |
| 4.30   | <code>linear_algebra::mat_sparse&lt; TN &gt;</code> Class Template Reference . . . . .      | 149 |
| 4.30.1 | Detailed Description . . . . .                                                              | 153 |
| 4.30.2 | Constructor & Destructor Documentation . . . . .                                            | 153 |
| 4.30.3 | Member Function Documentation . . . . .                                                     | 154 |
| 4.30.4 | Friends and Related Function Documentation . . . . .                                        | 162 |
| 4.30.5 | Member Data Documentation . . . . .                                                         | 162 |
| 4.31   | <code>linear_algebra::mat_sym_full&lt; TN &gt;</code> Class Template Reference . . . . .    | 163 |
| 4.31.1 | Detailed Description . . . . .                                                              | 168 |
| 4.31.2 | Constructor & Destructor Documentation . . . . .                                            | 168 |

|        |                                                                                            |     |
|--------|--------------------------------------------------------------------------------------------|-----|
| 4.31.3 | Member Function Documentation . . . . .                                                    | 169 |
| 4.31.4 | Member Data Documentation . . . . .                                                        | 181 |
| 4.32   | <code>linear_algebra::mat_sym_sparse&lt; TN &gt;</code> Class Template Reference . . . . . | 182 |
| 4.32.1 | Detailed Description . . . . .                                                             | 187 |
| 4.32.2 | Constructor & Destructor Documentation . . . . .                                           | 188 |
| 4.32.3 | Member Function Documentation . . . . .                                                    | 190 |
| 4.32.4 | Friends and Related Function Documentation . . . . .                                       | 198 |
| 4.32.5 | Member Data Documentation . . . . .                                                        | 199 |
| 4.33   | <code>linear_algebra::matrix&lt; TN &gt;</code> Class Template Reference . . . . .         | 199 |
| 4.33.1 | Constructor & Destructor Documentation . . . . .                                           | 200 |
| 4.33.2 | Member Function Documentation . . . . .                                                    | 201 |
| 4.33.3 | Member Data Documentation . . . . .                                                        | 202 |
| 4.34   | <code>noprune&lt; X &gt;</code> Class Template Reference . . . . .                         | 202 |
| 4.34.1 | Constructor & Destructor Documentation . . . . .                                           | 203 |
| 4.34.2 | Member Function Documentation . . . . .                                                    | 203 |
| 4.35   | <code>optimize_abstract</code> Class Reference . . . . .                                   | 204 |
| 4.35.1 | Constructor & Destructor Documentation . . . . .                                           | 205 |
| 4.35.2 | Member Function Documentation . . . . .                                                    | 205 |
| 4.35.3 | Member Data Documentation . . . . .                                                        | 206 |
| 4.36   | <code>linear_algebra::polynomial&lt; C &gt;</code> Class Template Reference . . . . .      | 206 |
| 4.36.1 | Constructor & Destructor Documentation . . . . .                                           | 208 |
| 4.36.2 | Member Function Documentation . . . . .                                                    | 208 |
| 4.36.3 | Member Data Documentation . . . . .                                                        | 213 |

|        |                                                                     |     |
|--------|---------------------------------------------------------------------|-----|
| 4.37   | Concepts::pruner< X > Class Template Reference . . . . .            | 213 |
| 4.37.1 | Member Function Documentation . . . . .                             | 214 |
| 4.37.2 | Member Data Documentation . . . . .                                 | 214 |
| 4.38   | pruner_abstract< X > Class Template Reference . . . . .             | 215 |
| 4.38.1 | Constructor & Destructor Documentation . . . . .                    | 216 |
| 4.38.2 | Member Function Documentation . . . . .                             | 217 |
| 4.38.3 | Member Data Documentation . . . . .                                 | 218 |
| 4.39   | linear_algebra::refcount< Type > Class Template Reference . . . . . | 218 |
| 4.39.1 | Constructor & Destructor Documentation . . . . .                    | 219 |
| 4.39.2 | Member Function Documentation . . . . .                             | 220 |
| 4.39.3 | Member Data Documentation . . . . .                                 | 220 |
| 4.40   | linear_algebra::refvector< TN > Class Template Reference . . . . .  | 220 |
| 4.40.1 | Constructor & Destructor Documentation . . . . .                    | 225 |
| 4.40.2 | Member Function Documentation . . . . .                             | 225 |
| 4.40.3 | Member Data Documentation . . . . .                                 | 233 |
| 4.41   | reorder_general_base< X > Class Template Reference . . . . .        | 233 |
| 4.41.1 | Detailed Description . . . . .                                      | 235 |
| 4.41.2 | Constructor & Destructor Documentation . . . . .                    | 235 |
| 4.41.3 | Member Function Documentation . . . . .                             | 235 |
| 4.41.4 | Member Data Documentation . . . . .                                 | 236 |
| 4.42   | simple_prune< X > Class Template Reference . . . . .                | 237 |
| 4.42.1 | Constructor & Destructor Documentation . . . . .                    | 238 |
| 4.42.2 | Member Function Documentation . . . . .                             | 238 |

|        |                                                                                                 |     |
|--------|-------------------------------------------------------------------------------------------------|-----|
| 4.43   | <code>linear_algebra::sparse_vector&lt; TN &gt;</code> Class Template Reference . . . . .       | 238 |
| 4.43.1 | Constructor & Destructor Documentation . . . . .                                                | 242 |
| 4.43.2 | Member Function Documentation . . . . .                                                         | 243 |
| 4.43.3 | Friends and Related Function Documentation . . . . .                                            | 248 |
| 4.43.4 | Member Data Documentation . . . . .                                                             | 249 |
| 4.44   | <code>linear_algebra::sparse_vector_infty&lt; TN &gt;</code> Class Template Reference . . . . . | 249 |
| 4.44.1 | Constructor & Destructor Documentation . . . . .                                                | 252 |
| 4.44.2 | Member Function Documentation . . . . .                                                         | 253 |
| 4.44.3 | Member Data Documentation . . . . .                                                             | 259 |
| 4.45   | <code>linear_algebra::square&lt; TN &gt;</code> Class Template Reference . . . . .              | 259 |
| 4.45.1 | Member Function Documentation . . . . .                                                         | 260 |
| 4.46   | <code>boost::stateful_thread_group</code> Class Reference . . . . .                             | 260 |
| 4.47   | <code>linear_algebra::symmetric&lt; TN &gt;</code> Class Template Reference . . . . .           | 260 |
| 4.47.1 | Member Function Documentation . . . . .                                                         | 262 |
| 4.48   | <code>valerg</code> Struct Reference . . . . .                                                  | 264 |
| 4.48.1 | Member Data Documentation . . . . .                                                             | 264 |
| 4.49   | <code>zmat</code> Class Reference . . . . .                                                     | 264 |
| 4.49.1 | Detailed Description . . . . .                                                                  | 267 |
| 4.49.2 | Constructor & Destructor Documentation . . . . .                                                | 267 |
| 4.49.3 | Member Function Documentation . . . . .                                                         | 267 |
| 4.49.4 | Member Data Documentation . . . . .                                                             | 272 |
| 4.50   | <code>zmat_connector</code> Class Reference . . . . .                                           | 272 |
| 4.50.1 | Detailed Description . . . . .                                                                  | 275 |

|           |                                                                   |            |
|-----------|-------------------------------------------------------------------|------------|
| 4.50.2    | Constructor & Destructor Documentation . . . . .                  | 275        |
| 4.50.3    | Member Function Documentation . . . . .                           | 275        |
| 4.50.4    | Member Data Documentation . . . . .                               | 276        |
| 4.51      | zmat_entry Class Reference . . . . .                              | 277        |
| 4.51.1    | Constructor & Destructor Documentation . . . . .                  | 279        |
| 4.51.2    | Member Function Documentation . . . . .                           | 279        |
| 4.51.3    | Friends and Related Function Documentation . . . . .              | 280        |
| 4.51.4    | Member Data Documentation . . . . .                               | 280        |
| 4.52      | zmat_opt Class Reference . . . . .                                | 280        |
| 4.52.1    | Detailed Description . . . . .                                    | 282        |
| 4.52.2    | Constructor & Destructor Documentation . . . . .                  | 282        |
| 4.52.3    | Member Function Documentation . . . . .                           | 282        |
| 4.52.4    | Member Data Documentation . . . . .                               | 283        |
| <b>5.</b> | <b>Discrete Optimization in Chemical Space File Documentation</b> | <b>284</b> |
| 5.1       | binary_entropic.hh File Reference . . . . .                       | 284        |
| 5.1.1     | Detailed Description . . . . .                                    | 284        |
| 5.2       | binary_line_search.hh File Reference . . . . .                    | 284        |
| 5.2.1     | Detailed Description . . . . .                                    | 285        |
| 5.3       | binary_opt.cc File Reference . . . . .                            | 285        |
| 5.3.1     | Function Documentation . . . . .                                  | 286        |
| 5.4       | binarygdmc.hh File Reference . . . . .                            | 286        |
| 5.4.1     | Detailed Description . . . . .                                    | 286        |
| 5.5       | binarysteepestdescent.hh File Reference . . . . .                 | 286        |

|        |                                          |     |
|--------|------------------------------------------|-----|
| 5.5.1  | Detailed Description . . . . .           | 287 |
| 5.6    | chem_opt.cc File Reference . . . . .     | 287 |
| 5.6.1  | Detailed Description . . . . .           | 287 |
| 5.6.2  | Define Documentation . . . . .           | 287 |
| 5.7    | chem_opt.hh File Reference . . . . .     | 287 |
| 5.8    | chemgroup.cc File Reference . . . . .    | 288 |
| 5.8.1  | Detailed Description . . . . .           | 288 |
| 5.8.2  | Function Documentation . . . . .         | 288 |
| 5.9    | chemgroup.hh File Reference . . . . .    | 289 |
| 5.9.1  | Detailed Description . . . . .           | 289 |
| 5.9.2  | Typedef Documentation . . . . .          | 289 |
| 5.10   | chemident.cc File Reference . . . . .    | 289 |
| 5.10.1 | Detailed Description . . . . .           | 290 |
| 5.10.2 | Function Documentation . . . . .         | 290 |
| 5.10.3 | Variable Documentation . . . . .         | 290 |
| 5.11   | chemident.hh File Reference . . . . .    | 290 |
| 5.11.1 | Detailed Description . . . . .           | 291 |
| 5.12   | concepts.hh File Reference . . . . .     | 291 |
| 5.12.1 | Detailed Description . . . . .           | 292 |
| 5.13   | entropic_aux.cc File Reference . . . . . | 292 |
| 5.13.1 | Function Documentation . . . . .         | 292 |
| 5.14   | entropic_aux.hh File Reference . . . . . | 293 |
| 5.14.1 | Detailed Description . . . . .           | 294 |

|        |                                                                   |     |
|--------|-------------------------------------------------------------------|-----|
| 5.14.2 | Function Documentation . . . . .                                  | 294 |
| 5.15   | gen_base_entropic.hh File Reference . . . . .                     | 294 |
| 5.16   | genbase-grad-ls.hh File Reference . . . . .                       | 294 |
| 5.16.1 | Detailed Description . . . . .                                    | 295 |
| 5.17   | genbase-l-s.hh File Reference . . . . .                           | 295 |
| 5.17.1 | Detailed Description . . . . .                                    | 295 |
| 5.18   | generalbaseiterator.cc File Reference . . . . .                   | 295 |
| 5.18.1 | Detailed Description . . . . .                                    | 296 |
| 5.19   | generalbaseiterator.hh File Reference . . . . .                   | 296 |
| 5.19.1 | Detailed Description . . . . .                                    | 296 |
| 5.20   | has_gradients_hessian_data.decl File Reference . . . . .          | 296 |
| 5.21   | has_gradients_hessian_data.hh File Reference . . . . .            | 296 |
| 5.22   | include/BCR_CPP_LA/ABmBA.hh File Reference . . . . .              | 297 |
| 5.22.1 | Detailed Description . . . . .                                    | 298 |
| 5.23   | include/BCR_CPP_LA/ABpBA.hh File Reference . . . . .              | 298 |
| 5.23.1 | Detailed Description . . . . .                                    | 300 |
| 5.24   | include/BCR_CPP_LA/asymmetric.decl File Reference . . . . .       | 300 |
| 5.24.1 | Detailed Description . . . . .                                    | 301 |
| 5.25   | include/BCR_CPP_LA/asymmetric.h File Reference . . . . .          | 301 |
| 5.25.1 | Detailed Description . . . . .                                    | 301 |
| 5.26   | include/BCR_CPP_LA/class_extensions.decl File Reference . . . . . | 301 |
| 5.26.1 | Detailed Description . . . . .                                    | 304 |
| 5.27   | include/BCR_CPP_LA/class_extensions.hh File Reference . . . . .   | 304 |



|                                                                       |     |
|-----------------------------------------------------------------------|-----|
| 5.27.1 Detailed Description . . . . .                                 | 306 |
| 5.28 include/BCR_CPP_LA/invert_sym.h File Reference . . . . .         | 306 |
| 5.28.1 Detailed Description . . . . .                                 | 307 |
| 5.29 Function Documentation . . . . .                                 | 307 |
| 5.30 include/BCR_CPP_LA/linear_algebra.decl File Reference . . . . .  | 308 |
| 5.30.1 Detailed Description . . . . .                                 | 309 |
| 5.31 include/BCR_CPP_LA/linear_algebra.h File Reference . . . . .     | 309 |
| 5.32 include/BCR_CPP_LA/mat_asym_full.decl File Reference . . . . .   | 309 |
| 5.32.1 Detailed Description . . . . .                                 | 310 |
| 5.33 include/BCR_CPP_LA/mat_asym_full.h File Reference . . . . .      | 310 |
| 5.33.1 Detailed Description . . . . .                                 | 311 |
| 5.34 include/BCR_CPP_LA/mat_asym_sparse.decl File Reference . . . . . | 311 |
| 5.35 include/BCR_CPP_LA/mat_asym_sparse.h File Reference . . . . .    | 311 |
| 5.36 include/BCR_CPP_LA/mat_full.decl File Reference . . . . .        | 312 |
| 5.36.1 Detailed Description . . . . .                                 | 312 |
| 5.37 include/BCR_CPP_LA/mat_full.h File Reference . . . . .           | 312 |
| 5.38 mat_full.h File Reference . . . . .                              | 313 |
| 5.39 include/BCR_CPP_LA/mat_sparse.decl File Reference . . . . .      | 313 |
| 5.39.1 Detailed Description . . . . .                                 | 314 |
| 5.40 include/BCR_CPP_LA/mat_sparse.h File Reference . . . . .         | 314 |
| 5.40.1 Detailed Description . . . . .                                 | 314 |
| 5.41 include/BCR_CPP_LA/mat_sym_full.decl File Reference . . . . .    | 314 |
| 5.41.1 Detailed Description . . . . .                                 | 315 |

|        |                                                                       |     |
|--------|-----------------------------------------------------------------------|-----|
| 5.42   | include/BCR_CPP_LA/mat_sym_full.h File Reference . . . . .            | 315 |
| 5.42.1 | Detailed Description . . . . .                                        | 315 |
| 5.43   | include/BCR_CPP_LA/mat_sym_sparse.decl File Reference . . . . .       | 316 |
| 5.43.1 | Detailed Description . . . . .                                        | 316 |
| 5.44   | include/BCR_CPP_LA/mat_sym_sparse.h File Reference . . . . .          | 316 |
| 5.44.1 | Detailed Description . . . . .                                        | 317 |
| 5.45   | include/BCR_CPP_LA/mat_sym_sparse_eigen.decl File Reference . . . . . | 317 |
| 5.45.1 | Detailed Description . . . . .                                        | 317 |
| 5.46   | include/BCR_CPP_LA/mat_sym_sparse_eigen.h File Reference . . . . .    | 317 |
| 5.46.1 | Detailed Description . . . . .                                        | 317 |
| 5.47   | include/BCR_CPP_LA/matrix.decl File Reference . . . . .               | 318 |
| 5.47.1 | Detailed Description . . . . .                                        | 318 |
| 5.48   | include/BCR_CPP_LA/matrix.h File Reference . . . . .                  | 318 |
| 5.48.1 | Detailed Description . . . . .                                        | 319 |
| 5.49   | include/BCR_CPP_LA/polynomial.decl File Reference . . . . .           | 319 |
| 5.49.1 | Detailed Description . . . . .                                        | 319 |
| 5.50   | include/BCR_CPP_LA/polynomial.h File Reference . . . . .              | 319 |
| 5.50.1 | Detailed Description . . . . .                                        | 320 |
| 5.51   | include/BCR_CPP_LA/refcount.decl File Reference . . . . .             | 320 |
| 5.51.1 | Detailed Description . . . . .                                        | 320 |
| 5.52   | include/BCR_CPP_LA/refcount.h File Reference . . . . .                | 320 |
| 5.52.1 | Detailed Description . . . . .                                        | 321 |
| 5.53   | include/BCR_CPP_LA/sparse_vector.decl File Reference . . . . .        | 321 |

|        |                                                                      |     |
|--------|----------------------------------------------------------------------|-----|
| 5.53.1 | Detailed Description . . . . .                                       | 322 |
| 5.54   | include/BCR_CPP_LA/sparse_vector.h File Reference . . . . .          | 322 |
| 5.54.1 | Detailed Description . . . . .                                       | 323 |
| 5.55   | include/BCR_CPP_LA/sparse_vector_infty.decl File Reference . . . . . | 323 |
| 5.55.1 | Detailed Description . . . . .                                       | 323 |
| 5.56   | include/BCR_CPP_LA/sparse_vector_infty.h File Reference . . . . .    | 323 |
| 5.56.1 | Detailed Description . . . . .                                       | 323 |
| 5.57   | include/BCR_CPP_LA/square.decl File Reference . . . . .              | 323 |
| 5.57.1 | Detailed Description . . . . .                                       | 324 |
| 5.58   | include/BCR_CPP_LA/square.h File Reference . . . . .                 | 324 |
| 5.58.1 | Detailed Description . . . . .                                       | 324 |
| 5.59   | include/BCR_CPP_LA/symmetric.decl File Reference . . . . .           | 324 |
| 5.59.1 | Detailed Description . . . . .                                       | 325 |
| 5.60   | include/BCR_CPP_LA/symmetric.h File Reference . . . . .              | 325 |
| 5.60.1 | Detailed Description . . . . .                                       | 325 |
| 5.61   | include/BCR_CPP_LA/trace_AB.hh File Reference . . . . .              | 325 |
| 5.61.1 | Detailed Description . . . . .                                       | 326 |
| 5.62   | include/sorting_functions.hh File Reference . . . . .                | 326 |
| 5.62.1 | Function Documentation . . . . .                                     | 326 |
| 5.63   | include/stateful_thread_group.hh File Reference . . . . .            | 326 |
| 5.64   | stateful_thread_group.hh File Reference . . . . .                    | 327 |
| 5.65   | isthreadableabstract.h File Reference . . . . .                      | 328 |
| 5.65.1 | Detailed Description . . . . .                                       | 328 |

|        |                                                |     |
|--------|------------------------------------------------|-----|
| 5.66   | Library_data.cc File Reference . . . . .       | 328 |
| 5.66.1 | Detailed Description . . . . .                 | 329 |
| 5.67   | Library_data.hh File Reference . . . . .       | 329 |
| 5.68   | noprune.h File Reference . . . . .             | 329 |
| 5.68.1 | Detailed Description . . . . .                 | 329 |
| 5.69   | optimizeabstract.h File Reference . . . . .    | 329 |
| 5.69.1 | Detailed Description . . . . .                 | 330 |
| 5.70   | prunerabstract.h File Reference . . . . .      | 330 |
| 5.70.1 | Detailed Description . . . . .                 | 330 |
| 5.71   | reordergeneralbase.hh File Reference . . . . . | 330 |
| 5.72   | simpleprune.cc File Reference . . . . .        | 330 |
| 5.72.1 | Detailed Description . . . . .                 | 330 |
| 5.73   | simpleprune.h File Reference . . . . .         | 330 |
| 5.73.1 | Detailed Description . . . . .                 | 331 |
| 5.74   | typedefs.hh File Reference . . . . .           | 331 |
| 5.74.1 | Typedef Documentation . . . . .                | 332 |
| 5.74.2 | Function Documentation . . . . .               | 332 |
| 5.74.3 | Detailed Description . . . . .                 | 333 |
| 5.75   | zmat.hh File Reference . . . . .               | 333 |
| 5.75.1 | Detailed Description . . . . .                 | 333 |
| 5.76   | zmat_opt.cc File Reference . . . . .           | 333 |
| 5.76.1 | Detailed Description . . . . .                 | 334 |
| 5.76.2 | Function Documentation . . . . .               | 334 |

|           |                                                                      |            |
|-----------|----------------------------------------------------------------------|------------|
| 5.77      | zmat_opt.hh File Reference . . . . .                                 | 334        |
| 5.77.1    | Detailed Description . . . . .                                       | 334        |
| <b>6.</b> | <b>Discrete Optimization in Chemical Space Example Documentation</b> | <b>334</b> |
| 6.1       | carbazoles.inp . . . . .                                             | 334        |
| 6.2       | ChemGroup . . . . .                                                  | 346        |
| 6.3       | vanilla-rings.inp . . . . .                                          | 346        |
|           | <b>Distribution</b>                                                  | <b>391</b> |

---

# 1. Introduction

---

## 1.1 Usage

These modules include a reference counted Linear Algebra namespace (`linear_algebra`) for easy manipulation and a general substitution class based on recursion (`ChemIdent`). In the main driver, the function `calc_property()` executes *./property\_script*. The secondary driver executes an energy computation via *./energy\_script*.

To test the program, copy all files from the test directory to your working directory and invoke

```
/path_to_src/discrete_opt ftc-input.inp
```

The result should be a simple optimization of the hyperpolarizability.

`calc_property()` will generate a *.zmat* file which will hold the Z-matrix of the current molecule. The prefix will be the current number of visited configurations followed by the current conformation and overall structures visited. See `calc_property()` for details. *energy\_script* must return *.rconsts*, *.rvars*, *.energy* files on a successful run.

- *.rconsts* must contain the optimized constants of the Z-matrix from the geometry optimization.
- *.rvars* must contain the optimized variables of the Z-matrix from the geometry optimization.
- *.result* holds the result of the objective function.
- *.energy* holds the energy of the final geometry.

*property\_script* must return *.result* and potentially *.penalty* files on a successful run.

- *.result* holds the result of the objective function.
- *.penalty* contains a penalty value.

To access the program flow go to `main()`. For information on the representation of molecules see class `ChemIdent`. For information on the input file syntax see `ChemIdent::ChemIdent(stringstream& s)`.

### 1.1.1 Command Line Options

#### Parameters:

`-p` Switches pruning or heuristics on.

#### See also:

`noprune`, `simple_prune`, `reorder_general_base`.

#### Parameters:

**-sub-method** <method>

**-sm** <method> details the sub-method to be used. Currently this includes *binary\_line\_search*, *GBLS* (*gen\_base\_LS*), *GBGLS* (*gen\_base\_grad\_LS*), *SD*, *steepest\_descent* (*binary\_steepest\_descent*)

**-m** <method> Supplies the method to be used. Options include all listed under sub-methods and *GBEN* (*gen\_base\_entropic*), *GDMC* (*binary\_gdmc*)

**-T** <temperature> provides a fictitious temperature for GDMC calculations.

**-TS** <steps> provides the number of steps to tighten selection criteria for GDMC.

**-MS** <steps> Maximum number of evaluations used in GBEN and GDMC.

## 1.2 Installation

- Untar and unzip the tarball. The source will be extracted into `discrete_opt/`.
- Change directory to `discrete_opt/src` and edit Doxyfile to change the documentation path to your liking.
- To build the code, execute: `make`
- For Documentation execute:

```
make docs
```

- For the debugable version execute:

```
make discrete_opt.d
```

MakeVars and Makefile contain the compilation information which you may want to change. The executable will be named *discrete\_opt*.

---

## 2. Discrete Optimization in Chemical Space Module Documentation

---

### 2.1 Discrete Optimization of Chemical Space

#### Files

- file `Library_data.cc`  
*Implementations of the Library class for discrete spaces.*

#### Classes

- class `has_gradients_data < X >`  
*Abstract class for discrete optimizations.*
- class `has_hessians_data < X >`  
*Abstract class for discrete optimizations.*
- class `Library_data`  
*Abstract class for discrete optimizations.*

#### Functions

- class `Library_data & Library_data::operator= (const Library_data &d)`  
*Assignment operator.*
- void `Library_data::set_Name (const string &A) const`  
*Sets the name as written for the purpose of writing files or other I/O.*

#### 2.1.1 Function Documentation

2.1.1.1 `Library_data & Library_data::operator= (const Library_data & d)`  
[inherited]

2.1.1.2 void `Library_data::set_Name (const string & A) const`  
[inherited]



## 2.2 Linear Algebra Package with Sparse Implementations

### Files

- file ABmBA.hh
- file ABpBA.hh
- file asymmetric.decl  
*Provide declarations for antisymmetric full matrices.*
- file asymmetric.h  
*Provide manipulations for antisymmetric full matrices.*
- file class\_extensions.decl  
*Declarations of general expansions for LA.*
- file class\_extensions.hh  
*General expansions for LA.*
- file linear\_algebra.decl  
*Namespace Class Declarations.*
- file mat\_asym\_full.decl  
*Provide declarations for antisymmetric full matrices.*
- file mat\_asym\_full.h  
*Provide manipulations for antisymmetric full matrices.*
- file mat\_sym\_sparse.decl  
*Provide declarations for antisymmetric sparse matrices.*
- file mat\_sym\_sparse.h  
*Provide functionality for symmetric sparse matrices.*
- file mat\_full.decl  
*Provide declarations for full matrix and vector interactions.*
- file mat\_sparse.decl  
*Provide declarations for sparse matrix and vector interactions.*
- file mat\_sparse.h  
*Provide functionality for sparse matrix and vector interactions.*
- file mat\_sym\_full.decl  
*Provide declarations for symmetric full matrices.*

- file `mat_sym_full.h`  
*Provide manipulations for symmetric full matrices.*
- file `mat_sym_sparse_eigen.decl`  
*Eigenvalue related declarations.*
- file `mat_sym_sparse_eigen.h`  
*Eigenvalue related routines.*
- file `matrix.decl`  
*Declares fundamentals of matrices.*
- file `matrix.h`  
*Describes fundamentals of matrices.*
- file `polynomial.decl`  
*Class declarations for polynomial.*
- file `polynomial.h`  
*Implementations for a class of polynomials.*
- file `refcount.decl`  
*Contains the declarations to a reference counted vector class as well as general reference counting.*
- file `refcount.h`  
*Contains a reference counted vector class as well as general reference counting.*
- file `sparse_vector.decl`  
*Provide declarations for sparse vector interactions.*
- file `sparse_vector.h`  
*Provide functionality for sparse vectors.*
- file `sparse_vector_infty.decl`  
*Provide declarations for sparse vector interactions of infinite dimensions.*
- file `sparse_vector_infty.h`  
*Provide functionality for sparse vectors of infinite dimension.*
- file `square.decl`  
*Declares fundamentals of square matrices.*
- file `square.h`  
*Describes fundamentals of matrices.*

- file `symmetric.decl`  
*Declares fundamentals of matrices.*
- file `symmetric.h`  
*Describes fundamentals of symmetric matrices.*
- file `trace_AB.hh`  
*get the trace of matrix products.*

## Namespaces

- namespace `linear_algebra`  
*Linear algebra Namespace.*

## Functions

- `void linear_algebra::mat_sym_sparse::gen_mat (const sparse_vector< TN > &a, const sparse_vector< TN > &b, TN correction)`  
*Produce the necessary  $ij^* + ji^*$ .*
- `mat_sparse< TN > linear_algebra::mat_sym_sparse::householder (vector< TN > &diagonal, vector< TN > &subdiagonal)`  
*This routine produces the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.*
- `mat_sparse< TN > linear_algebra::mat_sym_sparse::householder (refvector< TN > &diagonal, refvector< TN > &subdiagonal)`  
*This routine produces the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.*
- `void gen_mat (const sparse_vector< TN > &a, const sparse_vector< TN > &b, TN correction)`  
*Produce the necessary  $ij^* + ji^*$ .*
- `mat_sparse< TN > householder (vector< TN > &diagonal, vector< TN > &subdiagonal)`  
*This routine produces the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.*
- `mat_sparse< TN > householder (refvector< TN > &diagonal, refvector< TN > &subdiagonal)`  
*This routine produces the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.*

- `template<class TN> void std::operator>> (const vector< TN > &a, ostream &OUT)`
- `template<typename TN> void std::operator>> (const TN &a, ostream &OUT)`

## 2.2.1 Function Documentation

2.2.1.1 `void gen_mat (const sparse_vector< TN > & a, const sparse_vector< TN > & b, TN correction)`

2.2.1.2 `template<class TN> void linear_algebra::mat_sym_sparse< TN >::gen_mat (const sparse_vector< TN > & a, const sparse_vector< TN > & b, TN correction)` [inherited]

2.2.1.3 `mat_sparse< TN > householder (refvector < TN > & diagonal, refvector< TN > & subdiagonal)`

2.2.1.4 `mat_sparse< TN >householder (vector< TN > & diagonal, vector< TN > & subdiagonal)`

2.2.1.5 `template<class TN> mat_sparse< TN > linear_algebra::mat_sym_—sparse< TN >::householder (refvector< TN > & diagonal, refvector< TN > & subdiagonal)`[inherited]

2.2.1.6 `template<class TN> mat_sparse< TN > linear_algebra::mat_sym_—sparse< TN >::householder (vector< TN > & diagonal), vector< TN > & subdiagonal)`[inherited]The aspired transformation is a multiplication by  $(I - vv^*)$  where  $v$  is  $(\pm e_1 \|a_{12}\| + a_{12}) / \sqrt{\pm a_{12,1} \|a_{12}\| + \|a_{12}\|^2}$ . Notice that we begin counting at 0. Also see `mat_sym_full::householder` on the transformstions.

2.2.1.7 `template<typename TN> void std::operator>> (const TN & a, ostream & OUT)`

2.2.1.8 `template<class TN> void std::operator>> (const vector< TN > & a, ostream & OUT)`

---

### 3. Discrete Optimization in Chemical Space Namespace Documentation

---

#### 3.1 Boost Namespace Reference

##### Classes

- class `stateful_thread_group`

#### 3.2 Concepts Namespace Reference

Namespace reserved for concepts.

##### Classes

- class `Library`  
*Defines the concept of an addressable set of values.*
- class `base_iterator`  
*Defines the concept of an iterator for a class of bases.*
- class `pruner`  
*Defines the concept of an addressable set of values.*
- class `has_gradients`  
*Defines the concept of a class that has gradients.*
- class `has_hessians`  
*Defines the concept of a class that has hessians.*
- class `has_Name`  
*Defines the concept of a class that has a private string Name.*
- class `is_threadable`  
*Defines the concept of a class that is threadable.*
- class `has_optimize`  
*Defines the concept of a class that has an optimize member.*
- class `has_stacksize`  
*Defines the concept of a class that has a stack member.*

## Functions

- `template<typename T> void same_type (T const &, T const &)`  
*Dummy function to determine type equality.*

### 3.2.1 Function Documentation

#### 3.2.1.1 `template<typename T> void Concepts::same_type (T const &, T const &)`

## 3.3 linear\_algebra Namespace Reference

Linear algebra Namespace.

## Classes

- `class asymmetric`  
*Class of antisymmetric matrices.*
- `class mat_asym_full`  
*mat\_asym\_full uses packing for full antisymmetric matrices.*
- `class mat_asym_sparse`  
*mat\_asym\_sparse packs sparse column matrices into a comfortable format.*
- `class mat_full`  
*Class of full column matrices.*
- `class mat_sparse`  
*Class of sparse column matrices.*
- `class mat_sym_full`  
*mat\_sym\_full uses packing for full symmetric matrices.*
- `class mat_sym_sparse`  
*mat\_sym\_sparse packs sparse column matrices into a comfortable format.*
- `class matrix`  
*matrix is a basic class which covers all kinds of matrices.*
- `class polynomial`  
*A class for polynomials.*

- class refcount  
*Class refcount provides reference counted pointers. Most basic use.*
- class refvector  
*Class refvector provides reference counted vectors.*
- class sparse\_vector  
*A class for sparse vectors.*
- class sparse\_vector\_infty  
*A class for sparse vectors.*
- class square  
*The square matrix class contains all  $n$  by  $n$  matrices.*
- class symmetric  
*The symmetric class contains all symmetric matrices.*

## Functions

- `template<class TN> mat_sym_full< TN > & (const mat_sym_full< TN > A, const mat_asym_full< TN > B, mat_sym_full< TN > &r)`  
*Computes the commutator  $AB-BA$ .*
- `template<class TN> mat_asym_full< TN > ABmBA (const mat_sym_full< TN > A, const mat_asym_full< TN > B)`  
*Computes the commutator  $AB-BA$ .*
- `template<class TN> mat_asym_full< TN > & ABmBA (const mat_sym_full< TN > B, mat_asym_full< TN > &r)`  
*Computes the commutator  $AB-BA$ .*
- `template<class TN> mat_asym_full< TN > ABmBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B)`  
*Computes the commutator  $AB-BA$ .*
- `template<class TN> mat_asym_full< TN > & ABmBA (const mat_asym_full< TN > &A, const mat_asym_full< TN > &B, mat_asym_full< TN > &r)`  
*Computes the commutator  $AB-BA$ .*
- `template<class TN> mat_asym_full< TN > ABmBA (const mat_asym_full< TN > &A, const mat_asym_full< TN > &B)`  
*Computes the commutator  $AB-BA$ .*

- `template<class TN> mat_asym_full< TN > & ABpBA (const mat_sym_full< TN > &A, const mat_asym_full< TN > &B, mat_asym_full< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > ABpBA (const mat_sym_full< TN > &A, const mat_asym_full< TN > &B)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > ABpBA (const mat_full< TN > &A, const mat_asym_full< TN > &B, mat_asym_full< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > ABpBA (const mat_full< TN > A, const mat_asym_full< TN > B)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > & ABpBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B, mat_asym_full< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > ABpBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > & ABpBA (const mat_sym_full< TN > A, const mat_asym_sparse< TN > B, mat_asym_full< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > ABpBA (const mat_sym_full< TN > A, const mat_asym_sparse< TN > B)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > & ABpBA (const mat_asym_sparse< TN > A, const mat_sym_full< TN > B, mat_asym_full< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > ABpBA (const mat_asym_sparse< TN > A, const mat_sym_full< TN > B)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_sym_full< TN > & ABpBA (const mat_sym_full< TN > A, mat_sym_full< TN > B, mat_sym_full< TN > &r, mat_sym_full< TN > &I, mat_full< TN > &I2)`  
*Computes the anti-commutator  $AB+BA$ .*



- `template<class TN> mat_sym_full< TN > & ABpBA (const mat_sym_full< TN > A, mat_sym_full< TN > B, mat_sym_full< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_sym_full< TN > & ABpBA (const mat_sym_full< TN > &A, const mat_sym_full< TN > &B, mat_sym_full< TN > &r, mat_sym_full< TN > &I)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_sym_full< TN > ABpBA (const mat_sym_full< TN > &A, const mat_sym_full< TN > &B)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_sym_full< TN > & ABpBA (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_sym_full< TN > &r, mat_full< TN > &I2)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_sym_full< TN > & ABpBA (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_sym_full< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_sym_full< TN > ABpBA (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > & ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_full< TN > &r, mat_full< TN > &I2)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_sparse< TN > & ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_sparse< TN > &r, mat_sparse< TN > &I2)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > & ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_full< TN > &r, mat_sparse< TN > &I2)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > & ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_full< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*

- `template<class TN> mat_asym_sparse< TN > & ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_sparse< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class T> mat_sym_full< T > ABpBA (const mat_asym_full< T > &A, const mat_asym_full< T > &B)`  
*Commutes the anti-commutator  $AB+BA$ . Lame and lazy version.*
- `template<class TN> matrix< TN > & zero (matrix< TN > &r)`
- `template<class TN> mat_full< TN > & zero (mat_full< TN > &r)`
- `template<class TN> mat_sym_full< TN > & zero (mat_sym_full< TN > &r)`
- `template<class TN> mat_asym_full< TN > & zero (mat_asym_full< TN > &r)`
- `template<class TN> refvector< TN > & zero (refvector< TN > &r)`
- `double & zero (double &r)`
- `double & one (double &r)`
- `template<class TN> matrix< TN > & one (matrix< TN > &r)`
- `void copy (double &a, const double &b, const long i)`
- `void copy (double &a, const double &b)`
- `void copy (long &a, const long &b, const long i)`

- `void copy (long &a, const long &b)`
- `template<class TN> void copy (mat_sym_sparse< TN > &a, mat_sym_sparse< TN > &b, const long i)`
- `template<class TN> void copy (mat_sym_sparse< TN > &a, const mat_sym_sparse< TN > &b)`
- `template<class TN> void copy (mat_sym_sparse< TN > &a, mat_sym_sparse< TN > &b)`
- `template<class TN> void copy (mat_sym_full< TN > &a, mat_sym_full< TN > &b, const long i)`
- `template<class TN> void copy (mat_sym_full< TN > &a, const mat_sym_full< TN > &b)`
- `template<class TN> void copy (mat_sym_full< TN > &a, mat_sym_full< TN > &b)`
- `template<class TN> void copy (mat_full< TN > &a, mat_full< TN > &b, const long i)`
- `template<class TN> void copy (mat_full< TN > &a, const mat_full< TN > &b)`
- `template<class TN> void copy (mat_full< TN > &a, mat_full< TN > &b)`
- `template<class TN> void copy (mat_sparse< TN > &a, mat_sparse< TN > &b, const long i)`
- `template<class TN> void copy (mat_sparse< TN > &a, const mat_sparse< TN > &b)`
- `template<class TN> void copy (mat_sparse< TN > &a, mat_sparse< TN > &b)`

- `template<class TN> void copy (refvector< TN > &a, refvector< TN > &b, const long i)`
- `template<class TN> void copy (refvector< TN > &a, const refvector< TN > &b)`
- `template<class TN> void copy (refvector< TN > &a, refvector< TN > &b)`
- `template<class TN> void copy (sparse_vector< TN > &a, sparse_vector< TN > &b, const long i)`
- `template<class TN> void copy (sparse_vector< TN > &a, const sparse_vector< TN > &b)`
- `template<class C> void copy (C &a, const C &b)`
- `template<class TN> void display (const matrix< TN > &a, ostream &outs)`
- `template<class TN> void display (const refvector< TN > &a, ostream &outs)`
- `void display (const double &a, ostream &outs)`
- `template<class TNTN> mat_full< TNTN > exp (const mat_asym_full< TNTN > &A)`
- `template<class TNTN> mat_full< TNTN > dexp (mat_asym_full< TNTN > A, const long k, const long l)`
- `template<class TN> mat_sym_sparse< TN > & zero (mat_sym_sparse< TN > &r)`
- `template<class TN> mat_sparse< TN > & zero (mat_sparse< TN > &r)`
- `template<typename TN> TN & zero (TN &r)`

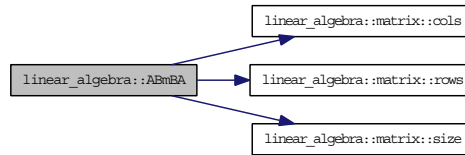
- `void copy (bool &a, const bool &b)`
- `template<class TN> void copy (mat_full< TN > &a, const mat_full< TN > &b, const long i)`
- `template<class TN> void copy (mat_sparse< TN > &a, const mat_sparse< TN > &b, const long i)`
- `template<class TN> void copy (mat_sym_sparse< TN > &a, const mat_sym_sparse< TN > &b, const long i)`
- `template<class TN> void copy (mat_sym_full< TN > &a, const mat_sym_full< TN > &b, const long i)`
- `template<class TN> void copy (refvector< TN > &a, const refvector< TN > &b, const long i)`
- `template<class TN> TN operator * (const std::vector< TN > &a, const std::vector< TN > &b)`  
*Scalar product of two full vectors.*
- `template<class TN> TN operator * (const vector< TN > &a, const vector< TN > &b)`  
*Scalar product of two full vectors.*
- `template<class TN> TN operator * (sparse_vector< TN > a, std::vector< TN > b)`  
*Scalar product of a sparse\_vector with a full vector.*
- `template<class TN> TN operator * (sparse_vector< TN > a, refvector< TN > b)`  
*Scalar product of a sparse\_vector with a full refvector.*
- `template<class TN> TN operator * (sparse_vector< TN > a, vector< TN > b)`  
*Scalar product of a sparse\_vector with a full vector.*
- `template<class TN> TN trace_AB (const mat_sym_full< TN > &A, const mat_sym_full< TN > &B)`

- `template<class TN> TN trace_AB (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B)`
- `template<class T> T trace_AB (const mat_asym_full< T > &A, const mat_asym_full< T > &B)`

### 3.3.1 Function Documentation

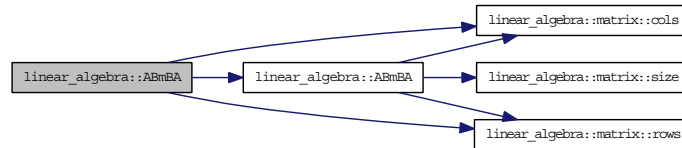
3.3.1.1 `template<class TN> mat_sym_full<TN>& linear_algebra::ABmBA (const mat_sym_full< TN > A, const mat_asym_full< TN > B, mat_sym_full< TN > & r)`

Here is the call graph for this function:



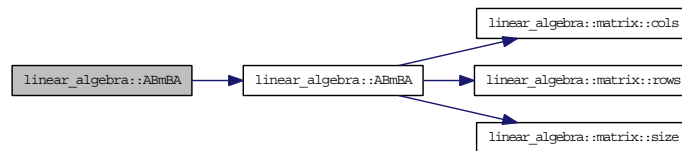
3.3.1.2 `template<class TN> mat_asym_full<TN> linear_algebra::ABmBA (const mat_sym_full< TN > A, const mat_asym_full< TN > B)`

Here is the call graph for this function:



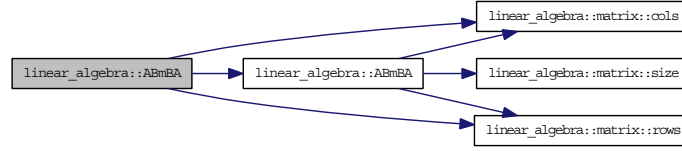
3.3.1.3 `template<class TN> mat_asym_full<TN>& linear_algebra::ABmBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B, mat_asym_full< TN > & r)`

Here is the call graph for this function:



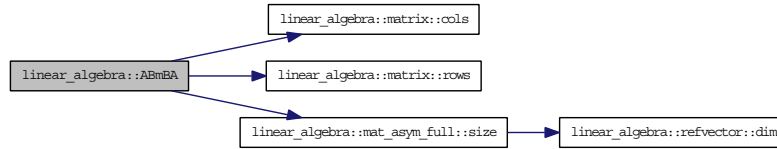
3.3.1.4 `template<class TN> mat_asym_full<TN> linear_algebra::ABmBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B)`

Here is the call graph for this function:



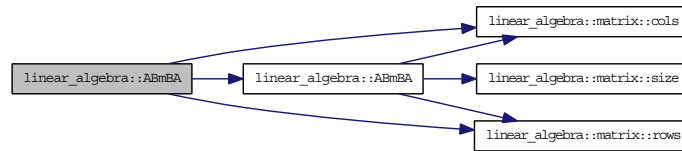
3.3.1.5 `template<class TN> mat_asym_full<TN>& linear_algebra::ABmBA (const mat_asym_full< TN > & A, const mat_asym_full< TN > & B, mat_asym_full< TN > & r)`

Here is the call graph for this function:



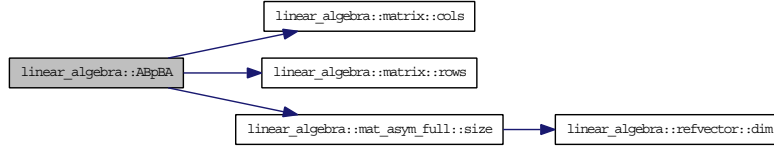
3.3.1.6 `template<class TN> mat_asym_full<TN> linear_algebra::ABmBA (const mat_asym_full< TN > & A, const mat_asym_full< TN > & B)`

Here is the call graph for this function:



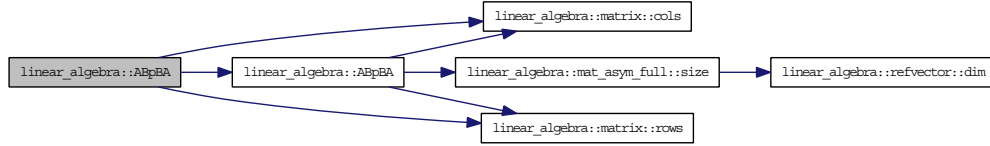
3.3.1.7 `template<class TN> mat_asym_full<TN>& linear_algebra::ABpBA (const mat_sym_full< TN > & A, const mat_asym_full< TN > & B, mat_asym_full< TN > & r)`

Here is the call graph for this function:



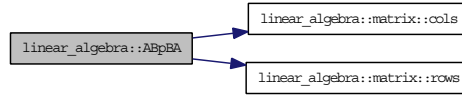
3.3.1.8 `template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_sym_full<TN> & A, const mat_asym_full<TN> & B—)`

Here is the call graph for this function:



3.3.1.9 `template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_full<TN> & A, const mat_asym_full<TN> & B, mat_asym_full<TN> & r)`

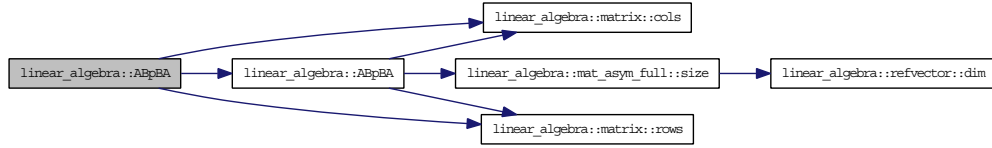
Here is the call graph for this function:



3.3.1.10 `template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_full<TN> A, const mat_asym_full<TN> B)`

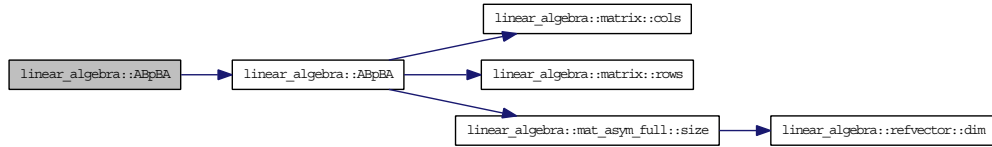
Here is the call graph for this function:





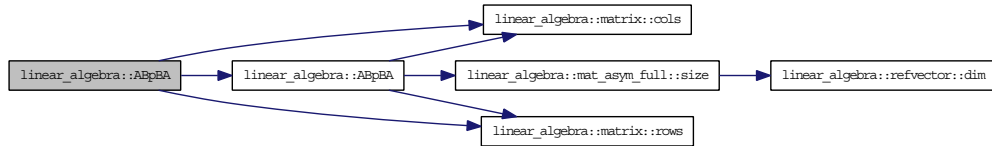
3.3.1.11 `template<class TN> mat_asym_full<TN>& linear_algebra::ABpBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B, mat_asym_full< TN > & r)`

Here is the call graph for this function:



3.3.1.12 `template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_asym_full< TN > A, const mat_sym_full< TN > B)`

Here is the call graph for this function:

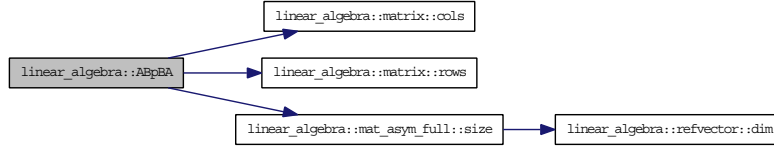


3.3.1.13 `template<class TN> mat4_4asym4_4full<TN>& linear_algebra::ABpBA (const mat_sym_full< TN > A, const mat_asym_sparse< TN > B, mat_asym_full< TN > & r)`

Todo

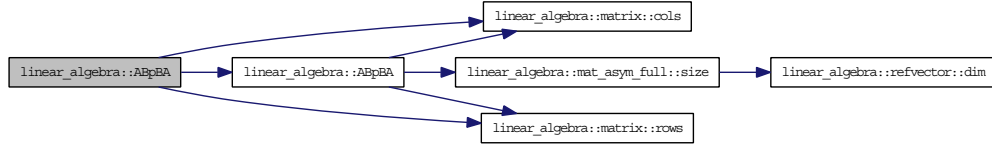
Need optimization taking advantage of sparse structure.

Here is the call graph for this function:



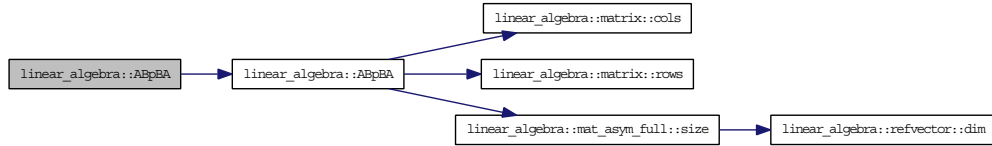
3.3.1.14 `template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_sym_full< TN > A, const mat_asym_sparse< TN > B)`

Here is the call graph for this function:



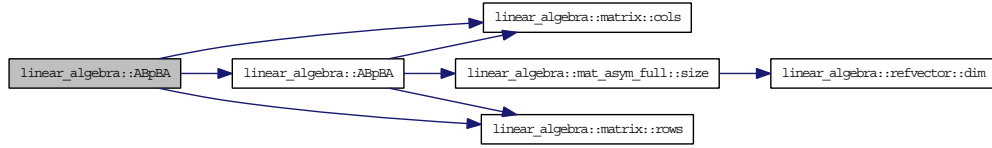
3.3.1.15 `template<class TN> mat_asym_full<TN>& linear_algebra::ABpBA (const mat_asym_sparse< TN > A, const mat_sym_full< TN > B, mat_asym_full< TN > & r)`

Here is the call graph for this function:



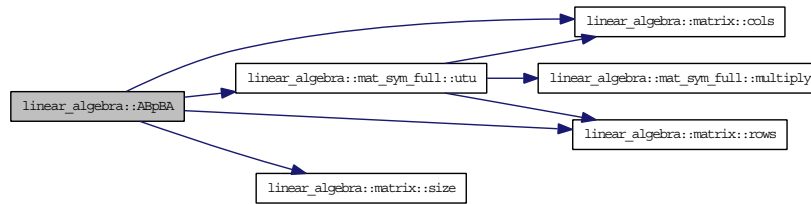
3.3.1.16 `template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_asym_sparse< TN > A, const mat_sym_full< TN > B)`

Here is the call graph for this function:



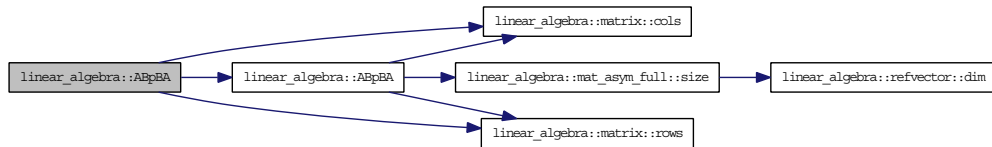
3.3.1.17 `template<class TN> mat_sym_full<TN>& linear_algebra::ABpBA (const mat_sym_full<TN> A, mat_sym_full<TN> B, mat_sym_full<TN> & r, mat_sym_full<TN> & I, mat_full<TN> & I2)`

Here is the call graph for this function:



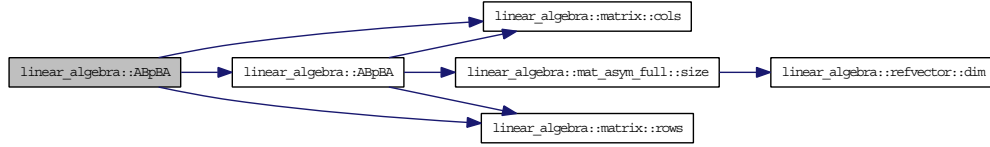
3.3.1.18 `template<class TN> mat_sym_full<TN>& linear_algebra::ABpBA (const mat_sym_full<TN> A, mat_sym_full<TN> B, mat_sym_full<TN> & r)`

Here is the call graph for this function:



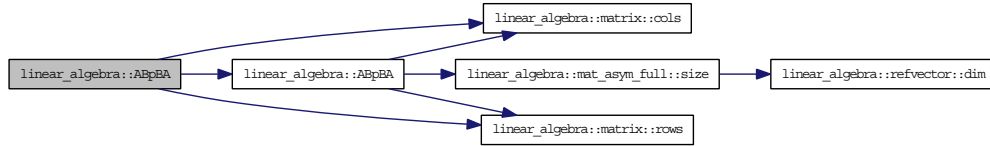
3.3.19 `template<class TN> mat_sym_full<TN>& linear_algebra::ABpBA (const mat_sym_full< TN > & A, const mat_sym_full< TN > & B, mat_sym_full< TN > & r, mat_sym_full< TN > & I)`

Here is the call graph for this function:



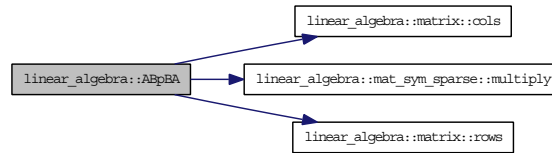
3.3.1.20 `template<class TN> mat_sym_full<TN> linear_algebra::ABpBA (const mat_sym_full< TN > & A, const mat_sym_full< TN > & B)`

Here is the call graph for this function:



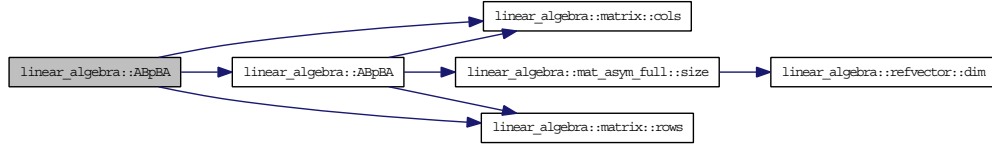
3.3.1.21 `template<class TN> mat_sym_full<TN>& linear_algebra::ABpBA (const mat_sym_full< TN > & A, const mat_sym_sparse< TN > & B, mat_sym_full< TN > & r, mat_full< TN > & I2)`

Here is the call graph for this function:



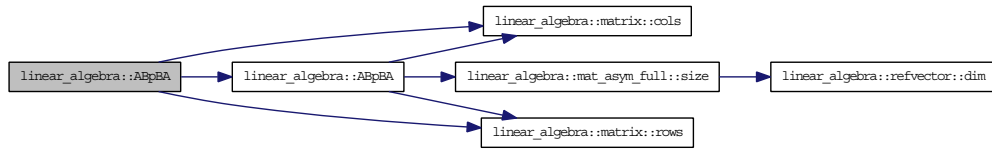
3.3.1.22 `template<class TN> mat_sym_full<TN>& linear_algebra::ABpBA (const mat_sym_full<TN> & A, const mat_sym_sparse< TN > & B, mat_sym_full< TN > & r)`

Here is the call graph for this function:



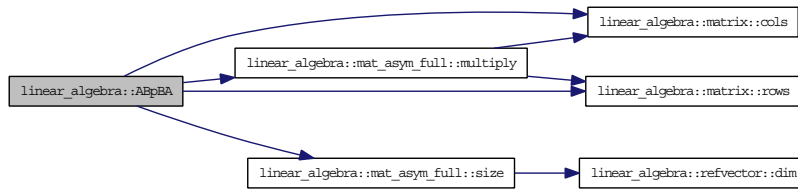
3.3.1.23 `template<class TN> mat_sym_full<TN> linear_algebra::ABpBA (const mat_sym_full<TN> & A, const mat_sym_sparse< TN > & B)`

Here is the call graph for this function:



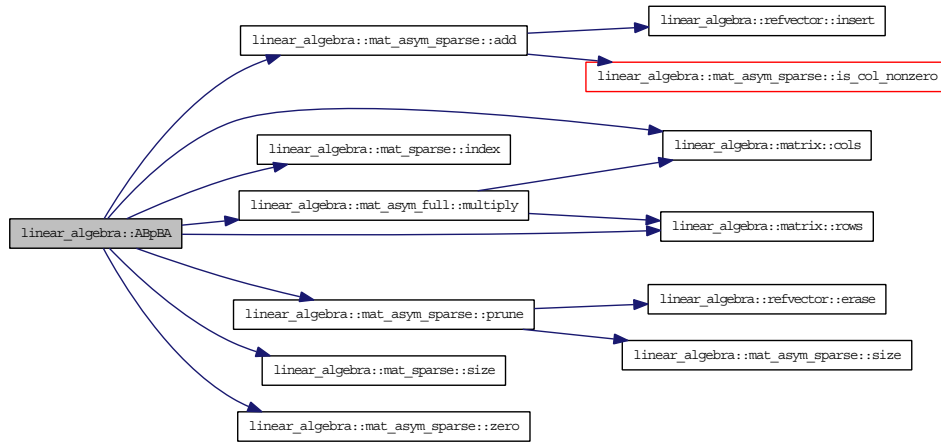
3.3.1.24 `template<class TN> mat_asym_full<TN>& linear_algebra::ABpBA (const mat_asym_full<TN> & A, const mat_sym_sparse< TN > & B, mat_asym_full< TN > & r, mat_full< TN > & I2)`

Here is the call graph for this function:



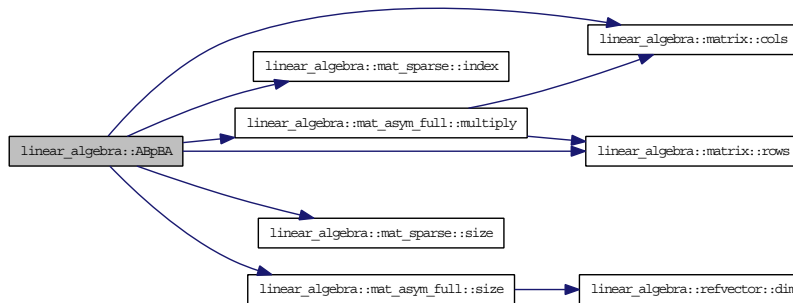
3.3.1.25 `template<class TN> mat_asym_sparse<TN>& linear_algebra::ABpBA (const mat_asym_full<TN> & A, const mat_sym_sparse< TN> & B, mat_asym_sparse< TN> & r, mat_sparse< TN> & I2)`

Here is the call graph for this function:



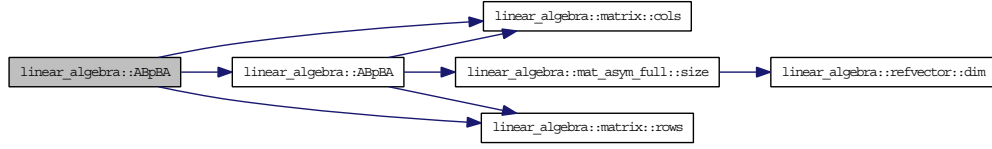
3.3.1.26 `template<class TN> mat_asym_full<TN>& linear_algebra::ABpBA (const mat_asym_full<TN> & A, const mat_sym_sparse< TN> & B, mat_asym_full< TN> & r, mat_sparse< TN> & I2)`

Here is the call graph for this function:



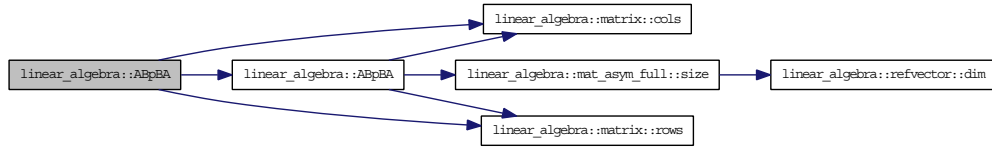
3.3.1.27 `template<class TN> mat_asym_full<TN>& linear_algebra::ABpBA (const mat_asym_full<TN> & A, const mat_sym_sparse< TN> & B, mat_asym_full< TN> & r)`

Here is the call graph for this function:



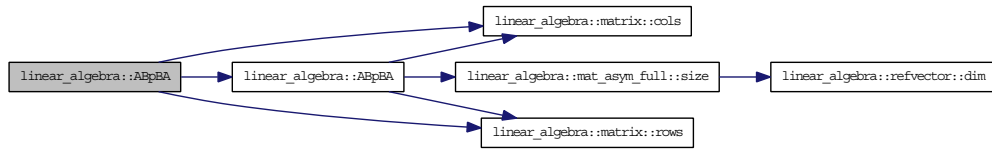
3.3.1.28 `template<class TN> mat_asym_sparse<TN>& linear_algebra::ABpBA (const mat_asym_full<TN> & A, const mat_sym_sparse< TN> & B, mat_asym_sparse< TN> & r)`

Here is the call graph for this function:



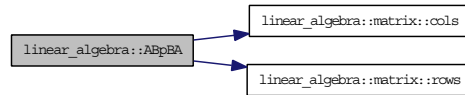
3.3.1.29 `template<class TN> mat_asym_full<TN> linear_algebra::ABpBA (const mat_asym_full<TN> & A, const mat_sym_sparse< TN> & B)`

Here is the call graph for this function:



3.3.1.30 `template<class T> mat_sym_full<T> linear_algebra::ABpBA (const mat_asym_full< T > & A, const mat_asym_full< T > & B)`

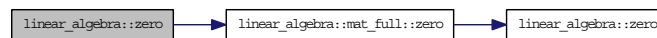
Here is the call graph for this function:



3.3.1.31 `template<class TN> matrix<TN>& linear_algebra::zero (matrix< TN > & r)[inline]`

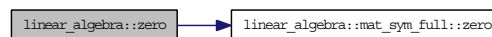
3.3.1.32 `template<class TN> mat_full< TN > & linear_algebra::zero (mat_full< TN > & r)[inline]`

Here is the call graph for this function:



3.3.1.33 `template<class TN> mat_sym_full< TN > & linear_algebra::zero (mat_sym_full< TN > & r)[inline]`

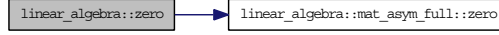
Here is the call graph for this function:



3.3.1.34 `template<class TN> mat_asym_full< TN > & linear_algebra::zero (mat_asym_full< TN > & r)[inline]`

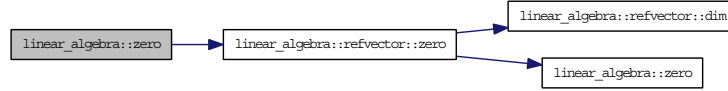
Here is the call graph for this function:





3.3.1.35 `template<class TN> refvector< TN > & linear_algebra::zero (refvector< TN > & r)[inline]`

Here is the call graph for this function:



3.3.1.36 `double & linear_algebra::zero (double & r)[inline]`

3.3.1.37 `double & linear_algebra::one (double & r)[inline]`

3.3.1.38 `template<class TN> matrix< TN > & linear_algebra::one (matrix< TN > & r)[inline]`

3.3.1.39 `void linear_algebra::copy (double & a, const double & b, const long i)[inline]`

3.3.1.40 `void linear_algebra::copy (double & a, const double & b)[inline]`

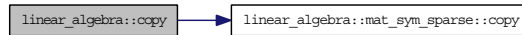
3.3.1.41 `void linear_algebra::copy (long & a, const long & b, const long i)[inline]`

3.3.1.42 `void linear_algebra::copy (long & a, const long & b)[inline]`

3.3.1.43 `template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > & a, mat_sym_sparse< TN > & b, const long i)tt [inline]`

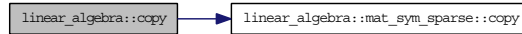
3.3.1.44 `template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > & a, const mat_sym_sparse< TN > & b)[inline]`

Here is the call graph for this function:



3.3.1.45 `template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > & a, mat_sym_sparse< TN > & b)[inline]`

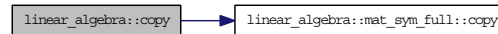
Here is the call graph for this function:



3.3.1.46 `template<class TN> void linear_algebra::copy (mat_sym_full< TN > & a, mat_sym_full< TN > & b, const long i)[inline]`

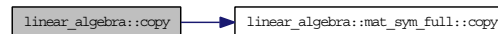
3.3.1.47 `template<class TN> void linear_algebra::copy (mat_sym_full< TN > & a, const mat_sym_full< TN > & b)[inline]`

Here is the call graph for this function:



3.3.1.48 `template<class TN> void linear_algebra::copy (mat_sym_full< TN > & a, mat_sym_full< TN > & b)[inline]`

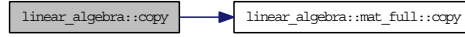
Here is the call graph for this function:



3.3.1.49 `template<class TN> void linear_algebra::copy (mat_full< TN > & a, mat_full< TN > & b, const long i)[inline]`

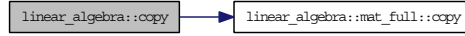
3.3.1.50 `template<class TN> void linear_algebra::copy (mat_full< TN > & a, const mat_full< TN > & b)[inline]`

Here is the call graph for this function:



3.3.1.51 `template<class TN> void linear_algebra::copy (mat_full< TN > & a, mat_full< TN > & b)[inline]`

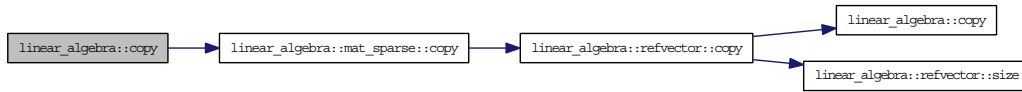
Here is the call graph for this function:



3.3.1.52 `template<class TN> void linear_algebra::copy (mat_sparse< TN > & a, mat_sparse< TN > & b, const long i)[inline]`

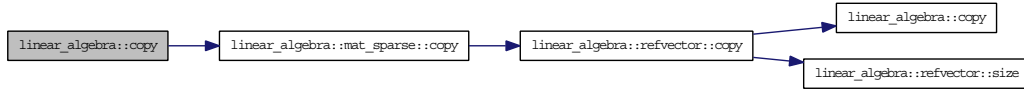
3.3.1.53 `template<class TN> void linear_algebra::copy (mat_sparse< TN > & a, const mat_sparse< TN > & b)[inline]`

Here is the call graph for this function:



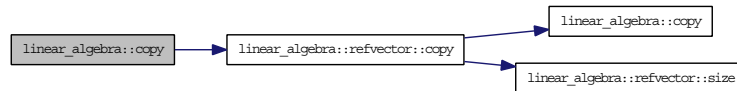
3.3.1.54 `template<class TN> void linear_algebra::copy (mat_sparse< TN > & a, mat_sparse< TN > & b)[inline]`

Here is the call graph for this function:



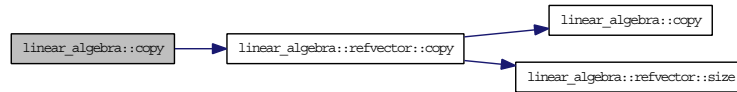
3.3.1.55 `template<class TN> void linear_algebra::copy (refvector< TN > & a, refvector< TN > & b, const long i)[inline]`

Here is the call graph for this function:



3.3.1.56 `template<class TN> void linear_algebra::copy (refvector< TN > & a, const refvector< TN > & b){[inline]`

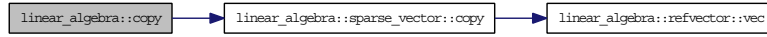
Here is the call graph for this function:



3.3.1.57 `template<class TN> void linear_algebra::copy (refvector< TN > & a, refvector< TN > & b){[inline]`

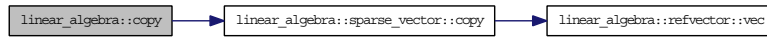
3.3.1.58 `template<class TN> void linear_algebra::copy (sparse_vector< TN > & a, sparse_vector< TN > & b, const long i)[inline]`

Here is the call graph for this function:



3.3.1.59 `template<class TN> void linear_algebra::copy (sparse_vector< TN > & a, const sparse_vector< TN > & b)[inline]`

Here is the call graph for this function:

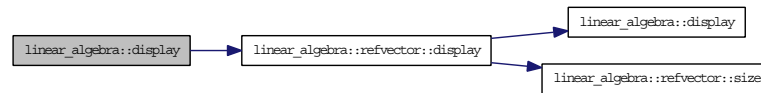


3.3.1.60 `template<class C> void linear_algebra::copy (C & a, const C & b)[inline]`

3.3.1.61 `template<class TN> void linear_algebra::display (const matrix< TN > & a, ostream & outs)[inline]`

3.3.1.62 `template<class TN> void linear_algebra::display (const refvector< TN > & a, ostream & outs)[inline]`

Here is the call graph for this function:



3.3.1.63 `void linear_algebra::display (const double & a, ostream & outs)[inline]`

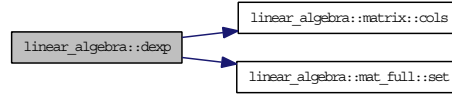
3.3.1.64 `template<class TNTN> mat_full< TNTN > linear_algebra::exp (const mat_asym_full< TNTN > & A)`

Here is the call graph for this function:



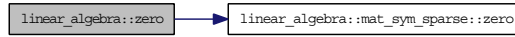
3.3.1.65 `template<class TNTN> mat_full< TNTN > linear_algebra::dexp (mat_asym_full< TNTN > A, const long k, const long l)`

Here is the call graph for this function:



3.3.1.66 `template<class TN> mat_sym_sparse<TN>& linear_algebra::zero (mat_sym_sparse< TN > & r)[inline]`

Here is the call graph for this function:



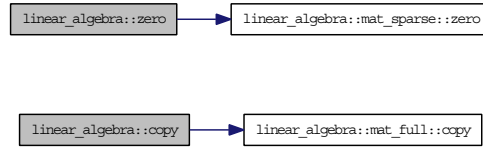
3.3.1.67 `template<class TN> mat_sparse<TN>& linear_algebra::zero (mat_sparse< TN > & r)[inline]`

3.3.1.68 `template<typename TN> TN& linear_algebra::zero (TN & r)[inline]`

3.3.1.69 `void linear_algebra::copy (bool & a, const bool & b)[inline]`

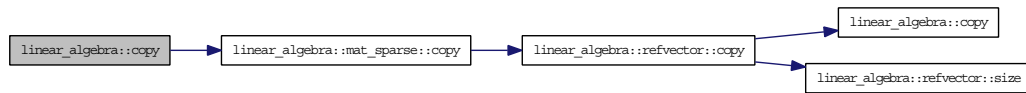
3.3.1.70 `template<class TN> void linear_algebra::copy (mat_full< TN > & a, const mat_full< TN > & b, const long i){[inline]`

Here is the call graph for this function:



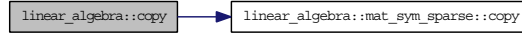
3.3.1.71 `template<class TN> void linear_algebra::copy (mat_sparse< TN > & a, const mat_sparse< TN > & b, const long i)[inline]`

Here is the call graph for this function:



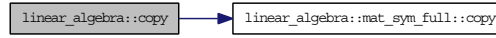
3.3.1.72 `template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > & a, const mat_sym_sparse< TN > & b, const long i)[inline]`

Here is the call graph for this function:



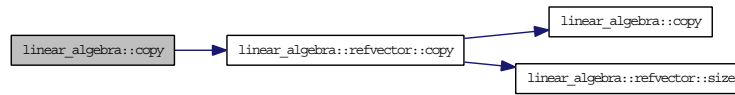
3.3.1.73 `template<class TN> void linear_algebra::copy (mat_sym_full< TN > & a, const mat_sym_full< TN > & b, const long i)[inline]`

Here is the call graph for this function:



3.3.1.74 `template<class TN> void linear_algebra::copy (refvector< TN > & a, const refvector< TN > & b, const long i)[inline]`

Here is the call graph for this function:



3.3.1.75 `template<class TN> TN linear_algebra::operator * (const std::vector< TN > & a, const std::vector< TN > & b)`

3.3.1.76 `template<class TN> TN linear_algebra::operator * (const vector< TN > & a, const vector< TN > & b)`

3.3.1.77 `template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, std::vector< TN > b)[inline]`

3.3.1.78 `template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, refvector< TN > b)[inline]`

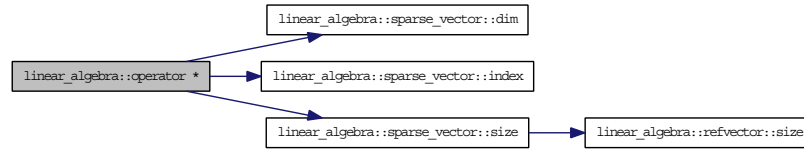
Here is the call graph for this function:





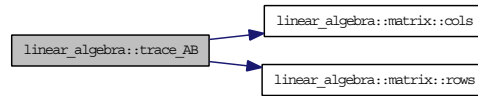
3.3.1.79 `template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, vector< TN > b)[inline]`

Here is the call graph for this function:



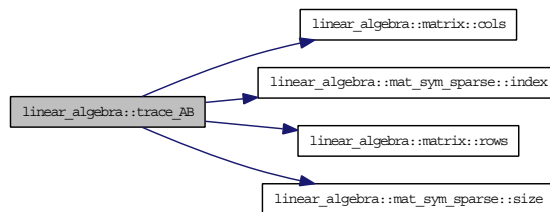
3.3.1.80 `template<class TN> TN linear_algebra::trace_AB (const mat_sym_full< TN > & A, const mat_sym_full< TN > & B)`

Here is the call graph for this function:



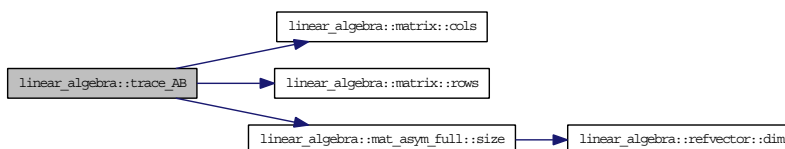
3.3.1.81 `template<class TN> TN linear_algebra::trace_AB (const mat_sym_full< TN > & A, const mat_sym_sparse< TN > & B)`

Here is the call graph for this function:



3.3.1.82 `template<class T> T linear_algebra::trace_AB (const mat_asym_full< T > & A, const mat_asym_full< T > & B)`

Here is the call graph for this function:



## 3.4 std Namespace Reference

### Functions

- `template<class TN> void (const vector< TN > &a, ostream &OUT)`
- `template<typename TN> void (const TN &a, ostream &OUT)`

---

## 4. Discrete Optimization in Chemical Space Class Documentation

---

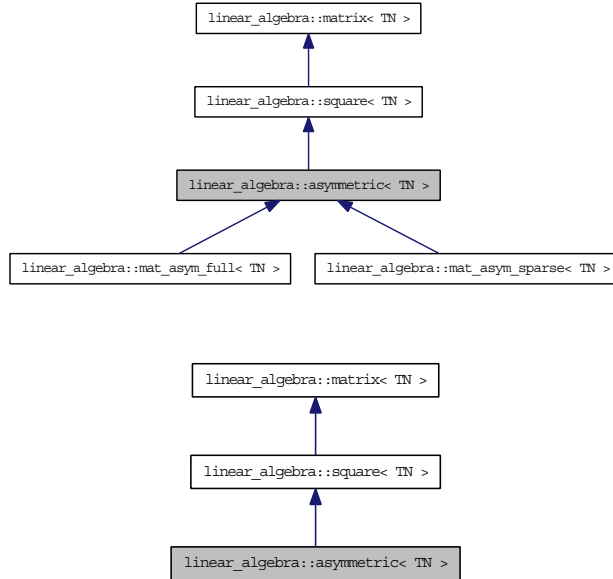
### 4.1 linear\_algebra::asymmetric< TN > Class Template Reference

Class of antisymmetric matrices.

Inheritance diagram for `linear_algebra::asymmetric< TN >`:

Collaboration diagram for `linear_algebra::asymmetric< TN >`:

### Public Member Functions



- `asymmetric< TN > & transpose ()`
- `asymmetric< TN > operator * (const TN a) const`

**template<class TN> class linear\_algebra::asymmetric< TN >**

#### 4.1.1 Member Function Documentation

4.1.1.1 `template<class TN> asymmetric< TN > linear_algebra::asymmetric< TN >::operator * (const TN a) const`

Reimplemented in `linear_algebra::mat_asym_full< TN >`.

4.1.1.2 `template<class TN> asymmetric< TN > & linear_algebra::asymmetric< TN >::transpose ()`

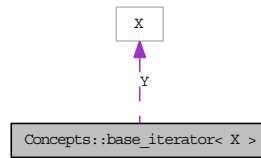
The documentation for this class was generated from the following files:

- `include/BCR_CPP_LA/asymmetric.decl`
- `include/BCR_CPP_LA/asymmetric.h`

## 4.2 Concepts::base\_iterator< X > Class Template Reference

Concept defines an iterator for a class of bases.

`#include <concepts.hh>` Collaboration diagram for `Concepts::base_iterator< X >`:



## Public Member Functions

- `BOOST_CONCEPT_USAGE` (`base_iterator`)

## Private Attributes

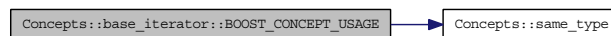
- `X & Y`

```
template<class X> class Concepts::base_iterator< X >
```

### 4.2.1 Member Function Documentation

4.2.1.1 `template<class X> Concepts::base_iterator< X >::BOOST_CONCEPT_USAGE  
base_iterator< X >`  
[inline]

Here is the call graph for this function:



### 4.2.2 Member Data Documentation

4.2.2.1 `template<class X> X& Concepts::base_iterator< X >::Y[private]`

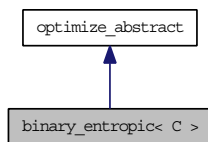
The documentation for this class was generated from the following file:

- `concepts.hh`

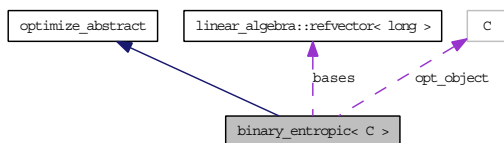
### 4.3 `binary_entropic< C >` Class Template Reference

Class for enhanced sampling using an entropic measure of coverage.

`#include <binary_entropic.hh>` Inheritance diagram for `binary_entropic< C >`:



Collaboration diagram for `binary_entropic< C >`:



#### Public Member Functions

- `binary_entropic (const C &a)`  
*Copy constructor.*
- `ulong optimize (ulong N) const`  
*Meta-Optimize by generating maximally distant starting configurations.*
- `valerg get_value (const ulong i) const`

#### Public Attributes

- `ulong nruns`  
*Number of runs.*
- `long max_steps`  
*Maximum number of steps.*

#### Protected Attributes

- `C opt_object`  
*This is an optimization object of type C. This is the underlying optimization method.*

- `refvector< long > bases`

*This vector defines the substitution bases used in the optimization and entropic maximization (`maximize_entropic_distance()`).*

## Private Member Functions

- `BOOST_CONCEPT_ASSERT ((Concepts::has_optimize< C >))`  
*Require C to conform to an optimization.*
- `BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))`  
*Require C to conform to the pruner concept.*

### 4.3.1 Detailed Description

`template<class C> class binary_entropic< C >`

This class provides a meta-optimization mechanism. The secondary optimizer C has to comply with the `Concepts::has_optimize` and `Concepts::pruner` concepts in order for the optimization to work. After each successful optimization the visited configurations are analysed for coverage of the library. `maximize_entropic_distance()` is then used to determine the next best starting point.

### 4.3.2 Constructor & Destructor Documentation

4.3.2.1 `template<class C> binary_entropic< C >::binary_entropic (const C & a)[inline]`

### 4.3.3 Member Function Documentation

4.3.3.1 `template<class C> binary_entropic< C >::BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >)) [private]`

4.3.3.2 `template<class C> binary_entropic< C >::BOOST_CONCEPT_ASSERT ((Concepts::has_optimize< C >)) [private]`

4.3.3.3 `template<class C> valerg binary_entropic< C >::get_value (const ulong i) const [inline]`

4.3.3.4 `template<class C> ulong binary_entropic< C >::optimize (ulong N) const [inline, virtual]`

The metric used to determine the distance is implemented and explained in `maximize_entropic_distance()`.

Implements `optimize_abstract`.

Here is the call graph for this function:



#### 4.3.4 Member Data Documentation

4.3.4.1 `template<class C> refvector<long> binary_entropic< C >::bases[protected]`

4.3.4.2 `template<class C> long binary_entropic< C >::max_steps`

4.3.4.3 `template<class C>ulong binary_entropic< C >::nruns`

4.3.4.4 `template<class C> C binary_entropic< C >::opt_object[protected]`

The documentation for this class was generated from the following file:

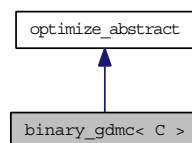
- `binary_entropic.hh`

#### 4.4 `binary_gdmc< C >` Class Template Reference

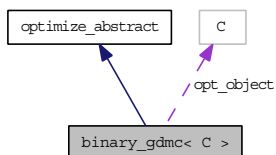
Gradient-directed Monte Carlo performed on a hypercube.

```
#include <binarygdmc.hh>
```

Inheritance diagram for `binary_gdmc< C >`:



Collaboration diagram for `binary_gdmc< C >`:



## Public Member Functions

- `binary_gdmc (const C &a)`  
*Copy Constructor.*
- `ulong optimize (ulong N) const`  
*Gradient-directed Monte-Carlo optimization.*
- `valerg get_value (const ulong i) const`  
*Return the value for molecule i.*

## Public Attributes

- `double T`  
*Temperature-analogue to be used for Monte-Carlo evaluations.*
- `ulong tight_steps`  
*Number of tightening steps. Determines the strictness of acceptance.*
- `ulong max_steps`  
*Maximum number of steps.*

## Private Member Functions

- `BOOST_CONCEPT_ASSERT ((Concepts::has_gradients< C >))`  
*Require C to conform to the Concepts::has\_gradients concept.*
- `BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))`  
*Require C to conform to the Concepts::pruner concept.*
- `BOOST_CONCEPT_ASSERT ((Concepts::has_optimize< C >))`  
*Require C to conform to an optimization.*



## Private Attributes

- C opt\_object

*This is the library object to be used in the optimization.*

### 4.4.1 Detailed Description

**template<class C> class binary\_gdmc< C >** This class takes a Library object and performs an optimization. The Library is arranged in a hypercube topology in which each molecule occupies a vertex on the hull of the cube. After a local optimum has been reached, a Monte-Carlo simulation is performed to determine a new starting point. Each time a new minimum is found the acceptance criteria become more strict.

### 4.4.2 Constructor & Destructor Documentation

4.4.2.1 `template<class C> binary_gdmc< C >::binary_gdmc (const C & a)[inline]`

### 4.4.3 Member Function Documentation

4.4.3.1 `template<class C> binary_gdmc< C >::BOOST_CONCEPT_ASSERT  
((Concepts::has_optimize< C >))[private]`

4.4.3.2 `template<class C> binary_gdmc< C >::BOOST_CONCEPT_ASSERT  
((Concepts::pruner< C >))[private]`

4.4.3.3 `template<class C> binary_gdmc< C >::BOOST_CONCEPT_ASSERT  
((Concepts::has_gradients< C >))[private]`

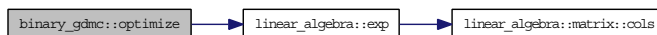
4.4.3.4 `template<class C> valerg binary_gdmc< C >::get_value (const ulong i) const[inline]`

4.4.3.5 `template<class C> ulong binary_gdmc< C >::optimize (ulong N) const[inline,  
virtual]`

The gradient to be used is determined by the `gradient()` method of the class C.

Implements `optimize_abstract`.

Here is the call graph for this function:



#### 4.4.4 Member Data Documentation

4.4.4.1 `template<class C> ulong binary_gdmc< C >::max_steps`

4.4.4.2 `template<class C> C binary_gdmc< C >::opt_object[private]`

4.4.4.3 `template<class C> double binary_gdmc< C >::T`

4.4.4.4 `template<class C> ulong binary_gdmc< C >::tight_steps`

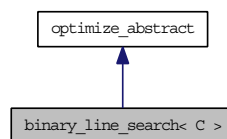
The documentation for this class was generated from the following file:

- `binarygdmc.hh`

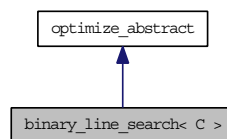
#### 4.5 `binary_line_search< C >` Class Template Reference

Search a pruned Library using a linesearch algorithm.

`#include <binary_line_search.hh>` Inheritance diagram for `binary_line_search< C >`:



Collaboration diagram for `binary_line_search< C >`:



#### Public Member Functions

- `binary_line_search ()`  
*Default constructor.*
- `binary_line_search (const binary_line_search &a)`  
*Copy constructor.*

- `ulong optimize (ulong N) const`  
*Default optimization with a clear history and no pruning in the first iteration.*
- `ulong optimize (ulong N, double lambda, refvector< ulong > &visited_run) const`  
*Optimize starting from conformation N.*

## Private Member Functions

- `BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))`  
*Require Concepts::pruner compliance of C.*
- `BOOST_CONCEPT_ASSERT ((boost::DefaultConstructible< C >))`

### 4.5.1 Detailed Description

`template<class C> class binary_line_search< C >`

In this method the Library is spanned via a hypercube. Each molecule in the library sits on one vertex of the cube. This leads to a bit representation of each molecule. The line search is then performed on the bits.

## Constructor & Destructor Documentation

4.5.1.1 `template<class C> binary_line_search< C >::binary_line_search ()`[inline]

4.5.1.2 `template<class C> binary_line_search< C >::binary_line_search (const binary_line_search< C > & a)`

### 4.5.2 Member Function Documentation

4.5.2.1 `template<class C> binary_line_search< C >::BOOST_CONCEPT_ASSERT ((boost::DefaultConstructible< C >))`[private]

4.5.2.2 `template<class C> binary_line_search< C >::BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))`[private]

4.5.2.3 `template<class C> ulong binary_line_search< C >::optimize (ulong N, double lambda, refvector< ulong > & visited_run) const`[inline]

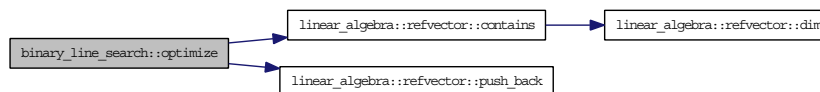
A Lagrange multiplier method is employed to enforce boundary constraints. The multiplier increases as the optimization proceeds, enforcing the constraint ever more rigorously.

In essence we are following an interior point method, in which we reestablish feasibility every  $\log(\text{library size})$  cycles. The precise rules for pruning and adjusting the Lagrange multiplier can be found in `prune()`. At each position in the bit string representing the reference molecule  $N$ , the molecule with that bit flipped from  $N$  is computed and compared. If there is an improvement in the objective function  $P + \lambda\pi$ , where  $P$  is the property,  $\pi$  is the penalty and  $\lambda$  is the current value of the Lagrange multiplier, then the new molecule becomes the new reference.

**See also:**

`noprune::adjust_lagrange()`, `simple_prune::prune()`,  
`reorder_general_base::prune()`

Here is the call graph for this function:



4.5.2.4 `template<class C> ulong binary_line_search< C >::optimize (ulong  $N$ ) const[inline, virtual]`

This is a clean slate optimization. Any pruned molecules from previous runs are eradicated.

**Parameters:**

$N$  gives the starting molecule's number in the library.

Implements `optimize_abstract`.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

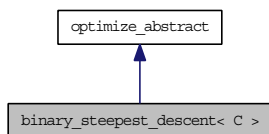
- `binary_line_search.hh`

## 4.6 `binary_steepest_descent< C >` Class Template Reference

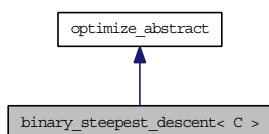
Searches a Library for an optimum using a steepest descent method.

```
#include <binarysteepestdescent.hh>
```

Inheritance diagram for `binary_steepest_descent< C >`:



Collaboration diagram for `binary_steepest_descent< C >`:



## Public Member Functions

- `ulong optimize (ulong N) const`  
*Optimize starting from index N.*

## Private Member Functions

- `BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))`  
*Require C to conform to the Concepts::pruner concept.*
- `BOOST_CONCEPT_ASSERT ((Concepts::has_gradients< C >))`  
*Require C to conform to the Concepts::has\_gradients concept.*

### 4.6.1 Detailed Description

```
template<class C> class binary_steepest_descent< C >
```

As all optimization classes in the `binary_*` vein, the library is organized and searched as a hypercube on which the molecules are the vertices on the hull. The resultant bit-string representation of each molecule is used to perform optimizations.

## 4.6.2 Member Function Documentation

4.6.2.1 `template<class C> binary_steepest_descent< C >::BOOST_CONCEPT_ASSERT  
((Concepts::has_gradients< C >)) [private]`

4.6.2.2 `template<class C> binary_steepest_descent< C >::BOOST_CONCEPT_ASSERT  
((Concepts::pruner< C >))[private]`

4.6.2.3 `template<class C> ulong binary_steepest_descent< C >::optimize (ulong N)  
const[inline, virtual]`

Each edge (i.e., bit) on the hypercube is interpreted as a search direction. A gradient is computed along each direction and is then searched for improvements. Constraints are enforced with Lagrange multipliers which are ramped after a local optimum is reached.

Implements `optimize_abstract`.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

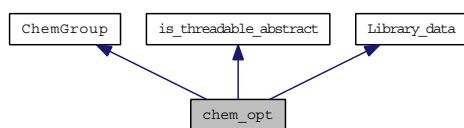
- `binarysteepestdescent.hh`

## 4.7 chem\_opt Class Reference

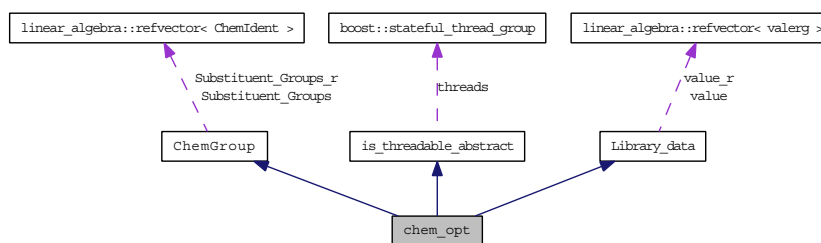
Chemical optimization class.

`#include <chem_opt.hh>`

Inheritance diagram for `chem_opt`:



Collaboration diagram for `chem_opt`:



## Public Member Functions

- `chem_opt ()`  
*Default constructor.*
- `chem_opt (const chem_opt &a)`  
*Default constructor.*
- `chem_opt (const ChemGroup &a)`  
*Construct from a ChemGroup.*
- `chem_opt (ulong nmax)`  
*Construction assigning maximum number of threads.*

- `chem_opt & operator= (const ChemGroup &a)`  
*Assignment operator.*
- `void output () const`  
*Output of the chemical pattern.*
- `valerg compute_property (ulong i) const`  
*Compute the property.*
- `ulong get_space_size () const`  
*Compute the size of the optimization space.*
- `ulong get_space_size (const long Group) const`  
*Compute the size of the optimization space of a specific group.*
- `ulong get_bits () const`  
*Compute the number of bits to address the optimization space.*

#### 4.7.1 Detailed Description

The class implements a Library of molecules that can be enumerated.

#### 4.7.2 Constructor & Destructor Documentation

4.7.2.1 `chem_opt::chem_opt ()`

4.7.2.2 `chem_opt::chem_opt (const chem_opt & a)`

4.7.2.3 `chem_opt::chem_opt (const ChemGroup & a)`

4.7.2.4 `chem_opt::chem_opt (ulong nmax)`

#### 4.7.3 Member Function Documentation

4.7.3.1 `valerg chem_opt::compute_property (ulong i) const[virtual]`

Computes the Z-matrix of molecule *i* and then performs a conformational search and finally computes and returns the computed property value (as well as constraint violations).

Implements `Library_data`.

4.7.3.2 `ulong chem_opt::get_bits () const[virtual]`

Implements `Library_data`.



4.7.3.3 `ulong chem_opt::get_space_size (const long Group) const`

4.7.3.4 `ulong chem_opt::get_space_size () const[virtual]`

Implements `Library_data`.

4.7.3.5 `chem_opt & chem_opt::operator= (const ChemGroup & a)`

This should really never be invoked on anything that has already been started.

4.7.3.6 `void chem_opt::output () const`

Reimplemented from `ChemGroup`. The documentation for this class was generated from the following files:

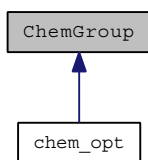
- `chem_opt.hh`
- `chem_opt.cc`

## 4.8 ChemGroup Class Reference

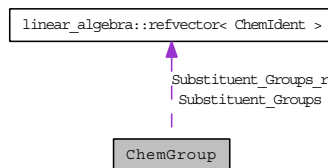
This class describes a group of substitution sites and their respective substitution options as well as computation history.

```
#include <chemgroup.hh>
```

Inheritance diagram for `ChemGroup`:



Collaboration diagram for `ChemGroup`:



## Public Member Functions

- `bool error_free () const`  
*Check whether there are any errors in the definition.*
- `mat build_zmat (long i, ulong &number, const zmat_connector &e, zmat &A, zmat_connector &y) const`  
*Build the Z-matrix using occupation i.*
- `zmat & build_zmat (long i, const zmat_connector &e, zmat &A, zmat_connector &y) const`  
*Build the Z-matrix using occupation i.*
- `void occupy (ulong number) const`  
*Set occupations according to number.*
- `ChemGroup & operator= (const ChemGroup &a)`  
*Assignment operator.*
- `bool operator== (const ChemGroup &a) const`  
*Comparison operator.*
- `void output () const`  
*Display to cout what's going on.*

## Constructors

- `ChemGroup ()`  
*Empty constructor.*
- `ChemGroup (const ChemGroup &a)`  
*Copy constructor.*
- `ChemGroup (stringstream &in)`  
*Construction from a stringstream.*
- `ChemGroup (istream &in)`  
*Construction from a file.*

## Substituent manipulations

- `long add_substituent (const ChemIdent &a)`  
*Add a substituent.*

- `refvector< long > add_substituents (const refvector< ChemIdent > &a)`  
*Add a list of substituents.*
- `void add_substituent (long i, long j, long k)`  
*Add a substituent to a site.*
- `void add_substituents (long i, long k, const refvector< long > &j)`  
*Add a list of substituents to a site.*

## Public Attributes

- `const refvector< ChemIdent > & Substituent_Groups_r`

## Private Attributes

- `refvector< ChemIdent > Substituent_Groups`

### 4.8.1 Detailed Description

It assumes the existence of a `valerg calc_property(const zmat& A, const stringstream& id)` function to compute properties. In essence this is a collection of `ChemIdent` objects. The first substitution group is used as the root for generating any Z-matrices based on the specific chosen substitutions.

The general strategy is to split each substitution group (`ChemIdent`) from the overall structure. Hence, a `ChemIdent` identifies a molecular framework/scaffold and possible substitution sites on this. Singletons don't have any substitution sites and cap substitutions. The `ChemGroup` collects these and gives meaning to specific substitutions at each defined site.

See class `ChemIdent` for the substitution rules. **Examples:**

`carbazoles.inp`, and `vanilla-rings.inp`.

### 4.8.2 Constructor & Destructor Documentation

4.8.2.1 `ChemGroup::ChemGroup ()`

4.8.2.2 `ChemGroup::ChemGroup (const ChemGroup & a)`

4.8.2.3 `ChemGroup::ChemGroup (stringstream & s)`

**See also:**

`ChemGroup::ChemGroup(istream& in)`

4.8.2.4 `ChemGroup::ChemGroup (istream & in)`

The input file format is as follows:`ChemGroup(ChemIdent1 ChemIdent2 ... )`

**See also:**

`ChemIdent::ChemIdent(istream& in)` for `ChemIdent` input format and examples.

### 4.8.3 Member Function Documentation

4.8.3.1 `void ChemGroup::add_substituent (long i, long j, long k)`

A new substituent is added at site *j* of group *i*. *k* references the global group that is taken from `ChemGroup` and linked to the substitution site. Hence, *j* has to be a valid site on `Substituent_Groups[i]` and *k* as well as *i* are an index of `Substituent_Groups`.

4.8.3.2 `long ChemGroup::add_substituent (const ChemIdent & a)`

This appends another `ChemIdent` object to the list of possible substitution groups.

4.8.3.3 `void ChemGroup::add_substituents (long i, long m, const refvecto< long > & j)`

This adds a number of substituents to site *m* on group *i*. Similar to `add_substituent()`.

4.8.3.4 `refvector< long > ChemGroup::add_substituents (const refvector< ChemIdent > & a)`

This appends a list of substituents as `add_substituent()` does for a single group.

4.8.3.5 `zmat & ChemGroup::build_zmat (long Group, const zmat_connector & e, zmat & A, zmat_connector & y) const`

This routine generates the local Z-matrix and combines it with the global Z-matrix using the default values as provided by occupation. Due to connectivity concerns it may be necessary to define appropriate dummy atoms in `zmat::Z`. A suggestion would be a dummy atom for each substitution group to define the topology.

#### Parameters:

*Group* index of the group to be added to the Z-matrix

*e* defines the connection of *Group* to the Z-matrix

*A* is the Z-matrix to which the *Group* topology will be attached. It will also be returned.

Note the non-const reference.

*y* defines the return connector.

4.8.3.6 `zmat & ChemGroup::build_zmat (long Group, ulong & number, const zmat_connector & e, zmat & A, zmat_connector & y) const`

This routine generates the local Z-matrix and combines it with the global Z-matrix for molecule number. Due to connectivity concerns it may be necessary to define appropriate dummy atoms in `zmat::Z`. A suggestion would be a dummy atom for each substitution group to define the topology. The combinatorics are different in this version, as it frees all constraints imposed by the general structure.

#### Parameters:

*Group* index of the group to be added to the Z-matrix

*number* index of the molecule to be built.

*A* is the Z-matrix to which the Group topology will be attached. It will also be returned.

Note the non-const reference.

*y* defines the return connector.

4.8.3.7 `bool ChemGroup::error_free () const`

4.8.3.8 `void ChemGroup::occupy (ulong number) const`

Each `ChemIdent` has default occupations for its sites. This sets these to reflect the molecule referenced by *number*.

4.8.3.9 `ChemGroup & ChemGroup::operator= (const ChemGroup & a)`

Assignment operator.

Reimplemented in `chem_opt`.

4.8.3.10 `bool ChemGroup::operator== (const ChemGroup & a) const`

4.8.3.11 `void ChemGroup::output () const`

Reimplemented in `chem_opt`.

## 4.8.4 Member Data Documentation

4.8.4.1 `refvector<ChemIdent> ChemGroup::Substituent_Groups [private]`

This array holds the possible `ChemIdent` substitutions for each substitution site.

#### 4.8.4.2 const refvector<ChemIdent>& ChemGroup::Substituent\_Groups\_r

This double array holds the possible substitutions for each substitution site. (READ ONLY)

The documentation for this class was generated from the following files:

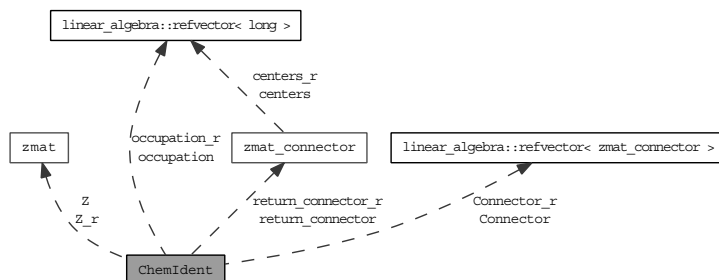
- chemgroup.h
- chemgroup.cc

## 4.9 ChemIdent Class Reference

This class describes a group of substitution sites on a Z-matrix and their respective substitution options.

```
#include <chemident.hh>
```

Collaboration diagram for ChemIdent:



## Public Member Functions

- refvector< int > & generate\_conformation\_list (long i, const zmat\_connector &e, refvector< int > &conformation, zmat\_connector &y) const  
*Build a list of indices for occupied substituents, configuration only. (DEPRECATED).*
- ChemIdent & operator= (const ChemIdent &a)  
*Assignment operator.*
- bool operator== (const ChemIdent &a) const  
*Comparison operator.*

- ChemIdent & set\_Z (const zmat &Z)  
*Set the Z-matrix.*
- ChemIdent & set\_return\_connector (const zmat\_connector &A)  
*Set the return connector.*
- ChemIdent & add\_to\_dihedrals (int zentry, int maxn, int n)  
*Add another dihedral for conformational searching.*
- ChemIdent & set\_dihedrals (int dihedrals, int n)  
*Set the value of a dihedral.*
- void occupy (long i, long j) const  
*Set occupation to generate the zmat.*
- long compute\_space\_size () const  
*Computes the size of the library of molecules described by this instance.*
- void fix\_Substituent\_dihedrals ()  
*Fix the substituent dihedrals for connection purposes.*
- void output () const  
*Display to cout what's going on.*

## Constructors

- ChemIdent ()  
*Empty constructor.*
- ChemIdent (const string &Name)  
*Primary constructor.*
- ChemIdent (const zmat &A)  
*Primary constructor from a Z-matrix.*
- ChemIdent (const ChemIdent &a)  
*Copy constructor.*
- ChemIdent (stringstream &in)  
*Construction from a stringstream.*
- ChemIden (istream &in)  
*Construction from a file.*

## Site manipulations

- ChemIdent & add\_substitution\_site (const refvector< long > &a, const zmat\_connector &e)  
*Increase the number of substitution sites and add a list of substituents for that site.*
- ChemIdent & add\_substitution\_site (long a, const zmat\_connector &e)  
*Increase the number of substitution sites and add a substituent number to that site.*

## Substituent manipulations

- ChemIdent & add\_substituents (long i, const refvector< long > &a)  
*Add a substituent.*
- ChemIdent & add\_substituent (long i, long j)  
*Add a substituent.*

## Static Public Member Functions

- static zmat\_entry & update\_connector (const zmat\_connector &a, const zmat\_connector &e, const long A, zmat\_connector &x)  
*Update a connector to fit, e.g., after combining two Z-matrices.*

## Public Attributes

- const refvector< refvector< long > > & allowed\_Substituents\_r
- const refvector< long > & occupation\_r
- const zma & Z\_
- const refvector< zmat\_connector > & Connector\_r
- const zmat\_connector & return\_connector\_r

## Private Attributes



- `refvector< refvector< long > > allowed_`
- `refvector< long > occupation`
- `long Space_Size`
- `zmat Z`
- `refvector< zmat_connector > Connector`
- `zmat_connector return_connector`

#### 4.9.1 Detailed Description

The representation of a chemical substitution pattern is subdivided into a list of substitution sites which are potentially linked (`return_connector`). In the case of linkage the groups are considered to be linked in a chain, i.e., only to the predecessor. The `return_connector` has three distinct references:

- `-6` to `-4` refer to the original connector of the following group. (default)
- `-3` to `-1` refer to values of the connector passed down to the current group. Values greater than `-1` refer to values in the local Z-matrix.

For the purpose of preoptimizing starting structures, which in the construction routine may be overlapped, the `zmat_connector::opt_val` and `zmat_entry::opt_val` structures direct the optimization of a specific entry. By default all dihedral angles between adjoined Z-matrices are optimized. This may cause problems in ring-structures. To change this the `fix_Substituent_dihedrals()` call cancels this behavior. Additionally it is possible to select dihedrals for conformational searching via `ChemIdent::dihedrals`. `zmat_connector` supplies `zmat_connector::angle` and `zmat_connector::angle_val` for conformational optimizations with respect to the connectors.

#### 4.9.2 Constructor & Destructor Documentation

4.9.2.1 `ChemIdent::ChemIdent ()`

4.9.2.2 `ChIdent::ChIdent (const string & Name)`

4.9.2.3 `ChemIdent::ChemIdent (const zmat & A)`

4.9.2.4 `ChemIdent::ChemIdent (const ChemIdent & a)`

#### 4.9.2.5 ChemIdent::ChemIdent (stringstream & s)

The input file format is as follows: ( Z(zmat)

ReturnConnector (zmat\_connector)

Connector ( zmat\_connector zmat\_connector ... )

allowed\_Substituents( ( group number 1, group number 2 , ... ) ... ) )

Under Z a zmat object is read in. The appropriate format can be found under zmat::zmat(stringstream& s). ReturnConnector reads in a zmat\_connector using zmat\_connector::zmat\_connector(stringstream& s). Connector reads in a list of zmat\_connector objects, each adhering to the format found under zmat\_connector::zmat\_connector(stringstream& s).

**See also:**

carbazoles.inp and vanilla-rings.inp in the examples section.

Read the connector.

Read the connector.

#### 4.9.2.6 ChemIdent::ChemIdent (istream & in)

**See also:**

ChemIdent::ChemIdent(stringstream& in)

### 4.9.3 Member Function Documentation

#### 4.9.3.1 ChemIdent & ChemIdent::add\_substituent (long i, long j)

#### 4.9.3.2 ChemIdent & ChemIdent::add\_substituents (long i, const refvector< long > & a)

#### 4.9.3.3 ChemIdent & ChemIdent::add\_substitution\_site (long a, const zmat\_connector & e)

#### 4.9.3.4 ChemIdent & ChemIdent::add\_substitution\_site (const refvector< long > & a, const zmat\_connector & e)

#### 4.9.3.5 ChemIdent & ChemIdent::add\_to\_dihedrals (int zentry, int maxn, int n)

**Parameters:**

*zentry* dihedral of Z-matrix entry to be optimized.

*maxn* is the number of division of 360 deg,  $\Delta\angle : \frac{360 \text{ deg}}{\text{maxn}}$ .  
*n* current instantiation (<maxn).

4.9.3.6 long ChemIdent::compute\_space\_size () const

4.9.3.7 void ChemIdent::fix\_Substituent\_dihedrals ()

4.9.3.8 refvector<int>& ChemIdent::generate\_conformation\_list (long *i*, const  
zmat\_connector & *e*, refvector< int > & *conformation*, zmat\_connector & *y*) const

4.9.3.9 void ChemIdent::occupy (long *i*, long *j*) const

4.9.3.10 ChemIdent & ChemIdent::operator= (const ChemIdent & *a*)

Assignment operator.

4.9.3.11 bool ChemIdent::operator== (const ChemIdent & *a*) const

Comparison operator.

4.9.3.12 void ChemIdent::output () const

4.9.3.13 ChemIdent& ChemIdent::set\_dihedrals (int *dihedrals*, int *n*)

4.9.3.14 ChemIdent & ChemIdent::set\_return\_connector (const zmat\_connector & *A*)

4.9.3.15 ChemIdent & ChemIdent::set\_Z (const zmat & *Z*)

4.9.3.16 zmat\_entry& ChemIdent::update\_connector (const zmat\_connector & *a*, const  
zmat\_connector & *e*, const long *A*, zmat\_connector & *x*)[static]

#### 4.9.4 Member Data Documentation

4.9.4.1 refvector<refvector<long> > ChemIdent::allowed\_Substituents[private]

This double array holds the possible substitutions for each substitution site.

4.9.4.2 const refvector<refvector<long> >& ChemIdent::allowed\_Substituents\_r

This double array holds the possible substitutions for each substitution site. (READ ONLY)

4.9.4.3 refvector<zmat\_connector> ChemIdent::Connector[private]

This array of entries holds the connectivity data on the substituents within Z and the

complete framework.

#### 4.9.4.4 `refvector<zmat_connector>& ChemIdent::Connector_r`

This array of entries holds the connectivity data on the substituents within Z and the complete framework. (READ ONLY)

#### 4.9.4.5 `refvector<long> ChemIdent::occupation[mutable, private]`

This array holds current substituents at each site

#### 4.9.4.6 `refvector<long>ChIdent::occupation_r`

This array holds current substituents at each site. (READ ONLY)

#### 4.9.4.7 `zmat_connector ChemIdent::return_connector[private]`

This Z-matrix entry is returned in order to connect subsequent groups to this chemgroup.

#### 4.9.4.8 `const zmat_connector& ChemIdent::return_connector_r`

This Z-matrix entry is returned in order to connect subsequent groups to this chemgroup.

#### 4.9.4.9 `long ChemIdent::Space_Size[mutable, private]`

This number holds the possible number of combinations for the substitution sites.

#### 4.9.4.10 `zmat ChemIdent::Z[private]`

This Z-matrix holds the connectivity data for this group. Respective substitution sites will be added via their own `build_zmat()` calls. For unambiguous building of Z-matrices it may be necessary to include dummy atoms. These are easily omitted upon conversion to cartesian coordinates.

#### 4.9.4.11 `const zma& ChemIdent::Z_r`

This Z-matrix holds the connectivity data for this group. Respective substitution sites will be added via their own `build_zmat()` calls. For unambiguous building of Z-matrices it may be necessary to include dummy atoms. These are easily omitted upon conversion to cartesian coordinates. (READ ONLY)

The documentation for this class was generated from the following files:

- `chemident.h`

- chemident.cc

## 4.10 double\_truct Reference

Pair of configurational and conformational index, respectively.

```
#include <typedefs.hh>
```

### Public Attributes

- int configuration
- int conformation

### 4.10.1 Member Data Documentation

4.10.1.1 int double\_index::configuration

4.10.1.2 int double\_index::conformation

The documentation for this struct was generated from the following file:

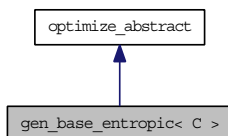
- typedefs.hh

## 4.11 gen\_base\_entropic< C > Class Template Reference

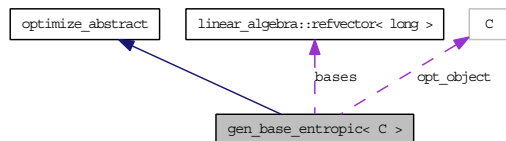
Meta-Optimize by generating maximally distant starting configurations.

```
#include <gen_base_entropic.hh>
```

Inheritance diagram for gen\_base\_entropic< C >:



Collaboration diagram for `gen_base_entropic< C >`:



## Public Member Functions

- `gen_base_entropic (const C &a, const refvector< long > &b)`
- `ulong optimize (ulong N) const`  
*Meta-Optimize by generating maximally distant starting configurations.*
- `valerg get_value (const ulong i) const`

## Public Attributes

- `ulong nruns`  
*Number of runs.*
- `long max_steps`  
*Maximum number of steps.*

## Protected Attributes

- `C opt_object`
- `refvector< long > bases`

## Private Member Functions

- `BOOST_CONCEPT_ASSERT ((Concepts::has_optimize< C >))`  
*Require C to conform to an optimization.*
- `BOOST_CONCEPT_ASSERT ((boost::CopyConstructible< C >))`  
*Require Copy constructibility.*

```
template<class C> class gen_base_entropic< C >
```

### 4.11.1 Constructor & Destructor Documentation

4.11.1.1 `template<class C> gen_base_entropic< C >::gen_base_entropic (const C & a, const refvector< long > & b)[inline]`

### 4.11.2 Member Function Documentation

4.11.2.1 `template<class C> gen_base_entropic< C >::BOOST_CONCEPT_ASSERT ((boost::CopyConstructible< C >))[private]`

4.11.2.2 `template<class C> gen_base_entropic< C >::BOOST_CONCEPT_ASSERT ((Concepts::has_optimize< C >))[private]`

4.11.2.3 `template<class C> valerg gen_base_entropic< C >::get_value (const ulong i) const[inline]`

4.11.2.4 `template<class C> ulong gen_base_entropic< C >::optimize (ulong N) const[inline, virtual]`

Implements `optimize_abstract`.

Here is the call graph for this function:



### 4.11.3 Member Data Documentation

4.11.3.1 `template<class C> refvector<long> gen_base_entropic< C >::bases[protected]`

4.11.3.2 `template<class C> long gen_base_entropic< C >::max_steps`

4.11.3.3 `template<class C> ulong gen_base_entropic< C >::nruns`

4.11.3.4 `template<class C> C gen_base_entropic< C >::opt_object[protected]`

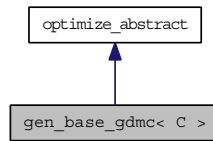
The documentation for this class was generated from the following file:

- `gen_base_entropic.hh`

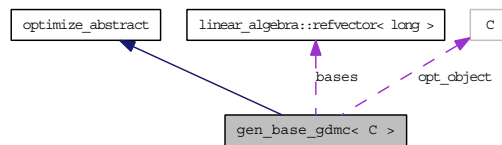
## 4.12 gen\_base\_gdmc< C > Class Template Reference

#include <genbasegdmc.hh>

Inheritance diagram for gen\_base\_gdmc< C >:



Collaboration diagram for gen\_base\_gdmc< C >:



### Public Member Functions

- gen\_base\_gdmc (const C &a, const refvector< long > &b)
- ulong optimize (ulong N) const  
*Gradient-directed Monte-Carlo optimization.*
- valerg get\_value (ulong i) const

### Public Attributes

- double T  
*Temperature.*
- ulong tight\_steps  
*Number of tightening steps.*
- ulong max\_steps  
*Maximum number of steps.*

### Private Member Functions



- BOOST\_CONCEPT\_ASSERT ((Concepts::has\_gradients< C >))  
*Require C to conform to the has\_gradients concept.*
- BOOST\_CONCEPT\_ASSERT ((Concepts::has\_optimize< C >))  
*Require C to conform to an optimization.*
- BOOST\_CONCEPT\_ASSERT ((Concepts::has\_stacksize< C >))  
*Require C to conform to providing computed number of configurations.*
- BOOST\_CONCEPT\_ASSERT ((boost::CopyConstructible< C >))  
*Require Copy constructibility.*

## Private Attributes

- C opt\_object
- refvector< long > bases

**template<class C> class gen\_base\_gdmc< C >**

### 4.12.1 Constructor & Destructor Documentation

4.12.1.1 template<class C> gen\_base\_gdmc< C >::gen\_base\_gdmc (const C & a, const refvector< long > & b)[inline]

### 4.12.2 Member Function Documentation

4.12.2.1 template<class C> gen\_base\_gdmc< C >::BOOST\_CONCEPT\_ASSERT ((boost::CopyConstructible< C >))[private]

4.12.2.2 template<class C> gen\_base\_gdmc< C >::BOOST\_CONCEPT\_ASSERT ((Concepts::has\_stacksize< C >))[private]

4.12.2.3 template<class C> gen\_base\_gdmc< C >::BOOST\_CONCEPT\_ASSERT ((Concepts::has\_optimize< C >))[private]

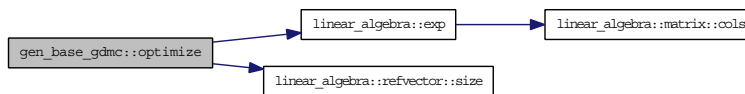
4.12.2.4 template<class C> gen\_base\_gdmc< C >::BOOST\_CONCEPT\_ASSERT ((Concepts::has\_gradients< C >))[private]

4.12.2.5 template<class C> valerg gen\_base\_gdmc< C >::get\_value (ulong i) const[inline]

4.12.2.6 template<class C> ulong gen\_base\_gdmc< C >::optimize (ulong N) const[inline, virtual]

Implements `optimize_abstract`.

Here is the call graph for this function:



### 4.12.3 Member Data Documentation

4.12.3.1 `template<class C> refvector<long> gen_base_gdmc< C >::bases[private]`

4.12.3.2 `template<class C> ulong gen_base_gdmc< C >::max_steps`

4.12.3.3 `template<class C> C gen_base_gdmc< C >::opt_object[private]`

4.12.3.4 `template<class C> double gen_base_gdmc< C >::T`

4.12.3.5 `template<class C> ulong gen_base_gdmc< C >::tight_steps`

The documentation for this class was generated from the following file:

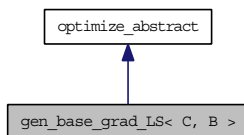
- `genbasegdmc.hh`

### 4.13 `gen_base_grad_LS< C, B >` Class Template Reference

Line search optimization class using general bases.

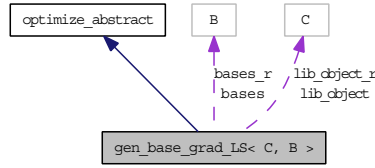
```
#include <genbase-grad-ls.hh>
```

Inheritance diagram for `gen_base_grad_LS< C, B >`:



Collaboration diagram for `gen_base_grad_LS< C, B >`:

#### Public Member Functions



- `gen_base_grad_LS (const C &Library)`
- `gen_base_grad_LS (const gen_base_grad_LS< C, B > &a)`  
*Copy constructor.*
- `refvector< valerg > gradient (const ulong conf1) const`  
*Gradient computation.*
- `refvector< valerg > & gradient (const ulong conf1, refvector< valerg > &r) const`  
*Gradient computation.*
- `long stacksize () const`
- `ulong optimize (ulong N) const`  
*Optimize starting from conformation i.*
- `valerg get_value (const ulong i) const`

## Public Attributes

- `const B & bases_r`
- `const C & lib_object_r`

## Private Member Functions

- `BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))`  
*Require Concepts::pruner compliance of C.*
- `BOOST_CONCEPT_ASSERT ((boost::CopyConstructible< C >))`
- `BOOST_CONCEPT_ASSERT ((Concepts::base_iterator< B >))`

## Private Attributes

- `const C lib_object`
- `B bases`

**template<class C, class B> class gen\_base\_grad\_LS< C, B >**

#### 4.13.1 Constructor & Destructor Documentation

4.13.1.1 `template<class C, class B> gen_base_grad_LS< C, B >::gen_base_grad_LS (const C & Library)`[inline]

4.13.1.2 `template<class C, class B> gen_base_grad_LS< C, B >::gen_base_grad_LS (const gen_base_grad_LS< C, B > & a)`[inline]

#### 4.13.2 Member Function Documentation

4.13.2.1 `template<class C, class B> gen_base_grad_LS< C, B >::BOOST_CONCEPT_ASSERT ((Concepts::base_iterator< B >))`[private]

4.13.2.2 `template<class C, class B> gen_base_grad_LS< C, B >::BOOST_CONCEPT_ASSERT ((boost::CopyConstructible< C >))`[private]

4.13.2.3 `template<class C, class B> gen_base_grad_LS< C, B >::BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))`[private]

4.13.2.4 `template<class C, class B> valerg gen_base_grad_LS< C, B >::get_value (const ulong i)` const[inline]

4.13.2.5 `template<class C, class B> refvector<valerg>& gen_base_grad_LS< C, B >::gradient (const ulong conf1, refvecto< valerg > & r)` const[inline]

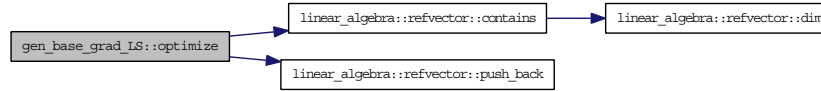
4.13.2.6 `template<class C, class B> refvector<valerg> gen_base_grad_LS< C, B >::gradient (const ulong conf1)` const[inline]

4.13.2.7 `template<class C, class B> ulong gen_base_grad_LS< C, B >::optimize (ulong N)` const[inline, virtual]

A Lagrange multiplier method is employed to enforce boundary constraints. The multiplier increases as the optimizatoin proceeds, enforcing the constraint ever more rigorously.

Implements `optimize_abstract`.

Here is the call graph for this function:



4.13.2.8 `template<class C, class B> long gen_base_grad_LS< C, B >::stacksize ()`  
`const[inline]`

### 4.13.3 Member Data Documentation

4.13.3.1 `template<class C, class B> B gen_base_grad_LS< C, B >::[mutable, private]`

4.13.3.2 `template<class C, class B> const B& gen_base_grad_LS< C, B >::bases_r`

4.13.3.3 `template<class C, class B> const C gen_base_grad_LS< C, B >::lib_object[private]`

4.13.3.4 `template<class C, class B> const C& gen_base_grad_LS< C, B >::lib_object_r`

The documentation for this class was generated from the following file:

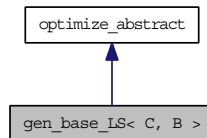
- `genbase-grad-ls.hh`

## 4.14 `gen_base_LS< C, B >` Class Template Reference

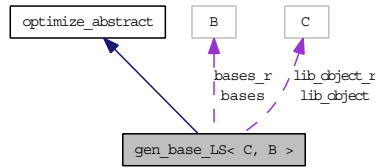
Line search optimization class using general bases.

`#include <genbase-l-s.hh>`

Inheritance diagram for `gen_base_LS< C, B >`:



Collaboration diagram for `gen_base_LS< C, B >`:



## \*Public Member Functions

- `gen_base_LS (const C &Library)`
- `ulong optimize (ulong N) const`  
*Optimize starting from conformation i.*
- `valerg get_value (const ulong i) const`

## Public Attributes

- `const B bases_r`
- `const C & lib_object_r`

## Private Member Functions

- `BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))`  
*Require Concepts::pruner compliance of C.*
- `BOOST_CONCEPT_ASSERT ((boost::CopyConstructible< C >))`
- `BOOST_CONCEPT_ASSERT ((Concepts::base_iterator< B >))`

## Private Attributes

- `const C lib_object`
- `B bases`

`template<class C, class B> class gen_base_LS< C, B >`

### 4.14.1 Constructor & Destructor Documentation

4.14.1.1 `template<class C, class B> gen_base_LS< C, B >::gen_base_LS (const C & Library)``[inline]`

### 4.14.2 Member Function Documentation

4.14.2.1 `template<class C, class B> gen_base_LS< C, B >::BOOST_CONCEPT_ASSERT ((Concepts::base_iterator< B >))``[private]`

4.14.2.2 `template<class C, class B> gen_base_LS< C, B >::BOOST_CONCEPT_ASSERT ((boost::CopyConstructible< C >))``[private]`

4.14.2.3 `template<class C, class B> gen_base_LS< C, B >::BOOST_CONCEPT_ASSERT ((Concepts::pruner< C >))``[private]`

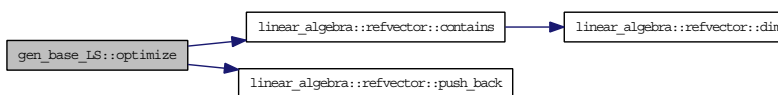
4.14.2.4 `template<class C, class B> valerg gen_base_LS< C, B >::get_value (const ulong i)``const``[inline]`

4.14.2.5 `template<class C, class B> ulong gen_base_LS< C, B >::optimize (ulong N)``const``[inline, virtual]`

A Lagrange multiplier method is employed to enforce boundary constraints. The multiplier increases as the optimizatoin proceeds, enforcing the constraint ever more rigorously.

Implements `optimize_abstract`.

Here is the call graph for this function:



### 4.14.3 Member Data Documentation

4.14.3.1 `template<class C, class B> B gen_base_LS< C, B >::bases``[mutable, private]`

4.14.3.2 `template<class C, class B> const B& gen_base_LS< C, B >::bases_r`

4.14.3.3 `template<class C, class B> const C gen_base_LS< C, B >::lib_object``[private]`

4.14.3.4 `template<class C, class B> const C& gen_base_LS< C, B >::lib_object_r`

The documentation for this class was generated from the following file:

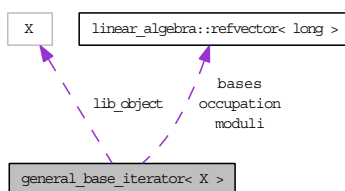
- genbase-l-s.hh

#### 4.15 `general_base_iterator< X >` Class Template Reference

Iterator over the potential bases.

```
#include <generalbaseiterator.hh>
```

Collaboration diagram for `general_base_iterator< X >`:



#### Public Member Functions

- `general_base_iterator (const X &b)`  
*Constructor. Mandatory library object.*
- `general_base_iterator (const general_base_iterator< X > &b)`  
*Copy Constructor.*
- `general_base_iterator< X > & operator++ (int i)`  
*Advance the iterator to the next number.*
- `ulong set_refstate (ulong newref)`  
*Set the reference state of the library to determine the relevance of subs.*
- `ulong get_refstate () const`  
*Retrieve the current refstate.*
- `long operator= (long i)`  
*Set the current state.*
- `long get_state () const`  
*Retrieve the current state.*



- long operator() () const  
*Retrieve the current base size.*
- long modulus () const  
*Modulus.*
- bool done () const  
*End of iterator?*
- long non\_empty\_size () const
- template<> bool done () const
- template<> void occupy (ulong number)
- template<> void compute\_bases (long Group, const refvector< long > &base\_sizes)
- template<> void compute\_bases ()
- template<> general\_base\_iterator (const general\_base\_iterator< chem\_opt > &b)
- template<> general\_base\_iterator (const chem\_opt &b)
- template<> long modulus () const
- template<> general\_base\_iterator< chem\_opt > & operator++ (int i)
- template<> ulong get\_refstate () const
- template<>ulong set\_refstate (ulong newref)
- template<> long operator= (long i)
- template<> long get\_state () const
- template<> long operator() () const
- template<> long non\_empty\_size () const

## Private Member Functions

- BOOST\_CONCEPT\_ASSERT ((Concepts::Library< X >))
- void compute\_bases ()  
*Compute offsets of groups.*

- void compute\_bases (long N, const refvector< long > &base\_sizes)  
*Compute the base for N.*
- void occupy (ulong number)  
*Set the occupation numbers w.r.t. number.*

## Private Attributes

- long number\_of\_bases
- ulong refstate  
*Reference state.*
- long state  
*Current base.*
- const X & lib\_object  
*Library object which is enumerated with the bases to be iterated.*
- refvector< long > bases  
*Offsets of the bases.*
- refvector< long > moduli  
*Moduli of the bases.*
- refvector< long > occupation  
*occupation of groups by current refstate.*

**template<class X> class general\_base\_iterator< X >**

### 4.15.1 Constructor & Destructor Documentation

4.15.1.1 template<class X> general\_base\_iterator< X >::general\_base\_iterator (const X & b)

4.15.1.2 template<class X> general\_base\_iterator< X >::general\_base\_iterator (const general\_base\_iterator< X > & b)

4.15.1.3 template<> general\_base\_iterator< chem\_opt >::general\_base\_iterator (const general\_base\_iterator< chem\_opt > & b)

4.15.1.4 template<> general\_base\_iterator< chem\_opt >::general\_base\_iterator (const chem\_opt & b)

## 4.15.2 Member Function Documentation

4.15.2.1 `template<class X> general_base_iterator< X >::BOOST_CONCEPT_ASSERT  
((Concepts::Library< X >))[private]`

4.15.2.2 `template<> void general_base_iterator< chem_opt >::compute_bases ()`

4.15.2.3 `template<> void general_base_iterator< chem_opt >::compute_bases (long Group,  
const refvector< long > & base_sizes)`

4.15.2.4 `template<class X> void general_base_iterator< X >::compute_bases (long N,  
const refvector< long > & base_sizes)[private]`

4.15.2.5 `template<class X> void general_base_iterator< X >::compute_bases ()[private]`

4.15.2.6 `template<> bool general_base_iterator< chem_opt >::done () const`

4.15.2.7 `template<class X> bool general_base_iterator< X >::done () const`

4.15.2.8 `template<> ulong general_base_iterator< chem_opt >::get_refstate () const`

4.15.2.9 `template<class X> ulong general_base_iterator< X >::get_refstate () const`

4.15.2.10 `template<> long general_base_iterator< chem_opt >::get_state () const`

4.15.2.11 `template<class X> long general_base_iterator< X >::get_state () const`

4.15.2.12 `template<> long general_base_iterator< chem_opt >::modulus () const`

4.15.2.13 `template<class X> long general_base_iterator< X >::modulus () const`

4.15.2.14 `template<> long general_base_iterator< chem_opt >::non_empty_size () const`

4.15.2.15 `template<class X> long general_base_iterator< X >::non_empty_size () const`

4.15.2.16 `template<> void general_base_iterator< chem_opt >::occupy (ulong number)`

4.15.2.17 `template<class X> void general_base_iterator< X >::occupy (ulong  
number)[private]`

4.15.2.18 `template<> long general_base_iterator< chem_opt >::operator() () const`

4.15.2.19 `template<class X> long general_base_iterator< X >::operator() () const`

4.15.2.20 `template<> general_base_iterator< chem_opt > & general_base_iterator<`

`chem_opt >::operator++ (int i)`

4.15.2.21 `template<class X> general_base_iterator<X>& general_base_iterator< X >::operator++ (int i)`

4.15.2.22 `template<> long general_base_iterator< chem_opt >::operator= (long i)`

4.15.2.23 `template<class X> long general_base_iterator< X >::operator= (long i)`

4.15.2.24 `template<> ulong general_base_iterator< chem_opt >::set_refstate (ulong newref)`

4.15.2.25 `template<class X> ulong general_base_iterator< X >::set_refstate (ulong newref)`

### 4.15.3 Member Data Documentation

4.15.3.1 `template<class X> refvector<long> general_base_iterator< X >::bases[mutable, private]`

4.15.3.2 `template<class X> const X& general_base_iterator< X >::lib_object[private]`

4.15.3.3 `template<class X> refvector<long> general_base_iterator< X >::moduli[mutable, private]`

4.15.3.4 `template<class X> long general_base_iterator< X >::number_of_bases[private]`

4.15.3.5 `template<class X> refvector<long> general_base_iterator< X >::occupation[private]`

4.15.3.6 `template<class X> ulong general_base_iterator< X >::refstate[private]`

4.15.3.7 `template<class X> long general_base_iterator< X >::state[private]`

The documentation for this class was generated from the following file:

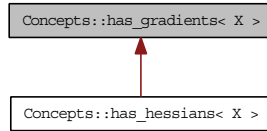
- `generalbaseiterator.hh`

## 4.16 Concepts::has\_gradients< X > Class Template Reference

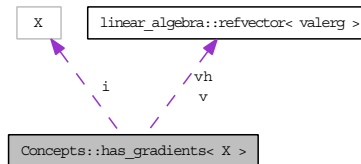
Concept defines a class that has gradients.

`#include <concepts.hh>`

Inheritance diagram for `Concepts::has_gradients< X >`:



Collaboration diagram for Concepts::has\_gradients< X >:



## Public Member Functions

- BOOST\_CONCEPT\_USAGE (has\_gradients)

## Private Attributes

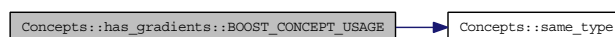
- X & i
- valer > v
- refvector< valerg > & vh

**template<class X> class Concepts::has\_gradients< X >**

### 4.16.1 Member Function Documentation

4.16.1.1 template<class X> Concepts::has\_gradients< X >::BOOST\_CONCEPT\_USAGE (has\_gradients< X >)[inline]

Here is the call graph for this function:



## 4.16.2 Member Data Documentation

4.16.2.1 `template<class X> X& Concepts::has_gradients< X >::i[private]`

Reimplemented in `Concepts::has_hessians< X >`.

4.16.2.2 `template<class X> refvector<valerg> Concepts::has_gradients< X >::v[private]`

Reimplemented in `Concepts::has_hessians< X >`.

4.16.2.3 `template<class X> refvector<valerg>& Concepts::has_gradients< X >::vh[private]`

Reimplemented in `Concepts::has_hessians< X >`.

The documentation for this class was generated from the following file:

- `concepts.hh`

## 4.17 `has_gradients_data< X >` Class Template Reference

Abstract class for discrete optimizations.

### Public Member Functions

- `has_gradients_data ()`
- `has_gradients_data (const has_gradients_data &a)`
- `has_gradients_data & operator= (const has_gradients_data &d)`
- `void diff (ulong n1, ulong n2, valerg &r) const`  
*compute the difference between numbers n1 and n2.*
- `refvector< valerg > gradient (const ulong i) const`  
*Gradient computation.*
- `refvector< valerg > & gradient (const ulong i, refvector< valerg > &v) const`  
*Gradient computation.*

### Protected Member Functions

- `void gradient_thread (boost::shared_mutex *mx, const ulong *i, ulong *k) const`

## Private Member Functions

- BOOST\_CONCEPT\_ASSERT ((Concepts::Library< X >))
- BOOST\_CONCEPT\_ASSERT ((Concepts::has\_Name< X >))
- BOOST\_CONCEPT\_ASSERT ((Concepts::is\_threadable< X >))

**template<class X> class has\_gradients\_data< X >**

### 4.17.1 Constructor & Destructor Documentation

4.17.1.1 template<class X> has\_gradients\_data< X >::has\_gradients\_data ()[inline]

4.17.1.2 template<class X> has\_gradients\_data< X >::has\_gradients\_data (const has\_gradients\_data< X > & a)[inline]

### 4.17.2 Member Function Documentation

4.17.2.1 template<class X> has\_gradients\_data< X >::BOOST\_CONCEPT\_ASSERT ((Concepts::is\_threadable< X >))[private]

4.17.2.2 template<class X> has\_gradients\_data< X >::BOOST\_CONCEPT\_ASSERT ((Concepts::has\_Name< X >))[private]

4.17.2.3 template<class X> has\_gradients\_data< X >::BOOST\_CONCEPT\_ASSERT ((Concepts::Library< X >))[private]

4.17.2.4 template<class X> void has\_gradients\_data< X >::diff (ulong *n1*, ulong *n2*, valerg & *r*) const

4.17.2.5 template<class X> refvector< valerg > & has\_gradients\_data< X >::gradient (const ulong *i*, refvector< valerg > & *v*) const

Here is the call graph for this function:



4.17.2.6 template<class X> refvector< valerg > has\_gradients\_data< X >::gradient (const ulong *i*) const

4.17.2.7 `template<class X> void has_gradients_data< X >::gradient_thread`  
`(boost::shared_mutex * mx, const ulong * i, ulong * k) const[protected]`

4.17.2.8 `template<class X> has_gradients_data& has_gradients_data< X >::operator=`  
`(const has_gradients_data< X > & d)`

The documentation for this class was generated from the following files:

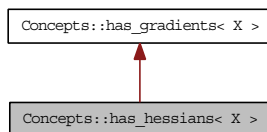
- `has_gradients_hessian_data.decl`
- `has_gradients_hessian_data.hh`

## 4.18 Concepts::has\_hessians< X > Class Template Reference

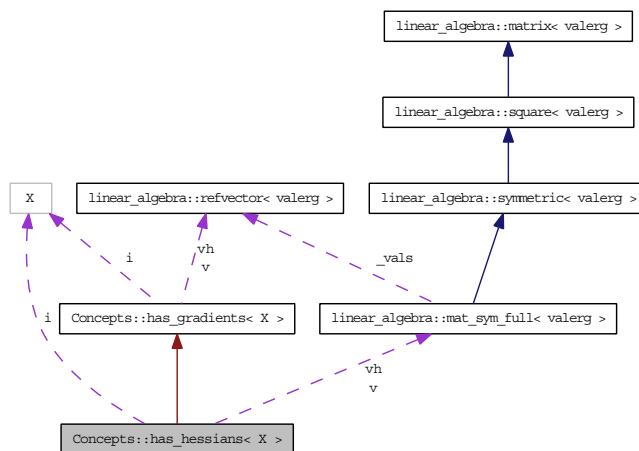
Concept defines a class that has hessians.

`#include <concepts.hh>`

Inheritance diagram for Concepts::has\_hessians< X >:



Collaboration diagram for Concepts::has\_hessians< X >:



## Public Member Functions



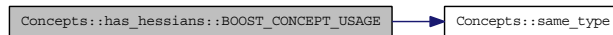
- BOOST\_CONCEPT\_USAGE (has\_hessians)
- X i
- mat\_sym\_full< valerg > v
- mat\_sym\_full< valerg > & vh

**template<class X> class Concepts::has\_hessians< X >**

#### 4.18.1 Member Function Documentation

4.18.1.1 template<class X> Concepts::has\_hessians< X >::BOOST\_CONCEPT\_USAGE (has\_hessians< X >)[inline]

Here is the call graph for this function:



#### 4.18.2 Member Data Documentation

4.18.2.1 template<class X> X Concepts::has\_hessians< X >::i[private]

Reimplemented from Concepts::has\_gradients< X >.

4.18.2.2 template<class X> mat\_sym\_full<valerg> Concepts::has\_hessians< X >::v[private]

Reimplemented from Concepts::has\_gradients< X >.

4.18.2.3 template<class X> mat\_sym\_full<valerg>& Concepts::has\_hessians< X >::vh[private]

Reimplemented from Concepts::has\_gradients< X >.

The documentation for this class was generated from the following file:

- concepts.hh

## 4.19 has\_hessians\_data< X > Class Template Reference

Abstract class for discrete optimizations.

### Public Member Functions

- has\_hessians\_data ()
- has\_hessians\_data (const has\_hessians\_data &a)
- has\_hessians\_data & operator= (const has\_hessians\_data &d)
- mat\_sym\_full< valerg > hessian (ulong i) const  
*Hessian computation.*
- mat\_sym\_full< valerg > & hessian (ulong i, mat\_sym\_full< valerg > &H) const  
*Hessian computation.*

### Public Attributes

- const string & Name\_r

### Private Member Functions

- BOOST\_CONCEPT\_ASSERT ((Concepts::has\_gradients< X >))
- BOOST\_CONCEPT\_ASSERT ((Concepts::Library< X >))
- BOOST\_CONCEPT\_ASSERT ((Concepts::has\_Name< X >))
- BOOST\_CONCEPT\_ASSERT ((Concepts::is\_threadable< X >))

**template<class X> class has\_hessians\_data< X >**

#### 4.19.1 Constructor & Destructor Documentation

4.19.1.1 template<class X> has\_hessians\_data< X >::has\_hessians\_data ()[inline]

4.19.1.2 template<class X> has\_hessians\_data< X >::has\_hessians\_data (const has\_hessians\_data< X > & a)[inline]

## 4.19.2 Member Function Documentation

4.19.2.1 `template<class X> has_hessians_data< X >::BOOST_CONCEPT_ASSERT  
((Concepts::is_threadable< X >))[private]`

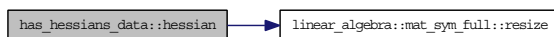
4.19.2.2 `template<class X> has_hessians_data< X >::BOOST_CONCEPT_ASSERT  
((Concepts::has_Name< X >))[private]`

4.19.2.3 `template<class X> has_hessians_data< X >::BOOST_CONCEPT_ASSERT  
((Concepts::Library< X >))[private]`

4.19.2.4 `template<class X> has_hessians_data< X >::BOOST_CONCEPT_ASSERT  
((Concepts::has_gradients< X >))[private]`

4.19.2.5 `template<class X> mat_sym_full< valerg > & has_hessians_data< X >::hessian  
(ulong i, mat_sym_full< valerg > & H) const`

Here is the call graph for this function:



4.19.2.6 `template<class X> mat_sym_full< valerg > has_hessians_data< X >::hessian  
(ulong i) const`

4.19.2.7 `template<class X> has_hessians_data& has_hessians_data< X >::operator= (const  
has_hessians_data< X > & d)`

## 4.19.3 Member Data Documentation

4.19.3.1 `template<class X> const string& has_hessians_data< X >::Name_r`

The documentation for this class was generated from the following files:

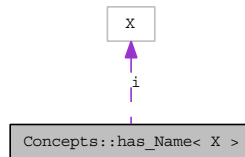
- `has_gradients_hessian_data.decl`
- `has_gradients_hessian_data.hh`

## 4.20 Concepts::has\_Name< X > Class Template Reference

Concept defines a class that has a private string Name.

```
#include <concepts.hh>
```

Collaboration diagram for Concepts::has\_Name< X >:



## Public Member Functions

- BOOST\_CONCEPT\_USAGE (has\_Name)

## Private Attributes

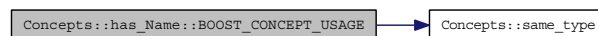
- X & i
- string N
- const string & Nh
- string M

```
template<class X> class Concepts::has_Name< X >
```

### 4.20.1 Member Function Documentation

4.20.1.1 template<class X> Concepts::has\_Name< X >::BOOST\_CONCEPT\_USAGE  
(has\_Name< X >)[inline]

Here is the call graph for this function:



## 4.20.2 Member Data Documentation

4.20.2.1 `template<class X> X& Concepts::has_Name< X >::i[private]`

4.20.2.2 `template<class X> string Concepts::has_Name< X >::M[private]`

4.20.2.3 `template<class X> string Concepts::has_Name< X >::N[mutable, private]`

4.20.2.4 `template<class X> const string& Concepts::has_Name< X >::Nh[private]`

The documentation for this class was generated from the following file:

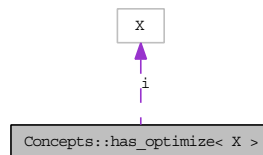
- `concepts.hh`

## 4.21 Concepts::has\_optimize< X > Class Template Reference

Concept defines a class that has an optimize member.

```
#include <concepts.hh>
```

Collaboration diagram for `Concepts::has_optimize< X >`:



### Public Member Functions

- `BOOST_CONCEPT_USAGE (has_optimize)`

### Private Attributes

- `X & i`
- `ulong index`
- `string & id`

```
template<class X> class Concepts::has_optimize< X >
```

### 4.21.1 Member Function Documentation

4.21.1.1 `template<class X> Concepts::has_optimize< X >::BOOST_CONCEPT_USAGE`  
(`has_optimize< X >`)[inline]

Here is the call graph for this function:



### 4.21.2 Member Data Documentation

4.21.2.1 `template<class X> X& Concepts::has_optimize< X >::i`[private]

4.21.2.2 `template<class X> string& Concepts::has_optimize< X >::id`[private]

4.21.2.3 `template<class X> ulong Concepts::has_optimize< X >::index`[private]

The documentation for this class was generated from the following file:

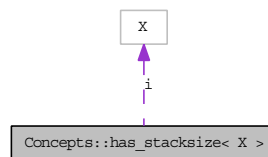
- `concepts.hh`

## 4.22 Concepts::has\_stacksize< X > Class Template Reference

Concept defines a class that has a stack member.

```
#include <concepts.hh>
```

Collaboration diagram for `Concepts::has_stacksize< X >`:



### Public Member Functions

- `BOOST_CONCEPT_USAGE` (`has_stacksize`)

## Private Attributes

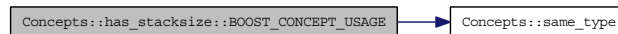
- X & i
- long index
- string & id

**template<class X> class Concepts::has\_stacksize< X >**

### 4.22.1 Member Function Documentation

4.22.1.1 `template<class X> Concepts::has_stacksize< X >::BOOST_CONCEPT_USAGE`  
(`has_stacksize< X >`)[inline]

Here is the call graph for this function:



### 4.22.2 Member Data Documentation

4.22.2.1 `template<class X> X& Concepts::has_stacksize< X >::i`[private]

4.22.2.2 `template<class X> string& Concepts::has_stacksize< X >::id`[private]

4.22.2.3 `template<class X> long Concepts::has_stacksize< X >::index`[private]

The documentation for this class was generated from the following file:

- `concepts.hh`

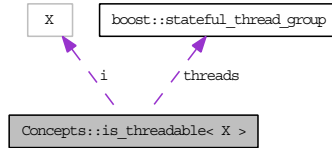
## 4.23 Concepts::is\_threadable< X > Class Template Reference

Concept defines a class that is threadable.

```
#include <concepts.hh>
```

Collaboration diagram for `Concepts::is_threadable< X >`:

### Public Member Functions



- BOOST\_CONCEPT\_USAGE (is\_threadable)

## Private Attributes

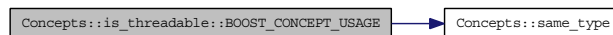
- X i
- ulong mt
- boost::shared\_mutex hm
- boost::stateful\_thread\_group threads

**template<class X> class Concepts::is\_threadable< X >**

### 4.23.1 Member Function Documentation

4.23.1.1 **template<class X> Concepts::is\_threadable< X >::BOOST\_CONCEPT\_USAGE (is\_threadable< X >)[inline]**

Here is the call graph for this function:



### 4.23.2 Member Data Documentation

4.23.2.1 **template<class X> boost::shared\_mutex Concepts::is\_threadabl< X >::hm[mutable, private]**

4.23.2.2 **template<class X> X Concepts::is\_threadable< X >::i[private]**

4.23.2.3 **template<class X> ulong Concepts::is\_threadable< X >::mt[private]**

4.23.2.4 **template<class X> boost::stateful\_thread\_group Concepts::is\_threadable< X >::threads[mutable, private]**



The documentation for this class was generated from the following file:

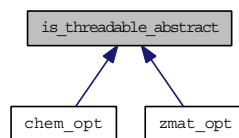
- concepts.hh

#### 4.24 is\_threadable\_abstract Class Reference

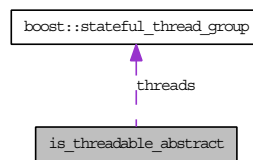
Abstract class implementing the Concepts::is\_threadable concept.

#include <isthreadableabstract.h>

Inheritance diagram for is\_threadable\_abstract:



Collaboration diagram for is\_threadable\_abstract:



#### Public Member Functions

- void set\_max\_threads (ulong N)  
*Set the maximum number of threads.*
- is\_threadable\_abstract ()  
*Default constructor. Minimum of one (1) threads.*
- is\_threadable\_abstract (const is\_threadable\_abstract &a)  
*Copy constructor.*
- is\_threadable\_abstract & operator= (const is\_threadable\_abstract &a)  
*Assignment operator.*

#### Protected Attributes

- `ulong max_threads`  
*Maximum number of concurrent threads.*
- `boost::shared_mutex history_mutex`  
*Shared mutex for locking during write operations.*
- `boost::stateful_thread_group threads`  
*Group of threads.*

#### 4.24.1 Constructor & Destructor Documentation

4.24.1.1 `is_threadable_abstract::is_threadable_abstract ()`[inline]

4.24.1.2 `is_threadable_abstract::is_threadable_abstract (const is_threadable_abstract & a)`[inline]

#### 4.24.2 Member Function Documentation

4.24.2.1 `is_threadable_abstract& is_threadable_abstract::operator= (const is_threadable_abstract & a)`[inline]

4.24.2.2 `ulong N`[inline]

#### 4.24.3 Member Data Documentation

4.24.3.1 `boost::shared_mutex is_threadable_abstract::history_mutex`[mutable, protected]

4.24.3.2 `ulong is_threadable_abstract::max_threads`[protected]

4.24.3.3 `boost::stateful_thread_group is_threadable_abstract::threads`[mutable, protected]

The documentation for this class was generated from the following file:

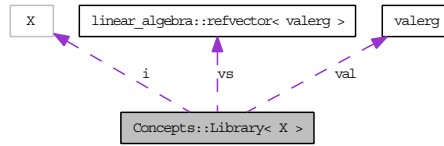
- `isthreadableabstract.h`

### 4.25 Concepts::Library< X > Class Template Reference

Concept defines an addressable set of values.

```
#include <concepts.hh>
```

Collaboration diagram for `Concepts::Library< X >`:



## Public Member Functions

- BOOST\_CONCEPT\_USAGE (Library)

## Private Attributes

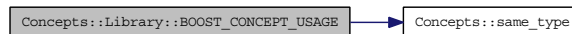
- X i
- valerg val
- ulong index
- ulong mi
- refvector< ulong > rd
- refvector< valerg > vs

**template<class X> class Concepts::Library< X >**

### 4.25.1 Member Function Documentation

4.25.1.1 template<class X> Concepts::Library< X >::BOOST\_CONCEPT\_USAGE (Library< X >)[inline]

Here is the call graph for this function:



## 4.25.2 Member Data Documentation

4.25.2.1 `template<class X> X& Concepts::Library< X >::i[private]`

4.25.2.2 `template<class X> ulong Concepts::Library< X >::index[private]`

4.25.2.3 `template<class X> ulong Concepts::Library< X >::mi[mutable, private]`

4.25.2.4 `template<class X> refvector<ulong> Concepts::Library< X >::rd[mutable, private]`

4.25.2.5 `template<class X> valerg Concepts::Library< X >::val[private]`

4.25.2.6 `template<class X> refvector<valerg> Concepts::Library< X >::vs[mutable, private]`

The documentation for this class was generated from the following file:

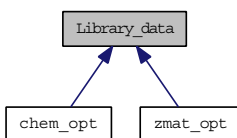
- `concepts.hh`

## 4.26 Library\_data Class Reference

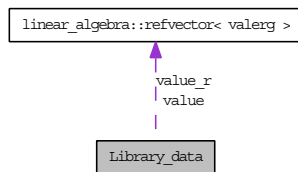
Abstract class for discrete optimizations.

`#include <Library_data.hh>`

Inheritance diagram for `Library_data`:



Collaboration diagram for `Library_data`:



## Public Member Functions

- `Library_data ()`  
*Default Constructor.*
- `Library_data (const Library_data &a)`  
*Copy constructor.*
- `virtual ~Library_data ()`  
*Destructor.*
- `Library_data & operator= (const Library_data &d)`  
*Assignment operator.*
- `virtual ulong get_space_size () const =0`  
*Compute the size of the optimization space.*
- `virtual ulong get_bits () const =0`  
*Compute the number of bits to address the optimization space.*
- `valerg get_value (ulong i) const`  
*Retrieve a value for a number.*
- `void set_Name (const string &A) const`  
*Sets the name for the purpose of writing files etc.*

## Public Attributes

- `const string & Name_r`  
*Access to Name (Read-Only).*
- `const refvector< ulong > & visited_r`  
*Visited numbers (Read-Only).*
- `const refvector< valerg > & value_r`  
*Values of visited numbers (Read-Only).*

## Protected Member Functions

- `virtual valerg compute_property (ulong i) const =0`  
*Class must have a way to compute values.*

## Protected Attributes

- `refvector< ulong > visited`  
*Visited numbers.*
- `refvector< valerg > value`  
*Values of visited numbers.*
- `ulong space_size`
- `bool space_size_computed`
- `ulong bits`
- `bool bits_computed`
- `string Nam`

#### 4.26.1 Detailed Description

This class provides the API that guarantees the `Concepts::Library` concept. The library is considered an access class to a static `Library`. Therefore, many of its members are mutable since the static library is too large to be stored completely. Instead, entries are computed on the fly and memoized.

#### 4.26.2 Constructor & Destructor Documentation

4.26.2.1 `Library_data::Library_data ()`[inline]

4.26.2.2 `Library_data::Library_data (const Library_data & a)`[inline]

4.26.2.3 `virtual Library_data::~~Library_data ()`[inline, virtual]

#### 4.26.3 Member Function Documentation

4.26.3.1 `virtual valerg Library_data::compute_property (ulong i) const`[protected, pure virtual]

Implemented in `chem_opt`, and `zmat_opt`.

4.26.3.2 `virtual ulong Library_data::get_bits () const`[pure virtual]

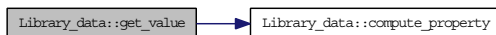
Implemented in `chem_opt`, and `zmat_opt`.

4.26.3.3 `virtual ulong Library_data::get_space_size () const`[pure virtual]

Implemented in `chem_opt`, and `zmat_opt`.

4.26.3.4 `valerg Library_data::get_value (ulong i) const[inline]`

Here is the call graph for this function:



## 4.26.4 Member Data Documentation

4.26.4.1 `ulong Library_data::bits[mutable, protected]`

4.26.4.2 `bool Library_data::bits_computed[mutable, protected]`

4.26.4.3 `string Library_data::Name[mutable, protected]`

4.26.4.4 `const string& Library_data::Name_r`

4.26.4.5 `ulong Library_data::space_size[mutable, protected]`

4.26.4.6 `bool Library_data::space_size_computed[mutable, protected]`

4.26.4.7 `refvector<valerg> Library_data::value[mutable, protected]`

4.26.4.8 `const refvector<valerg>& Library_data::value_r`

4.26.4.9 `refvector<ulong> Library_data::visited[mutable, protected]`

4.26.4.10 `const refvector<ulong>& Library_data::visited_r`

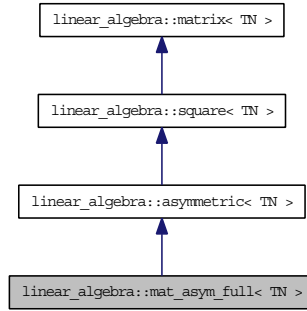
The documentation for this class was generated from the following files:

- `Library_data.hh`
- `Library_data.cc`

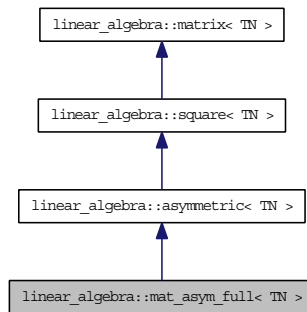
## 4.27 `linear_algebra::mat_asym_full< TN >` Class Template Reference

`mat_asym_full` uses packing for full antisymmetric matrices

Inheritance diagram for `linear_algebra::mat_asym_full< TN >`:



Collaboration diagram for linear\_algebra::mat\_asym\_full< TN >:



## Public Member Functions

- `mat_asym_full ()`  
*Default constructor.*
- `mat_asym_full (long n)`  
*Constructor of dimension  $n$  with uninitialised values.*
- `mat_asym_full (long n, const std::vector< TN > &vals)`  
*Constructor of dimension  $n$  with initialisation of values.*
- `mat_asym_full (long n, refvector< TN > &vals)`  
*Constructor of dimension  $n$  with initialisation of values.*
- `mat_asym_full (std::istream &IN)`  
*Constructor to read a stored matrix from a stream. Aimed at retrieving information from a binary save with `mat_full::operator>>()`.*
- `void operator>> (std::ostream &OUT) const`  
*Output an asymmetric, full matrix in binary format into a stream. Aimed at file storage.*



- `const TN & operator() (const long x, const long y) const`  
*Access operator which checks for integrity in debug mode.*
- `TN operator() (const long x, const long y)`  
*Access operator which checks for integrity in debug mode.*
- `const TN & operator[] (const long x) const`  
*Direct access operator which checks for integrity in debug mode.*
- `TN & operator[] (const long x)`  
*Direct access operator which checks for integrity in debug mode.*
- `mat_full< TN > & multiply (const mat_full< TN > &a, mat_full< TN > &r) const`  
*Multiply by a full matrix.*
- `mat_full< TN > operator * (const mat_full< TN > &a) const`  
*Multiply by a full matrix.*
- `mat_sparse< TN > multiply (const mat_sym_sparse< TN > &b, mat_sparse< TN > &r) const`  
*Multiply two sparse matrices.*
- `sparse_vector< TN > & multiply (const sparse_vector< TN > &v, sparse_vector< TN > &r) const`  
*Multiply an asymmetric full matrix with a sparse vector.*
- `mat_asym_full< TN > & operator *= (TN a)`  
*Multiply by a number back into the matrix.*
- `mat_asym_full< TN > operator * (TN a) const`  
*Multiply by a number.*
- `refvector< TN > & multiply (const refvector< TN > &v, refvector< TN > &r) const`  
*Multiply an asymmetric full matrix with a full vector.*
- `refvector< TN > operator * (const refvector< TN > &v) const`  
*Multiply an asymmetric full matrix with a full vector.*
- `refvector< TN > & multiply (const sparse_vector< TN > &v, refvector< TN > &r) const`  
*Multiply an asymmetric full matrix with a full vector.*
- `refvector< TN > operator * (const sparse_vector< TN > &v) const`  
*Multiply an asymmetric full matrix with a sparse vector.*

- `mat_asym_full< TN > operator+ (const mat_asym_full< TN > &a) const`  
*Add an antisymmetric full matrix to an existing one.*
- `mat_asym_full< TN > & operator+= (const mat_asym_full< TN > &a)`  
*Add an antisymmetric full matrix to an existing one.*
- `mat_asym_full< TN > & operator-= (const mat_asym_full< TN > &a)`  
*Subtract an antisymmetric full matrix to an existing one.*
- `refvector< TN > extract_col (const long x) const`  
*Extract a column from an antisymmetric matrix.*
- `mat_full< TN > & multiply (const mat_asym_full< TN > &B, mat_full< TN > &r)`  
`const`  
*Multiply an antisymmetric full matrix with an antisymmetric full matrix.*
- `mat_full< TN > operator * (const mat_asym_full< TN > &B) const`  
*Multiply an antisymmetric full matrix with an antisymmetric full matrix.*
- `mat_full< TN > operator * (const mat_sym_full< TN > &B) const`  
*Multiply an antisymmetric full matrix with a symmetric full matrix.*
- `mat_full< TN > & multiply (const mat_sym_sparse< TN > &B, mat_full< TN > &r)`  
`const`  
*Multiply an antisymmetric full matrix with a symmetric full matrix.*
- `mat_full< TN > operator * (const mat_sym_sparse< TN > &B) const`  
*Multiply an antisymmetric full matrix with a symmetric sparse matrix.*
- `mat_sym_full< TN > aa_() const`  
*Get the product  $AA^*$ .*
- `void add (const long x, const long y, TN a)`
- `bool display () const`  
*Simple display of matrix.*
- `void set (const long x, const long y, TN value)`  
*Assignment operation for compatibility with `mat_sym_sparse`.*
- `mat_asym_full< TN > & copy (const mat_asym_full< TN > a)`  
*Copy matrix with depth 1.*
- `mat_asym_full< TN > & copy (const mat_asym_full< TN > a, const long i)`  
*Copy matrix with depth i.*

- void clear ()  
*Reset matrix releasing all associated memory.*
- mat\_asym\_full< TN > & zero ()  
*Set matrix to 0, i.e., all elements.*
- long size () const  
*Returns the size of the vector containing the matrix data.*
- mat\_asym\_full< TN > & utu (const mat\_asym\_full< TN > &a, mat\_asym\_full< TN > &R, mat\_full< TN > &I) const  
*This a symmetric transform SAS.*
- mat\_asym\_full< TN > utu (const mat\_asym\_full< TN > &a, mat\_full< TN > &I) const  
*This an antisymmetric transform SAS.*
- mat\_asym\_full< TN > & utu (const mat\_asym\_full< TN > &a, mat\_asym\_full< TN > &R) const  
*This an antisymmetric transform SAS.*
- mat\_asym\_full< TN > utu (const mat\_asym\_full< TN > &a) const  
*This a symmetric transform SAS.*
- mat\_asym\_full< TN > & utu (const mat\_full< TN > &U, mat\_asym\_full< TN > &R) const  
*This is the unitary transform  $U^*AU$ .*
- mat\_asym\_full< TN > utu (const mat\_full< TN > &U) const  
*This is the unitary transform  $U^{\wedge t}AU$ .*
- mat\_asym\_full< TN > & uut (const mat\_full< TN > &U, mat\_asym\_full< TN > &R, mat\_full< TN > &I) const  
*This is the unitary transform  $UAU^*$ .*
- mat\_asym\_full< TN > & uut (const mat\_full< TN > &U, mat\_asym\_full< TN > &R) const  
*This is the unitary transform  $UAU^*$ .*
- mat\_asym\_full< TN > uut (const mat\_full< TN > &U) const  
*This is the unitary transform  $UAU^*$ .*
- void gen\_mat (const refvector< TN > &a, const refvector< TN > &b)  
*Produce the necessary  $ij^* - ji^*$ .*

- `mat_full< TN > householder (refvector< TN > &subdiagonal)`  
*This routine produces the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a full anti-symmetric matrix.*
- `mat_asym_full< TN > & prune (const TN tol)`  
*Prune the matrix of negligible values.*

## Private Attributes

- `refvector< TN > _vals`  
*Vector of values including the diagonal (which by necessity is 0).*

### 4.27.1 Detailed Description

`template<class TN> class linear_algebra::mat_asym_full< TN >`

The packing is optimised for column access. The matrix is stored as the upper triangle. The lower triangle is the negative of the upper triangle. It is important to note that the diagonal is stored explicitly and may be non-zero. As such, these matrices are not necessarily anti-symmetric, but pseudo antisymmetric in terms of the off diagonal terms.

### 4.27.2 Constructor & Destructor Documentation

4.27.2.1 `template<class TN> linear_algebra::mat_asym_full< TN >::mat_asym_full ()`

`mat_asym_full` uses packing for full antisymmetric matrices Default constructor.

4.27.2.2 `template<class TN> linear_algebra::mat_asym_full< TN >::mat_asym_full (long n)`

4.27.2.3 `template<class TN> linear_algebra::mat_asym_full< TN >::mat_asym_full (long n, const std::vector< TN > & vals)`

4.27.2.4 `template<class TN> linear_algebra::mat_asym_full< TN >::mat_asym_full (long n, refvector< TN > & vals)`

4.27.2.5 `template<class TN> linear_algebra::mat_asym_full< TN >::mat_asym_full (std::istream & IN)`

### 4.27.3 Member Function Documentation

4.27.3.1 `template<class TN> mat_sym_full< TN > linear_algebra::mat_asym_full< TN >::aa.t () const`

Test

4.27.3.2 `template<class TN> void linear_algebra::mat_asym_full< TN >::add (const long  $x$ , const long  $y$ , TN  $a$ )`

4.27.3.3 `template<class TN> void linear_algebra::mat_asym_full< TN >::clear ()`

4.27.3.4 `template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::copy (const mat_asym_full< TN >  $a$ , const long  $i$ )`

4.27.3.5 `template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_full< TN >::copy (const mat_asym_full< TN >  $a$ )`

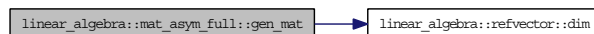
4.27.3.6 `template<class TN> bool linear_algebra::mat_asym_full< TN >::display () const`

4.27.3.7 `template<class TN> refvector< TN > linear_algebra::mat_asym_full< TN >::extract_col (const long  $x$ ) const`

4.27.3.8 `template<class TN> void linear_algebra::mat_asym_full< TN >::gen_mat (const refvector< TN > &  $a$ , const refvector< TN > &  $b$ )`

the result is  $+= ab^* - ba^*$ .

Here is the call graph for this function:



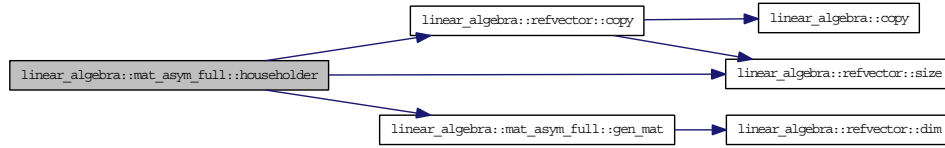
4.27.3.9 `template<class TN> mat_full< TN > linear_algebra::mat_asym_full< TN >::householder (refvector< TN > & subdiagonal)`

The aspired transformation is a multiplication by  $(I - vv^*)$  where  $v$  is  $(\pm e_1 \|a_{12}\| + a_{12}) / \sqrt{\pm a_{12,1} \|a_{12}\| + \|a_{12}\|^2}$ . Notice that we begin counting at 0. The matrix is split up according to  $A = \begin{pmatrix} a_{11} & a_{12}^* \\ a_{12} & A_{22} \end{pmatrix}$ . We use preconditioning such that the matrix is ordered with lowest off-diagonal value in the low right-hand corner.

$$\begin{pmatrix} a_{11} & \cdots & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & 0 \\ & & & & a_{nn} \end{pmatrix}$$

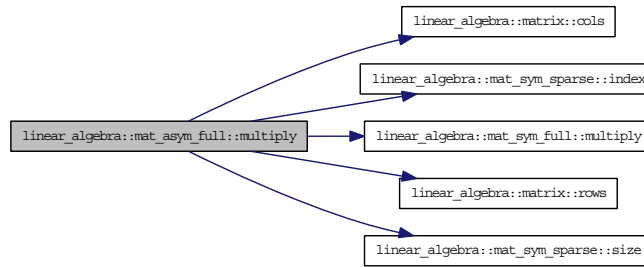
Here is the call graph for this function:

4.27.3.10 `template<class TN> mat_full< TN > & linear_algebra::mat_asym_4_full< TN`



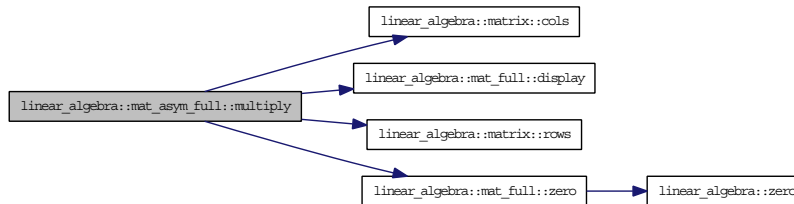
>::multiply (const mat\_sym\_sparse< TN > & B, mat\_full< TN > & r) const

Here is the call graph for this function:



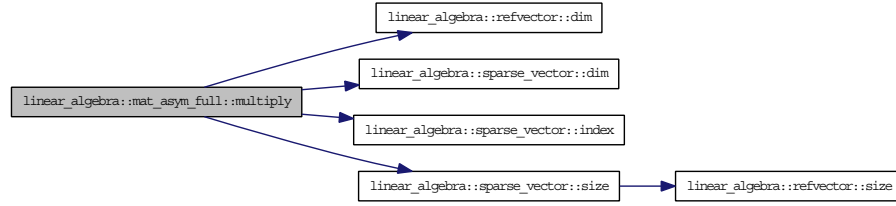
4.27.3.11 template<class TN> mat\_full< TN > & linear\_algebra::mat\_asym\_full< TN >::multiply (const mat\_asym\_full< TN > & B, mat\_full< TN > & r) const

Here is the call graph for this function:



4.27.3.12 template<class TN> refvector< TN > & linear\_algebra::mat\_asym\_full< TN >::multiply (const sparse\_vector< TN > & v, refvector< TN > & r) const

Here is the call graph for this function:



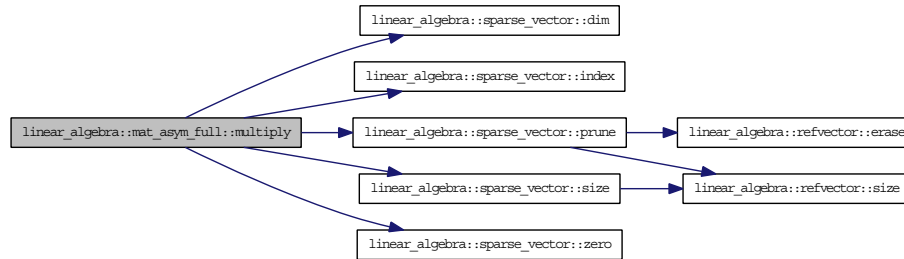
4.27.3.13 `template<class TN> refvector< TN > & linear_algebra::mat_asym_full< TN >::multiply (const refvector< TN > & v, refvector< TN > & r) const`

Here is the call graph for this function:



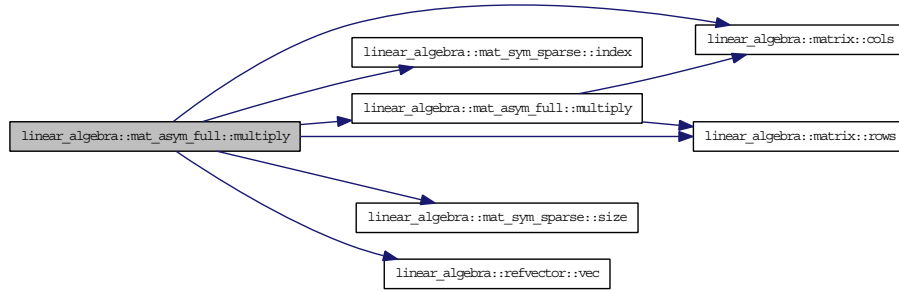
4.27.3.14 `template<class TN> sparse_vector< TN > & linear_algebra::mat_asym_full< TN >::multiply (const sparse_vector< TN > & v, sparse_vector< TN > & r) const`

Here is the call graph for this function:



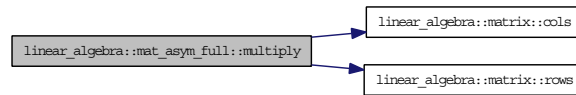
4.27.3.15 `template<class TN> mat_sparse< TN > & linear_algebra::mat_asym_full< TN >::multiply (const mat_sym_sparse< TN > & b, mat_sparse< TN > & r) const`

Here is the call graph for this function:



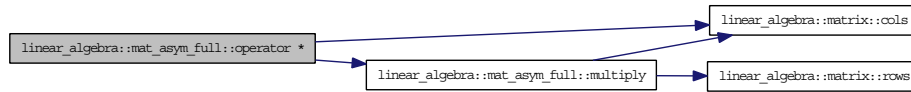
4.27.3.16 `template<class TN> mat_full< TN > & linear_algebra::mat_asym_full< TN >::multiply (const mat_full< TN > & a, mat_full< TN > & r) const`

Here is the call graph for this function:



4.27.3.17 `template<class TN> mat_full< TN > linear_algebra::mat_asym_full< TN >::operator * (const mat_sym_sparse< TN > & B) const`

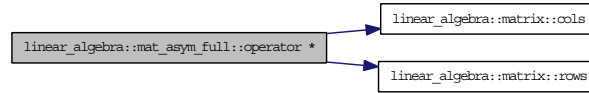
Here is the call graph for this function:



4.27.3.18 `template<class TN> mat_full< TN > linear_algebra::mat_asym_full< TN >::operator * (const mat_sym_full< TN > & B) const`

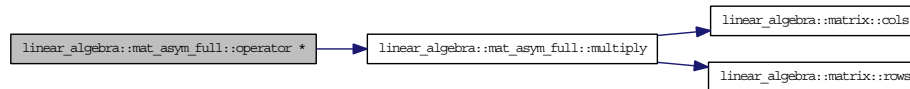
Here is the call graph for this function:





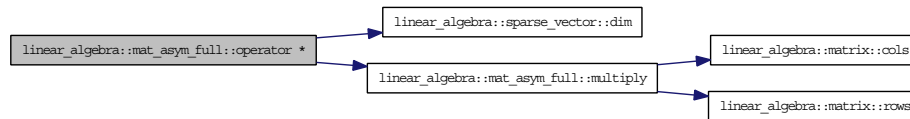
4.27.3.19 `template<class TN> mat_full< TN > linear_algebra::mat_asym_full< TN >::operator * (const mat_asym_full< TN > & B) const`

Here is the call graph for this function:



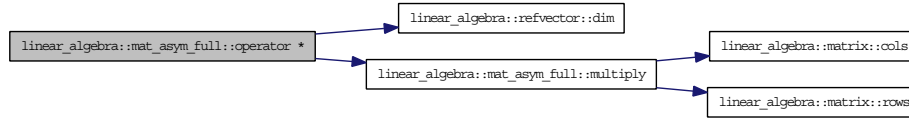
4.27.3.20 `template<class TN> refvector< TN > linear_algebra::mat_asym_full< TN >::operator * (const sparse_vector< TN > & v) const`

Here is the call graph for this function:



4.27.3.21 `template<class TN> refvector< TN > linear_algebra::mat_asym_full< TN >::operator * (const refvector< TN > & v) const`

Here is the call graph for this function:

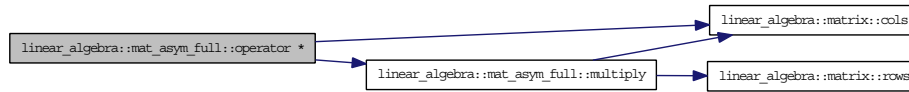


4.27.3.22 `template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_full< TN >::operator * (TN a) const`

Reimplemented from `linear_algebra::asymmetric< TN >`.

4.27.3.23 `template<class TN> mat_full< TN > linear_algebra::mat_asym_full< TN >::operator * (const mat_full< TN > & a) const`

Here is the call graph for this function:



4.27.3.24 `template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::operator *= (TN a)`

4.27.3.25 `template<class TN> TN linear_algebra::mat_asym_full< TN >::operator() (const long x, const long y)`

4/27/3/26 `template<class TN> const TN & linear_algebra::mat_asym_full< TN >::operator() (const long x, const long y) const[virtual]`

Implements `linear_algebra::matrix< TN >`.

4.27.3.27 `template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_full< TN >::operator+ (const mat_asym_full< TN > & a) const`

4.27.3.28 `template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::operator+= (const mat_asym_full< TN > & a)`

4.27.3.29 `template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::operator-= (const mat_asym_full< TN > & a)`

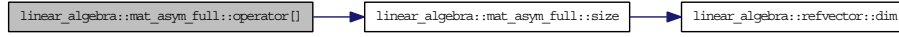
4.27.3.30 `template<class TN> void linear_algebra::mat_asym_full< TN >::operator>> (std::ostream & OUT) const[virtual]`

Reimplemented from `linear_algebra::matrix< TN >`.

4.27.3.31 `template<class TN> TN & linear_algebra::mat_asym_full< TN >::operator[]  
(const long x)`

4.27.3.32 `template<class TN> const TN & linear_algebra::mat_asym_full< TN  
>::operator[] (const long x) const`

Here is the call graph for this function:



4.27.3.33 `template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full<  
TN >::prune (const TN tol)`

4.27.3.34 `template<class TN> void linear_algebra::mat_asym_full< TN >::set (const long  
x, const long y, TN value)`

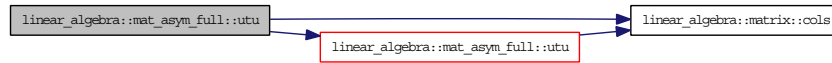
4.27.3.35 `template<class TN> long linear_algebra::mat_asym_full< TN >::size () const`

Here is the call graph for this function:



4.27.3.36 `template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_full< TN  
>::utu (const mat_full< TN > & U) const`

Here is the call graph for this function:

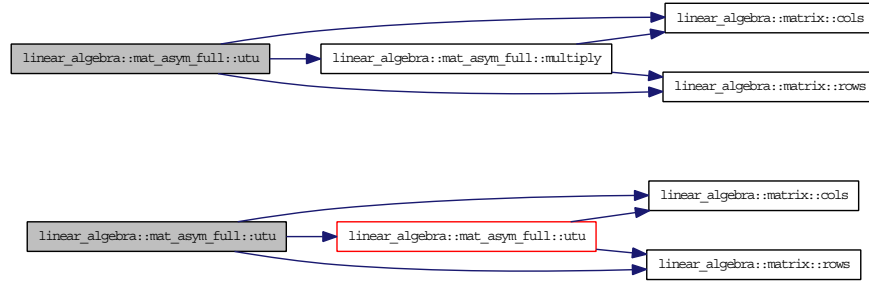


4.27.3.37 `template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full<  
TN >::utu (const mat_full< TN > & U, mat_asym_full< TN > & R) const`

Here is the call graph for this function:

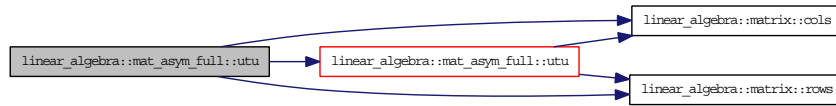
4.27.3.38 `template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_full< TN  
>::utu (const mat_asym_full< TN > & a) const`

Here is the call graph for this function:



4.27.3.39 `template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::utu (const mat_asym_full< TN > & a, mat_asym_full< TN > & R) const`

Here is the call graph for this function:



4.27.3.40 `template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_full< TN >::utu (const mat_asym_full< TN > & a, mat_full< TN > & I) const`

Here is the call graph for this function:

4.27.3.41 `template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::utu (const mat_asym_full< TN > & a, mat_asym_full< TN > & R, mat_full< TN > & I) const`

Here is the call graph for this function:

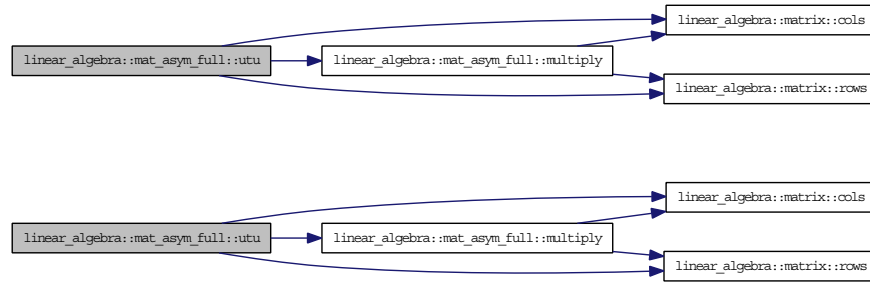
4.27.3.42 `template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_full< TN >::uut (const mat_full< TN > & U) const`

Here is the call graph for this function:

4.27.3.43 `template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::uut (const mat_full< TN > & U, mat_asym_full< TN > & R) const`

Here is the call graph for this function:

4.27.3.44 `template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_full< TN >::uut (const mat_full< TN > & U, mat_asym_full< TN > & R, mat_full< TN > & I) const`



Here is the call graph for this function:

4.27.3.45 `template<class T> mat_asym_full< T > & linear_algebra::mat_asym_full< T >::zero ()`

#### 4.27.4 Member Data Documentation

4.27.4.1 `template<class TN> refvector<TN> linear_algebra::mat_asym_full< TN >::_vals[private]`

The documentation for this class was generated from the following files:

- `include/BCR_CPP_LA/mat_asym_full.decl`
- `include/BCR_CPP_LA/mat_asym_full.h`

#### 4.28 `linear_algebra::mat_asym_sparse< TN >` Class Template Reference

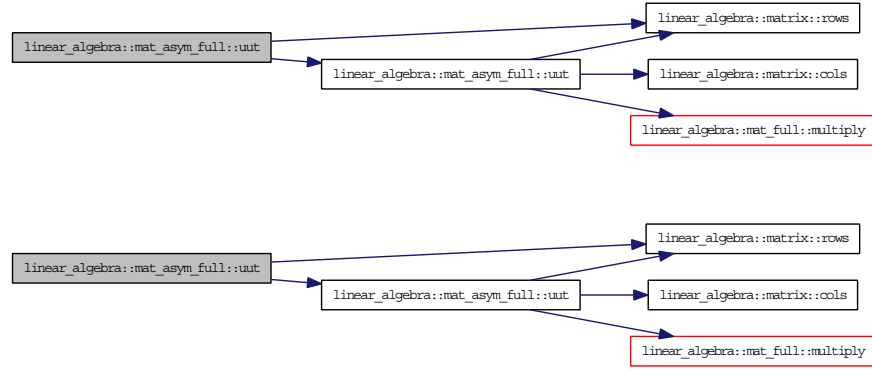
`mat_asym_sparse` packs sparse column matrices into a comfortable format.

Inheritance diagram for `linear_algebra::mat_asym_sparse< TN >`:

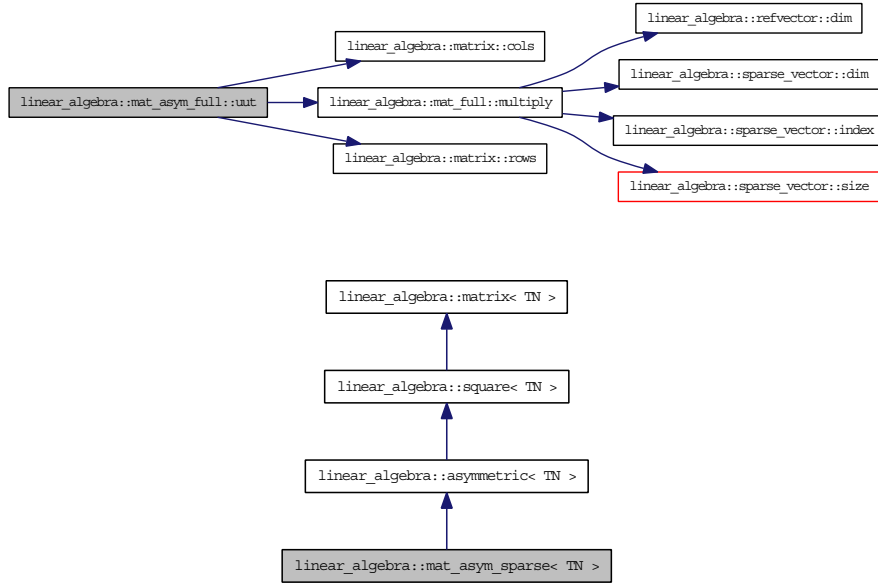
Collaboration diagram for `linear_algebra::mat_asym_sparse< TN >`:

##### Public Member Functions

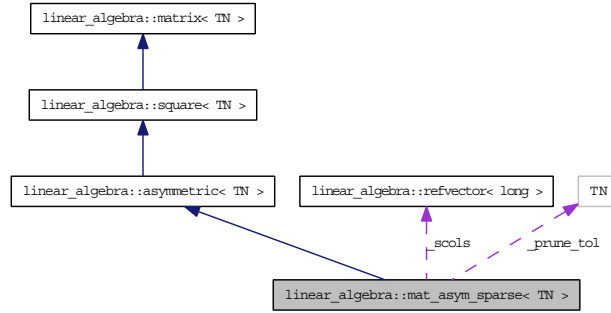
- `mat_asym_sparse ()`  
*Default constructor.*
- `mat_asym_sparse (long n)`  
*Constructor of dimension n with uninitialised values.*



- `mat_asym_sparse (long n, const vector< long > &scols, const vector< sparse_vector< TN > > &vals)`  
*Constructor of dimension n with initialisation of values.*
- `mat_asym_sparse (long n, const vector< long > &scols, const refvector< sparse_vector< TN > > &vals)`  
*Constructor of dimension n with initialisation of values.*
- `mat_asym_sparse (long n, const refvector< long > &scols, const refvector< sparse_vector< TN > > &vals)`  
*Constructor of dimension n with initialisation of values.*
- `mat_asym_sparse (long n, const refvector< long > &scols, const vector< sparse_vector< TN > > &vals)`  
*Constructor of dimension n with initialisation of values.*
- `mat_asym_spars (istream &IN)`  
*Constructor to read a stored matrix from a stream. Aimed at retrieving information from a binary save with `mat_asym_sparse::operator>>()`.*
- `mat_asym_sparse< TN > & operator= (const mat_asym_sparse< TN > &b)`
- `TN operator() (const long x, const long y) const`  
*Access operator.*
- `const sparse_vector< TN > operator[] (const long i) const`  
*Direct access operator.*
- `sparse_vector< TN > & operator[] (const long i)`  
*Direct access operator.*
- `void set (const long x, const long y, TN value)`  
*Set operation for safe assignments.*



- `mat_asym_sparse< TN > operator * (const TN &a) const`  
*Multiplies an antisymmetric sparse matrix with an object.*
- `mat_full< TN > operator * (mat_full< TN > A) const`  
*Multiply by a full matrix.*
- `mat_full< TN > & multiply (mat_full< TN > A, mat_full< TN > &r) const`  
*Multiply by a full matrix.*
- `refvector< TN > & multiply (const refvector< TN > &v, refvector< TN > &r) const`  
*Multiply a symmetric sparse matrix with a full vector into a full vector./test.*
- `refvector< TN > operator * (const refvector< TN > &v) const`  
*Multiply an antisymmetric sparse matrix with a full vector into a full vector./test.*
- `mat_asym_sparse< TN > & operator *= (const TN &x)`  
*Multiply by an object of TN into left side.*
- `mat_asym_sparse< TN > & multiply (long x, long y, TN a)`  
*Multiplies an element by a number.*
- `mat_full< TN > & multiply (const mat_asym_full< TN > &B, mat_full< TN > &r) const`  
*Multiply an antisymmetric sparse matrix with a full vector into a full vector./test.*
- `mat_full< TN > operator * (const mat_asym_full< TN > &B) const`  
*Multiply a symmetric sparse matrix with a full vector into a full vector./test.*



- `mat_sym_full< TN > aa_()` const  
*return a symmetric full matrix= $g \cdot g^*$ .*
- `mat_asym_sparse & operator/=(const TN &x)`  
*Divide by an object of TN into left side. This is very slow and not advisable.*
- `sparse_vector< TN > operator * (const sparse_vector< TN > &v) const`  
*Multiply a sparse symmetric matrix with a sparse vector.*
- `mat_sparse< TN > operator * (const mat_sparse< TN > &B) const`  
*Multiply a sparse symmetric matrix with a sparse matrix.*
- `mat_asym_sparse< TN > & operator+=(const mat_asym_sparse< TN > &B)`  
*Add a matrix into left object.*
- `mat_asym_sparse< TN > & operator-=(const mat_asym_sparse< TN > &B)`  
*Subtract a matrix into left object.*
- `bool is_col_nonzero (const long x, long &i) const`
- `bool is_nonzero (long x, long y, long &j, long &k) const`  
*Checks whether an access is non zero.*
- `bool add (long x, long y, const TN z)`  
*Adds a value.*
- `bool is_empty () const`
- `long size () const`  
*Returns size of non-zero entries in \_svals.*
- `void operator>> (ostream &OUT) const`  
*Output matrix in binary format into a stream. Aimed at file storage.*
- `mat_asym_sparse< TN > & prune (const TN x=0)`  
*Deletes all entries that are smaller in magnitude than x.*



- `TN max_element () const`  
*Finds a maximum element.*
- `bool display () const`  
*Simple display of matrix.*
- `vector< TN > extract_full_col (const long column, const long xmin, const long xmax) const`  
*Extracts a block from a column/row.*
- `vector< TN > extract_full_col (const long column, const long xmin=0) const`  
*Extracts a block from a column/row.*
- `mat_asym_full< TN > extract_full_asym_block (long xmin, long xmax) const`  
*Extracts a full symmetric block.*
- `mat_asym_full< TN > extract_full_asym_block (long xmin=0) const`  
*Extracts a full symmetric block.*
- `mat_sparse< TN > extract_sparse_block (long xmin, long xmax, long ymin, long ymax) const`  
*Extracts a block from anywhere in the matrix.*
- `sparse_vector< TN > extract_sparse_col (const long column, const long xmin=0) const`  
*Extracts a block from a column/row.*
- `sparse_vector< TN > extract_sparse_lower_col (const long column, const long xmin, const long xmax) const`  
*Extracts a sparse column up to the diagonal.*
- `sparse_vector< TN > extract_sparse_lower_col (const long column, const long xmin=0) const`  
*Extracts a block from a column/row.*
- `double density () const`  
*Return the density of non-zero elements.*
- `void clear ()`  
*Sets matrix to zero releasing all associated memory.*
- `mat_asym_sparse< TN > & zero ()`  
*Sets matrix to zero releasing all associated memory.*
- `long index (long i) const`  
*Returns the ith nonzero column number.*

- `mat_asym_full< TN > & utu (const mat_full< TN > &U, mat_asym_full< TN > &R) const`

*This is the unitary transform  $U^*AU$ .*

- `mat_asym_full< TN > utu (const mat_full< TN > &U) const`

*This is the unitary transform  $U^tAU$ .*

## Public Attributes

- `TN _prune_tol`

*Tolerance for multiplication with a sparse vector.*

## Private Attributes

- `refvector< long > _scols`

*Vector of column indices.*

- `refvector< sparse_vector< TN > > _svals`

*Vector of sparse columns.*

- `bool _empty`

*State of matrix.*

## Friends

- `class mat_sparse< TN >`

- `class mat_full< TN >`

### 4.28.1 Detailed Description

`template<class TN> class linear_algebra::mat_asym_sparse< TN >`

The sparse matrix is packed as a sparse vector of sparse vectors. Only the columns which are not empty are indexed and kept in memory. The sparse vector part takes care of the rows.

## 4.28.2 Constructor & Destructor Documentation

4.28.2.1 `template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse ()`

4.28.2.2 `template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse (long n)`

4.28.2.3 `template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse (long n, const vector< long > & scols, const vector< sparse_vector< TN > > & vals)`

### Parameters

*n* Dimension of matrix.

*srows* Vector of row indices.

*scols* Vector of column indices.

*vals* Vector of values.

Each value has an associated row and column index in *srows* and *scols*. Therefore they all have the same size. There is no integrity check.

4.28.2.4 `template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse (long n, const vector< long > & scols, const refvector< sparse_vector< TN > > & vals)`

*n* Dimension of matrix.

*srows* Vector of row indices.

*scols* Vector of column indices.

*vals* reference counted Vector of values.

Each value has an associated row and column index in *srows* and *scols*. Therefore they all have the same size. There is no integrity check.

4.28.2.5 `template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse (long n, const refvector< long > & scols, const refvector< sparse_vector< TN > > & vals)`

textbf Parameters:

*n* Dimension of matrix.

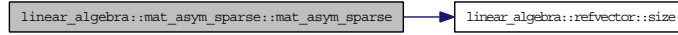
*srows* Vector of row indices.

*scols* Reference counted vector of column indices.

*vals* Reference counted vector of values.

Each value has an associated row and column index in *srows* and *scols*. Therefore they all have the same size. There is no integrity check.

Here is the call graph for this function:



4.28.2.6 `template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse`  
 (long *n*, const refvector< long > & *scols*, const vector< sparse\_vector< TN > > & *vals*)

#### Parameters:

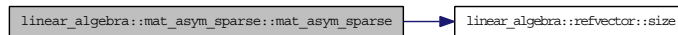
*n* Dimension of matrix.

*srows* Vector of row indices.

*scols*]Reference counted vector of column indices.

*vals* Vector of values. Each value has an associated row and column index in *srows* and *scols*. Therefore they all have the same size. There is no integrity check.

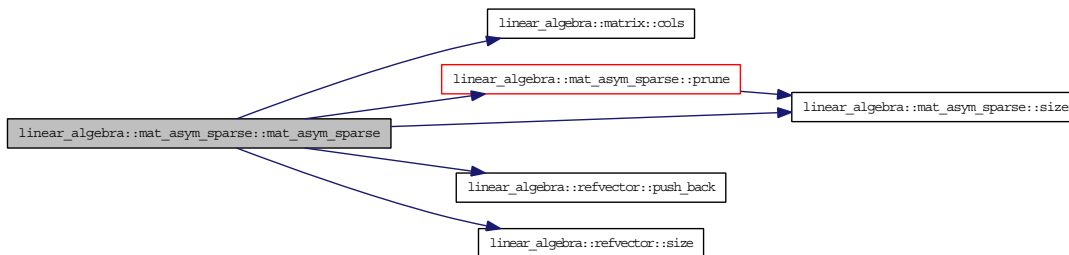
Here is the call graph for this function:



4.28.2.7 `template<class TN> linear_algebra::mat_asym_sparse< TN >::mat_asym_sparse`  
 (istream & *IN*)

The resultant vector is pruned to default pruning tolerance = 0. This should be completely superfluous but in case a different program not using this class produces the matrix as input we prune.

Here is the call graph for this function:



### 4.28.3 Member Function Documentation

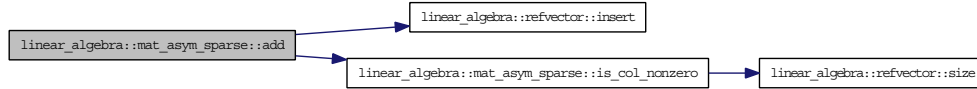
4.28.3.1 `template<class TN> mat_sym_full< TN > linear_algebra::mat_asym_sparse< TN`  
`>::aa_()` const

4.28.3.2 `template<class TN> bool linear_algebra::mat_asym_sparse< TN >::add (long x, long y, const TN z)`

**Parameters:**

*z* is added to (*x*,*y*). If the value is zero the associated vectors are expanded.

Here is the call graph for this function:



4.28.3.3 `template<class TN> void linear_algebra::mat_asym_sparse< TN >::clear ()`

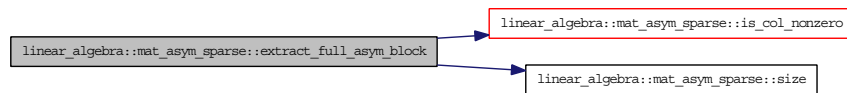
4.28.3.4 `template<class TN> double linear_algebra::mat_asym_sparse< TN >::density () const`

4.28.3.5 `template<class TN> bool linear_algebra::mat_asym_sparse< TN >::display () const`

4.28.3.6 `template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_sparse< TN >::extract_full_asym_block (long xmin = 0) const`

4.28.3.7 `template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_sparse< TN >::extract_full_asym_block (long xmin, long xmax) const`

Here is the call graph for this function:



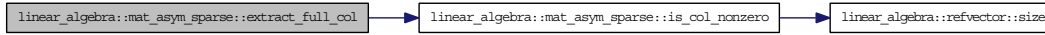
4.28.3.8 `template<class TN> vector< TN > linear_algebra::mat_asym_sparse< TN >::extract_full_col (const long column, const long xmin = 0) const`

Here is the call graph for this function:



4.28.3.9 `template<class TN> vector< TN > linear_algebra::mat_asym_sparse< TN >::extract_full_col (const long column, const long xmin, const long xmax) const`

Here is the call graph for this function:

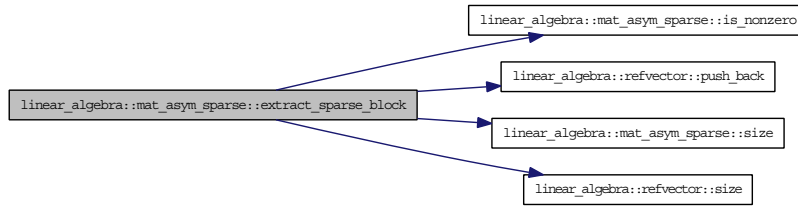


4.28.3.10 `template<class TN> mat_sparse< TN > linear_algebra::mat_asym_sparse< TN >::extract_sparse_block (long xmin, long xmax, long ymin, long ymax) const`

### Test

should work, but ...

Here is the call graph for this function:



4.28.3.11 `template<class TN> sparse_vector< TN > linear_algebra::mat_asym_sparse< TN >::extract_sparse_col (const long column, const long xmin = 0) const`

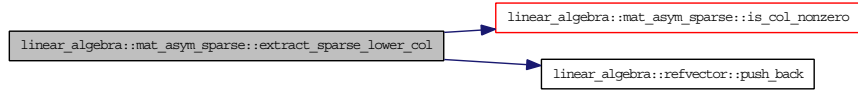
4.28.3.12 `template<class TN> sparse_vector< TN > linear_algebra::mat_asym_sparse< TN >::extract_sparse_lower_col (const long column, const long xmin = 0) const`

Here is the call graph for this function:



4.28.3.13 `template<class TN> sparse_vector< TN > linear_algebra::mat_sym_sparse< TN >::extract_sparse_lower_col (const long column, const long xmin, const long xmax) const`

Here is the call graph for this function:



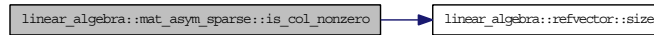
4.28.3.14 `template<class TN> long linear_algebra::mat_asym_sparse< TN >::index (long i) const`

Here is the call graph for this function:



4.28.3.15 `template<class TN> bool linear_algebra::mat_asym_sparse< TN >::is_col_nonzero (const long x, long & i) const`

Here is the call graph for this function:



4.28.3.16 `template<class TN> bool linear_algebra::mat_asym_sparse< TN >::is_empty () const`

4.28.3.17 `template<class TN> bool linear_algebra::mat_asym_sparse< TN >::is_nonzero (long x, long y, long & j, long & k) const`

### Parameters:

*j* holds the position of (x,y) in `_vals`. If (x,y) is 0. An insertion would happen here.

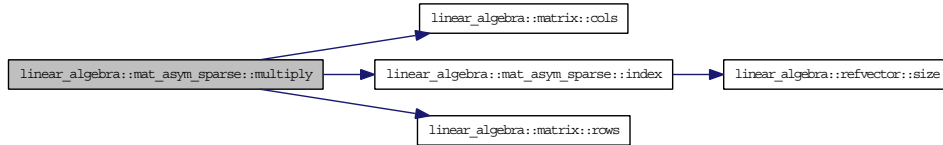
4.28.3.18 `template<class TN> TN linear_algebra::mat_asym_sparse< TN >::max_element () const`

Here is the call graph for this function:



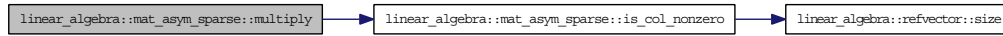
4.28.3.19 `template<class TN> mat_full< TN > & linear_algebra::mat_asym_sparse< TN >::multiply (const mat_asym_full< TN > & B, mat_full< TN > & r) const`

Here is the call graph for this function:



4.28.3.20 `template<class TN> mat_asym_sparse< TN > & linear_algebra::mat_asym_sparse< TN >::multiply (long x, long y, TN a)`

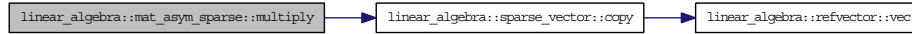
Here is the call graph for this function:



4.28.3.21 `template<class TN> refvector< TN > & linear_algebra::mat_asym_sparse< TN >::multiply (const refvector< TN > & v, refvector< TN > & r) const`

Here is the call graph for this function:



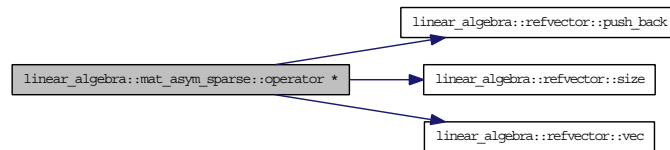


4.28.3.22 `template<class TN> mat_full< TN > & linear_algebra::mat_asym_sparse< TN >::multiply (mat_full< TN > A, mat_full< TN > & r) const`

## Test

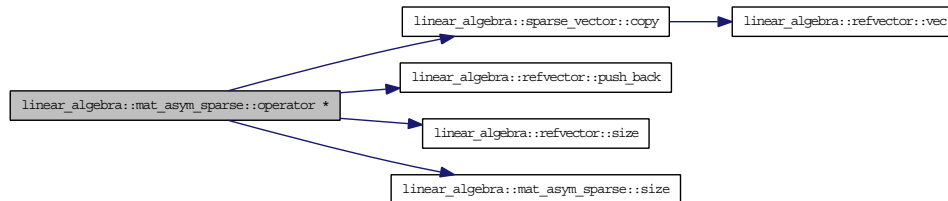
4.28.3.23 `template<class TN> mat_sparse< TN > linear_algebra::mat_asym_sparse< TN >::operator * (const mat_sparse< TN > & B) const`

Here is the call graph for this function:



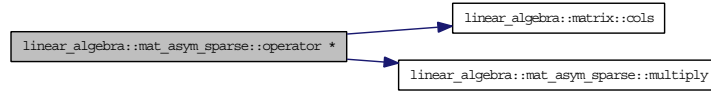
4.28.3.24 `template<class TN> sparse_vector< TN > linear_algebra::mat_asym_sparse< TN >::operator * (const sparse_vector< TN > & v) const`

Here is the call graph for this function:



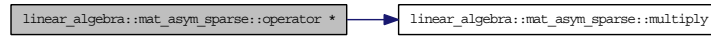
4.28.3.25 `template<class TN> mat_full< TN > linear_algebra::mat_asym_sparse< TN >::operator * (const mat_asym_full< TN > & B) const`

Here is the call graph for this function:



4.28.3.26 `template<class TN> refvector< TN > linear_algebra::mat_asym_sparse< TN >::operator * (const refvector< TN > & v) const`

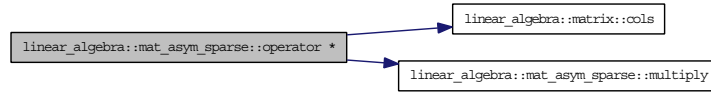
Here is the call graph for this function:



4.28.3.27 `template<class TN> mat_full< TN > linear_algebra::mat_asym_sparse< TN >::operator * (mat_full< TN > A) const`

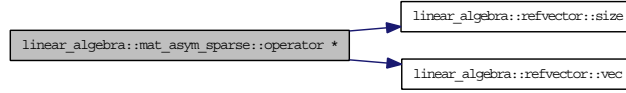
### Test

Here is the call graph for this function:



4.28.3.28 `template<class TN> mat_asym_sparse< TN > linear_algebra::mat_asym_sparse< TN >::operator * (const TN & a) const`

Here is the call graph for this function:



4.28.3.29 `template<class TN> mat_asym_sparse< TN > &  
linear_algebra::mat_asym_sparse< TN >::operator *= (const TN & x)`

4.28.3.30 `template<class TN> TN linear_algebra::mat_asym_sparse< TN >::operator()  
(const long x, const long y) const[virtual]`

Implements `linear_algebra::matrix< TN >`.

Here is the call graph for this function:



4.28.3.31 `template<class TN> mat_asym_sparse<TN>& linear_algebra::mat_asym_sparse<  
TN >::operator+= (const mat_asym_sparse< TN > & B)`

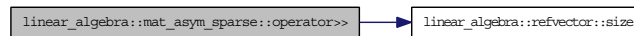
4.28.3.32 `template<class TN> mat_asym_sparse<TN>& linear_algebra::mat_asym_sparse<  
TN >::operator-= (const mat_asym_sparse< TN > & B)`

4.28.3.33 `template<class TN> mat_asym_sparse< TN > &  
linear_algebra::mat_asym_sparse< TN >::operator/= (const TN & x)`

4.28.3.34 `template<class TN> mat_asym_sparse< TN > &  
linear_algebra::mat_asym_sparse< TN >::operator= (const mat_asym_sparse< TN > & b)`

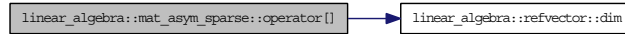
4.28.3.35 `template<class TN> void linear_algebra::mat_asym_sparse< TN >::operator>>  
(ostream & OUT) const`

Here is the call graph for this function:



4.28.3.36 `template<class TN> sparse_vector< TN > & linear_algebra::mat_asym_sparse<  
TN >::operator[] (const long i)`

Here is the call graph for this function:



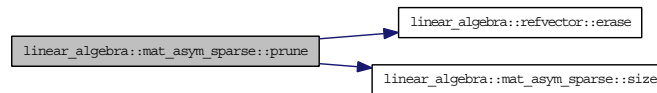
4.28.3.37 `template<class TN> const sparse_vector< TN >  
linear_algebra::mat_asym_sparse< TN >::operator[] (const long i) const`

Here is the call graph for this function:



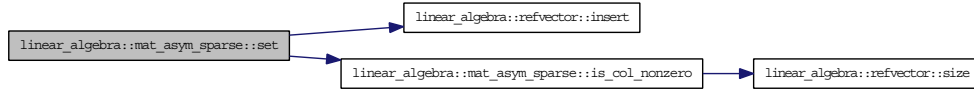
4.28.3.38 `template<class TN> mat_asym_sparse< TN > &  
linear_algebra::mat_asym_sparse< TN >::prune (const TN x = 0)`

Here is the call graph for this function:



4.28.3.39 `template<class TN> void linear_algebra::mat_asym_sparse< TN >::set (const  
long x, const long y, TN value)`

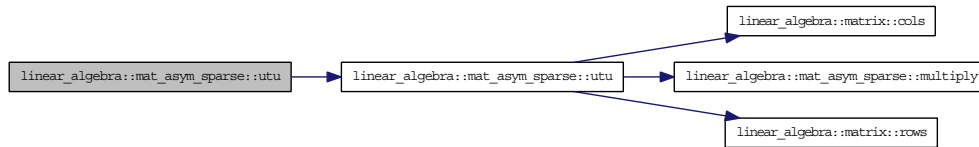
Here is the call graph for this function:



4.28.3.40 `template<class TN> long linear_algebra::mat_asym_sparse< TN >::size () const`

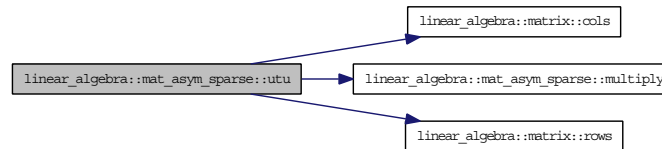
4.28.4.41 `template<class TN> mat_asym_full< TN > linear_algebra::mat_asym_sparse< TN >::utu (const mat_full< TN > & U) const`

Here is the call graph for this function:



4.28.3.42 `template<class TN> mat_asym_full< TN > & linear_algebra::mat_asym_sparse< TN >::utu (const mat_full< TN > & U, mat_asym_full< TN > & R) const`

Here is the call graph for this function:



4.28.3.43 `template<class TN> mat_asym_sparse< TN > &  
linear_algebra::mat_asym_sparse< TN >::zero ()`

#### 4.28.4 Friends and Related Function Documentation

4.28.4.1 `template<class TN> friend class mat_full< TN >[friend]`

4.28.4.2 `template<class TN> friend class mat_sparse< TN >[friend]`

#### 4.28.5 Member Data Documentation

4.28.5.1 `template<class TN> bool linear_algebra::mat_asym_sparse< TN  
>::_empty[private]`

4.28.5.2 `template<class TN> TN linear_algebra::mat_asym_sparse< TN >::_prune_tol`

4.28.5.3 `template<class TN> refvector<long> linear_algebra::mat_asym_sparse< TN  
>::_scols[private]`

4.28.5.4 `template<class TN> refvector<sparse_vector<TN> >  
linear_algebra::mat_asym_sparse< TN >::_svals[private]`

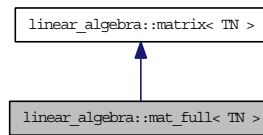
The documentation for this class was generated from the following files:

- `include/BCR_CPP_LA/mat_asym_sparse.decl`
- `include/BCR_CPP_LA/mat_asym_sparse.h`

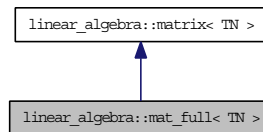
### 4.29 `linear_algebra::mat_full< TN >` Class Template Reference

Class of full column matrices.

Inheritance diagram for `linear_algebra::mat_full< TN >`:



Collaboration diagram for `linear_algebra::mat_full< TN >`:



## Public Member Functions

- `mat_full (long n, long m)`  
*Constructor of  $n$  columns and  $m$  rows with uninitialised values.*
- `mat_full (long n, long m, const std::vector< std::vector< TN > > &vals)`  
*Constructor of  $n$  columns and  $m$  rows with initialisation of values.*
- `mat_full (long n, long m, const TN *vals[])`  
*Constructor of  $n$  columns and  $m$  rows with initialisation of values. You had better be sure there is no buffer overrun.*
- `mat_full (long n, long m, const refvector< std::vector< TN > > &vals)`  
*Constructor of  $n$  columns and  $m$  rows with initialisation of values.*
- `mat_full (long n, long m, const refvector< refvector< TN > > &vals)`  
*Constructor of  $n$  columns and  $m$  rows with initialisation of values.*
- `mat_full (long n, long m, const std::vector< refvector< TN > > &vals)`  
*Constructor of  $n$  columns and  $m$  rows with initialisation of values.*
- `mat_full ()`
- `mat_full (std::istream &IN)`  
*Constructor to read a stored matrix from a stream. Aimed at retrieving information from a binary save with `mat_full::operator>>()`.*
- `~mat_full ()`

- `mat_full< TN > & operator= (mat_full< TN > &a)`  
*Assignment operator for non const objects.*
- `mat_full< TN > & operator= (const mat_full< TN > &a)`  
*Assignment operator for non const objects.*
- `bool operator== (const mat_full< TN > &b) const`  
*Compares two full matrices.*
- `bool same (const mat_full< TN > &b) const`
- `refvector< TN > & operator[] (const long x)`  
*Return a column through direct access. No integrity checked.*
- `const refvector< TN > & operator[] (const long x) const`  
*Return a column through direct access. No integrity checked.*
- `const TN & operator() (const long x, const long y) const`  
*Return value at (x,y).*
- `TN & operator() (const long x, const long y)`  
*Return value at (x,y).*
- `refvector< TN > & multiply (const sparse_vector< TN > &a, refvector< TN > &r)`  
`const`  
*Multiply a full matrix by a sparse vector.*
- `refvector< TN > operator * (const sparse_vector< TN > &a) const`  
*Multiply a full matrix by a sparse vector.*
- `mat_full< TN > operator * (const TN &a) const`  
*Multiply a full matrix by a number.*
- `mat_full< TN > & multiply (const TN &a, mat_full< TN > &r) const`  
*Multiply a full matrix by a number.*
- `mat_full< TN > & operator *= (const TN &x)`  
*Multiply by an object of TN into left side.*
- `mat_full< TN > & operator /= (const TN &x)`  
*Divide by an object of TN into left side. This is very slow and not advisable.*
- `mat_full< TN > & operator *= (const mat_asym_full< TN > &b)`  
*Multiply an asymmetric, full matrix from the left into this full matrix.*
- `mat_full< TN > & operator *= (const mat_sym_full< TN > &b)`  
*Multiply a symmetric, full matrix from the left into this full matrix.*



- `mat_full< TN > & multiply (const mat_full< TN > &b, mat_full< TN > &r) const`  
*Multiply two full matrices.*
- `mat_full< TN > operator * (const mat_full< TN > &b) const`  
*Multiply two full matrices.*
- `mat_full< TN > & multiply_t_a (const mat_full< TN > &b, mat_full< TN > &r) const`  
*Multiply the transpose of a full matrix with a full matrix.*
- `mat_full< TN > multiply_t_a (const mat_full< TN > &b) const`  
*Multiply the transpose of a full matrix with a full matrix.*
- `mat_full< TN > & multiply_a_t (const mat_full< TN > &b, mat_full< TN > &r) const`  
*Multiply full matrix with the transpose of a full matrix.*
- `mat_full< TN > multiply_a_t (const mat_full< TN > &b) const`  
*Multiply full matrix with the transpose of a full matrix.*
- `mat_full< TN > & multiply (const mat_sym_full< TN > &b, mat_full< TN > &r) const`  
*Multiply a full matrix by a symmetric full matrix.*
- `mat_full< TN > operator * (const mat_sym_full< TN > &b) const`  
*Multiply a full matrix by a symmetric full matrix.*
- `mat_full< TN > & multiply (const mat_asym_full< TN > &b, mat_full< TN > &r) const`  
*Multiply a full matrix by an antisymmetric full matrix.*
- `mat_full< TN > operator * (const mat_asym_full< TN > &b) const`  
*Multiply a full matrix by an antisymmetric full matrix.*
- `mat_full< TN > & multiply (const mat_sym_sparse< TN > &b, mat_full< TN > &r) const`  
*Multiply a full matrix by a symmetric sparse matrix.*
- `mat_full< TN > operator * (const mat_sym_sparse< TN > &b) const`  
*Multiply a full matrix by a symmetric sparse matrix.*
- `template<class TN2> refvector< TN > & multiply (const refvector< TN2 > &a, refvector< TN > &r) const`  
*Multiply a full matrix by a full vector.*

- `template<class TN2> refvector< TN > operator * (const refvector< TN2 > &a)  
const`  
*Multiply a full matrix by a full vector.*
- `mat_full< TN > & operator *= (const mat_full< TN > &b)`  
*Multiplies the **left** hand side from the { left } by the **right** hand side.*
- `mat_full< TN > & operator += (const mat_full< TN > &b)`  
*Add two matrices.*
- `mat_full< TN > & operator += (const mat_sym_full< TN > &b)`  
*Add two matrices.*
- `mat_full< TN > & operator -= (const mat_full< TN > &b)`  
*Subtract two matrices.*
- `mat_full< TN > & operator -= (const mat_sym_full< TN > &b)`  
*Add two matrices.*
- `mat_full< TN > operator+ (const mat_full< TN > &b) const`  
*Add two matrices.*
- `mat_full< TN > operator- (const mat_full< TN > &b) const`  
*Subtract two matrices.*
- `mat_full< TN > & copy (const mat_full< TN > &a)`  
*Copy matrix with full depth.*
- `mat_full< TN > & copy (const mat_full< TN > &a, const long i)`  
*Copy matrix with depth i.*
- `mat_full< TN > & copy (mat_full< TN > &a, const long i)`  
*Copy matrix with depth i.*
- `matrix< TN > & copy (const matrix< TN > &a)`  
*Copy matrix with full depth.*
- `matrix< TN > & copy (const matrix< TN > &a, long i)`  
*Copy matrix with depth i.*
- `matrix< TN > & copy (matrix< TN > &a, long i)`  
*Copy matrix with depth i.*
- `bool add(const long x, const long y, const TN z)`  
*Adds a value.*

- `void set (const long x, const long y, TN z)`  
*Set a value.*
- `bool display () const`  
*Simple display of matrix.*
- `bool display (std::ostream &os) const`  
*Simple display of matrix.*
- `mat_sym_full< TN > sym_upper_triangle () const`  
*Returns a mat\_sym\_full object of the upper triangle of a full matrix.*
- `mat_sym_full< TN > & sym_upper_triangle (mat_sym_full< TN > &r) const`  
*Returns a mat\_sym\_full object of the upper triangle of a full matrix.*
- `mat_asym_full< TN > asym_upper_triangle () const`  
*Returns a mat\_asym\_full object of the upper triangle of a full matrix.*
- `mat_sym_full< TN > sym_lower_triangle () const`  
*Returns a mat\_sym\_full object of the lower triangle of a full matrix.*
- `mat_sym_full< TN > & sym_lower_triangle (mat_sym_full< TN > &r) const`  
*Returns a mat\_sym\_full object of the lower triangle of a full matrix.*
- `mat_full< double > & gauss_elim (mat_full< TN > &T)`  
*Do half a householder and then do gaussian elimination for a left inverse.*
- `mat_full< TN > transpose () const`  
*Transposes the full matrix.*
- `mat_full< TN > & transpose (mat_full< TN > &t) const`  
*Transposes the full matrix.*
- `mat_full< TN > & grow (long n, long m)`  
*Extend the matrix by n columns and m rows.*
- `void clear ()`  
*Release the associated memory.*
- `mat_full< TN > & prune (TN a)`  
*Prune the matrix of negligible values.*
- `mat_full< TN > & zero ()`  
*Set the matrix to zero.*
- `mat_full< TN > & one ()`  
*Set the matrix to the identity.*

## Protected Attributes

- `refvector< refvector< TN > > _scols`  
    *< Vector of columns.*

### 4.29.1 Detailed Description

`template<class TN> class linear_algebra::mat_full< TN >`

'column' specifies that the values are ordered for speedy retrieval per column.

### 4.29.2 Constructor & Destructor Documentation

4.29.2.1 `template<class TN> linear_algebra::mat_full< TN >::mat_full (long n, long m)`

'column' specifies that the values are ordered for speedy retrieval per column.

4.29.2.2 `template<class TN> linear_algebra::mat_full< TN >::mat_full (long n, long m,  
const std::vector< std::vector< TN > > & vals)`

4.29.2.3 `template<class TN> linear_algebra::mat_full< TN >::mat_full (long n, long m,  
const TN * vals[])`

#### Parameters:

*n* Columns of matrix.

*m* Rows of matrix.

*vals* array of columns.

Each column is a full array.

Each value has an associated row and column index in *srows* and *scols*. Therefore they all have the same size.

4.29.2.4 `template<class TN> linear_algebra::mat_full< TN >::mat_full (long n, long m,  
const refvector< std::vector< TN > > & vals)`

4.29.2.5 `template<class TN> linear_algebra::mat_full< TN >::mat_full (long n, long m,  
const refvector< refvector< TN > > & vals)`

#### Parameters:

*n* Dimension of matrix.

*scols* Reference counted vector of column indices.

*vals* Reference counted vector of columns.

Each column is a full vector.

Each value has an associated row and column index in *srows* and *scols*. Therefore they all have the same size.

4.29.2.6 `template<class TN> linear_algebra::mat_full< TN >::mat_full (long n, long m,  
const std::vector< refvector< TN > > & vals)`

4.29.2.7 `template<class TN> linear_algebra::mat_full< TN >::mat_full ()`

4.29.2.8 `template<class TN> linear_algebra::mat_full< TN >::mat_full (std::istream & IN)`

4.29.4.9 `template<class TN> linear_algebra::mat_full< TN >::~~mat_full ()`

### 4.29.3 Member Function Documentation

4.29.3.1 `template<class TN> bool linear_algebra::mat_full< TN >::add (const long x,  
const long y, const TN z)`

#### Parameters:

*z* is added to (*x*,*y*). If the value is zero the associated vectors are expanded.

4.29.3.2 `template<class TN> mat_asym_full< TN > linear_algebra::mat_full< TN  
>::asym_upper_triangle () const`

4.29.3.3 `template<class TN> void linear_algebra::mat_full< TN >::clear ()`

4.29.3.4 `template<class TN> matrix< TN > & linear_algebra::mat_full< TN >::copy  
(matrix< TN > & a, long i)`

Here is the call graph for this function:



4.29.3.5 `template<class TN> matrix< TN > & linear_algebra::mat_full< TN >::copy  
(const matrix< TN > & a, long i)`

Here is the call graph for this function:



4.29.3.6 `template<class TN> matrix< TN > & linear_algebra::mat_full< TN >::copy  
(const matrix< TN > & a)`

Here is the call graph for this function:



4.29.3.7 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::copy  
(mat_full< TN > & a, const long i)`

4.29.3.8 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::copy  
(const mat_full< TN > & a, const long i)`

4.29.3.9 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::copy  
(const mat_full< TN > & a)`

4.29.3.10 `template<class TN> bool linear_algebra::mat_full< TN >::display (std::ostream  
& os) const`

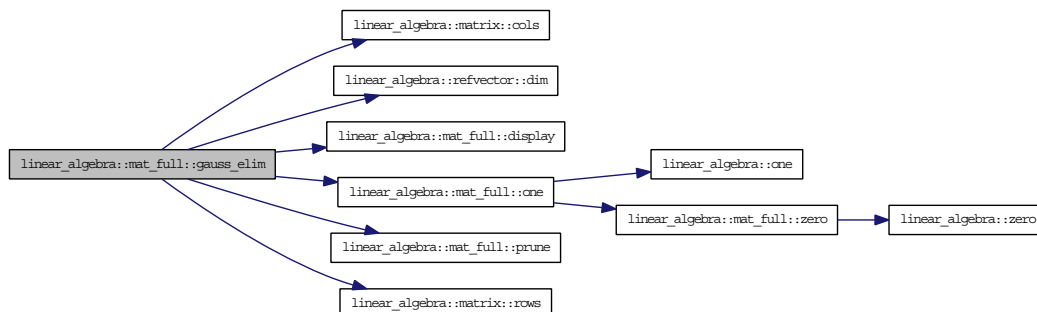
4.29.3.11 `template<class TN> bool linear_algebra::mat_full< TN >::display () const`

4.29.3.12 `template<class TN> mat_full< double > & linear_algebra::mat_full< TN  
>::gauss_elim (mat_full< TN > & T)`

## Todo

Throw exceptions for ill-conditioned matrices

Here is the call graph for this function:



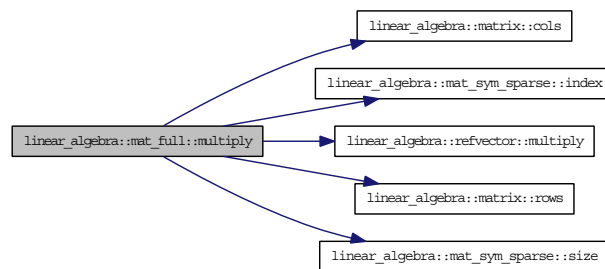
4.29.3.13 `template<class T> mat_full< T > & linear_algebra::mat_full< T >::grow (long n, long m)`

4.29.3.14 `template<class TN> template<class TN2> refvector< TN > & linear_algebra::mat_full< TN >::multiply (const refvector< TN2 > & a, refvector< TN > & r) const`

A little care needs to be taken here because any vector for which  $TN * TN2$  is defined this product is also defined.

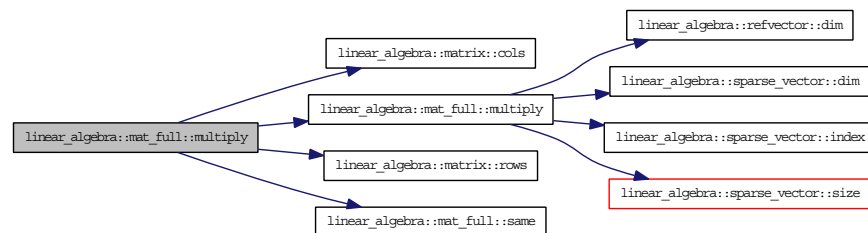
4.29.3.15 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply (const mat_sym_sparse< TN > & b, mat_full< TN > & r) const`

Here is the call graph for this function:



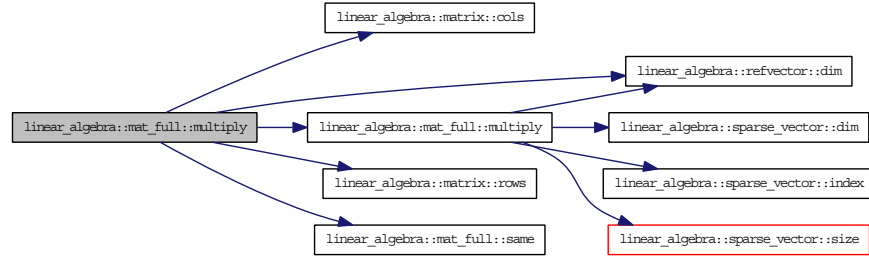
4.29.3.16 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply (const mat_asym_full< TN > & b, mat_full< TN > & r) const`

Here is the call graph for this function:



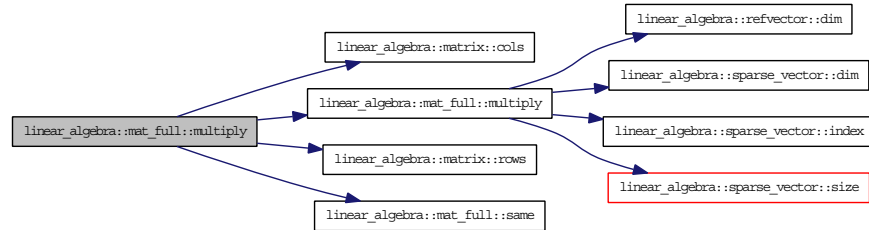
4.29.3.17 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply (const mat_sym_full< TN > &`

Here is the call graph for this function:



4.29.3.18 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply (const mat_full< TN > & b , mat_full< TN > & r) const`

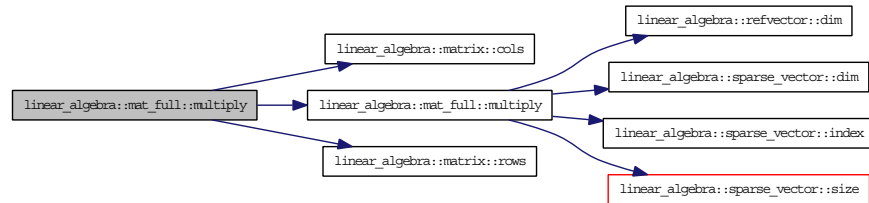
Here is the call graph for this function:



4.29.3.19 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply (const TN & a, mat_full< TN > & r) const`

**Test**

Here is the call graph for this function:

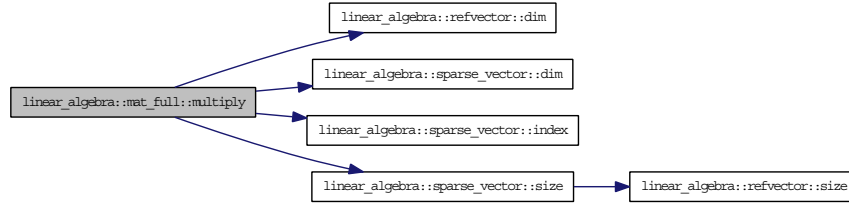


3.29.3.20 `template<class TN> refvector< TN > & linear_algebra::mat_full< TN >::multiply (const sparse_vector< TN > & a, refvector< TN > & r) const`

**Test**

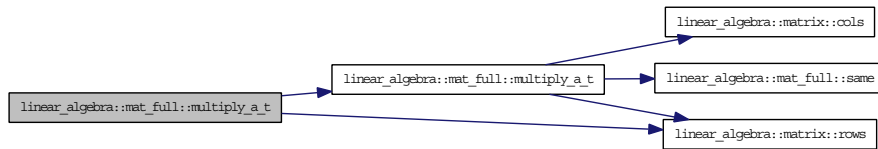
Here is the call graph for this function:





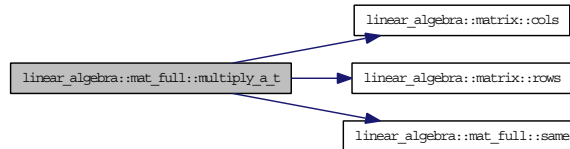
4.29.3.21 `template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::multiply_a_t (const mat_full< TN > & b) const`

Here is the call graph for this function:



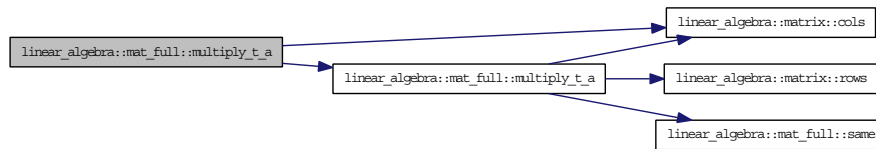
4.29.3.22 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply_a_t (const mat_full< TN > & b, mat_full< TN > & r) const`

Here is the call graph for this function:



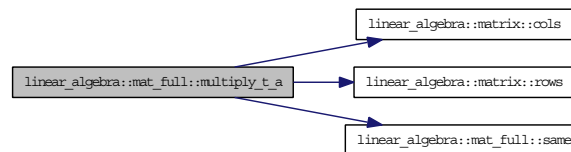
4.29.3.23 `template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::multiply_t_a (const mat_full< TN > & b) const`

Here is the call graph for this function:



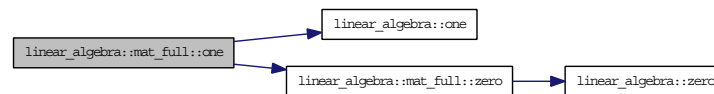
4.29.3.24 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::multiply_t_a (const mat_full< TN > & b, mat_full< TN > & r) const`

Here is the call graph for this function:



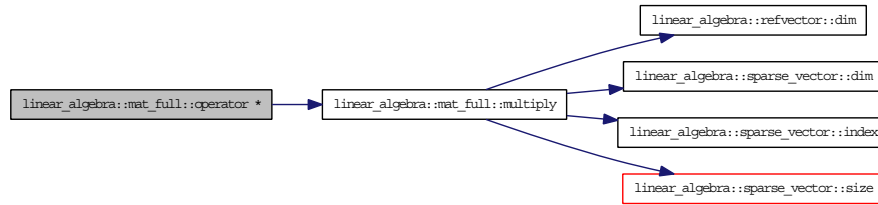
4.29.3.25 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::one ()`

Here is the call graph for this function:



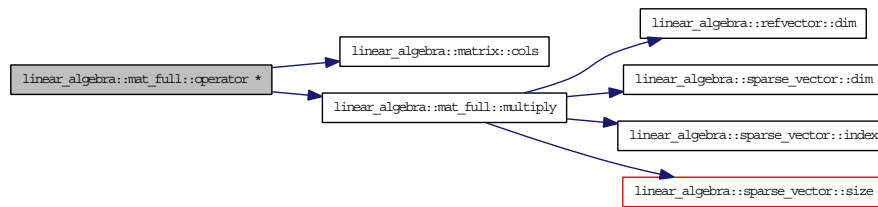
4.29.3.26 `template<class TN> template<class TN2> refvector< TN > linear_algebra::mat_full< TN >::operator * (const refvector< TN2 > & a) const`

Here is the call graph for this function:



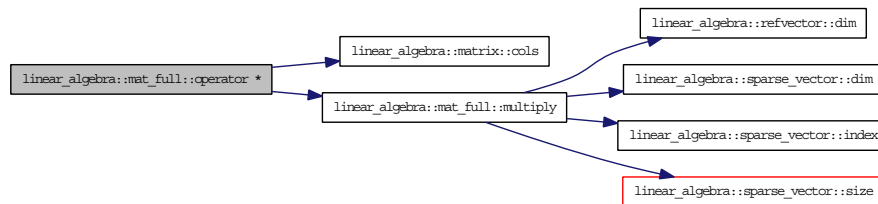
4.29.3.27 `template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator *  
(const mat_sym_sparse< TN > & b) const`

Here is the call graph for this function:



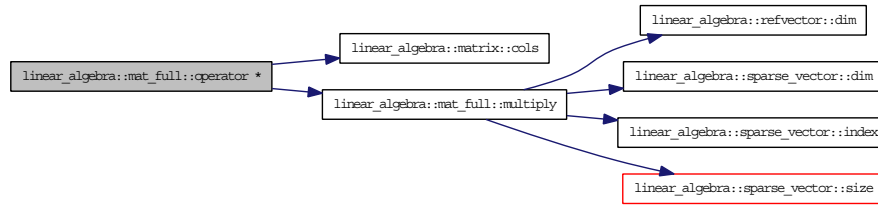
4.29.3.28 `template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator *  
(const mat_asym_full< TN > & b) const`

Here is the call graph for this function:



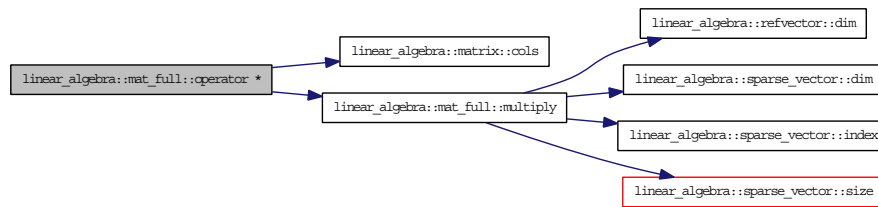
4.29.3.29 `template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator *  
(const mat_sym_full< TN > & b) const`

Here is the call graph for this function:



4.29.3.30 `template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator *  
(const mat_full< TN > & b) const`

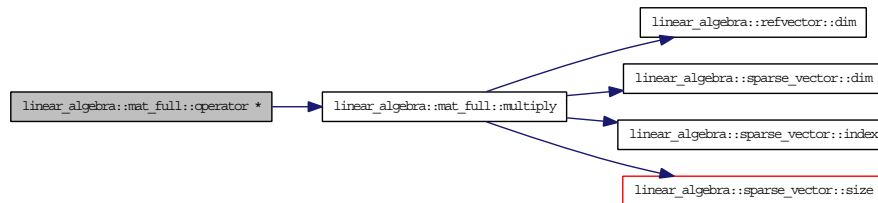
Here is the call graph for this function:



4.29.3.31 `template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator *  
(const TN & a) const`

### Test

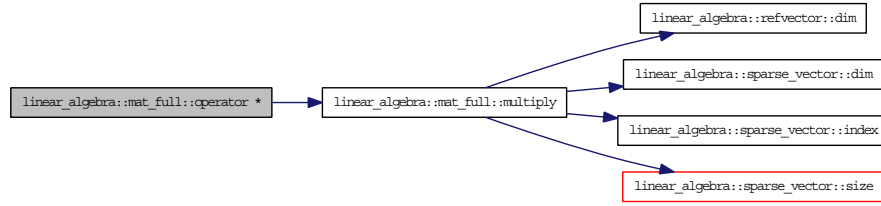
Here is the call graph for this function:



4.29.3.32 `template<class TN> refvector< TN > linear_algebra::mat_full< TN >::operator  
* (const sparse_vector< TN > & a) const`

### Test

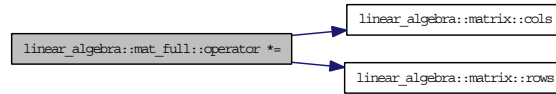
Here is the call graph for this function:



4.29.3.33 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator *= (const mat_full< TN > & b)`

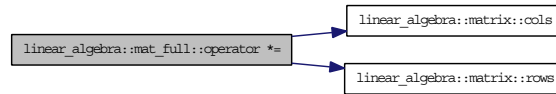
4.29.3.34 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator *= (const mat_sym_full< TN > & b)`

Here is the call graph for this function:



4.29.3.35 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator *= (const mat_asym_full< TN > & b)`

Here is the call graph for this function:



4.29.3.36 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator *= (const TN & x)`

4.29.3.37 `template<class TN> TN & linear_algebra::mat_full< TN >::operator() (const long x, const long y)`

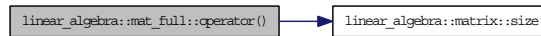
Here is the call graph for this function:



4.29.3.38 `template<class TN> const TN & linear_algebra::mat_full< TN >::operator()  
(const long x, const long y) const[virtual]`

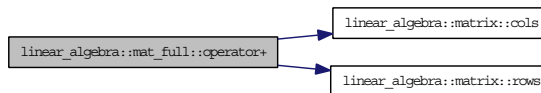
Implements `linear_algebra::matrix< TN >`.

Here is the call graph for this function:



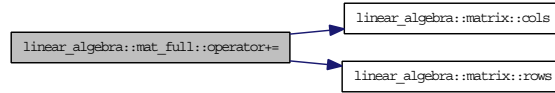
4.29.3.39 `template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator+  
(const mat_full< TN > & b) const`

Here is the call graph for this function:



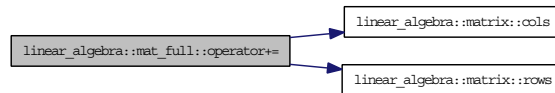
4.29.3.40 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN  
>::operator+= (const mat_sym_full< TN > & b)`

Here is the call graph for this function:



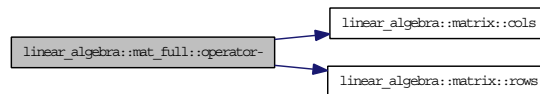
4.29.3.41 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator+= (const mat_full< TN > & b)`

Here is the call graph for this function:



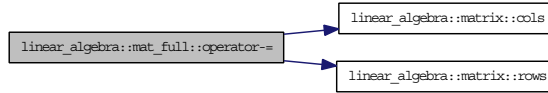
4.29.3.42 `template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::operator-(const < TN > & b) const`

Here is the call graph for this function:



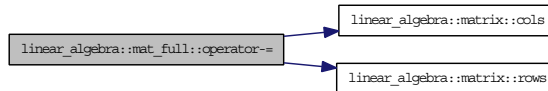
4.29.3.43 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator-= (const mat_sym_full< TN > & b)`

Here is the call graph for this function:



4.29.3.44 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator-= (const mat_full< TN > & b)`

Here is the call graph for this function:



4.29.3.45 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator/= (const TN & x)`

4.29.3.46 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator= (const mat_full< TN > & a)`

4.29.3.47 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::operator= (mat_full< TN > & a)`

4.29.3.48 `template<class TN> bool linear_algebra::mat_full< TN >::operator== (const mat_full< TN > & b) const`

4.29.3.49 `template<class TN> const refvector< TN > & linear_algebra::mat_full< TN >::operator[] (const long x) const`

4.29.3.50 `template<class TN> refvector< TN > & linear_algebra::mat_full< TN >::operator[] (const long x)`

4.29.3.51 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::prune TN a)`

4.29.3.52 `template<class TN> bool linear_algebra::mat_full< TN >::same (const mat_full< TN > & b) const`

4.29.3.53 `template<class TN> void linear_algebra::mat_full< TN >::set (const long x, const long y, TN z)`

4.29.3.54 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_full< TN >::sym_lower_triangle (mat_sym_full< TN > & r) const`



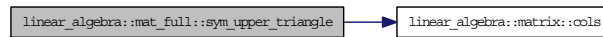
Here is the call graph for this function:



4.29.3.55 `template<class TN> mat_sym_full< TN > linear_algebra::mat_full< TN >::sym_lower_triangle () const`

4.29.3.56 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_full< TN >::sym_upper_triangle (mat_sym_full< TN > & r) const`

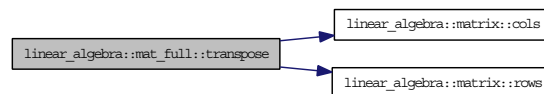
Here is the call graph for this function:



4.29.3.57 `template<class TN> mat_sym_full< TN > linear_algebra::mat_full< TN >::sym_upper_triangle () const`

4.29.3.58 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::transpose (mat_full< TN > & t) const`

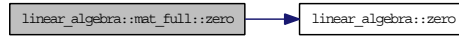
Here is the call graph for this function:



4.29.3.59 `template<class TN> mat_full< TN > linear_algebra::mat_full< TN >::transpose () const`

4.29.3.60 `template<class TN> mat_full< TN > & linear_algebra::mat_full< TN >::zero ()`

Here is the call graph for this function:



#### 4.29.4 Member Data Documentation

4.29.4.1 `template<class TN> refvector<refvector<TN> > linear_algebra::mat_full< TN >::_scols[protected]`

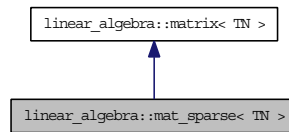
The documentation for this class was generated from the following files:

- `mat_full.decl`
- `mat_full.h`
- `mat_full.h`

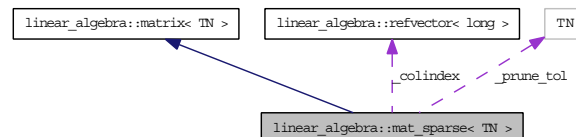
#### 4.30 `linear_algebra::mat_sparse< TN >` Class Template Reference

Class of sparse column matrices.

Inheritance diagram for `linear_algebra::mat_sparse< TN >`:



Collaboration diagram for `linear_algebra::mat_sparse< TN >`:



#### Public Member Functions

- `mat_sparse (long n, long m)`  
*Constructor of  $n$  columns and  $m$  rows with uninitialised values.*

- `mat_sparse (long n, long m, const vector< long > &scols, const vector< sparse_vector< TN > > &vals)`  
*Constructor of n columns and m rows with initialisation of values.*
- `mat_sparse (long n, long m, const vector< long > &scols, const refvector< sparse_vector< TN > > &vals)`  
*Constructor of n columns and m rows with initialisation of values.*
- `mat_sparse (long n, long m, const refvector< long > &scols, const refvector< sparse_vector< TN > > &vals)`  
*Constructor of n columns and m rows with initialisation of values.*
- `mat_sparse (long n, long m, const refvector< long > &scols, const vector< sparse_vector< TN > > &vals)`  
*Constructor of n columns and m rows with initialisation of values.*
- `mat_sparse ()`
- `mat_sparse< TN > & operator= (mat_sparse< TN > &a)`
- `sparse_vector< TN > & operator[] (const long x)`  
*Return a column through direct access. No integrity checked.*
- `const sparse_vector< TN > & operator[] (const long x) const`  
*Return a column through direct access. No integrity checked.*
- `virtual const TN & operator() (const long x, const long y) const`  
*Return value at (x,y).*
- `mat_sparse< TN > & operator *= (const TN &x)`  
*Multiply by an object of TN into left side.*
- `mat_sparse< TN > & operator /= (const TN &x)`  
*Divide by an object of TN into left side. This is very slow and not advisable.*
- `mat_sparse< TN > operator * (const mat_sparse< TN > &b) const`  
*Multiply two sparse matrices.*
- `sparse_vector< TN > operator * (const sparse_vector< TN > &a) const`  
*Multiply a sparse matrix by a sparse vector.*
- `mat_sparse< TN > operator * (const mat_sym_sparse< TN > &b) const`  
*Multiply a sparse matrix by a sparse symmetric matrix.*
- `mat_sparse< TN > & operator *= (const mat_sparse< TN > &b)`  
*Multiplies the left hand side from the { left } by the right hand side.*

- `mat_sparse< TN > & operator *=(const mat_full< TN > &b)`  
*Multiplies the **l**eft hand side from the {left } by the **r**ight hand side.*
- `refvector< TN > & multiply (const refvector< TN > &a, refvector< TN > &b) const`  
*Multiply a sparse matrix with a full refvector.*
- `refvector< TN > operator * (const refvector< TN > &a) const`  
*Multiply a sparse matrix with a full refvector.*
- `mat_full< TN > & multiply (const mat_full< TN > &a, mat_full< TN > &R) const`  
*Multiply a sparse matrix with a full matrix.*
- `mat_full< TN > operator * (const mat_full< TN > &a) const`  
*Multiply a sparse matrix with a full matrix.*
- `mat_sparse< TN > & copy (const mat_sparse< TN > &a)`  
*Copy matrix with deep copy.*
- `mat_sparse< TN > transpose () const`  
*Transpose the sparse matrix.*
- `void prune (const TN x)`  
*Prune the matrix.*
- `void prune ()`
- `bool is_nonzero (long x, long y, long &j, long &k) const`  
*Checks whether an access is non zero.*
- `bool is_col_nonzero (const long x, long &i) const`  
*Checks whether a column is empty.*
- `bool add (const long x, const long y, const TN z)`  
*Adds a value.*
- `bool display () const`  
*Simple display of matrix.*
- `vector< TN > extract_full_col (const long column, const long xmin, const long xmax) const`  
*Extracts a block from a column.*
- `vector< TN > extract_full_col (const long column, const long xmin) const`  
*Extracts a block from a column with default max=matrix<TN>\_rows.*
- `sparse_vector< TN > extract_sparse_col (const long column, const long xmin) const`  
*Extracts a block from a column with default max=matrix<TN>\_rows.*

- `sparse_vector< TN > extract_sparse_col (const long column, const long xmin, const long xmax) const`  
*Extracts a sparse block from a column.*
- `mat_sym_sparse< TN > sym_upper_triangle () const`  
*Returns a `mat_sym_sparse` object of the upper triangle of a sparse matrix.*
- `void erase (long c)`  
*Deletes described column from the matrix. Direct access deletion.*
- `double density () const`  
*Return the density of non-zero elements.*
- `long size () const`  
*Returns the number of nonzero columns.*
- `long index (const long x) const`  
*Returns the column of a given index.*
- `void set (const long x, const long y, TN z)`  
*Set a value.*
- `void clear ()`  
*Sets the matrix to zero releasing all associated memory.*
- `mat_sparse< TN > & zero ()`  
*Sets the matrix to zero releasing all associated memory.*

## Public Attributes

- `TN _prune_tol`  
*Pruning tolerance.*

## Protected Attributes

- `refvector< sparse_vector< TN > > _scols`  
*Vector of columns.*
- `refvector< long > _colindex`  
*Index of nonzero columns.*
- `bool _empty`  
*State of matrix.*

## Friends

- `class mat_sym_sparse< TN >`

### 4.30.1 Detailed Description

`template<class TN> class linear_algebra::mat_sparse< TN >`

'column' specifies that the values are ordered for speedy retrieval per column.

### 4.30.2 Constructor & Destructor Documentation

4.30.2.1 `template<class TN> linear_algebra::mat_sparse< TN >::mat_sparse (long n, long m)`

4.30.2.2 `template<class TN> linear_algebra::mat_sparse< TN >::mat_sparse (long n, long m, const vector< long > & scols, const vector< sparse_vector< TN > > & vals)`

*n* Dimension of matrix.

*scols* Vector of column indices.

*vals* Vector of columns. Each column is a sparse vector.

Each value has an associated row and column index in *scols* and *scols*. Therefore they all have the same size.

4.30.2.3 `template<class TN> linear_algebra::mat_sparse< TN >::mat_sparse (long n, long m, const vector< long > & scols, const refvector< sparse_vector< TN > > & vals)`

#### Parameters:

*n* Dimension of matrix.

*scols* Vector of column indices.

*vals* Reference counted vector of columns. Each column is a sparse vector.

Each value has an associated row and column index in *scols* and *scols*. Therefore they all have the same size.

4.30.2.4 `template<class TN> linear_algebra::mat_sparse< TN >::mat_sparse (long n, long m, const refvector< long > & scols, const refvector< sparse_vector< TN > > & vals)`

#### Parameters:

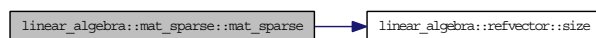
*n* Dimension of matrix.

*scols* Reference counted vector of column indices.

*vals* Reference counted vector of columns. Each column is a sparse vector.

Each value has an associated row and column index in *srows* and *scols*. Therefore they all have the same size.

Here is the call graph for this function:



4.30.2.5 `template<class TN> linear_algebra::mat_sparse< TN >::mat_sparse (long n, long m, const refvector< long > & scols, const vector< sparse_vector< TN > > & vals)`

### Parameters:

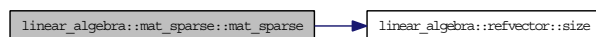
*n* Dimension of matrix.

*scols* Reference counted Vector of column indices.

*vals* Vector of columns. Each column is a sparse vector.

Each value has an associated row and column index in *srows* and *scols*. Therefore they all have the same size.

Here is the call graph for this function:



4.30.2.6 `template<class TN> linear_algebra::mat_sparse< TN >::mat_sparse ()`

## 4.30.3 Member Function Documentation

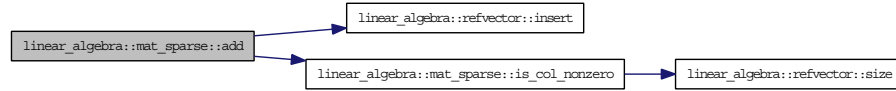
4.30.3.1 `template<class TN> bool linear_algebra::mat_sparse< TN >::add (const long x, const long y, const TN z)`

### Parameters:

*z* is added to (*x*,*y*). If the value is zero the associated vectors are expanded.

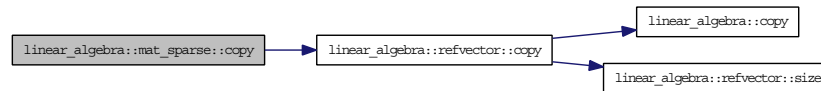
Here is the call graph for this function:

4.30.3.2 `template<class TN> void linear_algebra::mat_sparse< TN >::clear ()`



4.30.3.3 `template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::copy (const mat_sparse< TN > & a)`

Here is the call graph for this function:



4.30.3.4 `template<class TN> double linear_algebra::mat_sparse< TN >::density () const`

4.30.3.5 `template<class TN> bool linear_algebra::mat_sparse< TN >::display () const`

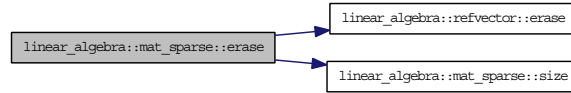
Here is the call graph for this function:



4.30.3.6 `template<class TN> void linear_algebra::mat_sparse< TN >::erase (long c)`

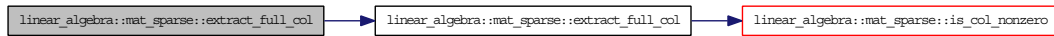
Here is the call graph for this function:





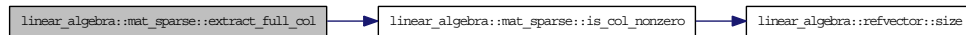
4.30.3.7 `template<class TN> vector< TN > linear_algebra::mat_sparse< TN >::extract_full_col (const long column, const long xmin) const`

Here is the call graph for this function:



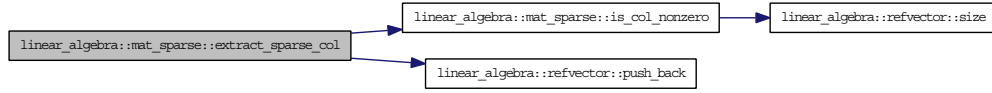
4.30.3.8 `template<class TN> vector< TN > linear_algebra::mat_sparse< TN >::extract_full_col (const long column, const long xmin, const long xmax) const`

Here is the call graph for this function:



4.30.3.9 `template<class TN> sparse_vector< TN > linear_algebra::mat_sparse< TN >::extract_sparse_col (const long column, const long xmin, const long xmax) const`

Here is the call graph for this function:

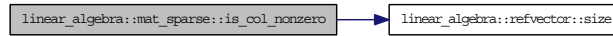


4.30.3.10 `template<class TN> sparse_vector< TN > linear_algebra::mat_sparse< TN >::extract_sparse_col (const long column, const long xmin) const`

4.30.3.11 `template<class TN> long linear_algebra::mat_sparse< TN >::index (const long x) const`

4.30.3.12 `template<class TN> bool linear_algebra::mat_sparse< TN >::is_col_nonzero (const long x, long & i) const`

Here is the call graph for this function:



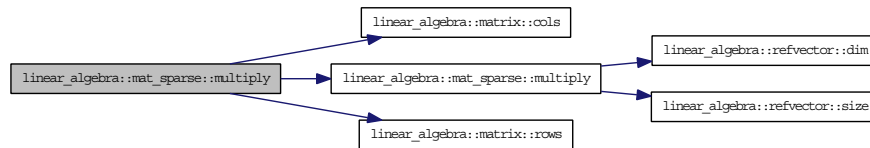
4.30.3.13 `template<class TN> bool linear_algebra::mat_sparse< TN >::is_nonzero (long x, long y, long & j, long & k) const`

### Parameters:

*j, k* hold the position of *x, y* in `_scols, _scols[j]`. If (*x, y*) is 0. An insertion would happen here.

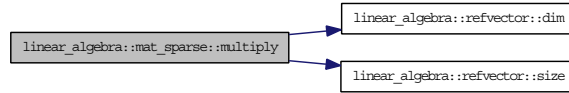
4.30.3.14 `template<class TN> mat_full< TN > & linear_algebra::mat_sparse< TN >::multiply (const mat_full< TN > & a, mat_full< TN > & R) const`

Here is the call graph for this function:



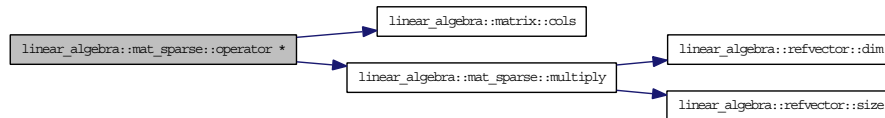
4.30.3.15 `template<class TN> refvector< TN > & linear_algebra::mat_sparse< TN >::multiply (const refvector< TN > & a, refvector< TN > & b) const`

Here is the call graph for this function:



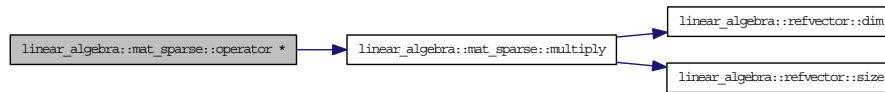
4.30.3.16 `template<class TN> mat_full< TN > linear_algebra::mat_sparse< TN >::operator * (const mat_full< TN > & a) const`

Here is the call graph for this function:



4.30.3.17 `template<class TN> refvector< TN > linear_algebra::mat_sparse< TN >::operator * (const refvector< TN > & a) const`

Here is the call graph for this function:



4.30.3.18 `template<class TN> mat_sparse< TN > linear_algebra::mat_sparse< TN >::operator * (const mat_sym_sparse< TN > & b) const`

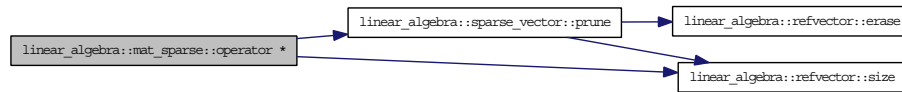
The result is automatically pruned for the default pruning value set in `_prune_tol`.

Here is the call graph for this function:



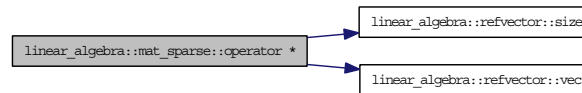
4.30.3.19 `template<class TN> sparse_vector< TN > linear_algebra::mat_sparse< TN >::operator * (const sparse_vector< TN > & a) const`

Here is the call graph for this function:



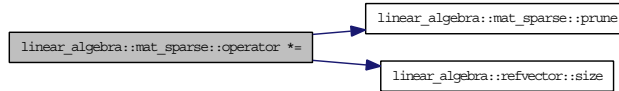
4.30.3.20 `template<class TN> mat_sparse< TN > linear_algebra::mat_sparse< TN >::operator * (const mat_sparse< TN > & b) const`

Here is the call graph for this function:



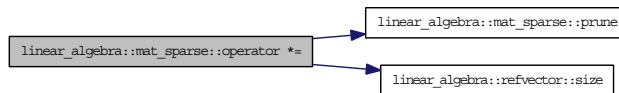
4.30.3.21 `template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::operator *= (const mat_full< TN > & b)`

Here is the call graph for this function:



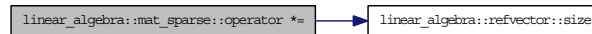
4.30.3.22 `template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::operator *= (const mat_sparse< TN > & b)`

Here is the call graph for this function:



4.30.3.23 `template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::operator *= (const TN & x)`

Here is the call graph for this function:



4.30.3.24 `template<class TN> const TN & linear_algebra::mat_sparse< TN >::operator() (const long x, const long y) const[virtual]`

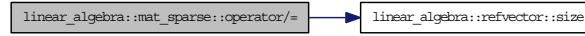
Implements `linear_algebra::matrix< TN >`.

Here is the call graph for this function:



4.30.3.25 `template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::operator/= (const TN & x)`

Here is the call graph for this function:



4.30.3.26 `template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::operator= (mat_sparse< TN > & a)`

4.30.3.27 `template<class TN> const sparse_vector< TN > & linear_algebra::mat_sparse< TN >::operator[] (const long x) const`

4.30.3.28 `template<class TN> sparse_vector< TN > & linear_algebra::mat_sparse< TN >::operator[] (const long x)`

4.30.3.29 `template<class TN> void linear_algebra::mat_sparse< TN >::prune ()`

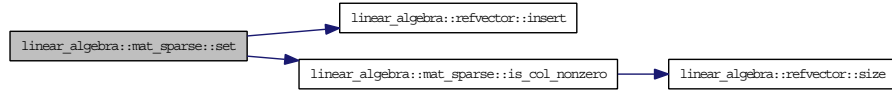
4.30.3.30 `template<class TN> void linear_algebra::mat_sparse< TN >::prune (const TN x)`

Here is the call graph for this function:



4.30.3.31 `template<class TN> void linear_algebra::mat_sparse< TN >::set (const long x, const long y, TN z)`

Here is the call graph for this function:

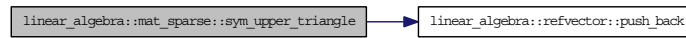


4.30.3.32 `template<class TN> long linear_algebra::mat_sparse< TN >::size () const`

4.30.3.33 `template<class TN> mat_sym_sparse< TN > linear_algebra::mat_sparse< TN >::sym_upper_triangle () const`

### Test

Here is the call graph for this function:



4.30.3.34 `template<class TN> mat_sparse< TN > linear_algebra::mat_sparse< TN >::transpose () const`

Here is the call graph for this function:



4.30.3.35 `template<class TN> mat_sparse< TN > & linear_algebra::mat_sparse< TN >::zero ()`

## 4.30.4 Friends and Related Function Documentation

4.30.4.1 `template<class TN> friend class mat_sym_sparse< TN >[friend]`

## 4.30.5 Member Data Documentation

4.30.5.1 `template<class TN> refvector<long> linear_algebra::mat_sparse< TN >::_colindex[protected]`

4.30.5.2 `template<class TN> bool linear_algebra::mat_sparse< TN >::_empty[protected]`

4.30.5.3 `template<class TN> TN linear_algebra::mat_sparse< TN >::_prune_tol`

4.30.5.4 `template<class TN> refvector<sparse_vector<TN> > linear_algebra::mat_sparse< TN >::_scols[protected]`

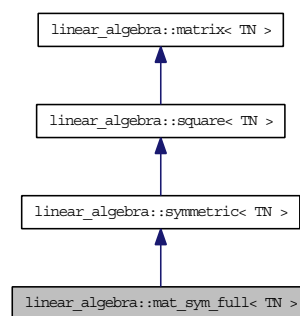
The documentation for this class was generated from the following files:

- include/BCR\_CPP\_LA/mat\_sparse.decl
- include/BCR\_CPP\_LA/mat\_sparse.h

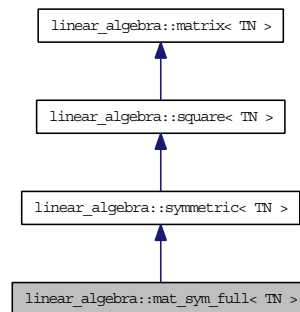
#### 4.31 linear\_algebra::mat\_sym\_full< TN > Class Template Reference

mat\_sym\_full uses packing for full symmetric matrices

Inheritance diagram for linear\_algebra::mat\_sym\_full< TN >:



Collaboration diagram for linear\_algebra::mat\_sym\_full< TN >:



#### Public Member Functions

- `mat_sym_full ()`  
*Default constructor.*
- `mat_sym_full (istream &IN)`  
*Constructor to read a stored matrix from a stream. Aimed at retrieving information from a binary save with `mat_sym_full::operator>>()`.*



- `mat_sym_full (long n)`  
*Constructor of dimension  $n$  with uninitialised values.*
- `mat_sym_full (long n, const vector< TN > &vals)`  
*Constructor of dimension  $n$  with initialisation of values.*
- `mat_sym_full (long n, refvector< TN > &vals)`  
*Constructor of dimension  $n$  with initialisation of values.*
- `void operator> > (ostream &OUT) const`  
*Output matrix in binary format into a stream. Aimed at file storage.*
- `mat_sym_full< TN > & operator= (mat_sym_full< TN > &a)`  
*Assignment operator for non-const matrices.*
- `mat_sym_full< TN > & operator= (const mat_sym_full< TN > &a)`  
*Assignment operator for non-const matrices.*
- `const TN & operator() (const long x, const long y) const`  
*Access operator which checks for integrity in debug mode.*
- `TN & operator() (const long x, const long y)`  
*Access operator which checks for integrity in debug mode.*
- `const TN & operator mbox[] (const long x) const`  
*Direct access operator which checks for integrity in debug mode.*
- `TN & operator mbox[] (const long x)`  
*Direct access operator which checks for integrity in debug mode.*
- `refvector< TN > operator() (const long column, const long xmin, const long xmax) const`  
*Extracts a block from a column/row.*
- `refvector< TN > operator() (const long column) const`  
*Extracts a column/row from the matrix.*
- `mat_sym_full< TN > & multiply (long x, long y, TN a)`  
*Addition operation for compatibility with `mat_sym_sparse`.*
- `mat_full< TN > & multiply (const mat_full< TN > &b, mat_full< TN > &r) const`  
*Multiplies a symmetric full matrix with a full matrix.*
- `mat_full< TN > operator * (const mat_full< TN > &b) const`

- `mat_full< TN > & multiply (const mat_sym_full< TN > &b, mat_full< TN > &r)`  
`const`  
*Multiply a symmetric full matrix with a symmetric full matrix.*
- `mat_full< TN > operator * (const mat_sym_full< TN > &b) const`  
*Multiply a symmetric full matrix with a symmetric full matrix.*
- `mat_full< TN > & multiply (const mat_asym_full< TN > &b, mat_full< TN > &r)`  
`const`  
*Multiply a symmetric full matrix with an antisymmetric full matrix.*
- `mat_full< TN > operator * (const mat_asym_full< TN > &b) const`  
*Multiply a symmetric full matrix with an antisymmetric full matrix.*
- `template<class TN2> refvector< TN > & multiply (const sparse_vector< TN2 >`  
`&v, refvector< TN > &r) const`  
*Multiply a symmetric full matrix with a sparse vector.*
- `template<class TN2> refvector< TN > operator * (const sparse_vector< TN2 > &v)`  
`const`  
*Multiply a symmetric full matrix with a sparse vector.*
- `template<class TN2> refvector< TN > & multiply (const refvector< TN2 > &v,`  
`refvector< TN > &r) const`  
*Multiply a symmetric full matrix with a full vector.*
- `template<class TN2> refvector< TN > operator * (const refvector< TN2 > &v)`  
`const`  
*Multiply a symmetric full matrix with a full vector.*
- `template<class TN2> mat_sym_full< TN > operator *= (const TN2 &a)`  
*Multiply a symmetric full matrix with an element.*
- `template<class TN2> mat_sym_full< TN > & operator /= (const TN2 &a)`  
*Divide a symmetric full matrix by an element.*
- `mat_sym_full< TN > & multiply (const TN &a, mat_sym_full< TN > &r) const`  
*Multiply a symmetric full matrix with an element.*
- `mat_sym_full< TN > operator * (const TN &a) const`  
*Multiply a symmetric full matrix with an element.*
- `mat_sym_full< TN > operator / (const TN &a) const`  
*Divide a symmetric full matrix by an element.*

- `mat_sym_full< TN > & operator+= (const mat_sym_full< TN > &a)`  
*Add two symmetric matrices.*
- `mat_sym_full< TN > & operator+= (const mat_sym_sparse< TN > &a)`  
*Add two symmetric matrices.*
- `mat_sym_full< TN > & operator-= (const mat_sym_full< TN > &a)`  
*Subtract two symmetric matrices.*
- `mat_sym_full< TN > & operator-= (const mat_sym_sparse< TN > &a)`  
*Subtract two symmetric matrices.*
- `mat_sym_full< TN > operator+ (const mat_sym_full< TN > &a) const`  
*Add two symmetric matrices.*
- `mat_sym_full< TN > operator- (const mat_sym_full< TN > &a) const`  
*Subtract two symmetric matrices.*
- `mat_sym_full< TN > operator- (const mat_sym_sparse< TN > &a) const`  
*Subtract two symmetric matrices, one full one sparse.*
- `bool display () const`  
*Simple display of matrix.*
- `void set (const long x, const long y, TN value)`  
*Assignment operation for compatibility with mat\_sym\_sparse.*
- `bool add (long x, long y, const TN value)`  
*Addition operation for compatibility with mat\_sym\_sparse.*
- `void gen_mat (const refvector< TN > &a, const refvector< TN > &b, TN correction)`  
*Produce the necessary  $ij^* + ji^*$ .*
- `mat_sym_full< TN > unitary_transform (const mat_full< TN > U) const`  
*Transforms a symmetric full matrix by a unitary matrix.  $O(n^4)$   $UAU^t$ .*
- `mat_full< TN > householder (refvector< TN > &diagonal, refvector< TN > &subdiagonal)`  
*This routine produces the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.*
- `mat_sym_full< TN > & copy (const mat_sym_full< TN > &a)`  
*Copy matrix with full depth.*
- `mat_sym_full< TN > & copy (const mat_sym_full< TN > &a, const long i)`  
*Copy matrix with depth i.*

- `mat_sym_full< TN > & copy mat_sym_full< TN > &a, const long i)`  
*Copy matrix with depth i.*
- `symmetric< TN > & copy (const symmetric< TN > &a)`  
*Copy matrix with depth 1.*
- `symmetric< TN > & copy (const symmetric< TN > &a, const long i)`  
*Copy matrix with depth i.*
- `symmetric< TN > & copy (symmetric< TN > &a, const long i)`  
*Copy matrix with depth i.*
- `void clear ()`  
*Release the memory associated with the matrix.*
- `mat_sym_full< TN > & uut(const mat_full< TN > &U, mat_sym_full< TN > &R, mat_full< TN > &I) const`  
*This is the unitary transform  $UAU^*$ .*
- `mat_sym_full< TN > & uut (const mat_full< TN > &U, mat_sym_full< TN > &R) const`  
*This is the unitary transform  $UAU^*$ .*
- `mat_sym_full< TN > uut (const mat_full< TN > &U) const`  
*This is the unitary transform  $UAU^*$ .*
- `mat_sym_full< TN > & utu (const mat_sym_full< TN > &a, mat_sym_full< TN > &R, mat_full< TN > &I) const`  
*This a symmetric transform SAS.*
- `mat_sym_full< TN > utu (const mat_sym_full< TN > &a, mat_full< TN > &I) const`  
*This a symmetric transform SAS.*
- `mat_sym_full< TN > & utu (const mat_sym_full< TN > &a, mat_sym_full< TN > &R) const`  
*This a symmetric transform SAS.*
- `mat_sym_full< TN > utu (const mat_sym_full< TN > &a) const`  
*This a symmetric transform SAS.*
- `mat_sym_full< TN > & utu (const mat_full< TN > &U, mat_sym_full< TN > &R) const`  
*This is the unitary transform  $U^*AU$ .*

- `mat_sym_full< TN > & utu (const mat_full< TN > &U, mat_sym_full< TN > &R, refvector< TN > &v) const`  
*This is the unitary transform  $U^*AU$ .*
- `mat_sym_full< TN > utu (const mat_full< TN > &U) const`  
*This is the unitary transform  $U^\wedge tAU$ .*
- `mat_sym_full< TN > utu (const mat_asym_full< TN > &a, mat_full< TN > &I) const`  
*This a symmetric transform  $SAS$ .*
- `mat_sym_full< TN > utu (const mat_asym_full< TN > &U) const`  
*This is the unitary transform  $U^\wedge tAU$ .*
- `mat_sym_full< TN > & prune (const TN tol)`  
*Prune the matrix of negligible values.*
- `mat_sym_full< TN > & zero ()`  
*Set the matrix to zero.*
- `mat_sym_full< TN > & resize (long n)`  
*Constructor of dimension  $n$  with uninitialised values.*

## Private Attributes

- `refvector< TN > _vals`  
*Vector of values.*

### 4.31.1 Detailed Description

`template<class TN> class linear_algebra::mat_sym_full< TN >`

The packing is optimised for column access.

### 4.31.2 Constructor & Destructor Documentation

4.31.2.1 `template<class TN> linear_algebra::mat_sym_full< TN >::mat_sym_full ()`

4.31.2.2 `template<class TN> linear_algebra::mat_sym_full< TN >::mat_sym_full (istream & IN)`

4.31.2.3 `template<class TN> linear_algebra::mat_sym_full< TN >::mat_sym_full (long n)`

4.31.2.4 `template<class TN> linear_algebra::mat_sym_full< TN >::mat_sym_full (long n, const vector< TN > & vals)`

4.31.2.5 `template<class TN> linear_algebra::mat_sym_full< TN >::mat_sym_full (long n, refvector< TN > & vals)`

### 4.31.3 Member Function Documentation

4.31.3.1 `template<class TN> bool linear_algebra::mat_sym_full< TN >::add (long x, long y, const TN value)`

Reimplemented from `linear_algebra::symmetric< TN >`.

4.31.3.2 `template<class TN> void linear_algebra::mat_sym_full< TN >::clear ()`

4.31.3.3 `template<class TN> symmetric< TN > & linear_algebra::mat_sym_full< TN >::copy (symmetric< TN > & a, const long i)[virtual]`

Implements `linear_algebra::symmetric< TN >`.

Here is the call graph for this function:



4.31.3.4 `template<class TN> symmetric< TN > & linear_algebra::mat_sym_full< TN >::copy (const symmetric< TN > & a, const long i)[virtual]`

Implements `linear_algebra::symmetric< TN >`.

Here is the call graph for this function:



4.31.3.5 `template<class TN> symmetric< TN > & linear_algebra::mat_sym_full< TN >::copy (const symmetric< TN > & a)[virtual]`

Implements `linear_algebra::symmetric< TN >`.

Here is the call graph for this function:



4.31.3.6 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::copy (mat_sym_full< TN > & a, const long i)`

4.31.3.7 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::copy (const mat_sym_full< TN > & a, const long i)`

4.31.3.8 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::copy (const mat_sym_full< TN > & a)`

4.31.3.9 `template<class TN> bool linear_algebra::mat_sym_full< TN >::display () const`

4.31.3.10 `template<class TN> void linear_algebra::mat_sym_full< TN >::gen_mat (const refvector< TN > & a, const refvector< TN > & b, TN correction)`

the result is  $- = ab^* + ba^*$  ( $- = ab^{\wedge} + ba^{\wedge}$ ).

Here is the call graph for this function:

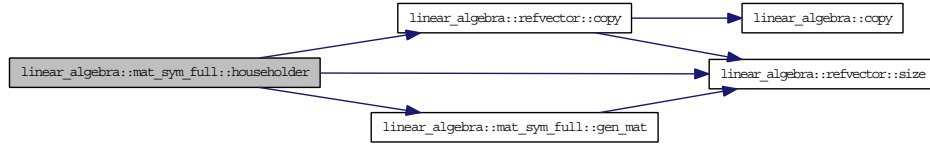


4.31.3.11 `template<class TN> mat_full< TN > linear_algebra::mat_sym_full< TN >::householder (refvector< TN > & diagonal, refvector< TN > & subdiagonal)`

The aspired transformation is a product of  $(I - v_i v_i^*)$  where  $v_i$  is  $(\pm e_1 \|a_{12}\| + a_{12}) / \sqrt{\pm a_{12,1} \|a_{12}\| + \|a_{12}\|^2}$ . Notice that we begin counting at 0. The matrix is split up according to  $A = \begin{pmatrix} a_{11} & a_{12}^* \\ a_{12} & A_{22} \end{pmatrix}$ . We use preconditioning such that the matrix is ordered with lowest off-diagonal value in the low right-hand corner.

$$\begin{pmatrix} a_{11} & \cdots & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & 0 \\ & & & & a_{nn} \end{pmatrix}$$
 The returned matrix U will transform S to tridiagonal form via  $U^* S U$ .

Here is the call graph for this function:



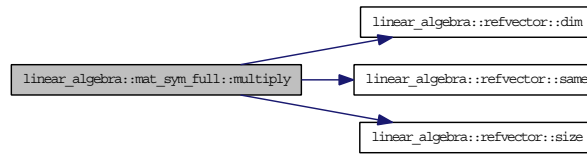
4.31.3.12 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::multiply (const TN & a, mat_sym_full< TN > & r) const`

Here is the call graph for this function:



4.31.3.13 `template<class TN> template<class TN2> refvector< TN > & linear_algebra::mat_sym_full< TN >::multiply (const refvector< TN2 > & v, refvector< TN > & r) const`

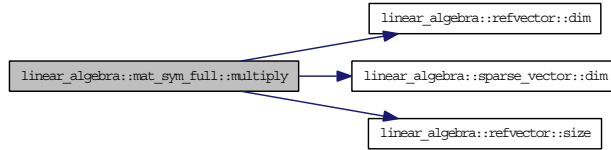
Here is the call graph for this function:



4.31.3.14 `template<class TN> template<class TN2> refvector< TN > & linear_algebra::mat_sym_full< TN >::multiply (const sparse_vector< TN2 > & v, refvector< TN > & r) const`

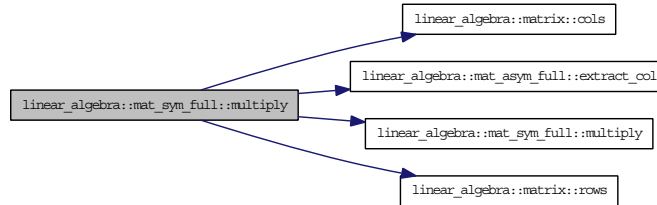
Here is the call graph for this function:





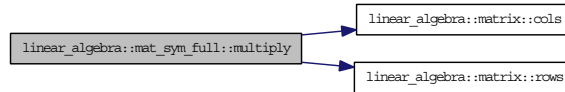
4.31.3.15 `template<class TN> mat_full< TN > & linear_algebra::mat_sym_full< TN >::multiply (const mat_asym_full< TN > & b, mat_full< TN > & r) const`

Here is the call graph for this function:



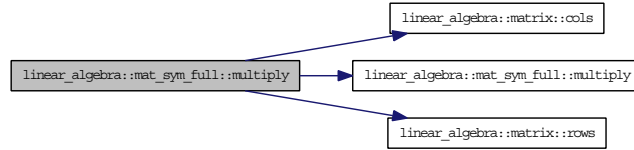
4.31.3.16 `template<class TN> mat_full< TN > & linear_algebra::mat_sym_full< TN >::multiply (const mat_sym_full< TN > & b, mat_full< TN > & r) const`

Here is the call graph for this function:



4.31.3.17 `template<class TN> mat_full< TN > & linear_algebra::mat_sym_full< TN >::multiply (const mat_full< TN > & b, mat_full< TN > & r) const`

Here is the call graph for this function:



4.31.3.18 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::multiply (long x, long y, TN a)`

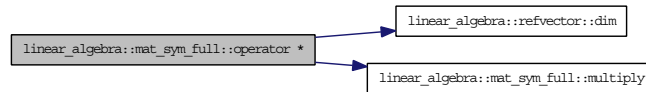
4.31.3.19 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::operator * (const TN & a) const`

Here is the call graph for this function:



4.31.3.20 `template<class TN> template<class TN2> refvector< TN > linear_algebra::mat_sym_full< TN >::operator * (const refvector< TN2 > & v) const`

Here is the call graph for this function:



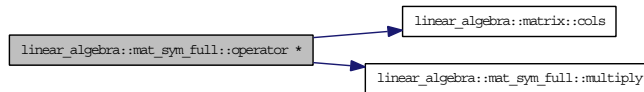
4.31.3.21 `template<class TN> template<class TN2> refvector< TN > linear_algebra::mat_sym_full< TN >::operator * (const sparse_vector< TN2 > & v) const`

Here is the call graph for this function:



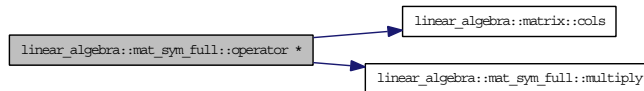
4.31.3.22 `template<class TN> mat_full< TN > linear_algebra::mat_sym_full< TN >::operator * (const mat_asym_full< TN > & b) const`

Here is the call graph for this function:



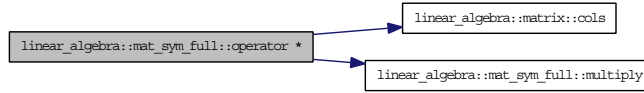
4.31.3.23 `template<class TN> mat_full< TN > linear_algebra::mat_sym_full< TN >::operator * (const mat_sym_full< TN > & b) const`

Here is the call graph for this function:



4.31.3.24 `template<class T> mat_full< T > linear_algebra::mat_sym_full< T >::operator * (const mat_full< TN > & b) const`

Here is the call graph for this function:



4.31.3.25 `template<class TN> template<class TN2> mat_sym_full< TN >  
linear_algebra::mat_sym_full< TN >::operator *= (const TN2 & a)`

4.31.3.26 `template<class TN> refvector< TN > linear_algebra::mat_sym_full< TN  
>::operator() (const long column) const`

4.31.3.27 `template<class TN> refvector< TN > linear_algebra::mat_sym_full< TN  
>::operator() (const long column, const long xmin, const long xmax) const`

4.31.3.28 `template<class TN> TN & linear_algebra::mat_sym_full< TN >::operator()  
(const long x, const long y)`

4.31.3.29 `template<class TN> const TN & linear_algebra::mat_sym_full< TN >::operator()  
(const long x, const long y) const[virtual]`

Implements `linear_algebra::matrix< TN >`.

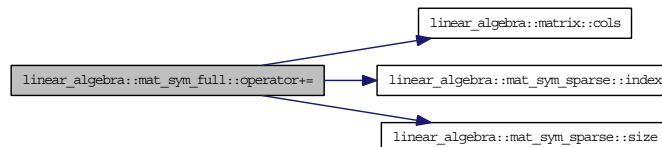
4.31.3.30 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN  
>::operator+ (const mat_sym_full< TN > & a) const`

Here is the call graph for this function:



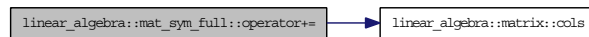
4.31.3.31 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN  
>::operator+= (const mat_sym_sparse< TN > & a)`

Here is the call graph for this function:



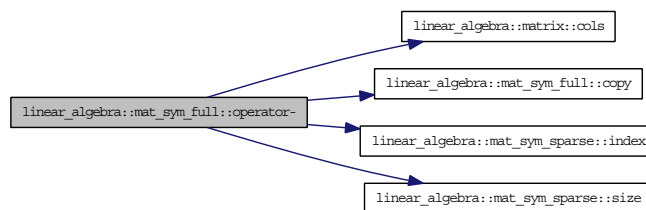
4.31.3.32 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN  
>::operator+= (const mat_sym_full< TN > & a)`

Here is the call graph for this function:



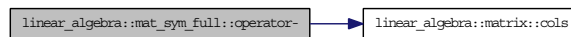
4.31.3.33 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::operator- (const mat_sym_sparse< TN > & a) const`

Here is the call graph for this function:



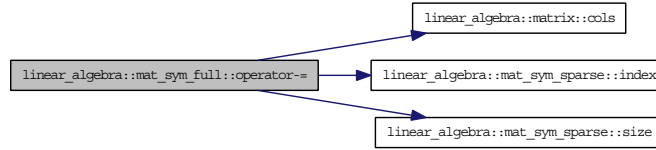
4.31.3.34 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::operator- (const mat_sym_full< TN > & a) const`

Here is the call graph for this function:



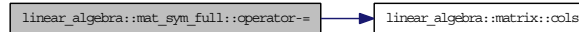
4.31.3.35 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::operator-= (const mat_sym_sparse < TN > & a)`

Here is the call graph for this function:



4.31.3.36 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::operator-= (const mat_sym_full< TN > & a)`

Here is the call graph for this function:



4.31.3.37 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::operator/ (const TN & a) const`

4.31.3.38 `template<class TN> template<class TN2> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::operator/= (const TN2 & a)`

4.31.3.39 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::operator= (const mat_sym_full< TN > & a)`

4.31.3.40 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::operator= (mat_sym_full< TN > & a)`

The default is to output a full (column) matrix.

4.31.3.41 `template<class TN> void linear_algebra::mat_sym_full< TN >::operator>> (ostream & OUT) const`

The default is to output a full (column) matrix.

4.31.3.42 `template<class TN> TN & linear_algebra::mat_sym_full< TN >::operator[] (const long )`

4.31.3.43 `template<class TN> const TN & linear_algebra::mat_sym_full< TN >::operator[] (const long x) const`

4.31.3.44 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::prune (const TN tol)`

Reimplemented from `linear_algebra::symmetric< TN >`.

4.31.3.45 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::resize (long n)`

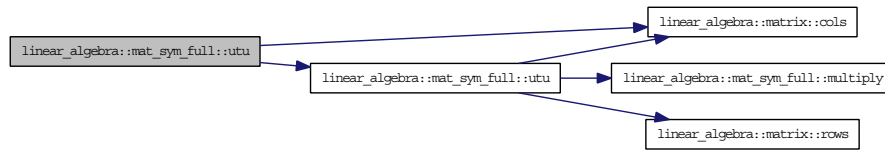
4.31.3.46 `template<class TN> void linear_algebra::mat_sym_full< TN >::set (const long x, const long y, TN value)`

4.31.3.47 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::unitary_transform (const mat_full< TN > U) const`

This transform is only suggested if memory management is of importance. Otherwise using two matrix mulitplications is faster  $O(n^3)$ .

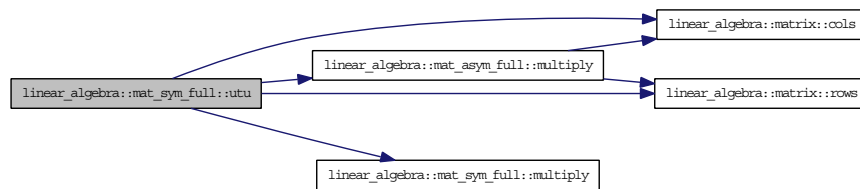
4.31.3.48 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::utu (const mat_asym_full< TN > & U) const`

Here is the call graph for this function:



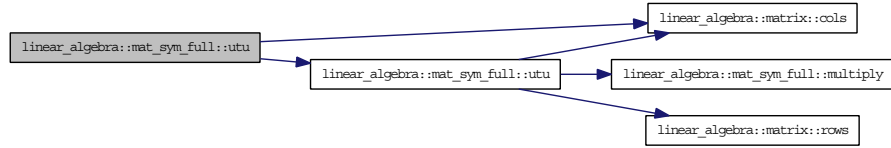
4.31.3.49 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::utu (const mat_asym_full< TN > & a, mat_full< TN > & I) const`

Here is the call graph for this function:



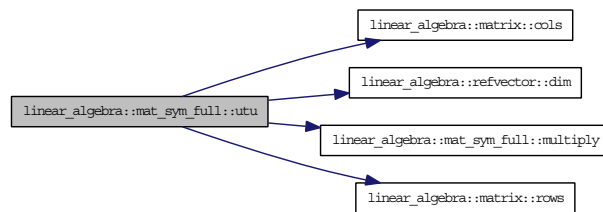
4.31.3.50 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::utu (const mat_full< TN > & U) const`

Here is the call graph for this function:



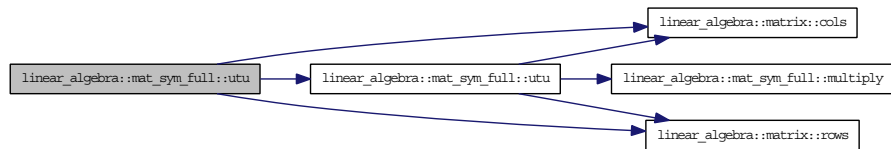
4.31.3.51 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::utu (const mat_full< TN > & U, mat_sym_full< TN > & R, refvector< TN > & v) const`

Here is the call graph for this function:



4.31.3.52 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::utu (const mat_full< TN > & U, mat_sym_full< TN > & R) const`

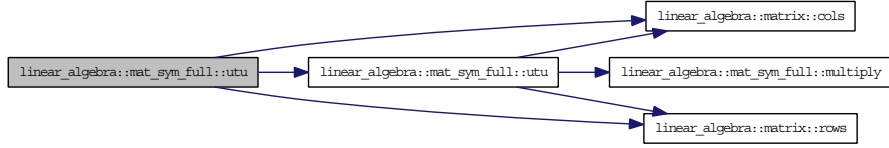
Here is the call graph for this function:



4.31.3.53 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::utu (const mat_sym_full< TN > & a) const`

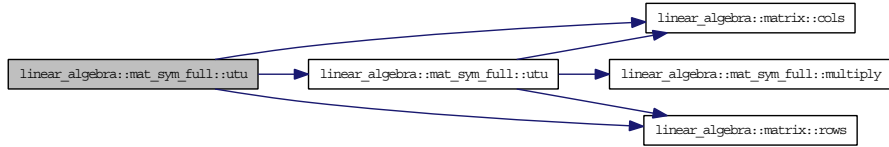
Here is the call graph for this function:





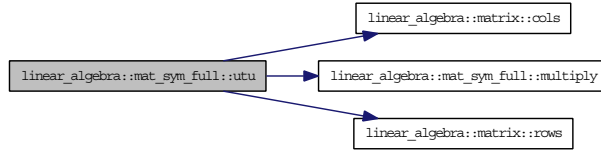
4.31.3.54 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::utu (const mat_sym_full< TN > & a, mat_sym_full< TN > & ) const`

Here is the call graph for this function:



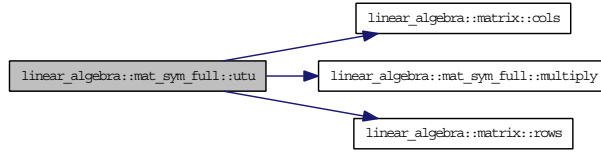
4.31.3.55 `template<class TN> mat_sym_ful< TN > linear_algebra::mat_sym_full< TN >::utu (const mat_sym_full< TN > & a, mat_full< TN > & I) const`

Here is the call graph for this function:



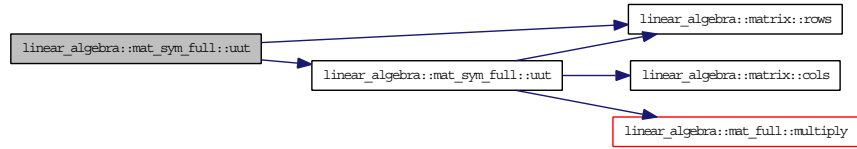
4.31.3.56 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::utu (const mat_sym_full< TN > & a, mat_sym_full< TN > & R, mat_full< TN > & I) const`

Here is the call graph for this function:



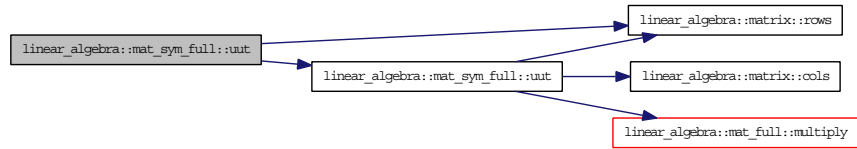
4.31.3.57 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_full< TN >::utu (const mat_full< TN > & U) const`

Here is the call graph for this function:



4.31.3.58 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::utu (const mat_full< TN > & U, mat_sym_full< TN > & R) const`

Here is the call graph for this function:



4.31.3.59 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::utu (const mat_full< TN > & U, mat_sym_full< TN > & R, mat_full< TN > & I) const`

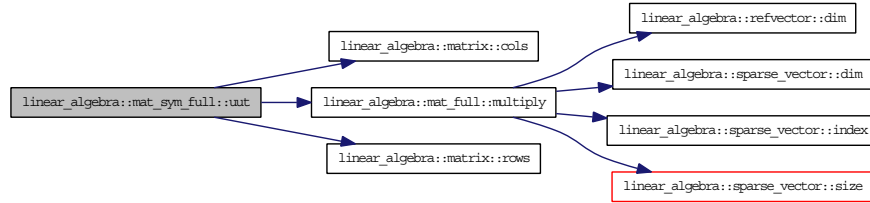
Here is the call graph for this function:

4.31.3.60 `template<class TN> mat_sym_full< TN > & linear_algebra::mat_sym_full< TN >::zero ()`

## 4.31.4 Member Data Documentation

4.31.4.1 `template<class TN> refvector<TN> linear_algebra::mat_sym_full< TN >::_vals[private]`

The documentation for this class was generated from the following files:

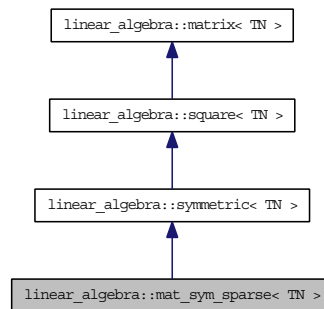


- `include/BCR_CPP_LA/mat_sym_full.decl`
- `include/BCR_CPP_LA/mat_sym_full.h`

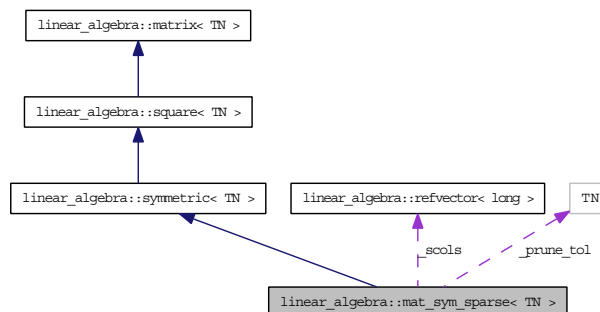
## 4.32 `linear_algebra::mat_sym_sparse< TN >` Class Template Reference

`mat_sym_sparse` packs sparse column matrices into a comfortable format.

Inheritance diagram for `linear_algebra::mat_sym_sparse< TN >`:



Collaboration diagram for `linear_algebra::mat_sym_sparse< TN >`:



## Public Member Functions

- `mat_sym_sparse ()`  
*Default constructor.*
- `mat_sym_sparse (long n)`  
*Constructor of dimension  $n$  with uninitialised values.*
- `mat_sym_sparse (long n, const vector< long > &scols, const vector< sparse_vector< TN > > &vals)`  
*Constructor of dimension  $n$  with initialisation of values.*
- `mat_sym_sparse (long n, const vector< long > &scols, refvector< sparse_vector< TN > > &vals)`  
*Constructor of dimension  $n$  with initialisation of values.*
- `mat_sym_sparse (long n, refvector< long > &scols, refvector< sparse_vector< TN > > &vals)`  
*Constructor of dimension  $n$  with initialisation of values.*
- `mat_sym_sparse (long n, refvector< long > &scols, const vector< sparse_vector< TN > > &vals)`  
*Constructor of dimension  $n$  with initialisation of values.*
- `mat_sym_sparse (istream &IN)`  
*Constructor to read a stored matrix from a stream. Aimed at retrieving information from a binary save with `mat_sym_sparse::operator>>()`.*
- `mat_sym_sparse< TN > & operator= (mat_sym_sparse< TN > &b)`
- `mat_sym_sparse< TN > & operator= (const mat_sym_sparse< TN > &b)`
- `const TN & operator() (const long x, const long y) const`  
*Access operator.*
- `const sparse_vector< TN > operator[] (const long i) const`  
*Direct access operator.*
- `sparse_vector< TN > & operator[] (const long i)`  
*Direct access operator.*
- `void set (const long x, const long y, TN value)`  
*Set operation for safe assignments.*
- `mat_sym_sparse< TN > operator * (const TN &a) const`  
*Multiplies a symmetric sparse matrix with an object.*

- `mat_full< TN > operator * (mat_full< TN > A) const`  
*Multiply by a full matrix.*
- `mat_full< TN > & multiply (mat_full< TN > A, mat_full< TN > &r) const`  
*Multiply by a full matrix.*
- `refvector< TN > & multiply (const refvector< TN > &v, refvector< TN > &r) const`  
*Multiply a symmetric sparse matrix with a full vector into a full vector./test.*
- `refvector< TN > operator * (const refvector< TN > &v) const`  
*Multiply a symmetric sparse matrix with a full vector into a full vector./test.*
- `mat_sym_sparse< TN > & operator *= (const TN &x)`  
*Multiply by an object of TN into left side.*
- `mat_sym_sparse< TN > & multiply (long x, long y, TN a)`  
*Multiplies an element by a number.*
- `mat_full< TN > & multiply (const mat_asym_full< TN > &B, mat_full< TN > &r)`  
`const`  
*Multiply a symmetric sparse matrix with a full vector into a full vector./test.*
- `mat_full< TN > operator * (const mat_asym_full< TN > &B) const`  
*Multiply a symmetric sparse matrix with a full vector into a full vector./test.*
- `mat_sym_sparse & operator /= (const TN &x)`  
*Divide by an object of TN into left side. This is very slow and not advisable.*
- `sparse_vector< TN > operator * (const sparse_vector< TN > &v) const`  
*Multiply a sparse symmetric matrix with a sparse vector.*
- `mat_sparse< TN > operator * (const mat_sparse< TN > &B) const`  
*Multiply a sparse symmetric matrix with a sparse matrix.*
- `mat_sym_sparse< TN > & operator += (const mat_sym_sparse< TN > &B)`  
*Add a matrix into left object.*
- `mat_sym|_sparse< TN > & operator -= (const mat_sym_sparse< TN > &B)`  
*Subtract a matrix into left object.*
- `bool is_col_nonzero (const long x, long &i) const`
- `bool is_nonzero (long x, long y, long &j, long &k) const`  
*Checks whether an access is non zero.*
- `bool add (long x, long y, const TN z)`  
*Adds a value.*

- `bool is_empty () const`
- `long size () const`  
*Returns size of non-zero entries in `_svals`.*
- `void operator>> (ostream &OUT) const`  
*Output matrix in binary format into a stream. Aimed at file storage.*
- `mat_sym_sparse< TN > & prune (const TN x=0)`  
*Deletes all entries that are smaller in magnitude than  $x$ .*
- `TN max_element () const`  
*Finds the maximum element.*
- `bool display () const`  
*Simple display of matrix.*
- `vector< TN > extract_full_col (const long column, const long xmin, const long xmax) const`  
*Extracts a block from a column/row.*
- `vector< TN > extract_full_col (const long column, const long xmin=0) const`  
*Extracts a block from a column/row.*
- `mat_sym_full< TN > extract_full_sym_block (long xmin, long xmax) const`  
*Extracts a full symmetric block.*
- `mat_sym_full< TN > extract_full_sym_block (long xmin=0) const`  
*Extracts a full symmetric block.*
- `mat_sparse< TN > extract_sparse_block (long xmin, long xmax, long ymin, long ymax) const`  
*Extracts a block from anywhere in the matrix.*
- `sparse_vecto< TN > extract_sparse_col (const long column, const long xmin=0) const`  
*Extracts a block from a column/row.*
- `sparse_vector< TN > extract_sparse_lower_col (const long column, const long xmin, const long xmax) const`  
*Extracts a sparse column up to the diagonal.*
- `sparse_vector< TN > extract_sparse_lower_col (const long column, const long xmin=0) const`  
*Extracts a block from a column/row.*

- double density () const  
*Return the density of non-zero elements.*
- void gen\_mat (const sparse\_vector< TN > &a, const sparse\_vector< TN > &b, TN correction)  
*Produce the necessary  $ij^* + ji^*$ .*
- mat\_sparse< TN > householder (vector< TN > &diagonal, vector< TN > &subdiagonal)  
*This routine produces a the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.*
- mat\_sparse< TN > householder (refvector< TN > &diagonal, refvector< TN > &subdiagonal)  
*This routine produces a the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.*
- void clear ()  
*Sets matrix to zero releasing all associated memory.*
- mat\_sym\_sparse< TN > & zero ()  
*Sets matrix to zero releasing all associated memory.*
- long index (long i) const  
*Returns the  $i$ th nonzero column number.*
- mat\_sym\_full< TN > utu (const mat\_full< TN > &U, mat\_sym\_full< TN > &R) const  
*This is the unitary transform  $U^*AU$ .*
- mat\_sym\_full< TN > utu (const mat\_full< TN > &U) const  
*This is the unitary transform  $U^\dagger AU$ .*
- TN trace () const  
*The square matrix class contains all  $n$  by  $n$  matrices.*
- void copy (mat\_sym\_sparse< TN > &b, long i)
- void copy (const mat\_sym\_sparse< TN > &b)
- void copy (mat\_sym\_sparse< TN > &b)
- symmetric< TN > & copy (const symmetric< TN > &a)  
*Copy matrix with depth 1.*

- `symmetric< TN > & copy (const symmetric< TN > &a, const long i)`  
*Copy matrix with depth i.*
- `symmetric< TN > & copy (symmetric< TN > &a, const long i)`  
*Copy matrix with depth i.*

## Public Attributes

- `TN _prune_tol`  
*Tolerance for multiplication with a sparse vector.*

## Private Attributes

- `refvector< long > _scols`  
*Vector of column indices.*
- `refvector< sparse_vector< TN > > _svals`  
*Vector of sparse columns.*
- `bool _pt`  
*State of matrix.*

## Friends

- `class mat_sparse< TN >`
- `class mat_full< TN >`
- `class mat_asym_full< TN >`

### 4.32.1 Detailed Description

**template<class TN> class linear\_algebra::mat\_sym\_sparse< TN >**

The sparse matrix is packed as a sparse vector of sparse vectors. Only the columns which are not empty are indexed and kept in memory. The sparse vector part takes care of the rows.



### 4.32.2 Constructor & Destructor Documentation

4.32.2.1 `template<class TN> linear_algebra::mat_sym_sparse< TN >::mat_sym_sparse ()`

4.32.2.2 `template<class TN> linear_algebra::mat_sym_sparse< TN >::mat_sym_sparse (long n)`

4.32.2.3 `template<class TN> linear_algebra::mat_sym_sparse< TN >::mat_sym_sparse (long n, const vector< long > & scols, const vector< sparse_vector< TN > > & vals)`

#### Parameters:

*n* Dimension of matrix.  
*srows* Vector of row indices.  
*scols* Vector of column indices.  
*vals* Vector of values.

Each value has an associated row and column index in *srows* and *scols*. Therefore they all have the same size. There is no integrity check.

4.32.2.4 `template<class TN> linear_algebra::mat_sym_sparse< TN >::mat_sym_sparse (long n, const vector< long > & scols, refvector< sparse_vector< TN > > & vals)`

#### Parameters:

*n* Dimension of matrix.  
*srows* Vector of row indices.  
*scols* Vector of column indices.  
*vals* reference counted Vector of values.

Each value has an associated row and column index in *srows* and *scols*. Therefore they all have the same size. There is no integrity check.

4.32.2.5 `template<class TN> linear_algebra::mat_sym_sparse< TN >::mat_sym_sparse (long n, refvector< long > & scols, refvector< sparse_vector< TN > > & vals)`

#### Parameters:

*n* Dimension of matrix.  
*srows* Vector of row indices.  
*scols* Reference counted vector of column indices.  
*vals* Reference counted vector of values.

Each value has an associated row and column index in *srows* and *scols*. Therefore they all

have the same size. There is no integrity check.

Here is the call graph for this function:



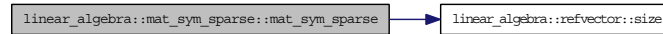
4.32.2.6 template<class TN> linear\_algebra::mat\_sym\_sparse< TN >::mat\_sym\_sparse  
(long *n*, refvector< long > & *scols*, const vector< sparse\_vector< TN > > & *vals*)

### Parameters:

- n* Dimension of matrix.
- srows* Vector of row indices.
- scols* Reference counted vector of column indices.
- vals* Vector of values.

Each value has an associated row and column index in *srows* and *scols*. Therefore they all have the same size. There is no integrity check.

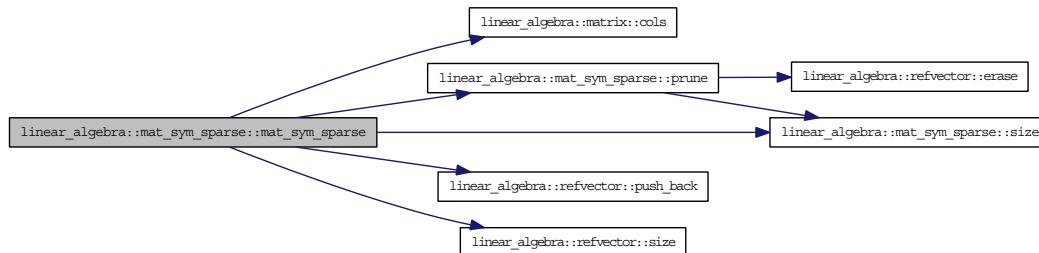
Here is the call graph for this function:



4.32.2.7 template<class TN> linear\_algebra::mat\_sym\_sparse< TN >::mat\_sym\_sparse  
(istream & *IN*)

The resultant vector is pruned to default pruning tolerance = 0. This should be completely superfluous but in case a different program not using this class produces the matrix as input we prune.

Here is the call graph for this function:



### 4.32.3 Member Function Documentation

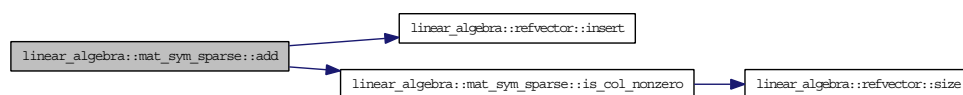
4.32.3.1 `template<class TN> bool linear_algebra::mat_sym_sparse< TN >::add (long  $x$ , long  $y$ , const TN  $z$ )`

#### Parameters:

$z$  is added to  $(x,y)$ . If the value is zero the associated vectors are expanded.

Reimplemented from `linear_algebra::symmetric< TN >`.

Here is the call graph for this function:



4.32.3.2 `template<class TN> void linear_algebra::mat_sym_sparse< TN >::clear ()`

4.32.3.3 `template<class TN> symmetric< TN > & linear_algebra::mat_sym_sparse< TN >::copy (symmetric< TN > &  $a$ , const long  $i$ )[virtual]`

Implements `linear_algebra::symmetric< TN >`.

4.32.3.4 `template<class TN> symmetric< TN > & linear_algebra::mat_sym_sparse< TN >::copy (const symmetric< TN > &  $a$ , const long  $i$ )[virtual]`

Implements `linear_algebra::symmetric< TN >`.

4.32.3.5 `template<class TN> symmetric< TN > & linear_algebra::mat_sym_sparse< TN >::copy (const symmetric< TN > &  $a$ )[virtual]`

Implements `linear_algebra::symmetric< TN >`.

4.32.3.6 `template<class TN> void linear_algebra::mat_sym_sparse< TN >::copy (mat_sym_sparse< TN > &  $b$ )`

4.32.3.7 `template<class TN> void linear_algebra::mat_sym_sparse< TN >::copy (const mat_sym_sparse< TN > &  $b$ )`

4.32.3.8 `template<class TN> void linear_algebra::mat_sym_sparse< TN >::copy (mat_sym_sparse< TN > &  $b$ , long  $i$ )`

4.32.3.9 `template<class TN> double linear_algebra::mat_sym_sparse< TN >::density ()`  
`const`

4.32.3.10 `template<class TN> bool linear_algebra::mat_sym_sparse< TN >::display ()`  
`const`

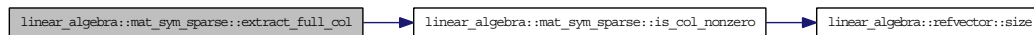
4.32.3.11 `template<class TN> vector< TN > linear_algebra::mat_sym_sparse< TN`  
`>::extract_full_col (const long column, const long xmin = 0) const`

Here is the call graph for this function:



4.32.3.12 `template<class TN> vector< TN > linear_algebra::mat_sym_sparse< TN`  
`>::extract_full_col (const long column, const long xmin, const long xmax) const`

Here is the call graph for this function:



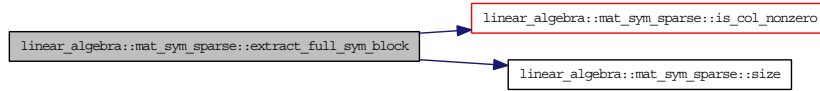
4.32.3.13 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_sparse< TN`  
`>::extract_full_sym_block (long xmin = 0) const`

Here is the call graph for this function:



4.32.3.14 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_sparse< TN`  
`>::extract_full_sym_block (long xmin, long xmax) const`

Here is the call graph for this function:

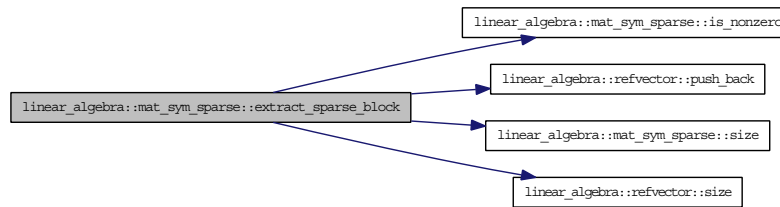


4.32.3.15 `template<class TN> mat_sparse< TN > linear_algebra::mat_sym_sparse< TN >::extract_sparse_block (long xmin, long xmax, long ymin, long ymax) const`

#### Test

should work, but ...

Here is the call graph for this function:



4.32.3.16 `template<class TN> sparse_vector< TN > linear_algebra::mat_sym_sparse< TN >::extract_sparse_col (const long column, const long xmin = 0) const`

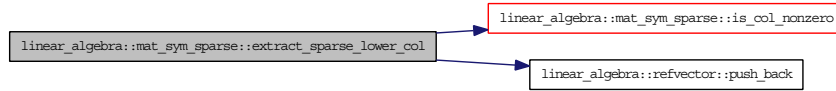
4.32.3.17 `template<class TN> sparse_vector< TN > linear_algebra::mat_sym_sparse< TN >::extract_sparse_lower_col (const long column, const long xmin = 0) const`

Here is the call graph for this function:



4.32.3.18 `template<class TN> sparse_vector< TN > linear_algebra::mat_sym_sparse< TN >::extract_sparse_lower_col (const long column, const long xmin, const long xmax) const`

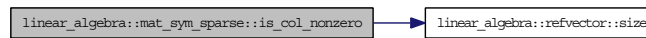
Here is the call graph for this function:



4.32.3.19 `template<class TN> long linear_algebra::mat_sym_sparse< TN >::index (long i) const`

4.32.3.20 `template<class TN> bool linear_algebra::mat_sym_sparse< TN >::is_col_nonzero (const long x, long & i) const`

Here is the call graph for this function:



4.32.3.21 `template<class TN> bool linear_algebra::mat_sym_sparse< TN >::is_empty () const`

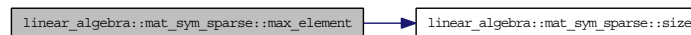
4.32.3.22 `template<class TN> bool linear_algebra::mat_sym_sparse< TN >::is_nonzero (long x, long y, long & j, long & k) const`

### Parameters:

*j* holds the position of (x,y) in `_vals`. If (x,y) is 0. An insertion would happen here.

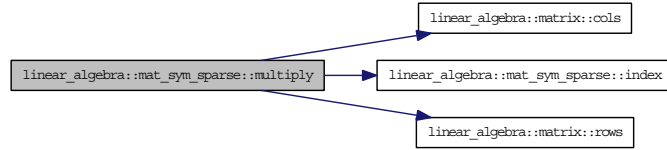
4.32.3.23 `template<class TN> TN linear_algebra::mat_sym_sparse< TN >::max_element () const`

Here is the call graph for this function:



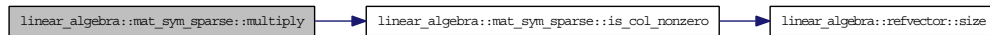
4.32.3.24 `template<class TN> mat_full< TN > & linear_algebra::mat_sym_sparse< TN >::multiply (const mat_asym_full< TN > & B, mat_full< TN > & const`

Here is the call graph for this function:



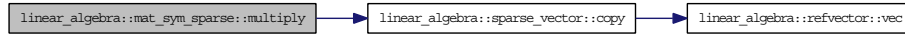
4.32.3.25 `template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse< TN >::multiply (long x, long y, TN a)`

Here is the call graph for this function:



4.32.3.26 `template<class TN> refvector< TN > & linear_algebra::mat_sym_sparse< TN >::multiply (const refvector< TN > & v, refvector< TN > & r) const`

Here is the call graph for this function:

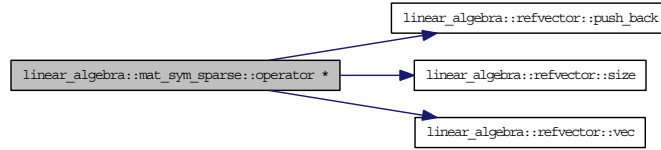


4.32.3.27 `template<class TN> mat_full< TN > &linear_algebra::mat_sym_sparse< TN >::multiply (mat_full< TN > A, mat_full< TN > & r) const`

## Test

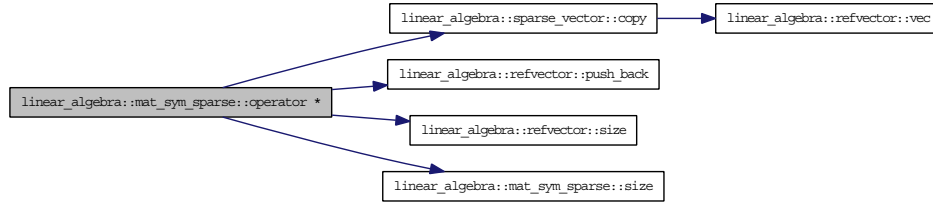
4.32.3.28 `template<class TN> mat_sparse< TN > linear_algebra::mat_sym_sparse< TN >::operator * (const mat_sparse< TN > & B) const`

Here is the call graph for this function:



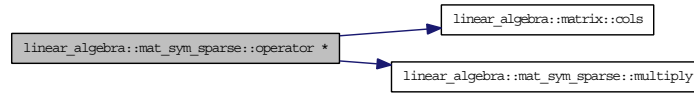
4.32.3.29 `template<class TN> sparse_vector< TN > linear_algebra::mat_sym_sparse< TN >::operator * (const sparse_vector< TN > & v) const`

Here is the call graph for this function:



4.32.3.30 `template<class TN> mat_full< TN > linear_algebra::mat_sym_sparse< TN >::operator * (const mat_asym_full< TN > & B) const`

Here is the call graph for this function:



4.32.3.31 `template<class TN> refvector< TN > linear_algebra::mat_sym_sparse< TN >::operator * (const refvector< TN > & v) const`

Here is the call graph for this function:

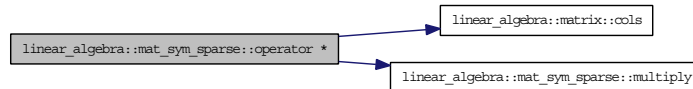




4.32.3.32 `template<class TN> mat_full< TN > linear_algebra::mat_sym_sparse< TN >::operator * (mat_full< TN > A) const`

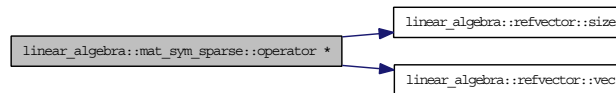
### Test

Here is the call graph for this function:



4.32.3.33 `template<class TN> mat_sym_sparse< TN > linear_algebra::mat_sym_sparse< TN >::operator * (const TN & a) const`

Here is the call graph for this function:



4.32.3.34 `template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse< TN >::operator *= (const TN & x)`

4.32.3.35 `template<class TN> const TN & linear_algebra::mat_sym_sparse< TN >::operator() (const long x, const long y) const[virtual]`

Implements `linear_algebra::matrix< TN >`.

Here is the call graph for this function:



4.32.3.36 `template<class TN> mat_sym_sparse<TN>& linear_algebra::mat_sym_sparse<TN >::operator+=(const mat_sym_sparse< TN > & B)`

4.32.3.37 `template<class TN> mat_sym_sparse<TN>& linear_algebra::mat_sym_sparse<TN >::operator-=(const mat_sym_sparse< TN > & B)`

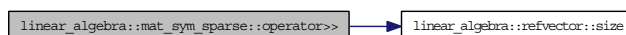
4.32.3.38 `template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse<TN >::operator/=(const TN & x)`

4.32.3.39 `template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse<TN >::operator=(const mat_sym_sparse< TN > & b)`

4.32.3.40 `template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse<TN >::operator=(mat_sym_sparse< TN > & b)`

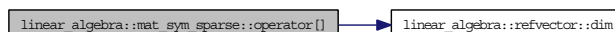
4.32.3.41 `template<class TN> void linear_algebra::mat_sym_sparse< TN >::operator>>(ostream & OUT) const`

Here is the call graph for this function:



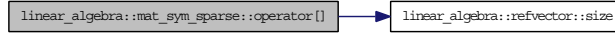
4.32.3.42 `template<class TN> sparse_vector< TN > & linear_algebra::mat_sym_sparse<TN >::operator[] (const long i)`

Here is the call graph for this function:



4.32.3.43 `template<class TN> const sparse_vector< TN > linear_algebra::mat_sym_sparse< TN >::operator[] (const long i) const`

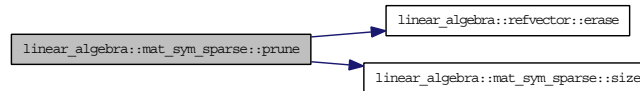
Here is the call graph for this function:



4.32.3.44 `template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse< TN >::prune (const TN x = 0)`

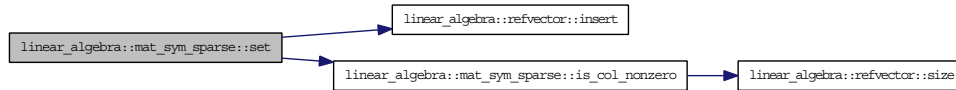
Reimplemented from `linear_algebra::symmetric< TN >`.

Here is the call graph for this function:



4.32.3.45 `template<class TN> void linear_algebra::mat_sym_sparse< TN >::set (const long x, const long y, TN value)`

Here is the call graph for this function:



4.32.3.46 `template<class TN> long linear_algebra::mat_sym_sparse< TN >::size () const`

4.32.3.47 `template<class TN> TN linear_algebra::mat_sym_sparse< TN >::trace () const`

Reimplemented from `linear_algebra::square< TN >`.

4.32.3.48 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_sparse< TN >::utu (const mat_full< TN > & U) const`

4.32.3.49 `template<class TN> mat_sym_full< TN > linear_algebra::mat_sym_sparse< TN >::utu (const mat_full< TN > & U, mat_sym_full< TN > & R) const`

4.32.3.50 `template<class TN> mat_sym_sparse< TN > & linear_algebra::mat_sym_sparse< TN >::zero ()`

## 4.32.4 Friends and Related Function Documentation

4.32.4.1 `template<class TN> friend class mat_asym_full< TN >[friend]`

4.32.4.2 `template<class TN> friend class mat_full< TN >[friend]`

4.32.4.3 `template<class TN> friend class mat_sparse< TN >[friend]`

### 4.32.5 Member Data Documentation

4.32.5.1 `template<class TN> bool linear_algebra::mat_sym_sparse< TN >::_empty[private]`

4.32.5.2 `template<class TN> TN linear_algebra::mat_sym_sparse< TN >::_prune_tol`

4.32.5.3 `template<class TN> refvector<long> linear_algebra::mat_sym_sparse< TN >::_scols[private]`

4.32.5.4 `template<class TN> refvector<sparse_vector<TN> > linear_algebra::mat_sym_sparse< TN >::_svals[private]`

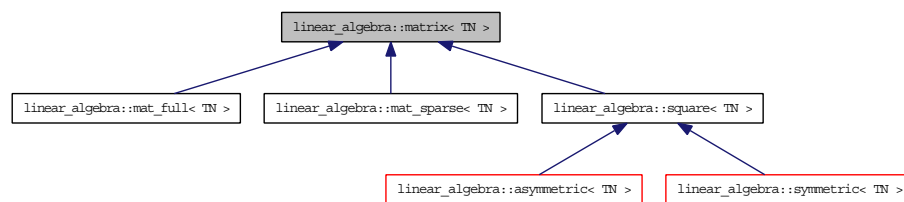
The documentation for this class was generated from the following files:

- `include/BCR_CPP_LA/mat_sym_sparse.decl`
- `include/BCR_CPP_LA/mat_sym_sparse.h`

### 4.33 `linear_algebra::matrix< TN >` Class Template Reference

`matrix` is a basic class which covers all kinds of matrices.

Inheritance diagram for `linear_algebra::matrix< TN >`:



### Public Member Functions

- `matrix ()`  
*Default constructor.*
- `virtual ~matrix ()`  
*Default destructor.*

- virtual void operator>> (std::ostream &OUT) const  
*Output matrix in binary format into a stream. Aimed at file storage.*
- virtual const TN & operator() (const long i, const long j) const =0
- long rows () const  
*Returns number of rows.*
- long cols () const  
*Returns number of columns.*
- long size ()  
*Returns the number of entries.*
- matrix< TN > & operator= (const matrix< TN > &b)
- TN trace () const  
*We demand that there is a zero function.*

## Protected Attributes

- long \_rows  
*Number of rows.*
- long \_cols  
*Number of columns.*
- long \_size  
*Size of std::vector<TN> \_vals.*

**template<class TN> class linear\_algebra::matrix< TN >**

### 4.33.1 Constructor & Destructor Documentation

4.33.1.1 template<class TN> linear\_algebra::matrix< TN >::matrix ()

matrix is a basic class which covers all kinds of matrices. Default constructor.

4.33.1.2 template<class TN> virtual linear\_algebra::matrix< TN >::~~matrix ()[inline, virtual]

## 4.33.2 Member Function Documentation

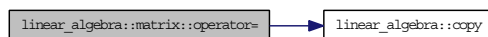
4.33.2.1 `template<class TN> long linear_algebra::matrix< TN >::cols () const`

4.33.2.2 `template<class TN> virtual const TN& linear_algebra::matrix< TN >::operator()  
(const long i, const long j) const[pure virtual]`

Implemented in `linear_algebra::mat_asym_full< TN >`, `linear_algebra::mat_asym_sparse< TN >`, `linear_algebra::mat_full< TN >`, `linear_algebra::mat_sparse< TN >`, `linear_algebra::mat_sym_full< TN >`, `linear_algebra::mat_sym_sparse< TN >`, `linear_algebra::mat_full< double >`, `linear_algebra::mat_full< bool >`, and `linear_algebra::mat_sym_full< valerg >`.

4.33.2.3 `template<class TN> matrix< TN > & linear_algebra::matrix< TN >::operator=  
(const matrix< TN > & b)`

Here is the call graph for this function:



4.33.2.4 `template<class TN> virtual void linear_algebra::matrix< TN >::operator>>  
(std::ostream & OUT) const[inline, virtual]`

The default is to output a full (column) matrix.

Reimplemented in `linear_algebra::mat_asym_full< TN >`.

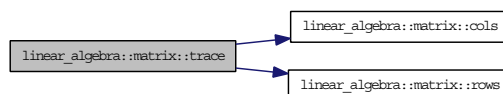
4.33.2.5 `template<class TN> long linear_algebra::matrix< TN >::rows () const`

4.33.2.6 `template<class TN> long linear_algebra::matrix< TN >::size ()`

4.33.2.7 `template<class TN> TN linear_algebra::matrix< TN >::trace () const`

Reimplemented in `linear_algebra::mat_sym_sparse< TN >`, `linear_algebra::square< TN >`, and `linear_algebra::square< valerg >`.

Here is the call graph for this function:



### 4.33.3 Member Data Documentation

4.33.3.1 `template<class TN> long linear_algebra::matrix< TN >::_cols[protected]`

4.33.3.2 `template<class TN> long linear_algebra::matrix< TN >::_rows[protected]`

4.33.3.3 `template<class TN> long linear_algebra::matrix< TN >::_size[protected]`

The documentation for this class was generated from the following files:

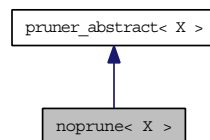
- `include/BCR_CPP_LA/matrix.decl`
- `include/BCR_CPP_LA/matrix.h`

### 4.34 `noprune< X >` Class Template Reference

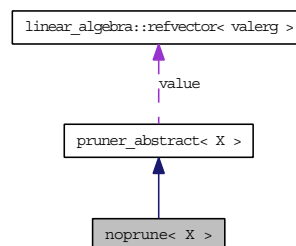
Dummy pruning class. No actual pruning done.

```
#include <noprune.h>
```

Inheritance diagram for `noprune< X >`:



Collaboration diagram for `noprune< X >`:



### Public Member Functions

- `noprune()`

- `noprune (const noprune< X > &a)`
- `void adjust_lagrange (double &lambda, ulong &conf1, ulong &conf2, long &config, const refvector< ulong > &visited_run) const`  
*Compute the increase of lambda and assess current best value.*
- `ulong prune (double &lambda, ulong &conf1, ulong &conf2, long &config, const refvector< ulong > &a) const`
- `ulong deprune (ulong N const`  
*self-explanatory*
- `ulong reprune (ulong N) const`  
*Given an "absolute" reference number return the current reference number.*

**template<class X> class noprune< X >**

#### 4.34.1 Constructor & Destructor Documentation

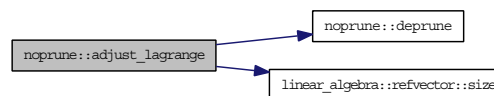
4.34.1.1 `template<class X> noprune< X >::noprune () [inline]`

4.34.1.2 `template<class X> noprune< X >::noprune (const noprune< X > & a) [inline]`

#### 4.34.2 Member Function Documentation

4.34.2.1 `template<class X> void noprune< X >::adjust_lagrange (double & lambda, ulong & conf1, ulong & conf2, long & config, const refvector< ulong > & visited_run) const [inline]`

Here is the call graph for this function:



4.34.2.2 `template<class X> ulong noprune< X >::deprune (ulong N) const [inline, virtual]`

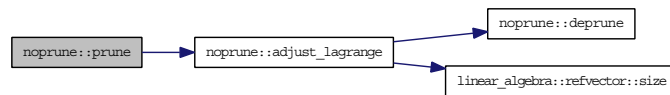
Implements `pruner_abstract< X >`.

4.34.2.3 `template<class X> ulong noprune< X >::prune (double & lambda, ulong & conf1, ulong & conf2, long & config, const refvector< ulong > & a) const [inline, virtual]`



Implements `pruner_abstract< X >`.

Here is the call graph for this function:



4.34.2.4 `template<class X> ulong noprune< X >::reprune (ulong N) const[inline, virtual]`

Implements `pruner_abstract< X >`.

The documentation for this class was generated from the following file:

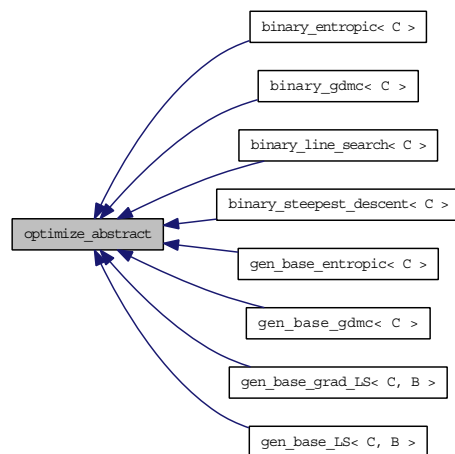
- `noprune.h`

## 4.35 `optimize_abstract` Class Reference

Abstract class that implements the `Concepts::has_optimize` concept. Only good for generic referncing of optimizations.

`#include <optimizeabstract.h>`

Inheritance diagram for `optimize_abstract`:



## Public Member Functions

- `virtual ~optimize_abstract ()`

- `ulong optimize () const`  
*Default optimize.*
- `virtual ulong optimize (const ulong N) const =0`
- `optimize_abstract ()`
- `optimize_abstract (const optimize_abstract &a)`
- `void set_id (const string &ID) const`

## Public Attributes

- `const string & id_r`

## Private Attributes

- `string id`  
*Identifier string;.*

### 4.35.1 Constructor & Destructor Documentation

4.35.1.1 `virtual optimize_abstract::~~optimize_abstract ()`text[inline, virtual]

4.35.1.2 `optimize_abstract::optimize_abstract ()`[inline]

4.35.1.3 `optimize_abstract::optimize_abstract (const optimize_abstract & a)`[inline]

### 4.35.2 Member Function Documentation

4.35.2.1 `virtual ulong optimize_abstract::optimize (const ulong N) const`[pure virtual]

Implemented in `binary_entropic< C >`, `binary_line_search< C >`, `binary_gdmc< C >`, `binary_steepest_descent< C >`, `gen_base_entropic< C >`, `gen_base_grad_LS< C, B >`, `gen_base_LS< C, B >`, and `gen_base_gdmc< C >`.

4.35.2.2 `ulong optimize_abstract::optimize () const`[inline]

4.35.2.3 `void optimize_abstract::set_id (const string & ID) const`[inline]

### 4.35.3 Member Data Documentation

4.35.3.1 string optimize\_abstract::id[mutable, private]

4.35.3.2 const string& optimize\_abstract::id\_r

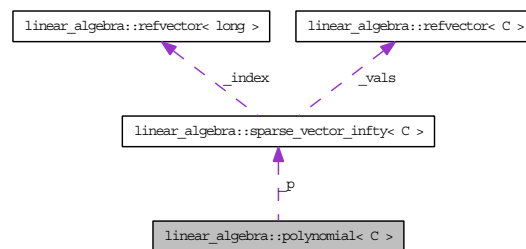
The documentation for this class was generated from the following file:

- optimizeabstract.h

### 4.36 linear\_algebra::polynomial< C > Class Template Reference

A class for polynomials.

Collaboration diagram for linear\_algebra::polynomial< C >:



#### Public Member Functions

- `polynomial ()`  
*Standard constructor.*
- `polynomial< C > & operator= (const polynomial< C > &a)`  
*Assignment operator.*
- `polynomial< C > & operator= (polynomial< C > &a)`  
*Assignment operator.*
- `polynomial< C > & zero ()`  
*Define a zero.*
- `long index (const long i) const`  
*Get the exponnent of the ith element.*

- `long size () const`  
*return the number of nonzeros.*

## Multiplications

- `polynomial< C > & multiply (const polynomial< C > &a, polynomial< C > &r)`  
`const`  
*Multiply two polynomials.*
- `polynomial< C > operator * (const polynomial< C > &a) const`  
*Multiply two polynomials.*
- `polynomial< C > operator *= (const polynomial< C > &a)`  
*Multiply two polynomials.*

## Additions

- `polynomial< C > operator+ (const polynomial< C > &a) const`  
*Add two polynomials.*
- `polynomial< C > & operator+= (const polynomial< C > &a)`  
*Add two polynomials.*

## Access

- `const C & operator() (const long i) const`  
*Acess an element.*
- `C & operator() (const long i)`  
*Acess an element.*

## Initialisers

- `void set (const long i, const C &a)`  
*The ith element to a.*
- `void set (const long i, C &a)`  
*The ith element to a.*

## Direct Acces operators

- `C operator[] (const long i) const`
- `C & operator[] (const long i)`

## Copy operations.

- `polynomial< C > & copy (const polynomial< C > &a)`  
*Copy assignment.*
- `polynomial< C > & copy (polynomial< C > &a)`  
*Copy assignment.*
- `polynomial< C > & copy (const polynomial< C > &a, const long i)`  
*Copy assignment.*
- `polynomial< C > & copy (polynomial< C > &a, const long i)`  
*Copy assignment.*

## Private Attributes

- `sparse_vector_inft< C > _`

`template<class C> class linear_algebra::polynomial< C >`

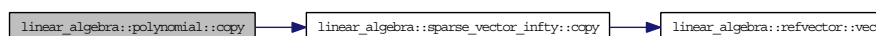
### 4.36.1 Constructor & Destructor Documentation

4.36.1.1 `template<class C> linear_algebra::polynomial< C >::polynomial ()`

### 4.36.2 Member Function Documentation

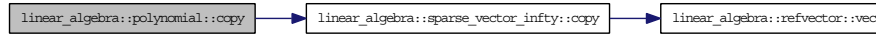
4.36.2.1 `template<class C> polynomial< C > & linear_algebra::polynomial< C >::copy (polynomial< C > & a, const long i)`

Here is the call graph for this function:



4.36.2.2 `template<class C> polynomial< C > & linear_algebra::polynomial< C >::copy (const polynomial< C > & a, const long i)`

Here is the call graph for this function:



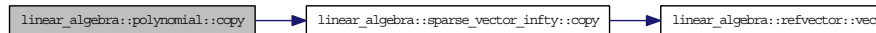
4.36.2.3 `template<class C> polynomial< C > & linear_algebra::polynomial< C >::copy (polynomial< C > & a)`

Here is the call graph for this function:



4.36.2.4 `template<class C> polynomial< C > & linear_algebra::polynomial< C >::copy (const polynomial< C > & a)`

Here is the call graph for this function:



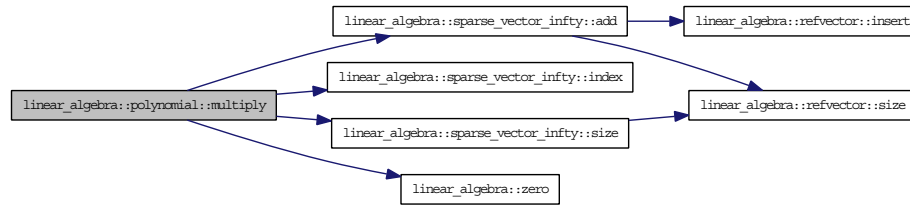
4.36.2.5 `template<class C> long int linear_algebra::polynomial< C >::index (const long i) const`

Here is the call graph for this function:



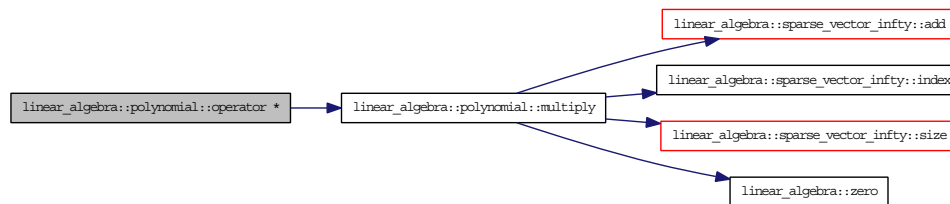
4.36.2.6 `template<class C> polynomial< C > & linear_algebra::polynomial< C >::multiply (const polynomial< C > & a, polynomial< C > & r) const`

Here is the call graph for this function:



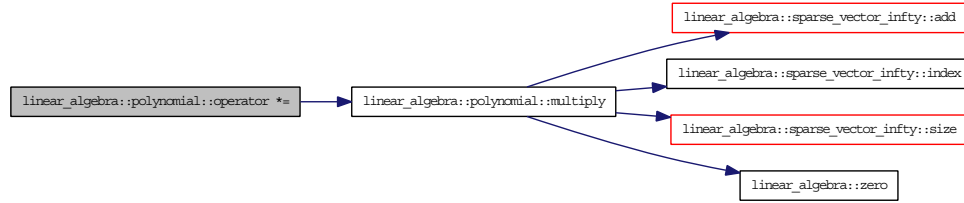
4.36.2.7 `template<class C> polynomial< C > linear_algebra::polynomial< C >::operator * (const polynomial< C > & a) const`

Here is the call graph for this function:



4.36.2.8 `template<class C> polynomial< C > linear_algebra::polynomial< C >::operator *= (const polynomial< C > & a)`

Here is the call graph for this function:

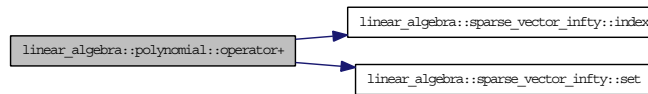


4.36.2.9 `template<class C> C & linear_algebra::polynomial< C >::operator() (const long i)`

4.36.2.10 `template<class C> const C & linear_algebra::polynomial< C >::operator() (const long i) const`

4.36.2.11 `template<class C> polynomial< C > linear_algebra::polynomial< C >::operator+ (const polynomial< C > & a) const`

Here is the call graph for this function:



4.36.2.12 `template<class C> polynomial< C > & linear_algebra::polynomial< C >::operator+= (const polynomial< C > & a)`

4.36.2.13 `template<class C> polynomial< C > & linear_algebra::polynomial< C >::operator= (polynomial< C > & a)`

4.36.2.14 `template<class C> polynomial< C > & linear_algebra::polynomial< C >::operator= (const polynomial< C > & a)`

4.36.2.15 `template<class C> C & linear_algebra::polynomial< C >::operator[] (const long i)`

4.36.2.16 `template<class C> C linear_algebra::polynomial< C >::operator[] (const long i) const`

4.36.2.17 `template<class C> void linear_algebra::polynomial< C >::set (const long i, C & a)`

Here is the call graph for this function:





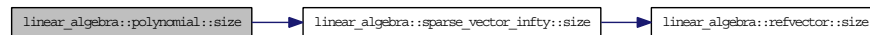
4.36.2.18 `template<class C> void linear_algebra::polynomial< C >::set (const long i, const C & a)`

Here is the call graph for this function:



4.36.2.19 `template<class C> long int linear_algebra::polynomial< C >::size () const`

Here is the call graph for this function:



4.36.2.20 `template<class C> polynomial< C > & linear_algebra::polynomial< C >::zero ()`

Here is the call graph for this function:



### 4.36.3 Member Data Documentation

4.36.3.1 `template<class C> sparse_vector_infty<C> linear_algebra::polynomial< C >::p[private]`

The documentation for this class was generated from the following files:

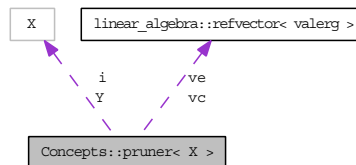
- `include/BCR_CPP_LA/polynomial.decl`
- `include/BCR_CPP_LA/polynomial.h`

### 4.37 Concepts::pruner< X > Class Template Reference

Concept defines an addressable set of values.

```
#include <concepts.hh>
```

Collaboration diagram for Concepts::pruner< X >:



#### Public Member Functions

- `BOOST_CONCEPT_USAGE (pruner)`

#### Private Member Functions

- `BOOST_CONCEPT_ASSERT ((Library< X >))`

#### Private Attributes

- `X & i`

- `const X & Y`
- `refvector< ulong > v`
- `refvector< valerg >ve`
- `const refvector< ulong > vd`
- `const refvector< valerg > vc`

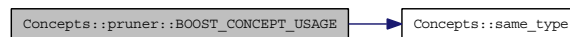
**template<class X> class Concepts::pruner< X >**

#### 4.37.1 Member Function Documentation

4.37.1.1 `template<class X> Concepts::pruner< X >::BOOST_CONCEPT_ASSERT  
((Library< X >))[private]`

4.37.1.2 `template<class X> Concepts::pruner< X >::BOOST_CONCEPT_USAGE  
(pruner< X >)[inline]`

Here is the call graph for this function:



#### 4.37.2 Member Data Documentation

4.37.2.1 `template<class X> X& Concepts::pruner< X >::i[private]`

4.37.2.2 `template<class X> refvector<ulong> Concepts::pruner< X >::v[private]`

4.37.2.3 `template<class X> const refvector<valerg> Concepts::pruner< X >::vc[private]`

4.37.2.4 `template<class X> const refvector<ulong> Concepts::pruner< X >::vd[private]`

4.37.2.5 `template<class X> refvector<valerg> Concepts::pruner< X >::ve[private]`

4.37.2.6 `template<class X> const X& Concepts::pruner< X >::Y[private]`

The documentation for this class was generated from the following file:

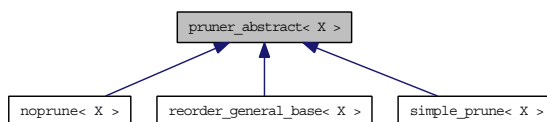
- `concepts.hh`

## 4.38 pruner\_abstract< X > Class Template Reference

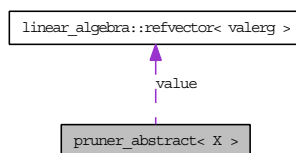
Abstract class implementing the Concepts::pruner concept.

```
#include <prunerabstract.h>
```

Inheritance diagram for pruner\_abstract< X >:



Collaboration diagram for pruner\_abstract< X >:



### Public Member Functions

- `pruner_abstract()`  
*Default constructor.*
- `pruner_abstract(const pruner_abstract< X > &a)`  
*Copy constructor.*
- `valerg compute_property(ulong N) const`  
*Wrap the Library::compute\_property to exclude pruned access.*
- `ulong get_space_size() const`  
*Adjust the space\_size to reflect the absence of pruned values.*
- `ulong get_bits() const`  
*Compute the bits of the adjusted space.*
- `valerg get_value(ulong N) const`  
*Same as compute\_property(ulong N), but absolute numbering.*
- `virtual ulong prune(double &lambda, ulong &conf1, ulong &conf2, long &config, const refvector< ulong > &visit) const =0`

- `ulong prune (double &lambda, ulong &conf1, ulong &conf2, long &config) const`
- `virtual ulong reprune (ulong N) const =0`  
*Given an "absolute" reference number return the current reference number.*
- `virtual ulong deprune (ulong N) const =0`  
*self-explanatory*

## Protected Attributes

- `refvector< ulong > pruned_visited`  
*List of pruned indices in Library.*
- `refvector< ulong > & visited`  
*Reference to Library::visited.*
- `refvector< valerg > & value`  
*Reference to Library::value.*
- `bool bits_computed`
- `bool space_size_computed`
- `ulong space_size`
- `ulong bits`

## Private Member Functions

- `BOOST_CONCEPT_ASSERT ((Concepts::Library< X >))`

`template<class X> class pruner_abstract< X >`

### 4.38.1 Constructor & Destructor Documentation

4.38.1.1 `template<class X> pruner_abstract< X >::pruner_abstract ()[inline]`

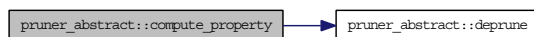
4.38.1.2 `template<class X> pruner_abstract< X >::pruner_abstract (const pruner_abstract< X > & a)[nline]`

## 4.38.2 Member Function Documentation

4.38.2.1 `template<class X> pruner_abstract< X >::BOOST_CONCEPT_ASSERT  
((Concepts::Library< X >))[private]`

4.38.2.2 `template<class X> valerg pruner_abstract< X >::compute_property (ulong N)  
const[inline]`

Here is the call graph for this function:



4.38.2.3 `template<class X> virtual ulong pruner_abstract< X >::deprune (ulong N)  
const[pure virtual]`

Implemented in `noprune< X >`, `reorder_general_base< X >`, and `simple_prune< X >`.

4.38.2.4 `template<class X> ulong pruner_abstract< X >::get_bits () const[inline]`

Here is the call graph for this function:



4.38.2.5 `template<class X> ulong pruner_abstract< X >::get_space_size () const[inline]`

4.38.2.6 `template<class X> valerg pruner_abstract< X >::get_value (ulong N) const[inline]`

4.38.2.7 `template<class X> ulong pruner_abstract< X >::prune (double & lambda, ulong  
& conf1, ulong & conf2, long & config) const[inline]`

Here is the call graph for this function:



4.38.2.8 `template<class X> virtual ulong pruner_abstract< X >::prune (double & lambda,  
ulong & conf1, ulong & conf2, long & config, const refvector< ulong > & visit) const[pure  
virtual]`

Implemented in `noprune< X >`, `reorder_general_base< X >`, and `simple_prune< X >`.

4.38.2.9 `template<class X> virtual ulong pruner_abstract< X >::reprune (ulong N)`  
`const[pure virtual]`

Implemented in `noprune< X >`, `reorder_general_base< X >`, and `simple_prune< X >`.

### 4.38.3 Member Data Documentation

4.38.3.1 `template<class X> ulong pruner_abstract< X >::bits[mutable, protected]`

4.38.3.2 `template<class X> bool pruner_abstract< X >::bits_computed[mutable, protected]`

4.38.3.3 `template<class X> refvector<ulong> pruner_abstract< X >::pruned_visited[mutable, protected]`

4.38.3.4 `template<class X> ulong pruner_abstract< X >::space_size[mutable, protected]`

4.38.3.5 `template<class X> bool pruner_abstract< X >::space_size_computed[mutable, protected]`

4.38.3.6 `template<class X> refvector<valerg>& pruner_abstract< X >::value[mutable, protected]`

4.38.3.7 `template<class X> refvector<ulong>& pruner_abstract< X >::visited[mutable, protected]`

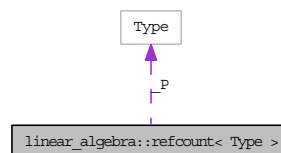
The documentation for this class was generated from the following file:

- `prunerabstract.h`

## 4.39 `linear_algebra::refcount< Type >` Class Template Reference

Class `refcount` provides reference counted pointers. Most basic use.

Collaboration diagram for `linear_algebra::refcount< Type >`:



Public Member Functions

- `refcount ()`  
*Default constructor;.*
- `refcount (Type *l)`  
*Constructor from a pointer.*
- `refcount (const refcount< Type > &rhs)`  
*Copy constructor, but no deep copy.*
- `refcount (refcount< Type > &rhs)`  
*Copy constructor, but no deep copy.*
- `~refcount ()`  
*Destructor checks for reference count and deletes if necessary.*
- `refcount< Type > & operator= (refcount< Type > &rhs)`  
*Assignment without deep copy. Old information is discarded if no references remain.*
- `operator Type * ()`  
*Let refcount behave like any other pointer.*
- `operator const Type * () const`

## Private Attributes

- `unsigned long * _refcount`  
*Reference count.*
- `Type * _P`  
*Data pointer.*

**template<typename Type> class linear\_algebra::refcount< Type >**

### 4.39.1 Constructor & Destructor Documentation

4.39.1.1 `template<class Type> linear_algebra::refcount< Type >::refcount ()`

4.39.1.2 `template<class Type> linear_algebra::refcount< Type >::refcount (Type * l)`

4.39.1.3 `template<class Type> linear_algebra::refcount< Type >::refcount (const refcount< Type > & rhs)`



4.39.1.4 `template<class Type> linear_algebra::refcount< Type >::refcount (refcount< Type > & rhs)`

4.39.1.5 `template<class Type> linear_algebra::refcount< Type >::~~refcount ()`

## 4.39.2 Member Function Documentation

4.39.2.1 `template<class Type> linear_algebra::refcount< Type >::operator const Type * () const`

4.39.2.2 `template<class Type> linear_algebra::refcount< Type >::operator Type * ()`

4.39.2.3 `template<class Type> refcount< Type > & linear_algebra::refcount< Type >::operator= (refcount< Type > & rhs`

## 4.39.3 Member Data Documentation

4.39.3.1 `template<typename Type> Type* linear_algebra::refcount< Type >::_P[private]`

4.39.3.2 `template<typename Type> unsigned long* linear_algebra::refcount< Type >::_refcount[private]`

The documentation for this class was generated from the following files:

- `include/BCR_CPP_LA/refcount.decl`
- `include/BCR_CPP_LA/refcount.h`

## 4.40 `linear_algebra::refvector< TN >` Class Template Reference

Class `refvector` provides reference counted vectors.

### *Public Member Functions*

- `~refvector ()`  
*Destructor checks for reference count and deletes if necessary.*
- `void clear ()`
- `bool operator== (const refvector< TN > &a) const`  
*Compares two refvectors.*

- `template<class T> bool same (const refvector< T > &a) const`  
*Determines if two refvectors are exactly the same.*
- `refvector< TN > & operator= (const refvector< TN > &rhs)`  
*Assignment with necessary deep copy. Old information is discarded if no references remain.*
- `refvector< TN > & operator= (refvector< TN > &rhs)`  
*Assignment without deep copy. Old information is discarded if no references remain.*
- `refvector< TN > & operator= (const std::vector< TN > &rhs)`
- `refvector< TN > & set (long i, TN &rhs)`
- `void operator>> (std::ostream &o) const`
- `refvector< TN > & operator *= (const refvector< TN > &a)`  
*Multiply by a refvector.*
- `refvector< TN > element_prod (const refvector< TN > &a) const`  
*Multiply by a refvector.*
- `TN operator * (const refvector< TN > &b) const`  
*Scalar product of refvectors.*
- `template<class TN3, class TN2> TN3 & multiply (const refvector< TN2 > &b, TN3 &r) const`  
*Scalar product of refvectors.*
- `template<class TN3, class TN2> TN3 operator * (const refvector< TN2 > &b) const`  
*Scalar product of refvectors.*
- `refvector< TN > & operator+= (const refvector< TN > &B)`  
*Add a refvector.*
- `refvector< TN > operator+ (const refvector< TN > &B) const`  
*Add two refvectors.*
- `refvector< TN > & operator-= (const refvector< TN > &B)`  
*Subtract a refvector.*
- `refvector< TN > operator- (const refvector< TN > &B) const`  
*Subtract two refvectors.*
- `refvector< TN > & subtract (const refvector< TN > &B, refvector< TN > &r) const`  
*Subtract two refvectors.*

- `refvector< TN > & operator+= (const sparse_vector< TN > &a)`  
*Add a sparse vector.*
- `refvector< TN > & operator *= (const TN x)`  
*Multiply by an element.*
- `template<class TN2> refvector< TN > operator *= (const TN2 &a)`  
*Multiply by an element into the vector.*
- `template<class TN2> refvector< TN > operator * (const TN2 x) const`  
*Multiply by an element.*
- `template<class TN2> refvector< TN > & multiply (const TN2 x, refvector< TN > &a) const`  
*Multiply by an element.*
- `template<class TN2> refvector< TN > operator/ (const TN2 x) const`  
*Divide by an element.*
- `void display (ostream &outs=cout) const`  
*Simple display of refvector.*
- `TN max () const`  
*Returns the maximum element.*
- `void normalise ()`  
*Normalise the vector in accordance with the  $\| \cdot \|_2$  2-norm.*
- `TN & operator[] (long i)`  
*Direct access operator.*
- `const TN & operator[] (long i) const`  
*Direct access operator.*
- `std::vector< TN > * operator → ()`  
*Dereferencing operator.*
- `const std::vector< TN > * operator → () const`  
*Dereferencing operator.*
- `std::vector< TN > & vec ()`  
*Return the vector.*
- `const std::vector< TN > & vec () const`  
*Return the vector.*

- `refvector< TN > & copy (const refvector< TN > &i, const long j)`  
*Do a copy of depth j of i into a referenced counter. /item refvector< TN > & copy (const refvector< TN > &i)*  
*Copy i with full depth.*
- `long dim () const`  
*Returns the dimension of the vector.*
- `long size () const`  
*Returns the length of the vector.*
- `refvector< TN > & push_back (const TN &a)`
- `void insert (long n, const TN &a)`
- `void resize (long n, const TN &a)`
- `void resize (long n)`
- `void erase (typename std::vector< TN >::iterator a)`
- `void write_to (std::ostream &OUTFILE)`  
*Write a refvector to a stream.*
- `refvector< TN > & zero ()`  
*Set the refvector to the zero vector of current dimension.*
- `long contains (const TN &a) const`  
*Determines whether the refvector contains a certain element and returns its first occurrence position.*
- `refvector< TN > & concat (const refvector< TN > &a)`  
*Appends another refvector to the end.*

## Constructors

- `refvector ()`  
*Dereferencing operator.*
- `refvector (const refvector< TN > &rhs)`  
*Copy constructor necessarily deep to first level.*
- `refvector (refvector< TN > &rhs)`  
*Copy constructor, but no deep copy.*

- `refvector (long i)`  
*Initialise the vector to have i entries.*
- `refvector (const std::vector< TN > &i)`  
*Do a deep copy of i into a referenced counter.*
- `refvector (long j, const TN *rhs)`  
*Do a deep copy of i into a referenced counter.*
- `refvector (std::istream &IN)`  
*Read a refvector from a stream as writexttten by refvector::write\_to().*

## Public Attributes

- `char _deli`

## Private Member Functions

- `const refvector< TN > & copy (const refvector< TN > &r) const`  
*Copy i with depth 1.*
- `const refvector< TN > & copy (const refvector< TN > &r, long i) const`  
*Do a copy of depth j of i into a referenced counter.*
- `const refvector< TN > & operator= (const refvector< TN > &r) const`  
*Assignment without deep copy. Old information is discarded if no references remain.*

## Private Attributes

- `unsigned long * _refcount`  
*Reference count.*
- `std::vector< TN > * _`  
*Data pointer.*

`template<class TN> class linear_algebra::refvector< TN >`

### 4.40.1 Constructor & Destructor Documentation

4.40.1.1 `template<class TN> linear_algebra::refvector< TN >::refvector ()`

Class refvector provides reference counted vectors. Default constructor;

4.40.1.2 `template<class TN> linear_algebra::refvector< TN >::refvector (const refvector< TN > & rhs)`

4.40.1.3 `template<class TN> linear_algebra::refvector< TN >::refvector (refvector< TN > & rhs)`

4.40.1.4 `template<class TN> linear_algebra::refvector< TN >::refvector (long i)`

4.40.1.5 `template<class TN> linear_algebra::refvector< TN >::refvector (const std::vector< TN > & i)`

4.40.1.6 `template<class TN> linear_algebra::refvector< TN >::refvector (long j, const TN * rhs)`

4.40.1.7 `template<class TN> linear_algebra::refvector< TN >::refvector (std::istream & IN)`

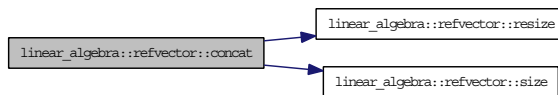
4.40.1.8 `template<class TN> linear_algebra::refvector< TN >::~~refvector ()`

### 4.40.2 Member Function Documentation

4.40.2.1 `template<class TN> void linear_algebra::refvector< TN >::clear ()`

4.40.2.2 `template<class TN> refvector< TN > & linear_algebra::refvector< TN >::concat (const refvector< TN > & a)`

Here is the call graph for this function:



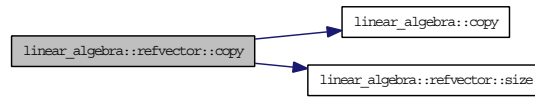
4.40.2.3 `template<class TN> long linear_algebra::refvector< TN >::contains (const TN & a) const`

Here is the call graph for this function:



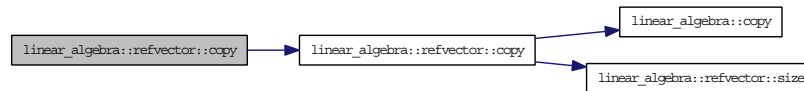
4.40.2.4 `template<class TN> const refvector< TN > & linear_algebra::refvector< TN >::copy (const refvector< TN > & r, long i) const[private]`

Here is the call graph for this function:



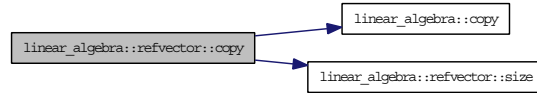
4.40.2.5 `template<class TN> const refvector< TN > & linear_algebra::refvector< TN >::copy (const refvector< TN > & r) const[private]`

Here is the call graph for this function:



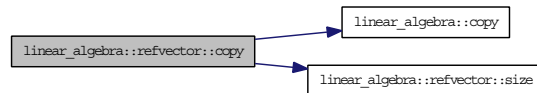
4.40.2.6 `template<class TN> refvector< TN > & linear_algebra::refvector< TN >::copy (const refvector< TN > & i)`

Here is the call graph for this function:



4.40.2.7 `template<class TN> refvector< TN > & linear_algebra::refvector< TN >::copy  
(const refvector< TN > & i, const long j)`

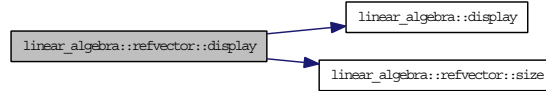
Here is the call graph for this function:



4.40.2.8 `template<class TN> long linear_algebra::refvector< TN >::dim () const`

4.40.2.9 `template<class TN> void linear_algebra::refvector< TN >::display (ostream &  
outs = cout) const`

Here is the call graph for this function:

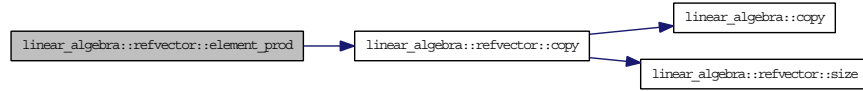


4.40.2.10 `template<class TN> refvector< TN > linear_algebra::refvector< TN  
>::element_prod (const refvector< TN > & a) const`

This is an elementwise multiplication which results in a new vector.  $c_i = c_i \cdot a_i$ . See `refvector::operator*=( )`.

Here is the call graph for this function:





4.40.2.11 `template<class TN> void linear_algebra::refvector< TN >::erase (typename std::vector< TN >::iterator a)`

4.40.2.12 `template<class TN> void linear_algebra::refvector< TN >::insert (long n, const TN & a)`

4.40.2.13 `template<class TN> TN linear_algebra::refvector< TN >::max () const`

Here is the call graph for this function:



4.40.2.14 `template<class TN> template<class TN2> refvector< TN > & linear_algebra::refvector< TN >::multiply (const TN2 x, refvector< TN > & a) const`

This may pose problems in some settings when `TN*TN2` is not defined.

4.40.2.15 `template<class TN> template<class TN3, class TN2> TN3& linear_algebra::refvector< TN >::multiply (const refvector< TN2 > & b, TN3 & r) const`

4.40.2.16 `template<class TN> void linear_algebra::refvector< TN >::normalise ()`

4.40.2.17 `template<class TN> template<class TN2> refvector< TN > linear_algebra::refvector< TN >::operator * (const TN2 x) const`

Here is the call graph for this function:



4.40.2.18 `template<class TN> template<class TN3, class TN2> TN3 linear_algebra::refvector< TN >::operator * (const refvector< TN2 > & b) const`

4.40.2.19 `template<class TN> TN linear_algebra::refvector< TN >::operator * (const refvector< TN > & b) const`

Here is the call graph for this function:



4.40.2.20 `template<class TN> template<class TN2> refvector< TN >  
linear_algebra::refvector< TN >::operator *= (const TN2 & a)`

4.40.2.21 `template<class TN> refvector< TN > & linear_algebra::refvector< TN  
>::operator *= (const TN x)`

4.40.2.22 `template<class TN> refvector< TN > & linear_algebra::refvector< TN  
>::operator *= (const refvector< TN > & a)`

This is an elementwise multiplication which results in a new vector.  $c_i = c_i \cdot a_i$ . See also `refvector::element_prod`

Here is the call graph for this function:



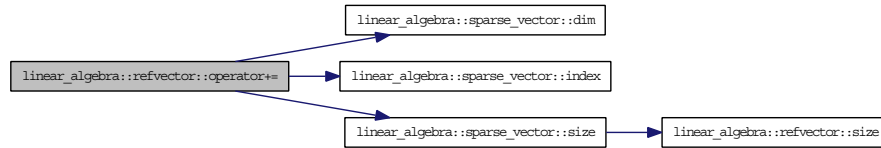
4.40.2.23 `template<class TN> refvector< TN > linear_algebra::refvector< TN  
>::operator+ (const refvector< TN > & B) const`

Here is the call graph for this function:



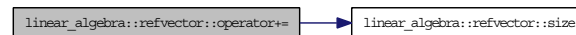
4.40.2.24 `template<class TN> refvector< TN > & linear_algebra::refvector< TN  
>::operator+= (const sparse_vector< TN > & a)`

Here is the call graph for this function:



4.40.2.25 `template<class TN> refvector< TN > & linear_algebra::refvector< TN >::operator+= (const refvector< TN > & B)`

Here is the call graph for this function:



4.40.2.26 `template<class TN> refvector< TN > linear_algebra::refvector< TN >::operator- (const refvector< TN > & ) const`

Here is the call graph for this function:



4.40.2.27 `template<class TN> refvector< TN > & linear_algebra::refvector< TN >::operator-= (const refvector< TN > & B)`

Here is the call graph for this function:



4.40.2.28 `template<class TN> const vector< TN > * linear_algebra::refvector< TN >::operator → () const`

4.40.2.29 `template<class TN> vector< TN > * linear_algebra::refvector< TN >::operator → ()`

4.40.2.30 `template<class TN> template<class TN2> refvector< TN > linear_algebra::refvector< TN >::operator/ (const TN2 x) const`

This may pose problems in some settings when `TN*=TN2` is not defined.

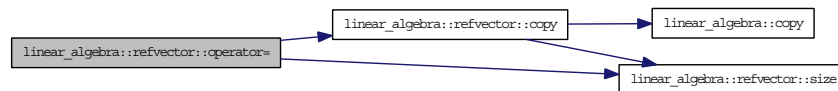
4.40.2.31 `template<class TN> const refvector< TN > & linear_algebra::refvector< TN >::operator= (const refvector< TN > & r) const[private]`

4.40.2.32 `template<class TN> refvector<TN>& linear_algebra::refvector< TN >::operator= (const std::vector< TN > & rhs)`

4.40.2.33 `template<class TN> refvector< TN > & linear_algebra::refvector< TN >::operator= (refvector< TN > & rhs)`

4.40.2.34 `template<class TN> refvector< TN > & linear_algebra::refvector< TN >::operator= (const refvector< TN > & rhs)`

Here is the call graph for this function:



4.40.2.35 `template<class TN> bool linear_algebra::refvector< TN >::operator==(const refvector< TN > & a) const`

Here is the call graph for this function:



4.40.2.36 `template<class TN> void linear_algebra::refvector< TN >::operator>>(std::ostream & o) const`

4.40.2.37 `template<class TN> const TN & linear_algebra::refvector< TN >::operator[](long i) const`

4.40.2.38 `template<class TN> TN & linear_algebra::refvector< TN >::operator[] (long i)`

4.40.2.39 `template<class TN> refvector< TN > & linear_algebra::refvector< TN >::push_back (const TN & a)`

4.40.2.40 `template<class TN> void linear_algebra::refvector< TN >::resize (long n)`

4.40.2.41 `template<class TN> void linear_algebra::refvector< TN >::resize (long n, const TN & a)`

4.40.2.42 `template<class TN> template<class T> bool linear_algebra::refvector< TN >::same (const refvector< T > & a) const`

4.40.2.43 `template<class TN> refvector< TN > & linear_algebra::refvector< TN >::set (long i, TN & rhs)`

Here is the call graph for this function:



4.40.2.44 `template<class TN> long linear_algebra::refvector< TN >::size () const`

4.40.2.45 `template<class TN> refvector< TN > & linear_algebra::refvector< TN >::subtract (const refvector< TN > & B, refvector< TN > & r) const`

Here is the call graph for this function:



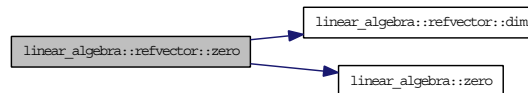
4.40.2.46 `template<class TN> const vector< TN > & linear_algebra::refvector< TN >::vec  
( ) const`

4.40.2.47 `template<class TN> vector< TN > & linear_algebra::refvector< TN >::vec (`

4.40.2.48 `template<class TN> void linear_algebra::refvector< TN >::write_to (std::ostream  
& OUTFILE)`

4.40.2.49 `template<class TN> refvector< TN > & linear_algebra::refvector< TN >::zero (`

Here is the call graph for this function:



### 4.40.3 Member Data Documentation

4.40.3.1 `template<class TN> char linear_algebra::refvector< TN >::_delim`

4.40.3.2 `template<class TN> std::vector<TN>* linear_algebra::refvector< TN  
>::_P[private]`

4.40.3.3 `template<class TN> unsigned long* linear_algebra::refvector< TN  
>::_refcount[private]`

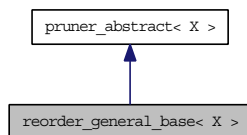
The documentation for this class was generated from the following files:

- `include/BCR_CPP_LA/refcount.decl`
- `include/BCR_CPP_LA/refcount.h`

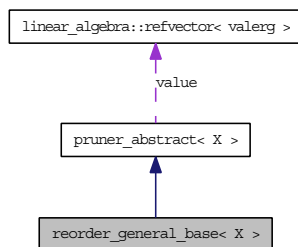
### 4.41 `reorder_general_base< X >` Class Template Reference

`#include <reordergeneralbase.hh>`

Inheritance diagram for `reorder_general_base< X >`:



Collaboration diagram for `reorder_general_base< X >`:



## Public Member Functions

- `reorder_general_base (reorder_general_base< X > &L)`  
*Copy Constructor.*
- `reorder_general_base (const reorder_general_base< X > &L)`  
*Copy Constructor.*
- `reorder_general_base (const refvector< long > &ibases)`  
*Constructor with a vector of bases.*
- `void adjust_lagrange (double &lambda, ulong &conf1, ulong &conf2, long &config, const refvector< ulong > &visited_run) const`  
*Compute the increase of lambda and assess current best value.*
- `ulong prune (double &lambda, ulong &conf1, ulong &conf2, long &config, const refvector< ulong > &visited_run) const`  
*Adjust the Lagrange multiplier and prune the library.*
- `ulong deprune (ulong N) const`  
*Reverse-map N to the global index.*
- `ulong reprune (ulong N) const`  
*Given an "absolute" reference number return the current reference number.*

## Private Attributes

- `refvector< refvector< long > > base_order`  
*Records the order of digits in each base.*
- `refvector< refvector< valerg > > base_averages`  
*Records the average values associated with each digit and base.*
- `ulong oldvisitedsize`  
*Records the size of visited prior to the latest iteration.*

#### 4.41.1 Detailed Description

**`template<class X> class reorder_general_base< X >`**

This class takes an array of bases and orders the bases so as to produce a smooth optimization process. Each digit per base is weighted by the average computed values for compounds with the appropriate digit. There is no actual pruning done!

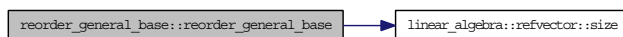
#### 4.41.2 Constructor & Destructor Documentation

4.41.2.1 `template<class X> reorder_general_base< X >::reorder_general_base (reorder_general_base< X > & L)[inline]`

4.41.2.2 `template<class X> reorder_general_base< X >::reorder_general_base (const reorder_general_base< X > & L)[inline]`

4.41.2.3 `template<class X> reorder_general_base< X >::reorder_general_base (const refvector< long > & ibases)[inline]`

Here is the call graph for this function:

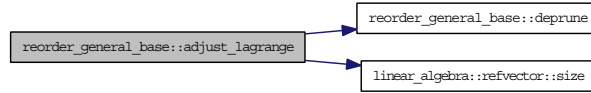


#### 4.41.3 Member Function Documentation

4.41.3.1 `template<class X> void reorder_general_base< X >::adjust_lagrange (double & lambda, ulong & conf1, ulong & conf2, long & config, const refvector< ulong > & visited_run) const[inline]`

Here is the call graph for this function:





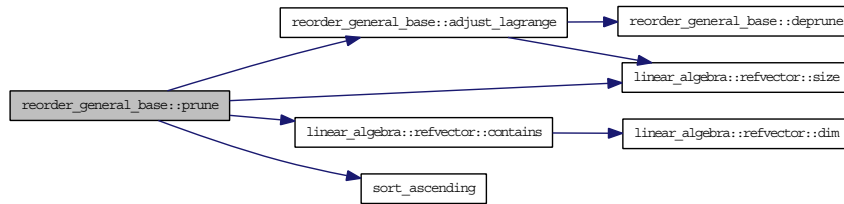
4.41.3.2 `template<class X> ulong reorder_general_base< X >::deprune (ulong N)`  
`const[inline, virtual]`

Implements `pruner_abstract< X >`.

4.41.3.3 `template<class X> ulong reorder_general_base< X >::prune (double & lambda,  
 ulon & conf1, ulong & conf2, long & config, const refvector< ulong > & visited_run)`  
`const[inline, virtual]`

Implements `pruner_abstract< X >`.

Here is the call graph for this function:



4.41.3.4 `template<class X> ulong reorder_general_base< X >::reprune (ulong N)`  
`const[inline, virtual]`

Implements `pruner_abstract< X >`.

## 4.41.4 Member Data Documentation

4.41.4.1 `template<class X> refvector< refvector<valerg> > reorder_general_base< X  
 >::base_averages[mutable, private]`

4.41.4.2 `template<class X> refvector< refvector < long > > reorder_general_base< X  
 >::base_order[mutable, private]`

4.41.4.3 `template<class X> ulong reorder_general_base< X >::oldvisitedsize[mutable,  
 private]`

The documentation for this class was generated from the following file:

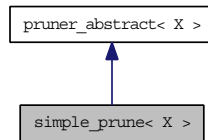
- reordergeneralbase.hh

#### 4.42 simple\_prune< X > Class Template Reference

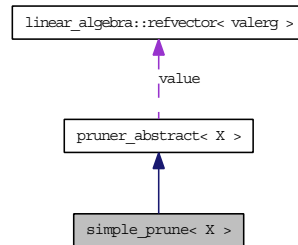
Class to prune a Library.

```
#include <simpleprune.h>
```

Inheritance diagram for simple\_prune< X >:



Collaboration diagram for simple\_prune< X >:



#### Public Member Functions

- `simple_prune ()`
- `simple_prune (const simple_prune< X > &a)`
- `ulong prune (double &lambda, ulong &conf1, ulong &conf2, long &config, const refvector< ulong > &a) const`  
*Prune the Library and adjust lambda. Pruned entries are in pruned\_visited.*
- `ulong deprune (ulong N) const`  
*Revert an index from the pruned Library to the original Library.*
- `ulong reprune (ulong N) const`  
*Convert an "absolute" index to a relative index.*

```
template<class X> class simple_prune< X >
```

### 4.42.1 Constructor & Destructor Documentation

4.42.1.1 `template<class X> simple_prune< X >::simple_prune ()`[inline]

4.42.1.2 `template<class X> simple_prune< X >::simple_prune (const simple_prune< X > & a)`[inline]

### 4.42.2 Member Function Documentation

4.42.2.1 `template<class X> ulong simple_prune< X >::deprune (ulong N) const`[virtual]

Implements `pruner_abstract< X >`.

4.42.2.2 `template<class X> ulong simple_prune< X >::prune (double & lambda, ulong & conf1, ulong & conf2, long & config, const refvector< ulong > & visited_run) const`[virtual]

Inferior values have larger penalties than the current best. Lambda is adjusted such that  $Prop_{conf1} - \lambda' Pen_{conf1} < Prop_j - \lambda' Pen_j$  for some  $j$  and  $\lambda' > \lambda$

Implements `pruner_abstract< X >`.

4.42.2.3 `template<class X> ulong simple_prune< X >::reprune (ulong N) const`[virtual]

Implements `pruner_abstract< X >`.

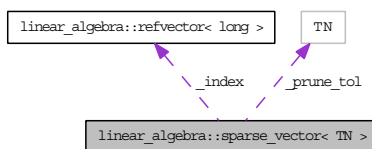
The documentation for this class was generated from the following files:

- `simpleprune.h`
- `simpleprune.cc`

## 4.43 `linear_algebra::sparse_vector< TN >` Class Template Reference

A class for sparse vectors.

Collaboration diagram for `linear_algebra::sparse_vector< TN >`:



## Public Member Functions

- `sparse_vector ()`  
*Default constructor. Dimension 0.*
- `sparse_vector (const long n)`  
*Constructs a sparse vector of dimension n without initialisation.*
- `sparse_vector (const long n, const std::vector< long > &a, const std::vector< TN > &b)`  
*Constructs a sparse vector of dimension n with initialisation.*
- `sparse_vector (const long n, const refvector< long > &a, const std::vector< TN > &b)`  
*Constructs a sparse vector of dimension n with initialisation.*
- `sparse_vector (const long n, const refvector< long > &a, const refvector< TN > &b)`  
*Constructs a sparse vector of dimension n with initialisation.*
- `sparse_vector (const long n, const std::vector< long > &a, const refvector< TN > &b)`  
*Constructs a sparse vector of dimension n with initialisation.*
- `sparse_vector (const sparse_vector &a)`  
*Copy constructor.—*
- `sparse_vector< TN > & operator *= (const sparse_vector< TN > &a)`  
*Multiply by a sparse vector.*
- `TN operator * (const sparse_vector &a) const`  
*Scalar product of sparse vectors.*
- `bool is_nonzero (const long x, long &i) const`  
*is (x) non zero*
- `TN operator[] (const long x)`  
*Direct access operator.*
- `TN operator[] (const long x) const`  
*Direct access operator.*
- `const TN & operator() (const long x) const`  
*Access by element operator.*
- `TN operator() (const long x)`  
*Access by element operator.*

- `sparse_vector< TN > & add (const long x, TN z)`  
*Add an element for consistency.*
- `sparse_vector< TN > & multiply (const long x, TN z)`  
*Multiply an element for consistency.*
- `sparse_vector< TN > & operator+= (const sparse_vector< TN > &B)`  
*Add a sparse vector.*
- `sparse_vector< TN > & operator-= (const sparse_vector< TN > &B)`  
*Subtract a sparse vector.*
- `sparse_vector< TN > operator- () const`  
*Subtract a sparse vector.*
- `sparse_vector< TN > & operator *= (const TN x)`  
*Multiply by an element.*
- `sparse_vector< TN > operator * (const TN x) const`  
*Multiply by an element.*
- `sparse_vector< TN > & operator/= (const TN x)`  
*Divide by an element.*
- `sparse_vector< TN > operator/ (const TN x) const`  
*Divide by an element.*
- `sparse_vector< TN > & operator= (const sparse_vector< TN > &b)`  
*Copy const sparse\_vector.*
- `sparse_vector< TN > & operator= (sparse_vector< TN > &b)`  
*Copy const sparse\_vector.*
- `sparse_vector< TN > & operator= (const std::vector< TN > b)`  
*Copy a const vector into a sparse vector.*
- `sparse_vector< TN > copy (const sparse_vector< TN > &b, long i)`  
*Copy sparse vector with copy of depth i.*
- `sparse_vector< TN > copy (const sparse_vector< TN > &b)`  
*Copy a sparse vector with depth=1.*
- `void set (long i, TN value)`
- `void prune (const TN x)`  
*Prune a sparse vector of negligible values.*

- `void prune ()`
- `bool display () const`  
*Simple display of sparse vector.*
- `TN max () const`  
*Returns the maximum element.*
- `void normalise ()`  
*Normalise the sparse vector in accordance with the  $\|\cdot\|_2$  2-norm.*
- `long size () const`  
*Returns the number of nonzero elements.*
- `long index (long i) const`  
*Returns the index of the  $i$ th nonzero element.*
- `long dim () const`  
*Returns the dimension of the vector.*
- `void clear ()`  
*Releases the associated memory, i.e., sets the vector to 0.*
- `void zero ()`  
*Retains the dimension but releases the associated memory, i.e., sets the vector to 0.*

## Public Attributes

- `TN _prune_tol`  
*Pruning tolerance.*

## Private Attributes

- `long _size`  
*Dimension.*
- `refvector< TN > _vals`  
*Vector of values.*
- `linear_algebra::refvector< long > _index`  
*Vector of indices.*

## Friends

- `class mat_sparse< TN >`
- `class mat_sym_sparse< TN >`
- `class mat_asym_sparse< TN >`
- `class mat_sym_full< TN >`
- `class mat_full< TN >`

`template<class TN> class linear_algebra::sparse_vector< TN >`

### 4.43.1 Constructor & Destructor Documentation

4.43.1.1 `template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector ()`

4.43.1.2 `template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector (const long n)`

4.43.1.3 `template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector (const long n, const std::vector< long > & a, const std::vector< TN > & b)`

4.43.1.4 `template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector (const long n, const refvector< long > & a, const std::vector< TN > & b)`

4.43.1.5 `template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector (const long n, const refvector< long > & a, const refvector< TN > & b)`

#### Parameters:

*a* Reference counted vector indices.  
Reference counted vector values.

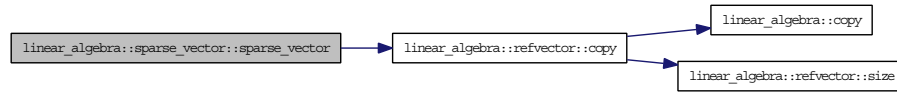
Here is the call graph for this function:



4.43.1.6 `template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector (const long n, const std::vector< long > & a, const refvector< TN > & b)`

4.43.1.7 `template<class TN> linear_algebra::sparse_vector< TN >::sparse_vector (const sparse_vector< TN > & a)`

Here is the call graph for this function:

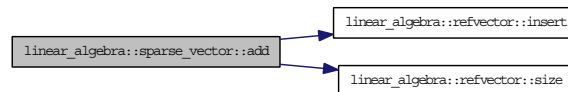


## 4.43.2 Member Function Documentation

4.43.2.1 `template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::add (const long x, TN z)`

One of the pitfalls of using this for assignments is that if `_size` is set incorrectly the throw argument will result in an SIGABORT if not caught.

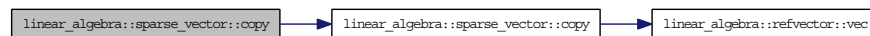
Here is the call graph for this function:



4.43.2.2 `template<class TN> void linear_algebra::sparse_vector< TN >::clear ()`

4.43.2.3 `template<class TN> sparse_vector< TN > linear_algebra::sparse_vector< TN >::copy (const sparse_vector< TN > & b)`

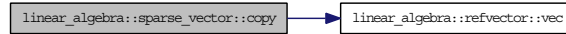
Here is the call graph for this function:



4.43.2.4 `template<class TN> sparse_vector< TN > linear_algebra::sparse_vector< TN >::copy (const sparse_vector< TN > & b, long i)`

Here is the call graph for this function:





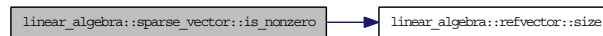
4.43.2.5 `template<class TN> long linear_algebra::sparse_vector< TN >::dim () const`

4.43.2.6 `template<class TN> bool linear_algebra::sparse_vector< TN >::display () const`

4.43.2.7 `template<class TN> long linear_algebra::sparse_vector< TN >::index (long i) const`

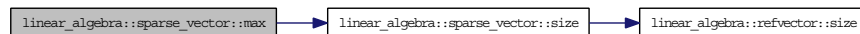
4.43.2.8 `template<class TN> bool linear_algebra::sparse_vector< TN >::is_nonzero (const long x, long & i) const`

Here is the call graph for this function:



4.43.2.9 `template<class TN> TN linear_algebra::sparse_vector< TN >::max () const`

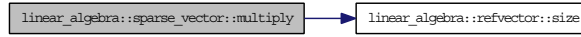
Here is the call graph for this function:



4.43.2.10 `template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::multiply (const long x, TN z)`

One of the pitfalls of using this for assignments is that if `_size` is set incorrectly the throw argument will result in an SIGABORT if not caught.

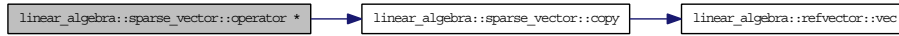
Here is the call graph for this function:



4.43.2.11 `template<class TN> void linear_algebra::sparse_vector< TN >::normalise ()`

4.43.2.12 `template<class TN> sparse_vector< TN > linear_algebra::sparse_vector< TN >::operator * (const TN x) const`

Here is the call graph for this function:



4.43.2.13 `template<class TN> TN linear_algebra::sparse_vector< TN >::operator * (const sparse_vector< TN > & a) const`

Here is the call graph for this function:

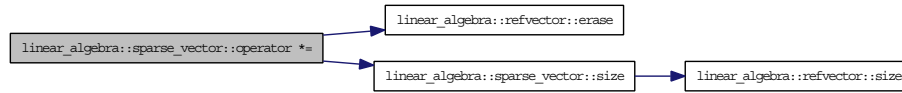


4.43.2.14 `template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::operator *= (const TN x)`

4.43.2.15 `template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::operator *= (const sparse_vector< TN > & a)`

This is an elementwise multiplication which results in a new vector.  $c_i = c_i \cdot a_i$ .

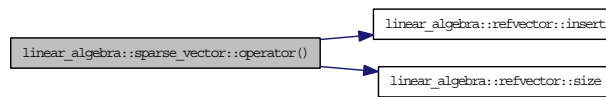
Here is the call graph for this function:



4.43.2.16 `template<class TN> TN & linear_algebra::sparse_vector< TN >::operator()  
(const long x)`

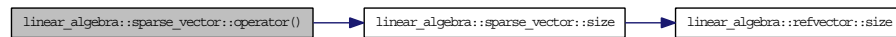
One of the pitfalls of using this for assignments is that if `_size` is set incorrectly the throw argument will result in an SIGABORT if not caught.

Here is the call graph for this function:



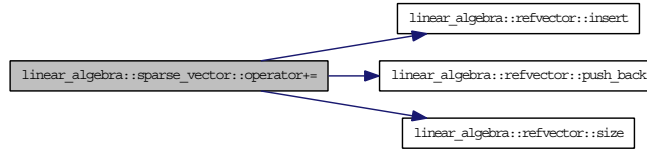
4.43.2.17 `template<class TN> const TN & linear_algebra::sparse_vector< TN  
>::operator() (const long x) const`

Here is the call graph for this function:



4.43.2.18 `template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN  
>::operator+= (const sparse_vector< TN > & B)`

Here is the call graph for this function:

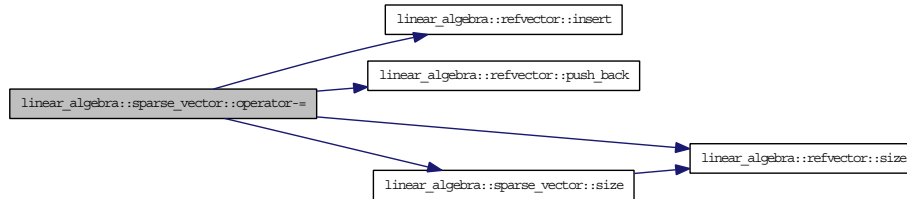


4.43.2.19 `template<class TN> sparse_vector< TN > linear_algebra::sparse_vector< TN >::operator- ( ) const`

**Test**

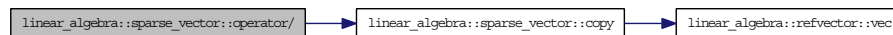
4.43.2.20 `template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::operator-= (const sparse_vector< TN > & B)`

Here is the call graph for this function:



4.43.2.21 `template<class TN> sparse_vector< TN > linear_algebra::sparse_vector< TN >::operator/ (const TN x) const`

Here is the call graph for this function:



4.443.2.22 `template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::operator/= (const TN x)`

4.43.2.23 `template<class TN> sparse_vector<TN>& linear_algebra::sparse_vector< TN >::operator= (const std::vector< TN > b)`

4.43.2.24 `template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::operator= (sparse_vector< TN > & b)`

4.43.2.25 `template<class TN> sparse_vector< TN > & linear_algebra::sparse_vector< TN >::operator= (const sparse_vector< TN > & b)`

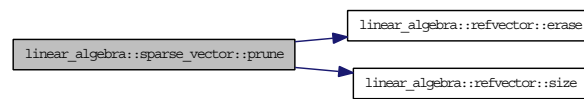
4.43.2.26 `template<class TN> TN linear_algebra::sparse_vector< TN >::operator[] (const long x) const`

4.43.2.27 `template<class TN> TN & linear_algebra::sparse_vector< TN >::operator[] (const long x)`

4.43.2.28 `template<class TN> void linear_algebra::sparse_vector< TN >::prune ()`

4.43.2.29 `template<class TN> void linear_algebra::sparse_vector< TN >::prune (const TN x)`

Here is the call graph for this function:



4.43.2.30 `template<class TN> void linear_algebra::sparse_vector< TN >::set (long i, TN value)`

4.43.2.31 `template<class TN> long linear_algebra::sparse_vector< TN >::size () const`

Here is the call graph for this function:



4.43.2.32 `template<class TN> void linear_algebra::sparse_vector< TN >::zero ()`

### 4.43.3 Friends and Related Function Documentation

4.43.3.1 `template<class TN> friend class mat_asym_sparse< TN >[friend]`

4.43.3.2 `template<class TN> friend class mat_full< TN >[friend]`

4.43.3.3 `template<class TN> friend class mat_sparse< TN >[friend]`

4.43.3.4 `template<class TN> friend class mat_sym_full< TN >[friend]`

4.43.3.5 `template<class TN> friend class mat_sym_sparse< TN >[friend]`

#### 4.43.4 Member Data Documentation

4.43.4.1 `template<class TN> linear_algebra::refvector< long >`

`linear_algebra::sparse_vector< TN >::_index[private]`

4.43.4.2 `template<class TN> TN linear_algebra::sparse_vector< TN >::_prune_tol`

4.43.4.3 `template<class TN> long linear_algebra::sparse_vector< TN >::_size[private]`

4.43.4.4 `template<class TN> refvector<TN> linear_algebra::sparse_vector< TN >::_vals[private]`

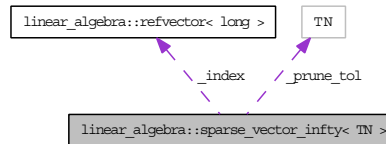
The documentation for this class was generated from the following files:

- `include/BCR_CPP_LA/sparse_vector.decl`
- `include/BCR_CPP_LA/sparse_vector.h`

#### 4.44 `linear_algebra::sparse_vector_infty< TN >` Class Template Reference

A class for sparse vectors.

Collaboration diagram for `linear_algebra::sparse_vector_infty< TN >`:



#### Public Member Functions

- `sparse_vector_infty ()`  
*Default constructor. Dimension 0.*
- `sparse_vector_infty (const std::vector< long > &a, const std::vector< TN > &b)`  
*Constructs an infinite sparse vector with initialisation.*
- `sparse_vector_infty (const refvector< long > &a, const std::vector< TN > &b)`  
*Constructs an infinite sparse vector with initialisation.*
- `sparse_vector_infty (const refvector< long > &a, const refvector< TN > &b)`  
*Constructs a sparse vector of infinite dimension with initialisation.*

- `sparse_vector_infty (const std::vector< long > &a, const refvector< TN > &b)`  
*Constructs an infinite sparse vector with initialisation.*
- `sparse_vector_infty (const sparse_vector_infty &a)`  
*Copy constructor.*
- `sparse_vector_infty< TN > & operator *= (const sparse_vector_infty< TN > &a)`  
*Multiply by a sparse vector.*
- `TN operator * (const sparse_vector_infty &a) const`  
*Scalar product of sparse vectors.*
- `bool is_nonzero (const long x, long &i) const`  
*is (x) non zero*
- `TN & operator[] (const long x)`  
*Direct access operator.*
- `TN operator[] (const long x) const`  
*Direct access operator.*
- `TN operator() (const long x) const`  
*Access by element operator.*
- `TN & operator() (const long x)`  
*Access by element operator.*
- `sparse_vector_infty< TN > & add (const long x, TN z)`  
*Add an element for consistency.*
- `sparse_vector_infty< TN > & multiply (const long x, TN z)`  
*Multiply an element for consistency.*
- `sparse_vector_infty< TN > & operator+= (const sparse_vector_infty< TN > &B)`  
*Add an infinite sparse vector.*
- `sparse_vector_infty< TN > & operator-= (const sparse_vector_infty< TN > &B)`  
*Subtract a sparse vector.*
- `sparse_vector_infty< TN > operator- () const`  
*Subtract a sparse vector.*
- `sparse_vector_infty< TN > & operator *= (const TN x)`  
*Multiply by an element.*
- `sparse_vector_infty< TN > operator * (const TN x) const`  
*Multiply by an element.*

- `sparse_vector_infty< TN > & operator/= (const TN x)`  
*Divide by an element.*
- `sparse_vector_infty< TN > operator/ (const TN x) const`  
*Divide by an element.*
- `sparse_vector_infty< TN > & operator= (const sparse_vector_infty< TN > &b)`  
*Copy const sparse\_vector\_infty.*
- `sparse_vector_infty< TN > & operator= (sparse_vector_infty< TN > &b)`  
*Copy const sparse\_vector\_infty.*
- `sparse_vector_infty< TN > & operator= (const std::vector< TN > b)`  
*Copy a const vector into a sparse vector.*
- `sparse_vector_infty< TN > copy (const sparse_vector_infty< TN > &b, long i)`  
*Copy sparse vector with copy of depth i.*
- `sparse_vector_infty< TN > copy (const sparse_vector_infty< TN > &b)`  
*Copy a sparse vector with depth=1.*
- `void set (long i, const TN &value)`
- `void set (long i, TN &value)`
- `void prune (const TN x)`  
*Prune a sparse vector of negligible values.*
- `void prune ()`
- `bool display () const`  
*Simple display of sparse vector.*
- `TN max () const`  
*Returns the maximum element.*
- `void normalise ()`  
*Normalise the sparse vector in accordance with the  $\| \cdot \|_2$  2-norm.*
- `long size () const`  
*Returns the number of nonzero elements.*
- `long index (long i) const`  
*Returns the index of the ith nonzero element.*
- `void clear ()`  
*Releases the associated memory, i.e., sets the vector to 0.*



- `void zero ()`  
*Retains the dimension but releases the associated memory, i.e., sets the vector to 0.*

## Public Attributes

- `TN _prune_tol`  
*Pruning tolerance.*

## Private Attributes

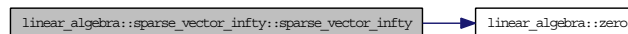
- `long _size`  
*Non-zero entries.*
- `refvector< TN > _vals`  
*Vector of values.*
- `refvector< long > _index`  
*Vector of indices.*

**template<class TN> class linear\_algebra::sparse\_vector\_infty< TN >**

### 4.44.1 Constructor & Destructor Documentation

4.44.1.1 `template<class TN> linear_algebra::sparse_vector_infty< TN >::sparse_vector_infty ()`

Here is the call graph for this function:



4.44.1.2 `template<class TN> linear_algebra::sparse_vector_infty< TN >::sparse_vector_infty (const std::vector< long > & a, const std::vector< TN > & b)`

4.44.1.3 `template<class TN> linear_algebra::sparse_vector_infty< TN >::sparse_vector_infty (const refvector< long > & a, const std::vector< TN > & b)`

4.44.1.4 `template<class TN> linear_algebra::sparse_vector_infty< TN >::sparse_vector_infty (const refvector< long > & a, const refvector< TN > & b)`

## Parameters:

- a* Reference counted vector indices.
- b* Reference counted vector values.

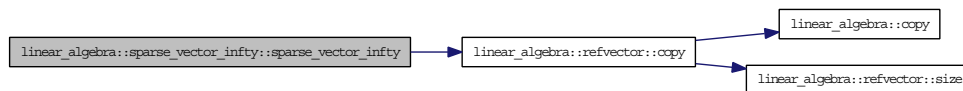
Here is the call graph for this function:



4.44.1.5 `template<class TN> linear_algebra::sparse_vector_infty< TN >::sparse_vector_infty (const std::vector< long > & a, const refvector< TN > & b)`

4.44.1.6 `template<class TN> linear_algebra::sparse_vector_infty< TN >::sparse_vector_infty (const sparse_vector_infty< TN > & a)`

Here is the call graph for this function:

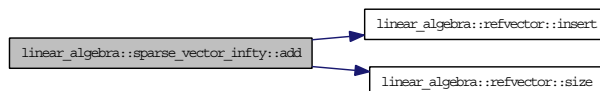


## 4.44.2 Member Function Documentation

4.44.2.1 `template<class TN> sparse_vector_infty< TN > & linear_algebra::sparse_vector_infty< TN >::add (const long x, TN z)`

One of the pitfalls of using this for assignments is that if `_size` is set incorrectly the throw argument will result in an SIGABORT if not caught.

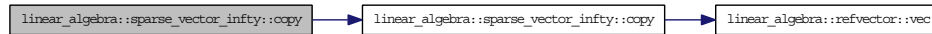
Here is the call graph for this function:



4.44.2.2 `template<class TN> void linear_algebra::sparse_vector_infty< TN >::clear ()`

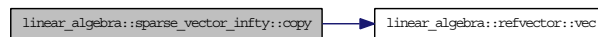
4.44.2.3 `template<class TN> sparse_vector_infty< TN >`  
`linear_algebra::sparse_vector_infty< TN >::copy (const sparse_vector_infty< TN > & b)`

Here is the call graph for this function:



4.44.2.4 `template<class TN> sparse_vector_infty< TN >`  
`linear_algebra::sparse_vector_infty< TN >::copy (const sparse_vector_infty< TN > & b,`  
`long i)`

Here is the call graph for this function:

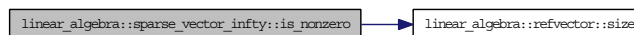


4.44.2.5 `template<class TN> bool linear_algebra::sparse_vector_infty< TN >::display ()`  
`const`

4.44.2.6 `template<class TN> long linear_algebra::sparse_vector_infty< TN >::index (long`  
`i) const`

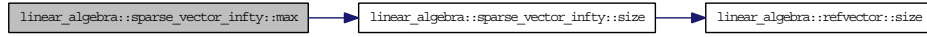
4.44.2.7 `template<class TN> bool linear_algebra::sparse_vector_infty< TN >::is_nonzero`  
`(const long x, long & i) const`

Here is the call graph for this function:



4.44.2.8 `template<class TN> TN linear_algebra::sparse_vector_infty< TN >::max () const`

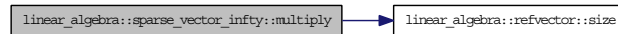
Here is the call graph for this function:



4.44.2.9 `template<class TN> sparse_vector_infty< TN > &  
linear_algebra::sparse_vector_infty< TN >::multiply (const long x, TN z)`

One of the pitfalls of using this for assignments is that if `_size` is set incorrectly the throw argument will result in an SIGABORT if not caught.

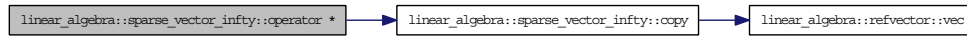
Here is the call graph for this function:



4.44.2.10 `template<class TN> void linear_algebra::sparse_vector_infty< TN >::normalise ()`

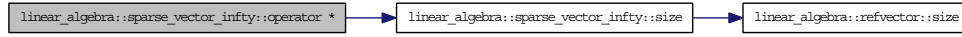
4.44.2.11 `template<class TN> sparse_vector_infty< TN >  
linear_algebra::sparse_vector_infty< TN >::operator * (const TN x) const`

Here is the call graph for this function:



4.44.2.12 `template<class TN> TN linear_algebra::sparse_vector_infty< TN >::operator *  
(const sparse_vector_infty< TN > & a) const`

Here is the call graph for this function:

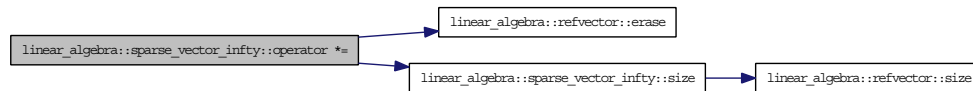


4.44.2.13 `template<class TN> sparse_vector_infty< TN > &`  
`linear_algebra::sparse_vector_infty< TN >::operator *= (const TN x)`

4.44.2.14 `template<class TN> sparse_vector_infty< TN > &`  
`linear_algebra::sparse_vector_infty< TN >::operator *= (const sparse_vector_infty< TN >`  
`& a)`

This is an elementwise multiplication which results in a new vector.  $c_i = c_i \cdot a_i$ .

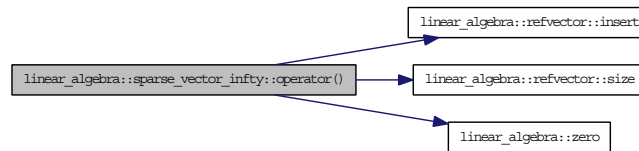
Here is the call graph for this function:



4.44.2.15 `template<class TN> TN & linear_algebra::sparse_vector_infty< TN >::operator()`  
`(const long x)`

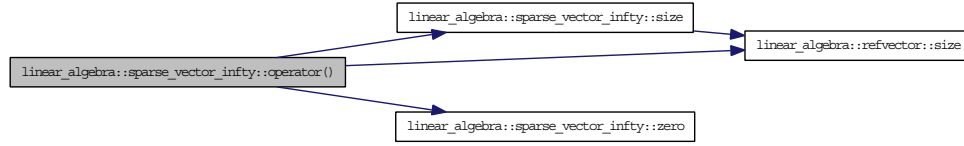
One of the pitfalls of using this for assignments is that if `_size` is set incorrectly the throw argument will result in an SIGABORT if not caught.

Here is the call graph for this function:



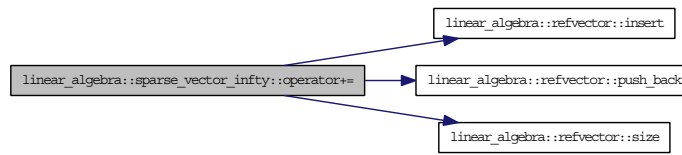
4.44.2.16 `template<class TN> TN linear_algebra::sparse_vector_infty< TN >::operator()`  
`(const long x) const`

Here is the call graph for this function:



4.44.2.17 `template<class TN> sparse_vector_infty< TN > &  
linear_algebra::sparse_vector_infty< TN >::operator+= (const sparse_vector_infty< TN >  
& B)`

Here is the call graph for this function:

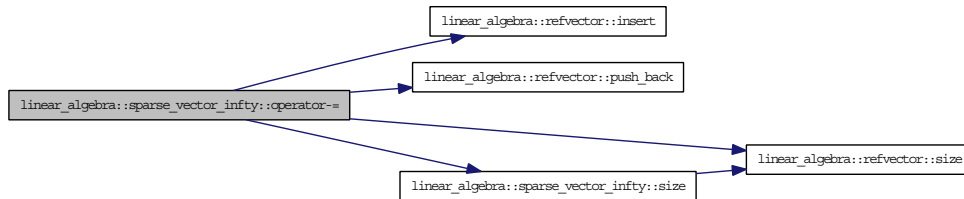


4.44.2.18 `template<class TN> sparse_vector_infty< TN >  
linear_algebra::sparse_vector_infty< TN >::operator- () const`

## Test

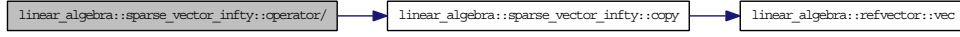
4.44.2.19 `template<class TN> sparse_vector_infty< TN > &  
linear_algebra::sparse_vector_infty< TN >::operator-= (const sparse_vector_infty< TN > &  
B)`

Here is the call graph for this function:



4.44.2.20 `template<class TN> sparse_vector_infty< TN >  
linear_algebra::sparse_vector_infty< TN >::operator/ (const TN x) const`

Here is the call graph for this function:



4.44.2.21 `template<class TN> sparse_vector_infty< TN > &  
linear_algebra::sparse_vector_infty< TN >::operator/= (const TN x)`

4.44.2.22 `template<class TN> sparse_vector_infty<TN>&  
linear_algebra::sparse_vector_infty< TN >::operator= (const std::vector< TN > b)`

4.44.2.23 `template<class TN> sparse_vector_infty< TN > &  
linear_algebra::sparse_vector_infty< TN >::operator= (sparse_vector_infty< TN > & b)`

4.44.2.24 `template<class TN> sparse_vector_infty< TN > &  
linear_algebra::sparse_vector_infty< TN >::operator= (const sparse_vector_infty< TN > &  
b)`

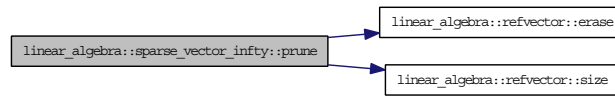
4.44.2.25 `template<class TN> TN linear_algebra::sparse_vector_infty< TN >::operator[]  
(const long x) const`

4.44.2.26 `template<class TN> TN & linear_algebra::sparse_vector_infty< TN >::operator[]  
(const long x)`

4.44.2.27 `template<class TN> void linear_algebra::sparse_vector_infty< TN >::prune ()`

4.44.2.28 `template<class TN> void linear_algebra::sparse_vector_infty< TN >::prune  
(const TN x)`

Here is the call graph for this function:



4.44.2.29 `template<class TN> void linear_algebra::sparse_vector_infty< TN >::set (long i,  
TN & value)`

4.44.2.30 `template<class TN> void linear_algebra::sparse_vector_infty< TN >::set (long i,  
const TN & value)`

4.44.2.31 `template<class TN> long linear_algebra::sparse_vector_infty< TN >::size () const`

Here is the call graph for this function:



4.44.2.32 `template<class TN> void linear_algebra::sparse_vector_infty< TN >::zero ()`

### 4.44.3 Member Data Documentation

4.44.3.1 `template<class TN> refvector<long> linear_algebra::sparse_vector_infty< TN >::index[private]`

4.44.3.2 `template<class TN> TN linear_algebra::sparse_vector_infty< TN >::_prune_tol`

4.44.3.3 `template<class TN> long linear_algebra::sparse_vector_infty< TN >::_sizetext[private]`

4.44.3.4 `template<class TN> refvector<TN> linear_algebra::sparse_vector_infty< TN >::_vals[private]`

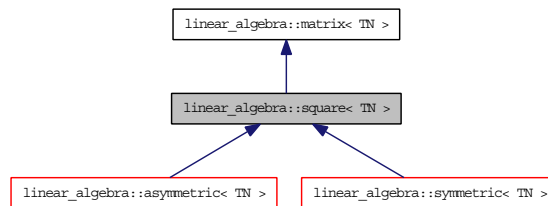
The documentation for this class was generated from the following files:

- `include/BCR_CPP_LA/sparse_vector_infty.decl`
- `include/BCR_CPP_LA/sparse_vector_infty.h`

## 4.45 linear\_algebra::square< TN > Class Template Reference

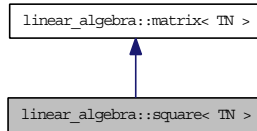
The square matrix class contains all n by n matrices.

Inheritance diagram for `linear_algebra::square< TN >`:



Collaboration diagram for `linear_algebra::square< TN >`:





## Public Member Functions

- TN trace () const

*The square matrix class contains all n by n matrices.*

**template<class TN> class linear\_algebra::square< TN >**

### 4.45.1 Member Function Documentation

4.45.1.1 template<class TN> TN linear\_algebra::square< TN >::trace () const

Reimplemented from linear\_algebra::matrix< TN >.

Reimplemented in linear\_algebra::mat\_sym\_sparse< TN >.

The documentation for this class was generated from the following files:

- include/BCR\_CPP\_LA/square.decl
- include/BCR\_CPP\_LA/square.h

## 4.46 boost::stateful\_thread\_group Class Reference

#include <stateful\_thread\_group.hh>

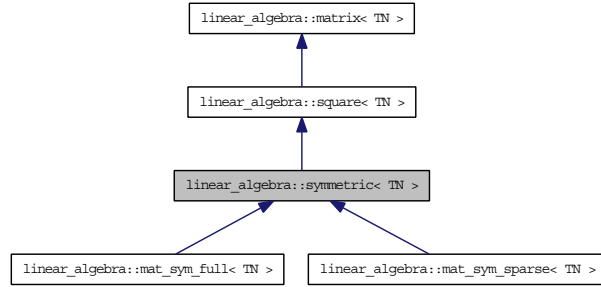
The documentation for this class was generated from the following file:

- include/stateful\_thread\_group.hh

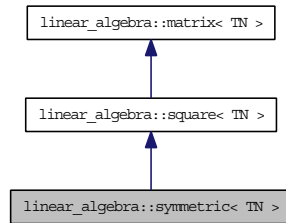
## 4.47 linear\_algebra::symmetric< TN > Class Template Reference

The symmetric class contains all symmetric matrices.

Inheritance diagram for linear\_algebra::symmetric< TN >:



Collaboration diagram for `linear_algebra::symmetric< TN >`:



- `matrix< TN > & transpose (matrix< TN > &r) const`  
*The symmetric class contains all symmetric matrices.*
- `matrix< TN > & transpose (matrix< TN > &r)`
- `symmetric< TN > & transpose (symmetric< TN > &r) const`
- `symmetric< TN > & transpose (symmetric< TN > &r)`
- `symmetric< TN > & transpose () const`
- `symmetric< TN > & transpose ()`
- `matrix< TN > & copy (const matrix< TN > &a)`
- `matrix< TN > & copy (const matrix< TN > &a, const long i)`
- `matrix< TN > & copy (matrix< TN > &a, const long i)`
- `virtual symmetric< TN > & copy (const symmetric< TN > &a)=0`
- `virtual symmetric< TN > & copy (const symmetric< TN > &a, const long i)=0`
- `virtual symmetric< TN > & copy (symmetric< TN > &a, const long i)=0`

- `bool add (long x, long y, const TN z)`  
*Adds a value.*
- `symmetric< TN > & prune (const TN x=0)`      *Deletes all entries that are smaller in magnitude than x.*

**template<class TN> class linear\_algebra::symmetric< TN >**

#### 4.47.1 Member Function Documentation

4.47.1.1 `template<class TN> bool linear_algebra::symmetric< TN >::add (long x, long y, const TN z)`

Reimplemented in `linear_algebra::mat_sym_full< TN >`, `linear_algebra::mat_sym_sparse< TN >`, and `linear_algebra::mat_sym_full< valerg >`.

4.47.1.2 `template<class TN> virtual symmetric<TN>& linear_algebra::symmetric< TN >::copy (symmetric< TN > & a, const long i)[pure virtual]`

Implemented in `linear_algebra::mat_sym_full< TN >`, `linear_algebra::mat_sym_sparse< TN >`, and `linear_algebra::mat_sym_full< valerg >`.

4.47.1.3 `template<class TN> virtual symmetric<TN>& linear_algebra::symmetric< TN >::copy (const symmetric< TN > & a, const long i)[pure virtual]`

Implemented in `linear_algebra::mat_sym_full< TN >`, `linear_algebra::mat_sym_sparse< TN >`, and `linear_algebra::mat_sym_full< valerg >`.

4.47.1.4 `template<class TN> virtual symmetric<TN>& linear_algebra::symmetric< TN >::copy (const symmetric< TN > & a)[pure virtual]`

Implemented in `linear_algebra::mat_sym_full< TN >`, `linear_algebra::mat_sym_sparse< TN >`, and `linear_algebra::mat_sym_full< valerg >`.

4.47.1.5 `template<class TN> matrix< TN > & linear_algebra::symmetric< TN >::copy (matrix< TN > & a, const long i)`

Here is the call graph for this function:



4.47.1.6 `template<class TN> matrix< TN > & linear_algebra::symmetric< TN >::copy  
(const matrix< TN > & a, const long i)`

Here is the call graph for this function:



4.47.1.7 `template<class TN> matrix< TN > & linear_algebra::symmetric< TN >::copy  
(const matrix< TN > & a)`

4.47.1.8 `template<class TN> symmetric<TN>& linear_algebra::symmetric< TN >::prune  
(const TN x = 0)`

Reimplemented in `linear_algebra::mat_sym_full< TN >`, `linear_algebra::mat_sym_sparse< TN >`, and `linear_algebra::mat_sym_full< valerg >`.

4.47.1.9 `template<class TN> symmetric< TN > & linear_algebra::symmetric< TN  
>::transpose ()`

4.47.1.10 `template<class TN> symmetric< TN > & linear_algebra::symmetric< TN  
>::transpose () const`

4.47.1.11 `template<class TN> symmetric< TN > & linear_algebra::symmetric< TN  
>::transpose (symmetric< TN > & r)`

4.47.1.12 `template<class TN> symmetric< TN > & linear_algebra::symmetric< TN  
>::transpose (symmetric< TN > & r) const`

4.47.1.13 `template<class TN> matrix< TN > & linear_algebra::symmetric< TN  
>::transpose (matrix< TN > & r)`

4.47.1.14 `template<class TN> matrix< TN > & linear_algebra::symmetric< TN  
>::transpose (matrix< TN > & r) const`

The documentation for this class was generated from the following files:

- `include/BCR_CPP_LA/symmetric.decl`
- `include/BCR_CPP_LA/symmetric.h`

## 4.48 valerg Struct Reference

Pair of property value and energy.

```
#include <typedefs.hh>
```

### Public Attributes

- double property
- double penalty
- double energy
- bool property\_computed
- bool energy\_computed

### 4.48.1 Member Data Documentation

4.48.1.1 double valerg::energy

4.48.1.2 bool valerg::energy\_computed

4.48.1.3 double valerg::penalty

4.48.1.4 double valerg::property

4.48.1.5 bool valerg::property\_computed

The documentation for this struct was generated from the following file:

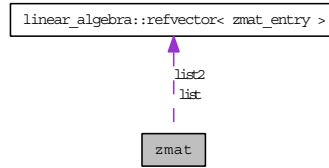
- typedefs.hh

## 4.49 zmat Class Reference

Class for Z-matrices. Just a wrapper class around a vector of entries.

```
#include <zmat.hh>
```

Collaboration diagram for zmat:



## Public Member Functions

- `zmat & operator= (const zmat &a)`  
*Assignment operator.*
- `bool operator== (const zmat &a) const`  
*Comparison operator.*
- `zmat & set_val (int i, int j, double val)`  
*Set a value.*
- `zmat & add_val (int i, int j, double val)`  
*Add to a value.*
- `zmat & add_entry (const zmat_entry &e)`  
*Add an entry to the Z-matrix.*
- `void output (ostream &out) const`  
*Output the z-matrix definition.*
- `zmat & add_increment (long i, long j, double a)`  
*Add another value to entry i in position j.*
- `zmat & add_zmat (const zmat &B)`  
*Combine two Z-matrices. B.offset must be zero!*
- `void set_opt_val (long i, long j, bool val)`  
*Set the optimization flag for zmat\_entry i in position j.*
- `zmat & add_zmat (const zmat &B, const zmat_connector &e)`  
*Combine two Z-matrices. B.offset must be zero!*
- `zmat & concat_zmat (const zmat&B)`  
*Concat two Z-matrices with different offsets.*
- `zmat & correct_zmat (const zmat_entry &x, const long newoff)`
- `long count_constants () const`  
*Count the number of constants.*

- `long count_variables () const`  
*Count the number of variables.*
- `void set_constants_variables (const refvector< double > &consts, const refvector< double > &vars)`  
*Set the constants and variables of a Z-matrix from two refvectors.*
- `stringstream & zmat_to_string (long N, stringstream &output) const`  
*Output a Z-matrix to a stringstream.*
- `void update_variables (const zmat &B)`  
*Update the variables in the matrix without touching connectivity or increments.*

## Constructors

- `zmat ()`  
*Default constructor.*
- `zmat (const long o)`  
*Constructor which uses a non-zero offset.*
- `zmat (const zmat &a)`  
*Copy constructor.*
- `zmat (const refvector< zmat_entry > &z)`  
*Construction from a vector of zmat\_entries.*
- `zmat (stringstream &s)`  
*Construct from a stringstream.*

## Public Attributes

- `const refvector< zmat_entry > & list`  
*read-only access to list2*
- `const long & offset_r`  
*read-only access to offset.*

## Private Attributes

- `refvector< zmat_entry > list2`  
*List of zmat\_entries.*

- long offset  
*Auxiliary offset for connecting matrices.*

### 4.49.1 Detailed Description

The Z-matrix may have negative connectivity entries in the first upper triangle of definition. This is for the purpose of combining Z-matrices. These entries are ignored when the zmat is output. The zmat entries are always checked for consistency upon construction.

### 4.49.2 Constructor & Destructor Documentation

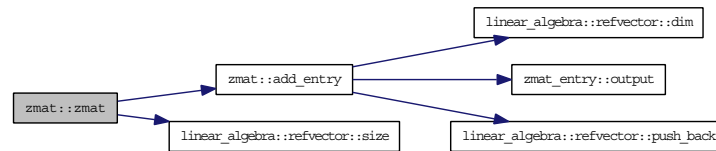
4.49.2.1 `zmat::zmat ()`[inline]

4.49.2.2 `zmat::zmat (const long o)`[inline]

4.49.2.3 `zmat::zmat (const zmat & a)`[inline]

4.49.2.4 `zmat::zmat (const refvector< zmat_entry > & z)`[inline]

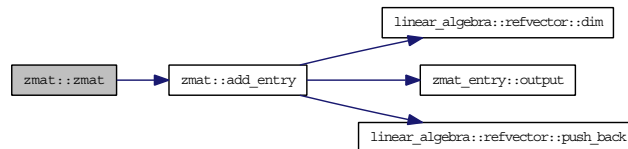
Here is the call graph for this function:



4.49.2.5 `zmat::zmat (stringstream & s)`[inline]

The format goes (zmat\_entry ... zmat\_entry)

Here is the call graph for this function:

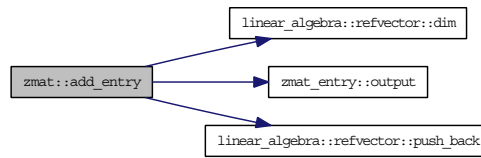


### 4.49.3 Member Function Documentation

4.49.3.1 `zmat& zmat::add_entry (const zmat_entry & e)`[inline]



Here is the call graph for this function:



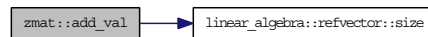
#### 4.49.3.2 `zmat& zmat::add_increment (long i, long j, double a)[inline]`

Here is the call graph for this function:



#### 4.49.3.3 `zmat& zmat::add_val (int i, int j, double val)[inline]`

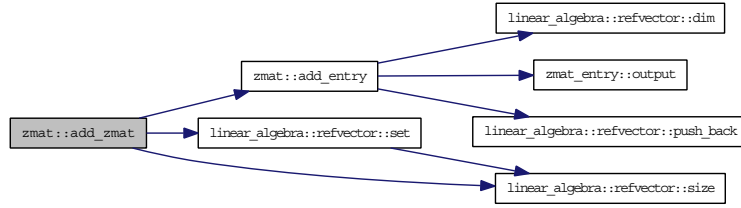
Here is the call graph for this function:



#### 4.49.3.4 `zmat& zmat::add_zmat (const zmat & B, const zmat_connector & e)[inline]`

The first entry is defined to be exactly `e` with the corresponding Name, i.e., `[ B.list[0].Name -1 e.connect[0] -2 e.connect[1] -3 e.connect[2] ]`.

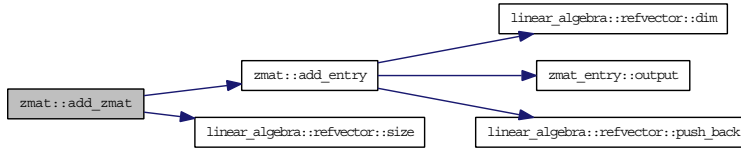
Here is the call graph for this function:



#### 4.49.3.5 `zmat& zmat::add_zmat (const zmat & B)[inline]`

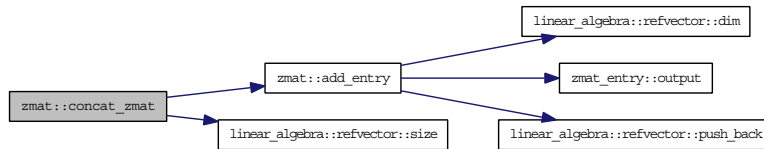
It is assumed that all B entries refer to the last three entries of `(*this)` or B itself.

Here is the call graph for this function:



#### 4.49.3.6 `zmat& zmat::concat_zmat (const zmat & B)[inline]`

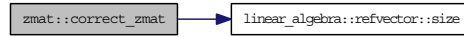
Here is the call graph for this function:



#### 4.49.3.7 `zmat& zmat::correct_zmat (const zmat_entry & x, const long newoff)[inline]`

This has the effect of inserting blank space between the original offset matrix and the new matrix as well as updating the connectors.

Here is the call graph for this function:



4.49.3.8 long zmat::count\_constants () const[inline]

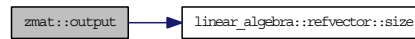
4.49.3.9 long zmat::count\_variables () const[inline]

4.49.3.10 zmat& zmat::operator= (const zmat & a)[inline]

4.49.3.11 bool zmat::operator== (const zmat & a) const[inline]

4.49.3.12 void zmat::output (ostream & out) const[inline]

Here is the call graph for this function:



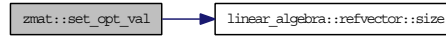
4.49.3.13 void zmat::set\_constants\_variables (const refvector< double > & consts, const refvector< double > & vars)[inline]

Here is the call graph for this function:



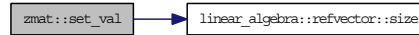
4.49.3.14 void zmat::set\_opt\_val (long i, long j, bool val)[inline]

Here is the call graph for this function:



4.49.3.15 `zmat& zmat::set_val (int i, int j, double val)[inline]`

Here is the call graph for this function:



4.49.3.16 `void zmat::update_variables (const zmat & B)[inline]`

Here is the call graph for this function:

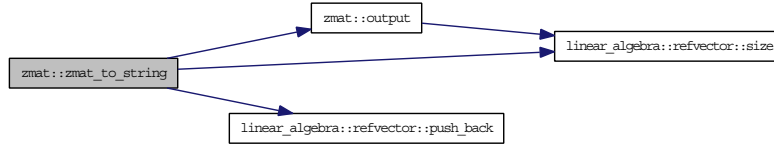


4.49.3.17 `stringstream& zmat::zmat_to_string (long N, stringstream & output) const[inline]`

### Parameters:

*N* signifies the conformation of the combinatorial product of options.

Here is the call graph for this function:



#### 4.49.4 Member Data Documentation

4.49.4.1 `const refvector<zmat_entry>& zmat::list`

4.49.4.2 `refvector<zmat_entry> zmat::list2[private]`

4.49.4.3 `long zmat::offset[private]`

4.49.4.4 `const long& zmat::offset_r`

The documentation for this class was generated from the following file:

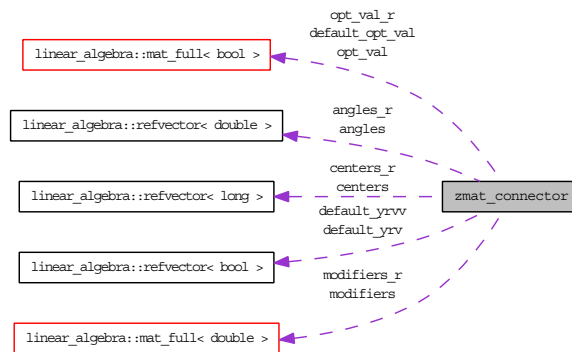
- `zmat.hh`

#### 4.50 zmat\_connector Class Reference

Class of connectors between z-matrices.

```
#include <zmat.hh>
```

Collaboration diagram for `zmat_connector`:



#### Public Member Functions

- `zmat_connector & operator= (const zmat_connector &A)`

*Assignment operator.*

- `bool operator== (const zmat_connector &A) const`  
*Comparison operator.*
- `stringstream & output (stringstream &s) const`  
*Output the connector to a stringstream.*
- `void set_opt_val (long i, long j, bool v)`  
*Set the optimization flag .*
- `void add_angle (double i)`  
*Add an angle to the.*

## Constructors

- `zmat_connector ()`  
*Default constructor.*
- `zmat_connector (const zmat_entry &a)`  
*Construct from a zmat\_entry.*
- `zmat_connector (const zmat_entry &a, const mat_full< bool > &nopt_val)`  
*Construct from a zmat\_entry and a set of optimization flags.*
- `zmat_connector (stringstream &s)`  
*Construct from an input string stream.*
- `zmat_connector (const zmat_connector &A)`  
*Construct from another zmat\_connector.*

## Static Public Member Functions

- `static zmat_connector & update_connector (const zmat_connector &a, const zmat_connector &e, const long add, zmat_connector &x)`  
*Update a connector to fit after two Z-matrices have been combined.*
- `static zmat_connector & update_connector (const zmat_connector &a, const long add, zmat_connector &x)`  
*Update a connector to fit after two Z-matrices have been combined.*

## Public Attributes

- `const refvector< long > & centers_r`  
*Translation vector of -3, -2, -1 to alternate centers. (READ ONLY).*
- `const mat_full< double > & modifiers_r`  
*Modification matrix. Depending on the use of each center modifications may be different. (READ ONLY).*
- `const mat_full< bool > & opt_val_r`  
*Matrix of flags to set optimization.*
- `const refvector< double > & angles_r`  
*deg/angle for conformational purposes. (READ ONLY)*

## Static Public Attributes

- `static const bool default_y [3] = {false, false, false}`
- `static const refvector< bool > default_yrv`
- `static const refvector< bool > default_yrvv [3]`
- `static const refvector< refvector< bool > > default_yrvrv`
- `static const mat_full< bool > default_opt_val`

## Private Attributes

- `refvector< long > centers`  
*Translation vector of -3, -2, -1 to alternate centers.*
- `mat_full< double > modifiers`  
*Modification matrix. Depending on the use of each center modifications may be different.*
- `mat_full< bool > opt_val`  
*Matrix of flags to set optimization.*
- `refvector< double > angles`  
*deg/angle for conformational purposes.*

### 4.50.1 Detailed Description

A `zmat_connector` has three distinct references:

- `-6` to `-4` refer to the original connector of the following group. (default)
- `-3` to `-1` refer to values of the connector passed down to the current group.
- `>-1` refers to values in the local Z-matrix.

### 4.50.2 Constructor & Destructor Documentation

4.50.2.1 `zmat_connector::zmat_connector ()`

4.50.2.2 `zmat_connector::zmat_connector (const zmat_entry & a)`

4.50.2.3 `zmat_connector::zmat_connector (const zmat_entry & a, const mat_full< bool > & nopt_val)`

4.50.2.4 `zmat_connector::zmat_connector (stringstream & s)`

The connector information is generated from: ( C1,C2,C3)

(V11,V12,V13)

(V21,V22,V23)

(V31,V32,V33)

(O11,O12,O13)

(O11,O12,O13)

(O11,O12,O13)

(Oa,Ob)

Ci are integers referencing connectors, Vij are variables depending on the position Ci in a `zmat_entry` Oij is a flag 0/1 for optimization of this variable

4.50.2.5 `zmat_connector::zmat_connector (const zmat_connector & A)`

### 4.50.3 Member Function Documentation

4.50.3.1 `void zmat_connector::add_angle (double i`

4.50.3.2 `zmat_connector & zmat_connector::operator= (const zmat_connector & A)`

4.50.3.3 `bool zmat_connector::operator== (const zmat_connector & A) const`



4.50.3.4 `stringstream & zmat_connector::output (stringstream & s) const`

4.50.3.5 `void zmat_connector::set_opt_val (long i, long j, bool v)`

4.50.3.6 `zmat_connector & zmat_connector::update_connector (const zmat_connector & a, const long add, zmat_connector & x)[static]`

4.50.3.7 `zmat_connector & zmat_connector::update_connector (const zmat_connector & a, const zmat_connector & e, const long add, zmat_connector & x)[static]`

## 4.50.4 Member Data Documentation

4.50.4.1 `refvector<double> zmat_connector::angles[private]`

4.50.4.2 `const refvector<double>& zmat_connector::angles_r`

4.50.4.3 `refvector<long> zmat_connector::centers[private]`

4.50.4.4 `const refvector<long>& zmat_connector::centers_r`

4.50.4.5 `const mat_full< bool > zmat_connector::default_opt_val[static]`

4.50.4.6 `const bool zmat_connector::default_y = {false, false, false}[static]`

4.50.4.7 `const refvector< bool > zmat_connector::default_yrv[static]`

4.50.4.8 `const refvector< refvector< bool > > zmat_connector::default_yrvrv[static]`

4.50.4.9 `const refvector< bool > zmat_connector::default_yrvv[static]`

### Initial value:

`refvector<bool>(3),  
zmat_connector::default_yrv,  
zmat_connector::default_yrv`

4.50.4.10 `mat_full<double> zmat_connector::modifiers[private]`

4.50.4.11 `const mat_full<double>& zmat_connector::modifiers_r`

4.50.4.12 `mat_full<bool> zmat_connector::opt_val[private]`

4.50.4.13 `mat_full<bool>& zmat_connector::opt_val_r`

The documentation for this class was generated from the following files:

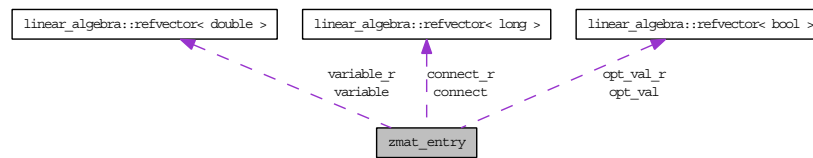
- zmat.hh
- zmat.cc

## 4.51 zmat\_entry Class Reference

Class of Z-matrix entries.

#include <zmat.hh>

Collaboration diagram for zmat\_entry:



### Public Member Functions

- zmat\_entry & operator= (const zmat\_entry &a)  
*Copy assignment.*
- bool operator== (const zmat\_entry &a) const  
*Comparison operator.*
- stringstream & output (stringstream &s) const  
*Output the entry to a stringstream.*
- void update\_variables (const zmat\_entry &b)  
*Update the variables in the zmat\_entry without touching connectivity or increments.*

### Constructors

- zmat\_entry ()  
*Default constructor. zmat\_entry (const zmat\_entry &a)*  
*Copy constructor. zmat\_entry (const string &N)*  
*Named default construction.*
- zmat\_entry (const string &N, const double \*v, const int \*c)  
*Construction with full initialization.*

- `zmat_entry (const string &N, const double *v)`  
*Construction with variable initialization.*
- `zmat_entry (const string &N, const refvector< double > &v)`  
*Construction with variable initialization.*
- `zmat_entry (const string &N, const refvector< double > &v, const refvector< long > &c)`  
*Construction from explicit values.*
- `zmat_entry (stringstream &s)`  
*Construction from a stringstream.*

## Public Attributes

- `const string & Name_r`  
*Atom name.*
- `const refvector< double > & variable_r`  
*Array of variables.*
- `const refvector< refvector< double > > & increment_r`  
*Potential alternate values for each variable.*
- `const refvector< long > & connect_r`  
*Connectivity data.*
- `const refvector< bool > & opt_val_r`  
*Optimization flags for each variable.*

## Static Public Attributes

- `static const bool default_opt_val [3] = {false,false, false}`

## Private Attributes

- `string Name`  
*Atom name.*
- `refvector< double > variable`  
*Array of variables.*

- `refvector< refvector< double > > increment`  
*Potential alternate values for each variable.*
- `refvector< long > connect`  
*Connectivity data.*
- `refvector< bool > opt_val`  
*Optimization flags for each variable.*

## Friends

- `class zmat`

### 4.51.1 Constructor & Destructor Documentation

4.51.1.1 `zmat_entry::zmat_entry ()`

4.51.1.2 `zmat_entry::zmat_entry (const zmat_entry & a)`

4.51.1.3 `zmat_entry::zmat_entry (const string & N)`

4.51.1.4 `zmat_entry::zmat_entry (const string & N, const double * v, const int * c)`

4.51.1.5 `zmat_entry::zmat_entry (const string & N, const double * v)`

4.51.1.6 `zmat_entry::zmat_entry (const string & N, const refvector< double > & v)`

4.51.1.7 `zmat_entry::zmat_entry (const string & N, const refvector< double > & v, const refvector< long > & textit c)`

4.51.1.8 `zmat_entry::zmat_entry (stringstream & s)`

Each entry in the string has the followin form: (Name, Integer(f), ength(double,...,double), Integer(f), Angle(double,...,double), Integer(f), Dihedral(double,...,double))

- Name is a string.
- f is a flag of 0 or 1. The inner parentheses are only needed if specific values are given.

### 4.51.2 Member Function Documentation

4.51.2.1 `zmat_entry & zmat_entry::operator= (const zmat_entry & a)`

4.51.2.2 `bool zmat_entry::operator==(const zmat_entry & a) const`

4.51.2.3 `stringstream & zmat_entry::output (stringstream & s) const`

4.51.2.4 `void zmat_entry::update_variables (const zmat_entry & b)`

### 4.51.3 Friends and Related Function Documentation

4.51.3.1 `friend class zmat[friend]`

### 4.51.4 Member Data Documentation

4.51.4.1 `refvector<long> zmat_entry::connect[private]`

4.51.4.2 `const refvector<long>& zmat_entry::connect_r`

4.51.4.3 `const bool zmat_entry::default_opt_val = {false,false, false}[static]`

4.51.4.4 `refvector<refvector<double> > zmat_entry::increment[private]`

4.51.4.5 `const refvector<refvector<double> >& zmat_entry::increment_r`

4.51.4.6 `string zmat_entry::Name[private]`

4.51.4.7 `const string& zmat_entry::Name_r`

4.51.4.8 `refvector<bool> zmat_entry::opt_val[private]`

4.51.4.9 `refvector<bool>& zmat_entry::opt_val_r`

4.51.4.10 `refvector<double> zmat_entry::variable[private]`

4.51.4.11 `const refvector<double>& zmat_entry::variable_r`

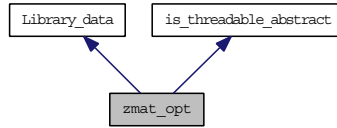
The documentation for this class was generated from the following files:

- `zmat.hh`
- `zmat.cc`

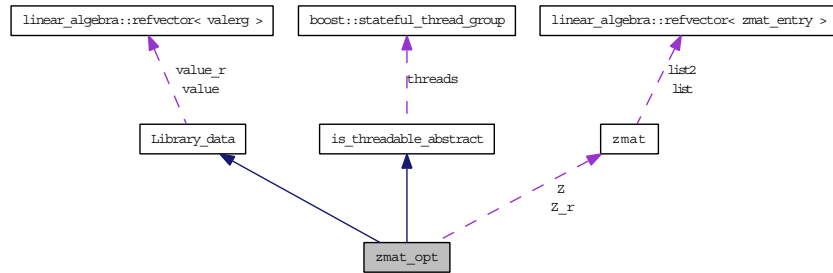
## 4.52 zmat\_opt Class Reference

```
#include <zmat_opt.hh>
```

Inheritance diagram for zmat\_opt:



Collaboration diagram for zmat\_opt:



## Public Member Functions

- `const zmat & operator= (const zmat &A)`  
*Assignment operator for a zmat. valerg compute\_property along i) const*  
*Compute the property. (Here energy is important).*
- `valerg compute_property (ulong i, zmat &A) const`  
*Compute the property. (Here energy is important).*
- `valerg compute_energy (ulong i) const`  
*Compute the energy. (Here energy is important).*
- `valerg compute_energy (ulong i, zmat &A) const`  
*Compute the energy. (Here energy is important).*
- `ulong get_space_size () const`  
*Compute the size of the optimization space.*
- `ulong get_bits () const`  
*Compute the number of bits to address the optimization space.*
- `bool pre_opt (ulong N) const`  
*Find a converged starting geometry.*

## Constructors

- `zmat_opt ()`  
*Default constructor. `zmat_opt (const zmat_opt &A)`*  
*Copy constructor. `zmat_opt (const zmat &A)`*  
*Initialize with a `zmatrix`.*

## Public Attributes

- `const zmat & Z_r`  
*Read-only access to `Z`.*
- `bool compute_property_flag`

## Private Attributes

- `zma`

### 4.52.1 Detailed Description

`zmat_opt` describes a threadable Library using class `zmat`. It has two associated properties as expressed by `compute_energy(ulong i)` and `compute_property(ulong i)`. Set `compute_property_flag` in order to compute the property and energy of a given conformation.

### 4.52.2 Constructor & Destructor Documentation

4.52.2.1 `zmat_opt::zmat_opt ()`

4.52.2.2 `zmat_opt::zmat_opt (const zmat_opt & A)`

4.52.2.3 `zmat_opt::zmat_opt (const zmat & A)`

### 4.52.3 Member Function Documentation

4.52.3.1 `valerg zmat_opt::compute_energy (ulong i, zmat & A) const`

This method is necessary for initialisation purposes.

4.52.3.2 `valerg zmat_opt::compute_energy (ulong i) const`

4.52.3.3 `valerg zmat_opt::compute_property (ulong i, zmat & A) const`

This method is necessary for initialisation purposes.

4.52.3.4 `valerg zmat_opt::compute_property (ulong i) const[virtual]`

Implements `Library_data`.

4.52.3.5 `ulong zmat_opt::get_bits () const[virtual]`

Implements `Library_data`.

4.52.3.6 `ulong zmat_opt::get_space_size () const[virtual]`

Implements `Library_data`.

4.52.3.7 `const zmat & zmat_opt::operator= (const zmat & A)`

4.52.3.8 `bool zmat_opt::pre_opt (ulong N) const`

#### 4.52.4 Member Data Documentation

4.52.4.1 `bool zmat_opt::compute_property_flag`

Changes the behavior of `compute_property`.

- `true`: compute property and energy
- `false`: compute energy only.

4.52.4.2 `zmat zmat_opt::Z[mutable, private]`

4.52.4.3 `const zmat& zmat_opt::Z_r`

The documentation for this class was generated from the following files:

- `zmat_opt.hh`
- `zmat_opt.cc`



---

## 5. Discrete Optimization in Chemical Space File Documentation

---

### 5.1 `binary_entropic.hh` File Reference

Definition and Implementation of the `binary_entropic` class.

```
#include <typedefs.hh>
#include <concepts.hh>
#include <iostream>
#include <optimizeabstract.h>
#include <cmath>
#include <BCR_CPP_LA/linear_algebra.h>
#include <entropic_aux.hh>
```

#### Namespaces

- namespace `std`

#### Classes

- class `binary_entropic< C >`  
*Class for enhanced sampling using an entropic measure of coverage.*

#### 5.1.1 Detailed Description

### 5.2 `binary_line_search.hh` File Reference

implementation of a line search optimization

```
#include <concepts.hh>
#include <optimizeabstract.h>
```

#### Classes

- class `binary_line_search< C >`  
*Search a pruned Library using a linesearch algorithm.*

### 5.2.1 Detailed Description

## 5.3 binary\_opt.cc File Reference

```
#include <BCR_CPP_LA/refcount.h>
#include <chemgroup.hh>
#include <chem_opt.hh>
#include <Library_data.hh>
#include <zmat.hh>
#include <sstream>
#include <fstream>
#include <iostream>
#include <cmath>
#include <noprune.h>
#include <simpleprune.h>
#include <simpleprune.cc>
#include <optimizeabstract.h>
#include <binary_line_search.hh>
#include <binarysteepestdescent.hh>
#include <binarygdmc.hh>
#include <genbase-l-s.hh>
#include <genbase-grad-ls.hh>
#include <boost/concept_check.hpp>
#include <concepts.hh>
#include <reordergeneralbase.hh>
#include <generalbaseiterator.hh>
#include <genbasegdmc.hh>
#include <gen_base_entropic.hh>
#include <binary_entropic.hh>
```

### Functions

- `valerg calc_property (const zmat &A, const string &out, const string &id, zmat &returnA)`  
*Setup the external computations and execute them.*
- `template<class X> void print (const X &D)`
- `template<class X> ulong run (const X &D)`
- `int main (int argc, char *argv[])`  
*Main execution routine. Handles command-line input and executes optimization.*

### 5.3.1 Function Documentation

5.3.1.1 `valerg calc_property (const zmat & A, const string & out, const string & id, zmat & returnA)`

5.3.1.2 `int main (int argc, char * argv[])`

5.3.1.3 `template<class X> void print (const X & D)`

5.3.1.4 `template<class X> ulong run (const X & D)`

### 5.4 binarygdmc.hh File Reference

Implementation of binary GDMC.

```
#include <typedefs.hh>
#include <concepts.hh>
#include <iostream>
#include <optimizeabstract.h>
```

#### Classes

- `class binary_gdmc< C >`  
*Gradient-directed Monte Carlo performed on a hypercube.*

#### 5.4.1 Detailed Description

### 5.5 binarysteepestdescent.hh File Reference

Implements the `binary_steepest_descent` optimization class.

```
#include <optimizeabstract.h>
#include <concepts.hh>
#include <BCR_CPP_LA/refcount.h>
#include <has_gradients_hessian_data.hh>
```

#### Classes

- `class binary_steepest_descent< C >`  
*Searches a Library for an optimum using a steepest descent method.*

### 5.5.1 Detailed Description

## 5.6 chem\_opt.cc File Reference

Implementation of chemical optimization class.

```
#include <BCR_CPP_LA/refcount.h>
#include <chemgroup.hh>
#include <zmat_opt.hh>
#include <chem_opt.hh>
#include <cmath>
#include <binary_line_search.hh>
```

### Defines

- #define CLASS template <class P> chem\_opt

### 5.6.1 Detailed Description

### 5.6.2 Define Documentation

5.6.2.1 #define CLASS template <class P> chem\_opt

## 5.7 chem\_opt.hh File Reference

```
#include <BCR_CPP_LA/refcount.h>
#include <typedefs.hh>
#include <chemgroup.hh>
#include <zmat.hh>
#include <Library_data.hh>
#include <isthreadableabstract.h>
#include <concepts.hh>
#include <boost/concept_check.hpp>
```

### Classes

- chem\_opt  
*Chemical optimization class.*

## 5.8 chemgroup.cc File Reference

Implementation file of class ChemGroup.

```
#include <BCR_CPP_LA/refcount.h>
#include <chemgroup.hh> %

#include <zmat.hh>
#include <string>
#include <sstream>
#include <iostream>
#include <cmath>
```

### Functions

- `bool filter (const ChemGroup &A)`
- `refvector< unsigned long > & sort_descending (const refvector< double > &E, unsigned long start, unsigned long end, refvector< unsigned long > &index)`  
*Do a qsort on E with immediate storage back into index. (perfectly parallelizable).*
- `refvector< unsigned long > & sort_descending (const refvector< double > &E, refvector< unsigned long > &index)`
- `static void spaces (long n)`  
*outputs a number of spaces*

### 5.8.1 Detailed Description

### 5.8.2 Function Documentation

5.8.2.1 `bool filter (const ChemGroup & A)`

5.8.2.2 `refvector<unsigned long>& sort_descending (const refvector< double > & E, refvector< unsigned long > & index)`

5.8.2.3 `refvector<unsigned long>& sort_descending (const refvector< double > & E, unsigned long start, unsigned long end, refvector< unsigned long > & index)`

5.8.2.4 `static void spaces (long n)[static]`

## 5.9 chemgroup.hh File Reference

Header file for ChemGroup class.

```
#include <BCR_CPP_LA/refcount.decl>
#include <zmat.hh>
#include <iostream>
#include <sstream>
#include <typedefs.hh>
#include <chemident.hh>
```

### Classes

- class ChemGroup

*This class describes a group of substitution sites and their respective substitution options as well as computation history.*

### Typedefs

- typedef unsigned long ulong

#### 5.9.1 Detailed Description

#### 5.9.2 Typedef Documentation

##### 5.9.2.1 typedef unsigned long ulong

## 5.10 chemident.cc File Reference

Implementation file of class ChemIdent.

```
#include <BCR_CPP_LA/refcount.h>
#include <chemident.hh>
#include <zmat.hh>
#include <zmat_opt.hh>
#include <string>
#include <sstream>
#include <iostream>
#include <cmath>
```

### Functions

- static void (long n)

## Variables

- static const double zeros [3] = {0.0,0.0,0.0}
- static const int conn [3] = {-6,-5,-4}
- static const zmat\_connector default\_return\_connector (zmat\_entry("-", zeros, conn))

### 5.10.1 Detailed Description

### 5.10.2 Function Documentation

5.10.2.1 static void spaces (long n)[static]

### 5.10.3 Variable Documentation

5.10.3.1 const int conn[3] = {-6,-5,-4}[static]

5.10.3.2 const zmat\_connector default\_return\_connector(zmat\_entry("-", zeros, conn))[static]

5.10.3.3 const double zeros[3] = {0.0,0.0,0.0}[static]

## 5.11 chemident.hh File Reference

Header file for ChemIdent class.

```
#include <BCR_CPP_LA/linear_algebra.h>
#include <zmat.hh>
#include <zmat_opt.hh>
#include <iostream>
#include <sstream>
#include <typedefs.hh>
```

## Classes

- class ChemIdent  
*This class describes a group of substitution sites on a Z-matrix and their respective substitution options.*

### 5.11.1 Detailed Description

## 5.12 concepts.hh File Reference

```
#include <boost/concept_check.hpp>
#include <BCR_CPP_LA/refcount.decl>
#include <BCR_CPP_LA/mat_sym_full.decl>
#include <typedefs.hh>
#include <boost/thread/shared_mutex.hpp>
#include <stateful_thread_group.hh>
#include <string>
```

### Namespaces

- namespace Concepts

### Classes

- class Concepts::Library< X >  
*Concept defines an addressable set of values.*
- class Concepts::base\_iterator< X >  
*Concept defines an iterator for a class of bases.*
- class Concepts::pruner< X >  
*Concept defines an addressable set of values.*
- class Concepts::has\_gradients< X >  
*Concept defines a class that has gradients.*
- class Concepts::has\_hessians< X >  
*Concept defines a class that has hessians.*
- class Concepts::has\_Name< X >  
*Concept defines a class that has a private string Name.*
- class Concepts::is\_threadable< X >  
*Concept defines a class that is threadable.*
- class Concepts::has\_optimize< X >  
*Concept defines a class that has an optimize member.*
- class Concepts::has\_stacksize< X >  
*Concept defines a class that has a stack member.*



## Functions

- `template<typename T> void Concepts::same_type (T const &, T const &)`  
*Dummy function to determine type equality.*

### 5.12.1 Detailed Description

This file contains the assembled concepts used in the project Discrete Molecular Optimization.

### 5.13 entropic\_aux.cc File Reference

```
#include <iostream>
#include <cmath>
#include <BCR_CPP_LA/linear_algebra.h>
```

## Functions

- `void linsolve_cg (const mat_sym_full< double > &J, const refvector< double > &G, refvector< double > &X)`  
*Conjugate gradients for linear solver.*
- `refvector< double > & set_gradient (const refvector< long > &b, const mat_full< double > &H, const refvector< double > &X, refvector< double > &G)`  
*We ignore infinities in the derivatives.*
- `mat_sym_full< double > & set_hessian (const refvector< long > &b, const mat_full< double > &H, const refvector< double > &X, mat_sym_full< double > &J)`  
*We ignore infinities in the derivatives.*
- `ulong maximize_entropic_distance (const refvector< ulong > &A, const refvector< long > &b)`  
*Maximize the entropic distance as declared in set\_gradient() using Newton-Raphson.*

### 5.13.1 Function Documentation

5.13.1.1 `const mat_sym_full< double > & J, const refvector< double > & G, refvector< double > & X)`

5.13.1.2 `ulong maximize_entropic_distance (const refvector< ulong > & A, const refvector< long > & b)`

5.13.1.3 `refvector<double>& set_gradient (const refvector< long > & b, const mat_full< double > & H, const refvector< double > & X, refvector< double > & G)`

Compute the gradient of the distance function  $f = \sum_i \ln \sqrt{\sum_j \sin(x_j - s_j^{(i)} \pi / n_j)^2}$  where  $n_j$  is the number of digits in component  $j$ .

5.13.1.4 `mat_sym_full<double>& set_hessian (const refvector< long > & b, const mat_full< double > & H, const refvector< double > & X, mat_sym_full< double > & J)`

Compute the hessian of the distance function  $f = \sum_i \ln \sqrt{\sum_j \sin(x_j - s_j^{(i)} \pi / n_j)^2}$  where  $n_j$  is the number of digits in component  $j$ .

## 5.14 `entropic_aux.hh` File Reference

Auxiliary functions for `*_entropic` classes.

```
#include <iostream>
#include <cmath>
#include <BCR_CPP_LA/linear_algebra.h>
```

### Functions

- `void linsolve_cg (const mat_sym_full< double > &J, const refvector< double > &G, refvector< double > &X)`  
*Conjugate gradients for linear solver.*
- `refvector< double > & set_gradient (const refvector< long > &b, const mat_full< double > &H, const refvector< double > &X, refvector< double > &G)`  
*We ignore infinities in the derivatives.*
- `mat_sym_full< double > & set_hessian (const refvector< long > &b, const mat_full< double > &H, const refvector< double > &X, mat_sym_full< double > &J)`  
*We ignore infinities in the derivatives.*
- `ulong maximize_entropic_distance (const refvector< ulong > &A, const refvector< long > &b)`  
*Maximize the entropic distance as declared in `set_gradient()` using Newton-Raphson.*

### 5.14.1 Detailed Description

### 5.14.2 Function Documentation

5.14.2.1 void linsolve\_cg (const mat\_sym\_full< double > & J, const refvector< double > & G, refvector< double > & X)

5.14.2.2 ulong maximize\_entropic\_distance (const refvector< ulong > & A, const refvector< long > & b)

5.14.2.3 refvector<double>& set\_gradient (const refvector< long > & b, const mat\_full< double > & H, const refvector< double > & X, refvector< double > & G)

Compute the gradient of the distance function  $f = \sum_i \ln \sqrt{\sum_j \sin(x_j - s_j^{(i)} \pi / n_j)^2}$  where  $n_j$  is the number of digits in component j.

5.14.2.4 mat\_sym\_full<double>& set\_hessian (const refvector< long > & b, const mat\_full< double > & H, const refvector< double > & X, mat\_sym\_full< double > & J)

Compute the hessian of the distance function  $f = \sum_i \ln \sqrt{\sum_j \sin(x_j - s_j^{(i)} \pi / n_j)^2}$  where  $n_j$  is the number of digits in component j.

## 5.15 gen\_base\_entropic.hh File Reference

```
#include <typedefs.hh>
#include <concepts.hh>
#include <iostream>
#include <optimizeabstract.h>
#include <cmath>
#include <BCR_CPP_LA/linear_algebra.h>
#include <entropic_aux.hh>
```

### Classes

- class gen\_base\_entropic< C >  
*Meta-Optimize by generating maximally distant starting configurations.*

## 5.16 genbase-grad-ls.hh File Reference

Line search optimization with general bases.

```
#include <typedefs.hh>
#include <concepts.hh>
#include <iostream>
#include <optimizeabstract.h>
#include <BCR_CPP_LA/refcount.h>
```

## Classes

- class gen\_base\_grad\_LS< C, B >  
*Line search optimization class using general bases.*

### 5.16.1 Detailed Description

## 5.17 genbase-l-s.hh File Reference

Line search optimization with general bases.

```
#include <typedefs.hh>
#include <concepts.hh>
#include <iostream>
#include <optimizeabstract.h>
#include <BCR_CPP_LA/refcount.h>
```

## Classes

- class gen\_base\_LS< C, B >
- *Line search optimization class using general bases.*

### 5.17.1 Detailed Description

## 5.18 generalbaseiterator.cc File Reference

TODO: Throw exceptions.

```
#include <chem_opt.hh>
#include <chemident.hh>
#include <generalbaseiterator.hh>
```

### 5.18.1 Detailed Description

## 5.19 generalbaseiterator.hh File Reference

```
#include <concepts.hh>
```

```
#include <BCR_CPP_LA/refcount.h>
```

```
#include <typedefs.hh>
```

```
#include <chemident.hh>
```

### Classes

- class general\_base\_iterator< X >  
*Iterator over the potential bases.*

### 5.19.1 Detailed Description

## 5.20 has\_gradients\_hessian\_data.decl File Reference

```
#include <BCR_CPP_LA/refcount.decl>
```

```
#include <BCR_CPP_LA/mat_sym_full.decl>
```

```
#include "typedefs.hh"
```

```
#include <string>
```

```
#include <boost/thread.hpp>
```

```
#include <stateful_thread_group.hh>
```

```
#include "concepts.hh"
```

### Classes

- class has\_gradients\_data< X >  
*Abstract class for discrete optimizations.*
- class has\_hessians\_data< X >  
*Abstract class for discrete optimizations.*

## 5.21 has\_gradients\_hessian\_data.hh File Reference

```
#include <has_gradients_hessian_data.decl>
```

```
#include <BCR_CPP_LA/refcount.h>
#include <BCR_CPP_LA/mat_sym_full.h>
#include <typedefs.hh>
#include <stdexcept>
```

## 5.22 include/BCR\_CPP\_LA/ABmBA.hh File Reference

```
#include <linear_algebra.h>
```

### Namespaces

- namespace linear\_algebra

### Functions

- template<class TN> mat\_sym\_full< TN > & linear\_algebra::ABmBA (const mat\_sym\_full< TN > A, const mat\_asym\_full< TN > B, mat\_sym\_full< TN > &r)  
*Computes the commutator AB-BA.*
- template<class TN> mat\_asym\_full< TN > linear\_algebra::ABmBA (const mat\_sym\_full< TN > A, const mat\_asym\_full< TN > B)  
*Computes the commutator AB-BA.*
- template<class TN> mat\_asym\_full< TN > & linear\_algebra::ABmBA (const mat\_asym\_full< TN > A, const mat\_sym\_full< TN > B, mat\_asym\_full< TN > &r)  
*Computes the commutator AB-BA.*
- template<class TN> mat\_asym\_full< TN > linear\_algebra::ABmBA (const mat\_asym\_full< TN > A, const mat\_sym\_full< TN > B)  
*Computes the commutator AB-BA.*
- template<class TN> mat\_asym\_full< TN > & linear\_algebra::ABmBA (const mat\_asym\_full< TN > &A, const mat\_asym\_full< TN > &B, mat\_asym\_full< TN > &r)  
*Computes the commutator AB-BA.*
- template<class TN> mat\_asym\_full< TN > linear\_algebra::ABmBA (const mat\_asym\_full< TN > &A, const mat\_asym\_full< TN > &B)  
*Computes the commutator AB-BA.*

### 5.22.1 Detailed Description

## 5.23 include/BCR\_CPP\_LA/ABpBA.hh File Reference

#include <linear\_algebra.h>

### Namespaces

- namespace linear\_algebra

### Functions

- template<class TN> mat\_asym\_full< TN > & linear\_algebra::ABpBA (const mat\_sym\_full< TN > &A, const mat\_asym\_full< TN > &B, mat\_asym\_full< TN > &r)  
*Computes the anti-commutator  $AB+BA$ .*
- template<class TN> mat\_asym\_full< TN > linear\_algebra::ABpBA (const mat\_sym\_full< TN > &A, const mat\_asym\_full< TN > &B)  
*Computes the anti-commutator  $AB+BA$ .*
- template<class TN> mat\_asym\_full< TN > linear\_algebra::ABpBA (const mat\_full< TN > &A, const mat\_asym\_full< TN > &B, mat\_asym\_full< TN > &r)  
*Computes the anti-commutator  $AB+BA$ .*
- template<class TN> mat\_asym\_full< TN > linear\_algebra::ABpBA (const mat\_full< TN > A, const mat\_asym\_full< TN > B)  
*Computes the anti-commutator  $AB+BA$ .*
- template<class TN> mat\_asym\_full< TN > & linear\_algebra::ABpBA (const mat\_asym\_full< TN > A, const mat\_sym\_full< TN > B, mat\_asym\_full< TN > &r)  
*Computes the anti-commutator  $AB+BA$ .*
- template<class TN> mat\_asym\_full< TN > linear\_algebra::ABpBA (const mat\_asym\_full< TN > A, const mat\_sym\_full< TN > B)  
*Computes the anti-commutator  $AB+BA$ .*
- template<class TN> mat\_asym\_full< TN > & linear\_algebra::ABpBA (const mat\_sym\_full< TN > A, const mat\_asym\_sparse< TN > B, mat\_asym\_full< TN > &r)  
*Computes the anti-commutator  $AB+BA$ .*

- `template<class TN> mat_asym_full< TN > linear_algebra::ABpBA (const mat_sym_full< TN > A, const mat_asym_sparse< TN > B)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > & linear_algebra::ABpBA (const mat_asym_sparse< TN > A, const mat_sym_full< TN > B, mat_asym_full< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > linear_algebra::ABpBA (const mat_asym_sparse< TN > A, const mat_sym_full< TN > B)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_sym_full< TN > & linear_algebra::ABpBA (const mat_sym_full< TN > A, mat_sym_full< TN > B, mat_sym_full< TN > &r, mat_sym_full< TN > &I, mat_full< TN > &I2)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_sym_full< TN > & linear_algebra::ABpBA (const mat_sym_full< TN > A, mat_sym_full< TN > B, mat_sym_full< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_sym_full< TN > & linear_algebra::ABpBA (const mat_sym_full< TN > &A, const mat_sym_full< TN > &B, mat_sym_full< TN > &r, mat_sym_full< TN > &I)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_sym_full< TN > linear_algebra::ABpBA (const mat_sym_full< TN > &A, const mat_sym_full< TN > &B)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_sym_full< TN > & linear_algebra::ABpBA (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_sym_full< TN > &r, mat_full< TN > &I2)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_sym_full< TN > & linear_algebra::ABpBA (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_sym_full< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_sym_full< TN > linear_algebra::ABpBA (const mat_sym_full< TN > &A, const mat_sym_sparse< TN > &B)`  
*Computes the anti-commutator  $AB+BA$ .*



- `template<class TN> mat_asym_full< TN > & linear_algebra::ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_full< TN > &r, mat_full< TN > &I2)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_sparse< TN > & linear_algebra::ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_sparse< TN > &r, mat_sparse< TN > &I2)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > & linear_algebra::ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_full< TN > &r, mat_sparse< TN > &I2)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > & linear_algebra::ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_full< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_sparse< TN > & linear_algebra::ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B, mat_asym_sparse< TN > &r)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class TN> mat_asym_full< TN > linear_algebra::ABpBA (const mat_asym_full< TN > &A, const mat_sym_sparse< TN > &B)`  
*Computes the anti-commutator  $AB+BA$ .*
- `template<class T> mat_sym_full< T > linear_algebra::ABpBA (const mat_asym_full< T > &A, const mat_asym_full< T > &B)`  
*Commutes the anti-commutator  $AB+BA$ . Lame and lazy version.*

### 5.23.1 Detailed Description

## 5.24 `include/BCR_CPP_LA/asymmetric.decl` File Reference

Provide declarations for antisymmetric full matrices.

```
#include <vector>
#include <linear_algebra.decl>
#include <matrix.decl>
#include <square.decl>
```

## Namespaces

- namespace `linear_algebra`

## Classes

- class `linear_algebra::asymmetric< TN >`  
*Class of antisymmetric matrices.*

### 5.24.1 Detailed Description

#### Todo

throw exceptions.

### 5.25 `include/BCR_CPP_LA/asymmetric.h` File Reference

Provide manipulations for antisymmetric full matrices.

```
#include <vector>
#include <square.h>
#include <asymmetric.decl>
```

## Namespaces

- namespace `linear_algebra`

### 5.25.1 Detailed Description

#### Todo

throw exceptions.

### 5.26 `include/BCR_CPP_LA/class_extensions.decl` File Reference

Declarations of general expansions for LA.

```
#include <stdexcept>
#include <sstream>
#include <iostream>
#include <linear_algebra.decl>
```

## Namespaces

- namespace `linear_algebra`

## Functions

- `template<class TN> matrix< TN > & linear_algebra::zero (matrix< TN > &r)`
- `template<class TN> mat_full< TN > & linear_algebra::zero (mat_full< TN > &r)`
- `template<class TN> mat_sym_full< TN > & linear_algebra::zero (mat_sym_full< TN > &r)`
- `template<class TN> mat_asym_full< TN > & linear_algebra::zero (mat_asym_full< TN > &r)`
- `template<class TN> refvector< TN > & linear_algebra::zero (refvector< TN > &r)`
- `double & linear_algebra::zero (double &r)`
- `double & linear_algebra::one (double &r)`
- `template<class TN> matrix< TN > & linear_algebra::one (matrix< TN > &r)`
- `void linear_algebra::copy (double &a, const double &b, const long i)`
- `void linear_algebra::copy (double &a, const double &b)`
- `void linear_algebra::copy (long &a, const long &b, const long i)`
- `void linear_algebra::copy (long &a, const long &b)`
- `template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > &a, mat_sym_sparse< TN > &b, const long i)`
- `template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > &a, const mat_sym_sparse< TN > &b)`
- `template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > &a, mat_sym_sparse< TN > &b)`
- `template<class TN> void linear_algebra::copy (mat_sym_full< TN > &a, mat_sym_full< TN > &b, const long i)`
- `template<class TN> void linear_algebra::copy (mat_sym_full< TN > &a, const mat_sym_full< TN > &b)`

- `template<class TN> void linear_algebra::copy (mat_sym_full< TN > &a, mat_sym_full< TN > &b)`
- `template<class TN> void linear_algebra::copy (mat_full< TN > &a, mat_full< TN > &b, const long i)`
- `template<class TN> void linear_algebra::copy (mat_full< TN > &a, const mat_full< TN > &b)`
- `template<class TN> void linear_algebra::copy (mat_full< TN > &a, mat_full< TN > &b)`
- `template<class TN> void linear_algebra::copy (mat_sparse< TN > &a, mat_sparse< TN > &b, const long i)`
- `template<class TN> void linear_algebra::copy (mat_sparse< TN > &a, const mat_sparse< TN > &b)`
- `template<class TN> void linear_algebra::copy (mat_sparse< TN > &a, mat_sparse< TN > &b)`
- `template<class TN> void linear_algebra::copy (refvector< TN > &a, refvector< TN > &b, const long i)`
- `template<class TN> void linear_algebra::copy (refvector< TN > &a, const refvector< TN > &b)`
- `template<class TN> void linear_algebra::copy (refvector< TN > &a, refvector< TN > &b)`
- `template<class TN> void linear_algebra::copy (sparse_vector< TN > &a, sparse_vector< TN > &b, const long i)`
- `template<class TN> void linear_algebra::copy (sparse_vector< TN > &a, const sparse_vector< TN > &b)`
- `template<class C> void linear_algebra::copy (C &a, const C &b)`
- `template<class TN> void linear_algebra::display (const matrix< TN > &a, ostream &outs)`
- `template<class TN> void linear_algebra::display (const refvector< TN > &a, ostream &outs)`
- `void linear_algebra::display (const double &a, ostream &outs)`

- `template<class TNTN> mat_full< TNTN > linear_algebra::exp (const mat_asym_full< TNTN > &A)`
- `template<class TNTN> mat_full< TNTN > linear_algebra::dexp (mat_asym_full< TNTN > A, const long k, const long l)`

### 5.26.1 Detailed Description

## 5.27 `include/BCR_CPP_LA/class_extensions.hh` File Reference

General expansions for LA.

```
#include <stdexcept>
#include <sstream>
#include <linear_algebra.decl>
#include <class_extensions.decl>
#include <refcount.decl>
#include <matrix.decl>
#include <mat_full.decl>
#include <mat_asym_full.decl>
#include <cmath>
#include <linear_algebra.h>
```

### Namespaces

- namespace `linear_algebra`

### Functions

- `template<class TN> mat_sym_sparse< TN > & linear_algebra::zero (mat_sym_sparse< TN > &r)`
- `template<class TN> mat_sym_full< TN > & linear_algebra::zero (mat_sym_full< TN > &r)`
- `template<class TN> mat_sparse< TN > & linear_algebra::zero (mat_sparse< TN > &r)`
- `template<class TN> mat_full< TN > & linear_algebra::zero (mat_full< TN > &r)`
- `template<class TN> mat_asym_full< TN > & linear_algebra::zero (mat_asym_full< TN > &r)`

- `template<class TN> refvector< TN > & linear_algebra::zero (refvector< TN > &r)`
- `double & linear_algebra::zero (double &r)`
- `template<typename TN> TN & linear_algebra::zero (TN &r)`
- `double & linear_algebra::one (double &r)`
- `template<class TN> matrix< TN > & linear_algebra::one (matrix< TN > &r)`
- `void linear_algebra::copy (double &a, const double &b, const long i)`
- `void linear_algebra::copy (double &a, const double &b)`
- `void linear_algebra::copy (long &a, const long &b, const long i)`
- `void linear_algebra::copy (long &a, const long &b)`
- `void linear_algebra::copy (bool &a, const bool &b)`
- `template<class TN> void linear_algebra::copy (mat_full< TN > &a, const mat_full< TN > &b, const long i)`
- `template<class TN> void linear_algebra::copy (mat_full< TN > &a, const mat_full< TN > &b)`
- `template<class TN> void linear_algebra::copy (mat_full< TN > &a, mat_full< TN > &b)`
- `template<class TN> void linear_algebra::copy (mat_sparse< TN > &a, const mat_sparse< TN > &b, const long i)`
- `template<class TN> void linear_algebra::copy (mat_sparse< TN > &a, const mat_sparse< TN > &b)`
- `template<class TN> void linear_algebra::copy (mat_sparse< TN > &a, mat_sparse< TN > &b)`
- `template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > &a, const mat_sym_sparse< TN > &b, const long i)`
- `template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > &a, const mat_sym_sparse< TN > &b)`
- `template<class TN> void linear_algebra::copy (mat_sym_sparse< TN > &a, mat_sym_sparse< TN > &b)`

- `template<class TN> void linear_algebra::copy (mat_sym_full< TN > &a, const mat_sym_full< TN > &b, const long i)`
- `template<class TN> void linear_algebra::copy (mat_sym_full< TN > &a, const mat_sym_full< TN > &b)`
- `template<class TN> void linear_algebra::copy (mat_sym_full< TN > &a, mat_sym_full< TN > &b)`
- `template<class TN> void linear_algebra::copy (refvector< TN > &a, refvector< TN > &b, const long i)`
- `template<class TN> void linear_algebra::copy (refvector< TN > &a, const refvector< TN > &b, const long i)`
- `template<class TN> void linear_algebra::copy (refvector< TN > &a, const refvector< TN > &b)`
- `template<class TN> void linear_algebra::copy (sparse_vector< TN > &a, sparse_vector< TN > &b, const long i)`
- `template<class TN> void linear_algebra::copy (sparse_vector< TN > &a, const sparse_vector< TN > &b)`
- `template<class C> void linear_algebra::copy (C &a, const C &b)`
- `template<class TN> void linear_algebra::display (const matrix< TN > &a, ostream &outs)`
- `template<class TN> void linear_algebra::display (const refvector< TN > &a, ostream &outs)`
- `void linear_algebra::display (const double &a, ostream &outs)`
- `template<class TNTN> mat_full< TNTN > linear_algebra::exp (const mat_asym_full< TNTN > &A)`
- `template<class TNTN> mat_full< TNTN > linear_algebra::dexp (mat_asym_full< TNTN > A, const long k, const long l)`

### 5.27.1 Detailed Description

## 5.28 `include/BCR_CPP_LA/invert_sym.h` File Reference

Inversion of symmetric matrices.

```
#include <linear_algebra.h>
#include <vector>
#include <iostream>
```

- `refvector< double > & invert_sym (const mat_sym_full< double > &S1, const refvector< double > &v, refvector< double > &r2)`  
*Invert a symmetric full matrix and multiply by a vector.*
- `refvector< double > invert_sym (const mat_sym_full< double > &S1, const refvector< double > &v)`  
*Invert a symmetric full matrix and multiply by a vector.*
- `mat_sym_full< double > invert_sym (mat_sym_full< double > &S)`  
*Invert a symmetric full matrix.*

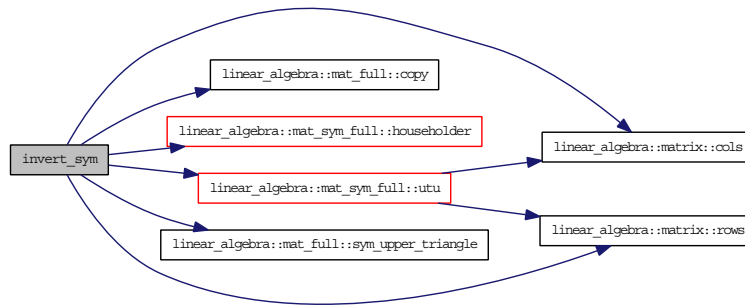
### 5.28.1 Detailed Description

## 5.29 Function Documentation

5.29.2.1 `mat_sym_full<double> invert_sym (mat_sym_full< double > & S)[inline]`

A symmetric matrix can be transformed to a tridiagonal matrix via a householder transformation ( $O(n^2)$ ). A following gaussian elimination of the tridiagonal matrix ( $O(n)$ ) can be used to solve for the inverted matrix.

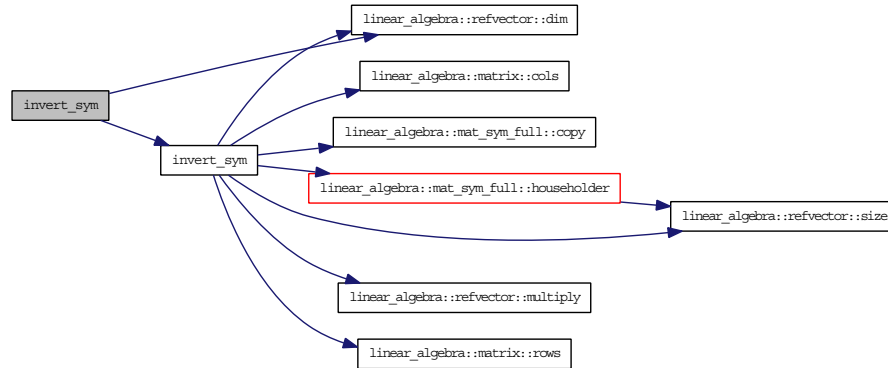
Here is the call graph for this function:





5.29.2.2 `refvector<double> invert_sym (const mat_sym_full< double > & S1, const refvector< double > & v)[inline]`

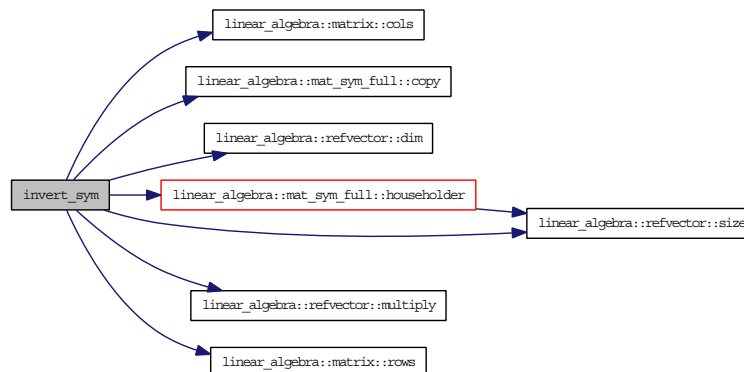
Here is the call graph for this function:



5.29.2.3 `refvector<double>& invert_sym (const mat_sym_full< double > & S1, const refvector< double > & v, refvector< double > & r2)[inline]`

A symmetric matrix can be transformed to a tridiagonal matrix via a householder transformation ( $O(n^2)$ ). A following gaussian elimination of the tridiagonal matrix ( $O(n)$ ). Can be used to solve for the inverted matrix.

Here is the call graph for this function:



## 5.30 include/BCR\_CPP\_LA/linear\_algebra.decl File Reference

Namespace Class Declarations.

`#include <refcount.decl>`

```

#include <matrix.decl>
#include <square.decl>
#include <symmetric.decl>
#include <asymmetric.decl>
#include <sparse_vector.decl>
#include <mat_sparse.decl>
#include <mat_full.decl>
#include <mat_sym_full.decl>
#include <mat_sym_sparse.decl>
#include <mat_asym_sparse.decl>
#include <mat_asym_full.decl>
#include <class_extensions.decl>

```

## Namespaces

- namespace `linear_algebra`

### 5.30.1 Detailed Description

#### 5.31 `include/BCR_CPP_LA/linear_algebra.h` File Reference

```

#include <linear_algebra.decl>
#include <refcount.h>
#include <matrix.h>
#include <square.h>
#include <symmetric.h>
#include <asymmetric.h>
#include <sparse_vector.h>
#include <mat_sparse.h>
#include <mat_sym_full.h>
#include <mat_full.h>
#include <mat_sym_sparse.h>
#include <mat_asym_sparse.h>
#include <mat_asym_full.h>
#include <class_extensions.hh>

```

#### 5.32 `include/BCR_CPP_LA/mat_asym_full.decl` File Reference

Provide declarations for antisymmetric full matrices.

```
#include <vector>
#include <linear_algebra.decl>
#include <asymmetric.decl>
```

## Namespaces

- namespace `linear_algebra`

## Classes

- class `linear_algebra::mat_asym_full< TN >`  
    `mat_asym_full` uses packing for full antisymmetric matrices

### 5.32.1 Detailed Description

## Todo

throw exceptions.

### 5.33 `include/BCR_CPP_LA/mat_asym_full.h` File Reference

Provide manipulations for antisymmetric full matrices.

```
#include <vector>
#include <linear_algebra.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_full.decl>
#include <mat_sym_full.decl>
#include <mat_sym_sparse.decl>
#include <mat_asym_full.decl>
#include <cmath>
#include <refcount.h>
#include <sparse_vector.h>
#include <mat_full.h>
#include <mat_sym_full.h>
#include <mat_sym_sparse.h>
```

## Namespaces

- namespace `linear_algebra`

### 5.33.1 Detailed Description

#### Todo

throw exceptions.

### 5.34 `include/BCR_CPP_LA/mat_asym_sparse.decl` File Reference

```
#include <vector>
#include <linear_algebra.decl>
#include <asymmetric.decl>
```

#### Namespaces

- namespace `linear_algebra`

#### Classes

- class `linear_algebra::mat_asym_sparse< TN >`  
*mat\_asym\_sparse packs sparse column matrices into a comfortable format.*

### 5.35 `include/BCR_CPP_LA/mat_asym_sparse.h` File Reference

```
#include <vector>
#include <linear_algebra.decl>
#include <mat_asym_sparse.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_full.decl>
#include <mat_sparse.decl>
#include <mat_sym_full.decl>
#include <mat_asym_full.decl>
#include <refcount.h>
#include <sparse_vector.h>
#include <mat_full.h>
#include <mat_sparse.h>
#include <mat_sym_full.h>
#include <mat_asym_full.h>
```

#### Namespaces

- namespace `linear_algebra`

### 5.36 `include/BCR_CPP_LA/mat_full.decl` File Reference

Provide declarations for full matrix and vector interactions.

```
#include <vector>
#include <iostream>
#include <stdexcept>
#include <linear_algebra.decl>
#include <matrix.decl>
```

#### Namespaces

- namespace `linear_algebra`

#### Classes

- class `linear_algebra::mat_full< TN >`  
*Class of full column matrices.*

#### 5.36.1 Detailed Description

### 5.37 `include/BCR_CPP_LA/mat_full.h` File Reference

```
#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <matrix.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_sym_full.decl>
#include <mat_asym_full.decl>
#include <mat_sym_sparse.decl>
#include <refcount.h>
#include <matrix.h>
#include <sparse_vector.h>
#include <mat_sym_full.h>
#include <mat_asym_full.h>
#include <mat_sym_sparse.h>
```

## Namespaces

- namespace `linear_algebra`

### 5.38 `mat_full.h` File Reference

```
#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <matrix.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_sym_full.decl>
#include <mat_asym_full.decl>
#include <mat_sym_sparse.decl>
#include <refcount.h>
#include <matrix.h>
#include <sparse_vector.h>
#include <mat_sym_full.h>
#include <mat_asym_full.h>
#include <mat_sym_sparse.h>
```

## Namespaces

- namespace `linear_algebra`

### 5.39 `include/BCR_CPP_LA/mat_sparse.decl` File Reference

Provide declarations for sparse matrix and vector interactions.

```
#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <matrix.decl>
#include <refcount.decl>
```

## Namespaces

- namespace `linear_algebra`

## Classes

- class linear\_algebra::mat\_sparse< TN >  
*Class of sparse column matrices.*

### 5.39.1 Detailed Description

## 5.40 include/BCR\_CPP\_LA/mat\_sparse.h File Reference

Provide functionality for sparse matrix and vector interactions.

```
#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <matrix.decl>
#include <mat_sparse.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_full.decl>
#include <mat_sym_sparse.decl>
#include <refcount.h>
#include <sparse_vector.h>
#include <mat_full.h>
#include <mat_sym_sparse.h>
```

## Namespaces

- namespace linear\_algebra

### 5.40.1 Detailed Description

## 5.41 include/BCR\_CPP\_LA/mat\_sym\_full.decl File Reference

Provide declarations for symmetric full matrices.

```
#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <symmetric.decl>
```

## Namespaces

- namespace `linear_algebra`

## Classes

- class `linear_algebra::mat_sym_full< TN >`  
`mat_sym_full` uses packing for full symmetric matrices

### 5.41.1 Detailed Description

#### Todo

throw exceptions.

## 5.42 `include/BCR_CPP_LA/mat_sym_full.h` File Reference

Provide manipulations for symmetric full matrices.

```
#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_full.decl>
#include <mat_asym_full.decl>
#include <mat_sym_full.decl>
#include <refcount.h>
#include <sparse_vector.h>
#include <mat_full.h>
#include <mat_asym_full.h>
```

## Namespaces

- namespace `linear_algebra`

### 5.42.1 Detailed Description

#### Todo

throw exceptions.



## 5.43 include/BCR\_CPP\_LA/mat\_sym\_sparse.decl File Reference

Provide declarations for antisymmetric sparse matrices.

```
#include <vector>
#include <linear_algebra.decl>
#include <symmetric.decl>
#include "mat_sym_sparse_eigen.decl"
```

### Namespaces

- namespace linear\_algebra

### Classes

- class linear\_algebra::mat\_sym\_sparse< TN >  
*mat\_sym\_sparse packs sparse column matrices into a comfortable format.*

### 5.43.1 Detailed Description

## 5.44 include/BCR\_CPP\_LA/mat\_sym\_sparse.h File Reference

Provide functionality for symmetric sparse matrices.

```
#include <vector>
#include <linear_algebra.decl>
#include <mat_sym_sparse.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <mat_full.decl>
#include <mat_sym_full.decl>
#include <mat_asym_full.decl>
#include "mat_sym_sparse_eigen.h"
#include <refcount.h>
#include <sparse_vector.h>
#include <mat_full.h>
#include <mat_sym_full.h>
#include <mat_asym_full.h>
```

### Namespaces

- namespace `linear_algebra`

#### 5.44.1 Detailed Description

### 5.45 `include/BCR_CPP_LA/mat_sym_sparse_eigen.decl` File Reference

Eigenvalue related declarations.

#### Functions

- `void gen_mat (const sparse_vector< TN > &a, const sparse_vector< TN > &b, TN correction)`  
Produce the necessary  $ij^* + ji^*$ .
- `mat_sparse< TN > householder (vector< TN > &diagonal, vector< TN > &subdiagonal)`  
*This routine produces a the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse symmetric matrix.*
- `mat_sparse< TN > householder (refvector< TN > &diagonal, refvector< TN > &subdiagonal)`  
*This routine produces a the diagonal, sub/superdiagonal and transformation matrix for the Householder transformation of a sparse ymmetric matrix.*

#### 5.45.1 Detailed Description

##### Todo

throw exceptions.

### 5.46 `include/BCR_CPP_LA/mat_sym_sparse_eigen.h` File Reference

Eigenvalue related routines.

#### 5.46.1 Detailed Description

##### Todo

throw exceptions.

## 5.47 `include/BCR_CPP_LA/matrix.decl` File Reference

Declares fundamentals of matrices.

```
#include <vector>
#include <iostream>
#include <linear_algebra.decl>
```

### Namespaces

- namespace `linear_algebra`

### Classes

- class `linear_algebra::matrix< TN >`  
*matrix is a basic class which covers all kinds of matrices.*

### Functions

- `template<class TN> TN linear_algebra::operator * (const std::vector< TN > &a, const std::vector< TN > &b)`  
*Scalar product of two full vectors.*

#### 5.47.1 Detailed Description

## 5.48 `include/BCR_CPP_LA/matrix.h` File Reference

Describes fundamentals of matrices.

```
#include <vector>
#include <iostream>
#include <linear_algebra.decl>
#include <matrix.decl>
```

### Namespaces

- namespace `linear_algebra`

## Functions

- `template<class TN> TN linear_algebra::operator * (const vector< TN > &a, const vector< TN > &b)`

*Scalar product of two full vectors.*

### 5.48.1 Detailed Description

## 5.49 `include/BCR_CPP_LA/polynomial.decl` File Reference

Class declarations for polynomial.

```
#include <linear_algebra.decl>
#include <sparse_vector_infty.decl>
```

## Namespaces

- namespace `linear_algebra`

## Classes

- class `linear_algebra::polynomial< C >`  
*A class for polynomials.*

### 5.49.1 Detailed Description

## 5.50 `include/BCR_CPP_LA/polynomial.h` File Reference

Implementation for a class of polynomials.

```
#include <sparse_vector_infty.h>
#include <polynomial.decl>
#include <linear_algebra.h>
#include <class_extensions.h>
```

## Namespaces

- namespace `linear_algebra`

### 5.50.1 Detailed Description

## 5.51 include/BCR\_CPP\_LA/refcount.decl File Reference

Contains the declarations to a reference counted vector class as well as general reference counting.

```
#include <vector>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <linear_algebra.decl>
```

### Namespaces

- namespace linear\_algebra

### Classes

- class linear\_algebra::refcount< Type >  
*Class refcount provides reference counted pointers. Most basic use.*
- class linear\_algebra::refvector< TN >  
*Class refvector provides reference counted vectors.*

### Functions

- template<class TN> void std::operator>>(const vector< TN > &a, ostream &OUT)

### 5.51.1 Detailed Description

## 5.52 include/BCR\_CPP\_LA/refcount.h File Reference

Contains a reference counted vector class as well as general reference counting.

```
#include <vector>
#include <stdexcept>
#include <sstream>
#include <iostream>
```

```
#include <linear_algebra.decl>
#include <refcount.decl>
#include <sparse_vector.decl>
#include <class_extensions.hh>
#include <cstdlib>
#include <sparse_vector.h>
```

## Namespaces

- namespace `linear_algebra`

## Functions

- `template<class TN> void std::operator>> (const vector< TN > &a, ostream &OUT)`
- `template<typename TN> void std::operator>> (const TN &a, ostream &OUT)`

### 5.52.1 Detailed Description

## 5.53 `include/BCR_CPP_LA/sparse_vector.decl` File Reference

Provide declarations for sparse vector interactions.

```
#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <refcount.decl> %
#include <matrix.decl>
```

## Namespaces

- namespace `linear_algebra`

## Classes

- class `linear_algebra::sparse_vector< TN >`  
*A class for sparse vectors.*

## Functions

- `template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, std::vector< TN > b)`  
*Scalar product of a sparse\_vector with a full vector.*
- `template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, refvector< TN > b)`  
*Scalar product of a sparse\_vector with a full refvector.*

### 5.53.1 Detailed Description

## 5.54 include/BCR\_CPP\_LA/sparse\_vector.h File Reference

Provide functionality for sparse vectors.

```
#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <sparse_ector.decl>
#include <refcount.decl>
#include <cstdlib>
#include <refcount.h>
```

## Namespaces

- namespace `linear_algebra`

## Functions

- `template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, vector< TN > b)`  
*Scalar product of a sparse\_vector with a full vector.*
- `template<class TN> TN linear_algebra::operator * (sparse_vector< TN > a, refvector< TN > b)`  
*Scalar product of a sparse\_vector with a full refvector.*

### 5.54.1 Detailed Description

## 5.55 `include/BCR_CPP_LA/sparse_vector_infty.decl` File Reference

Provide declarations for sparse vector interactions of infinite dimensions.

```
#include <vector>
#include <stdexcept>
#include <linear_algebra.decl>
#include <refcount.decl>
```

### Namespaces

- namespace `linear_algebra`

### Classes

- class `linear_algebra::sparse_vector_infty< TN >`  
*A class for sparse vectors.*

### 5.55.1 Detailed Description

## 5.56 `include/BCR_CPP_LA/sparse_vector_infty.h` File Reference

Provide functionality for sparse vectors of infinite dimension.

```
#include <vector>
#include <stdexcept>
#include <sparse_vector_infty.decl>
#include <refcount.h>
#include <class_extensions.h>
```

### Namespaces

- namespace `linear_algebra`

### 5.56.1 Detailed Description

## 5.57 `include/BCR_CPP_LA/square.decl` File Reference

Declares fundamentals of square matrices.



```
#include <linear_algebra.decl>
#include <matrix.decl>
```

## Namespaces

- namespace linear\_algebra

## Classes

- class linear\_algebra::square< TN >  
*The square matrix class contains all n by n matrices.*

### 5.57.1 Detailed Description

### 5.58 include/BCR\_CPP\_LA/square.h File Reference

Describes fundamentals of matrices.

```
#include <vector>
#include <iostream>
#include <square.decl>
#include <matrix.h>
```

## Namespaces

- namespace linear\_algebra

### 5.58.1 Detailed Description

### 5.59 include/BCR\_CPP\_LA/symmetric.decl File Reference

Declares fundamentals of matrices.

```
#include <linear_algebra.decl>
#include <square.decl>
```

## Namespaces

- namespace linear\_algebra

## Classes

- class `linear_algebra::symmetric< TN >`  
*The symmetric class contains all symmetric matrices.*

### 5.59.1 Detailed Description

## 5.60 `include/BCR_CPP_LA/symmetric.h` File Reference

Describes fundamentals of symmetric matrices.

```
#include <vector>
#include <iostream>
#include <square.h>
#include <symmetric.decl>
```

## Namespaces

- namespace `linear_algebra`

### 5.60.1 Detailed Description

## 5.61 `include/BCR_CPP_LA/trace_AB.hh` File Reference

get the trace of matrix products.

```
#include <linear_algebra.h>
```

## Namespaces

- namespace `linear_algebra`

## Functions

- template<class TN> TN `linear_algebra::trace_AB` (const `mat_sym_full< TN >` &A, const `mat_sym_full< TN >` &B)
- template<class TN> TN `linear_algebra::trace_AB` (const `mat_sym_full< TN >` &A, const `mat_sym_sparse< TN >` &B)
- template<class T> T `linear_algebra::trace_AB` (const `mat_asym_full< T >` &A, const `mat_asym_full< T >` &B)

### 5.61.1 Detailed Description

## 5.62 include/sorting\_functions.hh File Reference

```
#include <BCR_CPP_LA/refcount.h>
```

```
#include <sorting_functions.hh>
```

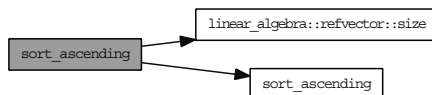
### Functions

- `template<class T1> static refvector< unsigned long > & sort_ascending (const refvector< T1 > &E, unsigned long start, unsigned long end, refvector< unsigned long > &index)`  
*Do a qsort on E with immediate storage back into index. ensure index has the right dimension. (perfectly parallelizable).*
- `template<class T1> static refvector< unsigned long > sort_ascending (const refvector< T1 > &E)`

### 5.62.1 Function Documentation

5.62.1.1 `template<class T1> static refvector<unsigned long> sort_ascending (const refvector< T1 > & E)[static]`

Here is the call graph for this function:



5.62.1.2 `template<class T1> static refvector<unsigned long>& sort_ascending (const refvector< T1 > & E, unsigned long start, unsigned long end, refvector< unsigned long > & index)[static]`

## 5.63 include/stateful\_thread\_group.hh File Reference

```
#include <boost/thread/exceptions.hpp>
```

```
#include <ostream>
```

```
#include <boost/thread/detail/move.hpp>
```

```
#include <boost/thread/mutex.hpp>
```

```

#include <boost/thread/xtime.hpp>
#include <boost/thread/detail/thread_heap_alloc.hpp>
#include <boost/utility.hpp>
#include <boost/assert.hpp>
#include <list>
#include <algorithm>
#include <boost/ref.hpp>
#include <boost/cstdint.hpp>
#include <boost/bind.hpp>
#include <stdlib.h>
#include <memory>
#include <boost/utility/enable_if.hpp>
#include <boost/type_traits/remove_reference.hpp>
#include <boost/thread.hpp>
#include <boost/config/abi_prefix.hpp>
#include <boost/config/abi_suffix.hpp>

```

## Namespaces

- namespace boost

## Classes

- class boost::stateful\_thread\_group

## 5.64 stateful\_thread\_group.hh File Reference

```

#include <boost/thread/exceptions.hpp>
#include <ostream>
#include <boost/thread/detail/move.hpp>
#include <boost/thread/mutex.hpp>
#include <boost/thread/xtime.hpp>
#include <boost/thread/detail/thread_heap_alloc.hpp>
#include <boost/utility.hpp>
#include <boost/assert.hpp>
#include <list>
#include <algorithm>
#include <boost/ref.hpp>
#include <boost/cstdint.hpp>

```

```

#include <boost/bind.hpp>
#include <stdlib.h>
#include <memory>
#include <boost/utility/enable_if.hpp>
#include <boost/type_traits/remove_reference.hpp>
#include <boost/thread.hpp>
#include <boost/config/abi_prefix.hpp>
#include <boost/config/abi_suffix.hpp>

```

## Namespaces

- namespace boost

## Classes

- class boost::stateful\_thread\_group

## 5.65 isthreadableabstract.h File Reference

implementation of is\_threadable\_abstract

```

#include <boost/thread/shared_mutex.hpp>
#include <stateful_thread_group.hh>

```

## Classes

- class is\_threadable\_abstract  
*Abstract class implementing the Concepts::is\_threadable concept.*

### 5.65.1 Detailed Description

## 5.66 Library\_data.cc File Reference

Implementations of the Library class for discrete spaces.

```

#include <BCR_CPP_LA/linear_algebra.h>
#include <Library_data.hh>
#include <boost/thread.hpp>
#include <boost/mem_fn.hpp>

```

### 5.66.1 Detailed Description

## 5.67 Library\_data.hh File Reference

```
#include <BCR_CPP_LA/refcount.decl>
#include <typedefs.hh>
#include <string>
```

### Classes

- class Library\_data  
*Abstract class for discrete optimizations.*

## 5.68 noprune.h File Reference

Header of class noprune.

```
#include <concepts.hh>
#include <prunerabstract.h>
```

### Classes

- class noprune< X >  
*Dummy pruning class. No actual pruning done.*

### 5.68.1 Detailed Description

## 5.69 optimizeabstract.h File Reference

implement abstract class optimize\_abstract

### Classes

- class optimize\_abstract  
*Abstract class that implements the Concepts::has\_optimize concept. Only good for generic referncing of optimizations.*

### 5.69.1 Detailed Description

## 5.70 prunerabstract.h File Reference

pruner\_abstract implementation

```
#include <BCR_CPP_LA/refcount.h>
```

```
#include <concepts.hh>
```

### Classes

- class pruner\_abstract< X >  
*Abstract class implementing the Concepts::pruner concept.*

### 5.70.1 Detailed Description

## 5.71 reordergeneralbase.hh File Reference

```
#include <prunerabstract.h>
```

```
#include <typedefs.hh>
```

```
#include <BCR_CPP_LA/refcount.h>
```

```
#include <sorting_functions.hh>
```

### Classes

- class reorder\_general\_base< X >

## 5.72 simpleprune.cc File Reference

Implementation file of simple\_prune.

```
#include <BCR_CPP_LA/refcount.h>
```

```
#include <simpleprune.h>
```

```
#include <sorting_functions.hh>
```

### 5.72.1 Detailed Description

## 5.73 simpleprune.h File Reference

simple pruning class.

```
#include <typedefs.hh>
#include <boost/concept_check.hpp>
#include <concepts.hh>
#include <BCR_CPP_LA/refcount.decl>
#include <prunerabstract.h>
```

## Classes

- class simple\_prune< X >  
*Class to prune a Library.*

### 5.73.1 Detailed Description

## 5.74 typedefs.hh File Reference

```
#include <cmath>
#include <limits>
```

## Classes

- struct valerg  
*Pair of property value and energy.*
- struct double\_index  
*Pair of configurational and conformational index, respectively.*

## Typedefs

- typedef unsigned long ulong

## Functions

- bool operator== (const double\_index &a, const double\_index &b)  
*Comparison operator.*
- bool operator== (const valerg &a, const valerg &b)  
*Comparison operator. item valerg operator- (const valerg &a, const valerg &b)  
Subtract two valerg.*



- valerg operator-= (valerg &a, const valerg &b)  
*Subtract two valerg.*
- valerg operator+= (valerg &a, const valerg &b)  
*Subtract two valerg.*
- valerg operator+ (const valerg &a, const valerg &b)  
*Add two valerg.*
- valerg operator \* (const valerg &a, const double b)  
*Multiply.*
- valerg & operator \*= (valerg &r, const double b  
)  
*Multiply.*

### 5.74.1 Typedef Documentation

5.74.1.1 typedef unsigned long ulong

### 5.74.2 Function Documentation

5.74.2.1 valerg operator \* (const valerg & a, const double b)[inline]

5.74.2.2 valerg& operator \*= (valerg & r, const double b)[inline]

5.74.2.3 valerg operator+ (const valerg & a, const valerg & b)[inline]

5.74.2.4 valerg operator+= (valerg & a, const valerg & b)[inline]

5.74.2.5 valerg operator- (const valerg & a, const valerg & b)[inline]

7.74.2.6 valerg operator-= (valerg & a, const valerg & b)[inline]

5.74.2.7 bool operator== (const valerg & a, const valerg & b)[inline]

5.74.2.8 bool operator== (const double\_index & a, const double\_index & b)[inline]

Define zmat and zmat\_entry.

```
#include <sstream>
#include <BCR_CPP_LA/linear_algebra.h>
#include <zmat.hh>
```

### 5.74.3 Detailed Description

## 5.75 zmat.hh File Reference

Define zmat and zmat\_entry.

```
#include <sstream>
#include <BCR_CPP_LA/linear_algebra.h>
```

### Classes

- class zmat\_entry  
*Class of Z-matrix entries.*
- class zmat\_connector  
*Class of connectors between z-matrices.*
- class zmat  
*Class for Z-matrices. Just a wrapper class around a vector of entries.*

### 5.75.1 Detailed Description

## 5.76 zmat\_opt.cc File Reference

Z-matrix optimization class implementation.

```
#include <typedefs.hh>
#include <zmat.hh>
#include <zmat_opt.hh>
#include <iostream>
#include <fstream>
```

### Functions

- valerg calc\_property (const zmat &A, const string &out, const string &id, zmat &returnA)  
*Setup the external computations and execute them.*
- static valerg calc\_energy (const zmat &A, const string &out, const string &id, zmat &returnA)  
*Setup the external computations for the energy and execute them.*

### 5.76.1 Detailed Description

### 5.76.2 Function Documentation

5.76.2.1 static valerg calc\_energy (const zmat & *A*, const string & *out*, const string & *id*, zmat & *returnA*)[static]

requires

```
./energy_script
```

5.76.2.2 valerg calc\_property (const zmat & *A*, const string & *out*, const string & *id*, zmat & *eturnA*)

## 5.77 zmat\_opt.hh File Reference

Z-matrix optimization class definition.

```
#include <typedefs.hh>
#include <zmat.hh>
#include <Library_data.hh>
#include <noprune.h>
#include <isthreadableabstract.h>
```

### Classes

- class zmat\_opt

### 5.77.1 Detailed Description

---

## 6. Discrete Optimization in Chemical Space Example Documentation

---

### 6.1 carbazoles.inp

```
1 ChemGroup(
2   (
3     Z(
```

```

4      (c, -3, 0.7, -2, 0, -1, 0)
5      (c, 0, 1.4, -3, 126, -2, 0)
6      (c, 1, 1.4, 0, 108, -3, 180)
7      (c, 2, 1.4, 1, 108, 0, 0)
8  )
9  ReturnConnector()
10 Connector(
11  (
12      (0,1,2)
13      (0.7,0,0)
14      (0,120,0)
15      (0,0,180)
16
17      (0,0,1)
18      (0,0,1)
19      (0,0,0)
20      ()
21  )
22  (
23      (-3,-2,-1)
24      (0,0,0)
25      (0,0,0)
26      (0,0,0)
27
28      (0,0,0)
29      (0,0,0)
30      (0,0,0)
31      ()
32  )
33  (
34      (-3,-2,-1)
35      (0,0,0)
36      (0,0,0)
37      (0,0,0)
38
39      (0,0,0)
40      (0,0,0)
41      (0,0,0)
42      ()
43  )

```

```

44      (
45      (-3,-2,-1)
46      (0,0,0)
47      (0,0,0)
48      (0,0,0)
49
50      (0,0,0)
51      (0,0,0)
52      (0,0,0)
53      ()
54      )
55      (
56      (-3,-2,-1)
57      (0,0,0)
58      (0,0,0)
59      (0,0,0)
60
61      (0,0,0)
62      (0,0,0)
63      (0,0,0)
64      ()
65      )
66      (
67      (3,2,1)
68      (0.7,0,0)
69      (0,120,0)
70      (0,0,180)
71
72      (0,0,1)
73      (0,0,1)
74      (0,0,0)
75      ()
76      )
77      (
78      (-3,-2,-1)
79      (0,0,0)
80      (0,0,0)
81      (0,0,0)
82
83      (0,0,0)

```

```

84      (0,0,0)
85      (0,0,0)
86      ()
87      )
88      (
89      (-3,-2,-1)
90      (0,0,0)
91      (0,0,0)
92      (0,0,0)
93
94      (0,0,0)
95      (0,0,0)
96      (0,0,0)
97      ()
98      )
99      (
100     (-3,-2,-1)
101     (0,0,0)
102     (0,0,0)
103     (0,0,0)
104
105     (0,0,0)
106     (0,0,0)
107     (0,0,0)
108     ()
109     )
110     (
111     (-3,-2,-1)
112     (0,0,0)
113     (0,0,0)
114     (0,0,0)
115
116     (0,0,0)
117     (0,0,0)
118     (0,0,0)
119     ()
120     )
121     (
122     (0,1,2)
123     (1.4,0,0)

```

```

124      (0, 108, 0)
125      (0, 0, 0)
126
127      (0,0,1)
128      (0,0,1)
129      (0,0,0)
130      ()
131    )
132  )
133  allowed_groups(
134    (7,2,3,4,5,6,8,9,10,11,12)
135    (7,2,3,4,5,6,8,9,10,11,12)
136    (7,2,3,4,5,6,8,9,10,11,12)
137    (7,2,3,4,5,6,8,9,10,11,12)
138    (1)
139    (7,2,3,4,5,6,8,9,10,11,12)
140    (7,2,3,4,5,6,8,9,10,11,12)
141    (7,2,3,4,5,6,8,9,10,11,12)
142    (7,2,3,4,5,6,8,9,10,11,12)
143    (1)
144    (13,14,15)
145  )
146 )
147 (
148   Z()
149   ReturnConnector(
150     (
151       (-6,-5,-4)
152       (0,0, 0)
153       (0,0, 0)
154       (0,0, 0)
155
156       (0,0,0)
157       (0,0,0)
158       (0,0,0)
159     )
160   )
161 )
162 Connector()
163 allowed_groups()

```

```

164 )
165 (
166   Z(
167     (C, -3, 0.7, -2, 0, -1, 0)
168     (C, 0, 1.4, -3, 120, -2, 180)
169     (N, 0, 2.7, -3, 120, -2, 180)
170   )
171   ReturnConnector(
172     (
173       (0,1,-3)
174       (0.7, 0, 0)
175       (0, 120, 180)
176       (0,0, 180)
177
178       (0,0,1)
179       (0,0,1)
180       (0,0,0)
181     )
182   )
183 )
184 Connector()
185 allowed_groups()
186 )
187 (
188   Z(
189     (C, -3, 0.7, -2, 0, -1, 0)
190     (C, 0, 1.4, -3, 120, -2, 180)
191     (O, 1, 1.4, 0, 120, -3, -91)
192     (H, 1, 1.1, 2, 120, 0, 180)
193   )
194   ReturnConnector(
195     (
196       (0,1,-3)
197       (0.7, 0, 0)
198       (0, 120, 180)
199       (0,0, 180)
200
201       (0,0,1)
202       (0,0,0)
203       (0,0,0)

```



```

204     ()
205     )
206     )
207     Connector()
208     allowed_groups()
209     )
210     (
211     Z(
212     (C, -3, 0.7, -2, 0, -1, 0)
213     (0, 0, 1.4, -3, 120, -2, 180)
214     (C, 1, 1.4, 0, 120, -3, -91(180))
215     (H, 2, 1.1, 1, 109.47, 0, 180)
216     (H, 2, 1.1, 1, 109.47, 3, 120)
217     (H, 2, 1.1, 1, 109.47, 4, 120)
218     )
219     ReturnConnector(
220     (
221     (0,1,-3)
222     (0.7, 0, 0)
223     (0, 120, 180)
224     (0,0, 180)
225
226     (0,0,1)
227     (0,0,0)
228     (0,0,0)
229     ()
230     )
231     )
232     Connector()
233     allowed_groups()
234     )
235     (
236     Z(
237     (C, -3, 0.7, -2, 0, -1, 0)
238     (C, 0, 1.4, -3, 120, -2, 180)
239     (0, 1, 1.3, 0, 120, -3, -91(180))
240     (C, 1, 1.5, 2, 120, 0, 180)
241     (H, 3, 1.1, 1, 109.47, 2, 180)
242     (H, 3, 1.1, 1, 109.47, 4, 120)
243     (H, 3, 1.1, 1, 109.47, 5, 120)

```

```

244 )
245 ReturnConnector(
246 (
247 (0,1,-3)
248 (0.7, 0, 0)
249 (0, 120, 180)
250 (0,0, 180)
251
252 (0,0,1)
253 (0,0,0)
254 (0,0,0)
255 ()
256 )
257 )
258 Connector()
259 allowed_groups()
260 )
261 (
262 Z(
263 (C, -3, 0.7, -2, 0, -1, 0)
264 (N, 0, 1.4, -3, 120, -2, 180)
265 (C, 1, 1.4, 0, 120, -3, -91(180))
266 (O, 2, 1.3, 1, 120, 0, 0(180))
267 (C, 2, 1.5, 1, 120, 3, 180)
268 (H, 4, 1.1, 2, 109.47, 1, 180)
269 (H, 4, 1.1, 2, 109.47, 5, 120)
270 (H, 4, 1.1, 2, 109.47, 6, 120)
271
272 (H, 1, 1.1, 2, 120, 0, 180)
273 )
274 ReturnConnector(
275 (
276 (0,1,-3)
277 (0.7, 0, 0)
278 (0, 120, 180)
279 (0,0, 180)
280
281 (0,0,1)
282 (0,0,0)
283 (0,0,0)

```

```

284     ()
285     )
286     )
287     Connector()
288     allowed_groups()
289     )
290     (
291     Z(
292     (C,  -3, 0.7, -2, 0,   -1, 0)
293     (H,   0, 1.1, -3, 120, -2, 180)
294     )
295     ReturnConnector(
296     (
297     (0,1,-3)
298     (0.7, 0, 0)
299     (0, 120, 180)
300     (0,0, 180)
301
302     (0,0,1)
303     (0,0,0)
304     (0,0,0)
305     ()
306     )
307     )
308     Connector()
309     allowed_groups()
310     )
311     (
312     Z(
313     (C, -3, 0.7, -2,   0, -1,   0)
314     (N,   0, 1.4, -3, 120,   -2, 180)
315     (0,   1, 1.2,   0, 120, -3, -80)
316     (0,   1, 1.2,   2, 120,   0, 180)
317     )
318     ReturnConnector(
319     (
320     (0,1,-3)
321     (0.7, 0, 0)
322     (0, 120, 180)
323     (0,0, 180)

```

```

324
325     (0,0,1)
326     (0,0,0)
327     (0,0,0)
328     ()
329 )
330 )
331 Connector()
332 allowed_groups()
333 )
334 (
335   Z(
336     (C, -3, 0.7, -2, 0, -1, 0)
337     (0, 0, 1.3, -3, 120, -2, 180)
338     (c, 1, 1.4, 0, 120, -3, 0(180))
339     (o, 2, 1.3, 1, 120, 0, 0(180))
340     (c, 2, 1.5, 1, 120, 3, 180)
341     (H, 4, 1.1, 2, 109.47, 1, 180)
342     (h, 4, 1.1, 2, 109.47, 5, 120)
343     (h, 4, 1.1, 2, 109.47, 6, 120)
344   )
345   ReturnConnector(
346     (
347       (0,1,-3)
348       (0.7, 0, 0)
349       (0, 120, 180)
350       (0,0, 180)
351
352       (0,0,1)
353       (0,0,0)
354       (0,0,0)
355       ()
356     )
357   )
358   Connector()
359   allowed_groups()
360 )
361 (
362   Z(
363     (C, -3, 0.7, -2, 0, -1, 0)

```

```

364     (0, 0, 1.4, -3, 120, -2, 180)
365     (H, 1, 1.1, 0, 120, -3, -91(180))
366 )
367 ReturnConnector(
368     (
369         (0,1,-3)
370         (0.7, 0, 0)
371         (0, 120, 180)
372         (0,0, 180)
373
374         (0,0,1)
375         (0,0,0)
376         (0,0,0)
377         ()
378     )
379 )
380 Connector()
381 allowed_groups()
382 )
383 (
384     Z(
385         (C, -3, 0.7, -2, 0, -1, 0)
386         (N, 0, 1.4, -3, 120, -2, 180)
387         (H, 1, 1.1, 0, 120, -3, 0)
388         (H, 1, 1.1, 2, 109.47, 0, 120(-240))
389     )
390 ReturnConnector(
391     (
392         (0,1,-3)
393         (0.7, 0, 0)
394         (0, 120, 180)
395         (0,0, 180)
396
397         (0,0,1)
398         (0,0,0)
399         (0,0,0)
400         ()
401     )
402 )
403 Connector()

```

```

404     allowed_groups()
405 )
406 (
407     Z(
408         (C, -3, 0.7, -2, 0, -1, 0)
409         (N, 0, 1.4, -3, 120, -2, 180)
410         (C, 1, 1.4, 0, 120, -3, 60)
411         (C, 1, 1.4, 2, 120, 0, 120(-240))
412         (H, 2, 1.1, 1, 109.47, 3, 180)
413         (H, 2, 1.1, 1, 109.47, 4, 120)
414         (H, 2, 1.1, 1, 109.47, 5, 120)
415         (H, 3, 1.1, 1, 109.47, 2, 80)
416         (H, 3, 1.1, 1, 109.47, 7, 120)
417         (H, 3, 1.1, 1, 109.47, 8, 120)
418     )
419     ReturnConnector(
420         (
421             (0,1,-3)
422             (0.7, 0, 0)
423             (0, 120, 180)
424             (0,0, 180)
425
426             (0,0,1)
427             (0,0,0)
428             (0,0,0)
429         )
430     )
431 )
432     Connector()
433     allowed_groups()
434 )
435 (
436     Z(
437         (o, -3, 0, -2,0,-1,0)
438     )
439     ReturnConnector()
440     Connector()
441     allowed_groups()
442 )
443 (

```

```

444      Z(
445      (s, -3, 0, -2,0,-1,0)
446      )
447      ReturnConnector()
448      Connector()
449      allowed_groups()
450  )
451  (
452      Z(
453      (c, -3, 0, -2, 0,-1,0)
454      (o, 0, 1.3, -3, 126, -2, 180)
455      )
456      ReturnConnector()
457      Connector()
458      allowed_groups()
459  )
460 )

```

## 6.2 ChemGroup

Default construction. ChemIdent("H")

## 6.3 vanilla-rings.inp

```

1 ChemGroup(
2   (
3     Z()
4     ReturnConnector()
5     Connector(
6       (
7         (-3,-2,-1)
8         (0,0,0)
9         (0,0,0)
10        (0,0,0)
11
12        (0,0,0)
13        (0,0,0)
14        (0,0,0)
15        ()
16      )

```

```

17      (
18      (-3,-2,-1)
19      (0,0,0)
20      (0,0,0)
21      (0,0,0)
22
23      (0,0,0)
24      (0,0,0)
25      (0,0,0)
26      ()
27      )
28      (
29      (-3,-2,-1)
30      (0,0,0)
31      (0,0,0)
32      (0,0,0)
33
34      (0,0,0)
35      (0,0,0)
36      (0,0,0)
37      ()
38      )
39      (
40      (-3,-2,-1)
41      (0,0,0)
42      (0,0,0)
43      (0,0,0)
44
45      (0,0,0)
46      (0,0,0)
47      (0,0,0)
48      ()
49      )
50      (
51      (-3,-2,-1)
52      (0,0,0)
53      (0,0,0)
54      (0,0,0)
55
56      (0,0,0)

```



```

57      (0,0,0)
58      (0,0,0)
59      ()
60      )
61      (
62      (-3,-2,-1)
63      (0,0,0)
64      (0,0,0)
65      (0,0,0)
66
67      (0,0,0)
68      (0,0,0)
69      (0,0,0)
70      ()
71      )
72      (
73      (-3,-2,-1)
74      (0,0,0)
75      (0,0,0)
76      (0,0,0)
77
78      (0,0,0)
79      (0,0,0)
80      (0,0,0)
81      ()
82      )
83      )
85      (36,37,38,39,40,41,42,43,44,45,46)
86      (47,1,2, 7,12)
87      (47,1,3, 8,13)
88      (47,1,4, 9,14)
89      (47,1,5,10,15)
90      (47,1,6,11,16)
91      (37,36,38,39,40,41,42,43,44,45,46)
92      )
93      )
94      (
95      Z(
96      (C, -3, 0.7, -2, 0, -1, 0 )
97      (C, 0, 1.3, -3, 170.0, -2, 180 )

```

```

98     )
99     ReturnConnector(
100     (
101         (1,0,-3)
102         (0.7, 0, 0)
103         ( 0, 179.0, 0)
104         ( 0, 0, 180)
105
106         (0,0,1)
107         (0,0,1)
108         (0,0,0)
109     )
110 )
111 )
112 Connector()
113 allowed_groups()
114 )
115
116 (
117     Z(
118     )
119     ReturnConnector(
120     (
121         (-3,-2,-1)
122         (2.2,    0,    0)
123         ( 0, 30.0,    0)
124         ( 0,    0 , 0)
125
126         (0,0,0)
127         (0,0,0)
128         (0,0,0)
129     )
130 )
131 )
132 Connector(
133     (
134         (-3,-2,-1)
135         (0, 0, 0)
136         (0, 0, 0)
137         (0, 0, 0)

```

```

138
139     (0, 0, 1)
140     (0, 0, 1)
141     (0, 0, 1)
142     ()
143 )
144 (
145     (-3,-2,-1)
146     (0.9, 0, 0)
147     (0, 10, 180)
148     (0, 0, 0)
149
150     (0, 0, 1)
151     (0, 0, 1)
152     (0, 0, 1)
153     ()
154 )
155 )
156 allowed_groups(
157     (48,49,50,51,52,53,54,55,56,57,58)
158     (48,49,50,51,52,53,54,55,56,57,58)
159 )
160 )
161 (
162     Z(
163     )
164     ReturnConnector(
165     (
166         (-3,-2,-1)
167         (2.2,      0,      0)
168         ( 0, 30.0,      0)
169         ( 0,      0 , 0)
170
171         (0,0,0)
172         (0,0,0)
173         (0,0,0)
174         ()
175     )
176 )
177 Connector(

```

```

178      (
179      (-3,-2,-1)
180      (0, 0, 0)
181      (0, 0, 0)
182      (0, 0, 0)
183
184      (0, 0, 1)
185      (0, 0, 1)
186      (0, 0, 1)
187      ()
188      )
189      (
190      (-3,-2,-1)
191      (0.9, 0, 0)
192      (0, 10, 180)
193      (0, 0, 0)
194
195      (0, 0, 1)
196      (0, 0, 1)
197      (0, 0, 1)
198      ()
199      )
200      )
201      allowed_groups(
202      (48,49,50,51,52,53,54,55,56,57,58)
203      (48,49,50,51,52,53,54,55,56,57,58)
204      )
205      )
206      (
207      Z(
208      )
209      ReturnConnector(
210      (
211      (-3,-2,-1)
212      (2.2,      0,      0)
213      ( 0, 30.0,      0)
214      ( 0,      0      , 0)
215
216      (0,0,0)
217      (0,0,0)

```

```

218      (0,0,0)
219      ()
220    )
221  )
222  Connector(
223    (
224      (-3,-2,-1)
225      (0, 0, 0)
226      (0, 0, 0)
227      (0, 0, 0)
228
229      (0, 0, 1)
230      (0, 0, 1)
231      (0, 0, 1)
232    )
233  )
234    (
235      (-3,-2,-1)
236      (0.9, 0, 0)
237      (0, 10, 180)
238      (0, 0, 0)
239
240      (0, 0, 1)
241      (0, 0, 1)
242      (0, 0, 1)
243    )
244  )
245  )
246  allowed_groups(
247    (48,49,50,51,52,53,54,55,56,57,58)
248    (48,49,50,51,52,53,54,55,56,57,58)
249  )
250  )
251  (
252    Z(
253    )
254    ReturnConnector(
255      (
256        (-3,-2,-1)
257        (2.2,      0,      0)

```

```

258      ( 0, 30.0, 0)
259      ( 0, 0 , 0)
260
261      (0,0,0)
262      (0,0,0)
263      (0,0,0)
264      ()
265  )
266  )
267  Connector(
268  (
269      (-3,-2,-1)
270      (0, 0, 0)
271      (0, 0, 0)
272      (0, 0, 0)
273
274      (0, 0, 1)
275      (0, 0, 1)
276      (0, 0, 1)
277      ()
278  )
279  (
280      (-3,-2,-1)
281      (0.9, 0, 0)
282      (0, 10, 180)
283      (0, 0, 0)
284
285      (0, 0, 1)
286      (0, 0, 1)
287      (0, 0, 1)
288      ()
289  )
290  )
291  allowed_groups(
292      (48,49,50,51,52,53,54,55,56,57,58)
293      (48,49,50,51,52,53,54,55,56,57,58)
294  )
295  )
296  (
297  Z(

```

```

298 )
299 ReturnConnector(
300 (
301 (-3,-2,-1)
302 (2.2, 0, 0)
303 ( 0, 30.0, 0)
304 ( 0, 0 , 0)
305
306 (0,0,0)
307 (0,0,0)
308 (0,0,0)
309 ()
310 )
311 )
312 Connector(
313 (
314 (-3,-2,-1)
315 (0, 0, 0)
316 (0, 0, 0)
317 (0, 0, 0)
318
319 (0, 0, 1)
320 (0, 0, 1)
321 (0, 0, 1)
322 ()
323 )
324 (
325 (-3,-2,-1)
326 (0.9, 0, 0)
327 (0, 10, 180)
328 (0, 0, 0)
329
330 (0, 0, 1)
331 (0, 0, 1)
332 (0, 0, 1)
333 ()
334 )
335 )
336 allowed_groups(
337 (48,49,50,51,52,53,54,55,56,57,58)

```

```

338     (48,49,50,51,52,53,54,55,56,57,58)
339 )
340 )
341
342 (
343     Z(
344         (C, -3, 0.7, -2, 0, -1, 0 )
345         (C, -3, 3.7, 0,1.0, -1, 180 )
346     )
347     ReturnConnector(
348         (
349             (1,0,-3)
350             (0.7, 0, 0)
351             ( 0, 178.0, 0)
352             ( 0, 0, 180)
353
354             (0,0,1)
355             (0,0,1)
356             (0,0,0)
357             ()
358         )
359     )
360     Connector(
361         (
362             (0,1,-3)
363             (0.7,0,0)
364             (0,60,0)
365             (0,0,180)
366
367             (0,0,1)
368             (0,0,1)
369             (0,0,0)
370             ()
371         )
372     (
373         (0,1,-3)
374         (0.7,0,0)
375         (0,60,0)
376         (0,0,0)
377

```



```

378      (0,0,1)
379      (0,0,1)
380      (0,0,0)
381      ()
382    )
383    (
384      (1,0,-3)
385      (0.7,0,-180)
386      (0,60,0)
387      (0,0,180)
388
389      (0,0,1)
390      (0,0,1)
391      (0,0,0)
392      ()
393    )
394    (
395      (1,0,-3)
396      (0.7,0,-180)
397      (0,60,)
398      (0,0,0)
399
400      (0,0,1)
401      (0,0,1)
402      (0,0,0)
403      ()
404    )
405  )
406  allowed_groups(
407    (17,18,19,20,21,22,23,24,25,26,27)
408    (17,18,19,20,21,22,23,24,25,26,27)
409    (17,18,19,20,21,22,23,24,25,26,27)
410    (17,18,19,20,21,22,23,24,25,26,27)
411  )
412  )
413  (
414    Z(
415      (C, -3, 0.7, -2, 0, -1, 0 )
416      (C, -3, 3.7, 0,1.0, -1, 180 )
417    )

```

```

418 ReturnConnector(
419     (
420         (1,0,-3)
421         (0.7, 0, 0)
422         ( 0, 178.0, 0)
423         ( 0, 0, 180)
424
425         (0,0,1)
426         (0,0,1)
427         (0,0,0)
428         ()
429     )
430 )
431 Connector(
432     (
433         (0,1,-3)
434         (0.7,0,0)
435         (0,60,0)
436         (0,0,180)
437
438         (0,0,1)
439         (0,0,1)
440         (0,0,0)
441         ()
442     )
443     (
444         (0,1,-3)
445         (0.7,0,0)
446         (0,60,0)
447         (0,0,0)
448
449         (0,0,1)
450         (0,0,1)
451         (0,0,0)
452         ()
453     )
454     (
455         (1,0,-3)
456         (0.7,0,-180)
457         (0,60,0)

```

```

458      (0,0,180)
459
460      (0,0,1)
461      (0,0,1)
462      (0,0,0)
463      ()
464  )
465  (
466      (1,0,-3)
467      (0.7,0,-180)
468      (0,60,)
469      (0,0,0)
470
471      (0,0,1)
472      (0,0,1)
473      (0,0,0)
474      ()
475  )
476  )
477  allowed_groups(
478      (17,18,19,20,21,22,23,24,25,26,27)
479      (17,18,19,20,21,22,23,24,25,26,27)
480      (17,18,19,20,21,22,23,24,25,26,27)
481      (17,18,19,20,21,22,23,24,25,26,27)
482  )
483  )
484  (
485      Z(
486          (C, -3, 0.7, -2, 0, -1, 0 )
487          (C, -3, 3.7, 0,1.0, -1, 180 )
488      )
489      ReturnConnector(
490          (
491              (1,0,-3)
492              (0.7, 0, 0)
493              ( 0, 178.0, 0)
494              ( 0, 0, 180)
495
496              (0,0,1)
497              (0,0,1)

```

```

498      (0,0,0)
499      ()
500    )
501  )
502  Connector(
503    (
504      (0,1,-3)
505      (0.7,0,0)
506      (0,60,0)
507      (0,0,180)
508
509      (0,0,1)
510      (0,0,1)
511      (0,0,0)
512    ()
513  )
514    (
515      (0,1,-3)
516      (0.7,0,0)
517      (0,60,0)
518      (0,0,0)
519
520      (0,0,1)
521      (0,0,1)
522      (0,0,0)
523    ()
524  )
525    (
526      (1,0,-3)
527      (0.7,0,-180)
528      (0,60,0)
529      (0,0,180)
530
531      (0,0,1)
532      (0,0,1)
533      (0,0,0)
534    ()
535  )
536    (
537      (1,0,-3)

```

```

538      (0.7,0,-180)
539      (0,60,)
540      (0,0,0)
541
542      (0,0,1)
543      (0,0,1)
544      (0,0,0)
545      ()
546  )
547 )
548  allowed_groups(
549      (17,18,19,20,21,22,23,24,25,26,27)
550      (17,18,19,20,21,22,23,24,25,26,27)
551      (17,18,19,20,21,22,23,24,25,26,27)
552      (17,18,19,20,21,22,23,24,25,26,27)
553  )
554 )
555 (
556  Z(
557      (C, -3, 0.7, -2, 0, -1, 0 )
558      (C, -3, 3.7, 0,1.0, -1, 180 )
559  )
560  ReturnConnector(
561      (
562          (1,0,-3)
563          (0.7, 0, 0)
564          ( 0, 178.0, 0)
565          ( 0, 0, 180)
566
567          (0,0,1)
568          (0,0,1)
569          (0,0,0)
570          ()
571      )
572  )
573  Connector(
574      (
575          (0,1,-3)
576          (0.7,0,0)
577          (0,60,0)

```

578       (0,0,180)  
 579  
 580       (0,0,1)  
 581       (0,0,1)  
 582       (0,0,0)  
 583       ()  
 584       )  
 585       (  
 586       (0,1,-3)  
 587       (0.7,0,0)  
 588       (0,60,0)  
 589       (0,0,0)  
 590  
 591       (0,0,1)  
 592       (0,0,1)  
 593       (0,0,0)  
 594       ()  
 595       )  
 596       (  
 597       (1,0,-3)  
 598       (0.7,0,-180)  
 599       (0,60,0)  
 600       (0,0,180)  
 601  
 602       (0,0,1)  
 603       (0,0,1)  
 604       (0,0,0)  
 605       ()  
 606       )  
 607       (  
 608       (1,0,-3)  
 609       (0.7,0,-180)  
 610       (0,60,)  
 611       (0,0,0)  
 612  
 613       (0,0,1)  
 614       (0,0,1)  
 615       (0,0,0)  
 616       ()  
 617       )

```

618 )
619     allowed_groups(
620         (17,18,19,20,21,22,23,24,25,26,27)
621         (17,18,19,20,21,22,23,24,25,26,27)
622         (17,18,19,20,21,22,23,24,25,26,27)
623         (17,18,19,20,21,22,23,24,25,26,27)
624     )
625 )
626 (
627     Z(
628         (C, -3, 0.7, -2, 0, -1, 0 )
629         (C, -3, 3.7, 0,1.0, -1, 180 )
630     )
631     ReturnConnector(
632         (
633             (1,0,-3)
634             (0.7, 0, 0)
635             ( 0, 178.0, 0)
636             ( 0, 0, 180)
637
638             (0,0,1)
639             (0,0,1)
640             (0,0,0)
641             ()
642         )
643     )
644     Connector(
645         (
646             (0,1,-3)
647             (0.7,0,0)
648             (0,60,0)
649             (0,0,180)
650
651             (0,0,1)
652             (0,0,1)
653             (0,0,0)
654             ()
655         )
656     (
657         (0,1,-3)

```

```

658      (0.7,0,0)
659      (0,60,0)
660      (0,0,0)
661
662      (0,0,1)
663      (0,0,1)
664      (0,0,0)
665      ()
666  )
667  (
668      (1,0,-3)
669      (0.7,0,-180)
670      (0,60,0)
671      (0,0,180)
672
673      (0,0,1)
674      (0,0,1)
675      (0,0,0)
676      ()
677  )
678  (
679      (1,0,-3)
680      (0.7,0,-180)
681      (0,60,)
682      (0,0,0)
683
684      (0,0,1)
685      (0,0,1)
686      (0,0,0)
687      ()
688  )
689  )
690  allowed_groups(
691      (17,18,19,20,21,22,23,24,25,26,27)
692      (17,18,19,20,21,22,23,24,25,26,27)
693      (17,18,19,20,21,22,23,24,25,26,27)
694      (17,18,19,20,21,22,23,24,25,26,27)
695  )
696  )
697

```



```

698 (
699   Z(
700     (C, -3, 0.7, -2, 0, -1, 0)
701     (C, 0, 2.5, -3,162, -2, 180)
702   )
703   ReturnConnector(
704     (
705       (1,0,-3)
706       (0.7, 0, 0)
707       ( 0, 150, 0)
708       ( 0, 0, 0)
709
710       (0,0,1)
711       (0,0,1)
712       (0,0,0)
713       ()
714     )
715   )
716   Connector(
717     (
718       (1,0,-3)
719       (0.7, 0, 0)
720       (0, 36, 0)
721       (0,0,0)
722
723       (0,0,1)
724       (0,0,1)
725       (0,0,0)
726       ()
727     )
728     (
729       (0,1,-3)
730       (0.7, 0, 0)
731       (0, 72, 0)
732       (0,0,180)
733
734       (0,0,1)
735       (0,0,1)
736       (0,0,0)
737       ()

```

```

738     )
739     (
740         (1,0,-3)
741         (0.7, 0, 0)
742         (0, 72, 0)
743         (0,0,180)
744
745         (0,0,1)
746         (0,0,1)
747         (0,0,0)
748     )
749 )
750 )
751 allowed_groups(
752     (28,29,30,31,32,33,34,35)
753     (17,18,19,20,21,22,23,24,25,26,27)
754     (17,18,19,20,21,22,23,24,25,26,27)
755 )
756 )
757 (
758     Z(
759         (C, -3, 0.7, -2, 0, -1, 0)
760         (C, 0, 2.5, -3,162, -2, 180)
761     )
762     ReturnConnector(
763         (
764             (1,0,-3)
765             (0.7, 0, 0)
766             ( 0, 150, 0)
767             ( 0, 0, 0)
768
769             (0,0,1)
770             (0,0,1)
771             (0,0,0)
772         )
773     )
774 )
775 Connector(
776     (
777         (1,0,-3)

```

```

778      (0.7, 0, 0)
779      (0, 36, 0)
780      (0,0,0)
781
782      (0,0,1)
783      (0,0,1)
784      (0,0,0)
785      ()
786  )
787  (
788      (0,1,-3)
789      (0.7, 0, 0)
790      (0, 72, 0)
791      (0,0,180)
792
793      (0,0,1)
794      (0,0,1)
795      (0,0,0)
796      ()
797  )
798  (
799      (1,0,-3)
800      (0.7, 0, 0)
801      (0, 72, 0)
802      (0,0,180)
803
804      (0,0,1)
805      (0,0,1)
806      (0,0,0)
807      ()
808  )
809  )
810  allowed_groups(
811      (28,29,30,31,32,33,34,35)
812      (17,18,19,20,21,22,23,24,25,26,27)
813      (17,18,19,20,21,22,23,24,25,26,27)
814  )
815  )
816  (
817  Z(

```

```

818     (C, -3, 0.7, -2, 0, -1, 0)
819     (C, 0, 2.5, -3,162, -2, 180)
820 )
821 ReturnConnector(
822     (
823         (1,0,-3)
824         (0.7, 0, 0)
825         ( 0, 150, 0)
826         ( 0, 0, 0)
827
828         (0,0,1)
829         (0,0,1)
830         (0,0,0)
831         ()
832     )
833 )
834 Connector(
835     (
836         (1,0,-3)
837         (0.7, 0, 0)
838         (0, 36, 0)
839         (0,0,0)
840
841         (0,0,1)
842         (0,0,1)
843         (0,0,0)
844         ()
845     )
846     (
847         (0,1,-3)
848         (0.7, 0, 0)
849         (0, 72, 0)
850         (0,0,180)
851
852         (0,0,1)
853         (0,0,1)
854         (0,0,0)
855         ()
856     )
857     (

```

```

858      (1,0,-3)
859      (0.7, 0, 0)
860      (0, 72, 0)
861      (0,0,180)
862
863      (0,0,1)
864      (0,0,1)
865      (0,0,0)
866      ()
867  )
868  )
869  allowed_groups(
870      (28,29,30,31,32,33,34,35)
871      (17,18,19,20,21,22,23,24,25,26,27)
872      (17,18,19,20,21,22,23,24,25,26,27)
873  )
874  )
875  (
876  Z(
877      (C, -3, 0.7, -2, 0, -1, 0)
878      (C, 0, 2.5, -3,162, -2, 180)
879  )
880  ReturnConnector(
881      (
882          (1,0,-3)
883          (0.7, 0, 0)
884          ( 0, 150, 0)
885          ( 0, 0, 0)
886
887          (0,0,1)
888          (0,0,1)
889          (0,0,0)
890          ()
891      )
892  )
893  Connector(
894      (
895          (1,0,-3)
896          (0.7, 0, 0)
897          (0, 36, 0)

```

```

898      (0,0,0)
899
900      (0,0,1)
901      (0,0,1)
902      (0,0,0)
903      ()
904  )
905  (
906      (0,1,-3)
907      (0.7, 0, 0)
908      (0, 72, 0)
909      (0,0,180)
910
911      (0,0,1)
912      (0,0,1)
913      (0,0,0)
914      ()
915  )
916  (
917      (1,0,-3)
918      (0.7, 0, 0)
919      (0, 72, 0)
920      (0,0,180)
921
922      (0,0,1)
923      (0,0,1)
924      (0,0,0)
925      ()
926  )
927  )
928  allowed_groups(
929      (28,29,30,31,32,33,34,35)
930      (17,18,19,20,21,22,23,24,25,26,27)
931      (17,18,19,20,21,22,23,24,25,26,27)
932  )
933  )
934  (
935  Z(
936      (C, -3, 0.7, -2, 0, -1, 0)
937      (C, 0, 2.5, -3,162, -2, 180)

```

```

938 )
939     ReturnConnector(
940     (
941         (1,0,-3)
942         (0.7, 0, 0)
943         ( 0, 150, 0)
944         ( 0, 0, 0)
945
946         (0,0,1)
947         (0,0,1)
948         (0,0,0)
949         ()
950     )
951 )
952     Connector(
953     (
954         (1,0,-3)
955         (0.7, 0, 0)
956         (0, 36, 0)
957         (0,0,0)
958
959         (0,0,1)
960         (0,0,1)
961         (0,0,0)
962         ()
963     )
964     (
965         (0,1,-3)
966         (0.7, 0, 0)
967         (0, 72, 0)
968         (0,0,180)
969
970         (0,0,1)
971         (0,0,1)
972         (0,0,0)
973         ()
974     )
975     (
976         (1,0,-3)
977         (0.7, 0, 0)

```

```

978      (0, 72, 0)
979      (0,0,180)
980
981      (0,0,1)
982      (0,0,1)
983      (0,0,0)
984      ()
985  )
986  )
987  allowed_groups(
988      (28,29,30,31,32,33,34,35)
989      (17,18,19,20,21,22,23,24,25,26,27)
990      (17,18,19,20,21,22,23,24,25,26,27)
991  )
992  )
993
994  (
995      Z(
996      (C, -3, 0.7, -2, 0, -1, 0)
997      (H, 0, 1.1, -3, 120, -2, 180)
998  )
999  ReturnConnector()
1000  Connector()
1001  allowed_groups()
1002  )
1003  (
1004      Z(
1005      (C, -3, 0.7, -2, 0, -1, 0)
1006      (N, 0, 1.4, -3, 120, -2, 180)
1007      (H, 1, 1.1, 0, 120, -3, 150)
1008      (H, 1, 1.1, 0, 120, 2, 150)
1009  )
1010  ReturnConnector()
1011  Connector()
1012  allowed_groups()
1013  )
1014  (
1015      Z(
1016      (C, -3, 0.7, -2, 0, -1, 0)
1017      (N, 0, 1.4, -3, 120, -2, 180)

```



```

1018    (0, 1, 1.2, 0, 120, -3, 150)
1019    (0, 1, 1.2, 0, 120, 2, 180)
1020 )
1021 ReturnConnector()
1022 Connector()
1023 allowed_groups()
1024 )
1025 (
1026 Z(
1027   (C, -3, 0.7, -2, 0, -1, 0)
1028   (N, 0, 1.4, -3, 120, -2, 180)
1029   (H, 1, 1.1, 0, 120, -3, 150)
1030   (C, 1, 1.4, 0, 120, 2, 180)
1031   (O, 3, 1.3, 1, 120, 2, 0(180))
1032   (C, 3, 1.5, 1, 120, 4, 180)
1033   (H, 5, 1.1, 3, 109.47, 4, 180)
1034   (H, 5, 1.1, 3, 109.47, 6, 120)
1035   (H, 5, 1.1, 3, 109.47, 7, 120)
1036 )
1037 ReturnConnector()
1038 Connector()
1039 allowed_groups()
1040 )
1041 (
1042 Z(
1043   (C, -3, 0.7, -2, 0, -1, 0)
1044   (C, 0, 1.4, -3, 120, -2, 180)
1045   (O, 1, 1.3, 0, 120, -3, -30(150))
1046   (C, 1, 1.5, 0, 120, 2, 180)
1047   (H, 3, 1.1, 1, 109.47, 2, 180)
1048   (H, 3, 1.1, 1, 109.47, 4, 120)
1049   (H, 3, 1.1, 1, 109.47, 5, 120)
1050 )
1051 ReturnConnector()
1052 Connector()
1053 allowed_groups()
1054 )
1055 (
1056 Z(
1057   (C, -3, 0.7, -2, 0, -1, 0)

```

```

1058     (0, 0, 1.4, -3, 120, -2, 180)
1059     (C, 1, 1.4, 0, 120, -3, -30(150))
1060     (0, 2, 1.3, 1, 120, 0, 0(180))
1061     (C, 2, 1.5, 1, 120, 3, 180)
1062     (H, 4, 1.1, 2, 109.47, 3,180)
1063     (H, 4, 1.1, 2, 109.47, 5, 120)
1064     (H, 4, 1.1, 2, 109.47, 6, 120)
1065 )
1066 ReturnConnector()
1067 Connector()
1068 allowed_groups()
1069 )
1070 (
1071   Z(
1072     (C, -3, 0.7, -2, 0, -1, 0)
1073     (C, 0, 1.5, -3, 120, -2, 180)
1074     (N, 0, 2.8, -3, 170, -2, 180)
1075   )
1076 ReturnConnector()
1077 Connector()
1078 allowed_groups()
1079 )
1080 (
1081   Z(
1082     (C, -3, 0.7, -2, 0, -1, 0)
1083     (N, 0, 1.4, -3, 120, -2, 180)
1084     (C, 1, 1.2, 0, 120, -3, 150)
1085     (C, 1, 1.2, 0, 120, 2, 180)
1086
1087     (H, 2, 1.1, 1, 109.47, 0,180)
1088     (H, 2, 1.1, 1, 109.47, 4, 120)
1089     (H, 2, 1.1, 1, 109.47, 5, 120)
1090
1091     (H, 3, 1.1, 1, 109.47, 0,180)
1092     (H, 3, 1.1, 1, 109.47, 7, 120)
1093     (H, 3, 1.1, 1, 109.47, 8, 120)
1094   )
1095 ReturnConnector()
1096 Connector()
1097 allowed_groups()

```

```

1098 )
1099 (
1100   Z(
1101     (C, -3, 0.7, -2, 0, -1, 0)
1102     (O, 0, 1.4, -3, 120, -2, 180)
1103     (C, 1, 1.5, 0, 120, -3, -30(150))
1104     (H, 2, 1.1, 1, 109.47, 0, 180)
1105     (H, 2, 1.1, 1, 109.47, 3, 120)
1106     (H, 2, 1.1, 1, 109.47, 4, 120)
1107   )
1108   ReturnConnector()
1109   Connector()
1110   allowed_groups()
1111 )
1112 (
1113   Z(
1114     (C, -3, 0.7, -2, 0, -1, 0)
1115     (S, 0, 1.6, -3, 120, -2, 180)
1116     (C, 1, 1.5, 0, 120, -3, -30(150))
1117     (H, 2, 1.1, 1, 109.47, 0, 180)
1118     (H, 2, 1.1, 1, 109.47, 3, 120)
1119     (H, 2, 1.1, 1, 109.47, 4, 120)
1120   )
1121   ReturnConnector()
1122   Connector()
1123   allowed_groups()
1124 )
1125 (
1126   Z(
1127     (N, -3, 0.6, -2, 0, -1, 0)
1128   )
1129   ReturnConnector()
1130   Connector()
1131   allowed_groups()
1132 )
1133
1134
1135 (
1136   Z(
1137     (O, -3, 0.7, -2, 0, -1, 0)

```

```

1138 )
1139 ReturnConnector()
1140 Connector()
1141 allowed_groups()
1142 )
1143 (
1144 Z(
1145 (S, -3, 0.7, -2, 0, -1, 0)
1146 )
1147 ReturnConnector()
1148 Connector()
1149 allowed_groups()
1150 )
1151 (
1152 Z(
1153 (Si, -3, 0.7, -2, 0, -1, 0)
1154 (H, 0, 1.4, -3, 109.47, -2, 120)
1155 (H, 0, 1.4, 1, 109.47, -3, -120)
1156 )
1157 ReturnConnector()
1158 Connector()
1159 allowed_groups()
1160 )
1161 (
1162 Z(
1163 (S, -3, 0.7, -2, 0, -1, 0)
1164 (O, 0, 1.9, -3, 109.47, -2, 120)
1165 (O, 0, 1.9, 1, 109.47, -3, -120)
1166 )
1167 ReturnConnector()
1168 Connector()
1169 allowed_groups()
1170 )
1171 (
1172 Z(
1173 (C, -3, 0.7, -2, 0, -1, 0)
1174 (H, 0, 1.1, -3, 109.47, -2, 120)
1175 (H, 0, 1.1, 1, 109.47, -3, -120)
1176 )
1177 ReturnConnector()

```

```

1178 Connector()
1179 allowed_groups()
1180 )
1181 (
1182 Z(
1183 (C, -3, 0.7, -2, 0, -1, 0)
1184 (O, 0, 1.3, -3, 120, -2(1), -180)
1185 )
1186 ReturnConnector()
1187 Connector()
1188 allowed_groups()
1189 )
1190 (
1191 Z(
1192 (N, -3, 0.7, -2, 0, -1, 0)
1193 (C, 0, 1.5, -3, 120.00, -2, -180)
1194 (H, 1, 1.1, 0, 109.47, -3, 60)
1195 (H, 1, 1.1, 0, 109.47, 2, 120)
1196 (H, 1, 1.1, 0, 109.47, 3, 120)
1197 )
1198 ReturnConnector()
1199 Connector()
1200 allowed_groups()
1201 )
1202 (
1203 Z(
1204 (C, -3, 0.7, -2, 0, -1, 0)
1205 (C, 0, 1.4, -3, 120, -2, 180)
1206 (H, 1, 1.1, 0, 120.0, -3, 180)
1207 (H, 1, 1.1, 0, 120.0, 2, 180)
1208 )
1209 ReturnConnector()
1210 Connector()
1211 allowed_groups()
1212 )
1213
1214
1215 (
1216 Z(
1217 (N, -3, 0.7, -2, 0, -1, 0)

```

```

1218 (C, 0, 1.4, -3, 120, -2, 60)
1219 (C, 0, 1.4, 1, 120, -3, 120(-240))
1220 (H, 1, 1.1, 0, 109.47, 2(1), 180)
1221 (H, 1, 1.1, 0, 109.47, 3, 120)
1222 (H, 1, 1.1, 0, 109.47, 4, 120)
1223 (H, 2, 1.1, 0, 109.47, 1, 80)
1224 (H, 2, 1.1, 0, 109.47, 6, 120)
1225 (H, 2, 1.1, 0, 109.47, 7, 120)
1226 )
1227 ReturnConnector(
1228 (
1229 (0,1,2)
1230 (0.7, 0,0)
1231 (0,109.47,0)
1232 (0,0,120)
1233 (0,0,1)
1234 (0,0,1)
1235 (0,0,0)
1236 ()
1237 )
1238 )
1239 Connector()
1240 allowed_groups()
1241 )
1242 (
1243 Z(
1244 (C, -3, 0.7, -2, 0, -1, 0)
1245 (N, 0, 1.2, -3, 179.0, -2, 180)
1246 )
1247 ReturnConnector(
1248 (
1249 (0,1,-3)
1250 (0.7, 0,0)
1251 (0,170,0)
1252 (0,0,180)
1253 (0,0,1)
1254 (0,0,1)
1255 (0,0,0)
1256 ()
1257 )

```

```

1258 )
1259 Connector()
1260 allowed_groups()
1261 )
1262 (
1263 Z(
1264 (C, -3, 0.7, -2, 0, -1, 0)
1265 (0, 0, 1.4, -3, 120, -2, -91(180))
1266 (H, 0, 1.1, 1, 120, -3, 180)
1267 )
1268 ReturnConnector(
1269 (
1270 (0,1,2)
1271 (0.7, 0,0)
1272 (0,109.47,0)
1273 (0,0,180)
1274 (0,0,1)
1275 (0,0,1)
1276 (0,0,0)
1277 ()
1278 )
1279 )
1280 Connector()
1281 allowed_groups()
1282 )
1283 (
1284 Z(
1285 (0, -3, 1.4, -2, 0, -1, 0)
1286 (C, 0, 1.4, -3, 120, -2, -91(180))
1287 (H, 1, 1.1, 0, 109.47, -3, 180)
1288 (H, 1, 1.1, 0, 109.47, 2, 120)
1289 (H, 1, 1.1, 0, 109.47, 3, 120)
1290 )
1291 ReturnConnector(
1292 (
1293 (0,1,2)
1294 (0.7, 0,0)
1295 (0,109.47,0)
1296 (0,0,120)
1297 (0,0,1)

```

```

1298      (0,0,1)
1299      (0,0,0)
1300      ()
1301  )
1302  )
1303  Connector()
1304  allowed_groups()
1305  )
1306  (
1307  Z(
1308  (C, -3, 1.5, -2, 0, -1, 0)
1309  (0, 0, 1.3, -3, 120, -2, 180)
1310  (C, 0, 1.5, 1, 120, -3, 180)
1311  (H, 2, 1.1, 0, 109.47, 1, 180)
1312  (H, 2, 1.1, 0, 109.47, 3, 120)
1313  (H, 2, 1.1, 0, 109.47, 4, 120)
1314  )
1315  ReturnConnector(
1316  (
1317      (0,1,2)
1318      (0.7, 0,0)
1319      (0,109.47,0)
1320      (0,0,180)
1321      (0,0,1)
1322      (0,0,1)
1323      (0,0,0)
1324      ()
1325  )
1326  )
1327  Connector()
1328  allowed_groups()
1329  )
1330  (
1331  Z(
1332  (N, -3, 0.7, -2, 0, -1, 0)
1333  (C, 0, 1.4, -3, 120, -2, -91)
1334  (0, 1, 1.3, 0, 120, -3, 0)
1335  (C, 1, 1.5, 0, 120, 2, 180)
1336  (H, 3, 1.1, 1, 109.47, 0, 180)
1337  (H, 3, 1.1, 1, 109.47, 4, 120)

```



```

1338     (H,  3, 1.1,  1, 109.47,  5, 120)
1339
1340     (H,  0, 1.1,  1, 120, 2, 180)
1341 )
1342 ReturnConnector(
1343     (
1344         (0,1,2)
1345         (0.7, 0,0)
1346         (0,109.47,0)
1347         (0,0,10)
1348         (0,0,1)
1349         (0,0,1)
1350         (0,0,0)
1351     )
1352 )
1353 )
1354 Connector()
1355 allowed_groups()
1356 )
1357 (
1358     Z(
1359     (H,  -3, 0.4, -2, 0, -1, 0)
1360     )
1361     ReturnConnector(
1362     (
1363         (0,-3,-2)
1364         (0.4, 0,0)
1365         (0,109.47,0)
1366         (0,0,180)
1367         (0,0,1)
1368         (0,0,1)
1369         (0,0,0)
1370     )
1371 )
1372 )
1373 Connector()
1374 allowed_groups()
1375 )
1376 (
1377     Z(

```

```

1378 (N, -3, 0.7, -2, 0, -1, 0)
1379 (0, 0, 1.2, -3, 120, -2, -80)
1380 (0, 0, 1.2, 1, 120, -3, 180)
1381 )
1382 ReturnConnector(
1383 (
1384 (0,1,2)
1385 (0.7, 0,0)
1386 (0,109.47,0)
1387 (0,0,180)
1388 (0,0,1)
1389 (0,0,1)
1390 (0,0,0)
1391 ()
1392 )
1393 )
1394 Connector()
1395 allowed_groups()
1396 )
1397 (
1398 Z(
1399 (o, -3, 0.6, -2, 0, -1, 0)
1400 (c, 0, 1.4, -3, 120, -2, 0(180))
1401 (o, 1, 1.3, 0, 120, -3, 0(180))
1402 (c, 1, 1.5, 0, 120, 2, 180)
1403 (H, 3, 1.1, 1, 109.47, 0, 180)
1404 (h, 3, 1.1, 1, 109.47, 4, 120)
1405 (h, 3, 1.1, 1, 109.47, 5, 120)
1406 )
1407 ReturnConnector(
1408 (
1409 (0,1,2)
1410 (0.7, 0,0)
1411 (0,120,0)
1412 (0,0,180)
1413 (0,0,1)
1414 (0,0,1)
1415 (0,0,0)
1416 ()
1417 )

```

```

1418 )
1419 Connector()
1420 allowed_groups()
1421 )
1422 (
1423 Z(
1424 (0, -3, 0.6, -2, 0, -1, 0)
1425 (H, 0, 1.1, -3, 120, -2, -91(180))
1426 )
1427 ReturnConnector(
1428 (
1429 (0,1,-3)
1430 (0.7, 0,0)
1431 (0,109.47,0)
1432 (0,0,120)
1433 (0,0,1)
1434 (0,0,1)
1435 (0,0,0)
1436 ()
1437 )
1438 )
1439 Connector()
1440 allowed_groups()
1441 )
1442 (
1443 Z(
1444 (N, -3, 0.7, -2, 0, -1, 0)
1445 (H, 0, 1.1, -3, 120, -2, 180)
1446 (H, 0, 1.1, 1, 109.47, -3, 120(-120))
1447 )
1448 ReturnConnector(
1449 (
1450 (0,1,2)
1451 (0.7, 0,0)
1452 (0,109.47,0)
1453 (0,0,120)
1454 (0,0,1)
1455 (0,0,1)
1456 (0,0,0)
1457 ()

```

```

1458     )
1459     )
1460     Connector()
1461     allowed_groups()
1462     )
1463     (
1464     Z()
1465     ReturnConnector(
1466     (
1467     (-3,-2,-1)
1468     (0,0,0)
1469     (0,0,0)
1470     (0,0,0)
1471     (0,0,0)
1472     (0,0,0)
1473     (0,0,0)
1474     ()
1475     )
1476     )
1477     Connector()
1478     allowed_groups()
1479     )
1480
1481     (
1482     Z(
1483     (C, -3, 0.7, -2, 0, -1, 0)
1484     (H, 0, 1.1, -3, 120, -2, 180)
1485     )
1486     ReturnConnector(
1487     (
1488     (0,1,-3)
1489     (0.7,0,0)
1490     (0,120,0)
1491     (0,0,180)
1492
1493     (0,0,1)
1494     (0,0,1)
1495     (0,0,0)
1496     ()
1497     )

```

```

1498 )
1499 Connector()
1500 allowed_groups()
1501 )
1502 (
1503 Z(
1504   (C, -3, 0.7, -2, 0, -1, 0)
1505   (N, 0, 1.4, -3, 120, -2, 180)
1506   (H, 1, 1.1, 0, 120, -3, -30(150))
1507   (H, 1, 1.1, 0, 120, 2, 150)
1508 )
1509 ReturnConnector(
1510   (
1511     (0,1,-3)
1512     (0.7,0,0)
1513     (0,120,0)
1514     (0,0,180)
1515
1516     (0,0,1)
1517     (0,0,1)
1518     (0,0,0)
1519   )
1520 ))
1521 Connector()
1522 allowed_groups()
1523 )
1524 (
1525 Z(
1526   (C, -3, 0.7, -2, 0, -1, 0)
1527   (N, 0, 1.4, -3, 120, -2, 180)
1528   (O, 1, 1.2, 0, 120, -3, 150)
1529   (O, 1, 1.2, 0, 120, 2, 180)
1530 )
1531 ReturnConnector(
1532   (
1533     (0,1,-3)
1534     (0.7,0,0)
1535     (0,120,0)
1536     (0,0,180)
1537

```

```

1538     (0,0,1)
1539     (0,0,1)
1540     (0,0,0)
1541     ()
1542 ))
1543 Connector()
1544 allowed_groups()
1545 )
1546 (
1547   Z(
1548     (C, -3, 0.7, -2, 0, -1, 0)
1549     (N, 0, 1.4, -3, 120, -2, 180)
1550     (H, 1, 1.1, 0, 120, -3, -30(150))
1551     (C, 1, 1.4, 0, 120, 2, 180)
1552     (O, 3, 1.3, 1, 120, 2, 0(180))
1553     (C, 3, 1.5, 1, 120, 4, 180)
1554     (H, 5, 1.1, 3, 109.47, 4, 180)
1555     (H, 5, 1.1, 3, 109.47, 6, 120)
1556     (H, 5, 1.1, 3, 109.47, 7, 120)
1557   )
1558   ReturnConnector(
1559     (
1560       (0,1,-3)
1561       (0.7,0,0)
1562       (0,120,0)
1563       (0,0,180)
1564
1565       (0,0,1)
1566       (0,0,1)
1567       (0,0,0)
1568     )
1569   ))
1570 Connector()
1571 allowed_groups()
1572 )
1573 (
1574   Z(
1575     (C, -3, 0.7, -2, 0, -1, 0)
1576     (C, 0, 1.4, -3, 120, -2, 180)
1577     (O, 1, 1.3, 0, 120, -3, -30(150))

```

```

1578 (C, 1, 1.5, 0, 120, 2, 180)
1579 (H, 3, 1.1, 1, 109.47, 2, 180)
1580 (H, 3, 1.1, 1, 109.47, 4, 120)
1581 (H, 3, 1.1, 1, 109.47, 5, 120)
1582 )
1583 ReturnConnector(
1584 (
1585 (0,1,-3)
1586 (0.7,0,0)
1587 (0,120,0)
1588 (0,0,180)
1589
1590 (0,0,1)
1591 (0,0,1)
1592 (0,0,0)
1593 ()
1594 ))
1595 Connector()
1596 allowed_groups()
1597 )
1598 (
1599 Z(
1600 (C, -3, 0.7, -2, 0, -1, 0)
1601 (0, 0, 1.4, -3, 120, -2, 180)
1602 (C, 1, 1.4, 0, 120, -3, -30(150))
1603 (0, 2, 1.3, 1, 120, 0, 0(180))
1604 (C, 2, 1.5, 1, 120, 3, 180)
1605 (H, 4, 1.1, 2, 109.47, 3, 180)
1606 (H, 4, 1.1, 2, 109.47, 5, 120)
1607 (H, 4, 1.1, 2, 109.47, 6, 120)
1608 )
1609 ReturnConnector(
1610 (
1611 (0,1,-3)
1612 (0.7,0,0)
1613 (0,120,0)
1614 (0,0,180)
1615
1616 (0,0,1)
1617 (0,0,1)

```

```

1618      (0,0,0)
1619      ()
1620      ))
1621  Connector()
1622  allowed_groups()
1623  )
1624  (
1625      Z(
1626          (C, -3, 0.7, -2, 0, -1, 0)
1627          (C, 0, 1.5, -3, 120, -2, 180)
1628          (N, 0, 2.8, 1, 10, -3, 180)
1629      )
1630  ReturnConnector(
1631      (
1632          (0,1,-3)
1633          (0.7,0,0)
1634          (0,120,0)
1635          (0,0,180)
1636
1637          (0,0,1)
1638          (0,0,1)
1639          (0,0,0)
1640      )
1641      ))
1642  Connector()
1643  allowed_groups()
1644  )
1645  (
1646      Z(
1647          (C, -3, 0.7, -2, 0, -1, 0)
1648          (N, 0, 1.4, -3, 120, -2, 180)
1649          (C, 1, 1.2, 0, 120, -3, 150)
1650          (C, 1, 1.2, 0, 120, 2, 180)
1651
1652          (H, 2, 1.1, 1, 109.47, 0, 180)
1653          (H, 2, 1.1, 1, 109.47, 4, 120)
1654          (H, 2, 1.1, 1, 109.47, 5, 120)
1655
1656          (H, 3, 1.1, 1, 109.47, 0, 180)
1657          (H, 3, 1.1, 1, 109.47, 7, 120)

```



```

1658     (H,  3, 1.1,  1, 109.47, 8, 120)
1659 )
1660 ReturnConnector(
1661     (
1662         (0,1,-3)
1663         (0.7,0,0)
1664         (0,120,0)
1665         (0,0,180)
1666
1667         (0,0,1)
1668         (0,0,1)
1669         (0,0,0)
1670     )
1671 ))
1672 Connector()
1673 allowed_groups()
1674 )
1675 (
1676     Z(
1677         (C, -3, 0.7, -2,  0, -1,  0)
1678         (0,  0, 1.4, -3, 120, -2, 180)
1679         (C,  1, 1.5,  0, 120, -3, -30(150))
1680         (H,  2, 1.1,  1, 109.47, 0,180)
1681         (H,  2, 1.1,  1, 109.47, 3, 120)
1682         (H,  2, 1.1,  1, 109.47, 4, 120)
1683     )
1684 ReturnConnector(
1685     (
1686         (0,1,-3)
1687         (0.7,0,0)
1688         (0,120,0)
1689         (0,0,180)
1690
1691         (0,0,1)
1692         (0,0,1)
1693         (0,0,0)
1694     )
1695 ))
1696 Connector()
1697 allowed_groups()

```

```

1698 )
1699 (
1700 Z(
1701   (C, -3, 0.7, -2, 0, -1, 0)
1702   (S, 0, 1.6, -3, 120, -2, 180)
1703   (C, 1, 1.5, 0, 120, -3, -30(150))
1704   (H, 2, 1.1, 1, 109.47, 0, 180)
1705   (H, 2, 1.1, 1, 109.47, 3, 120)
1706   (H, 2, 1.1, 1, 109.47, 4, 120)
1707 )
1708 ReturnConnector(
1709   (
1710     (0,1,-3)
1711     (0.7,0,0)
1712     (0,120,0)
1713     (0,0,180)
1714
1715     (0,0,1)
1716     (0,0,1)
1717     (0,0,0)
1718   )
1719 ))
1720 Connector()
1721 allowed_groups()
1722 )
1723 (
1724 Z(
1725   (N, -3,0.6, -2,0,-1,0)
1726 )
1727 ReturnConnector(
1728   (
1729     (0,-3,-2)
1730     (0.7,0,0)
1731     (0,120,0)
1732     (0,0,180)
1733
1734     (0,0,1)
1735     (0,0,1)
1736     (0,0,0)
1737   )

```

```
1738    ))
1739    Connector()
1740    allowed_groups()
1741  )
1742
1743  )
%
```

- 1 DEFENSE TECHNICAL  
(PDF INFORMATION CTR  
only) DTIC OCA  
8725 JOHN J KINGMAN RD  
STE 0944  
FORT BELVOIR VA 22060-6218
- 1 DIRECTOR  
US ARMY RESEARCH LAB  
IMAL HRA  
2800 POWDER MILL RD  
ADELPHI MD 20783-1197
- 1 DIRECTOR  
US ARMY RESEARCH LAB  
RDRL CIO LL  
2800 POWDER MILL RD  
ADELPHI MD 20783-1197
- 1 DIRECTOR  
US ARMY RESEARCH LAB  
RDRL CIO LT  
2800 POWDER MILL RD  
ADELPHI MD 20783-1197
- 3 DIRECTOR  
US ARMY RESEARCH LAB  
RDRL-WMM-G  
B. CHRISTOPHER RINDERSPACHER  
JAN W. ANDZELM  
TANYA L. CHANTAWANSRI  
APG MD 21005

INTENTIONALLY LEFT BLANK.