



ARL-MR-1090 • DEC 2023



Diagnostics for SPPARKS' AppSinter Class

by Efraín Hernández-Rivera and Cristina García-Cardona

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

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.



Diagnosics for SPPARKS' AppSinter Class

by Efraín Hernández-Rivera
DEVCOM Army Research Laboratory

Cristina García-Cardona
Los Alamos National Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

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.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) December 2023		2. REPORT TYPE Memorandum Report		3. DATES COVERED (From - To) 2–31 October 2023	
4. TITLE AND SUBTITLE Diagnostics for SPPARKS' AppSinter Class				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Efraín Hernández–Rivera and Cristina García–Cardona				5d. PROJECT NUMBER AH80	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) DEVCOM Army Research Laboratory ATTN: FCDD-RLA-MB Aberdeen Proving Ground, MD 21005-5066				8. PERFORMING ORGANIZATION REPORT NUMBER ARL-MR-1090	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES primary author's email: <efrain.hernandez18.civ@army.mil>. ; ORCID ID: Efrain Hernandez-Rivera, 0000-0002-7855-1027					
14. ABSTRACT Diagnostics within the SPPARKS' framework enable the computation of different model metrics, which are used to characterize microstructural evolution. A set of diagnostics was developed and released as part of the SPPARKS AppSinter class to quantify different metrics of the evolution of a sintered synthetic green body (e.g., density, porosity as a function of time). This report documents the extension of the diagnostic files currently available, including the addition of useful classes. The new diagnostic implementations are contributed to the SPPARKS project via GitHub.					
15. SUBJECT TERMS SPPARKS, diagnostic, AppSinter, GitHub, Sciences of Extreme Materials					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 31	19a. NAME OF RESPONSIBLE PERSON Efraín Hernández
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (Include area code) 410-278-6010

Contents

List of Figures	iv
1. Introduction	1
2. Extension to Diagnostics	2
3. References	4
Appendix A. Density Calculation	5
Appendix B. Free Energy Calculation	9
Appendix C. Pore Free Energy Calculation	13
Appendix D. Pore Curvature Calculation	18
Distribution List	25

List of Figures

Fig. 1	Representation of the default subdomain for calculating the green body properties	2
Fig. 2	Relative density as a function of window size.....	3

1. Introduction

Sintering is a materials processing technique used to manufacture components from powder constituents (i.e., green body compacts). Understanding the evolution of these green bodies is essential for predicting the performance of a manufactured part. One approach to model the sintering process consists on a kinetic Monte Carlo algorithm,¹ which has been shown to accurately predict the densification of a Cu-powder compact.² Furthermore, this model was extended to include diagnostics to calculate microstructural features like the density, pore curvature, and interfacial energy.³

Due to SPPARKS' extendability, these diagnostics can easily be accessed by simple commands, for example,

```
diag_style      sinter_density
```

which would activate the diagnostics to calculate the relative density. Unfortunately, at the time of this report, the pore curvature diagnostic was not included in the SPPARKS GitHub repository. Therefore, an effort was made to update a legacy version of the missing diagnostic.

Currently, the diagnostics calculate the microstructural features from a computational subdomain, which is defined as the inner third microstructural region. This is shown in Fig. 1, where a representation of a two-dimensional (2-D) green body is illustrated. To enable users to have more flexibility over the subdomain size, the diagnostics were extended to include a “window” flag. This flag ($w \in \mathbb{R} : \{0 \leq w < 0.5\}$) defines the computation subdomain as a fraction of the entire domain size. A frame of width w is uniformly imposed on all domain boundaries to specify the region that will not be used for the calculations, as shown in Fig. 1. For instance, if the domain size is 100×100 pixels and the window size is set to $w = 0.10$, the diagnostics would be calculated on the $[11,89] \times [11,89]$ subdomain, excluding a 10-pixel frame per side. Therefore, the window size must be less than 0.5, and can be enabled by using the following command

```
diag_style      sinter_density value
```

where `value` is the width (fraction) of the frame to exclude.

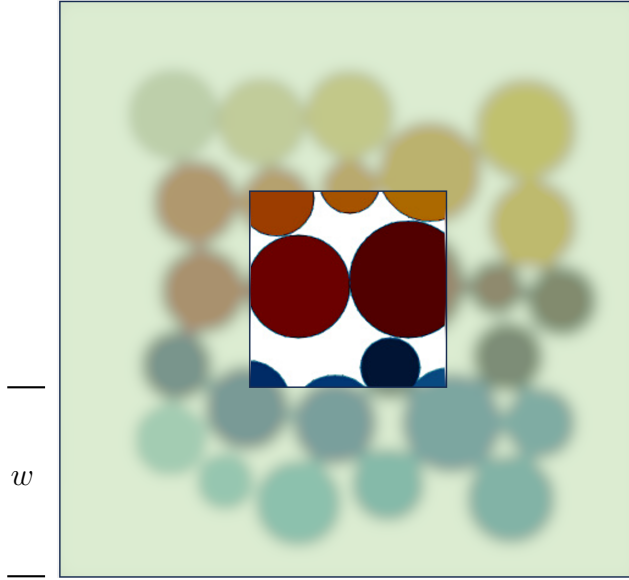


Fig. 1 Representation of the default subdomain for calculating the green body properties

The following classes are included in the release:

- `DiagSinterDensity` (Appendix A): calculate the relative density,
- `DiagSinterFreeEnergy` (Appendix B): calculate the interfacial energy,
- `DiagSinterFreeEnergyPore` (Appendix C): calculate the free surface interfacial energy,
- `DiagSinterPoreCurvature` (Appendix D): calculate pore curvature based on the innies and outties approach.³

2. Extension to Diagnostics

To demonstrate the importance of the frame width, Fig. 2 shows the relative density of the green body as a function of w . The default setting is shown by the crossed lines ($w = 0.333333$). The figure shows that for the no-window case ($w = 0$) the density is smallest due to the large empty region along the domain boundary. However, for a large window ($w = 0.45$), the calculation is focused on the center particles. This leads to a density overestimation since a particle is located near to the domain center.

Moreover, the diagnostics were adapted to include 2-D domains. This required the extension of the `edgenigh` list, which defines the voxels that constitute the edge and face neighbors. The code defines the list depending on the simulation dimension, as outlined in the `DiagSinterPoreCurvature::init` function (see Appendix D).

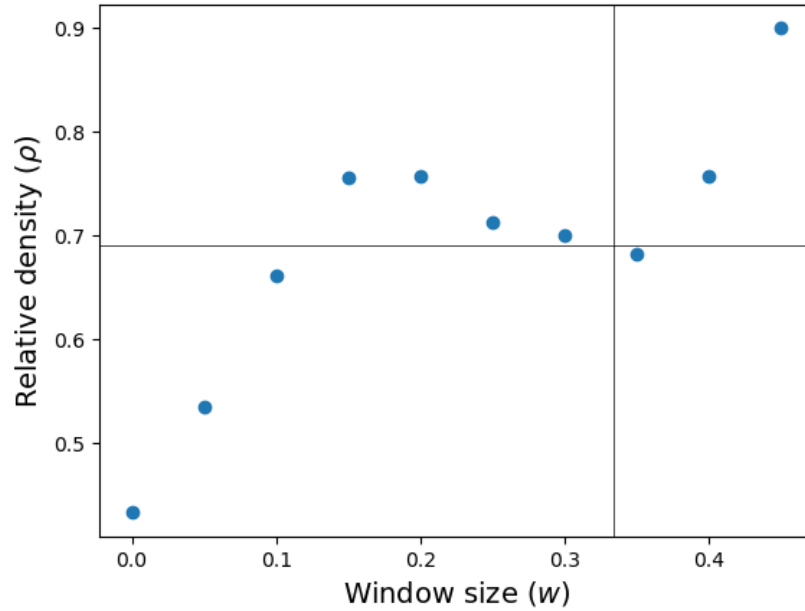


Fig. 2 Relative density as a function of window size

To fully integrate these extensions to the main SPPARKS package, a `git push` will be performed via command line operations.

3. References

1. SPPARKS AppSinter documentation. [accessed 2023 Oct 20]. https://spparks.github.io/doc/app_sinter.html
2. Tikare V, Braginsky M, Bouvard D, Vagnon A. Numerical simulation of microstructural evolution during sintering at the mesoscale in a 3D powder compact. *Computational Materials Science*. 2010;48(2):317–325.
3. Cardona CG, Tikare V, Patterson BR, Olevsky E. On sintering stress in complex powder compacts. *Journal of the American Ceramic Society*. 2012;95(8):2372–2382.

Appendix A. Density Calculation

This appendix includes the source and header files for the density diagnostic.

```
1  /* -----  
2  SPPARKS - Stochastic Parallel PARTICle Kinetic Simulator  
3  http://www.cs.sandia.gov/~sjplimp/spparks.html  
4  Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories  
5  
6  Copyright (2008) Sandia Corporation. Under the terms of Contract  
7  DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains  
8  certain rights in this software. This software is distributed under  
9  the GNU General Public License.  
10  
11  See the README file in the top-level SPPARKS directory.  
12  ----- */  
13  
14 #include "mpi.h"  
15 #include "stdlib.h"  
16 #include "string.h"  
17 #include "output.h"  
18 #include "memory.h"  
19 #include "app.h"  
20 #include "error.h"  
21 #include "timer.h"  
22 #include "diag_sinter_density.h"  
23 #include "app_lattice.h"  
24 #include "app_sinter.h"  
25 #include "comm_lattice.h"  
26 #include "domain.h"  
27  
28 using namespace SPPARKS_NS;  
29  
30 /* ----- */  
31  
32 DiagSinterDensity::DiagSinterDensity(SPPARKS *spk, int narg, char **arg) :  
33   Diag(spk, narg, arg)  
34 {  
35   if (app->appclass != App::LATTICE)  
36     error->all(FLERR, "Diag style incompatible with app style");  
37  
38   if (narg>2)  
39     error->all(FLERR, "Too many arguments for diag style");  
40  
41   if (narg==2) {  
42     window = atof(arg[1]);  
43     if (window>=0.5)  
44       error->all(FLERR, "Window size is too large (set to w<0.5)");  
45   }  
46   else  
47     window = 0.333333;  
48 }  
49  
50 /* ----- */  
51  
52 void DiagSinterDensity::init()  
53 {  
54   appsinter = (AppSinter *) app;  
55   nlocal = appsinter->nlocal;  
56   density = 0.0;  
57  
58   initialize_parameters_density_calculation();  
59 }  
60  
61 /* ----- */  
62
```

```

63 void DiagSinterDensity::compute()
64 {
65     double grain_sites = 0;
66     double total_sites = 0;
67     int xgrid, ygrid, zgrid;
68
69     int *spin = appsinter->spin;
70     tagint *id = appsinter->id;
71     const int VACANT ( AppSinter::VACANT );
72
73     for ( int i = 0; i < nlocal; i++ ) {
74         appsinter->global_to_grid( id[i], xgrid, ygrid, zgrid );
75         if ( ((xgrid > xstart_density && xgrid < xend_density) || nx==1) &&
76             ((ygrid > ystart_density && ygrid < yend_density) || ny==1) &&
77             ((zgrid > zstart_density && zgrid < zend_density) || nz==1) ) {
78             total_sites++;
79             if ( spin[i] > VACANT ) grain_sites++;
80         }
81     }
82
83     double local_density_info[2];
84     local_density_info[0] = grain_sites;
85     local_density_info[1] = total_sites;
86
87     double global_density_info[2] = {0,0};
88     MPI_Allreduce(local_density_info, global_density_info, 2, MPI_DOUBLE,
89                 MPI_SUM, world);
90
91     density = global_density_info[0] / global_density_info[1];
92 }
93 /* ----- */
94
95 void DiagSinterDensity::stats(char *strtmp)
96 {
97     sprintf(strtmp, " %10.6lf", density);
98 }
99
100 /* ----- */
101
102 void DiagSinterDensity::stats_header(char *strtmp)
103 {
104     sprintf(strtmp, " %10s", "Density");
105 }
106
107 /* ----- */
108
109 void DiagSinterDensity::initialize_parameters_density_calculation()
110 {
111     // num sites along each axis in lattice
112     nx = domain->nx;
113     ny = domain->ny;
114     nz = domain->nz;
115
116     xstart_density = static_cast<int>(nx*window);
117     xend_density   = static_cast<int>(nx*(1-window));
118     ystart_density = static_cast<int>(ny*window);
119     yend_density   = static_cast<int>(ny*(1-window));
120     zstart_density = static_cast<int>(nz*window);
121     zend_density   = static_cast<int>(nz*(1-window));
122 }

```

Listing 1 Source for the density calculation

```

1  /* -----
2  SPPARKS - Stochastic Parallel PARTicle Kinetic Simulator
3  http://www.cs.sandia.gov/~sjplimp/spparks.html
4  Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories
5
6  Copyright (2008) Sandia Corporation. Under the terms of Contract
7  DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
8  certain rights in this software. This software is distributed under
9  the GNU General Public License.
10
11  See the README file in the top-level SPPARKS directory.
12 ----- */
13
14 #ifndef DIAG_CLASS
15 DiagStyle(sinter_density, DiagSinterDensity)
16
17 #else
18
19 #ifndef SPK_DIAG_SINTER_DENSITY_H
20 #define SPK_DIAG_SINTER_DENSITY_H
21
22 #include "stdio.h"
23 #include "diag.h"
24
25 namespace SPPARKS_NS {
26
27 class DiagSinterDensity : public Diag {
28
29 public:
30   DiagSinterDensity(class SPPARKS *, int, char **);
31   ~DiagSinterDensity() {}
32   void init();
33   void compute();
34   void stats(char *);
35   void stats_header(char *);
36
37 protected:
38   void initialize_parameters_density_calculation();
39
40 private:
41   class AppSinter *appsinter;
42   int nlocal;
43   double density, window;
44
45   int xstart_density, xend_density, nx;
46   int ystart_density, yend_density, ny;
47   int zstart_density, zend_density, nz;
48 };
49
50 }
51
52 #endif
53
54 #endif

```

Listing 2 Header for the density calculation

Appendix B. Free Energy Calculation

This appendix includes the source and header files for the global free surface energy diagnostic.

```

1  /* -----
2     SPPARKS - Stochastic Parallel PARTicle Kinetic Simulator
3     http://www.cs.sandia.gov/~sjplimp/spparks.html
4     Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories
5
6     Copyright (2008) Sandia Corporation. Under the terms of Contract
7     DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
8     certain rights in this software. This software is distributed under
9     the GNU General Public License.
10
11     See the README file in the top-level SPPARKS directory.
12 ----- */
13
14 #include "mpi.h"
15 #include "stdlib.h"
16 #include "string.h"
17 #include "output.h"
18 #include "memory.h"
19 #include "app.h"
20 #include "error.h"
21 #include "timer.h"
22 #include "diag_sinter_free_energy.h"
23 #include "app_lattice.h"
24 #include "app_sinter.h"
25 #include "comm_lattice.h"
26
27 using namespace SPPARKS_NS;
28
29 /* ----- */
30
31 DiagSinterFreeEnergy::DiagSinterFreeEnergy(SPPARKS *spk, int narg, char **arg
32 ) :
33   Diag(spk, narg, arg)
34 {
35   if (app->appclass != App::LATTICE)
36     error->all(FLERR, "Diag style incompatible with app style");
37 }
38 /* ----- */
39
40 void DiagSinterFreeEnergy::init()
41 {
42   appsinter = (AppSinter *) app;
43   nlocal = appsinter->nlocal;
44   interfacialFE = 0.0;
45 }
46
47 /* ----- */
48
49 void DiagSinterFreeEnergy::compute()
50 {
51   int *spin = appsinter->spin;
52   int *numneigh = appsinter->numneigh;
53   int **neighbor = appsinter->neighbor;
54
55   const int VACANT ( AppSinter::VACANT );
56
57   double interfacialFEtmp = 0.0;
58   for (int i = 0; i < nlocal; i++) {
59     int ispin = spin[i];
60     double surface = 0;

```

```

61
62 // If I am a grain site add the number of neighbors that are pore sites
63 if ( ispin > VACANT ) {
64     for (int j = 0; j < numneigh[i]; j++)
65         if (spin[neighbor[i][j]] == VACANT) surface++;
66     }
67     interfacialFEtmp += surface;
68 }
69 MPI_Allreduce(&interfacialFEtmp,&interfacialFE,1,MPI_DOUBLE,MPI_SUM,world);
70 }
71
72 /* ----- */
73
74 void DiagSinterFreeEnergy::stats(char *strtmp)
75 {
76     sprintf(strtmp, "%g", interfacialFE);
77 }
78
79 /* ----- */
80
81 void DiagSinterFreeEnergy::stats_header(char *strtmp)
82 {
83     sprintf(strtmp, "%10s", "InterfFE");
84 }

```

Listing 3 Source for the free surface energy calculation

```

1  /* -----
2  SPPARKS - Stochastic Parallel PARTicle Kinetic Simulator
3  http://www.cs.sandia.gov/~sjplimp/spparks.html
4  Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories
5
6  Copyright (2008) Sandia Corporation. Under the terms of Contract
7  DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
8  certain rights in this software. This software is distributed under
9  the GNU General Public License.
10
11  See the README file in the top-level SPPARKS directory.
12 ----- */
13
14 #ifndef DIAG_CLASS
15 DiagStyle(sinter_free_energy,DiagSinterFreeEnergy)
16
17 #else
18
19 #ifndef SPK_DIAG_SINTER_FREE_ENERGY_H
20 #define SPK_DIAG_SINTER_FREE_ENERGY_H
21
22 #include "stdio.h"
23 #include "diag.h"
24
25 namespace SPPARKS_NS {
26
27 class DiagSinterFreeEnergy : public Diag {
28
29 public:
30   DiagSinterFreeEnergy(class SPPARKS *, int, char **);
31   ~DiagSinterFreeEnergy() {}
32   void init();
33   void compute();
34   void stats(char *);
35   void stats_header(char *);
36
37 private:
38   class AppSinter *appsinter;
39   int nlocal;
40   double interfacialFE;
41 };
42
43 }
44
45 #endif
46
47 #endif

```

Listing 4 Header for the free surface energy calculation

Appendix C. Pore Free Energy Calculation

This appendix includes the source and header files for the subdomain's free surface energy diagnostic.

```

1  /* -----
2  SPPARKS - Stochastic Parallel PARTicle Kinetic Simulator
3  http://www.cs.sandia.gov/~sjplimp/spparks.html
4  Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories
5
6  Copyright (2008) Sandia Corporation. Under the terms of Contract
7  DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
8  certain rights in this software. This software is distributed under
9  the GNU General Public License.
10
11  See the README file in the top-level SPPARKS directory.
12 ----- */
13
14 #include "mpi.h"
15 #include "stdlib.h"
16 #include "string.h"
17 #include "output.h"
18 #include "memory.h"
19 #include "app.h"
20 #include "error.h"
21 #include "timer.h"
22 #include "diag_sinter_free_energy_pore.h"
23 #include "app_lattice.h"
24 #include "app_sinter.h"
25 #include "comm_lattice.h"
26 #include "domain.h"
27
28 using namespace SPPARKS_NS;
29
30 /* ----- */
31
32 DiagSinterFreeEnergyPore::DiagSinterFreeEnergyPore(SPPARKS *spk, int narg,
33     char **arg) :
34     Diag(spk, narg, arg)
35 {
36     if (app->appclass != App::LATTICE)
37         error->all(FLERR, "Diag style incompatible with app style");
38
39     if (narg > 2)
40         error->all(FLERR, "Too many arguments for diag style");
41
42     if (narg == 2) {
43         window = atof(arg[1]);
44         if (window >= 0.5)
45             error->all(FLERR, "Window size is too large (set to w < 0.5)");
46     }
47     else
48         window = 0.333333;
49 }
50 /* ----- */
51
52 void DiagSinterFreeEnergyPore::init()
53 {
54     appsinter = (AppSinter *) app;
55     nlocal = appsinter->nlocal;
56     interfacialFE = 0.0;
57
58     initialize_parameters_calculation();
59 }
60

```

```

61 /* ----- */
62
63 void DiagSinterFreeEnergyPore::compute()
64 {
65     const int VACANT ( AppSinter::VACANT );
66     double total_sites = 0;
67     int xgrid, ygrid, zgrid;
68
69     int *spin = appsinter->spin;
70     tagint *id = appsinter->id;
71     int *numneigh = appsinter->numneigh;
72     int **neighbor = appsinter->neighbor;
73
74     double interfacialFEtmp = 0.0;
75     for (int i = 0; i < nlocal; i++) {
76         appsinter->global_to_grid( id[i], xgrid, ygrid, zgrid );
77         if ( ((xgrid > xstart_ && xgrid < xend_) || nx==1) &&
78             ((ygrid > ystart_ && ygrid < yend_) || ny==1) &&
79             ((zgrid > zstart_ && zgrid < zend_) || nz==1) ) {
80
81             total_sites++;
82
83             int ispin = spin[i];
84
85             // If I am a grain site add the number of neighbors that are pore sites
86             if ( ispin > VACANT ) {
87                 double surface = 0;
88                 for (int j = 0; j < numneigh[i]; j++)
89                     if (spin[neighbor[i][j]] == VACANT) surface++;
90                 interfacialFEtmp += surface;
91             }
92         }
93     }
94
95     vector<double> local_info( 2 );
96     local_info[0] = interfacialFEtmp;
97     local_info[1] = total_sites;
98
99     vector<double> info_all( 2, 0 );
100     MPI_Allreduce(&local_info[0], &info_all[0], 3, MPI_DOUBLE, MPI_SUM, world);
101
102     interfacialFE = info_all[0] / info_all[1];
103 }
104
105 /* ----- */
106
107 void DiagSinterFreeEnergyPore::stats(char *strtmp)
108 {
109     sprintf(strtmp, " %10.6lf ", interfacialFE);
110 }
111
112 /* ----- */
113
114 void DiagSinterFreeEnergyPore::stats_header(char *strtmp)
115 {
116     sprintf(strtmp, " %10s ", "FE_psite");
117 }
118
119 /* ----- */
120 void DiagSinterFreeEnergyPore::initialize_parameters_calculation()
121 {
122     // num sites along each axis in lattice
123     nx = domain->nx;
124     ny = domain->ny;

```

```
125  nz = domain->nz;
126
127  xstart_ = static_cast<int>(nx*window);
128  xend_   = static_cast<int>(nx*(1-window));
129  ystart_ = static_cast<int>(ny*window);
130  yend_   = static_cast<int>(ny*(1-window));
131  zstart_ = static_cast<int>(nz*window);
132  zend_   = static_cast<int>(nz*(1-window));
133 }
```

Listing 5 Source for the subdomain's free surface energy calculation

```

1  /* -----
2  SPPARKS - Stochastic Parallel PARTicle Kinetic Simulator
3  http://www.cs.sandia.gov/~sjplimp/spparks.html
4  Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories
5
6  Copyright (2008) Sandia Corporation. Under the terms of Contract
7  DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
8  certain rights in this software. This software is distributed under
9  the GNU General Public License.
10
11  See the README file in the top-level SPPARKS directory.
12 ----- */
13
14 #ifndef DIAG_CLASS
15 DiagStyle(sinter_free_energy_pore,DiagSinterFreeEnergyPore)
16
17 #else
18
19 #ifndef SPK_DIAG_SINTER_FREE_ENERGY_PORE_H
20 #define SPK_DIAG_SINTER_FREE_ENERGY_PORE_H
21
22 #include "stdio.h"
23 #include "diag.h"
24
25 namespace SPPARKS_NS {
26
27 class DiagSinterFreeEnergyPore : public Diag {
28
29 public:
30   DiagSinterFreeEnergyPore(class SPPARKS *, int, char **);
31   ~DiagSinterFreeEnergyPore() {}
32   void init();
33   void compute();
34   void stats(char *);
35   void stats_header(char *);
36
37 protected:
38   void initialize_parameters_calculation();
39
40 private:
41   class AppSinter *appsinter;
42   int nlocal;
43   double interfacialFE, window;
44
45   int xstart_, xend_, nx;
46   int ystart_, yend_, ny;
47   int zstart_, zend_, nz;
48 };
49
50 }
51
52 #endif
53
54 #endif

```

Listing 6 Header for the subdomain's free surface energy calculation

Appendix D. Pore Curvature Calculation

This appendix includes the source and header files for the subdomain's pore curvature diagnostic.

```

1  /* -----
2     SPPARKS - Stochastic Parallel PARTicle Kinetic Simulator
3     http://www.cs.sandia.gov/~sjplimp/spparks.html
4     Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories
5
6     Copyright (2008) Sandia Corporation. Under the terms of Contract
7     DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
8     certain rights in this software. This software is distributed under
9     the GNU General Public License.
10
11     See the README file in the top-level SPPARKS directory.
12 ----- */
13
14 #include "mpi.h"
15 #include "domain.h"
16 #include "stdlib.h"
17 #include "string.h"
18 #include "output.h"
19 #include "lattice.h"
20 #include "memory.h"
21 #include "app.h"
22 #include "math.h"
23 #include "error.h"
24 #include "timer.h"
25 #include "diag_sinter_pore_curvature.h"
26 #include "app_lattice.h"
27 #include "app_sinter.h"
28 #include "comm_lattice.h"
29
30 using namespace SPPARKS_NS;
31
32 // same as in create_sites.cpp and diag_cluster.cpp and lattice.cpp
33 enum{NONE, LINE_2N, SQ_4N, SQ_8N, TRI, SC_6N, SC_26N, FCC, BCC, DIAMOND,
34      FCC_OCTA_TETRA, RANDOM_1D, RANDOM_2D, RANDOM_3D};
35
36 /* ----- */
37
38 DiagSinterPoreCurvature::DiagSinterPoreCurvature(SPPARKS *spk, int narg, char
39             **arg) :
40     Diag(spk, narg, arg)
41 {
42     if (app->appclass != App::LATTICE)
43         error->all(FLERR, "Diag style incompatible with app style");
44
45     if (narg>2)
46         error->all(FLERR, "Too many arguments for diag style");
47
48     if (narg==2) {
49         window = atof(arg[1]);
50         if (window>=0.5)
51             error->all(FLERR, "Window size is too large (set to w<0.5)");
52     }
53     else
54         window = 0.333333;
55 }
56 /* ----- */
57
58 void DiagSinterPoreCurvature::init()
59 {
60     appsinter = (AppSinter *) app;

```

```

61 poreCurvature = 0.0;
62 tripleJunction = 0.0;
63
64 dimension = domain->dimension;
65 int style = domain->lattice->style;
66
67 if (dimension==3 && style==SC_26N) {
68     // 12 edges, 3 neighbors per edge
69     memory->create(edgeneigh,12,3,"diagcurvature:edgeneigh");
70
71     edgeneigh[0][0] = 21; //(x+1, y, z)
72     edgeneigh[0][1] = 15; //( x,y+1, z)
73     edgeneigh[0][2] = 24; //(x+1,y+1, z)
74
75     edgeneigh[1][0] = 21; //(x+1, y, z)
76     edgeneigh[1][1] = 13; //( x, y,z+1)
77     edgeneigh[1][2] = 22; //(x+1, y,z+1)
78
79     edgeneigh[2][0] = 15; //( x,y+1, z)
80     edgeneigh[2][1] = 13; //( x, y,z+1)
81     edgeneigh[2][2] = 16; //( x,y+1,z+1)
82
83     edgeneigh[3][0] = 15; //( x,y+1, z)
84     edgeneigh[3][1] = 4; //(x-1, y, z)
85     edgeneigh[3][2] = 7; //(x-1,y+1, z)
86
87     edgeneigh[4][0] = 13; //( x, y,z+1)
88     edgeneigh[4][1] = 4; //(x-1, y, z)
89     edgeneigh[4][2] = 5; //(x-1, y,z+1)
90
91     edgeneigh[5][0] = 4; //(x-1, y, z)
92     edgeneigh[5][1] = 10; //( x,y-1, z)
93     edgeneigh[5][2] = 1; //(x-1,y-1, z)
94
95     edgeneigh[6][0] = 4; //(x-1, y, z)
96     edgeneigh[6][1] = 12; //( x, y,z-1)
97     edgeneigh[6][2] = 3; //(x-1, y,z-1)
98
99     edgeneigh[7][0] = 21; //(x+1, y, z)
100    edgeneigh[7][1] = 10; //( x,y-1, z)
101    edgeneigh[7][2] = 18; //(x+1,y-1, z)
102
103    edgeneigh[8][0] = 13; //( x, y,z+1)
104    edgeneigh[8][1] = 10; //( x,y-1, z)
105    edgeneigh[8][2] = 11; //( x,y-1,z+1)
106
107    edgeneigh[9][0] = 10; //( x,y-1, z)
108    edgeneigh[9][1] = 12; //( x, y,z-1)
109    edgeneigh[9][2] = 9; //( x,y-1,z-1)
110
111    edgeneigh[10][0] = 21; //(x+1, y, z)
112    edgeneigh[10][1] = 12; //( x, y,z-1)
113    edgeneigh[10][2] = 20; //(x+1, y,z-1)
114
115    edgeneigh[11][0] = 15; //( x,y+1, z)
116    edgeneigh[11][1] = 12; //( x, y,z-1)
117    edgeneigh[11][2] = 14; //( x,y+1,z-1)
118
119    memory->create(faceneigh,6,"diagcurvature:faceneigh");
120    faceneigh[0] = 21; //x+1
121    faceneigh[1] = 4; //x-1
122    faceneigh[2] = 15; //y+1
123    faceneigh[3] = 10; //y-1
124    faceneigh[4] = 13; //z+1

```

```

125     faceneigh[5] = 12; //z-1
126 }
127 else if (dimension==2 && style==SQ_8N) {
128     // 4 edges/vertices, 3 neighbors per edge/vertex
129     memory->create(edgeneigh,4,3,"diagcurvature:edgeneigh");
130
131     edgeneigh[0][0] = 6; //(x+1, y)
132     edgeneigh[0][1] = 4; //( x,y+1)
133     edgeneigh[0][2] = 7; //(x+1,y+1)
134
135     edgeneigh[1][0] = 1; //(x-1, y)
136     edgeneigh[1][1] = 4; //( x,y+1)
137     edgeneigh[1][2] = 2; //(x-1,y+1)
138
139     edgeneigh[2][0] = 6; //(x+1, y)
140     edgeneigh[2][1] = 3; //( x,y-1)
141     edgeneigh[2][2] = 5; //(x+1,y-1)
142
143     edgeneigh[3][0] = 1; //(x-1, y)
144     edgeneigh[3][1] = 3; //( x,y-1)
145     edgeneigh[3][2] = 0; //(x-1,y-1)
146
147     memory->create(faceneigh,4,"diagcurvature:faceneigh");
148     faceneigh[0] = 6; //x+1
149     faceneigh[1] = 1; //x-1
150     faceneigh[2] = 3; //y+1
151     faceneigh[3] = 4; //y-1
152 }
153 else
154     error->all(FLERR,"Lattice style not compatible with diagnostic");
155
156 initialize_parameters_calculation();
157 }
158
159 /* ----- */
160
161 void DiagSinterPoreCurvature::compute()
162 {
163     const int VACANT ( AppSinter::VACANT );
164     double pore_sites = 0, saTmp = 0;
165     int xgrid, ygrid, zgrid;
166
167     appsinter->comm->all();
168
169     int nlocal = appsinter->nlocal;
170
171     int *spin = appsinter->spin;
172     tagint *id = appsinter->id;
173     int *numneigh = appsinter->numneigh;
174     int **neighbor = appsinter->neighbor;
175
176     double innies = 0, outties = 0, exceptions = 0;
177     double tripleJunction_outties = 0, tripleJunction_sameface = 0;
178
179     int nedges = dimension*pow(2,dimension-1);
180
181     for (int i = 0; i < nlocal; i++) {
182         appsinter->global_to_grid( id[i], xgrid, ygrid, zgrid );
183         if ( ((xgrid > xstart_ && xgrid < xend_) || nx==1) &&
184             ((ygrid > ystart_ && ygrid < yend_) || ny==1) &&
185             ((zgrid > zstart_ && zgrid < zend_) || nz==1) ) {
186
187             int ispin = spin[i], nbor;
188             if ( ispin == VACANT ) { // If I am a pore site

```

```

189     pore_sites++;
190
191     for (int n=0; n<2*dimension; n++)
192         if (spin[neighbor[i][faceneigh[n]]] > VACANT) saTmp++;
193
194     for(nbor = 0; nbor < numneigh[i]; nbor++) { // check that it is pore
195     surface
196         int neigh = neighbor[i][nbor];
197         int value = spin[neigh];
198         if ( value > VACANT ) { // At least one of the neighbors is grain
199         site
200             break;
201         }
202     }
203     if ( nbor == numneigh[i] ) { // Previous cycle complete -> no grain
204     site found in the neighborhood
205         continue;
206     }
207     for ( int edge = 0; edge < nedges; edge++ ) {
208         int likeENEighs = 0;
209         for ( int enbor = 0; enbor < 3; enbor++ ) {
210             int eneigh = neighbor[i][edgeneigh[edge][enbor]];
211             if ( spin[eneigh] == VACANT ) { // Another pore site
212                 likeENEighs++;
213             }
214         }
215         if (likeENEighs == 0) {
216             outties++;
217             // Look for triple junction
218             if ( (spin[neighbor[i][edgeneigh[edge][0]]] != spin[neighbor[i][
219     edgeneigh[edge][1]]])
220                 || (spin[neighbor[i][edgeneigh[edge][1]]] != spin[neighbor[i]
221     ][edgeneigh[edge][2]])) ) {
222                 tripleJunction_outties++;
223             }
224         }
225         else if (likeENEighs == 2) innies++;
226         else if (likeENEighs == 1){
227             if (spin[neighbor[i][edgeneigh[edge][2]]] == VACANT)
228                 exceptions++;
229             else { // Look for triple junction
230                 if ( (spin[neighbor[i][edgeneigh[edge][0]]] != spin[neighbor[i]
231     ][edgeneigh[edge][1]]])
232                     && (spin[neighbor[i][edgeneigh[edge][1]]] != spin[neighbor[i]
233     ][edgeneigh[edge][2]])) ) {
234                 tripleJunction_sameface++;
235             }
236         }
237     }
238 }
239 }
240 // exceptions are not considered part of the curvature calculation
241 double poreCurvaturetmp = outties - tripleJunction_outties - innies / 3;
242 double total_tripleJunction = tripleJunction_outties +
243     tripleJunction_sameface / 2;
244
245 vector<double> local_info( 4 );
246 local_info[0] = poreCurvaturetmp;
247 local_info[1] = pore_sites;
248 local_info[2] = total_tripleJunction;
249 local_info[3] = saTmp;

```

```

245 vector<double> info_all( 4, 0 );
246
247 MPI_Allreduce(&local_info[0], &info_all[0], 4, MPI_DOUBLE, MPI_SUM, world);
248
249 poreCurvature = info_all[0] / info_all[1];
250 tripleJunction = info_all[2] / info_all[1];
251 surfaceArea = static_cast<int>(info_all[3]);
252 }
253
254 /* ----- */
255
256 void DiagSinterPoreCurvature::stats(char *strtmp)
257 {
258     sprintf(strtmp, " %10.6lf %10.6lf %10d ", poreCurvature, tripleJunction,
259             surfaceArea);
260 }
261 /* ----- */
262
263 void DiagSinterPoreCurvature::stats_header(char *strtmp)
264 {
265     sprintf(strtmp, " %10s %10s %10s ", "PCurvature", "TripleJunc", "SurfaceArea"
266             );
267 }
268 void DiagSinterPoreCurvature::initialize_parameters_calculation()
269 {
270     // num sites along each axis in lattice
271     nx = domain->nx;
272     ny = domain->ny;
273     nz = domain->nz;
274
275     xstart_ = static_cast<int>(nx*window);
276     xend_   = static_cast<int>(nx*(1-window));
277     ystart_ = static_cast<int>(ny*window);
278     yend_   = static_cast<int>(ny*(1-window));
279     zstart_ = static_cast<int>(nz*window);
280     zend_   = static_cast<int>(nz*(1-window));
281 }

```

Listing 7 Source for the pore curvature calculation

```

1  /* -----
2  SPPARKS - Stochastic Parallel PARTicle Kinetic Simulator
3  http://www.cs.sandia.gov/~sjplimp/spparks.html
4  Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories
5
6  Copyright (2008) Sandia Corporation. Under the terms of Contract
7  DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
8  certain rights in this software. This software is distributed under
9  the GNU General Public License.
10
11  See the README file in the top-level SPPARKS directory.
12 ----- */
13
14 #ifndef DIAG_CLASS
15 DiagStyle(sinter_pore_curvature,DiagSinterPoreCurvature)
16
17 #else
18
19 #ifndef SPK_DIAG_SINTER_PORE_CURVATURE_H
20 #define SPK_DIAG_SINTER_PORE_CURVATURE_H
21
22 #include "stdio.h"
23 #include "diag.h"
24
25 namespace SPPARKS_NS {
26
27 class DiagSinterPoreCurvature : public Diag {
28 public:
29   DiagSinterPoreCurvature(class SPPARKS *, int, char **);
30   ~DiagSinterPoreCurvature() {}
31   void init();
32   void compute();
33   void stats(char *);
34   void stats_header(char *);
35
36 private:
37   class AppSinter *appsinter;
38   double poreCurvature;
39   double tripleJunction;
40   double window;
41   int **edgeneigh, *faceneigh, surfaceArea;
42   void initialize_parameters_calculation();
43
44   int dimension;
45
46   int xstart_, xend_, nx;
47   int ystart_, yend_, ny;
48   int zstart_, zend_, nz;
49 };
50
51 }
52
53 #endif
54
55 #endif

```

Listing 8 Header for the pore curvature calculation

1 DEFENSE TECHNICAL
(PDF) INFORMATION CTR
DTIC OCA

1 DEVCOM ARL
(PDF) FCDD RLB CI
TECH LIB

10 DEVCOM ARL
(PDF) FCDD RLA A
A RAWLETT

FCDD RLA M
K BEHLER
V BLAIR
K CHO
C HOPPEL
FCDD RLA MB
E HERNANDEZ
D O'BRIEN
FCDD RLA ME
M GOLT
M GUZIEWSKI
L VARGAS-GONZALEZ

1 LANL
(PDF) C GARCIA-CARDONA

1 SNL
(PDF) J MITCHELL