ARL-MR-1090 • Dec 2023

DEVCOM ARMY RESEARCH LABORATORY

Diagnostics for SPPARKS' AppSinter Class

by Efraín Hernández–Rivera and Cristina García–Cardona

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.

ARL-MR-1090 ● DEC 2023

Diagnostics 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

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
December 2023	Memorandum Report	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	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Efraín Hernández
Unclassified	Unclassified	Unclassified	UU	31	19b. TELEPHONE NUMBER (Include area code) 410-278-6010

Contents

List of Figures

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]×[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.

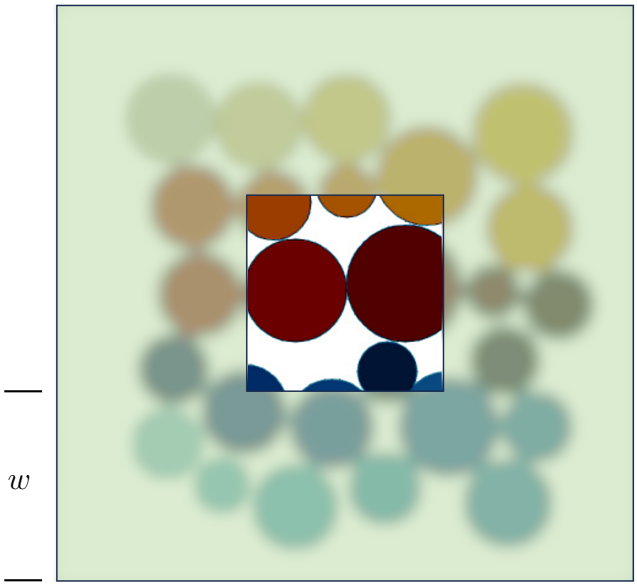

Fig. 1 Representation of the default subdomain for calculating the green body properties

The following classes are included in the release:

- `DiagSinterDensity` (Appendix A): calculate the relative density,
- `DiagSinterFreeEnergy` (Appendix B): calculate the intefacial 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 `edgeneigh` list, which defines the voxels that constitute the edge and face neighbors. The code defines the list depending on the simmulation dimension, as outlined in the `DiagSinterPoreCurvature::init` function (see Appendix D).

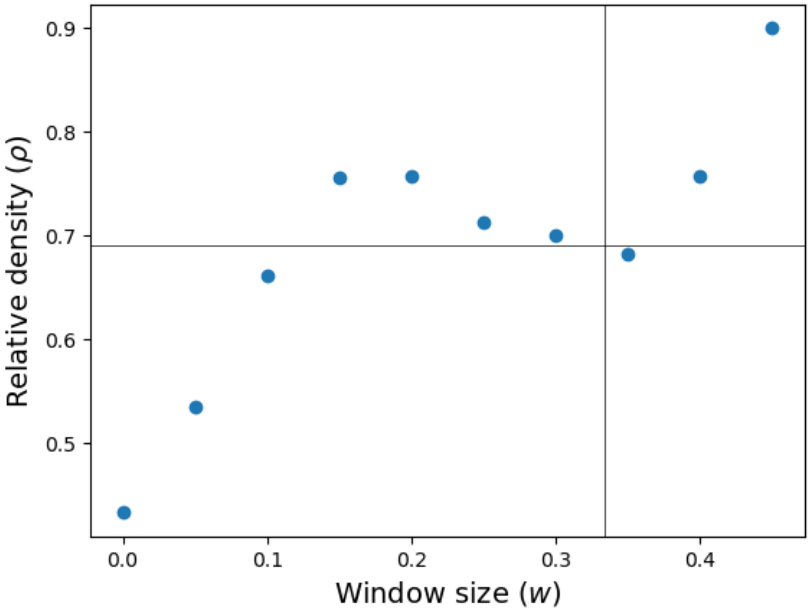

Fig. 2 Relative density as a function of window size

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.

```
/* ----------------------------------------------------------------------
   SPPARKS - Stochastic Parallel PARticle Kinetic Simulator
   http://www.cs.sandia.gov/~sjplimp/spparks.html
   Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories

   Copyright (2008) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level SPPARKS directory.
------------------------------------------------------------------------- */

#include "mpi.h"
#include "stdlib.h"
#include "string.h"
#include "output.h"
#include "memory.h"
#include "app.h"
#include "error.h"
#include "timer.h"
#include "diag_sinter_density.h"
#include "app_lattice.h"
#include "app_sinter.h"
#include "comm_lattice.h"
#include "domain.h"

using namespace SPPARKS_NS;

/* ---------------------------------------------------------------------- */

DiagSinterDensity::DiagSinterDensity(SPPARKS *spk, int narg, char **arg) :
  Diag(spk,narg,arg)
{
  if (app->appclass != App::LATTICE)
    error->all(FLERR,"Diag style incompatible with app style");

  if (narg>2)
    error->all(FLERR,"Too many arguments for diag style");

  if (narg==2) {
    window = atof(arg[1]);
    if (window>=0.5)
      error->all(FLERR,"Window size is too large (set to w<0.5)");
  }
  else
    window = 0.333333;
}

/* ---------------------------------------------------------------------- */

void DiagSinterDensity::init()
{
  appsinter = (AppSinter *) app;
  nlocal = appsinter->nlocal;
  density = 0.0;

  initialize_parameters_density_calculation();
}

/* ---------------------------------------------------------------------- */

```

```
void DiagSinterDensity::compute()
{
  double grain_sites = 0;
  double total_sites = 0;
  int xgrid, ygrid, zgrid;

  int *spin = appsinter->spin;
  tagint *id = appsinter->id;
  const int VACANT ( AppSinter::VACANT );

  for ( int i = 0; i < nlocal; i++ ) {
    appsinter->global_to_grid( id[i], xgrid, ygrid, zgrid );
    if (  ((xgrid > xstart_density && xgrid < xend_density) || nx==1) &&
        ((ygrid > ystart_density && ygrid < yend_density) || ny==1) &&
        ((zgrid > zstart_density && zgrid < zend_density) || nz==1)) {
      total_sites++;
      if ( spin[i] > VACANT ) grain_sites++;
    }
  }

  double local_density_info[2];
  local_density_info[0] = grain_sites;
  local_density_info[1] = total_sites;

  double global_density_info[2] = {0,0};
  MPI_Allreduce(local_density_info, global_density_info, 2, MPI_DOUBLE,
    MPI_SUM, world);

  density = global_density_info[0] / global_density_info[1];
}

/* ---------------------------------------------------------------------- */

void DiagSinterDensity::stats(char *strtmp)
{
  sprintf(strtmp," %10.6lf",density);
}

/* ---------------------------------------------------------------------- */

void DiagSinterDensity::stats_header(char *strtmp)
{
  sprintf(strtmp," %10s","Density");
}

/* ---------------------------------------------------------------------- */

void DiagSinterDensity::initialize_parameters_density_calculation()
{
  // num sites along each axis in lattice
  nx = domain->nx;
  ny = domain->ny;
  nz = domain->nz;

  xstart_density = static_cast<int>(nx*window);
  xend_density   = static_cast<int>(nx*(1-window));
  ystart_density = static_cast<int>(ny*window);
  yend_density   = static_cast<int>(ny*(1-window));
  zstart_density = static_cast<int>(nz*window);
  zend_density   = static_cast<int>(nz*(1-window));
}
```

Listing 1 Source for the density calculation

```
/* ----------------------------------------------------------------------
   SPPARKS - Stochastic Parallel PARticle Kinetic Simulator
   http://www.cs.sandia.gov/~sjplimp/spparks.html
   Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories

   Copyright (2008) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level SPPARKS directory.
------------------------------------------------------------------------- */

#ifdef DIAG_CLASS
DiagStyle(sinter_density,DiagSinterDensity)

#else

#ifndef SPK_DIAG_SINTER_DENSITY_H
#define SPK_DIAG_SINTER_DENSITY_H

#include "stdio.h"
#include "diag.h"

namespace SPPARKS_NS {

class DiagSinterDensity : public Diag {

 public:
  DiagSinterDensity(class SPPARKS *, int, char **);
  ~DiagSinterDensity() {}
  void init();
  void compute();
  void stats(char *);
  void stats_header(char *);

 protected:
  void initialize_parameters_density_calculation();

 private:
  class AppSinter *appsinter;
  int nlocal;
  double density, window;

  int xstart_density, xend_density, nx;
  int ystart_density, yend_density, ny;
  int zstart_density, zend_density, nz;
};

}

#endif

#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.

```
/* ----------------------------------------------------------------------
   SPPARKS - Stochastic Parallel PARticle Kinetic Simulator
   http://www.cs.sandia.gov/~sjplimp/spparks.html
   Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories

   Copyright (2008) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level SPPARKS directory.
------------------------------------------------------------------------- */

#include "mpi.h"
#include "stdlib.h"
#include "string.h"
#include "output.h"
#include "memory.h"
#include "app.h"
#include "error.h"
#include "timer.h"
#include "diag_sinter_free_energy.h"
#include "app_lattice.h"
#include "app_sinter.h"
#include "comm_lattice.h"

using namespace SPPARKS_NS;

/* ---------------------------------------------------------------------- */

DiagSinterFreeEnergy::DiagSinterFreeEnergy(SPPARKS *spk, int narg, char **arg
    ) :
  Diag(spk,narg,arg)
{
  if (app->appclass != App::LATTICE)
    error->all(FLERR,"Diag style incompatible with app style");
}

/* ---------------------------------------------------------------------- */

void DiagSinterFreeEnergy::init()
{
  appsinter = (AppSinter *) app;
  nlocal = appsinter->nlocal;
  interfacialFE = 0.0;
}

/* ---------------------------------------------------------------------- */

void DiagSinterFreeEnergy::compute()
{
  int *spin = appsinter->spin;
  int *numneigh = appsinter->numneigh;
  int **neighbor = appsinter->neighbor;

  const int VACANT ( AppSinter::VACANT );

  double interfacialFEtmp = 0.0;
  for (int i = 0; i < nlocal; i++) {
    int ispin = spin[i];
    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

```
/* ----------------------------------------------------------------------
   SPPARKS - Stochastic Parallel PARticle Kinetic Simulator
   http://www.cs.sandia.gov/~sjplimp/spparks.html
   Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories

   Copyright (2008) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level SPPARKS directory.
------------------------------------------------------------------------- */

#ifdef DIAG_CLASS
DiagStyle(sinter_free_energy,DiagSinterFreeEnergy)

#else

#ifndef SPK_DIAG_SINTER_FREE_ENERGY_H
#define SPK_DIAG_SINTER_FREE_ENERGY_H

#include "stdio.h"
#include "diag.h"

namespace SPPARKS_NS {

class DiagSinterFreeEnergy : public Diag {

 public:
  DiagSinterFreeEnergy(class SPPARKS *, int, char **);
  ~DiagSinterFreeEnergy() {}
  void init();
  void compute();
  void stats(char *);
  void stats_header(char *);

 private:
  class AppSinter *appsinter;
  int nlocal;
  double interfacialFE;
};

}

#endif

#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.

```
/* ----------------------------------------------------------------------
   SPPARKS - Stochastic Parallel PARticle Kinetic Simulator
   http://www.cs.sandia.gov/~sjplimp/spparks.html
   Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories

   Copyright (2008) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level SPPARKS directory.
------------------------------------------------------------------------- */

#include "mpi.h"
#include "stdlib.h"
#include "string.h"
#include "output.h"
#include "memory.h"
#include "app.h"
#include "error.h"
#include "timer.h"
#include "diag_sinter_free_energy_pore.h"
#include "app_lattice.h"
#include "app_sinter.h"
#include "comm_lattice.h"
#include "domain.h"

using namespace SPPARKS_NS;

/* ---------------------------------------------------------------------- */

DiagSinterFreeEnergyPore::DiagSinterFreeEnergyPore(SPPARKS *spk, int narg,
    char **arg) :
  Diag(spk,narg,arg)
{
  if (app->appclass != App::LATTICE)
    error->all(FLERR,"Diag style incompatible with app style");

  if (narg>2)
    error->all(FLERR,"Too many arguments for diag style");

  if (narg==2) {
    window = atof(arg[1]);
    if (window>=0.5)
      error->all(FLERR,"Window size is too large (set to w<0.5)");
  }
  else
    window = 0.333333;
}

/* ---------------------------------------------------------------------- */

void DiagSinterFreeEnergyPore::init()
{
  appsinter = (AppSinter *) app;
  nlocal = appsinter->nlocal;
  interfacialFE = 0.0;

  initialize_parameters_calculation();
}

```

```
/* ---------------------------------------------------------------------- */

void DiagSinterFreeEnergyPore::compute()
{
  const int VACANT ( AppSinter::VACANT );
  double total_sites = 0;
  int xgrid, ygrid, zgrid;

  int *spin = appsinter->spin;
  tagint *id = appsinter->id;
  int *numneigh = appsinter->numneigh;
  int **neighbor = appsinter->neighbor;

  double interfacialFEtmp = 0.0;
  for (int i = 0; i < nlocal; i++) {
    appsinter->global_to_grid( id[i], xgrid, ygrid, zgrid );
    if ( ((xgrid > xstart_ && xgrid < xend_) || nx==1) &&
         ((ygrid > ystart_ && ygrid < yend_) || ny==1) &&
         ((zgrid > zstart_ && zgrid < zend_) || nz==1) ) {

      total_sites++;

      int ispin = spin[i];

      // If I am a grain site add the number of neighbors that are pore sites
      if ( ispin > VACANT ) {
        double surface = 0;
        for (int j = 0; j < numneigh[i]; j++)
          if (spin[neighbor[i][j]] == VACANT) surface++;
        interfacialFEtmp += surface;
      }
    }
  }

  vector<double> local_info( 2 );
  local_info[0] = interfacialFEtmp;
  local_info[1] = total_sites;

  vector<double> info_all( 2, 0 );
  MPI_Allreduce(&local_info[0], &info_all[0], 3, MPI_DOUBLE, MPI_SUM, world);

  interfacialFE = info_all[0] / info_all[1];
}

/* ---------------------------------------------------------------------- */

void DiagSinterFreeEnergyPore::stats(char *strtmp)
{
  sprintf(strtmp," %10.6lf ",interfacialFE);
}

/* ---------------------------------------------------------------------- */

void DiagSinterFreeEnergyPore::stats_header(char *strtmp)
{
  sprintf(strtmp," %10s ","FE_psite");
}

/* ---------------------------------------------------------------------- */
void DiagSinterFreeEnergyPore::initialize_parameters_calculation()
{
  // num sites along each axis in lattice
  nx = domain->nx;
  ny = domain->ny;
```

```
  nz = domain->nz;

  xstart_ = static_cast<int>(nx*window);
  xend_   = static_cast<int>(nx*(1-window));
  ystart_ = static_cast<int>(ny*window);
  yend_   = static_cast<int>(ny*(1-window));
  zstart_ = static_cast<int>(nz*window);
  zend_   = static_cast<int>(nz*(1-window));
}
```

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

```
/* ----------------------------------------------------------------------
   SPPARKS - Stochastic Parallel PARticle Kinetic Simulator
   http://www.cs.sandia.gov/~sjplimp/spparks.html
   Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories

   Copyright (2008) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level SPPARKS directory.
------------------------------------------------------------------------- */

#ifdef DIAG_CLASS
DiagStyle(sinter_free_energy_pore,DiagSinterFreeEnergyPore)

#else

#ifndef SPK_DIAG_SINTER_FREE_ENERGY_PORE_H
#define SPK_DIAG_SINTER_FREE_ENERGY_PORE_H

#include "stdio.h"
#include "diag.h"

namespace SPPARKS_NS {

class DiagSinterFreeEnergyPore : public Diag {

 public:
  DiagSinterFreeEnergyPore(class SPPARKS *, int, char **);
  ~DiagSinterFreeEnergyPore() {}
  void init();
  void compute();
  void stats(char *);
  void stats_header(char *);

protected:
  void initialize_parameters_calculation();

 private:
  class AppSinter *appsinter;
  int nlocal;
  double interfacialFE, window;

  int xstart_, xend_, nx;
  int ystart_, yend_, ny;
  int zstart_, zend_, nz;
};

}

#endif

#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.

```
/* ----------------------------------------------------------------------
   SPPARKS - Stochastic Parallel PARticle Kinetic Simulator
   http://www.cs.sandia.gov/~sjplimp/spparks.html
   Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories

   Copyright (2008) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level SPPARKS directory.
------------------------------------------------------------------------- */

#include "mpi.h"
#include "domain.h"
#include "stdlib.h"
#include "string.h"
#include "output.h"
#include "lattice.h"
#include "memory.h"
#include "app.h"
#include "math.h"
#include "error.h"
#include "timer.h"
#include "diag_sinter_pore_curvature.h"
#include "app_lattice.h"
#include "app_sinter.h"
#include "comm_lattice.h"

using namespace SPPARKS_NS;

// same as in create_sites.cpp and diag_cluster.cpp and lattice.cpp
enum{NONE,LINE_2N,SQ_4N,SQ_8N,TRI,SC_6N,SC_26N,FCC,BCC,DIAMOND,
  FCC_OCTA_TETRA,RANDOM_1D,RANDOM_2D,RANDOM_3D};

/* ---------------------------------------------------------------------- */

DiagSinterPoreCurvature::DiagSinterPoreCurvature(SPPARKS *spk, int narg, char
     **arg) :
  Diag(spk,narg,arg)
{
  if (app->appclass != App::LATTICE)
    error->all(FLERR,"Diag style incompatible with app style");

  if (narg>2)
    error->all(FLERR,"Too many arguments for diag style");

  if (narg==2) {
    window = atof(arg[1]);
    if (window>=0.5)
      error->all(FLERR,"Window size is too large (set to w<0.5)");
  }
  else
    window = 0.333333;
}

/* ---------------------------------------------------------------------- */

void DiagSinterPoreCurvature::init()
{
  appsinter = (AppSinter *) app;
```

```
  poreCurvature = 0.0;
  tripleJunction = 0.0;

  dimension = domain->dimension;
  int style = domain->lattice->style;

  if (dimension==3 && style==SC_26N) {
    // 12 edges, 3 neighbors per edge
    memory->create(edgeneigh,12,3,"diagcurvature:edgeneigh");

    edgeneigh[0][0] = 21; //(x+1,  y,  z)
    edgeneigh[0][1] = 15; //(  x,y+1,  z)
    edgeneigh[0][2] = 24; //(x+1,y+1,  z)

    edgeneigh[1][0] = 21; //(x+1,  y,  z)
    edgeneigh[1][1] = 13; //(  x,  y,z+1)
    edgeneigh[1][2] = 22; //(x+1,  y,z+1)

    edgeneigh[2][0] = 15; //(  x,y+1,  z)
    edgeneigh[2][1] = 13; //(  x,  y,z+1)
    edgeneigh[2][2] = 16; //(  x,y+1,z+1)

    edgeneigh[3][0] = 15; //(  x,y+1,  z)
    edgeneigh[3][1] = 4;  //(x-1,  y,  z)
    edgeneigh[3][2] = 7;  //(x-1,y+1,  z)

    edgeneigh[4][0] = 13; //(  x,  y,z+1)
    edgeneigh[4][1] = 4;  //(x-1,  y,  z)
    edgeneigh[4][2] = 5;  //(x-1,  y,z+1)

    edgeneigh[5][0] = 4;  //(x-1,  y,  z)
    edgeneigh[5][1] = 10; //(  x,y-1,  z)
    edgeneigh[5][2] = 1;  //(x-1,y-1,  z)

    edgeneigh[6][0] = 4;  //(x-1,  y,  z)
    edgeneigh[6][1] = 12; //(  x,  y,z-1)
    edgeneigh[6][2] = 3;  //(x-1,  y,z-1)

    edgeneigh[7][0] = 21; //(x+1,  y,  z)
    edgeneigh[7][1] = 10; //(  x,y-1,  z)
    edgeneigh[7][2] = 18; //(x+1,y-1,  z)

    edgeneigh[8][0] = 13; //(  x,  y,z+1)
    edgeneigh[8][1] = 10; //(  x,y-1,  z)
    edgeneigh[8][2] = 11; //(  x,y-1,z+1)

    edgeneigh[9][0] = 10; //(  x,y-1,  z)
    edgeneigh[9][1] = 12; //(  x,  y,z-1)
    edgeneigh[9][2] = 9;  //(  x,y-1,z-1)

    edgeneigh[10][0] = 21; //(x+1,  y,  z)
    edgeneigh[10][1] = 12; //(  x,  y,z-1)
    edgeneigh[10][2] = 20; //(x+1,  y,z-1)

    edgeneigh[11][0] = 15; //(  x,y+1,  z)
    edgeneigh[11][1] = 12; //(  x,  y,z-1)
    edgeneigh[11][2] = 14; //(  x,y+1,z-1)

    memory->create(faceneigh,6,"diagcurvature:faceneigh");
    faceneigh[0] = 21; //x+1
    faceneigh[1] =  4; //x-1
    faceneigh[2] = 15; //y+1
    faceneigh[3] = 10; //y-1
    faceneigh[4] = 13; //z+1
```

```
    faceneigh[5] = 12; //z-1
  }
  else if (dimension==2 && style==SQ_8N) {
    // 4 edges/vertices, 3 neighbors per edge/vertex
    memory->create(edgeneigh,4,3,"diagcurvature:edgeneigh");

    edgeneigh[0][0] = 6; //(x+1,  y)
    edgeneigh[0][1] = 4; //(  x,y+1)
    edgeneigh[0][2] = 7; //(x+1,y+1)

    edgeneigh[1][0] = 1; //(x-1,  y)
    edgeneigh[1][1] = 4; //(  x,y+1)
    edgeneigh[1][2] = 2; //(x-1,y+1)

    edgeneigh[2][0] = 6; //(x+1,  y)
    edgeneigh[2][1] = 3; //(  x,y-1)
    edgeneigh[2][2] = 5; //(x+1,y-1)

    edgeneigh[3][0] = 1; //(x-1,  y)
    edgeneigh[3][1] = 3; //(  x,y-1)
    edgeneigh[3][2] = 0; //(x-1,y-1)

    memory->create(faceneigh,4,"diagcurvature:faceneigh");
    faceneigh[0] = 6; //x+1
    faceneigh[1] = 1; //x-1
    faceneigh[2] = 3; //y+1
    faceneigh[3] = 4; //y-1
  }
  else
    error->all(FLERR,"Lattice style not compatible with diagnostic");

  initialize_parameters_calculation();
}

/* ---------------------------------------------------------------------- */

void DiagSinterPoreCurvature::compute()
{
  const int VACANT ( AppSinter::VACANT );
  double pore_sites = 0, saTmp = 0;
  int xgrid, ygrid, zgrid;

  appsinter->comm->all();

  int nlocal = appsinter->nlocal;

  int *spin = appsinter->spin;
  tagint *id = appsinter->id;
  int *numneigh = appsinter->numneigh;
  int **neighbor = appsinter->neighbor;

  double innies = 0, outties = 0, exceptions = 0;
  double tripleJunction_outties = 0, tripleJunction_sameface = 0;

  int nedges = dimension*pow(2,dimension-1);

  for (int i = 0; i < nlocal; i++) {
    appsinter->global_to_grid( id[i], xgrid, ygrid, zgrid );
    if ( ((xgrid > xstart_ && xgrid < xend_) || nx==1) &&
         ((ygrid > ystart_ && ygrid < yend_) || ny==1) &&
         ((zgrid > zstart_ && zgrid < zend_) || nz==1) ) {

      int ispin = spin[i], nbor;
      if ( ispin == VACANT ) { // If I am a pore site
```

```
      pore_sites++;

      for (int n=0; n<2*dimension; n++)
        if (spin[neighbor[i][faceneigh[n]]] > VACANT) saTmp++;

      for(nbor = 0; nbor < numneigh[i]; nbor++) { // check that it is pore surface
        int neigh = neighbor[i][nbor];
        int value = spin[neigh];
        if ( value > VACANT ) { // At least one of the neighbors is grain site
          break;
        }
      }
      if ( nbor == numneigh[i] ) { // Previous cycle complete -> no grain site found in the neighborhood
        continue;
      }
      for ( int edge = 0; edge < nedges; edge++ ) {
        int likeENeighs = 0;
        for ( int enbor = 0; enbor < 3; enbor++ ) {
          int eneigh = neighbor[i][edgeneigh[edge][enbor]];
          if ( spin[eneigh] == VACANT ) { // Another pore site
            likeENeighs++;
          }
        }
        if (likeENeighs == 0) {
          outties++;
          // Look for triple junction
          if ( (spin[neighbor[i][edgeneigh[edge][0]]] != spin[neighbor[i][edgeneigh[edge][1]]])
              || (spin[neighbor[i][edgeneigh[edge][1]]] != spin[neighbor[i][edgeneigh[edge][2]]]) ) {
            tripleJunction_outties++;
          }
        }
        else if(likeENeighs == 2) innies++;
        else if(likeENeighs == 1){
          if(spin[neighbor[i][edgeneigh[edge][2]]] == VACANT)
            exceptions++;
          else { // Look for triple junction
            if ( (spin[neighbor[i][edgeneigh[edge][0]]] != spin[neighbor[i][edgeneigh[edge][1]]])
                && (spin[neighbor[i][edgeneigh[edge][1]]] != spin[neighbor[i][edgeneigh[edge][2]]]) ) {
              tripleJunction_sameface++;
            }
          }
        }
      }
    }
  }
}
// exceptions are not considered part of the curvature calculation
double poreCurvaturetmp = outties - tripleJunction_outties - innies / 3;
double total_tripleJunction = tripleJunction_outties +
  tripleJunction_sameface / 2;

vector<double> local_info( 4 );
local_info[0] = poreCurvaturetmp;
local_info[1] = pore_sites;
local_info[2] = total_tripleJunction;
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,
        surfaceArea);
259 }
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

```
/* ----------------------------------------------------------------------
   SPPARKS - Stochastic Parallel PARticle Kinetic Simulator
   http://www.cs.sandia.gov/~sjplimp/spparks.html
   Steve Plimpton, sjplimp@sandia.gov, Sandia National Laboratories

   Copyright (2008) Sandia Corporation.  Under the terms of Contract
   DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains
   certain rights in this software.  This software is distributed under
   the GNU General Public License.

   See the README file in the top-level SPPARKS directory.
------------------------------------------------------------------------- */

#ifdef DIAG_CLASS
DiagStyle(sinter_pore_curvature,DiagSinterPoreCurvature)

#else

#ifndef SPK_DIAG_SINTER_PORE_CURVATURE_H
#define SPK_DIAG_SINTER_PORE_CURVATURE_H

#include "stdio.h"
#include "diag.h"

namespace SPPARKS_NS {

class DiagSinterPoreCurvature : public Diag {
 public:
  DiagSinterPoreCurvature(class SPPARKS *, int, char **);
  ~DiagSinterPoreCurvature() {}
  void init();
  void compute();
  void stats(char *);
  void stats_header(char *);

 private:
  class AppSinter *appsinter;
  double poreCurvature;
  double tripleJunction;
  double window;
  int **edgeneigh, *faceneigh, surfaceArea;
  void initialize_parameters_calculation();

  int dimension;

  int xstart_, xend_, nx;
  int ystart_, yend_, ny;
  int zstart_, zend_, nz;
};

}

#endif

#endif
```

Listing 8 Header for the pore curvature calculation

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR
DTIC OCA

1 (PDF) DEVCOM ARL
FCDD RLB CI
TECH LIB

10 (PDF) DEVCOM ARL

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 (PDF) LANL
C GARCIA-CARDONA

1 (PDF) SNL
J MITCHELL