SPPARKS for microstructural science

Brian DeCost

11 July 2014

```
#include "string.h"
#include "spparks.h"
#include "memory.h"
#include "error.h"
#include "universe.h"
#include "input.h"
#in
#in
#in
#include "potential.h"
#include "random_mars.h"
#include "timer.h"
#include "output.h"

using namespace SPPARKS_NS;

/* ----------------------------------------------------------------------
   allocate fundamental classes (memory, error, universe, input)
   parse input switches
   initialize communicators, screen & logfile output
   input is allocated at end after MPI info is setup
------------------------------------------------------------------------- */

SPPARKS::SPPARKS(int narg, char **arg, MPI_Comm communicator)
{
  error = new Error(this);
```

Stochastic Parallel PARticle Kinetic Simulator[1]

SPPARKS is an efficient *parallel* Kinetic Monte Carlo framework

- Mesoscale materials phenomena as stochastic processes
- Scalable to thousands of processors
- Modular C++ facilitates user-defined models/statistics
- Open Source (GPLv2)
- Related to the LAMMPS Molecular Dynamics framework

[1] http://spparks.sandia.gov/

How SPPARKS is used

SPPARKS is a command-line application. Simulations are defined by input scripts composed of SPPARKS commands

The most important commands are:

- app_style
- solve_style / sweep_style
- diag_style

What applications is SPPARKS useful for?

lattice-based models

- Ising model
- Potts model and variants (pin, strain, *aniso*)
- Vacancy diffusion models (surface and bulk)

off-lattice models

- Atomic relaxation

generic KMC applications

- biochemical reaction network

In this session, we'll focus on the Potts model

- Modeling grain growth, recrystallization, and related phenomena
- Annealing synthetic microstructures to achieve local equilibrium along triple lines
- Generation of simple synthetic microstructures

Modeling abnormal grain growth with SPPARKS

Variable grain boundary mobility: imaginary red/blue texture components

The SPPARKS distribution

Download SPPARKS from
`http://www.sandia.gov/~sjplimp/download.html`

Unpack the SPPARKS source archive

```
mkdir -p ~/Software && cd ~/Software
tar xvzf ~/Downloads/spparks.tar.gz
mv spparks-19Nov13 spparks
cd spparks/src
ls ~/Software/spparks/
```

```
brian@eeepc:spparks-19Nov13 $ ls ~/Software/spparks-19Nov13
doc  examples  LICENSE  python  README  src  tools
```

SPPARKS uses the Unix build system make

SPPARKS is distributed with several makefiles, found in src/MAKE/.

```
bdecost@unix6 > ls MAKE                              :~/Software/spparks-19Nov13/src
Makefile.blackrose  Makefile.mac        Makefile.serial
Makefile.cygwin     Makefile.mac_debug  Makefile.serial_backup
Makefile.g++        Makefile.mac_mpi    Makefile.serial_debug
Makefile.linux      Makefile.redsky     Makefile.xt5
```

You will likely have to edit Makefile.linux, Makefile.g++, or Makefile.mac_mpi to reflect your system to get SPPARKS to compile.[2]

[2] You'll likely need to supply the location of the MPI library files on your system.

Building SPPARKS

Try to compile SPPARKS by running make with the appropriate build target for your system as an argument.[3]

Building SPPARKS using Makefile.demo

```
make demo
```

The result is an executable named spk_demo in src/.

[3] To use the demo makefiles, first copy Makefile.demo and Makefile.demo_serial are in the SPPARKS src/MAKE directory.

Building serial SPPARKS

SPPARKS comes with a placeholder MPI library for building a serial version.

First build the MPI stubs:

```
cd STUBS
make
cd ..
```

Build the demo_serial version of SPPARKS

```
make demo_serial
```

"Installing" SPPARKS[4]

To avoid copying the SPPARKS executable or typing the full path to it repeatedly, define a *shell alias*:

```
alias spk=~/Software/spparks/src/spk_demo_serial
```

Now typing `spk` into the shell prompt from any working directory runs SPPARKS. Try this now.

[4] To make this persistent, append this command to `~/.bashrc` (`~/.bash_profile` on a mac)

Running SPPARKS with no input

SPPARKS reads commands from stdin

```
brian@rsfern > spk
SPPARKS (19 Nov 2013)
```

Press Ctrl-C to exit.

Demo: SPPARKS input scripts

SPPARKS is controlled by a script of commands which are executed line-by-line.

Run the basic demo

```
cd ~/Downloads/spparks-demo-master/demos/basic
spk < basic.spkin
```

SPPARKS will print some diagnostic information to the screen, and to a log file called `log.spparks`.

SPPARKS output


```
brian@rsfern > spk < basic.spkin
SPPARKS (19 Nov 2013)
Created box = (0 0 0) to (32 32 32)
  1 by 1 by 1 processor grid
Creating sites ...
  32768 sites
  32768 sites have 26 neighbors
Setting site values ...
  32768 settings made for site
Setting up run ...
      Time    Naccept    Nreject    Nsweeps         CPU     Energy
         0          0          0          0           0     843324
   10.0008     166567          0          0        6.46     301516
   20.0003     187372          0          0        6.93     243102
   30.0003     201713          0          0        7.19     206800
   40.0001     213353          0          0        7.39     179268
   50.0001     222915          0          0        7.58     159700
   60.0033     231650          0          0        7.74     141190
   70.0009     238089          0          0        7.85     130340
   80.0003     243626          0          0        7.95     124082
   90.0005     248183          0          0        8.02     119936
       100     252342          0          0        8.09     116864
Loop time of 8.09225 on 1 procs

Solve time (%) = 0.133569 (1.65058)
Update time (%) = 0 (0)
Comm  time (%) = 0 (0)
Outpt time (%) = 0.0541012 (0.668556)
App   time (%) = 7.87678 (97.3373)
Other time (%) = 0.0278049 (0.343599)
```

Anatomy of an input script: `basic.spkin`


```
# Minimal SPPARKS Potts script
seed                56789
app_style           potts 100
dimension           3
lattice             sc/26n 1.0
region              box block 0 32 0 32 0 32
create_box          box
create_sites        box
set                 site range 1 100
solve_style         tree
temperature         0.0
diag_style          energy
stats               10.0
run                 100.0
```

Demo: Grain growth kinetics

Run the grain-growth-kinetics demo

```
cd ../grain-growth-kinetics
spk < potts2d.spkin
```

This script uses the `cluster` diagnostic, which computes grain size information.

Grain growth kinetics

This demo includes a `python` script which extracts grain growth kinetics from `log.spparks` and attempts to plot them.

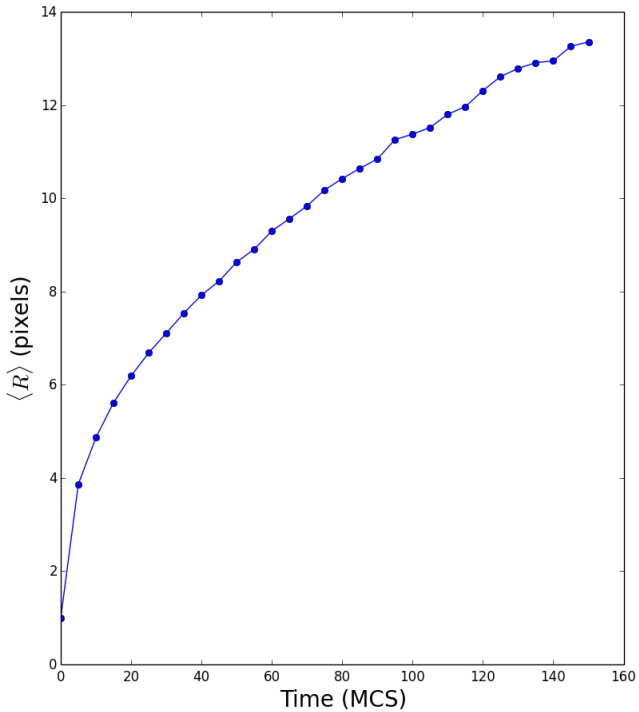
Run the python script

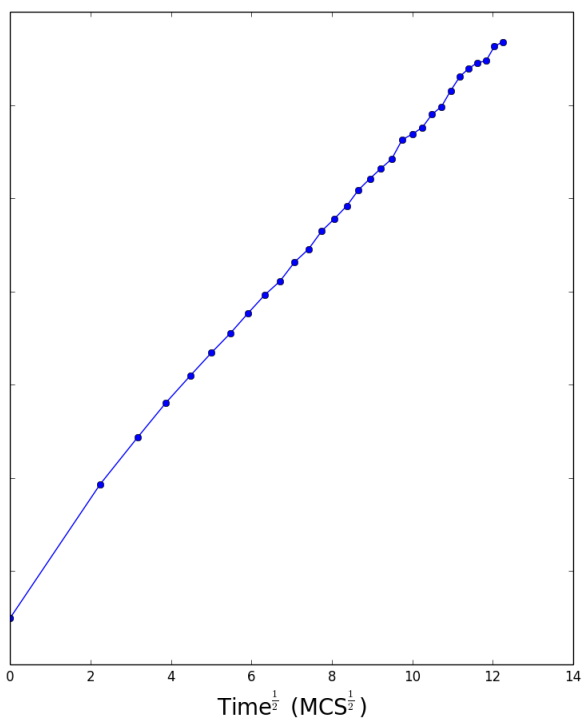
```
python parse_log.py log.spparks
```

The script produces a file `cluster-stats.csv` that you can open in `Excel`.

Grain growth kinetics

Grain size (after an initial transient) is linear in `sqrt`-time!





Visualization: the dump command

Run the visualization demo

```
cd ../visualization
spk < viz.spkin
```

SPPARKS has a similar data dump format to LAMMPS

```
dump 1.0 text potts.dump
```

These need to be post-processed `pizza.py` or custom scripts.[5]

[5] pizza.py is a post-processing toolkit for LAMMPS
http://pizza.sandia.gov/

Conversion to vtk with pizza.py

The python script pizza_dump2vtk.py uses tools from the open source pizza.py toolkit to create vtk files from the SPPARKS many-timestep dump file.

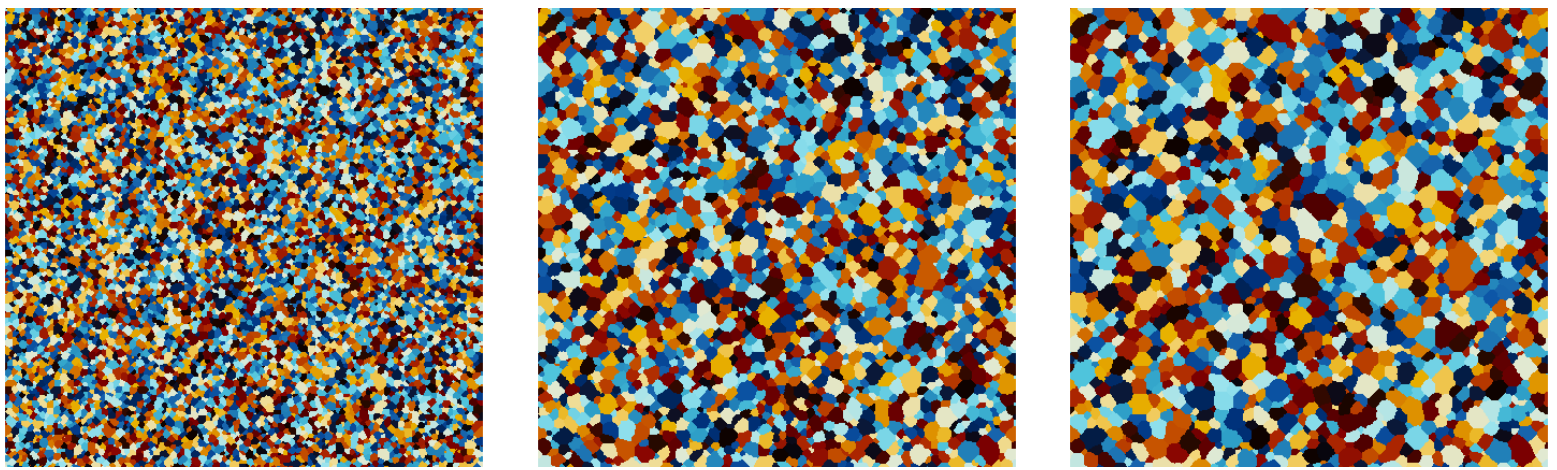
Run pizza_dump2vtk.py

```
python pizza_dump2vtk.py potts.dump
```

The viz3D.spkin script will make a 3D grain growth dataset if you're willing to wait for it.

Visualizing with ParaView

Open the vtk files in ParaView to create a movie.



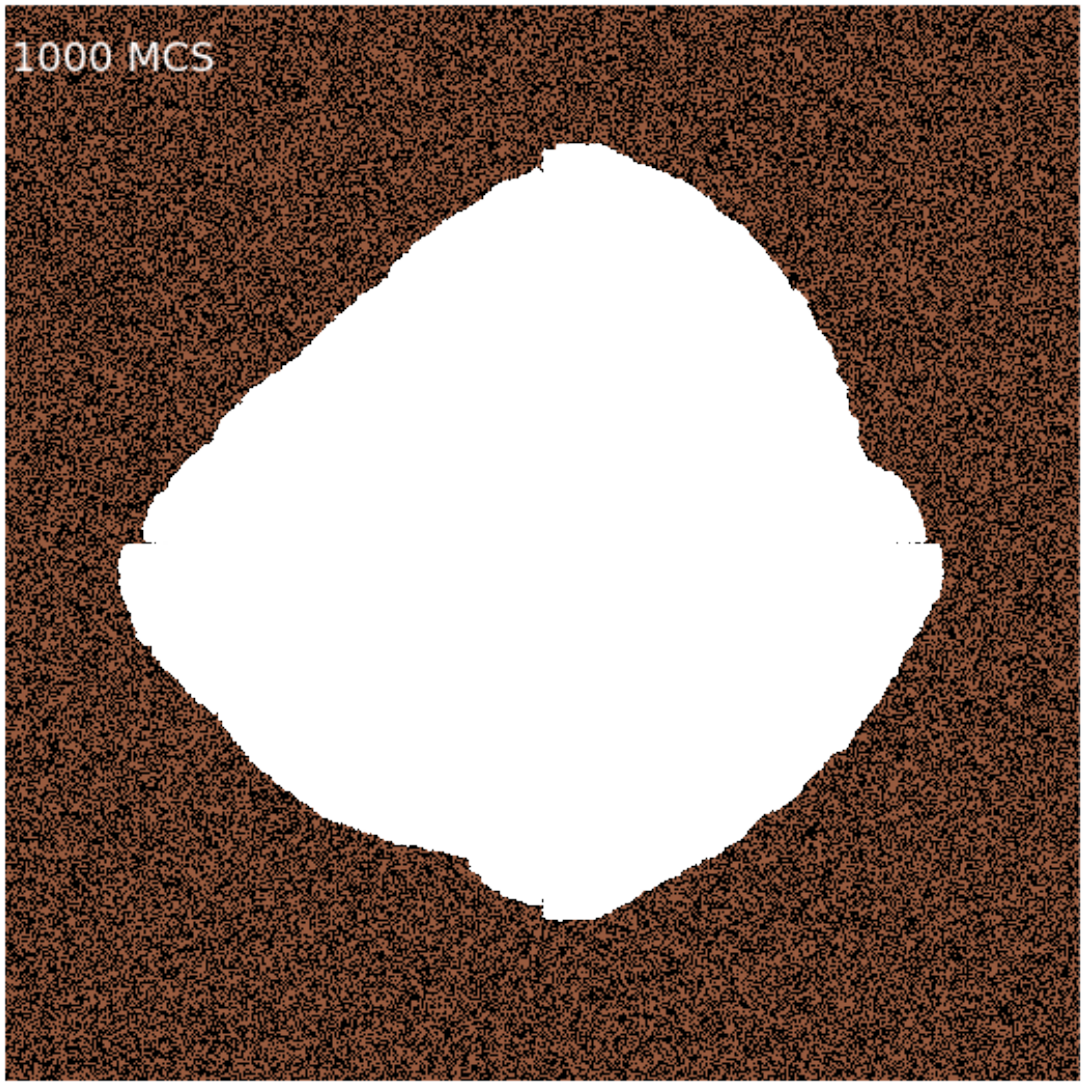
mpirun

Due to the way SPPARKS is parallelized, parallelized simulations must use the `sector` command.

Run SPPARKS with mpirun

```
mpirun -np 4 spk < viz3D.spkin
```

Parallel KMC is an approximation



Parallel KMC is an approximation