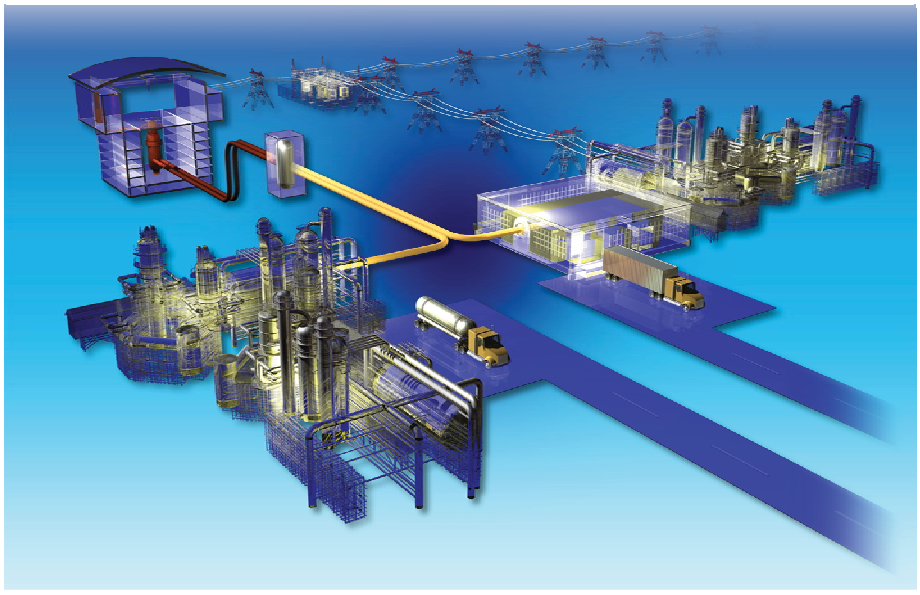
INL/EXT-09-16774

Development Status of the PEBBLES Code for Pebble Mechanics: Improved Physical Models and Speed-up

Joshua J. Cogliati
Abderrafi M. Ougouag

September 2009



The INL is a U.S. Department of Energy National Laboratory
operated by Battelle Energy Alliance.

Idaho National Laboratory

INL/EXT-09-16774

Development Status of the PEBBLES Code for Pebble Mechanics: Improved Physical Models and Speed-up

Joshua J. Cogliati and Abderrafi M. Ougouag

September 2009

Idaho National Laboratory
Next Generation Nuclear Plant Project
Idaho Falls, Idaho 83415

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Next Generation Nuclear Plant Project

Development Status of the PEBBLES Code for Pebble Mechanics: Improved Physical Models and Speed-up

INL/EXT-09-16774

September 2009

Approved by:

Joshua J. Cogliati NGNP Physics	Date
Abderrafi M. Ougouag NGNP Physics Technical Lead	Date
Gary Roberts VHTR TDO Quality Assurance Lead	Date
Hans D. Gougar VHTR TDO Deputy Technical Lead	Date

ABSTRACT

PEBBLES is a code for simulating the motion of all the pebbles in a pebble bed reactor. Because fuel elements in pebble bed reactors are randomly packed rather than precisely placed, their location in the reactor is not deterministically known. Instead, individual realizations of their stochastic locations can be found when by deterministically stimulating the motion of the pebbles. In addition to individual pebbles positions, the PEBBLES code can output information relevant to other subsequent simulations of pebble bed reactors such as slip distances of pebble pairs, packing fraction change in an earthquake, and velocity profiles created by recirculation.

The goal of this level-three milestone was to speed up the PEBBLES code through implementing on massively parallel computers. Work on this goal during this fiscal year resulted in both the speeding up of the single processor version and in the creation of a new parallel version of PEBBLES.

Both the single processor version and parallel running capability of the PEBBLES code have improved since the fiscal year began. The hybrid Message Passing Interface (MPI)/Open multiprocessing (MP) PEBBLES version was created this year to run on the increasingly common cluster hardware profile that combines nodes with multiple processors that share memory and a cluster of nodes that are networked together. The OpenMP portions use the OpenMP shared memory parallel processing model to split the task across processors in a single node that shares memory. The MPI portion uses messages to communicate between different nodes over a network. The following are wall clock speed up for simulating an NGNP-600 sized reactor. The single processor version runs 1.5 times faster than it did at the beginning of the fiscal year. This speed-up is primarily because of an improved static friction model described in the report. When running on 64 processors, the new MPI/OpenMP hybrid version has a wall-clock speed up of 22 times faster than the current single processor version. When using 88 processors, a speed up of 23 times is achieved.

This speed up and other improvements of PEBBLES combine to make PEBBLES more capable and more useful for simulation of a pebble bed reactor. This report details the implementation and effects of the speed-up work performed over the course of the fiscal year.

A new benchmark, based on the stability of a pyramid of pebbles, has been devised and used to test and verify the PEBBLES code.

ACKNOWLEDGEMENTS

Work supported by the U.S. Department of Energy, Assistant Secretary for the office of Nuclear Energy, under DOE Idaho Operations Office Contract DEAC07-05ID14517.

CONTENTS

ABSTRACT vi

Development Status of the PEBBLES Code for Pebble Mechanics: Improved Physical Models and Speed-up 1

1. INTRODUCTION 1

2. OVERVIEW OF RESULTS 1

3. PHYSICAL MODEL 2

4. IMPLEMENTATION OF THE PHYSICS 5

5. IMPLEMENTATION OF PARALLELIZATION 6

6. USES OF MODEL 8

7. IMPROVEMENTS TO FUNCTIONALITY AND PHYSICAL MODELS 9

8. CODE PROFILING AND TIMING INFORMATION 9

9. SOFTWARE PARAMETERS 11

10. PYRAMID STATIC FRICTION TEST 12
    10.1 Calculating Sphere Locations 12
    10.2 Calculating Minimum Static Friction Coefficient 13

11. FUTURE SPEED-UP STRATEGIES 15

12. CONCLUSION 17

13. REFERENCES 17

FIGURES

Figure 1. Key vectors for two pebbles. 3

Figure 2. Sphere location diagram. 12

Figure 3. Pyramid diagram. 13

Figure 4. Force diagram. 14

TABLES

Table 1. Sample PEBBLES parameters. 6
Table 2. NGNP model OpenMP version speed-ups. 10
Table 3. NGNP model MPI/OpenMP version speed-ups. 10
Table 4. AVR model speed-ups. 11
Table 5. Sphere location table. 13

Development Status of the PEBBLES Code for Pebble Mechanics: Improved Physical Models and Speed-up

1. INTRODUCTION

PEBBLES is a computer code for simulating the motion of all the pebbles in a pebble bed reactor. Since the fuel elements in pebble bed reactors are randomly packed, rather than precisely placed, their location in the reactor is not deterministically known. Instead, individual realization of the stochastic distribution of locations can be found by simulating deterministically the motion of the pebbles. The PEBBLES code can output data relevant for many simulation needs of the pebble bed reactors. These data include the positions of the pebbles in the reactor, the change in packing fraction during earthquakes, and the velocity profiles created by recirculation. The PEBBLES code is under development and continued improvement. As such, it has not yet been frozen to a stable version that would have undergone verification and validation (V&V).

The goal of the work documented in this level three milestone was to speed up the PEBBLES code through its implementation on massively parallel computers. Work on this goal has resulted in speeding up both the single processor version and creating a new parallel version of PEBBLES. The goal was achieved because both the single processor version of the PEBBLES code and the parallel running one constitute a significantly improved capability over that of the beginning of the fiscal. The single processor version runs 1.5 times faster with an NGNP-600 sized reactor, and the new hybrid Message Passing Interface (MPI)/Open multiprocessing (MP) version can achieve a speed-up factor of 23 times over the single processor version (about 35 times faster than the starting point of October 2008). The speed up in the single processor version was achieved through algorithm refining as well as improved physical models, as discussed below.

This year, the PEBBLES static friction model was improved in both computational speed and accuracy. The new model is a differential formulation written in the form

$$\mathbf{y}' = \mathbf{f}(t,\mathbf{y}),\ \mathbf{y}(t_0)=\mathbf{y}_0.$$

To the authors' knowledge, this is the first model of static friction written in this form. The new model requires fewer memory accesses to the pebble position, velocity, and angular velocity data, causing the single processor PEBBLES to run faster. This increased efficiency of memory usage also decreases the amount of data requiring transmission at the interfaces between zones (and hence computing nodes as clarified in the next section) in the parallel version, which improves parallel performance through communication overhead reduction. The new static friction model successfully passes a comparison test with an analytical benchmark of a canon balls pyramid, which was devised in this work.

These improvements to the PEBBLES code greatly increase its potential utility for pebble bed reactor design.

2. OVERVIEW OF RESULTS

Overall, the PEBBLES code runs faster than at the beginning of the fiscal year. This was achieved in two parts. The first part is by creating a hybrid MPI/OpenMP PEBBLES version that runs on the increasingly common cluster hardware profile that combines nodes with multiple processors sharing

memory and clusters of nodes that are networked together. The OpenMP portions use the Open Multi-Processing shared memory parallel processing model to split the task across processors in a memory-sharing single node. The MPI portion uses messages to communicate between different nodes over a network. The pebble motion calculation splits up the reactor geometrically, and different nodes process different portions of the reactor. Nodes communicate with other nodes that share a geometric interface. These communicating nodes send updated information on the pebble variables, such as position and velocity. The static friction module was improved for both increased speed and increased accuracy.

The following speed-ups are calculated from the wall clock speed-up for simulating an NGNP-600-sized reactor. The single processor version runs 1.5 times faster than it did at the beginning of the fiscal year, primarily because of the improved static friction model described later. When running on 64 processors, the new MPI/OpenMP hybrid version has a wall clock speedup of 22-times compared to the current single processor version. When using 88 processors, a speedup of 23 times is achieved.

Because of this speed-up, the PEBBLES code can now be used to perform a greater variety of simulation tasks within a reasonable amount of time. The earthquake simulations of the NGNP-600 model with 480,000 pebbles used last fiscal year took 40 days to generate. Now, the PEBBLES code can perform the same simulations significantly faster. Indeed, the code is now fast enough for the earthquake simulation to take 19 hours when using 64 processors with the MPI/OpenMP hybrid version on a cluster, and 81 hours when using eight processors with OpenMP on a shared memory system. With the smaller AVR reactor model (100,000 pebbles), this earthquake simulation take 9 hours for the cluster version and 49 hours for the shared memory simulation. A full recirculation using the cluster version with 64 processors would take 7 years for the NGNP reactor model, and 191 days for the AVR model. The best available version of PEBBLES at the start of the fiscal year would have taken approximately 57 years to calculate a full recirculation of a NGNP-600 sized reactor, much longer than the 7 years of the current version. However, most simulation uses do not require a full recirculation. The PEBBLES model can be applied for many more simulations within a practical amount of time, which was not possible at the beginning of the year.

3. PHYSICAL MODEL

The PEBBLES code simulates the motion of pebbles in a pebble bed reactor using the Discrete Element Method. Each pebble in the reactor is simulated separately. The pebbles experience the force of gravity, a Hooke's law force, and friction forces applied to them by their mutual contacts and contacts with vessel structure. The forces are used to calculate the subsequent motion. The equations used to calculate the forces are described below.

The classical mechanics time derivatives[1] are integrated to produce the subsequent locations, velocities and angular velocities. The linear velocity is the direction and speed of the pebble and the angular velocity is the speed of the rotation about the instantaneous axis of rotation. The following time derivatives are used:

$$\frac{d\mathbf{v}_i}{dt} = \frac{m_i\mathbf{g} + \sum_{i \neq j} \mathbf{F}_{ij} + \mathbf{F}_{ci}}{m_i} \tag{1}$$

$$\frac{d\mathbf{p}_i}{dt} = \mathbf{v}_i \tag{2}$$

$$\frac{d\boldsymbol{\omega}_i}{dt} = \frac{\sum_{i \neq j} \mathbf{F}_{\|ij} \times r_i \hat{\mathbf{n}}_{ij} + \mathbf{F}_{\|ci} \times r_i \hat{\mathbf{n}}_{ci}}{I_i} \tag{3}$$

In these equations, $\mathbf{F}_{ij}$ is the force acting from pebble *j* on pebble *i,* $\mathbf{F}_{ci}$ is the force of the container on pebble *i,* $\mathbf{g}$ is the gravitational acceleration constant, m_i is the mass of pebble *i,* $\mathbf{v}_i$ is the velocity of pebble *i,* $\mathbf{p}_i$ is the position vector for pebble *i*, $\boldsymbol{\omega}_i$ is the angular velocity of pebble *i*, $\mathbf{F}_{\|ij}$ is the tangential force between pebbles *i* and *j,* r_i is the radius of pebble *i*, $\mathbf{F}_{\|ci}$ is the tangential force of the container on pebble *i*, I_i is the moment of inertia for pebble *i*, $\hat{\mathbf{n}}_{ij}$ is the unit vector pointing from the position of pebble *i* to that of pebble *j*, and $\hat{\mathbf{n}}_{cj}$ is the unit vector normal to the container wall on pebble *i*. Key variables are shown in Figure 1.

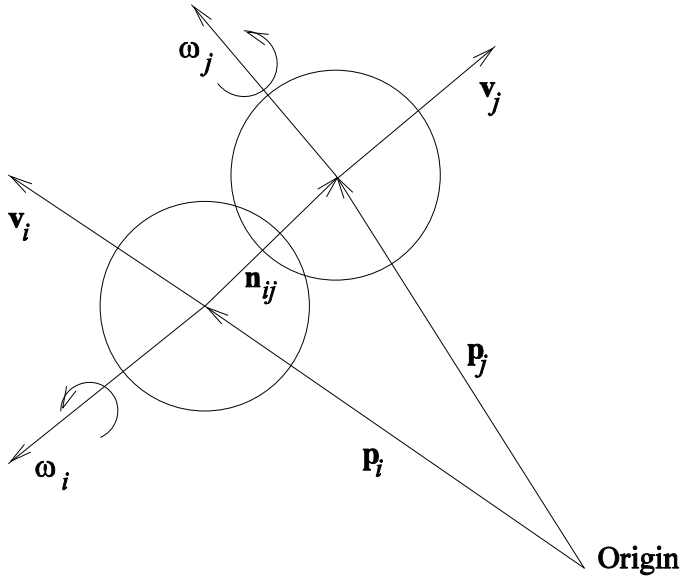


Figure 1. Key vectors for two pebbles.

The dynamic friction forces are based on the model in Wait[2] and are shown in Equations (4) and (5). Equation 4 calculates both the Hooke's law force based on the overlap of the pebbles, which defines the hardness of the pebbles, and a dissipative force based on the normal velocity of the pebbles at the contact point. Equation 5 describes a force that dissipates energy when the pebbles are rubbing against each other.

$$\mathbf{F}_{\perp ij} = h l_{ij} \hat{\mathbf{n}}_{ij} - C_{\perp} \mathbf{v}_{\perp ij}, l_{ij} > 0 \tag{4}$$

$$\mathbf{F}_{d\|ij} = -C_{\|} \mathbf{v}_{\|ij}, l_{ij} > 0 \tag{5}$$

In these equations, $C_{\|}$ is the tangential dashpot constant, $C_{\perp}$ is the normal dashpot constant, $\mathbf{F}_{\perp ij}$ is the normal force between pebbles *i* and *j*, $\mathbf{F}_{d\|ij}$ is the tangential dynamic friction force between pebbles *i* and *j,* *h* is the normal Hooke's law constant, l_{ij} is the overlap between pebbles *i* and *j,* $\mathbf{v}_{\|ij}$ is the component of the velocity between two pebbles perpendicular to the line joining their centers, $\mathbf{v}_{\perp ij}$ is the component of the velocity between two pebbles parallel to the line joining their centers, and $\mathbf{v}_{ij}$ is the relative velocity between pebbles *i* and *j*. The extra variables above can be calculated in terms of primary variables. Equation 6 calculates the relative velocity at the pebbles point of contact by using the difference in linear velocity and the surface velocity due to the rotation at the point of contact. Equations (6–9) provide the relationships between supplemental variables and the primary variables:

$$\mathbf{v}_{ij} = \left(\mathbf{v}_i + \boldsymbol{\omega}_i \times r_i \hat{\mathbf{n}}_{ij}\right) - \left(\mathbf{v}_j + \boldsymbol{\omega}_j \times r_j \hat{\mathbf{n}}_{ji}\right) \tag{6}$$

$$\mathbf{F}_{ij} = \mathbf{F}_{\perp ij} + \mathbf{F}_{\parallel ij} \tag{7}$$

$$\mathbf{v}_{\perp ij} = \left(\mathbf{v}_{ij} \cdot \hat{\mathbf{n}}_{ij}\right)\hat{\mathbf{n}}_{ij} \tag{8}$$

$$\mathbf{v}_{\parallel ij} = \mathbf{v}_{ij} - \mathbf{v}_{\perp ij} \tag{9}$$

Similar equations are used in calculating the wall interactions, with modified equations for determination of relative velocity and normal direction.

The static friction model is a simplified differential version of the model of Vu-Quoc and Zhang.[3] This model allows slip between pebbles, but assumes that as the pebbles slip they build up static friction force opposite the direction of slip. Slip changes when the velocity $\mathbf{v}_{\parallel ij}$ is non-zero, but the physical contact point is not moving because of local distortion of the pebbles.[4] If the pebbles contact point was not fixed by static friction, they would be sliding. This method replicates real static friction effects including non-zero angles of repose and bridging in small outlet chutes.

The value of the slip is used to calculate the force of static friction according to

$$\mathbf{F}_{s\parallel ij} = h_s \mathbf{s}_{ij}, \left|\mathbf{v}_{\parallel ij}\right| < v_{max} \ . \tag{10}$$

The friction force is then bounded by the friction coefficient and the normal force, to prevent it from being too great:

$$\mathbf{F}_{f\parallel ij} = \mathbf{F}_{s\parallel ij} + \mathbf{F}_{d\parallel ij} \tag{11}$$

$$\mathbf{F}_{\parallel ij} = min\left(\mu\left|\mathbf{F}_{\perp ij}\right|, \left|\mathbf{F}_{f\parallel ij}\right|\right)\hat{\mathbf{F}}_{f\parallel ij} \quad , \tag{12}$$

where $\mathbf{F}_{s\parallel ij}$ is the static friction force between pebbles *i* and *j*, $\mathbf{F}_{d\parallel ij}$ is the kinetic friction force between pebbles *i* and *j*, h_s is the coefficient for force from slip, $\mathbf{s}_{ij}$ is the slip distance perpendicular to the normal force between pebbles *i* and *j*, v_{max} is the maximum velocity under which static friction is allowed to operate, and μ is the static friction coefficient when the velocity is less than v_{max} and the kinetic friction coefficient when the velocity is greater. Equation 12 fully enforces the first requirement of a static friction method, $\left|\mathbf{F}_{\parallel}\right| \le \mu\left|\mathbf{F}_{\perp}\right|$.

There are two parts of updating the slip. First, it needs to be increased by the amount of relative distance that the contact point would have slid if they were free to slide (instead of being stuck together and slipping). Second, the direction of the slip needs to be rotated so that it remains perpendicular to the contact dimension. Both these changes are calculated from the equation for updating the slip between two pebbles as

$$\frac{d\mathbf{s}_{ij}}{dt} = \frac{\left[\left(\left(\mathbf{v}_i - \mathbf{v}_j\right) \times \left(\mathbf{p}_i - \mathbf{p}_j\right)\right) \times \mathbf{s}_{ij}\right]}{\left|\mathbf{p}_i - \mathbf{p}_j\right|^2} + \mathbf{v}_{\parallel ij} \ . \tag{13}$$

The equation for updating the slip between the wall and the pebble is

$$\frac{d\mathbf{s}_i}{dt} = (\mathbf{s}_i \cdot \hat{\mathbf{n}}_{ci}) \frac{[(\hat{\mathbf{n}}_{ci} \times \mathbf{s}_i) \times \mathbf{s}_i]}{|\hat{\mathbf{n}}_{ci} \times \mathbf{s}_i| \|\hat{\mathbf{n}}_{ci}\| \mathbf{s}_i|} + \mathbf{v}_{\|ci} \tag{14}$$

where $\mathbf{s}_i$ is the slip between pebble *i* and the wall and the other variables are defined above. Note that these equations are approximations, since they force a linear relation between the force of the slip and the distance of the slip.

Recirculation of the pebbles is simulated via removing a pebble from the bottom of the reactor, and dropping in a new pebble at the top of the reactor. Earthquakes are simulated by moving the location of the walls and adjusting the wall contact velocity.

4. IMPLEMENTATION OF THE PHYSICS

Implementing the equations requires both integration of the differential equations and a fast method of determining adjacent pebbles. This section describes both of these features and the following section describes the parallelization of these calculations.

Euler's method of integration is used for determining the change of the dependent variables over time. The derivatives for the position, velocity, angular velocity, and slips are calculated then multiplied by a time step and added to the old, previous time step, value of the variable. Improvements to the integration method will be considered, now that the new static friction model has been implemented. In addition, both the Runge-Kutta method and the Adams-Moulton method are being considered for implementation in the code.

The PEBBLES simulation uses a grid-based method to determine which pebbles are in contact with each other. PEBBLES divides the domain of locations into a three-dimensional (3-D), generalized parallelepiped, uniform grid, the cells of which have annular-wedge bases. Integer grid indices can be calculated from the pebble position using the grid spacing and the lowest grid location. With this grid, PEBBLES stores a mapping between all the pebble identification numbers (ids) and their geometric location indices. In this scheme each pebble is assigned a unique, sequential identification number (id). This proximity pebble map is a 4-D array with an x, a y, and a z entry, and an extra array dimension to store the ids of pebbles in that grid location. There is also a 3-D array that stores the number of pebble ids that are in the map. At each time step, PEBBLES updates this map of all the pebble locations. Updating is done by first zeroing the array that stores the number of pebbles, and secondly for each pebble position, determining where in the map the pebble should go, and storing the pebble id at that new location. When updating the number of pebbles in zone array, PEBBLES uses an atomic "add" and "fetch" so that the updating can be done in parallel without locks. PEBBLES then uses the updated map to determine which pebbles are in contact by determining the grid location of the pebble, and then searching adjacent locations in the map until enough distance has been searched, as the number of locations that needs to be searched depends on the grid spacing. The uses of this algorithm in PEBBLES are linear in time with respect to the number of pebbles.

Table 1 shows typical values for the parameters used in the simulation. For final calculations, constants measured from the actual graphites to be used in the pebble bed reactor will be needed as a function of temperature because these parameters can vary, depending on the graphite manufacturing process and on the operating temperature.

Table 1. Sample PEBBLES parameters.

Constant	Value
Gravitational Acceleration g	9.8 m/s^2
Mass of pebble *m*	0.2071 kg
Pebble moment of inertia *I*	7.367e-5 kg m^2
Radius of pebble *r*	0.03 m
Hooke's law constant *h*	1.0e6 N/m
Dashpot constants $C_{\parallel}$ and $C_{\perp}$	200.0
Kinetic friction coefficient μ	0.4
Static friction coefficient μ_s	0.65
Static friction hooke h_s	1.0e6
Maximum static friction velocity v_{max}	0.1 m/s

5. IMPLEMENTATION OF PARALLELIZATION

Parallelizing the computation for a full recirculation of an NGNP-600 sized reactor is highly desired, because the single processor PEBBLES code would otherwise take over a century of wall clock time for the calculation. Even with shorter calculations, faster wall-clock computation time is desirable. Two types of parallel computation are implemented and performed in PEBBLES. The first is shared memory parallelization, which shares the data via use of common memory, but the computation needs to be careful not to incorrectly overwrite common data. The second is message passing where the needed data is passed to different modules using messages. In this case, the computation needs to be careful to minimize the amount of information in each message and the number of messages while making sure that the calculations always have access to the updated data. Both methods are used in PEBBLES to combine faster shared memory parallelism with the ability to distribute the computation over non-shared memory nodes in a cluster.

The first method uses MPI to distribute the computational work across different nodes as part of the hybrid MPI/OpenMP PEBBLES code. The hybrid parallelization splits the calculation of the derivatives and the new variables geometrically and passes the data at the geometry boundaries between nodes using messages. Each pebble has a primary node and may also have various boundary nodes. The pebble primary node is responsible for updating the pebble position, velocity, angular velocity, and slips. The pebble primary node also sends data about the pebble to any nodes that are the pebble boundary nodes and will transfer the pebble to a different node if the pebble crosses the geometric boundary of the node. Node 0 is the master node and does processing that is simplest to do on one node, such a writing restart data to disk and initializing the pebble data. The following steps are used for initializing the nodes and then transferring data between them:

1. Node 0 calculates or loads initial positions of pebbles.
2. Node 0 creates the initial domain to node mapping.
3. Node 0 sends domain to node mapping to other nodes.
4. Node 0 sends other nodes their needed pebble data.

Order of calculation and data transfers in main loop:

1. Calculate derivatives for node primary and boundary pebbles.
2. Apply derivatives to node primary pebble data.
3. For every primary pebble, check with the domain module to determine the current primary node and any boundary nodes for the pebble.
    a. If the pebble now has a different primary node, add the pebble id to the transfer list to send to the new primary node.
    b. If the pebble has any boundary nodes, add the pebble id to the boundary send list to send to the node for which it is a boundary.
4. If this is a time step where Node 0 needs all the pebble data (such as when restart data is being written), add all the primary pebbles to Node 0 boundary send list.
5. Send the number of transfers and the number of boundary sends that this node has to all the other nodes using buffered sends.
6. Initialize three Boolean lists:
    a. Other nodes that this node has data to send to with true if the number of transfers or boundary sends is nonzero, and false otherwise
    b. Other nodes that this node has received data from to false
    c. Other nodes that this node has received the number of transfers and the number of boundary sends with false.
7. While this node has data to send to other nodes and other nodes have data to send to this node loop:
    a. Probe to see if any nodes that this node needs data from have data available.
        (1) If yes, then receive the data and update pebble data and the Boolean lists as appropriate
    b. If there are any nodes that this node has data to send to, and this node has received the number of transfers and boundary sends from, then send the data to those nodes. Update the Boolean data send list for those nodes.
8. Flush the network buffers so any remaining data gets received.
9. Node 0 calculates needed tallies.
10. If this is a time to rebalance the execution load, do the following:
    a. Send wall clock time spent computing since last rebalancing to node 0
    b. Node 0 uses information to adjust geometric boundaries to move work towards nodes with low computation time and away from nodes with high computation time
    c. Node 0 sends new boundary information to other nodes, and needed data to other nodes.
11. Continue to next timestep and repeat process.

All the information and subroutines to calculate the primary and boundary nodes that a pebble belongs to are calculated and stored in a FORTRAN 95 module named network_domain_module. The module uses two derived types: network_domain_type and network_domain_location_type. Both types have no public components, thus the implementation of the domain calculation and the location information can be changed without changing anything but the module. The location type stores the primary node and the boundary nodes of a pebble. The module contains subroutines for determining the location type of a pebble based on its position, the primary and boundary nodes for a location type, as well as subroutines for initialization, load balancing and transferring the domain information over the

network. The internals of the module can be changed without changing the rest of the PEBBLES code. The current method of dividing the nodes into geometric domains uses a list of boundaries between the z (i.e., axial) locations. This list is searched via binary search to find the nearest to the pebble position, as well as the nodes within the boundary layer distance above and below the zone interface in order to identify all the boundary nodes that participate in data transfers. The location type that is a result of this is cached on a fine grid, and the cached value is returned when the location type data is needed. The module contains a subroutine that takes a work parameter (which can be the computation time of each of the nodes) and can redistribute the z boundaries to shift work towards nodes that are taking less time computing their share of information. If needed in the future, the z-only method of dividing the geometry could be replaced by a full 3-D version by modifying the network domain module.

The shared memory parallelization uses OpenMP to distribute the calculation over multiple processes on a single node. OpenMP allows directives to be given to the compiler that direct how portions of code are to be parallelized. This allows a single piece of code to be used for both the single processor version and the OpenMP version. The PEBBLES parallelization typically uses OpenMP directives to cause loops that iterate over all the pebbles to be run in parallel. For the parallelization of the calculation of acceleration and torque, some details need to be taken into consideration. The physical accelerations imposed by the wall are treated in parallel, and there is no problem with writing over the data because each processor is assigned a portion of the total zone inventory of pebbles. For calculating the pebble to pebble forces, each processor is assigned a fraction of the pebbles, but there is a possibility of the force addition computation overwriting another calculation because the forces on a pair of pebbles are calculated and then the calculated force is added to the force on each pebble. In this case, it is possible for one processor to read the current force from memory and add the new force from the pebble pair while another processor is reading the current force from memory and adding its new force to that value; they both could then write back the values they have computed. This would be incorrect because each calculation has only added one of the new pebble pair forces. Instead, PEBBLES uses an OpenMP directive to force the addition to be performed atomically, thereby guaranteeing that the addition uses the latest value of the force sum and saves it before a different processor has a chance to read it. For calculating the sum of the derivatives using Euler's method, updating concurrently poses no problem because each individual pebble has derivatives calculated. For storing the pebble to pebble slips, the data structure is similar to the data structure used for the nearby pebble map. There is a 2-D array, where one index is the from-pebble, and the other index is for storing the pebbles that have slip with the first pebble. There is a second array that contains the number of ids stored, and that number is always added and fetched atomically, which allows the slip data to be updated by multiple processors at once. These combine to allow the program to run efficiently on shared memory architectures.

6. USES OF MODEL

The PEBBLES code has a variety of potential uses. The pebble positions can be determined at every simulated timestep and then used as input to other models. Such models can then calculate pebble-position-dependent parameters. For example, pebble positions have been used in calculating Dancoff factors.[5] The PEBBLES code can calculate the velocity profile across the reactor during recirculation. The profile can be then used to determine the asymptotic behavior of the reactor using the neutronics code PEBBED. A bed of pebbles can be created and then shaken using motion data for an earthquake.[6] This use can provide densification data for simulation of the neutronic effect of an earthquake. If accurate graphite wear coefficients are available, PEBBLES can be used for calculating the dust production of the reactor bed as it recirculates.[7]

7. IMPROVEMENTS TO FUNCTIONALITY AND PHYSICAL MODELS

Many improvements were implemented into the PEBBLES code in FY 2009. Additional force tallies were added to determine the pressure against the wall, the force transmitted downward, and other forces. These tests can be used in determining structural needs and potential validation of the PEBBLES code. This work was performed within this task because saturation of forces could provide the physical rationale for domain reduction and concomitant speed-up, without loss of fidelity.

The method of calculating static friction was improved. This differential method (as described in Sections 1 and 3) is both faster than the previous method used in PEBBLES and properly passes a new static friction pyramid analytical benchmark test. The pyramid benchmark test uses five spheres stacked as a pyramid as described in Section 10. The minimum pebble-to-pebble static friction coefficient required for stable stacking was analytically determined to be $\sqrt{2}-1$. The PEBBLES predicted friction factor for stability of the stacking is 0.415, which compares very well with the theoretical value of $\sqrt{2}-1$ shown above.

The modularity of the PEBBLES code has been improved. The pebble-to-pebble slip storage, earthquake movement calculation and the geometry calculation have been cast as separate FORTRAN modules. The geometry calculation was also restructured to make future geometry improvements easier. This has already simplified the modification of the static friction calculation. This modularization has made PEBBLES easier to change and will benefit future development.

The parallelization of the code was greatly improved. A new peer-to-peer hybrid MPI/OpenMP cluster implementation of PEBBLES was created. The shared memory OpenMP code was improved by replacing locking and critical sections with atomic operations such as atomic add and atomic add-and-fetch. These changes have vastly improved the efficiency and the speed-up of parallel execution of PEBBLES.

Lastly, the Pebble Removal Incremental Method (PRIMe)[8] was added as a new initial packing method. This method works by generating a large number of possible pebble positions and adding only the ones that fit (i.e., do not conflict with the walls and other pebbles). The PRIMe method can generate reasonably dense random packing structure and packing fractions. It does so substantially faster than the previous methods available within PEBBLES.

8. CODE PROFILING AND TIMING INFORMATION

The data in Table 2, Table 3, and Table 4 provide information on the time used with the current version of PEBBLES for running 80 simulation timesteps on two models. The first model is an NGNP-600 model that has 480,000 pebbles. The second model is an AVR model that contains 100,000 pebbles. All times are reported in units of wall-clock seconds. The single processor NGNP-600 model took 243.683 seconds and the AVR single processor model took 51.993 seconds when running the current version. Using the beginning of the fiscal year PEBBLES code, the NGNP-600 model took 377.259 seconds and the AVR model took 79.897 seconds on a single processor. All these were run using Intel Xeon X5355 quad core 2.66 GHz processors.

Table 2. NGNP model OpenMP version speed-ups.

Processors	Seconds	Efficiency	Speed-up
1	279.051	87.3%	0.87
2	148.325	82.1%	1.64
3	104.429	77.8%	2.33
4	78.772	77.3%	3.09
5	66.762	73.0%	3.65
6	59.317	68.5%	4.11
7	52.4	66.4%	4.65
8	46.854	65.0%	5.20

Table 3. NGNP model MPI/OpenMP version speed-ups.

Nodes	Processors	Seconds	Efficiency	Speed-up
1	8	51.743	58.9%	4.71
2	16	28.671	53.1%	8.50
3	24	21.889	46.4%	11.13
4	32	16.767	45.4%	14.53
5	40	15.669	38.9%	15.55
6	48	15.325	33.1%	15.90
7	56	14.795	29.4%	16.47
8	64	11.051	34.5%	22.05
9	72	11.306	29.9%	21.55
10	80	11.003	27.7%	22.15
11	88	10.537	26.3%	23.13
12	96	11.062	22.9%	22.03
13	104	10.955	21.4%	22.24
14	112	10.883	20.0%	22.39
15	120	11.372	17.9%	21.43
16	128	11.492	16.6%	21.20
17	136	12.818	14.0%	19.01
18	144	12.193	13.9%	19.99
19	152	12.618	12.7%	19.31
20	160	12.454	12.2%	19.57

Table 4. AVR model speed-ups.

Nodes	Processors	Seconds	Efficiency	Speed-up
1	1	59.825	86.9%	0.87
1	2	39.604	65.6%	1.31
1	3	32.771	52.9%	1.59
1	4	29.824	43.6%	1.74
1	5	27.544	37.8%	1.89
1	6	26.991	32.1%	1.93
1	7	27.589	26.9%	1.88
1	8	28.561	22.8%	1.82
1	8	30.169	21.5%	1.72
2	16	15.937	20.4%	3.26
3	24	11.163	19.4%	4.66
4	32	8.792	18.5%	5.91
5	40	7.209	18.0%	7.21
6	48	6.548	16.5%	7.94
7	56	6.198	15.0%	8.39
8	64	5.256	15.5%	9.89
9	72	4.742	15.2%	10.96
10	80	4.435	14.7%	11.72
11	88	4.042	14.6%	12.86
12	96	3.999	13.5%	13.00
13	104	3.801	13.2%	13.68
14	112	3.744	12.4%	13.89
15	120	3.738	11.6%	13.91
16	128	3.665	11.1%	14.19
17	136	3.552	10.8%	14.64
18	144	3.538	10.2%	14.70
19	152	3.552	9.6%	14.64
20	160	3.55	9.2%	14.65

9. SOFTWARE PARAMETERS

The PEBBLES code used for this report is primarily written using FORTRAN 95 and OpenMP 3.0. One module uses FORTRAN 2003 features for the FLUSH statement, determining the output unit, and obtaining command line arguments. An atomic add and fetch function is written in x86_64 assembly language; a slower FORTRAN 95 OpenMP 3.0 version is also provided. Some post processing tools are written in Python 2 and bash. The gfortran complier version 4.3 was used for compiling the PEBBLES software. For the MPI version, the mvapich2 version 1.2p1 implementation was used. The timing runs were carried out on a cluster with two Intel Xeon X5355 2.66 GHz processors per node with a DDR 4X InfiniBand interconnect network. The operating system for these is the SUSE Linux Enterprise 10. The svn revision 851 version of the PEBBLES code was used for the timing runs. The force on wall study was carried out on a cluster with two Intel Xeon 5150 2.66 GHz CPUs per node, with the OpenSUSE

11.1 operating system. The svn revision 810 was used for the force on wall study. Both reversions 810 and 851 use the physics model described in this report.

10. PYRAMID STATIC FRICTION TEST

Static friction is an important physical feature in the implementation of mechanical models of pebbles motion in a pebble bed. A pyramid static friction test model was devised as a simple tool for verifying the implementation of a static friction model within the code. The main advantages of the pyramid test are that the model test is realistic and that it can be modeled analytically, providing an exact basis for the comparison. The test benchmark consists of a pyramid of five spheres on a flat surface. This configuration is used because the forces acting on each pebble can be calculated simply and the physical behavior of a model with only kinetic friction is fully predictable on physical and mathematical grounds: with only kinetic friction and no static friction, the pyramid will quickly flatten. Even insufficient static friction will result in the same outcome. The four bottom spheres are arranged as closely as possible in a square, and the fifth sphere is placed on top of them as shown in Figure 2.

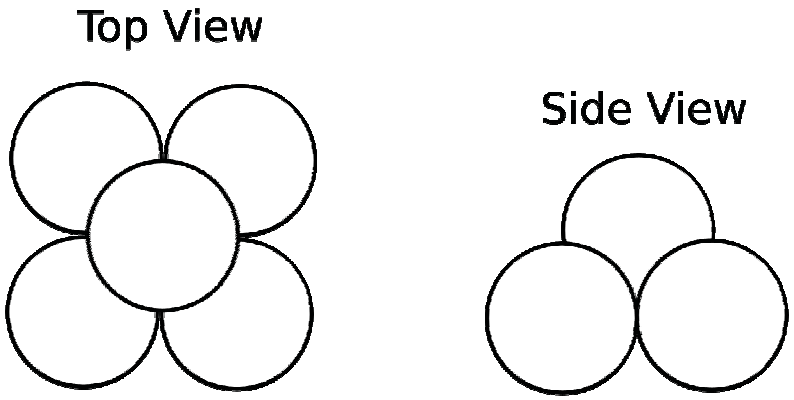


Figure 2. Sphere location diagram.

10.1 Calculating Sphere Locations

The lines connecting the centers of the spheres form a pyramid with sides 2R, as shown in Figure 3, where R is the radius of the spheres. The length of "a" in the figure is $\frac{2R}{\sqrt{2}}$, and because "b" is part of a right triangle, $(2R)^2 - \left(\frac{2R}{\sqrt{2}}\right)^2 = b^2 = 4R^2 - \frac{4R^2}{2} = 2R^2$, so "b" has the same length as "a," and thus the elevation angle for all vertices of the pyramid are 45 degrees from horizontal.

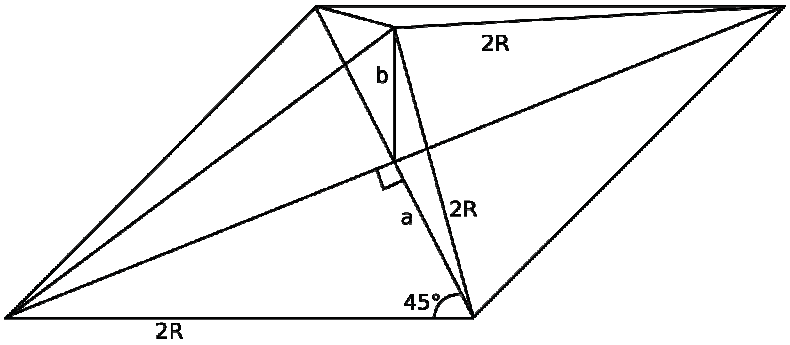


Figure 3. Pyramid diagram.

Taking for origin of the coordinates system the projection of the pyramid summit onto the ground, the locations (i.e., coordinates) of the sphere centers are given in **Error! Not a valid bookmark self-reference.**.

Table 5. Sphere location table.

X	Y	Z
-R	-R	R
R	-R	R
-R	R	R
R	R	R
0	0	$R\left(1+\sqrt{2}\right)$

10.2 Calculating Minimum Static Friction Coefficient

The initial forces on a base sphere are the force of gravity m**g** and the normal forces **Tn** and **Fn** as shown in Figure 4. This causes initial slip which will cause **Fs** to develop to counter the slip, and **Ts** to counter the rotation of the base sphere relative to the top sphere. The top sphere will have no rotation because the forces from the four spheres below it will be symmetric and counteract each other.

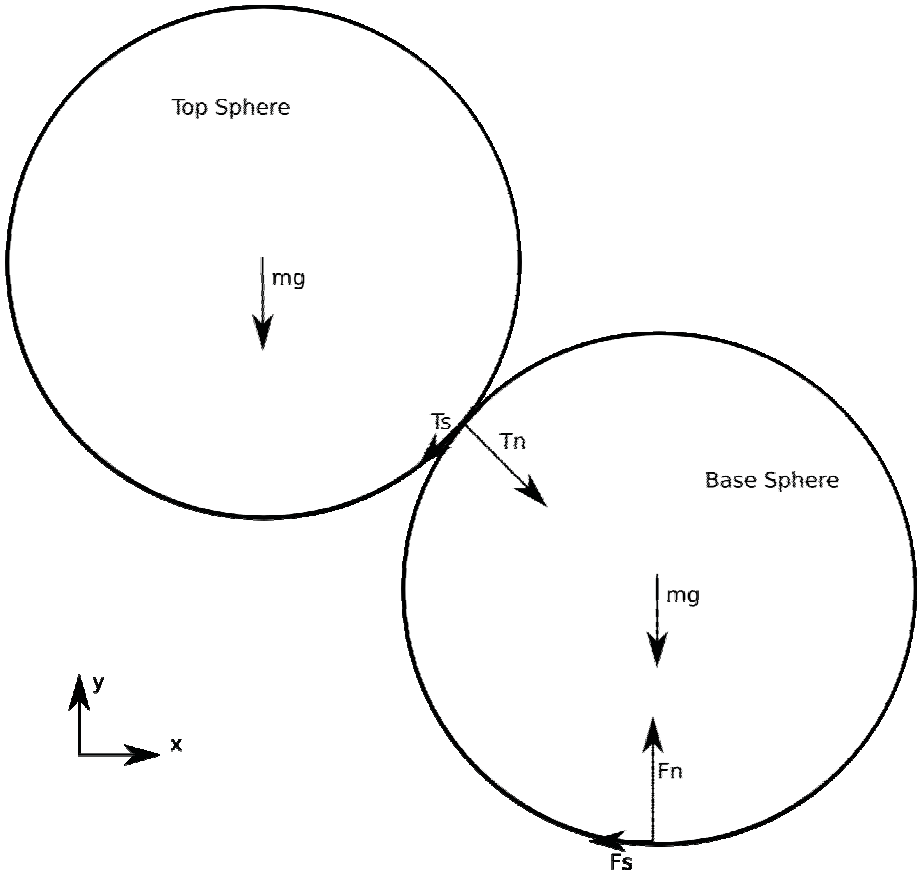


Figure 4. Force diagram.

The forces on the base sphere are:

Tn – Normal force from the top sphere

Ts – Static friction force from the top sphere

m**g** – Force of gravity on the base sphere

Fn – Normal force from floor

Fs – Static friction force from the floor

Note that **Fn** is larger than **Tn** since **Tn** is only a portion of the m**g** force since the top sphere transmits (and splits) its force onto all four base spheres.

There are three requirements for the base sphere to be non-accelerated.

If the base sphere is not rotating, then there is no torque, so

$$|\mathbf{Fs}| = |\mathbf{Ts}| \tag{15}$$

The resultant of all forces must also be zero in the x and the y directions (vector notation dropped since they are in one dimension and therefore scalars) as follows:

$$-Fs - Tsx + Tnx = 0 \tag{16}$$

$$-mg - Tsy - Tny + Fn = 0 \tag{17}$$

Since the angle of contact between the base sphere and the top sphere is 45 degrees, the following two equations hold (where *Ts* is the magnitude of **Ts** and *Tn* is the magnitude of **Tn**):

$$Tsx = Tsy = \frac{Ts}{\sqrt{2}} \tag{18}$$

$$Tnx = Tny = \frac{Tn}{\sqrt{2}} \tag{19}$$

This changes Equations 16 and 17 into:

$$-Fs - \frac{Ts}{\sqrt{2}} + \frac{Tn}{\sqrt{2}} = 0 \tag{20}$$

$$-mg - \frac{Ts}{\sqrt{2}} - \frac{Tn}{\sqrt{2}} + Fn = 0 \tag{21}$$

Combining Equations 15 and 20 provides

$$-Ts - \frac{Ts}{\sqrt{2}} + \frac{Tn}{\sqrt{2}} = 0 \tag{22}$$

Which gives the relation

$$Tn = Ts\left(\sqrt{2} + 1\right) \tag{23}$$

By static friction,

$$Ts \le \mu Tn \tag{24}$$

Combining Equations 23 and 24 and simplifying gives the requirement

$$\sqrt{2} - 1 \le \mu \quad . \tag{25}$$

For use with testing, the static friction program can be tested twice with a friction coefficient slightly above 0.414 and one slightly below 0.414. In the first case, the pyramid should be stable. In the second case, the top ball should fall to the floor.

This test was inspired by an observation of lead cannon balls stacked into a pyramid. A desktop experiment was performed in which an attempt was made to stack glass marbles into a five ball pyramid. The attempt failed, as such a pyramid proved to be unstable. Since lead has a static friction coefficient around 0.9 and glass has a static friction coefficient around 0.1, the physics of pyramid stability was further investigated, resulting in this benchmark test of static friction modeling.

11. FUTURE SPEED-UP STRATEGIES

Even after all the improvements implemented this past fiscal year, the full recirculation computation for NGNP-600-class pebble bed reactor would require about 7 years. Although the recirculation modeling needs to be performed only a few times in the course of the design of a pebble bed reactor (with the

generation of pebble flow maps that are retained), seven year is still excessive. It is highly desirable to have the running time decreased enough to allow full recirculation simulations within more reasonable wall clock times. A possible approach to progress toward this goal would be to introduce approximations that although speed up the computation would also introduce a decrease in model fidelity. Such a tradeoff would in turn require further analysis for justification and validation. In this work, thus far, the model retained maximum fidelity by systematically avoiding approximations. This choice is to be retained for the foreseeable future and no compromise on fidelity will be contemplated. Therefore, there remain two alternatives for improving the speed. One is to increase the computation speed without changing the physics. The other is to use alternative methods of modeling the physics that retain the required fidelity, that is to use alternate, equally high-fidelity, mathematical formulations for the same physics.

Although parallelization needs to be improved, the characteristics of the problem at hand make significant progress in this area difficult. Basically, PEBBLES transfers data between nodes at each timestep so that each node has access to all the relevant information for the following timestep. Wall-clock timings for a characteristic timestep when running the NGNP-600 model are summarized in the next paragraph.

A single OpenMP 8 processor timestep takes about 585 ms. Timesteps for eight nodes, each having eight processors, take about 128 ms. If the speed-up was perfectly linear, this would take one-eighth the time, or about 73 ms, which means that about 55 ms are attributable to overhead. What takes up the additional 55 ms? About 12 ms is taken up with deciding which pebbles need to be transferred and where (i.e., which pebbles need to be assigned to another node). About 8 ms is taken up by actually transferring the data over the network. The calculation times range from about 90 ms to 100 ms. This computation takes longer than 73 ms for two reasons. First, about 10% of the pebbles that are having computations performed about them are boundary pebbles, so the calculations end up being performed in two nodes instead of just one. This effect is responsible for about 10 ms of the time. The additional ~10 ms of computation time possibly comes from additional overhead involved in synchronizing memory that is required when multiple processors need to access and update the same memory. Finally, the load balancing is not perfect and the difference in time between the slowest computation and the fastest results in about 10 ms of waiting time for some of the nodes. These five causes add up to roughly 50 ms of overhead that are caused by inherent inefficiencies in the implementation of parallelization. For reference, the improvement in load balancing took approximately 2 weeks of programmer time and reduced the overhead by about 20 ms per timestep. There is potential for further overhead reduction in the parallelization, but they will take significant programmer time.

The NGNP-600 model on the OpenMP PEBBLES version has 65% efficient parallelization, but the smaller AVR model is only 23% efficient. The main difference between the models is in their relative magnitudes: the number of pebbles in the AVR model is about one-fifth that of the NGNP-600 model. However, the cause of this discrepancy in efficiency between the two reactors needs to be investigated further. Further possible steps to improve parallelization will be decided on when the causes study is completed.

Another possibility for increasing the speed is to increase the size of the boundary layers so that data do not need to be transferred at every timestep. This increases the amount of data that needs to be transferred and the amount of computation done on the boundary pebbles, but may be offset by having to do the determination of data transfer and the transmittal of transfer data less frequently.

The remaining speed-up strategies are changes in the implementation of the physics models. One possibility for speed up is through taking advantage of the fact that much of the reactor may be geometrically homogenous. Between the loading chutes at the top and the outlet chutes at the bottom the reactor vessel typically consists of repeating wall elements. As long as the properties that are being computed are homogenous in this middle region of the reactor, the computation of the pebbles speed and

trajectories in most of this mid-section can be omitted. This is true except for a top layer and a bottom one within this middle section. One key determinant of the validity of this approach is whether the forces on the pebbles saturate. If they do saturate, then the approach would be valid. Otherwise, even though the reactor vessel is geometrically and materially uniform in the middle region, it still would be dynamically variable in the axial direction, with the lower pebbles experiencing greater forces from the effect of higher pebbles. The saturation of forces is a consequence of of static friction. The mechanism at work is a transmission of the downward forces laterally towards the walls. With sufficiently large static friction and with sufficient depth, the force exerted by one layer of pebbles on the next lower layer stops increasing with depth. This phenomenon is well documented in the literature.[9,10] If this saturation of forces occurs, all relevant variables are expected to be homogenous and invariant, on average, in the middle region below where saturation first occurs. In this case, the homogenous region can be omitted from the calculation.

One final possibility for increasing the speed of modeling pebbles recirculation is improving the integration method to allow the use of larger timesteps without loss of accuracy. In the existing code, simulation timesteps of 0.1 ms are used. This is finer than necessary for a recirculation, but the current method of integration becomes unstable if much greater timesteps are used. With the new differential version of the static friction calculation, other integration methods besides Euler's method are possible. The Runge-Kutta method and the Adams-Moulton method might allow larger timesteps to be used. If the increased timesteps were large enough, there could result a speed-up of the PEBBLES code because overall fewer timesteps would be needed for the simulation of a given real time duration. In conclusion, continued work on speeding up the PEBBLES code is needed and is expected to result in further decreases of the wall-clock time. However the successful strategies are yet to be determined from among the many possibilities.

12. CONCLUSION

The PEBBLES code has been speeded-up substantially through improvements in the implementation of its physics models and through improved implementation on parallel computer architectures. Further work is needed before PEBBLES can be used for simulating a full recirculation of a NGNP-600 sized reactor within reasonable wall clock times. One possible method of further speeding up the simulation is changing the integration method. Another possibility is reducing the calculation in homogeneous regions of the reactor.

13. REFERENCES

1. Jerry B. Marion, and Stephen T. Thornton, *Classical Dynamics of Particles and Systems*, Harcourt, Austin, Texas, Chap. 2, 1995, p. 11.
2. R. Wait, "Discrete Element Models of Particle Flows," *Mathematical Modeling and Analysis*, Vol. 6, No.1, 2001, pp. 156–164.
3. Loc Vu-Quoc, Xiang Zhang, and O. R. Walton, "A 3-D Discrete-element Method for Dry Granular Flows of Ellipsoidal Particles," *Computer Methods in Applied Mechanics and Engineering*, Vol. 187, 2000, pp. 483–528.
4. R. D. Mindlin, H. Deresiewicz, "Elastic spheres in contact under varying oblique forces", *ASME J. Applied Mechanics*, September 1953, pp. 327–344.
5. A. M. Ougouag, J. J. Cogliati, and J-L Kloosterman, "Methods of Modeling the Packing of Fuel Elements in Pebble Bed Reactors," *Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications, Avignon, France, September 12–15, 2005*, American Nuclear Society.

6. A. M. Ougouag, J. Ortensi, and H. Hiruta, “Analysis of an Earthquake-Initiated-Transient in a PBR,” *International Conference on Mathematics, Computational Methods & Reactor Physics, Saratoga Springs, New York, May 3–7, 2009*, American Nuclear Society.

7. J. J. Cogliati and A. M. Ougouag, “Pebble Bed Reactor Dust Production Model,” *Proceedings of the 4th International Topical Meeting on High Temperature Reactor Technology, September 28 – October 1, 2008*, Washington, D.C., USA.

8. J-L Kloosterman and A. M. Ougouag, “Computation of Dancoff Factors for Fuel Elements Incorporating Randomly Packed TRISO Particles,” INEEL/EXT-05-02593, Idaho National Laboratory, January 2005.

9. H. A. Janssen, “Experiments on Corn Pressure in Silo Cells,” Engineer in Bremen, Germany, 31st August 1895.

10. D. M. Walker, “An approximate theory for pressures and arching in hoppers,” *Chemical Engineering Science*, Vol. 21, 1966, pp. 975997.