APCOMP 290R Modulo 3: Battery Modeling

Group CPU

December 9, 2015

Team members:

Xiaowen Chang

Feifei Peng

Zelong Qiu

Qing Zhao

I. Introduction:

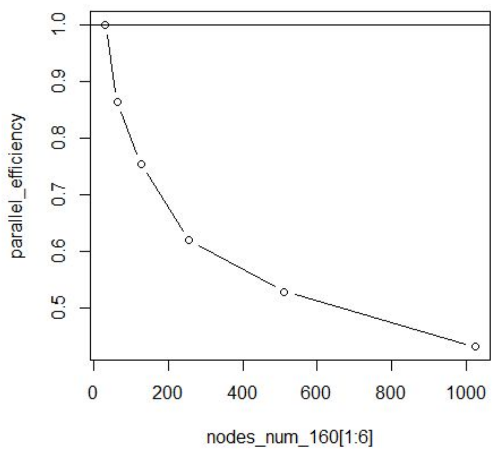
Li-Mn batteries are very common in consumer electronics. They have a high energy density, small memory effect and only a low loss of charge when not in use so it is one of the most popular types of rechargeable batteries for portable electronics. The battery consists of three parts: electrode, SEI and electrolyte. The electrode contains Mn and the Electrolyte contains Li. The goal of our project is to minimize Mn diffusion and maximize Li diffusion.

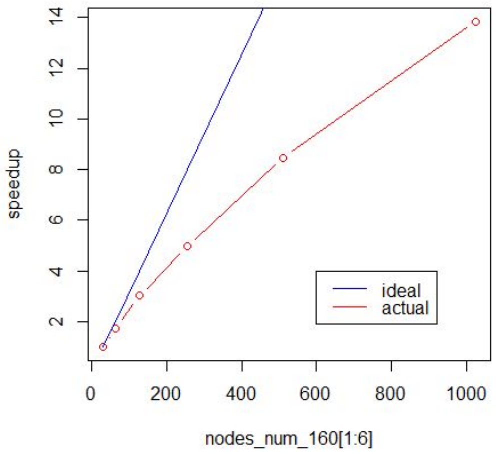
Kinetic Monte Carlo (KMC) algorithm was used to simulate the movement of Li and Mn when the battery is working. It is a computationally expensive problem and hence we used supercomputer provided by ALCF to do the simulation. In the following part, we will first discuss about the parallel efficiency of SPPARKS code on large-scale computers and then we will talk about our design strategy to inhibit Mn dissolution from electrode, the factors captured at large scale vs at small scale and additional physics for improving quality of simulations. Lastly, we will further explore more about Monte Carlo algorithm and its application.

II. Parallel Efficiency of SPPARKS code on large-scale computers

1. Estimating the resources and runtime required to simulate a battery system on 8,192 nodes.

To estimate the resources and runtime required to simulate a battery system on 8192, we run the simulation code on 32, 64, 128, 256, 512, 1024 nodes on Cetus with 160*160*160 box size. We first plotted the speedup VS number of nodes and parallel efficiency VS number of nodes to observe how parallel efficiency changed as the number of nodes increased.





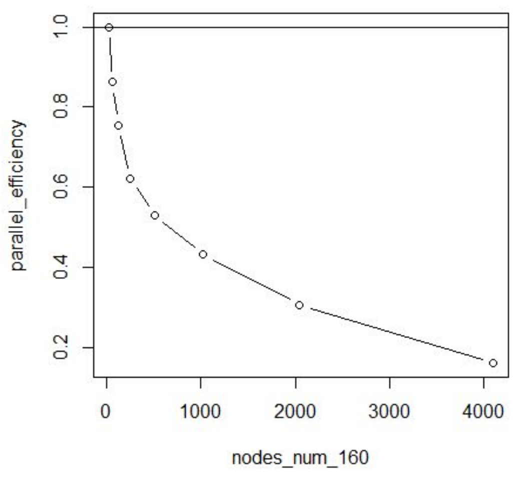
Based on the Amdahl's law, if the 1/N of the compute time is used in serial code, then the maximum speedup is N on any number processors. From plots above, we can see that when the number of nodes increased, the parallel efficiency decreased and the speedup is more and more far way the ideal situation(the maximum speedup). The reason for it is that in real situation, the parallelized code is not able to scale linearly with the number of workers once parallelized

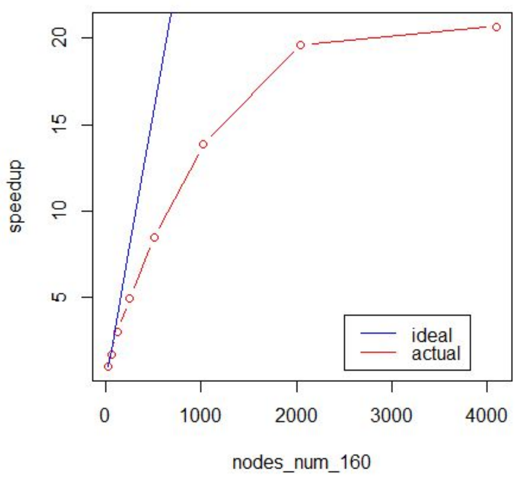
To estimate the time of the simulation on 8192 nodes, we modeled parallel efficiency VS number of nodes with the function y=C*N^(a) where y is the parallel efficiency, N is the number of nodes, C and a are parameters of the model. Using the data we have, the model we get is
y = 2.125728*(N^(-0.2175914)).

So when N=8192, the estimate for parallel efficiency is 0.299218 and the estimate for the run time is 0.1828105(s).

2. Validate performance as best you can up to 8,192 nodes on Mira.

The maximum nodes we could run on Mira is 4096. So to validate the performance, we run the code on 2048 and 4096 nodes. Then we plotted the speedup VS number of nodes and parallel efficiency VS number of nodes again to observe the performance.





From the speedup VS number of nodes plot, we could see that the speedup of 4096 nodes did not increase much from the speedup of 2048 nodes. And it seems like the speedup of 4096 nodes is already very close to the best speedup the code could achieve through parallelization. The reason for it is that there are some parts of the SPPARKS code can not be parallelized. So the best speedup the program could achieve is 1/(1-f) where f is the fraction of parallelizable code. So in this case the run time on 8192 nodes should be very close to the run time on 4096 nodes, which is between 6s and 7s.

III. The design strategy to inhibit Mn dissolution from electrode

Our battery consist of three parts Electrode, Electrolyte, Solid Electrolyte Interfaces. The electrode contains Mn and the Electrolyte contains Li. Ideally, as the Li ion move from electrolyte to electrode, the electrical power is generated by the battery. However, in reality the Mn in electrode keeps dissolving into electrolyte, which decrease the performance of battery. To inhibit Mn from dissolution, scientist designed a layer called Solid Electrolyte Interfaces, also called SEI, to try to prevent Mn from dissolving into electrolyte. The SEI is placed between electrode and electrolyte as a protector of electrode. SEI could consist of different compounds. In our in.battery script, we use i2 value to denote the element at a point and we have 4 different kind of SEI, which are i2 = 6 is sei1(LiF), i2 = 7 is sei2(Li2O), i2 = 8 is sei3(Li2CO3), and i2 = 9 is sei4(carbonate). In our simulation script, we also have the choice to determine the thickness of the SEI layer through the variables interface1 and interface2. The interface1 specify the low value of the SEI layer and the interface2 specify the higher value of the SEI layer. The difference of interface1 and interface2 is the thickness of the SEI layer.

Therefore, to enhance the performance of battery, we want to find the best way to reduce the diffusion rate of Mn. Based on the diffusion equation, we know that the temperature plays an important role in the diffusion process. The higher the temperature is, the fast the diffusion process is. Also, since the SEI layer is designed to prevent Mn from dissolving, we believe that the thicker the layer is the lower the diffusion rate is. In addition, another very important factor regarding SEI layer is the component of SEI. Since there are four possible chemical elements that are able to consist SEI layer, any subsets of this four elements set could also form SEI layers. We believe that the component of the SEI layers also have significant impact on the Mn’s dissolution rate.

Since we are not sure which one of the sei1-sei4 or which subset of these four elements could make Mn dissolve slowest, we conduct simulation with each combination of these four elements. To do that, we change the batter build parameters in in.battery script as 5 which refer to electrolyte with each subset of {6,7,8,9} which refer to different combination of these four SEI materials.

To analyze the output, we choose the file named as “spk_analyze.dump_ion_x.dat”, which contains two rows. First rows has six values, li1, li2, li3, Mn1, Mn2, and Mn3, which describe at time 0 the fraction of Li ions and Mn ions in Electrode, Solid Electrolyte Interfaces, Electrolyte respectively. The second row also have six values but it provide information for the fractions of Li and Mn ions location at time 100. One sample output is shown below:

	li1	li2	li3	mn1	mn2	mn3
t 0	0.026908	0.000000	0.973092	1.000000	0.000000	0.000000
t 100	0.026377	0.000824	0.972799	0.980495	0.019505	0.000000

t 200	0.026390	0.001279	0.972330	0.980186	0.019814	0.000000
t 300	0.026390	0.001501	0.972109	0.980557	0.019443	0.000000
t 400	0.026390	0.001735	0.971875	0.980495	0.019505	0.000000
t 500	0.026390	0.001914	0.971696	0.980619	0.019381	0.000000
t 600	0.026390	0.002100	0.971510	0.980681	0.019319	0.000000
t 700	0.026390	0.002305	0.971305	0.980681	0.019319	0.000000
t 800	0.026390	0.002500	0.971110	0.980681	0.019319	0.000000
t 900	0.026390	0.002660	0.970950	0.980681	0.019319	0.000000
t 1000	0.026390	0.002849	0.970761	0.980681	0.019319	0.000000

At time 0, the li2 has 0 in SEI but at time 100, li2 has .000824 in SEI. Thus, 0.0824% of Li has moved into SEI during the period of first 100 steps of simulation. For Mn, it has 0% in SEI at time 0 but at t = 100 it has 1.95% in SEL. Thus, 1.95% of Mn has dissolved from electrode during this period of first 100 steps of simulation. Therefore, to calculate the rate of dissolution we take the difference of li2 and mn2 between different time steps and then take the average of these differences. In the sections below, we only list the result averages as the dissolution rate of Li ion and Mn ion.

To have better performance, we want to minimize the dissolution rate of Mn and maximize the movement of Li. We run the in.battery script with default thickness of 12-4 = 8 and default temperature 0.034 to find the best combination of SEI. All tests are run on box 160 * 160 * 160. The results of dissolution rate for Li and Mn for different combination of sei1 - sei2 are listed below.

	56789	5678	5679	5689	5789
Li	0.012343	0.012703	0.012073	0.012263	0.012208
Mn	0.016285	0.011517	0.021920	0.021920	0.021610

	567	568	569	578	579	589
Li	0.012857	0.012594	0.011601	0.012594	0.011569	0.011964
Mn	0.013808	0.013622	0.024087	0.013622	0.024334	0.025015

	56	57	58	59
Li	0.012237	0.012285	0.012044	0.008809
Mn	0.019876	0.021734	0.020495	0.021981

From the data above, we see that the combination of 5678 has lowest dissolution rate for Mn. Then, we fix the component of SEI and run simulation to see how the temperature affects the dissolution rate. All tests are run with same SEI thickness as 12 - 4 = 8 and save SEI components 6,7,8. The results are listed below:

Temp		0.030	0.034	0.039	0.043
Li		0.010152	0.012703	0.012237	0.017538
Mn		0.009907	0.011517	0.019876	0.022353

From the result, we can see that as the temperature increases, the dissolution of Mn increases which is consistent with the diffusion equation. Thus, to inhibit the Mn dissolution from electrode, we want to the temperature of battery to be lower. The Mn dissolved slowest at temperature at 0.026 eV, which is 300K.

So far, we reach the optimum setting for minimum Mn dissolution as using SEI consist of sei1, sei2, sei3 and set the battery temperature to. Finally, we explored how the thickness of SEI layer affect the dissolution rate of Mn from electrode. We keep the battery build parameters in the in.battery script as 5 6 7 8 and keep the temperature as 0.026. Then set the interface2 to 10, 12, 14, 16 respectively to see the results of thickness 6, 8, 10, 12. The results are listed in the table below:

Thickness	6	8	10	12
Li	0.014616	0.010152	0.000807	0.012237
Mn	0.019752	0.009907	0.009536	0.019876

From the table above, we can see that as the thickness increases, the dissolution rates of both Li and Mn decreased. However, we could see that as the thickness increase from 8 to 10, the reduce of the dissolution rate of Mn isn't much but the dissolution rate of Li dropped a lot. Since we want to minimize the dissolution of Mn from electrode and maximize the Li movement from electrolyte, we believe that the thickness of 8 is the best choice.

All in all, we inhibit the dissolution of Mn from electrode by choosing the components of SEI which are sei1, sei2, sei3, by setting the high temperature to 300k and by keeping the thickness as 8.

IV. Factors captured at large scale vs at small scale

There are two things related to scale in our model. One is the box size, or side length, and another is the time steps. The box size we used here is 160*160*160. Although it is not very small, the boundary condition might still cause our model to behave differently. The time step we used here is 1000, which is very short. Significant changes might not occur and some certain diffusion might not be captured within such short period of time. If we could have longer time or more computational resources, we could certainly try to increase the number of time steps to 10k or 30k to get better simulation results.

V. Additional physics for improving quality of simulations

Although our simulation model has plausible accuracy and the simulated results are consistent with the theorem, they may be more physics could be incorporated into the simulation model to further improve the quality of the simulation.

Firstly of all, our model use interface1 and interface2 as the lower and higher position of the SEI, which assume the interface between electrode and Solid Electrolyte, as well as the interface between electrolyte and Solid Electrolyte Interfaces are ideally regular flat surfaces. However, in reality, these surfaces may be actually irregular or non flat. Therefore, may be we could model the thickness of the SEI as a function of the position x instead of a constant number calculated by interface2 minus interface1.

Secondly, our model assume that for a single ion, located in a specific region, the energy it takes to jump to a vacancy are same through fcc, oct, tet sites. For example, in the in.battery script, it takes Mn, in SEI component #3, 0.53 to jump to a vacancy location no matter the Mn ion is at fcc, oct, or tet sites. In reality, these may be different for different type of sites. If we could incorporate this physic, we may improve the quality of our simulation a little bit more.

VI. Further exploration about Monte Carlo

1. List at least three differences between kinetic Monte Carlo (kMC) and Molecular Dynamics.

1) Rather than following the trajectory through every vibrational period, KMC evolves systems dynamically from state to state and these state-to-state transitions are treated directly.
2) Molecular Dynamics algorithm requires first choosing an interatomic potential for the atoms and a set of boundary conditions.
3) For Molecular Dynamics algorithm, accurate integration requires time steps short enough to resolve the atomic vibrations. Consequently, the total simulation time is typically limited to less than one microsecond, while processes we wish to study such as diffusion often take place on much longer time scales while KMC can reach vastly longer time scales, typically seconds and often well beyond.

2. Give three other applications that one could use Monte Carlo techniques, with your rationale.

Monte Carlo plays a very important role in numerical optimization area. One of very popular applications is to find the global minima/maxima of some complicated function. The rationale is as following:

```
1. Initialize the starting solution points X, step size L and temperature T.
2. for i through imax:
    a. Propose a new x∗←neighbor(xi,L),
    b. Draw u~Uniform(0,1),
    c. if u<P(X=x∗),
        i. xi=x∗
    d. Update T and L.
3. xopt=xi.
4. Return xopt.
```

Besides, Monte Carlo also has some applications in artificial intelligence area such as search for the best move in a game. This is a NP-hard problem. Monte Carlo could give us a better result than greedy algorithm without searching through all possible situations. The rationale is as following:

1. Starting at root node of the tree, select optimal child nodes until a leaf node is reached.
2. Expand the leaf node and choose one of its children.
3. Play a simulated game starting with that node.
4. Use the results of that simulated game to update the node and its ancestors.

In addition, since Monte Carlo could help approximate distribution using samples, it is also widely used in applied statistics area. An interesting application is that it could provide implementations of hypothesis tests that are more efficient than exact tests such as permutation tests. The rationale is as following:

1. Generate S independent data sets under the conditions of interest.
2. Compute the numerical value of the estimator/test statistic T(data) for each data set ⇒ T1,...,TS.
3. If S is large enough, summary statistics across T1,...,TS should be good approximations. to the true sampling properties of the estimator/test statistic under the conditions of interest.

3. Give the MC algorithm for a problem that you are interested in solving.

Monte Carlo Algorithm is a physically accurate and intuitively accessible approach to radioactive decay, which can be used for the iterative computation of the time-varying number of nuclei. Radioactive decay is a discrete process with each atom having some finite probability of decay. Since Monte Carlo methods are associated with randomness, specifically, generating random numbers and using them to attempt to solve the radioactive decay problem is reasonable. We can test each atom individually to see if it decays, and for a large number of atoms, this approach can be implemented using computational tools.

The rationale behind using MC algorithm to solve this problem is that radioactive decay is a truly random process and the probability of decay is constant. As a result, Monte Carlo methods are an appropriate way to model the decay. For all the nuclei the decay rate is $dN(t)/dt = -\lambda N(t)$, where λ is the decay constant in units of probability per unit time for each nucleus of a given species.

For a given nucleus, the probability P that a nucleus undergoes radioactive decay in time dt is $P=\lambda\, dt$. Therefore if we know the initial number of nuclei N, we can calculate randomly for each nuclei if that nuclei decays or not for some small dt. If it does, we remove it from the sample and recalculate for the next small interval dt.

The specific algorithm used for the iterative computation of the time-varying number of nuclei is as following:

```
1.  Input initial number of nuclei N_0
2.  Input decay constant λ
3.  Input number of time steps M
4.  Input time step dt
5.  Loop: i from 0 to M
6.      Loop: j from 0 to N_i
7.          Generate a random number r
8.          If r < λ*dt, then N_i=N_i-1
9.      Output i*dt and N_i to a file
10.     N_i+1 = N_i
11. Plot N_i vs i*dt for all i
```