Rahul Vig
Software Engineering
Project 1 Report

Spec Checklist:

- ✓ Submitted solution in a zip archive with the correct file name format.
- ✓ Program zip contains all the program source code
- ✓ Code compiles and runs on Mars server without timing out
- ✓ Project states the values of N_m and N_f to answer Section 1.9
- ✓ Values of B & C are stated and used to answer the challenges that I've solved
- ✓ Project contains graphics and screenshots of my work
- ✓ Challenge 4 states the highest peak value of <m>
- ✓ Challenge 5 states the lowest minimum value of <cp>

Values of N_M and N_F

In designing and implementing the specs of the project I found the following values for N_m and N_f to be optimal:

$$N_m = 15$$
$$N_f = 20$$

Mission Statement: To design a program that uses the Metropolis Algorithm for simulated annealing. The use case for this specific project was to use the algorithm to find the most optimal configuration of quantum spins. This optimal configuration of spins should minimize the energy difference and once an optimal energy difference is found then the Magnetization Per Spin and Pair Correlation Per Spin can be calculated. The Metropolis Algorithm only finds one optimal configuration and in order to increase accuracy we must call the Metropolis algorithm N_M times while figuring out what is a good number for the number of flips N_F to also increase the preciseness of the algorithm. After the algorithm runs N_M times we can calculate the local mean of MPS and PCPS of a single thread. However, simply using one thread is not enough so we must store each local mean values from each thread in order to find a global mean once all the threads have finished execution. However, <M> and <C> are dependent on temperature and so in order to plot at least 10 points for our graph, the program must be able to run 10,000 threads efficiently.

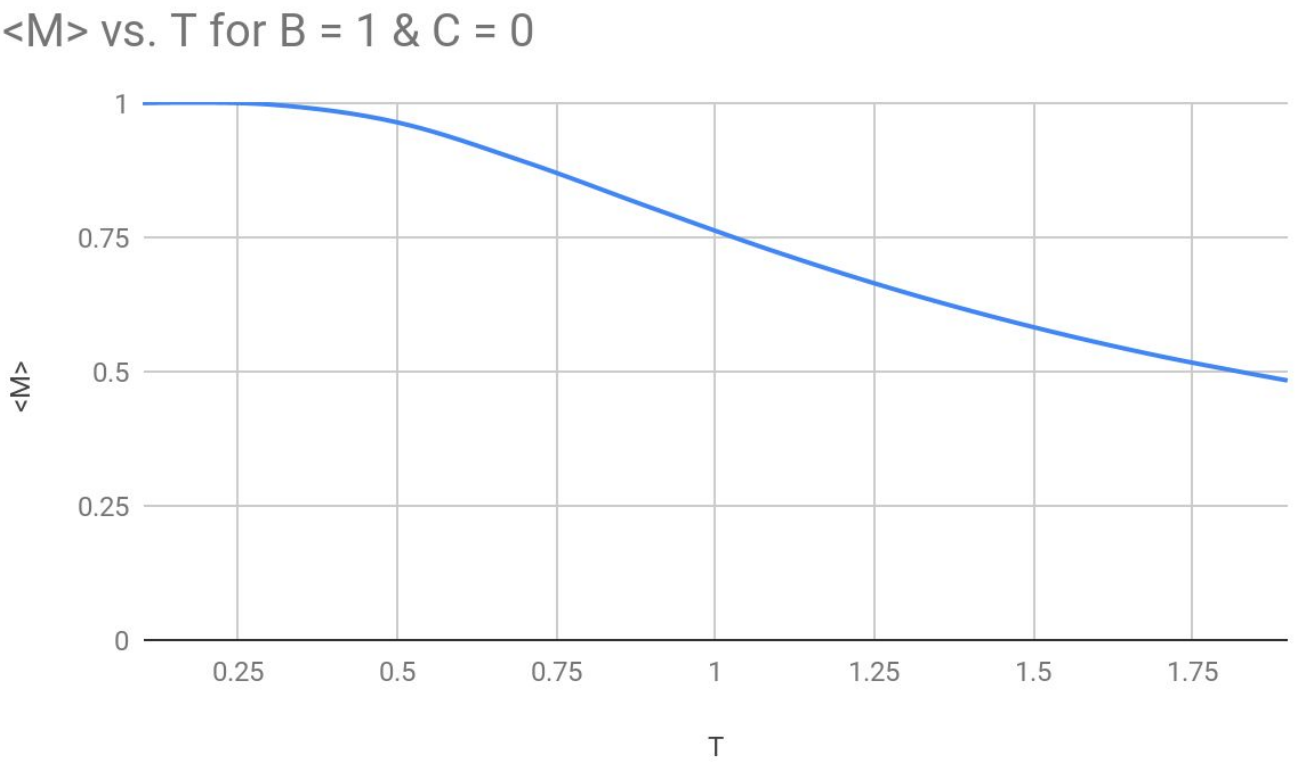
Architecture: The program will be implemented in the Java programming language with a class for the Metropolis algorithm and its methods (Metropolis.java), a class for handling Multithreading (Multithreading.java) and finally a main class (Driver.java) in which the program begins execution, i.e the threads start execution.

Driver -> Multithreading -> Metropolis

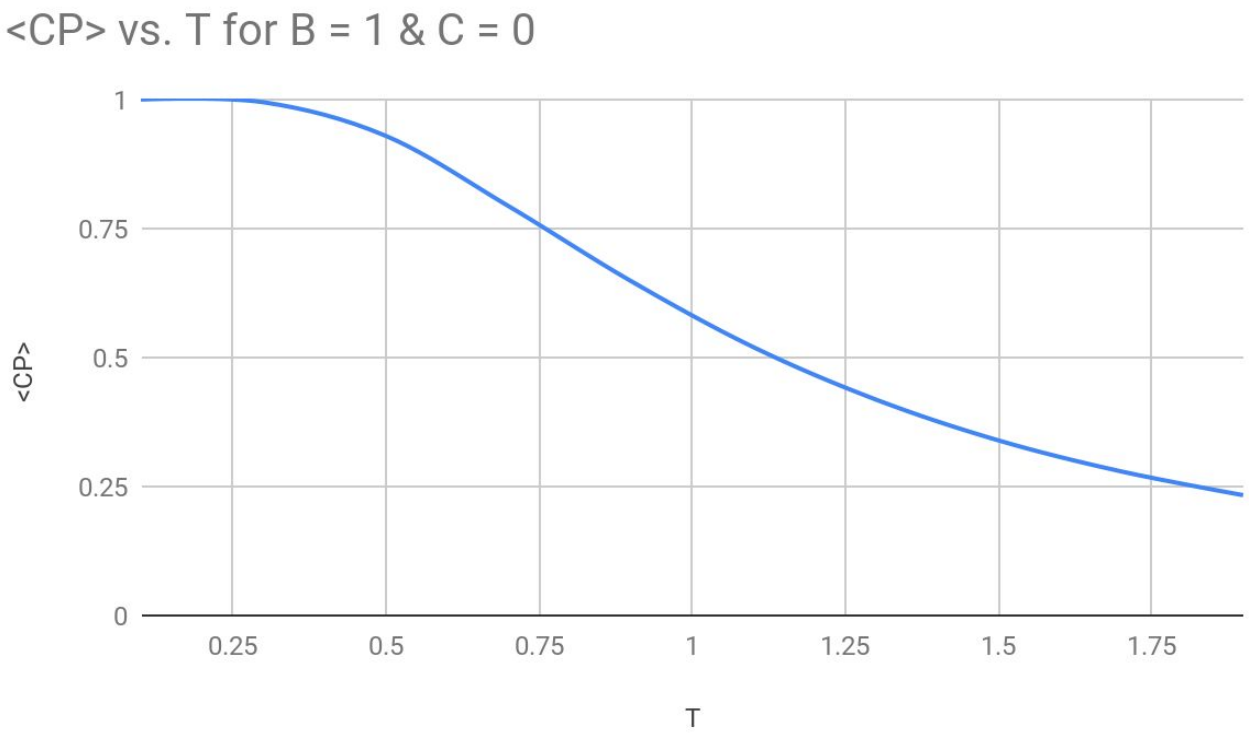
Usage: Compile all three java files and then just run the Driver.
javac Driver.java Metropolis.java Multithreading.java
&& to run:
java Driver

The values of N_M and N_F can be modified by the user and the amount of plot points desired may also be modified by the user, but keep in mind the number of plot points must correspond with the size of mRange and cRange which are arrays storing the global means of mps and pcps for each temperature value, else you will get an index out of bound execution. Currently if compiled the program will plot 3 points. If changing the number of plots make the changes in line 10 & 11 (Driver.java) to modify mRange and cRange size. The for loop that controls temp values can be changed at line 23. Also if changing the values for temp cRange, mRange, be sure to modify the last block of code printing to data points to console line 201 (Driver.java)

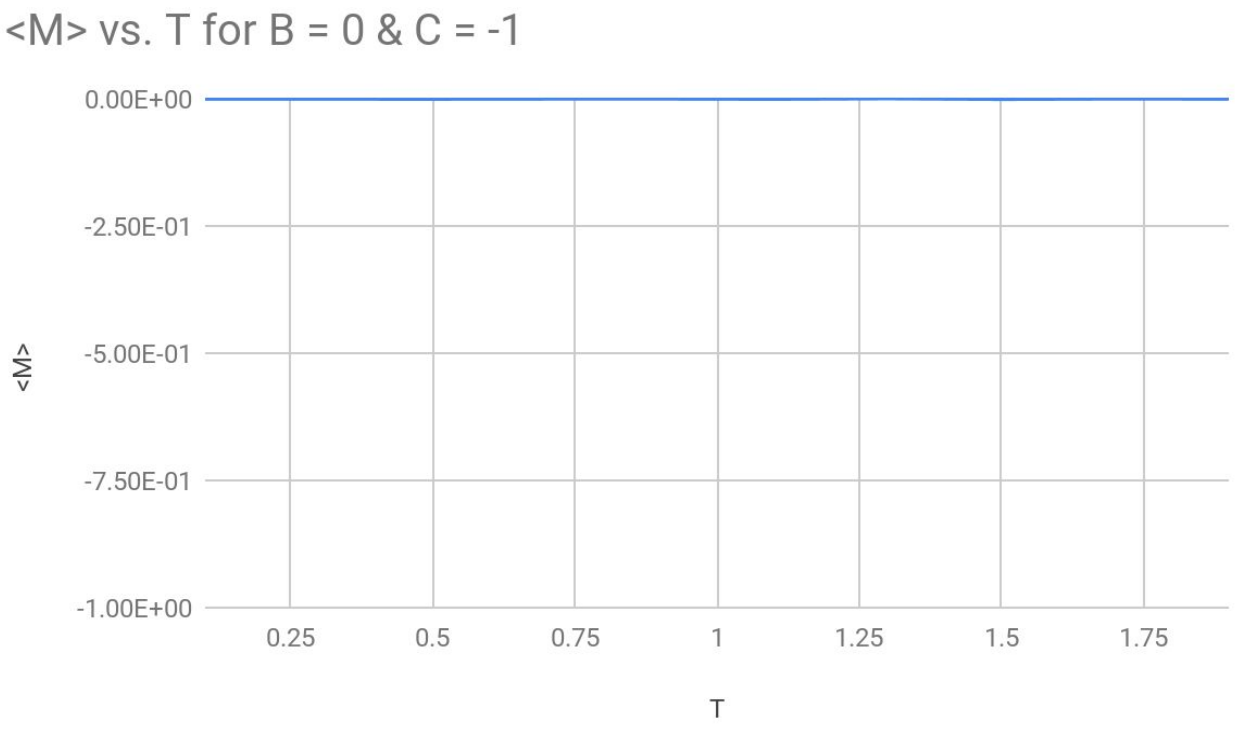
Graphs and Tables



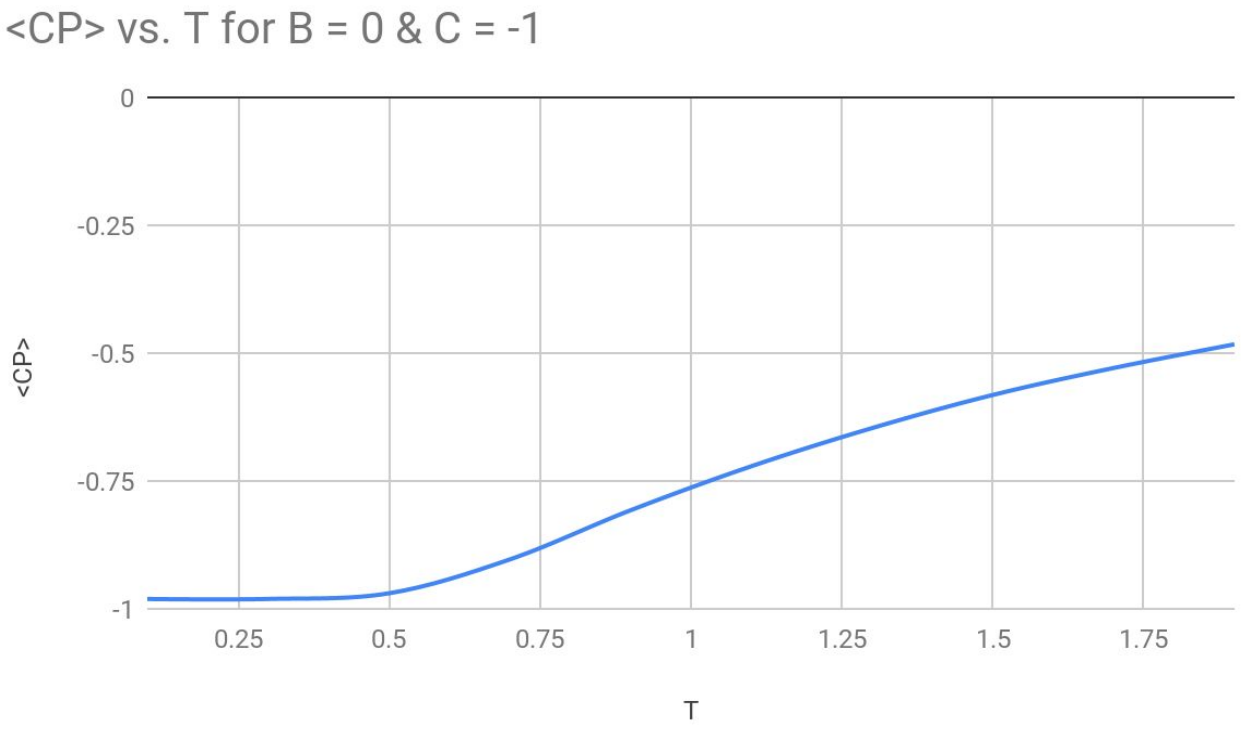
T	<M>
0.1	1
0.3	0.9973466667
0.5	0.9641146667
0.7	0.8908293333
0.9	0.8048706667
1.1	0.721136
1.3	0.646848
1.5	0.5828253333
1.7	0.5284706667
1.9	0.4833373333
B,C Values:	**B = 1, C=0**



T	<CP>
0.1	1
0.3	0.994704
0.5	0.9294986667
0.7	0.793624
0.9	0.648328
1.1	0.5201733333
1.3	0.4185626667
1.5	0.339432
1.7	0.2794186667
1.9	0.2331893333
B, C Values:	**B = 1, C=0**



T	<M>
0.1	2.53E-05
0.3	-3.07E-05
0.5	-1.39E-04
0.7	1.88E-04
0.9	2.61E-04
1.1	-2.40E-04
1.3	8.01E-04
1.5	-4.57E-04
1.7	3.37E-04
1.9	6.27E-05
B,C Values:	**B= 0, C=-1**



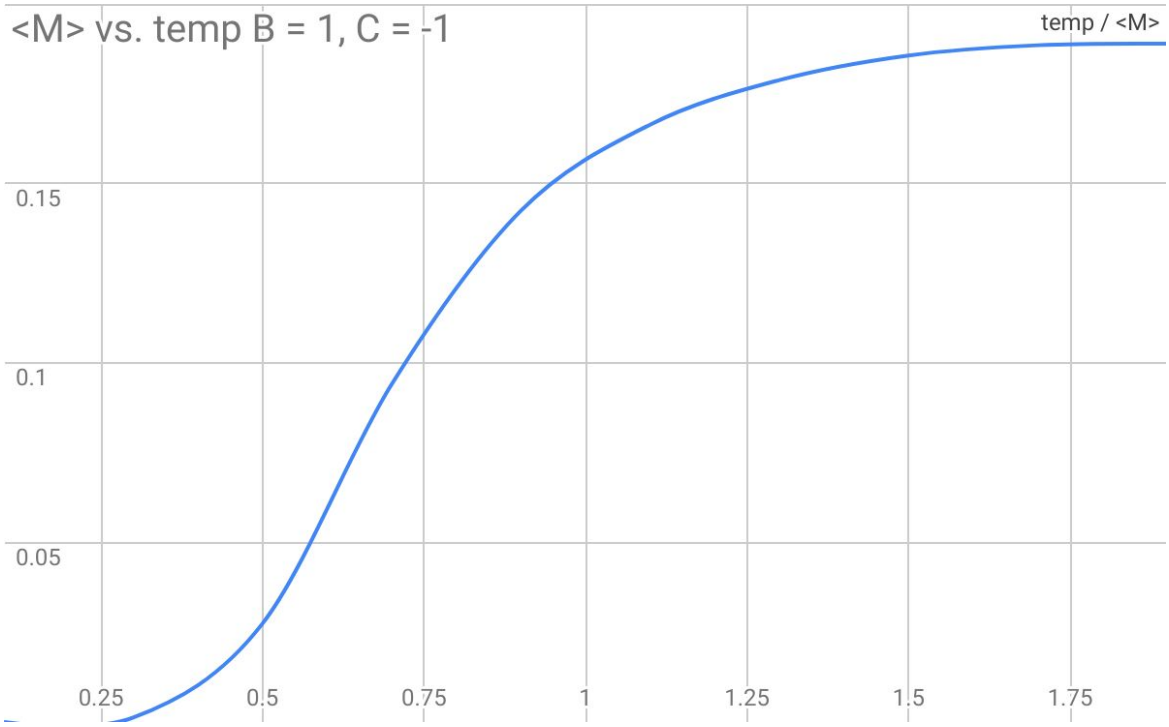
T	<CP>
0.1	-0.980784
0.3	-0.9806933333
0.5	-0.9696293333
0.7	-0.9034773333
0.9	-0.807384
1.1	-0.7212826667
1.3	-0.646632
1.5	-0.5818533333
1.7	-0.529152
1.9	-0.482896
B,C Values:	B=0, C=-1

Challenge 1:
I found the following values for B & C to match Figure 3 in the specs closely:
B = 0.5
C = -0.5

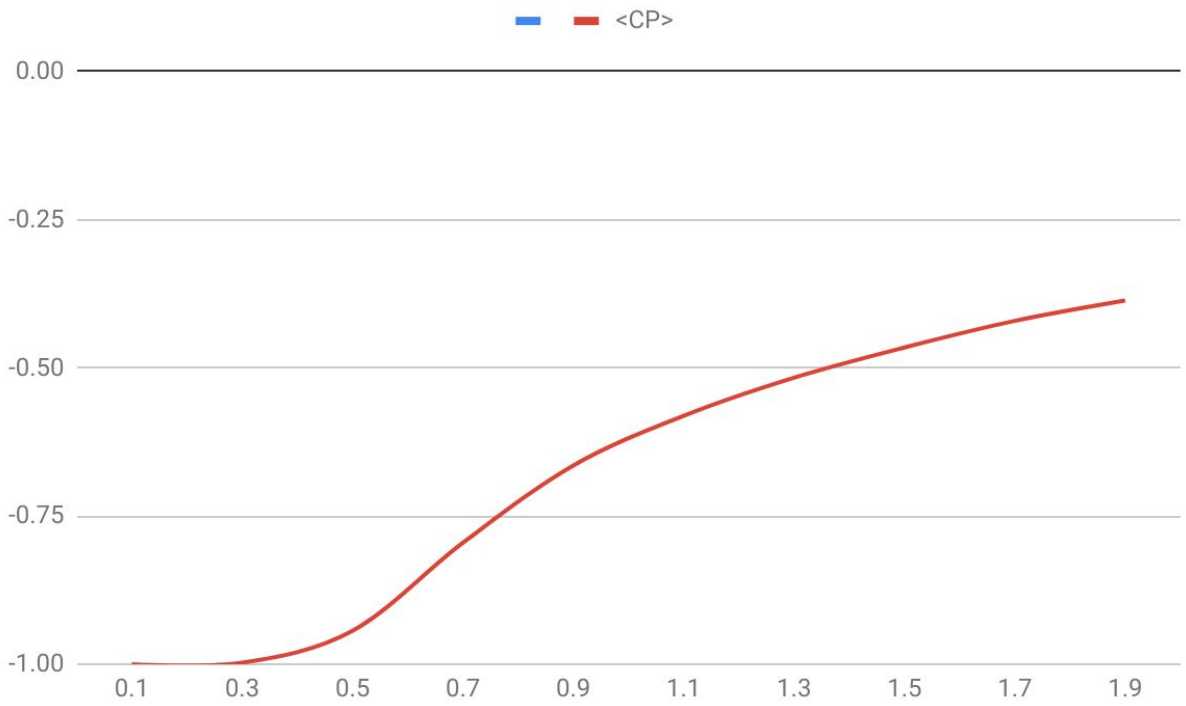
By using the Metropolis Algorithm implemented in my program and using:
$N_M = 15$
$N_F = 20$
Number of Spins = 100
Number of Threads = 1000
Temperature Range -> 0.01 → 1.96 (Incrementing by 0.05)
Total of 40,000 Threads looping through the temperature range to get the following data for B = 0.5 && C = -0.5

I was able to plot the data and through experimentation I found that when B > 0 && <= 2 and when C < 0 && >= -2, but both B and C were complete opposites of each other, then the shape desired starts to be exhibited

For example, when B = 1 and C = -1 the graph for <M> begins to resembles the graph from figure 3 in the specs



As does the graph for <CP>



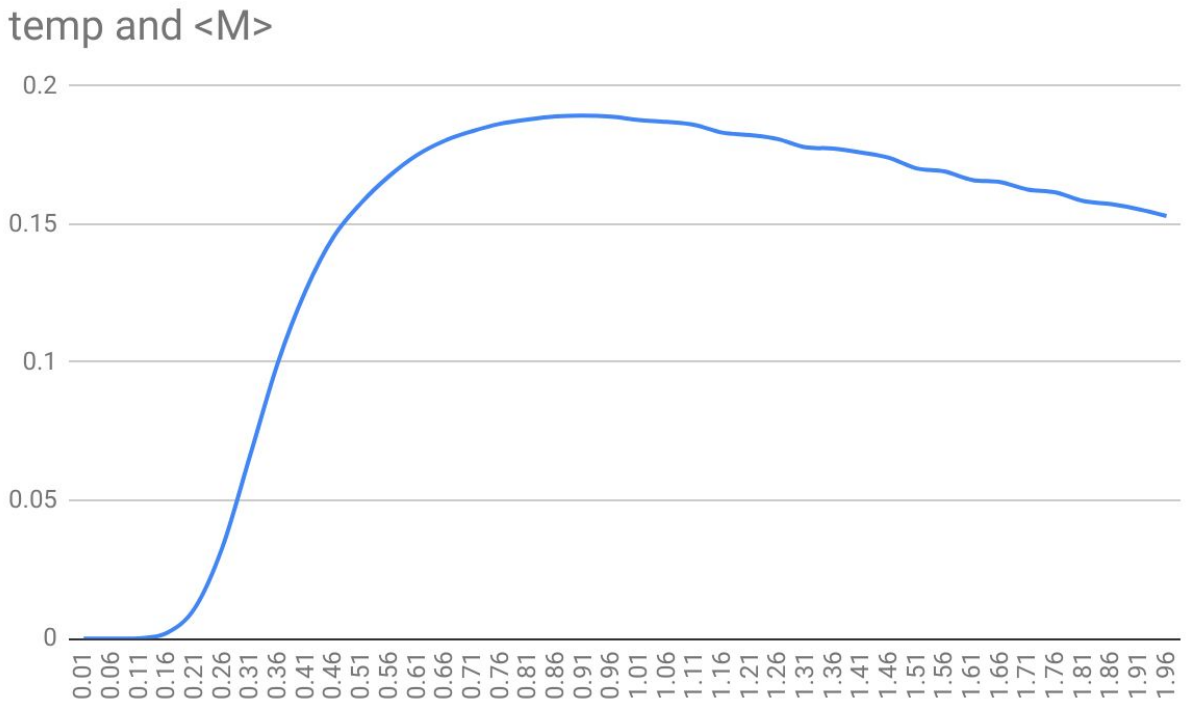
And so I found that at B = 0.5 and C = -0.5 the graphs most closely matched Figure 3.

The following is my chart and graph for B = 0.5 && C = -0.5

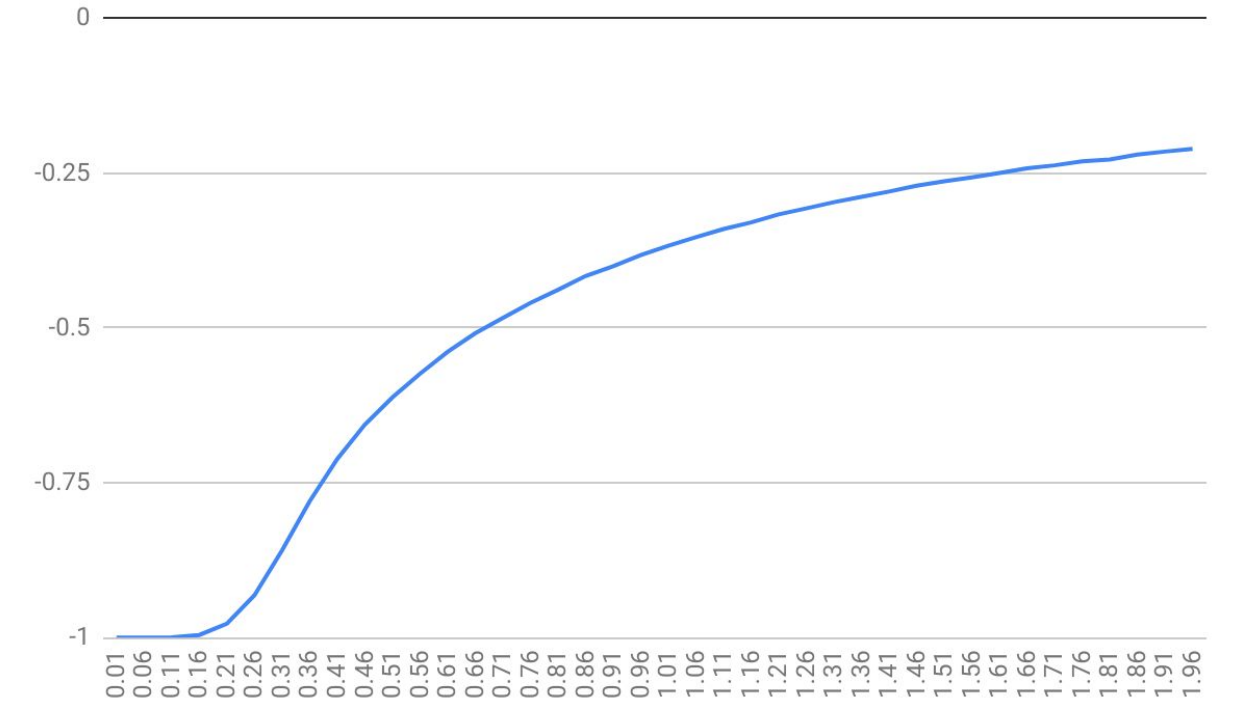
temp	<M>	<CP>
0.01	0	-1
0.06	0	-1
0.11	0.00010533	-0.99978933
0.16	0.00204	-0.99591467
0.21	0.01095333	-0.97784533
0.26	0.033096	-0.93190933
0.31	0.06640267	-0.85945333
0.36	0.09998133	-0.780336
0.41	0.12606933	-0.711936
0.46	0.14540533	-0.65644267
0.51	0.15776933	-0.612456
0.56	0.16738933	-0.574248
0.61	0.17499333	-0.538888
0.66	0.180224	-0.50910133
0.71	0.18369067	-0.484248
0.76	0.18635733	-0.459944
0.81	0.187896	-0.438792
0.86	0.18902133	-0.41657067
0.91	0.189356	-0.40074933
0.96	0.18893733	-0.38292267
1.01	0.18774667	-0.36771467
1.06	0.187044	-0.35397867
1.11	0.185956	-0.34061333
1.16	0.18316667	-0.329936
1.21	0.18225467	-0.31699733
1.26	0.180836	-0.3074
1.31	0.17782933	-0.297384
1.36	0.177428	-0.28889333
1.41	0.17596933	-0.280184
1.46	0.17405067	-0.270616

1.51	0.17021467	-0.26377333
1.56	0.16912	-0.257432
1.61	0.16605867	-0.250128
1.66	0.165268	-0.2428
1.71	0.16258533	-0.23790133
1.76	0.161528	-0.23151733
1.81	0.15843733	-0.22849867
1.86	0.157276	-0.220368
1.91	0.15544533	-0.21569067
1.96	0.15299333	-0.21110133

The peak occurs at:
T = 0.91 with a maximum <M> value of 0.18902133 at B = 0.5 && C = -0.5



Graph of Temp vs. <CP> at B = 0.5 && C = -0.5



Challenge #2:

I found the following values for B & C to match Figure 3 in the specs closely:

B = 0.2

C = -0.08

By using the Metropolis Algorithm implemented in my program and using:

$N_M = 15$

$N_F = 20$

Number of Spins = 100

Number of Threads = 1000

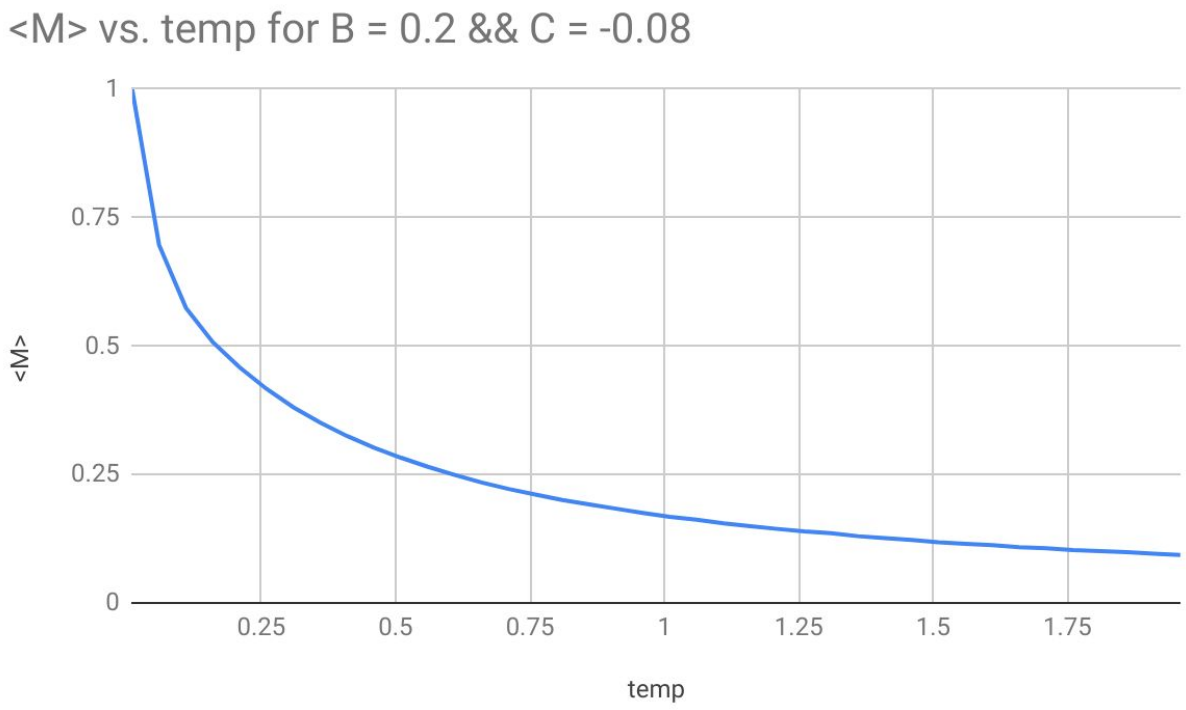
Temperature Range -> 0.01 → 1.96 (Incrementing by 0.05)

Total of 40,000 Threads looping through temperature range to get the following data for B = 0.2 && C = -0.08

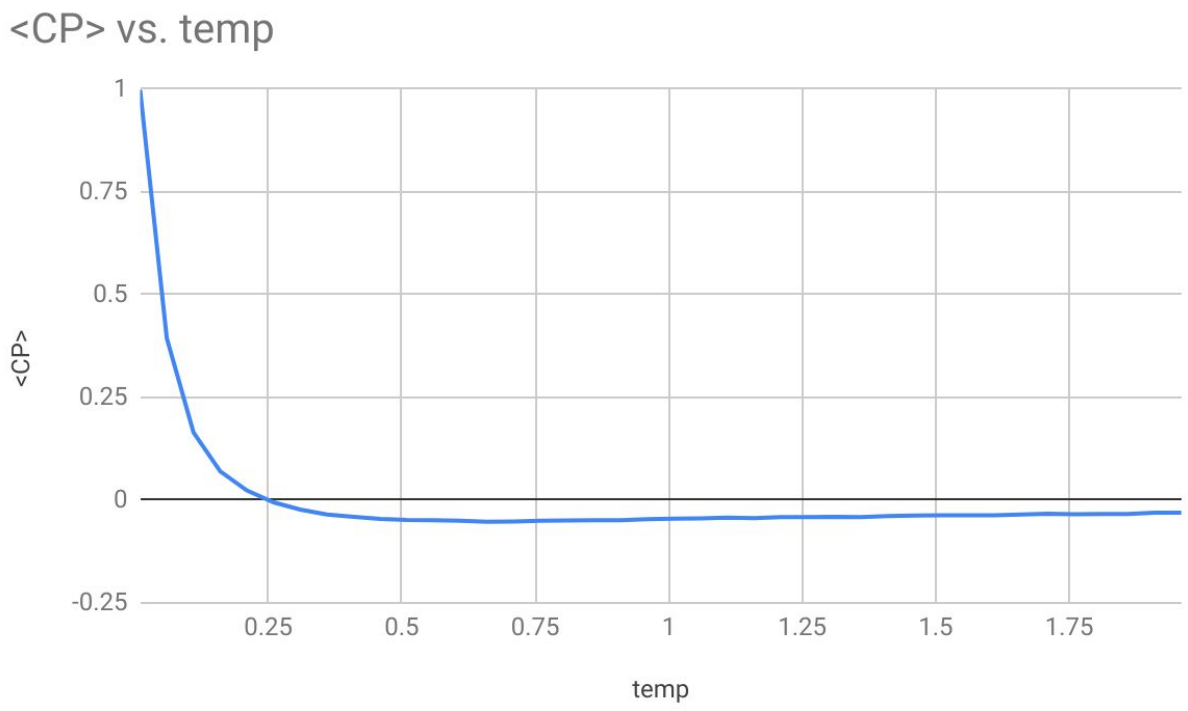
Graphs and Charts for B = 0.2 && C = -0.08

temp	<M>	<CP>
0.01	0.999352	0.998704
0.06	0.69676933	0.39417333
0.11	0.57421467	0.16497067
0.16	0.507264	0.07005067
0.21	0.45833867	0.023496
0.26	0.416408	-0.00536533
0.31	0.38063333	-0.023064
0.36	0.35077867	-0.034896
0.41	0.32459733	-0.04049067
0.46	0.30215733	-0.04584
0.51	0.28258933	-0.048232
0.56	0.26489333	-0.04877867
0.61	0.24897733	-0.05034133
0.66	0.23428533	-0.05285333
0.71	0.221772	-0.05174133
0.76	0.21080933	-0.05008267
0.81	0.20037067	-0.04934133
0.86	0.19181067	-0.04860533
0.91	0.18331467	-0.04868
0.96	0.175128	-0.046584
1.01	0.16732	-0.04479733
1.06	0.162044	-0.04445867
1.11	0.15476	-0.042832
1.16	0.14948667	-0.043704
1.21	0.14396933	-0.04114933
1.26	0.13897733	-0.04134933
1.31	0.135412	-0.04038667
1.36	0.129836	-0.041024
1.41	0.12611467	-0.03871733
1.46	0.12259067	-0.03736267

1.51	0.117828	-0.03702933
1.56	0.11491333	-0.036832
1.61	0.11237333	-0.03675733
1.66	0.10835867	-0.034824
1.71	0.10653467	-0.03330133
1.76	0.10258533	-0.03463467
1.81	0.10091067	-0.03373067
1.86	0.09886667	-0.034008
1.91	0.09571333	-0.030736
1.96	0.09339867	-0.03081333



The minimum value of <C> is exhibited at T = 0.66 with a value of: -0.05285333



Challenge 3:

I found the following values for B & C to match Figure 3 in the specs closely:

B = 1.6

C = -0.8

By using the Metropolis Algorithm implemented in my program and using:

$N_M = 15$

$N_F = 20$

Number of Spins = 100

Number of Threads = 1000

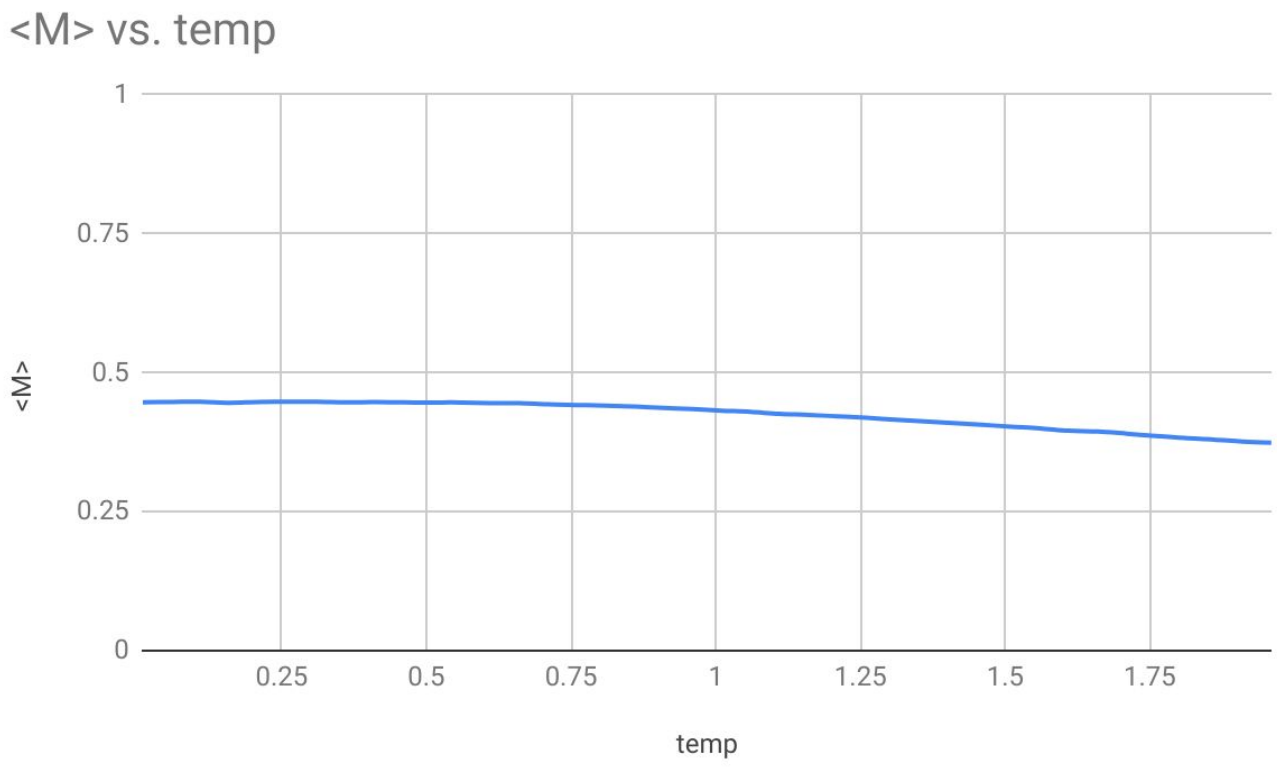
Temperature Range -> 0.01 → 1.96 (Incrementing by 0.05)

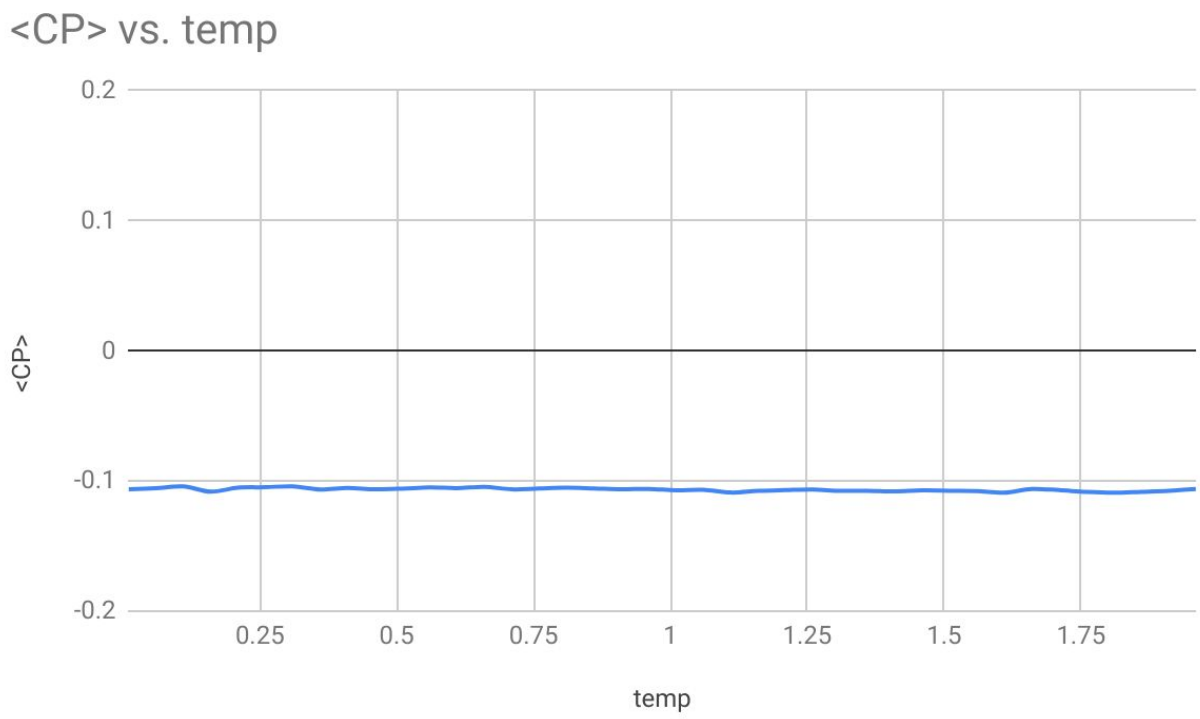
Total of 40,000 Threads looping through temperature range to get the following data for B = 1.6 && C = -0.8

The following are chart and graphs for B = 1.6 && C = -0.8

temp	<M>	<CP>
0.01	0.44682133	-0.10635733
0.06	0.447228	-0.105544
0.11	0.44792	-0.10416
0.16	0.445904	-0.108192
0.21	0.44742	-0.10516
0.26	0.44759467	-0.1048
0.31	0.447936	-0.104104
0.36	0.44669467	-0.10653067
0.41	0.44718133	-0.10534933
0.46	0.446496	-0.10637067
0.51	0.446432	-0.10589867
0.56	0.44639067	-0.10494667
0.61	0.44539467	-0.10552533
0.66	0.44515467	-0.104456
0.71	0.44293867	-0.10643733
0.76	0.4419	-0.10577333
0.81	0.440832	-0.10516267
0.86	0.43888667	-0.10572533
0.91	0.43656933	-0.10640267
0.96	0.43449067	-0.10610667
1.01	0.431756	-0.107128
1.06	0.42954267	-0.106848
1.11	0.425776	-0.10887733
1.16	0.42406267	-0.10758133
1.21	0.42145467	-0.10699733
1.26	0.418724	-0.10658667
1.31	0.415352	-0.107528
1.36	0.41245333	-0.10754933
1.41	0.409104	-0.108016

1.46	0.40624533	-0.10726133
1.51	0.40283467	-0.107528
1.56	0.39998533	-0.107736
1.61	0.395476	-0.109
1.66	0.39414133	-0.10612
1.71	0.39023333	-0.10704267
1.76	0.38598267	-0.1084
1.81	0.38233467	-0.10901067
1.86	0.37978667	-0.108296
1.91	0.375912	-0.10753333
1.96	0.37384933	-0.106192





Challenge 4:

Using the data from challenge 1, the highest peak value of <m> was found to be 0.18902133 at T = 0.91

Challenge 5:

Using the data from challenge 1, the lowest peak value of <c> was found to be -0.05285333 at T = 0.66
