Name: Wilmer Henao

Optimizations used.

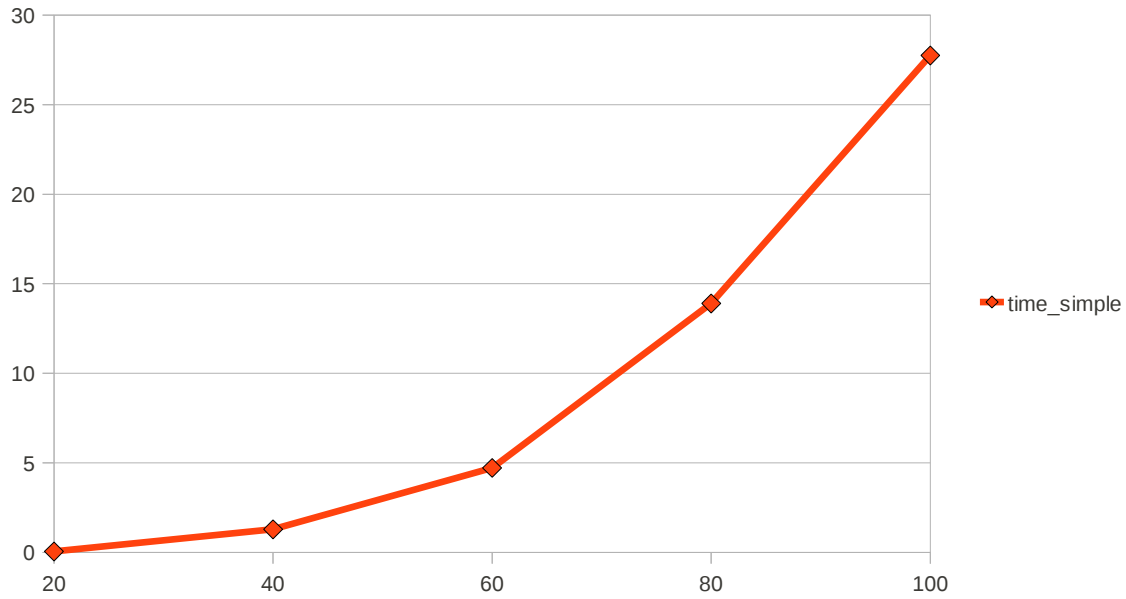
1) Jacobi
2) Gauss-Seidel: Runs in around 70% of the time
3) Parallelizing: It increases speed somewhere between 2 and 6.5 times faster, depending on the size of the problem
4) SOR algorithm: Increases speed at a very high order (improvement of 68x faster for a 150,150,150 grid) which runs in 11 seconds

ALL OF THE GRAPHS PLOT SIZE ON THE X AXIS AND TIME ON THE Y AXIS. EXCEPT FOR THE SPEEDUP PLOT WHICH PLOTS... SPEEDUP

Step 1:

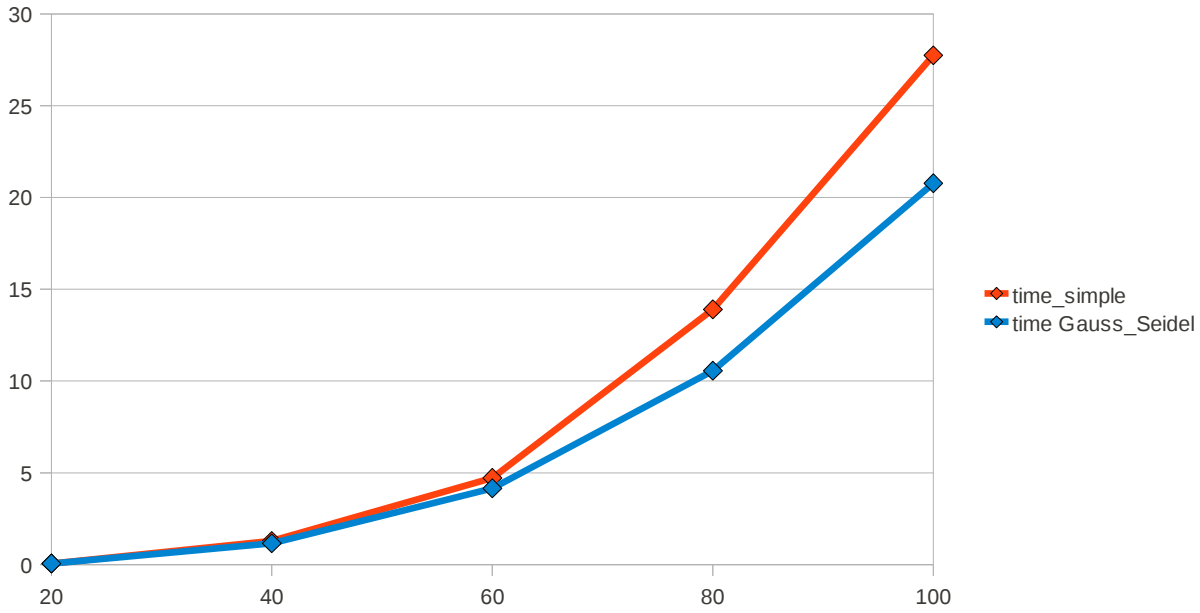
I basically changed the way the inputs work because I thought it would be better if I just automatize everything from the beginning. There's stuff that I just won't optimize (like converting those arguments from char to int). Just because it's a small time.

Commits 1, 2 and 3; This is where I set up everything for my runs. For a while I will just try to beat this benchmark



Here I am plotting 20, 40, 60, 80 and 100 grids (I am using perfect partitions only ex. {100, 100, 100} for simplicity.

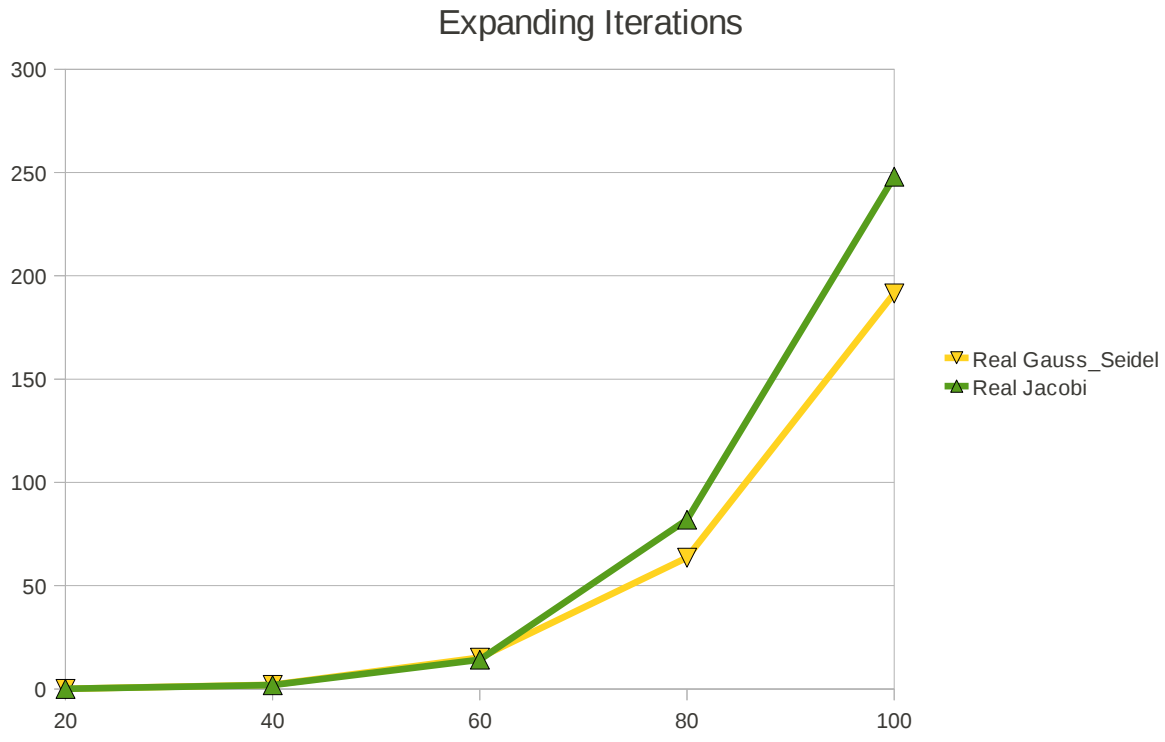
Commit 4; The first thing I try to implement is Gauss-Seidel instead of Jacobi. The problem is that I spent a lot of time looking for an optimal algorithm and in the end I had to settle for a double set of for cycles. That is, first I solve everything for the “red” cells and then I solve everything for the “black” cells. The good thing is that in the process I don't need to use a U_OLD vector (which is good). But I end up using a “red-black” char vector (which is bad). In the end, however I improve performance a little bit, specially for bigger matrices. I'm sure for huge matrices the improvement is more.



Not only that but the maximum error (after the limit runs out) is smaller 6.45318e-01 vs. 3.98618e-01

Still huge for the 100 by 100 case. So I might want to run it again with a higher limit of iterations.

After increasing the limit of iterations to the order of the thousands I arrive to



So now. Gauss-Seidel is better. I want to go further and create an SOR function, unfortunately something doesn't work and I have to replace the omega constant with = 1. I still have a Gauss-Seidel but I figure that if I have time later, I can just play with the omega and see if I can improve the solution.

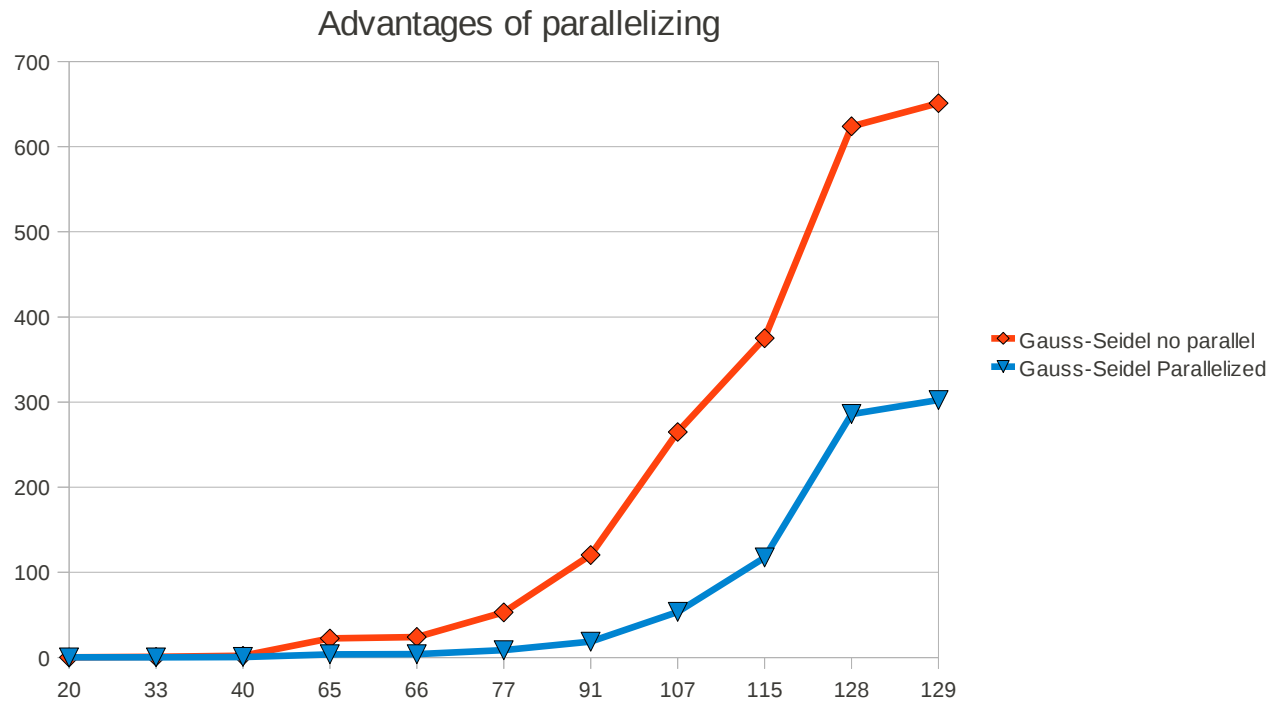
Parallelizing.

The first real problem that I encounter parallelizing is that there is no reduction for maximum in C. Only in fortran, so I switched norms to the euclidean norm instead of the maximum for ease. The first trials are a complete failure and I decide to clean the code a little bit from a serial point of view. I improve my serial code a little bit more by optimizing the entries to my red-black algorithm. I commit right at this point.

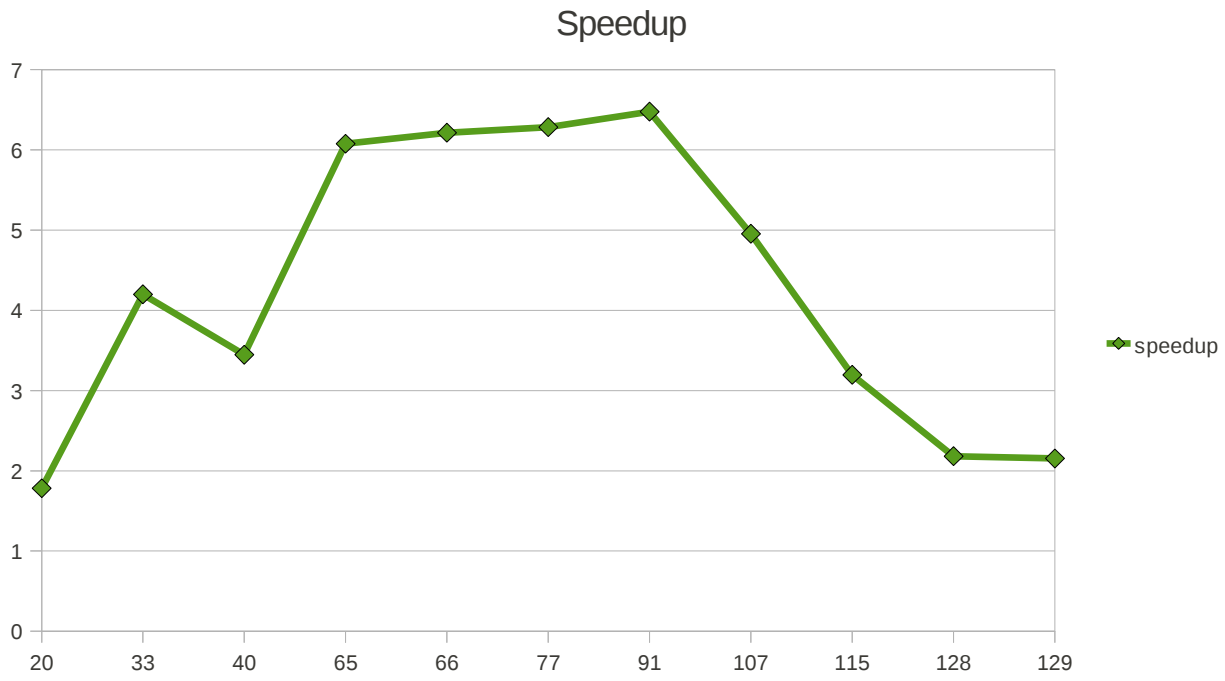
OVERHEAD:
The first thing I notice is an incredible amount of overhead.

I try to work around this problem and arrive at a solution. 7 threads seem to be better for some reason. I decide that I should run a different sample of points to check for the speedup (which of course should be less than 7.

This is the graph of the time



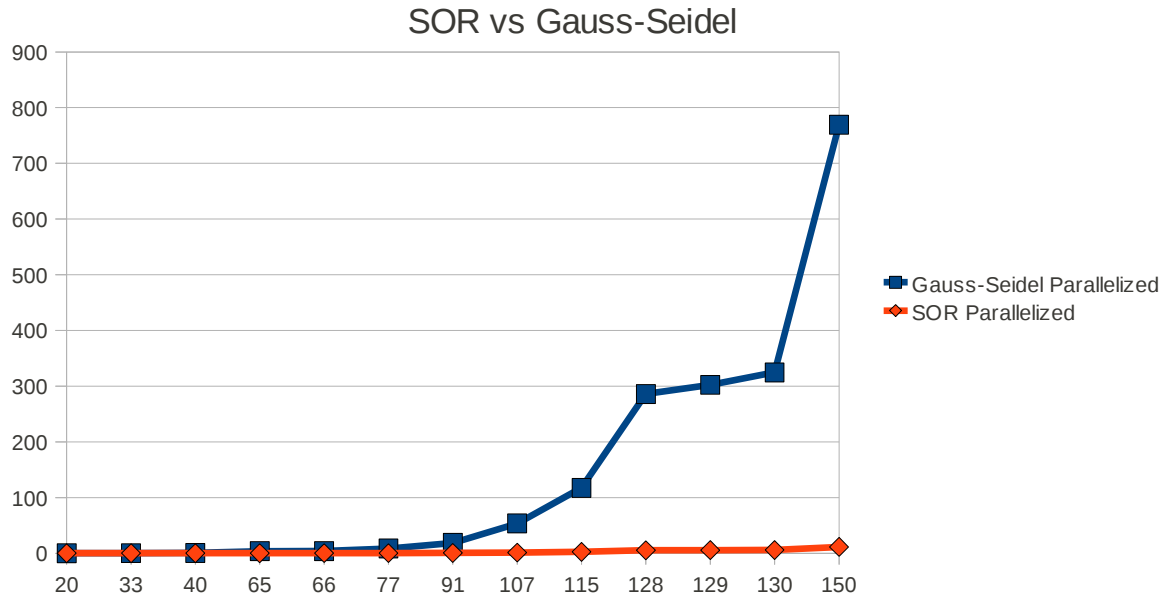
And this is a plot of the speedup



The speedup seems to be highest around 66 cubed steps. And worse around 129 for some reason that I don't know. There is a huge variation in this speedup which I was not expecting before I ran these examples. The good thing is that there seems to always be at least some improvement.

SOR algorithm (the Biggest improvement):

I tried parallelizing other places in the code and as a matter of fact, I included most of the changes in the final version. But the biggest increase in performance was the correct implementation of SOR algorithm. I found the SOR algorithm in the book “Numerical Analysis of Differential Equations” by Arieh Iserles. Chapter 12. It requires a factor “omega” that in Gauss-Seidel equals 1. But varies bet. 1 and 2 for SOR. The optimal value obeys a trigonometric formula and is usually around 1.9 (I also optimize this value). Here's the graph of this improvement and the table (after playing with the flags which doesn't do much.



The order of improvement is huge. At the end of this document I show a run, to prove that not only it runs fast. It converges at levels close to machine epsilon! (see appendix).

Finally I play a little bit with the flags. Not a lot of fun so I don't care much what I get. This is a table with some results from my latest runs.

size	Gauss-Seidel no parallel	Gauss-Seidel Parallelized	SOR Parallelized	SOR with Flags	SOR with Flags last
20	0.060255	0.033812	0.012798	0.012417	0.012224
33	0.730437	0.173895	0.012014	0.012524	0.011925
40	1.9457	0.564652	0.031813	0.032782	0.031719
65	22.2293	3.65775	0.133105	0.134602	0.130108
66	24.0339	3.86748	0.138141	0.14055	0.135632
77	53.0365	8.44078	0.258398	0.261529	0.251105
91	120.363	18.5835	0.474929	0.486697	0.479242
107	264.863	53.4577	1.12252	1.1458	1.11669
115	375.054	117.326	2.32933	2.34163	2.31249
128	623.903	285.898	5.21816	5.25684	5.30338
129	651.145	302.352	5.48659	5.50213	5.56844
130		324.702	5.78531	5.80504	5.7968
150		769.316	11.3437	11.2714	11.3678

Finally this is what I use to compile
icc -O2 -align -Zp8 -axP -unroll -openmp -c serial_hw2.c
icc -o xserial serial_hw2.o timing.o -openmp -lm -lrt

Appendix
Last Run:

```
[weh227@compute-0-138 hw2]$ ./xserial 20 33 40 65 66 77 91 107 115 128 129 130 150
Discretizing with 20 20 20 points in x y and z
 Final iteration     64   norm update   6.484490e-15 and using an omega of   1.740580e+00

  max Error in computed soln:     1.49066e-11
  l2 norm of Error on   20 by   20 by   20 grid
  (dx  1.05263e-01 dy  1.05263e-01 dz  1.05263e-01):    7.42412e-09
 Total time:   1.2737000e-02 seconds
Discretizing with 33 33 33 points in x y and z
 iteration    100   norm update     4.0904e-14
 Final iteration    106   norm update   9.390431e-15 and using an omega of   1.831052e+00

  max Error in computed soln:     5.18955e-13
  l2 norm of Error on   33 by   33 by   33 grid
  (dx  6.25000e-02 dy  6.25000e-02 dz  6.25000e-02):    8.54712e-09
 Total time:   1.1981000e-02 seconds
Discretizing with 40 40 40 points in x y and z
 iteration    100   norm update     6.8254e-10
 Final iteration    132   norm update   8.625247e-15 and using an omega of   1.857788e+00

  max Error in computed soln:     1.73813e-12
  l2 norm of Error on   40 by   40 by   40 grid
  (dx  5.12821e-02 dy  5.12821e-02 dz  5.12821e-02):    2.57675e-09
 Total time:   3.1794000e-02 seconds
Discretizing with 65 65 65 points in x y and z
 iteration    100   norm update     9.0543e-05
 iteration    200   norm update     4.5494e-13
 Final iteration    225   norm update   8.451592e-15 and using an omega of   1.909159e+00

  max Error in computed soln:     4.71311e-15
  l2 norm of Error on   65 by   65 by   65 grid
```

```
 (dx  3.12500e-02 dy  3.12500e-02 dz  3.12500e-02):    4.17122e-09
 Total time:   1.3263300e-01 seconds
Discretizing with 66 66 66 points in x y and z
 iteration    100   norm update     1.4314e-04
 iteration    200   norm update     7.2502e-13
 Final iteration    228   norm update   9.889040e-15 and using an omega of   1.910453e+00

  max Error in computed soln:     7.17010e-15
  l2 norm of Error on   66 by   66 by   66 grid
  (dx  3.07692e-02 dy  3.07692e-02 dz  3.07692e-02):    4.50318e-09
 Total time:   1.3765300e-01 seconds
Discretizing with 77 77 77 points in x y and z
 iteration    100   norm update     6.2547e-03
 iteration    200   norm update     8.4153e-11
 Final iteration    268   norm update   9.388757e-15 and using an omega of   1.922585e+00

  max Error in computed soln:     1.37292e-14
  l2 norm of Error on   77 by   77 by   77 grid
  (dx  2.63158e-02 dy  2.63158e-02 dz  2.63158e-02):    4.59003e-09
 Total time:   2.5762000e-01 seconds
Discretizing with 91 91 91 points in x y and z
 iteration    100   norm update     8.1321e-02
 iteration    200   norm update     4.7290e-08
 iteration    300   norm update     7.2012e-14
 Final iteration    318   norm update   8.679337e-15 and using an omega of   1.933972e+00

  max Error in computed soln:     9.41703e-15
  l2 norm of Error on   91 by   91 by   91 grid
  (dx  2.22222e-02 dy  2.22222e-02 dz  2.22222e-02):    4.43916e-09
 Total time:   4.7364100e-01 seconds
Discretizing with 107 107 107 points in x y and z
 iteration    100   norm update     4.4192e-01
 iteration    200   norm update     5.5017e-06
 iteration    300   norm update     1.3546e-12
 Final iteration    372   norm update   9.885476e-15 and using an omega of   1.943475e+00

  max Error in computed soln:     6.02394e-15
  l2 norm of Error on  107 by  107 by  107 grid
  (dx  1.88679e-02 dy  1.88679e-02 dz  1.88679e-02):    4.55143e-09
 Total time:   1.1206230e+00 seconds
Discretizing with 115 115 115 points in x y and z
 iteration    100   norm update     8.1225e-01
 iteration    200   norm update     2.6107e-05
 iteration    300   norm update     4.8168e-11
 Final iteration    399   norm update   9.427566e-15 and using an omega of   1.947269e+00

  max Error in computed soln:     4.12878e-15
  l2 norm of Error on  115 by  115 by  115 grid
  (dx  1.75439e-02 dy  1.75439e-02 dz  1.75439e-02):    4.37644e-09
 Total time:   2.3270590e+00 seconds
Discretizing with 128 128 128 points in x y and z
 iteration    100   norm update     1.7899e+00
 iteration    200   norm update     3.9196e-04
 iteration    300   norm update     1.4757e-08
 iteration    400   norm update     1.9441e-13
 Final iteration    440   norm update   9.749474e-15 and using an omega of   1.952456e+00

  max Error in computed soln:     2.35490e-15
```

```
 l2 norm of Error on  128 by  128 by  128 grid
 (dx  1.57480e-02 dy  1.57480e-02 dz  1.57480e-02):    4.36482e-09
 Total time:   5.2153050e+00 seconds
Discretizing with 129 129 129 points in x y and z
 iteration    100   norm update     1.8865e+00
 iteration    200   norm update     4.7560e-04
 iteration    300   norm update     1.9510e-08
 iteration    400   norm update     2.9080e-13
 Final iteration    443   norm update   9.732924e-15 and using an omega of   1.952813e+00

  max Error in computed soln:     1.77323e-15
 l2 norm of Error on  129 by  129 by  129 grid
 (dx  1.56250e-02 dy  1.56250e-02 dz  1.56250e-02):    4.36869e-09
 Total time:   5.4836860e+00 seconds
Discretizing with 130 130 130 points in x y and z
 iteration    100   norm update     1.9868e+00
 iteration    200   norm update     5.7603e-04
 iteration    300   norm update     2.5386e-08
 iteration    400   norm update     4.2035e-13
 Final iteration    446   norm update   9.690365e-15 and using an omega of   1.953164e+00

  max Error in computed soln:     2.44272e-15
 l2 norm of Error on  130 by  130 by  130 grid
 (dx  1.55039e-02 dy  1.55039e-02 dz  1.55039e-02):    4.37013e-09
 Total time:   5.7823680e+00 seconds
Discretizing with 150 150 150 points in x y and z
 iteration    100   norm update     4.7676e+00
 iteration    200   norm update     1.2222e-02
 iteration    300   norm update     3.7664e-06
 iteration    400   norm update     1.5841e-10
 Final iteration    495   norm update   9.345219e-15 and using an omega of   1.959240e+00

  max Error in computed soln:     7.31530e-15
 l2 norm of Error on  150 by  150 by  150 grid
 (dx  1.34228e-02 dy  1.34228e-02 dz  1.34228e-02):    4.09908e-09
 Total time:   1.1339290e+01 seconds
```