Anthony Stange – Write-up for Jacobian Algorithm

For my choice of algorithm, I followed the Jacobi algorithm exactly from the book except for when I was getting the max diagonals. For the non-sorting I used all the values in the upper right section of the matrix and after using each of the off diagonal values, the program started over with the first off diagonal value until the OffB of the matrix was below the 10^-9. For the sorting, I find the largest off diagonal value and use that to compute the OffB.

Here is a sample output for the sorting version:

```
OffB values for Sort Method:

21.363760400211444
14.261798873955422
8.528754494866389
2.621656704223981
0.1444360297976632
0.10598330132713078
0.038969717515608605
0.019998721429076553
0.01472647023830576
0.010123633236780862
0.00635070708707856
0.004208590563208725
0.002001422802194051
0.001143970902215951
4.7738021627417865E-4
2.3656095063149846E-4
3.5752806578374097E-6
8.828791042380274E-7
8.437102037890984E-8
1.7349973694206066E-8
5.339191910850007E-9
```

Here is a sample output for the non-sorting version:

```
OffB values for No Sort Method:

13.060283249803213
11.108479703284443
7.5469634493473805
9.81345554782288
7.554682014631283
6.967868471002721
4.918348004720417
```

2.743254678326051
0.782770651663454
7.748403870017183
0.360159265865657
0.3231188907256137
0.5006429010137387
0.1992200222617025
0.1961733783185198
0.12166309497745639
0.1211508972622508
0.11984681152452205
0.10346437766561555
0.10260656865653914
0.10142651811080797
0.06438446582321337
0.08248305770787084
0.08650102073914091
0.08794638796052805
0.048123398787951306
0.06394139339257504
0.06696361808096442
0.11535405512066997
0.11933722778790103
0.11628145258153488
0.03853724268418005
0.04754743675305175
0.04342326847786049
0.04442719597624027
0.044288403036425616
0.03746140925086858
0.03113690414674735
0.02542835221326952
0.025990940536497094
0.02507309135035187
0.041412496343766446
0.07646123867492623
0.060953269888775
0.06478794001803795
0.05394349750719633
0.01964249965464772
0.02096479369365458
0.021107552613750618
0.021708832048215502
0.02059674019532419
0.021750882851199638
0.014505956548922722

```
0.01759757326590793
0.029658156975496317
0.12094500401186109
0.11129188565450818
0.010223244102697291
0.0171035219944035
0.0113543847637129
0.007290709369795542
0.004341573028361817
0.006537493734864494
0.005815901541899656
0.006907604598709751
0.012114879652081264
0.007678730098678185
0.0074112177329929314
0.0036709408301834126
0.0025831469184841443
0.0027003560099757994
0.002698102551045529
0.002325004998867353
0.008877183339088707
0.007337573731203883
0.06804188291767944
0.08375016009540848
0.08551110381561543
0.06711030529583972
0.0013953742394273762
0.0014994355123925094
0.0014966542692229114
0.0011653298446604864
5.347859333341144E-4
5.034535850551632E-4
2.2120383409419283E-4
0.001983481612415154
0.002214924742367356
1.1545886309920816E-4
1.1631986161962109E-4
7.441671248082446E-5
7.355001696624988E-5
1.4863896478981679E-4
1.484577042986012E-4
1.5985353739256727E-4
1.6060714851062324E-4
4.3094778407631495E-5
4.484061916227476E-5
2.9155504387759462E-5
```

```
2.9069807113599617E-5
2.9085176733049782E-5
2.9515705724977095E-5
1.6065871454033596E-5
1.7096881227745327E-5
6.157145179432616E-6
5.175093130785283E-6
5.172073638138839E-6
6.514370609493617E-9
6.70732540982351E-9
2.2023950544061883E-9
2.4621227045669095E-9
5.376402274670994E-9
5.36949280536462E-9
```

As you can see from the data and included graphs in the folder, the slope for the non-sorting version is much shallower than the sorting version. This means more iterations to complete the non-sorting step than the sorting one. So, even though the sorting step is more time consuming, it takes less iteration so despite its expense on large matrices, it is still better to use the sorting step.