Linear Programming [CS 521]

Aim –Visualization of Potential-Reduction Algorithm for Diagonal Scaling

Shreepad Patil (sp1467)
Saurabh Singh (ss2716)
Jay Kalyanaraman (jak471)

*Please see the next page for report.

Visualization of Potential-Reduction Algorithm for Diagonal Scaling

a) The zip contains the following files:

1) DiagonalScaling.py
2) createGrid.py
3) potentialReduction.py

b) How to run:

Please follow the steps below to compile and run the code.

1) Cd into the directory where the code is
2) run the code using the following command:
   Python DiagonalScaling.py
3) use the user interface window that pops up to input your 2x2 positive semi definite matrix and press the button "show". Wait for some time for the result.
4) Press the quit button on the interface dialog box to quit the program

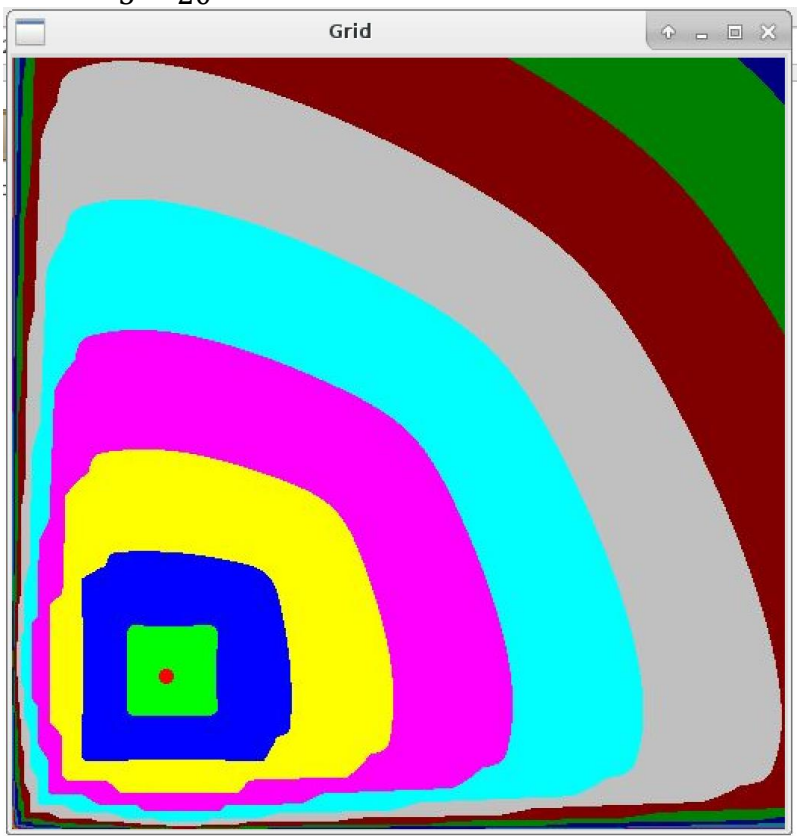
c) Summary

- We try to visually represent the number of iteration the Potential-Reduction algorithm will take to converge.
- The program takes in the matrix Q (a positive semi definite 2x2 matrix) as input.
- We create a grid of size 512x512 with the left bottom corner representing (0,0) and the right top corner representing (1,1). In other words, the square of 1 unit with its left bottom vertex placed at origin is scaled to a 512x512 grid so that image can be better observed
- We provide the Ɛ= .002
- The coordinates of the point we start at represent the point *d* such that *D=diag(d)*. We try to reduce $\| DQDe - e \|$ on every iteration and try to converge on a point *d* such that $DQDe = e$,where *D=diag(d)*.
- We start at every point on this scaled 512x512 grid (that essentially represent a point in the unit square) and count the number of iteration it took for the algorithm to converge starting at that point.
- Different color on the image represent different number of iterations starting with red which represents 1 iteration. Each subsequent color adds one iteration.

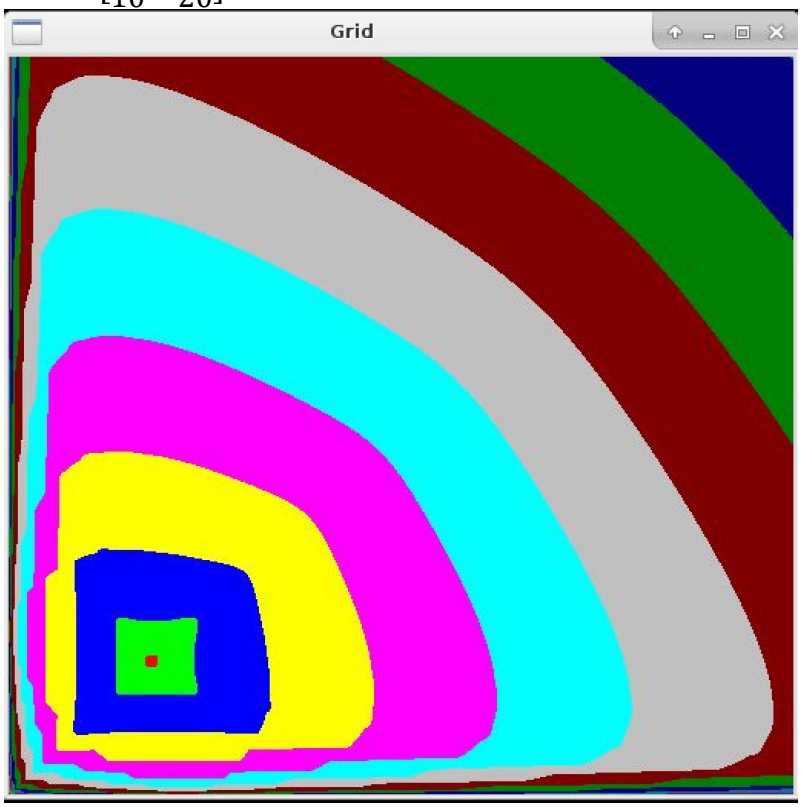
d) Observations:

- We observe interesting outputs when we run the code for different values of Q (the input PSD matrix). Following are the results we obtained:

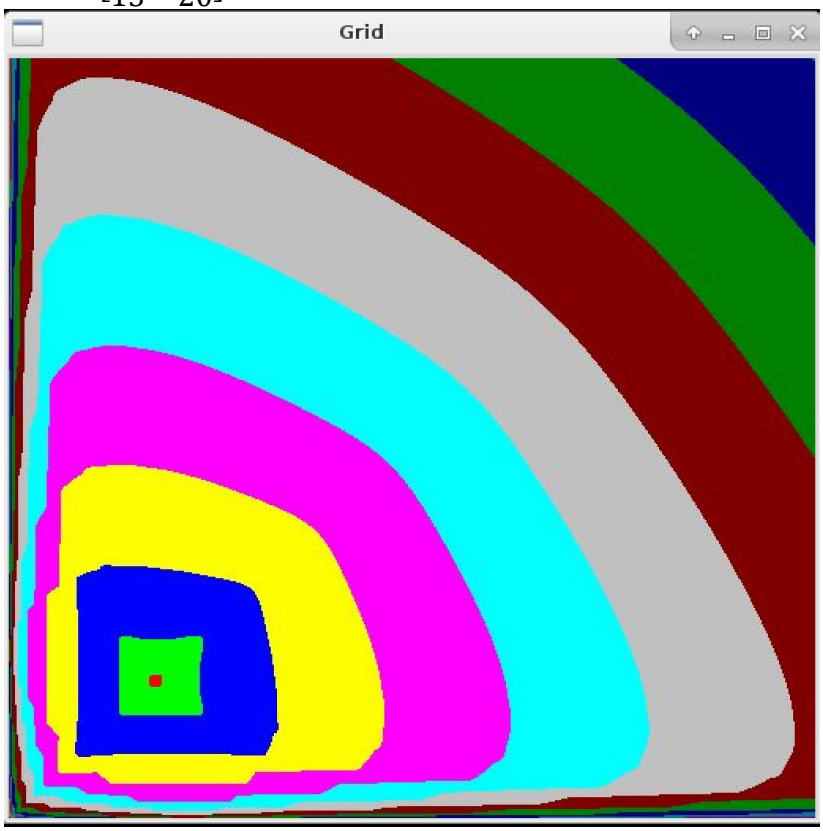
For Q= $\begin{bmatrix} 20 & 5 \\ 5 & 20 \end{bmatrix}$



For Q= $\begin{bmatrix} 20 & 10 \\ 10 & 20 \end{bmatrix}$



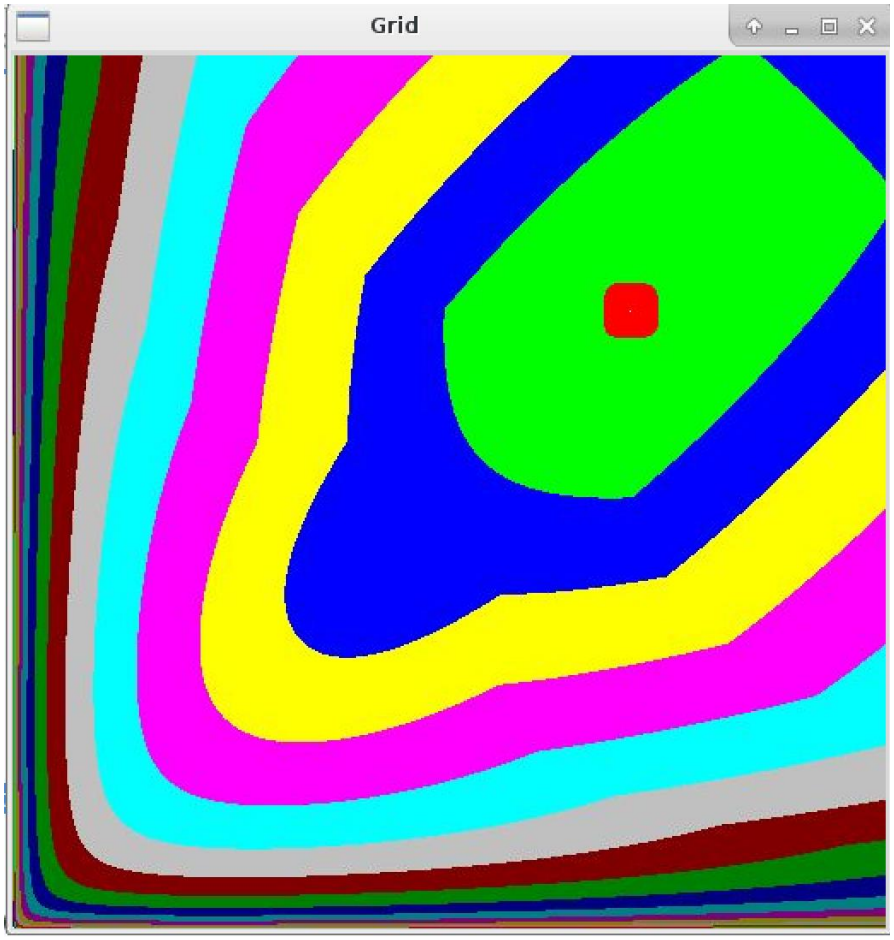
For Q= $\begin{bmatrix} 20 & 15 \\ 15 & 20 \end{bmatrix}$



For Q= $\begin{bmatrix} 20 & 19.5 \\ 19.5 & 20 \end{bmatrix}$

We also plotted for the interesting case where we take negative entries for the non-diagonal elements in the input matrix Q.

For Q= $\begin{bmatrix} 20 & -19.5 \\ -19.5 & 20 \end{bmatrix}$



*Note: The code designed and tested on i-labs machines.

----------The End----------