Projection Tutorial

Tom Wright
University of Oxford

December 6, 2000

This tutorial[1] is designed to take you through an example where the the time needed to create a high-resolution publication quality plot of the pseudospectra of a matrix is significantly reduced by projection onto a carefully chosen invariant subspace.

You must have the Pseudospectra GUI installed on your machine to be able to work through this tutorial. You can download it from `http://www.comlab.ox.ac.uk/oucl/work/tom.wright/psgui/download/`.

The tutorial is organised into five parts:

- Step 1: Set up the data
- Step 2: Select the portion of the complex plane of interest
- Step 3: Begin to introduce projection
- Step 4: Reduce the Safety value still further
- Step 5: Compute the pseudospectra on a fine grid of size 200 by 200

You can download the code used to generate the matrices used in this tutorial from `http://www.comlab.ox.ac.uk/oucl/work/tom.wright/psgui/proj_tutorial/lnt_comp_psa_mtxs.m`. This is taken from Trefethen's paper, Computation of Pseudospectra (Acta Numerica, 1999), which describes projection in greater detail.

[1] Also available electronically at: `http://www.comlab.ox.ac.uk/oucl/work/tom.wright/psgui/proj_tutorial`

- Step 1: Set up the data.

  Generate the matrix B by downloading the code and running it with $N = 200$:

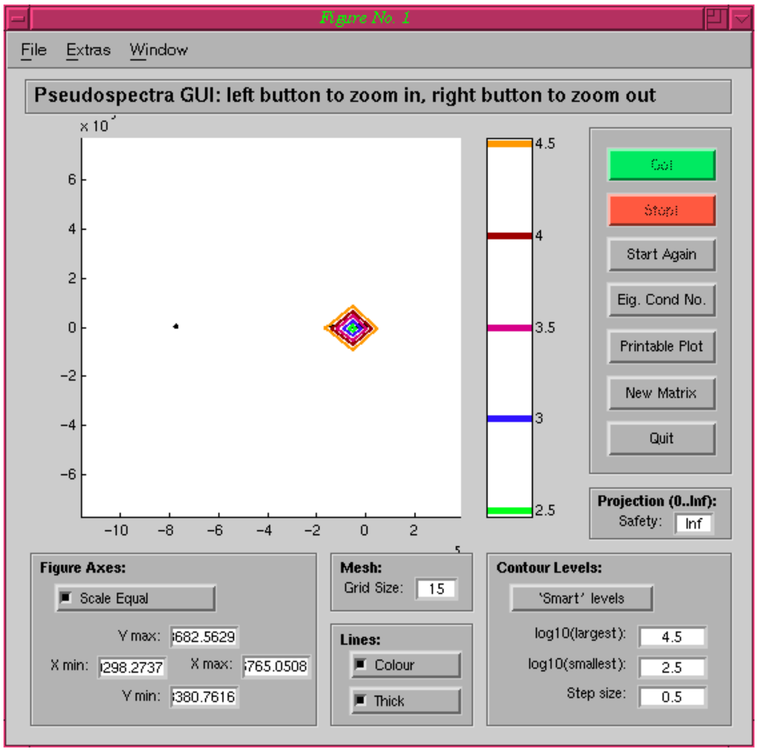
  ```
  B = lnt_comp_psa_mtxs(200);
  ```

  Compute the pseudospectra using the default settings:

  ```
  psa(B)
  ```

  You should end up with something like the following image, which takes about 1 minute to generate on my Sun Ultra 5 workstation.

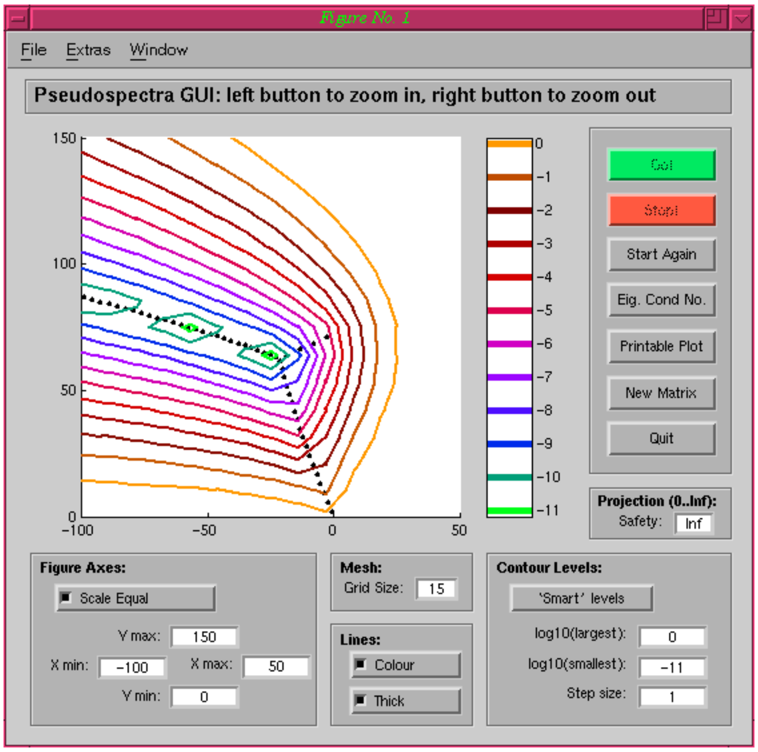
  Note the large magnitudes of the x and y axes. Thes axes are chosen automatically by the Pseudospectra GUI, and are this big because of the outlying eigenvalue near $-80,000$.



- Step 2: Select the portion of the complex plane of interest.

  This plot shows the pseudospectra in a region of the complex plane which contains the entire spectrum of the matrix B. Our interest, however, is in the rightmost eigenvalues, the ones with largest real part. To focus on these, we change the axis limits (using the text boxes towards the bottom left of the GUI) so that the real axis ranges from -100 to 50 and the imaginary axis from 0 to 150.

  After changing these limits, click on 'Go!' to recompute the pseudospectra on this new domain. If you get a message saying that there are no contours to plot in the range specified, click on 'Smart' levels to change the levels plotted. You should end up with something like the following image, which my Sun Ultra 5 takes about 30 seconds to compute:

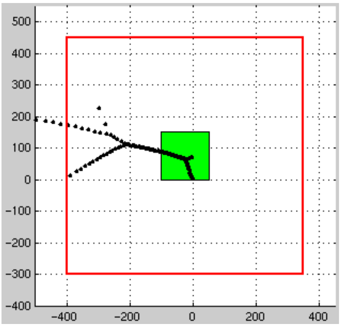


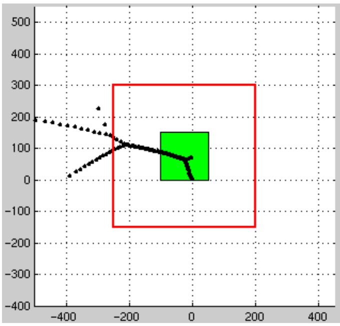
- Step 3: Begin to introduce projection.

  The previous image is perfectly adequate for getting a rough idea of the behaviour of the pseudospectra near the imaginary axis, but it would be better to have a higher quality plot for inclusion in a publication or talk. The problem is that although this plot took only a few seconds to generate, it is only computed on a coarse 15 by 15 grid (with 225 gridpoints in total). To move to a grid of 200 by 200 points to give a high quality plot would entail a wait of nearly an hour and a half on my Sun Ultra 5. This may be feasible for this reasonably small matrix, but we can do better.

  The key is to project our matrix of dimension 200 onto a subspace spanned by the invariant subspace corresponding to the eigenvalues near the region of the complex plane we are interested in. The projected matrix will hopefully have dimension much less than 200, and so the computation will be faster.

  The following figures indicate how our projection algorithm operates. The green square represents the area of the complex plane visible in the GUI (the area our grid is defined over), while the red square represents the eigenvalues whose eigenvectors we project onto. In the left figure, the projection level ('Safety') is set to 2, while in the right one the projection level is set to 1. The red rectangle is defined as the region which is $(1+2b)$ times as high and $(1+2b)$ times as wide as the rectangle visible in the GUI, where b is the Safety value.

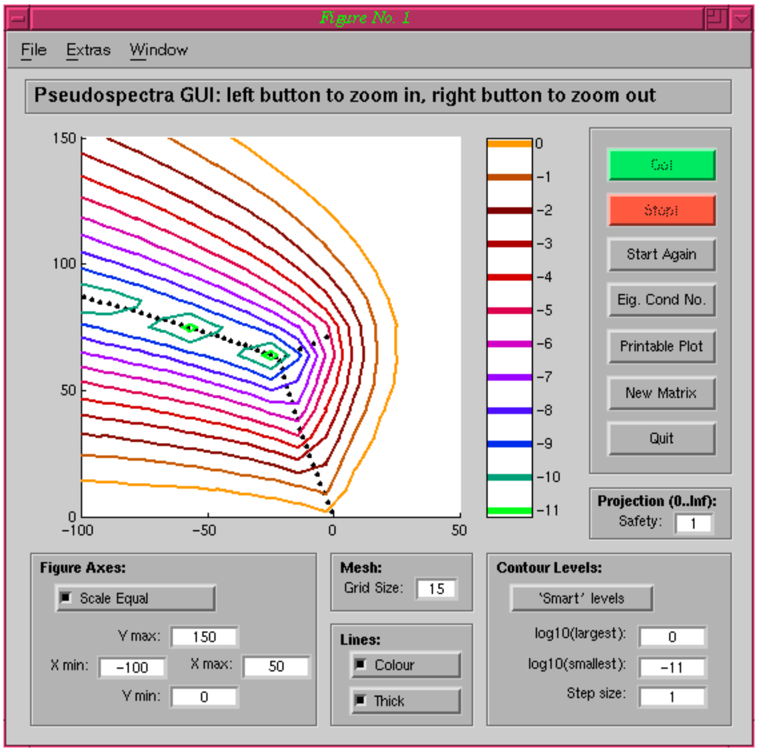




  When you click 'Go!', the matrix is first projected onto the space spanned by eigenvectors whose eigenvalues are within the red square. This is a non-trivial operation, but the benefits will far outweigh this extra cost if we are computing on a fine grid. The pseudospectra of this smaller, projected matrix are then computed. As long as the effect of the eigenvalues we have left out of our projection is small, the plot of the pseudospectra will look essentially identical, and importantly, the computation will be faster.

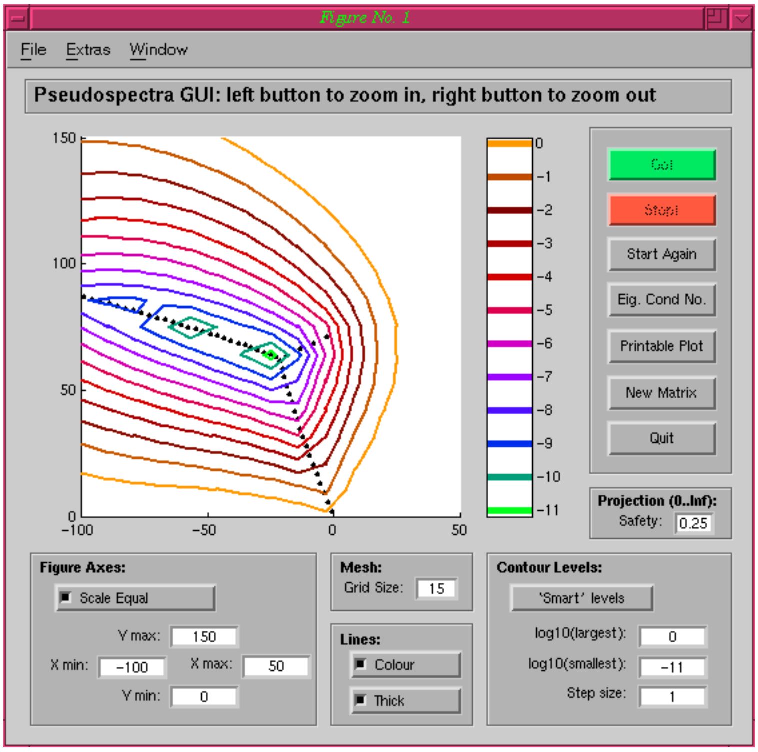
  First, try reducing the 'Safety' value to 2 and recomputing the pseudospectra. The plot you get should look exactly the same as the one obtained before (with infinite safety i.e. no projection). On the coarse grid used here, the computation time is much the same (we have to take account of the time taken to project the matrix), but we will see real differences when we move to the fine grid. This takes about 30 seconds on my Sun Ultra 5 workstation.

Since we saw no difference for Safety=2, now reduce Safety to 1 and re-compute again. Once more, there is no change to the pseudospectra. Your Pseudospectra GUI should now look like this:



- Step 4: Reduce the Safety value still further.

  As there was once again no difference, try reducing Safety again, this time to 0.5. If you look carefully at this plot, you can see that the pseudospectra are not the same as the previous plots at the left edge of the axes. If you reduce Safety still further, to 0.25, you should get the plot below, which again takes about 30 seconds to compute on my computer. The important thing to note here is that the pseudospectra of the projected matrix may be misleading when the invariant subspace that is projected upon makes a small angle with an eigenvector that is omitted from the projection; you must be careful not to project too far.



- Step 5: Compute the pseudospectra on a fine grid of size 200 by 200.

  First of all, set the value of Safety back to 1. Now set the number of gridpoints to 200, and hit the 'Go!' button. The computation takes me about 25 minutes, less than a third of the predicted time without projection. The final figure is shown below, and although the lines appear jagged on the screen, they will be smooth when printed out as a postscript file.