Homework 3 by Maksat Kuanyshbay

Problem 1.

1. For m = 400 and m = 800, respectively:

```
>> poisson
Error relative to true solution of PDE =  9.001e-08
>> poisson
Error relative to true solution of PDE =  2.256e-08
```

Log2 of the error ratio:

```
>> log2(9.001e-08/2.256e-08)

ans =

    1.9963
```

Since we can estimate the order p based on any two calculations for small enough h, the above result is enough to verify that the method is indeed second order accurate.

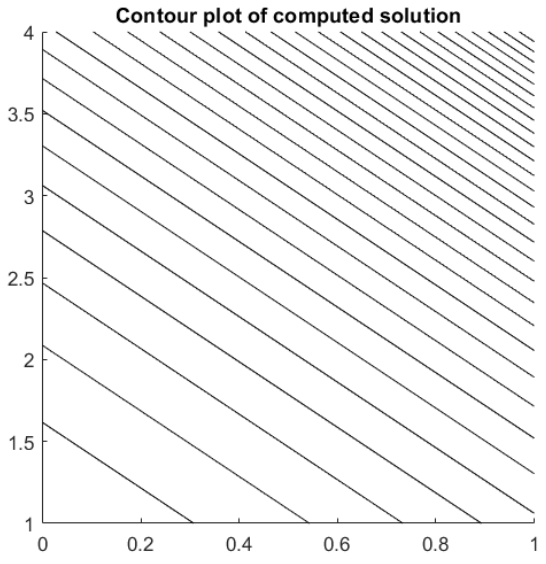
2. For $a_x = 0$, $b_x = 1$, $a_y = 1$, $b_y = 4$:

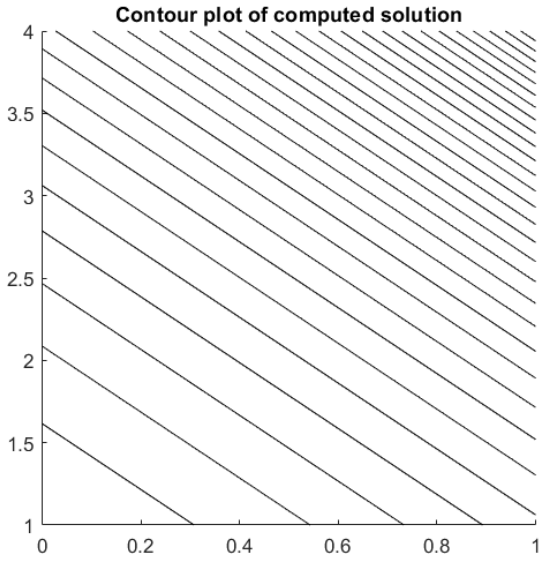
```
>> poisson
Error relative to true solution of PDE =  1.903e-04
```

3. For $a_x = 0$, $b_x = 1$, $a_y = 1$, $b_y = 4$ and m = 20, n = 100:

```
>> poisson
Error relative to true solution of PDE =  1.835e-04
```

Corresponding contour plots for 2 (left) and 3 (right):





Problem 2.

1. Derivation of the truncation error for the 9-point Laplacian:

$$\nabla_9^2 u_{ij} = \nabla_5^2 u_{ij} + \frac{1}{6}h^2 u_{xxyy} + O(h^4)$$
$$= f(x_i, y_j) + \frac{1}{12}h^2(u_{xxxx} + u_{yyyy}) + O(h^4) + \frac{1}{6}h^2 u_{xxyy} + O(h^4)$$
$$= f(x_i, y_j) + \frac{1}{12}h^2(u_{xxxx} + 2u_{xxyy} + u_{yyyy}) + O(h^4)$$
$$= \nabla^2 u + \frac{1}{12}h^2(u_{xxxx} + 2u_{xxyy} + u_{yyyy}) + O(h^4)$$

2. For m = 20 and m = 40, respectively:

```
>> poisson2_2
Error relative to true solution of PDE =  6.014e-09
>> poisson2_2
Error relative to true solution of PDE =  4.140e-10
```

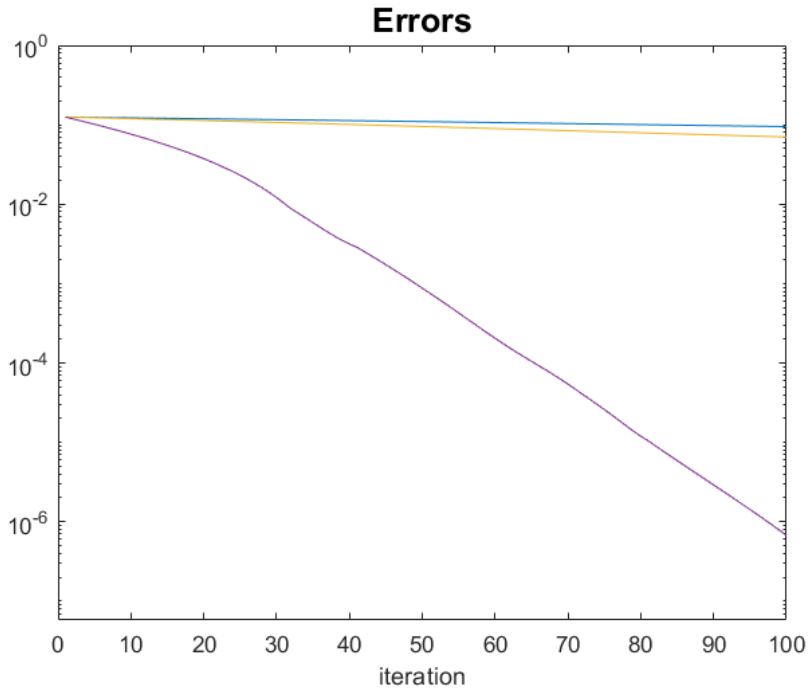
Log2 of the error ratio:

```
>> log2(6.014e-09/4.140e-10)

ans =

    3.8606
```

Since we can estimate the order p based on any two calculations for small enough h, the above result is enough to verify that the method is indeed fourth order accurate.

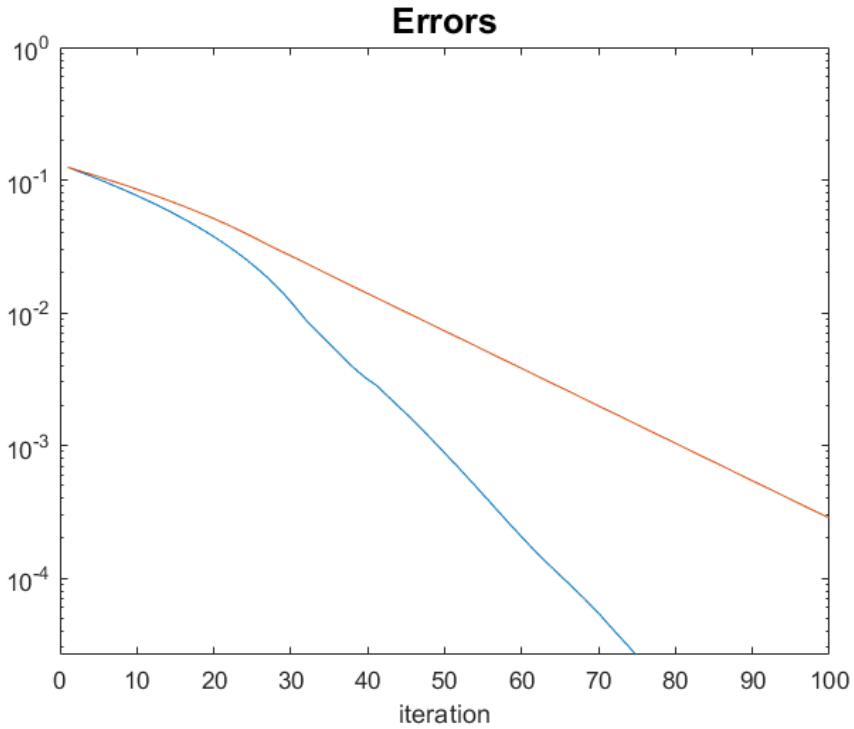
Problem 3.

1. A plot similar to Figure 4.1 after running the program for each method:



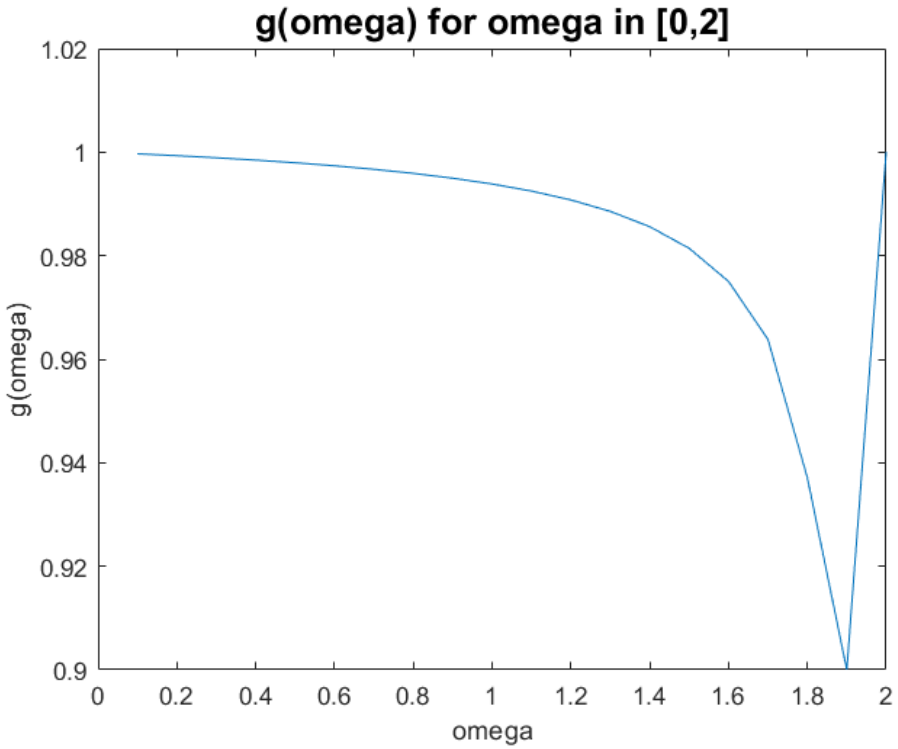
Jacobi – blue, GS – yellow, SOR – violet

2. A plot of the errors for omega optimal and omega = 1.8:

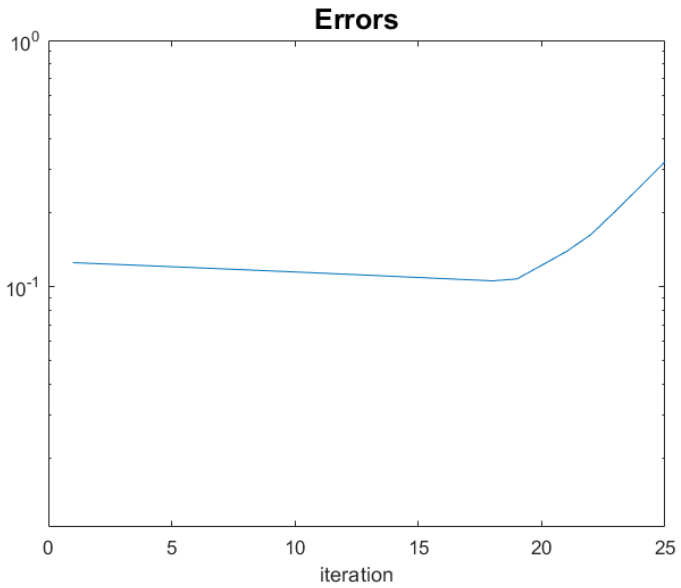


Optimal omega – blue

3. You can find the program in the attachments (program3_3.m). Below is a plot produced by the program:

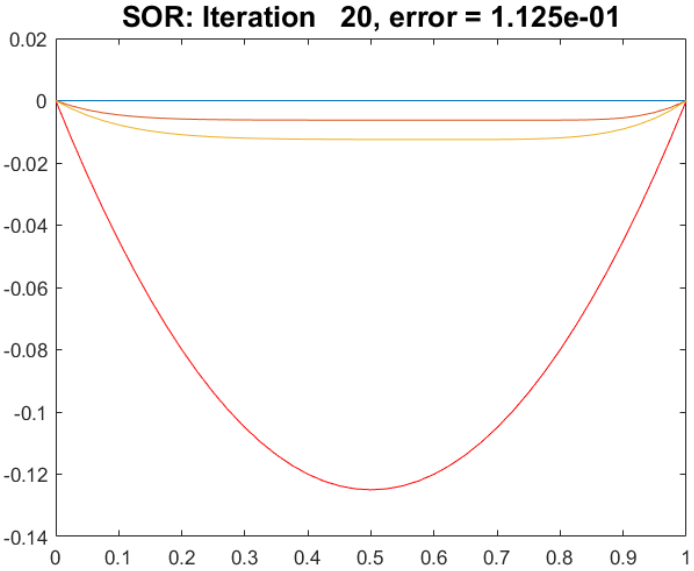


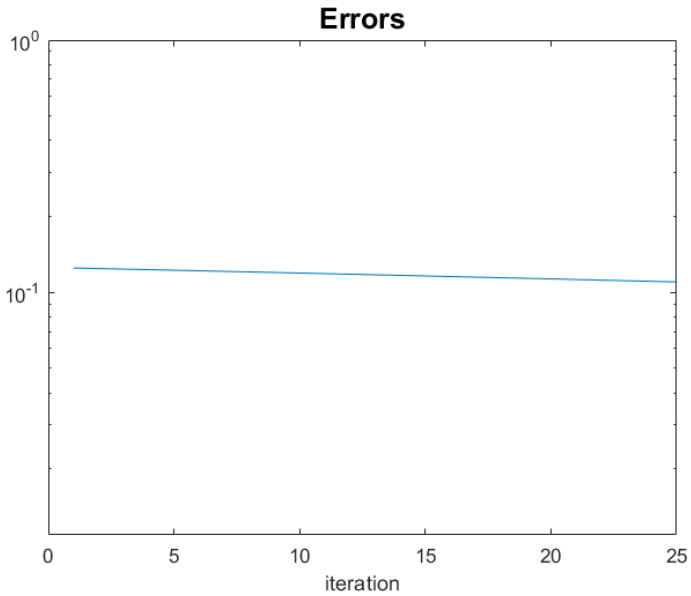
4. Results given by modified code for optimal omega:



As it can be seen from the plots, as number of iterations increase at some point the program blows up and the error starts to grow.

Results given by modified code for omega = 1:





But for suitable omega, the results are fine. It happens because the given update uses $u^{GS(k+1)}_{i-1}$ instead of $u^{(k+1)}_{i-1}$ as in (4.22) (due to the construction of the matrices) and for omega > 1.5 update starts moving in the wrong direction and blows up.

Derivation of the expression (4.24):

Equation (4.22):

$$u_i^{GS} = \frac{1}{2}(u_{i-1}^{[k+1]} + u_{i+1}^{[k]} - h^2 f_i)$$

$$u_i^{[k+1]} = u_i^{[k]} + \omega(u_i^{GS} - u_i^{[k]})$$

Combining above:

$$u_i^{[k+1]} = u_i^{[k]} + \omega(\frac{1}{2}(u_{i-1}^{[k+1]} + u_{i+1}^{[k]} - h^2 f_i) - u_i^{[k]})$$

$$u_i^{[k+1]} = \frac{\omega}{2}(u_{i-1}^{[k+1]} + u_{i+1}^{[k]} - h^2 f_i) + (1-\omega)u_i^{[k]}$$

$$2u_i^{[k+1]} - \omega u_{i-1}^{[k+1]} = \omega u_{i+1}^{[k]} + (1-\omega)2u_i^{[k]} - \omega h^2 f_i$$

$$\frac{2u_i^{[k+1]} - \omega u_{i-1}^{[k+1]}}{h^2} = \frac{\omega u_{i+1}^{[k]} + (1-\omega)2u_i^{[k]}}{h^2} - \omega f_i$$

$$(-D + \omega L)u^{[k+1]} = (\omega U + (1-\omega)D)u^{[k]} - \omega f$$

$$\frac{1}{\omega}(D - \omega L)u^{[k+1]} = \frac{1}{\omega}((1-\omega)D + \omega U)u^{[k]} + f$$

$$Mu^{[k+1]} = Nu^{[k]} + f$$

Hence from above follows the expression (4.24):

$$M = \frac{1}{\omega}(D - \omega L), \quad N = \frac{1}{\omega}((1-\omega)D + \omega U)$$

Problem 4.

1. Backwards GS method:

$$u_i^{[k+1]} = \frac{1}{2}(u_{i-1}^{[k]} + u_{i+1}^{[k+1]} - h^2 f_i)$$

$$2u_i^{[k+1]} - u_{i+1}^{[k+1]} = u_{i-1}^{[k]} - h^2 f_i$$

$$-\frac{2u_i^{[k+1]} - u_{i+1}^{[k+1]}}{h^2} = -\frac{1}{h^2}u_{i-1}^{[k]} + f_i$$

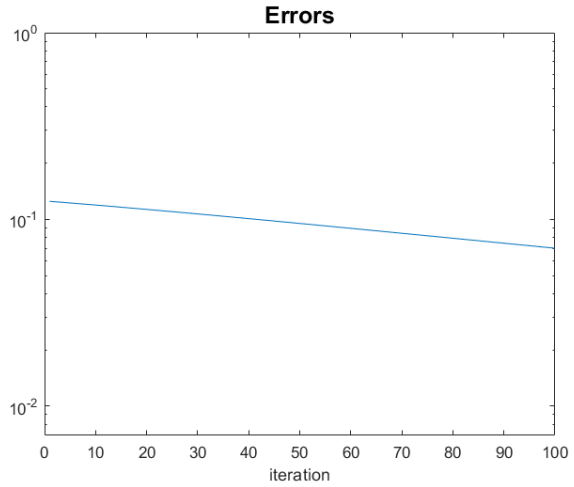
$$(D - U)u^{[k+1]} = Lu^{[k]} + f$$

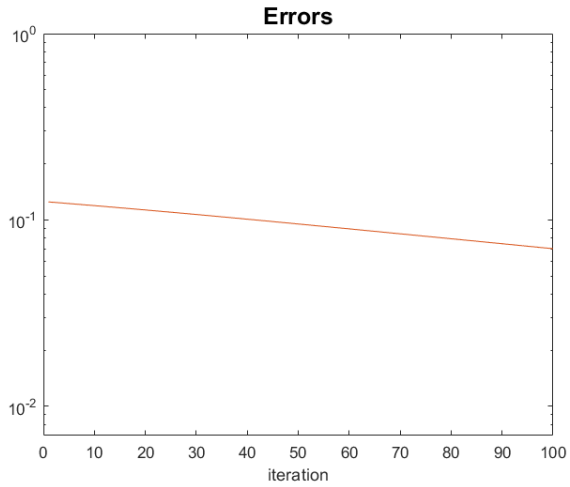
$$Mu^{[k+1]} = Nu^{[k]} + f$$

Hence from above:

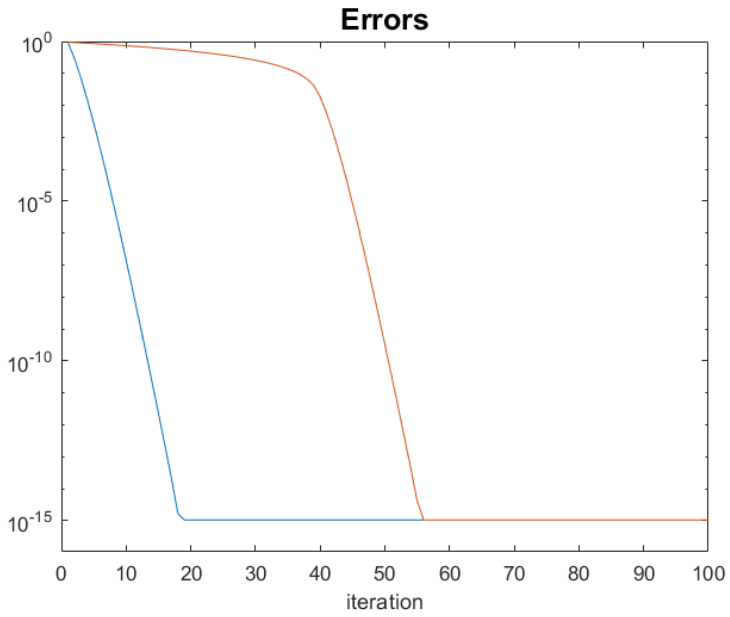
$$M = D - U, \quad N = L$$

2. You can find the program in the attachments (iter_bvp_Asplit4_2.m). Below are the results produced by the program for forwards and backwards implementations. You can see that the rate is the same. I have even plotted them on different graphs because they were overlapping.





3. You can find the program in the attachments (iter_bvp_Asplit4_3.m). Below are the results produced by the program for forwards and backwards implementations:

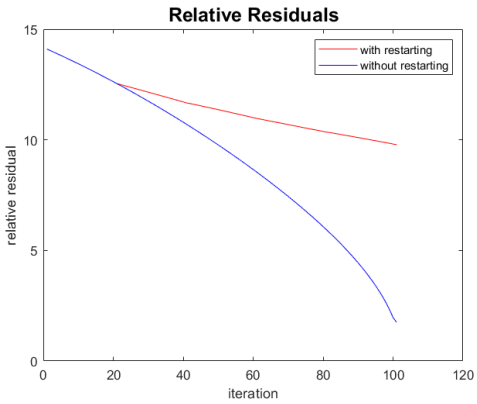


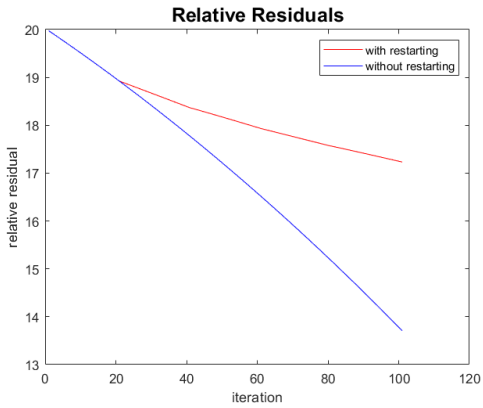
Forward – blue, backward – red

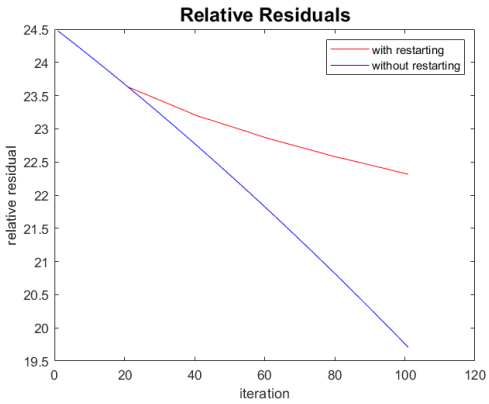
Sweeping in forward direction works much better because backwards one-sided approximation was used. If forwards one-sided approximation was used, the results would have been vice versa. It makes sense because when we reconstruct the A matrix for the new equation forwards one-sided approximation adds a term multiplied by 1/h to the matrix's part affecting forward updates, making update steps larger than the conservative GS steps.

Problem 5.

a) You can find the program in the attachments (program5_1.m). Below are the results produced by the program:



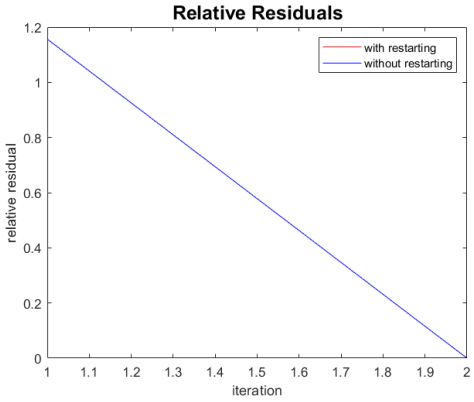


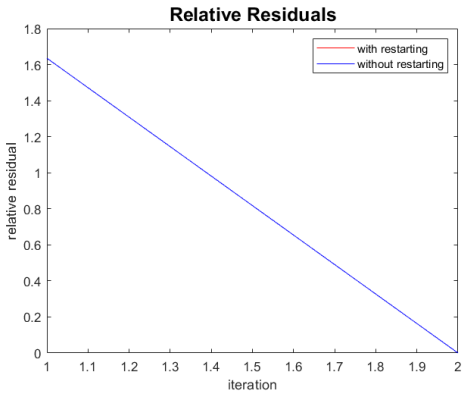


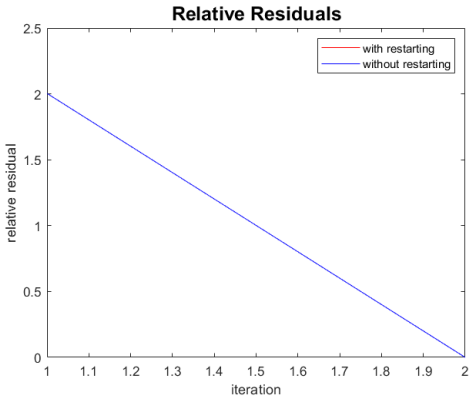
200, 400 and 800 points, respectively

You can notice that the one without restarting is converging way faster than the one with restarting. If you set a specific tolerance, it will converge to it depending on the number of iterations, while the one with restarting fails to converge in most of the cases or the number of iterations should be too big.

b) You can find the program in the attachments (program5_2.m). Below are the results produced by the program:







200, 400 and 800 points, respectively

Incomplete LU preconditioner along with GMRES was used. Both with and without restarting method converged in two iterations to the exact solution.

c) Before preconditioning and after, respectively:

```
>> eigs(A)

ans =

   1.0e+06 *

    1.4400
    1.4400
    1.4399
    1.4398
    1.4398
    1.4396
```

```
>> eigs(inv(L*U)*A)

ans =

   1.0000 + 0.0000i
   1.0000 - 0.0000i
   1.0000 + 0.0000i
   1.0000 - 0.0000i
   1.0000 + 0.0000i
   1.0000 + 0.0000i
```

Since eigenvalues before preconditioning are around 1e+06*1.4400 and after they are around 1 I guess there is indeed a clustering.