MA398 Assignment 3

1.

a) $R = -(L+D)^{-1}U$

$\det(R-\lambda I) = \det((-L+D)^{-1}U - \lambda I) = 0$

$\Rightarrow$ ~~det~~
Define $A_\lambda := \lambda L + \lambda D + U$

$\Rightarrow \det(A_\lambda) = \det(\lambda L + \lambda D + U) = 0$

Since $A = L+D+U$ satisfies criteria, then we can confirm for $|\lambda| \geq 1$, A_λ also satisfies the criteria.

$\Rightarrow$ non-singular

$\Rightarrow \det(A_\lambda) = 0$ can never be true

$\Rightarrow |\lambda| < 1$ must always be the case and so we have convergence

b) $Ax = b$

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \quad n \times n$$

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

$$A = L + D + U$$

$$L = \begin{pmatrix} 0 & & & 0 \\ a_{21} & \ddots & & \\ \vdots & & \ddots & \\ a_{n1} & \cdots & a_{nn-1} & 0 \end{pmatrix}$$

$$D = (a_{ii}) = \begin{pmatrix} a_{11} & & 0 \\ & \ddots & \\ 0 & & a_{nn} \end{pmatrix}$$

$$U = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & \ddots & & \vdots \\ 0 & & \ddots & a_{n-1n} \\ & & & 0 \end{pmatrix}$$

PTO

$\Rightarrow (L+D+U)x = b$

$\Rightarrow \omega(L+D+U)x = \omega b$

$\Rightarrow (\omega(L+U) + \omega D)x = \omega b$

$\Rightarrow \quad \omega D = (\omega-1)D + D$

$\Rightarrow (\omega(L+U) + (\omega-1)D + D)x = \omega b$

$\Rightarrow (\omega L + D + (\omega-1)D + \omega U)x = \omega b$

$\Rightarrow (D+\omega L)x + (\omega U + (\omega-1)D)x = \omega b$

$\Rightarrow (D+\omega L)x = \omega b - (\omega U + (\omega-1)D)x$

$\therefore \quad x^{(k-1)} = (D+\omega L)^{-1}(\omega b - (\omega U + (\omega-1)D)x^{(k)})$

$(D+\omega L)$ is triangular

$\therefore$ forward substitution

$$\Rightarrow x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\Big(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i} a_{ij}x_j^{(k)}\Big)$$

for $i = 1, \dots, n$

c) One reason is that Gauss-Seidel converges faster (i.e. in fewer iterations). This means that G-S will be closer to the true result of x_k, within a finite number of iterations (call this itr). If G-S and Jacobi both converge in m,n ~~great~~ iterations s.t. $m,n > itr$, G-S will be

PTO

closer to the correct value of x_k due to the faster conversions.

The reason for the faster conversion of G-S when compared to Jacobi is that in the algorithm for G-S, values are updated during ~~one iteration~~ on the current iteration, whereas with Jacobi, values are only ever updated on the next iterative step, leading to Jacobi taking a greater number of iterations to reach the result.

d) See:

sor.m

I modified Jacobi method given in the question to make this.

I also made both the ~~SOR~~ Jacobi method and the SOR method return the results x_k at each step k. (xarr in method)

5

My code for e) is very inefficient so it will take some time to run

e) $A = \begin{pmatrix} 4.1 & 2 & & 0 \\ 2 & \ddots & \ddots & \\ & \ddots & \ddots & 2 \\ 0 & & 2 & 4.1 \end{pmatrix}$ $n \times n$

$b = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$ n

i) $\|A\|_\infty$ ~~for~~

$n = 1$

$A = (4.1)$

$\Rightarrow \|A\|_\infty = 4.1$

$n = 2$

$A = \begin{pmatrix} 4.1 & 2 \\ 2 & 4.1 \end{pmatrix}$

$\|A\|_\infty = 6.1$

$n = 3$

$\begin{pmatrix} 4.1 & 2 & 0 \\ 2 & 4.1 & 2 \\ 0 & 2 & 4.1 \end{pmatrix}$

$\|A\|_\infty = 8.1$

$n \geq 3$

$\|A\|_\infty = 8.1$

$\|b\|_\infty$ is always 1, ~~regardless~~ for $n \geq 1$

$\therefore$ ~~for~~ uniformly bounded in n

for (ii) – (v) please also see MA398_Assignment 3_Exercise1e.m

ii) see matlab and graph

$\|e_0\|_\infty$ converges and so is uniformly bounded in n

iii) see matlab code and graphs

Regardless of n, G-S takes 265 iterations

Jacobi is 838 for 128 and 840 for the rest.

$\|R\|_\infty$

iv) $e_k = R^k e_0$

$e_k = x - x_k$

$e_0 = x - x_0$

in this case $x_0 = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

$\Rightarrow e_0 = x$

$\Rightarrow x - x_k = R^k x$

$\Rightarrow R^k = x^{-1}(x - x_k)$

$\Rightarrow R = \left(x^{-1}(x - x_k)\right)^{\frac{1}{k}} = \left(x^{-1}x - x^{-1}x_k\right)^{\frac{1}{k}}$

~~$\|R\|_{\infty}$~~

$\Rightarrow R = (I - x^{-1}x_k)^{\frac{1}{k}} \Leftarrow$ should tend to 0

$\|R\|_\infty = \max \|(I - x^{-1}x_k)^{\frac{1}{k}}\|_\infty$

should eventually be 0, like the gradient is so $\|R\|_\infty$ is as expected.

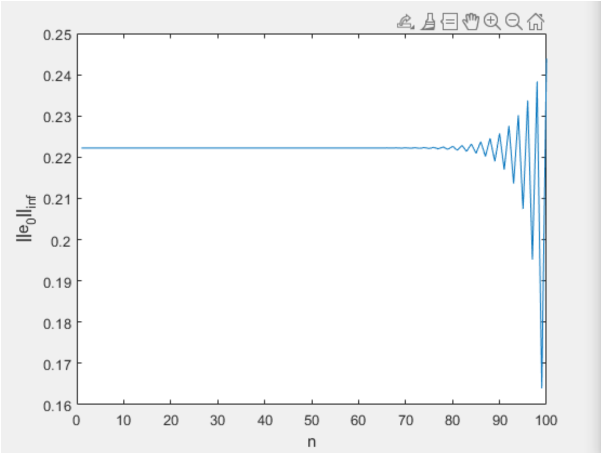
iv) ~~I couldn't~~
I couldn't get it to work for $\omega = 0.2, 0.4$ within 10000 iterations.
I did from 0.6 – 2 instead

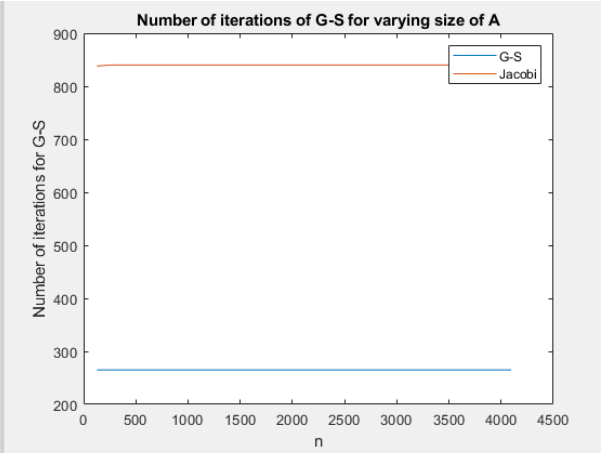
Smaller ω values ~ 0.6 worked better and got worse as it got higher

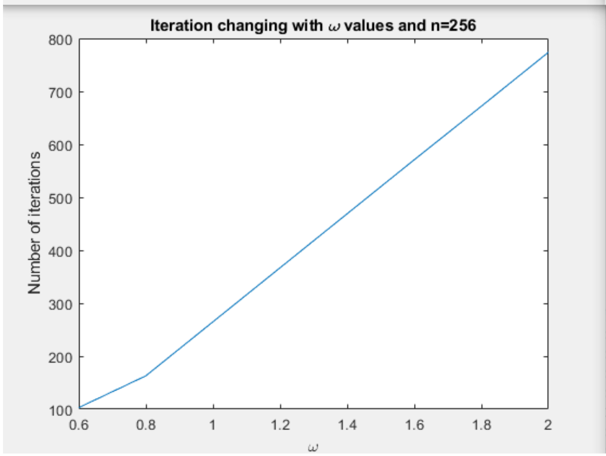
Jacobi took 840 iterations, where SOR with $\omega = 0.6$ only took just over 100

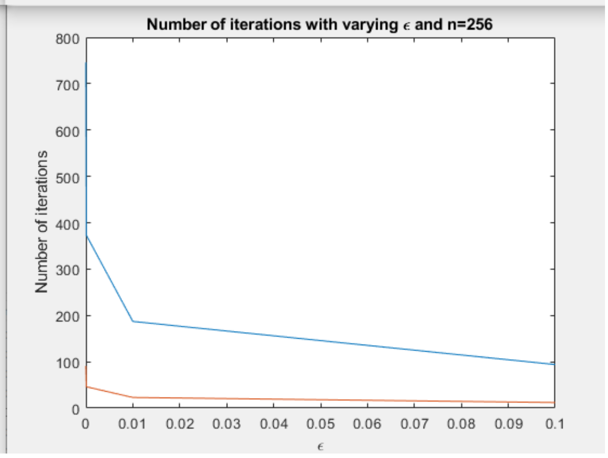
v) As visible in the plot, the lower values of ε required more iterations. The higher, more generous ε values took fewer iterations. This makes sense since it takes longer to get smaller residual norms.
more iterations

(Also N.B. I called ε tol in my code)


$||e_0||_{Inf}$
n


Number of iterations of G-S for varying size of A
G-S
Jacobi
Number of iterations for G-S
n


Iteration changing with ω values and n=256
Number of iterations
ω


Number of iterations with varying ϵ and n=256
Number of iterations
ϵ

2. See:

MA398_Assignment3_Exercise2.m

SD.m

CG.m

SD.m is steepest descent

~~CG~~.m is conjugate gradient

The algorithms ~~in~~ the notes suggest that within the loop, it is checked if $\|r^{(k-1)}\|_2 \leq \varepsilon_r$ then return $x^{(k-1)}$. Matlab doesn't really work like this, so my while loop runs for the opposite condition. (i.e. $\|r^{(k-1)}\| > \varepsilon_r$)

Similarly done for S~~D~~ and CG

The residual norms vs. k for steepest descent are greater than those for gradient descent.

This would imply that gradient descent is "better" and this can be analysed. It does seem to converge to the correct result for $\underline{x}$ in this case. This, of course, also depends on the given tolerance. Matlab's inbuilt cgs() method uses a tolerance of 1×10^{-6}, which I have decided to use for both my sd and ~~gd~~ cg methods. (Declared outside of the functions)

With this tolerance, the cg ~~ds~~ ~~dg~~ algorithm takes 59 iterations to reach a satisfying result for $\underline{x}$, whereas sd does not reach it within 100 iterations.

I plotted 2 sets of graphs. One for the residual norms against k ~~and~~ of sd and ~~gd~~ cg

The second set is the plot of $\|e^{(k)}\|_A / \|e^{(0)}\|_A$

In the notes:

Theorem 25.2 states

$$\|e^{(k)}\|_A \leq \left(\sqrt{1-\frac{1}{K_2(A)}}\right)^k \|e^{(0)}\|_A$$

$$\Rightarrow \|e^{(k)}\|_A / \|e^{(0)}\|_A \leq \left(\sqrt{1-\frac{1}{K_2(A)}}\right) \qquad \text{sd}$$

and

Theorem 25.4 ~~states~~ similarly ~~states~~ results in

$$\frac{\|e^{(k)}\|_A}{\|e^{(0)}\|_A} \leq 2\left(\left(\frac{\sqrt{K_2(A)}+1}{\sqrt{K_2A}-1}\right)^k + \left(\frac{\sqrt{K_2(A)}+1}{\sqrt{K_2(A)}-1}\right)^{-k}\right)^{-1} \qquad \text{cg}$$

So I have decided to plot each side of the inequalities against eachother.

They should both be bounded between 0 and 1.

$$\|e^{(k)}\| = \|A^{-1}r^{(k)}\|$$

$$\Rightarrow \|e^{(k)}\|_A = \|A^{-1}r^{(k)}\|_A$$

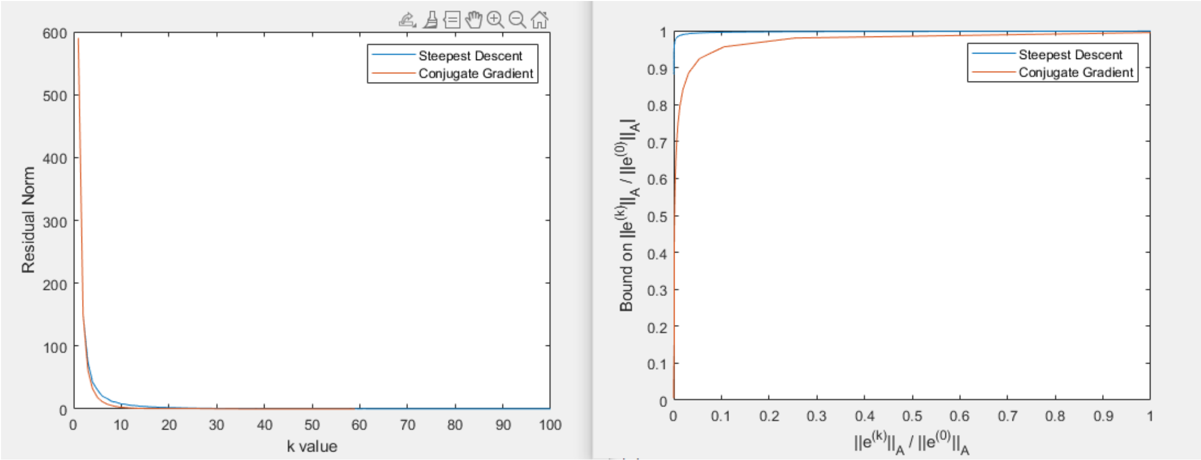
$$= \sqrt{\langle e^{(k)}, \cancel{A}\, e^{(k)}\rangle_A} = \sqrt{\langle \cancel{e} A^{-1}r^{(k)}, A^{-1}r^{(k)}\rangle_A}$$

$$\cancel{A} = \sqrt{(r^{(k)})^T (r^{(k)})}$$

So $\frac{\|e^{(k)}\|_A}{\|e^{(0)}\|_A}$ can be calculated like this

RTG

From analysing the value of $\frac{\|e^{(k)}\|_A}{\|e^{(0)}\|_A}$ ~~agains~~ and it's bound, we can see ~~that~~ the inequality always holds. This can be visualised by a plot.



Steepest Descent
Conjugate Gradient
Residual Norm
k value
Bound on $||e^{(k)}||_A / ||e^{(0)}||_A$
$||e^{(k)}||_A / ||e^{(0)}||_A$

3. $$A = \begin{pmatrix} d & a & & & 0 \\ a & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & a \\ 0 & & & a & d \end{pmatrix} \quad m \times m$$

Let's generalise this so (b) is quicker.

$$M_m = \begin{pmatrix} \alpha & \beta & & 0 \\ \gamma & \ddots & \ddots & \\ & \ddots & \ddots & \beta \\ 0 & & \gamma & \alpha \end{pmatrix} \quad m \times m$$

Let's define 2 more matrices, B_m, C_m s.t.:

B_m and C_m are both diagonal matrices with elements b_{jj} and c_{jj} respectively.

$b_{jj} = \gamma^{(j-2)/2} / \beta^{(j-1)/2}$

$c_{jj} = \beta^{(j-2)/2} / \gamma^{(j-1)/2}$

for $j = 1, \dots, m$

define $z := \dfrac{\alpha - \lambda}{\beta^{1/2} \gamma^{1/2}}$

$\Rightarrow |C_m (M_m - \lambda I) B_m| = 0$

Call this $P_m(z)$

$$\Rightarrow P_m(z) = \left| \begin{pmatrix} z & 1 & & 0 \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & & 1 & z \end{pmatrix} \right| = 0$$

PTO

Chebyshev recurrence formula

~~$P_m(z) = z P_m \quad z P_{m-1}$~~

$$P_m(z) = z P_{m-1}(z) - P_{m-2}(z)$$

Initial values

$$P_1(z) = z$$
$$P_2(z) = z^2 - 1$$

$$z_k = 2\cos\left(\frac{k\pi}{m+1}\right) \qquad k = 1, \dots, m$$

So now we can obtain the eigenvalues of M_m from our defn. of z.

$$\Rightarrow \lambda_k = d - 2\beta^{\frac{1}{2}}\gamma^{\frac{1}{2}}\cos\left(\frac{k\pi}{m+1}\right) \qquad k = 1, \dots, m$$

Due to symmetry, ~~we~~ we can say

$$\lambda_k = d \pm 2\beta^{\frac{1}{2}}\gamma^{\frac{1}{2}}\cos\left(\frac{k\pi}{m+1}\right) \qquad k = 1, \dots m$$

So $\lambda_k = d + 2\sqrt{\beta\gamma}\cos\left(\frac{k\pi}{m+1}\right)$

So for ~~Q~~

a) eigenvalues are: $\beta, \gamma = a \Rightarrow \sqrt{\beta\gamma} = a$
$d = d$

$$\Rightarrow \lambda_k = d + 2a\cos\left(\frac{k\pi}{m+1}\right)$$

b) $\beta = a,\ \gamma = b,\ d = d$

$$\Rightarrow \lambda_k = d + 2\sqrt{ab}\cos\left(\frac{k\pi}{m+1}\right)$$

PTO

Eigenvectors

by properties of eigen values and eigenvectors, we know:

$(A - \lambda_k I)\underline{x_k} = 0$

where $\underline{x_k}$ is k^{th} eigen vector

The j^{th} row is given by

$a x_{kj-1} + (d - \lambda_k) x_{kj} + a x_{kj+1} = 0 \quad j = 1, \ldots, m$

we can shift the indices to obtain

$a x_{kj} + (d - \lambda_k) x_{kj+1} + a x_{kj+2} = 0 \quad j = 1, \ldots, m-1$

Again a difference equation.

I think it would be more interesting to use another method to solve this.

I propose using the unilateral z transform.

Doing this results in:

$$a X_k(z) + (d - \lambda_k)(z X_k(z) - z x_{k0}) + a(z^2 X_k(z) - z^2 x_{k0} - z x_{k1}) = 0$$

Setting $x_{k1} = 0 \Rightarrow \underline{x_k} = 0$

PTO

Define arbitrary constant n_k s.t. $x_{k1} = n_k \neq 0$

Eigenvectors can be scaled so this is fine.

We now have:

(rearranging prev equation)

$$X_k(z)(a + z(d-\lambda_k) + z^2 a) = (d-\lambda_k) z x_{k0} + a z^2 x_{k0} + z x_{k1}$$

$$\Rightarrow X_k(z)\,(a + (d-\lambda_k)z + a z^2) = z n_k$$

$$\Rightarrow X_k(z) = \frac{z n_k}{(a + (d-\lambda_k)z + a z^2)}$$

Finding inverse transform of X_k gives us eigenvectors

roots of contents of denominator

$$z_{\pm} = \frac{-(d-\lambda_k) \pm \sqrt{(d-\lambda_k)^2 - 4a^2}}{2a}$$

Since $\lambda_k = d - 2a\cos\left(\frac{k\pi}{m+1}\right)$

$$\Rightarrow d - \lambda_k = 2a\cos\left(\frac{k\pi}{m+1}\right)$$

$\Rightarrow$ sub into $z_{\pm}$ gives

$$z_{\pm} = \cos\left(\frac{k\pi}{m+1}\right) \pm i\sqrt{1-\cos^2\left(\frac{k\pi}{m+1}\right)}$$

$$\Rightarrow z_{\pm} = \cos\left(\frac{k\pi}{m+1}\right) \pm i\sin\left(\frac{k\pi}{m+1}\right)$$

PTO

15

$$X_k(z) = \frac{z n_k}{a + (d-\lambda_k)z + a z^2} = z n_k \left(\frac{A}{z-z_+} - \frac{B}{z-z_-}\right.$$

$$A = \frac{1}{z_+ - z_-} = \frac{1}{2i \sin\left(\frac{k\pi}{m+1}\right)}$$

$$B = -\frac{1}{z_+ - z_-} = -\frac{1}{2i \sin\left(\frac{k\pi}{m+1}\right)}$$

$$\Rightarrow X_k(z) = \frac{z n_k}{2i \sin\left(\frac{k\pi}{m+1}\right)} \left(\frac{1}{z-z_+} - \frac{1}{z-z_-}\right)$$

$$= \frac{n_k}{2i \sin\left(\frac{k\pi}{m+1}\right)} \left(\frac{1}{1 - z_+/z} - \frac{1}{1 - z_-/z}\right)$$

$$= \frac{n_k}{2i \sin\left(\frac{k\pi}{m+1}\right)} \left((1 + z_+ z^{-1} + z_+^2 z^{-2} + \dots + z_+^m z^{-m} + \dots)\right.$$

$$\left. - (1 + z_- z^{-1} + z_-^2 z^{-2} + \dots + z_-^m z^{-m} + \dots)\right)$$

$$= \frac{n_k}{\sin\left(\frac{k\pi}{m+1}\right)} \left(\sin\left(\frac{k\pi}{m+1}\right) z^{-1} + \sin\left(\frac{2k\pi}{m+1}\right) z^{-2} + \dots + \sin\left(\frac{mk\pi}{m+1}\right) z^{-m} + \dots\right)$$

choose n_k to be $\sin\left(\frac{k\pi}{m+1}\right)$

$$\Rightarrow x_{kj} = \sin\left(\frac{jk\pi}{m+1}\right) \qquad j = 1, \dots, m \qquad (a)$$

$(x_{kj} \equiv x_j)$

PTO

~~x~~ Similarly to eigenvalues, eigenvectors for $\begin{pmatrix} d & a & \cdots & 0 \\ b & \ddots & \ddots & a \\ 0 & \cdots & b & d \end{pmatrix}$ are:

b) $x_j = \left(\frac{b}{a}\right)^{k/2} \sin\left(\frac{jk\pi}{m+1}\right)$