Computational homework 2
Nonlinear equations

Both exercises 1 and 2 (marked *) to be submitted.
At least one of these will be marked.
Deadline: 23:59hrs Sunday 27th November

It is recommended that you use some of the Matlab `.m` files that are provided on Moodle. Please submit your solutions via Crowdmark. You should submit a single pdf file, created using the Matlab `publish` command, formatted as per the template provided on the Moodle page. Your file should show the code you used, its output (including figures) and your comments. Please do not include the code of any of the mfiles I gave you. Note that before submitting to Crowdmark you should "flatten" your pdf file by opening it in a pdf viewer (like Adobe Reader) and printing it to a new pdf using the Print dialogue (see instructions on Moodle). Otherwise Crowdmark may not display it properly.

EXERCISE 1(*) Consider the linear system $A_\epsilon \mathbf{x} = \mathbf{b}_\epsilon$, where, for $\epsilon \in [0,1]$ and $n \geq 5$, the pentadiagonal $n \times n$ matrix A_ϵ and the n-vector $\mathbf{b}_\epsilon$ are defined by

$$A_\epsilon = \begin{pmatrix} 1 & \epsilon & \epsilon^2 & & 0 \\ \epsilon & 1 & \epsilon & \ddots & \\ \epsilon^2 & \epsilon & \ddots & \ddots & \epsilon^2 \\ & \ddots & \ddots & 1 & \epsilon \\ 0 & & \epsilon^2 & \epsilon & 1 \end{pmatrix}, \quad \mathbf{b}_\epsilon = A_\epsilon \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}.$$

For a given size `n` and a given value `epsi` of ϵ, you can construct the matrix A_ϵ and the vector $\mathbf{b}_\epsilon$ using the command `[A,b] = matrix(n,epsi)` (code available on the Moodle page).

(a) We know from lectures that if A_ϵ is strictly diagonal dominant then the Jacobi and Gauss-Seidel methods both converge. For which ϵ does this hold? (Assume that $n \geq 5$.)

(b) Compute the solution of the system in the case `n=5` with $\epsilon = 0.3$, using both the Jacobi and Gauss-Seidel methods. To do this, use the command `itermeth` (code available on the Moodle page) and choose first `'J'` and then `'G'` as the method. Use the tolerance `tol` $= 10^{-10}$, maximum number of iterations 10^3, and initial guess $\mathbf{x}^{(0)} = (0,0,0,0,0)^T$. Report the number of iterations necessary for convergence in both cases.

(c) Still using `n=5`, use the built-in Matlab command `eig` to compute the spectral radius of the iteration matrices of the Jacobi method $B_J(\epsilon)$ and the Gauss-Seidel method $B_{GS}(\epsilon)$ for $\epsilon = 0, 0.01, 0.02, \ldots, 1$. Recall that

$$B_J(\epsilon) = -D_\epsilon^{-1}(L_\epsilon + U_\epsilon), \qquad B_{GS}(\epsilon) = -(D_\epsilon + L_\epsilon)^{-1} U_\epsilon,$$

where D_ϵ is the diagonal part of A_ϵ, L_ϵ is the lower triangular part of A_ϵ (excluding the diagonal) and U_ϵ is the upper triangular part of A_ϵ (excluding the diagonal). Plot the spectral radius against the value of ϵ for both methods on the same axes (the `hold` command might be useful).

For which (approximate) range of ϵ do you expect the two methods to converge? How does your answer compare to the answer you obtained in part (a)?

When both methods converge, which do you expect to be faster?

Which method would you recommend using if `n=5` and $\epsilon = 0.5$? Run both methods for this case and check that the outcomes are consistent with your recommendation.

EXERCISE 2(*) Consider the boundary value problem: find $y : [0,1] \mapsto \mathbb{R}$ such that

$$-\frac{d^2y}{dx^2} = \sin(\pi\, x) \quad \text{in } I := [0,1], \qquad y(0) = y(1) = 0, \tag{1}$$

the exact solution of which is $y(x) = \pi^{-2}\sin(\pi\, x)$.

In lectures we are soon going to discuss how the solution y can be approximated using a finite difference method on a set of nodes $\{x_i\}_{i=0}^N$ defined by $x_n = nh$, $n = 0, \ldots, N$, where $h = 1/N$ is the step size. We seek a set of $N+1$ numbers $\{u_i\}_{i=0}^N$ approximating the values of the solution y at the nodes $\{x_i\}_{i=0}^N$. To impose the boundary conditions we set $u_0 = u_N = 0$. To approximate the ODE we replace the second derivative operator by the finite difference operator

$$D^2 u_n := \frac{u_{n+1} - 2u_n + u_{n-1}}{h^2}.$$

We find that the numerical approximation $\mathbf{u} = (u_1, \ldots, u_{N-1})^T$ to the interior values satisfies the linear equation $A\mathbf{u} = \mathbf{b}$, for a certain matrix A and vector $\mathbf{b}$. For a given $N \in \mathbb{N}$ these may be constructed in Matlab using the script

```
>> h=1/N;
>> A= (2/h^2)*diag(ones(N-1,1)) - (1/h^2)*diag(ones(N-2,1),1)...
 - (1/h^2)*diag(ones(N-2,1),-1);
>> b = transpose(sin(pi*h*(1:N-1)));
```

(If you are interested in knowing more about the background theory, then the section of the notes where this material is discussed is §5.9. But you don't need to look at §5.9 to answer this question - everything you need is given to you here.)

(a) Use the built-in Matlab command `eig` to compute the spectral radii of the iteration matrices B_{J} and B_{GS} of the Jacobi and Gauss-Seidel methods applied to the system above, for each $N \in \{5, 10, 20, 40, 80\}$. Report the results and store them in two vectors `rhoBJ` and `rhoBGS`.

Hint: the matrices D, L, U needed to construct the iteration matrices for the two methods can be constructed using the commands

```
>> D = diag(diag(A));
>> L = tril(A)-D;
>> U = triu(A)-D;
```

Do you predict that the two iterative methods should be convergent for these values of N? If so, which should converge faster?

What seems to be happening to $\rho(B_{\mathrm{J}})$ and $\rho(B_{\mathrm{GS}})$ as the system size N increases? As a result, what do you expect to happen to the performance of the Jacobi and Gauss-Seidel methods?

To investigate this behaviour further, present the plots produced by the commands

```
>> loglog(Nvec,1-rhoBJ)
>> hold
>> loglog(Nvec,1-rhoBG)
```

where `Nvec=[5,10,20,40,80]` is the vector of matrix sizes.

Use your results to propose an approximate relationship between the spectral radii and the size of the matrix N that you conjecture to hold in the limit as $N \to \infty$. Give estimated values for any constants that appear in your relationships. *(I.e. if you conjecture that* $\rho(B) \approx 1 - C\alpha^N$ *or* $\rho(B) \approx 1 - CN^\alpha$ *then give numerical estimates for* C *and* α*.)*

Assuming that the error in both methods is approximately proportional to $\rho(B)^k$, and using the relationship you derive above, and can you predict (roughly) how the number of iterations k_{tol} required to achieve a fixed solution accuracy *tol* will grow as $N \to \infty$, for both methods? Comment on the implications for the practicality (or otherwise) of using the Jacobi and Gauss-Seidel methods (compared to e.g. direct methods) for this problem when N is large.

(b) Solve the system for $N = 5, 10, 20, 40, 80$ using the Gauss-Seidel method (use the command `itermeth` as in Exercise 1, with a tolerance 10^{-10} and an increased maximum iteration number 10^5), starting from an appropriate initial guess. Plot the resulting solutions *on the same axes* (don't forget u_0 and u_N), along with a plot of the exact solution $y(x)$ (formula stated at the start of this exercise), evaluated on the finest mesh ($N = 80$). Do the solutions appear to converge as N increases?

For each N compute the error

$$e(N) = \max_{n=0,\ldots,N} |u_n - y(x_n)|, \tag{2}$$

and store your results in a vector `error_vect`.

In lectures we will show that $e(N) \leq CN^{-2}$ as $N \to \infty$ (equivalently, $e(N) \leq Ch^2$ as $h = 1/N \to 0$), where the constant C is proportional to $\max_{x\in[0,1]} |y''''(x)|$. (This comes from the error in replacing d^2/dx^2 by D^2 — recall theoretical exercise sheet 1.) Do your numerical results support or contradict this theoretical estimate?

Hint: you may find it useful to plot your results using the command

```
>> loglog(hvect, error_vect)
```

where `hvect=1./Nvec` *is a vector of step sizes. Recall that if* $E(h) = Ch^p$ *then a* `loglog` *plot of* E *against* h *will produce a straight line with slope* p *and offset* $\log C$*. Once you have determined appropriate values of* p *and* C *you can check them by adding an extra plot to your figure using* `hold` *and the command*

```
>> loglog(hvect, C*hvect.^p)
```

(c) Now consider the equation (1), but with the right hand side $\sin(\pi x)$ replaced by 1. The exact solution in this case is $y(x) = (1/2)x(1-x)$. Once again, solve the system for $N = 5, 10, 20, 40, 80$ (you will need to modify the vector `b` in your code), plot the numerical solutions and the exact solution on the same axes, and compute the corresponding errors (2). You might initially be surprised by what you see when you plot

```
>> loglog(hvect, error_vect)
```

Can you explain what is going on here?