

CSUMS Seminar One – Wednesday, September 17, 2008

Computer Tasks

Instructions: With your group members, please complete the following tasks. All work is expected to be done in `Matlab`. Write a summary explaining what happened in each task. Your summary and `Matlab` code (`newton.m`, `f.m`, `fp.m`, `jacobi.m`, `jacobiscript.m`, `gaussseidel.m`, and `gaussseidelscript.m`) should be emailed to Professors Stolarska and Van Fleet (`mastolarska`, `pjvanfleet`) on or before noon on Tuesday, September 23, 2008.

1. **Newton's Method.** During the seminar, we created three `Matlab` *m*-files: `f.m` (defines the function $f(x)$), `fp.m` (defines $f'(x)$), and `newton.m` (performs n iterations of Newton's method). In this task, you are to use the Help documentation that comes with `Matlab` to learn about the `if` and `while` commands and then use these commands to add a stopping criteria to `newton.m`. The number of arguments for `newton.m` should increase by one - you will add a third argument `eps` that is a stopping criteria. The function call now is `newton(x0,m,eps)` and the routine should expect to perform n iterations of Newton's method, but stop and return x should $|x| < \epsilon$. Using $m = 100$, test your function for each of the following cases:

(a) $f_1(x) = x^3 - 4x + 1$, $x_0 = 3$, $\epsilon = 10^{-6}$.

(b) $f_2(x) = x^3 - 20 * x$, $x_0 = 2$, $\epsilon = 10^{-6}$.

(c) $f_3(x) = \tan^{-1}(x)$, $x = 2$, $\epsilon = 10^{-2}$.

Record what happens in each case in your report. Turn in your `Matlab` code for (c).

2. **Newton's Method.** Modify your `newton.m` file once again so that m is treated as a maximum number of iterations. That is, if the routine performs n iterations and the root is not reached to the desired tolerance, have the routine print a message stating that the maximum number of iterations has been reached and the routine has failed. Test your new function on the three test cases from Problem 1. Record what happens in each case in your report. Turn in your `Matlab` code for (c).
3. **Jacobi Iterative Method.** Modify the *m*-file `jacobi.m` so that it checks the diagonals of the input matrix A to see if any of the $a_{ii} = 0$ and if so, exit the function with an error message. (*Hint:* The `MatLab` commands `diag` and `prod` will be useful here - much cheaper to use these than to actually compute D^{-1} .) Test your function (with initial guess the zero vector and $m = 20$) on the following systems:

- (a) $Ax = b$ where

$$A = \begin{bmatrix} -2 & 1 & 1/2 \\ 1 & -2 & -1/2 \\ 0 & 1 & 2 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 4 \\ -4 \\ 0 \end{bmatrix}$$

(b) $A\mathbf{x} = \mathbf{b}$ where

$$A = \begin{bmatrix} 4 & 1 & -1 & 1 \\ 1 & 4 & -1 & -1 \\ -1 & -1 & 5 & 1 \\ 1 & -1 & 1 & 3 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} -2 \\ -1 \\ 0 \\ 1 \end{bmatrix}$$

(c) $A\mathbf{x} = \mathbf{b}$ where

$$A = \begin{bmatrix} 1 & 3 & 2 \\ 1 & 0 & 1 \\ 0 & 1 & 4 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 6 \\ 2 \\ 5 \end{bmatrix}$$

(d) Your function should have determined that one of the diagonal elements in part (c) was zero. However, A^{-1} exists! Is there any way we can use the Jacobi iterative method to solve the system in part (c)? If so, describe how you would proceed.

Put your work in script file `jacobiscript.m`.

4. **Jacobi Iterative Method.** Modify the file `jacobi.m` so that it takes a fifth argument `eps`. This argument is a tolerance that determines when the routine has reached the desired accuracy. Also treat the input argument `n` as a maximum number of iterations and if the desired accuracy has not been obtained after `m` iterations, return a message that indicates failure to converge. Test your function on the systems in Problem 3(a),(b), and (d) in the script file `jacobiscript.m`.
5. **Gauss–Seidel Iterative Method.** Write a function called `gaussseidel.m`. This function has five input arguments: `gs(A,b,m,x0,eps)`. The routine estimates the solution to the system $A\mathbf{x} = \mathbf{b}$ with an initial guess of x_0 . The routine estimates the solution to accuracy ϵ and terminates with an error message if the routine fails to converge after `m` iterations. Test your routine on the systems in Problem 3(a),(b),(d) in the script file `gsscript.m`.
6. **Large Sparse System.** Consider the system $A\mathbf{x} = \mathbf{b}$ where A is the 100×100 tridiagonal matrix with 1 on the main diagonal and $-\frac{1}{2}$ on the off diagonals and $\mathbf{b} = [\frac{1}{2}, 0, \dots, 0, \frac{1}{2}]^T$.
The solution of this system is $\mathbf{x} = [1, 1, \dots, 1, 1]^T$. Set $m = 1000$ and $\epsilon = 10^{-4}$ and use both `jacobi.m` and `gaussseidel.m` to estimate the solution to the system. Which routine took fewer iterations? (*Hint:* The difficult part of this problem is defining A and \mathbf{b} ! Look at the command `diag` in the Matlab help to get some ideas.) Put your work in the script file `tridiagonalscript.m`.