Math 464, Computer Projects

Regarding computer projects:

- You may use your own machine or your MSCC account, but all computations should be done in double precision (about 16 digits of accuracy). All projects can be done by writing Matlab script and function files (or Splus functions) and you are urged to do so. However you may not use a Matlab routine such as fzero in place of your own routine unless I explicitly give you permission. Van Loan's Chapter 1.5 has useful information on designing and documenting functions. You may write in C,C++, Fortran or other higher order compiled language if you wish. However, use of these languages will necessitate much more attention to details of formatting input and output and possibly with interfacing with separate plotting routines. This will substantially increase the work needed to achieve the desired results.
- Interpret your results:
  - Observe what happened.
  - Explain what happened in light of the theory we have studied.
- Turn in your write-up of your interpretations, your program listings, and your program output (in that order), attached together, at the beginning of class on the due date.
- Please start early. No late sets will be accepted. Leave time to interpret your results. NOTICE the observations and explanations are critical to beginning to appreciate and understand numerical analysis.

---

1 Tridiagonal Systems – due Friday, October 22

Write a program to solve a tridiagonal system $Tx = y$ of linear equations using Gaussian elimination without pivoting. Consult pages 40-41 of *Johnson and Riess* for an algorithm. Your program should accept as inputs:

1. An n dimensional vector, which is the diagonal of T
2. An $n-1$ dimensional vector, which is the subdiagonal of T
3. An $n-1$ dimensional vector, which is the superdiagonal of T
4. An n dimensional vector, y

Your program should return either:

1. The solution of $Tx = y$, found by solving $Lz = y$ and $Ux = z$, or
2. An error message if a 0 is encountered in a pivot position. It is acceptable to have a solver function inside your program which returns both a solution and a numeric flag variable set to 0 if the solution exists and set to k=position of the 0 pivot. Presumably your main program would then use this flag to determine whether to output an error message or to return the solution.

Run your program on

(a)

$$T = \begin{bmatrix} 4 & 2 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 2 & 4 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 2 \\ 0 \\ 3 \\ 1 \end{bmatrix}$$

(b)

$$T = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 3 & 2 \\ 0 & 2 & 2 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

Here are some questions to think about when interpreting your results:

1. Does your program work as you expect it to?
2. How large is the residual vector $Tx - y$ for your computed solution y?
3. Find the exact solution in (a). What is the error in your computed solution? Did you get close to machine accuracy?

2 Gauss-Seidel – due Friday, November 5

Write a program to solve a linear system $Ax = b$ using the Gauss-Seidel method. Your program should accept as inputs:

1. An $n \times n$ matrix A
2. An n-vector b (the right hand side vector in the system)
3. An n-vector w (an initial guess for the solution x)
4. A positive number t (a stopping tolerance on the approximate x)
5. A positive number it (the maximum number of Gauss-Seidel iterations)

Your program should return either :

1. The first vector, $x^{(k)}$ (and its iteration index k), computed by the iteration, for which $\|x^{(k)} - x^{(k-1)}\| < t$, (using the norm $\|\quad\|_1$), if $k \leq it$.
2. An error message indicating that the stopping condition has not been met after it iterations

(a) Run your program on system (a) in computer project 1 with $w = 0$, $t = 10^{-4}$, and $it = 5$; and again with $it = 100$.

(b) Run your program on system (b) in computer project 1 with $w = 0$, $t = 10^{-4}$, and $it = 5$; and again with $it = 100$.

(c) Run your program on the following system with $w = 0$, $t = 10^{-4}$, and $it = 5$; and again with $it = 100$:

$$= \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 1 \\ 2 \\ 3 \\ 1 \\ 2 \\ 3 \end{bmatrix}$$

Remember to interpret your results. The questions used in project 1 are a good beginning.

3 Finding roots of nonlinear equations – due Friday, November 19

Newton's Method. Write a program to find a root of an equation $f(x) = 0$ using Newton's method. Your program should take as inputs the function, its derivative, and an initial guess for a root. You will need to include a test for convergence and a limit on the number of iterations permitted. Use your program to solve the following problems:

1. The polynomial $2x^4 + 24x^3 + 61x^2 - 16x + 1$ has two zeros near 0.1. Find them.
2. The polynomial $x^3 + 94x^2 - 389x + 294$ has zeros $1, 3, -98$. Try to find the two small zeros by starting near 2 and explain what happens.
3. For the complex polynomial $z^3 - 1$, find three nearby starting points so that the resulting sequence of Newton iterates converges to different roots.

Secant Method. Write a program to find a root of an equation $f(x) = 0$ using the secant method. Your program should take as inputs the function and two initial guesses for a root. You will need to include a test for convergence and a limit on the number of iterations permitted. Use your program to answer the first two questions above on Newton's method.

Fixed-Point Iteration. This is simply a Matlab exercise. Each of the following functions has a fixed point at $x = 1$. Apply 50 steps of fixed-point iteration starting with $x_0 = 1.1$. Observe what happens and explain the results.

1. $f_1(x) = x^3 - 6x^2 + 10x - 4$
2. $f_2(x) = x^3 - 2.4x + 2.4$
3. $f_3(x) = x^3 - 2.9x + 2.9$
4. $f_4(x) = x^3 - 3x + 3$

To interpret your results, develop and answer your own questions. For example:

- Does the method appear to converge for the given problems?
- How fast does the method converge?
- Is there anything in the theory we have studied that might have made it possible for you to predict what you have observed?

4 Newton Interpolating Polynomial–Due Monday, December 6

Write a program which takes as input:

1. An $n+1$-vector $x = [x_0, \dots, x_n]$ and an $n+1$-vector $y = [y_0, \dots, y_n]$.
2. The endpoints a and b of an interval $[a, b]$
3. A positive integer m.

The program should compute the coefficients $a_0, a_1, \dots, a_n$ of the Newton form of the interpolating polynomial

$$p(x) = a_0 + a_1(x - x_0) + \cdots + a_n(x - x_0) \cdots (x - x_{n-1})$$

The program should return two vectors of length $m+1$, $U = (u_0, \dots, u_m)$ and $V = (p(u_0), \dots, p(u_m))$, where where $u_0 = a$, $u_m = b$ and $u_k = a + k(b-a)/m$. *Nested multiplication* should be used to compute the values $p(u_k)$. In each of the following runs, use a plotting routine to plot a graph of the points $(u_0, p(u_0)), \dots, (u_m, p(u_m))$.

Run your program on the following problems:

(a) Use the values
$\{(1,1.89512),(2,4.95423),(3,9.93383),(4,19.6309),(5,40.1853),(6,85.9898)\}$.
In this case $n = 5$. Make a plot with $[a, b] = [1, 6]$ and $m = 50$.

(b) Use the values
$\{(0,3),(1,2),(2,3),(3,5),(4,3),(5,4),(6,3),(7,2),(8,2),(9,3),(10,2)\}$.
In this case $n = 10$. Make a plot with $[a, b] = [-2, 12]$ and $m = 70$.

(c) Use the data $(x_j, f(x_j))$, where $j = 0, \dots, 10$, $x_j = -1 + 0.2j$, and $f(x) = 1/(1 + 25x^2)$. Make a plot with $[a, b] = [-1, 1]$ and $m = 40$. Also compute the error $f(u_k) - p(u_k)$, $k = 0, \dots, 40$.

(d) Use the data $(x_j, f(x_j))$, where $j = 0, \ldots, 10$,

$$x_j = -\cos\left[\frac{2j+1}{2n+2}\pi\right]$$

and $f(x) = 1/(1+25x^2)$. In this case, $n = 10$. Make a plot with $[a, b] = [-1, 1]$ and $m = 40$. Also compute the error $f(u_k) - p(u_k)$, $k = 0, \ldots, 40$.