

- Instructions: 465/565 students must answer all the questions unless noted otherwise.

1. Secant and Newton Methods:

- a. Write a general function (using MATLAB preferably) for solving the (scalar) nonlinear equation $f(x) = 0$ using the *secant method*. Your function should take as input the scalar function $f(x)$, two initial starting guesses x_0 and x_1 , a stopping criterion ϵ , and an integer, N_{max} , for the maximum number of iterations that should be computed. Your function should output the approximate solution and the number of iterations required. An example MATLAB function declaration for your secant method is:

```
[x,iter] = secant(f,x0,x1,tol,Nmax)
```

Avoid unnecessary function evaluations. Turn in a listing of your program.

- b. Write a general function (using MATLAB preferably) for solving the (scalar) nonlinear equation $f(x) = 0$ using *Newton's method*. Your function should take as input the scalar function $f(x)$, the derivative of $f(x)$, an initial starting guess x_0 , a stopping criterion ϵ , and an integer, N_{max} , for the maximum number of iterations that should be computed. The output should be the same as your `secant` function. An example MATLAB function declaration is:

```
[x,iter] = newton(f,df,x0,tol,Nmax)
```

Turn in a listing of your program.

- c. Use your secant and Newton functions to solve the nonlinear equation

$$f(x) = e^{-x} - x = 0, \quad (1)$$

which has a solution $x = 0.56714329040978$. Construct a table for each method that reports how the successive iterates approach the true solution. Verify that the methods approximately converge at their respective theoretical rates. For the secant method use initial starting guesses $x_0 = 5.5$ and $x_1 = 5.25$. For Newton's method use the initial starting guess $x_0 = 5.25$. For both methods use the stopping criterion $\epsilon = 10^{-14}$.

See, for example, the tutorial "Introduction to MATLAB Programming II" on the course webpage (<http://math.boisestate.edu/~wright/courses/matlab/programmingintro2.html>) for an example of how to input functions as variables to other functions and how to evaluate them.

2. The almost universally used algorithm to compute \sqrt{a} is the recursion

$$x_{n+1} = \frac{1}{2} \left(x_n + \frac{a}{x_n} \right), \quad (2)$$

easily obtained by means of Newton's method. Other approaches than Newton's method are available for generating fast iterative methods for computing \sqrt{a} . One approach starts by noting that

$$\sqrt{a} = x(1-r)^{-\frac{1}{2}} \quad \text{where } r = 1 - \frac{x^2}{a}$$

is an identity for any (positive) value of x . Hence the iteration

$$x_{n+1} = x_n(1-r)^{-\frac{1}{2}} \quad \text{where } r = 1 - \frac{x_n^2}{a}$$

will converge in one step to \sqrt{a} . However, this iteration is useless since it requires the computation of a square root (we could have just as well computed \sqrt{a} directly!).

To make this iteration useful, we note that r becomes small if x_n is a good approximation to \sqrt{a} , which suggests replacing $(1-r)^{-\frac{1}{2}}$ by a truncated Taylor expansion about $r = 0$. By including different numbers of terms in the Taylor expansion, we get iterative methods of arbitrary order of convergence.

- (a) From the preceding discussion, derive the quadratically convergent recursion

$$x_{n+1} = \frac{x_n}{2} \left(3 - \frac{x_n^2}{a} \right). \quad (3)$$

Explain why it is quadratically convergent. Print out a table of successive iterates for $a = 2$ and $x_0 = 1$ and note how the number of correct digits increases with n .

- (b) Let $a = 2$ and $x_n = x$ in the right hand side of (2) and (3), then plot each of the resulting functions on separate graphs together with the line $y = x$. Using the graphical fixed-point iteration process discussed in class, estimate in which intervals around the root $x = \sqrt{2}$ each of these iterations will converge. Verify your prediction with numerical results.
3. Assume you are working with a very simple processor that only supports addition, subtraction, multiplication and halving (a subtraction of one in the exponent of a base-2 number), but not a general divide. Devise a fast algorithm using Newton's method for this processor to directly approximate $\frac{1}{\sqrt{a}}$ (it then only remains to multiply with a to get \sqrt{a}).

4. Cubically Convergent Newton's Method

- (a) Generalize the Taylor series argument presented in class that led to Newton's method so that it instead leads to the cubically convergent iteration

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{[f(x_n)]^2 f''(x_n)}{2[f'(x_n)]^3}$$

Hint: Let \bar{x} be the true solution to $f(x) = 0$ and $\Delta = \bar{x} - x_n$. Consider the Taylor expansion of $f(x_n + \Delta)$ about the point x_n :

$$0 = f(x_n + \Delta) \approx f(x_n) + \Delta f'(x_n) + \frac{\Delta^2}{2} f''(x_n). \quad (4)$$

Solve for the first Δ in the RHS to get $\Delta \approx -\frac{f(x_n)}{f'(x_n)} - \Delta^2 \frac{f''(x_n)}{2f'(x_n)}$. By setting $\Delta = 0$ in this RHS, we get in the LHS the Newton approximation for Δ . Plug this approximation into the Δ^2 term in the RHS of (4) to get what we want.

- (b) We can verify that the regular Newton's method is quadratically convergent using the following two lines of *Maple*

```
x1:=simplify(series(x0-f(x0)/diff(f(x0),x0),x0=0,3)):
x1:=subs(f(0)=0,x1);
```

Here we have simplified the algebra by assuming that the zero of $f(x)$ occurs at $x = 0$. This simplification is justified by the fact that Newton's method is invariant to translations in the x -direction.

Your task is to use *Maple* to similarly verify that the iteration in part a is indeed cubically convergent.

5. [565 only] Newton's method gets stuck repeating the infinite cycle $x_n = (-1)^n x_0$ when trying to find the zero of the function $f(x) = c \cdot \text{sign}(x) \cdot \sqrt{|x|}$, no matter how the initial guess x_0 is chosen. Show that this is the only function with this property (apart from functions which differ only in trivial sign changes).
6. [565 only] As we discussed in class, the *growth factor* associated with Gaussian elimination with partial pivoting of a n -by- n matrix A is defined as

$$g(n) = \frac{\max_{1 \leq i, j, k \leq n} |a_{ij}^{(k)}|}{\max_{1 \leq i, j \leq n} |a_{ij}|}.$$

Show that the growth factor of the matrix

$$A = \begin{bmatrix} -1 & 0 & 0 & \cdots & \cdots & 0 & 1 \\ -1 & 1 & 0 & \cdots & \cdots & 0 & 1 \\ -1 & -1 & 1 & \ddots & \ddots & 0 & 1 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ -1 & -1 & -1 & \cdots & \cdots & -1 & 1 \end{bmatrix}$$

is $g(n) = 2^{n-1}$.