Homework No.2, 553.481/681, Due February 23, 2024.

Problem 1. (a) Compute $\sqrt{5}$ by using the bisection method to solve the equation $x^2 = 5$ on $[2, 3]$ with tolerance set to $tol = 10^{-15}$. State the total number of iterations required and also the wall clock time (obtained in Matlab with `tic` and `toc` commands). Rewrite the Matlab script `bisect.m` to save the sequence of iterates and use these to estimate the constants c and p in the asymptotic error relation

$$e_{n+1} \sim c\, e_n^p$$

where $e_n = |\sqrt{5} - x_n|$ is the error of the nth iterate. You can obtain these constants by plotting $\ln e_{n+1}$ versus $\ln e_n$ and using `polyfit` in Matlab to fit a straight line. Compare your numerical results to the theoretical estimates for both c and p.

(b) Repeat part (a) for the Newton method with initial guess $x_0 = 2$.

(c) Repeat part (a) for the secant method with initial guess $a = 2$ and generate b with the Newton method (counting this as the first iteration for secant).

(d) Repeat part (a) for the IQI method with initial guess $a = 2$ and generate b with the Newton method and c with the secant method (counting these as the first and second iterations for IQI).

Problem 2. (a) Find a condition on the initial guess x_0 so that Newton's method for solving $f(x) := \frac{x^4}{1+x^4} = 0$ diverges. Verify numerically that the method diverges when this condition is satisfied.

(b) Consider now an initial guess x_0 for which Newton's method does converge and use your numerical results to estimate the asymptotic rate of convergence. Explain how your results are consistent with theoretical estimates for convergence rates of Newton's method.

Problem 3. Given below is a table of iterates from a linearly convergent iteration $x_{n+1} = g(x_n)$. Estimate from this data (a) the rate of linear convergence, (b) the error $x_7 - x_*$ and (c) the fixed point x_*.

n	x_n
0	1.0000000
1	0.4319787
2	0.6461285
3	0.5869089
4	0.6057665
5	0.5999709
6	0.6017728
7	0.6012146

Given that the true fixed point is $x_* = 0.601346767725820$ to 16 decimals, compare your best estimates for x_* and the error $x_7 - x_*$ with the true values.

Problem 4. Show that the iterative method to solve $f(x) = 0$ given by:

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

for $n = 1, 2, \cdots$, will generally yield cubic convergence.

Problem 5. (a) Verify that the Sherman-Morrison formula is correct by checking that

$$\left[\mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{x}\mathbf{y}^\top\mathbf{A}^{-1}}{1+\langle \mathbf{y}, \mathbf{A}^{-1}\mathbf{x}\rangle}\right](\mathbf{A}+\mathbf{x}\mathbf{y}^\top) = \mathbf{I}.$$

(b) Use the Sherman-Morrison formula to verify that the approximation $D\mathbf{f}(\mathbf{x}_n) \doteq \mathbf{A}_n$ used in Broyden's method has an inverse $\mathbf{A}_n^{-1}$ that can be iterated with the formulas

$$\Delta\mathbf{p}_{n-1} = \mathbf{A}_{n-1}^{-1}\Delta\mathbf{f}_{n-1}, \quad \mathbf{A}_n^{-1} = \mathbf{A}_{n-1}^{-1} + \frac{(\Delta\mathbf{x}_{n-1} - \Delta\mathbf{p}_{n-1})\,\Delta\mathbf{x}_{n-1}^\top\mathbf{A}_{n-1}^{-1}}{\langle \Delta\mathbf{x}_{n-1}, \Delta\mathbf{p}_{n-1}\rangle}$$

where $\Delta\mathbf{x}_{n-1} = \mathbf{x}_n - \mathbf{x}_{n-1}$ and $\Delta\mathbf{f}_{n-1} = \mathbf{f}(\mathbf{x}_n) - \mathbf{f}(\mathbf{x}_{n-1})$.

Problem 6. Use both Newton-Raphson and Broyden's methods for a starting vector $(x^{(0)}, y^{(0)}, z^{(0)}) = (0,0,2)$ to approximate the solution of the following system of equations to tolerance $tol = eps$:

$$\begin{cases} x^2+y^2+z^4-16=0 \\ (x-1)^2+xy+y^2-1=0 \\ x+y+z\ln z-2=0 \end{cases}$$

Compare the number of iterations and also the wall clock time required for this accuracy with the two methods. In order to make a more accurate estimate of the relative clock time, repeat the root-finding a large number of times N_{repeat} for both methods (in the same loop) and average over the repeated trials.

Problem 7*. (a) Here we illustrate the *relaxation method* to solve nonlinear boundary problems by converting them into high-dimensional root-finding problems. As an example, we consider the nonlinear elliptic problem

$$-x''(u) + 6\,(x(u))^2 = 0,\ u\in[1,2], \quad x(1)=1, \quad x(2)=\frac{1}{4}$$

whose exact solution can easily be checked to be $x(u) = u^{-2}$. In the relaxation method, one introduces a numerical grid $u_i = 1 + \frac{i}{n+1}$, $i = 0,1,...,n+1$ with spacing $\Delta u = \frac{1}{n+1}$ and corresponding function values $x_i \doteq x(u_i)$. With a finite-difference approximation to the second derivative

$$x''(u_i) \doteq \frac{x_{i+1}+x_{i-1}-2x_i}{(\Delta u)^2}$$

the elliptic ODE becomes a nonlinear fixed-point condition $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ for the solution vector $\mathbf{x} = (x_1, x_2, ..., x_n)^\top$. Use the Newton-Raphson method to solve the resulting fixed-point problem for $n = 1000$ with initial guess $x_0(u) = (7-3u)/4$ to a tolerance $tol = 10^{-8}$, and report the number of iterations and the error estimate at each iteration. Finally, plot your numerical solution x_i versus u_i for $i = 0,1,2,...,101$ together with the exact solution $x(u_i)$ for those same u_i values.

(b) Repeat part (a) using Broyden's method.

(c) Repeat part (a) using the Levenberg-Marquardt method implemented in Matlab's function `fsolve`. Use the function `optimoption` to choose the algorithm in `fsolve` to be Levenberg-Marquardt, to set `display` to `iter`, and to set `FunctionTolerance` to 10^{-8}.

Problem 8*. The Levenberg-Marquardt algorithm is an example of a method which solves nonlinear equations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ by *least-squares minimization*, or, in other words, by minimizing the scalar function

$$\Phi(\mathbf{x}) = \frac{1}{2}\|\mathbf{f}(\mathbf{x})\|_2^2 \tag{$*$}$$

by a descent algorithm which combines *Steepest Descent* and *Gauss-Newton* methods. These methods make updates of the successive iterates by $\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n \mathbf{d}_n$ where the vector $\mathbf{d}_n$ is a <u>descent direction</u> for which

$$\langle \mathbf{d}_n, \nabla\Phi(\mathbf{x}_n)\rangle \leq 0.$$

(a) Find the direction $\mathbf{d}_n$ which is the direction of steepest descent for the scalar function $\Phi(\mathbf{x})$ defined in (*) at the point $\mathbf{x} = \mathbf{x}_n$.

(b) Show that the Newton update vector $\Delta\mathbf{x}_n := -(\mathbf{Df}(\mathbf{x}_n))^{-1}\mathbf{f}(\mathbf{x}_n)$ is a descent direction for $\Phi(\mathbf{x})$ defined in (*) at $\mathbf{x} = \mathbf{x}_n$.