Numerical Methods I, Fall 2010
Assignment IV: Non-linear Equations and Optimization

Aleksandar Donev
Courant Institute, NYU, donev@courant.nyu.edu

October 21st, 2010
Due: November 4th, 2010

Choose the problems that interest you, including any of the extra credit ones. Anything above 60 points is very good (A), above 70 is excellent (A+). “Extra credit” simply marks problems that are more free-style and do not come with very specific directions.

1 Newton-Raphson Method and Variants

It is recommended that you choose one of these two problems:

1.1 [10 pts] Multiple roots

Assume that we are using Newton's method in the vicinity of a root of multiplicity $m > 1$, meaning that the first nonzero derivative of the function $f(x)$ is $f^{(m)}(x)$.

1. [5pts] Show that Newton's method gives linear convergence. Then show that modifying Newton's method to account for the multiplicity,
$$x^{n+1} = x^n - m\frac{f(x^n)}{f'(x^n)},$$
gives second-order of convergence.
2. [5pts] Also show that applying Newton's method to the equation
$$\tilde{f}(x) = \frac{f(x)}{f'(x)} = 0$$
restores the quadratic convergence (but requires evaluating second-derivatives of f as well).
Note: This is not something to do in practice, it is merely a toy example to practice analysis on!

1.2 [Up to 20 pts] Steffensen's Method

Newton's method requires evaluation of the first derivative of the function, which can be a problem. An obvious alternative is to use a finite-difference approximation for the derivative:
$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$
which however has some truncation and roundoff error in it (recall problem 3.1 in homework 1). *Note: This is not something to do in practice, it is merely a toy example to practice analysis on!*

Our task is to figure out how to choose h to get the best convergence without too much extra work. Note that the centered difference requires evaluating the function also at $f(x-h)$ so it costs a lot more and is likely not worth it. The secant method sets $h = x_{n-1} - x_n$ so that the previous function evaluation $f(x_{n-1})$ can be reused. However, the secant method is not second-order convergent.

1. [10pts] Intuition says that in order to get true (formal) second-order convergence the derivative should become more accurate near the root, i.e., h should become smaller as $f(x)$ becomes smaller. Indeed, Steffensen's method sets $h = f(x)$. Prove that this choice gives second-order convergence. [*Hint: Express Steffensen's method as a fixed-point iteration and then apply the convergence theory from the lectures. Be careful in calculating the derivatives using the chain rule, and be very careful about the limit $x \to \alpha$.*]
2. [10pts extra credit] Implement Steffensen's method in MATLAB and verify second-order convergence. Is roundoff a problem and if so can you do anything about it?

1.3 [30 pts] Newton's Method for Simple Roots

Consider finding the three roots of the polynomial

$$f(x) = 816x^3 - 3835x^2 + 6000x - 3125,$$

which happen to all be real and all contained in the interval [1.4, 1.7] [due to Cleve Moler].

1. [5pts] Plot this function on the interval [1.4, 1.7] and find all of the zeros of this polynomial using the MATLAB function *fzero* [*Hint: The roots values can be obtained in MATLAB using the built-in function roots but Maple tells us the roots are* 25/16, 25/17 *and* 5/3].
2. [10pts] Implement Newton's method (no safeguards necessary) and test it with some initial guess in the interval [1.4, 1.7]. Verify that the order of convergence is quadratic, as predicted by the theory from class:
$$\frac{\left|e^{k+1}\right|}{\left|e^k\right|^2} \to \left|\frac{f''(\alpha)}{2f'(\alpha)}\right|.$$
[*Hint: Due to roundoff errors and the very fast convergence, the error quickly becomes comparable to roundoff, so one must be careful not to use very large* k]
3. [15pts] Starting from many (say 100) guesses in the interval [1.4, 1.7], run 100 iterations of Newton's method and see plot the value to which it converges, if it does, as a function of the initial guess. If the initial guess is sufficiently close to one of the roots α, i.e., if it is within the *basin of attraction* for root α, it should converge to α. What is the basin of attraction for the middle root ($\alpha = 25/16$) based on the plot?
The theory from the lecture suggested an estimate for the width of each basin of attraction around a given root α of the form:
$$\left|x^0 - \alpha\right| \le \left|\frac{f''(\alpha)}{2f'(\alpha)}\right|^{-1}.$$
Compare these estimates to what is actually observed numerically (you do not have to separately plot the estimates, just estimate the intervals and write them down). Is the estimate particularly bad for one of the roots, and if so, can you think of reasons why?

2 [65 pts] Quadratically-Constrained Quadratic Optimization

Consider the quadratically-constrained quadratic convex optimization problem of finding the point on an ellipse/ellipsoid that is closest to the origin:

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ f(\boldsymbol{x}) = \|\boldsymbol{x}\|_2^2 = \boldsymbol{x}\cdot\boldsymbol{x} = \textstyle\sum_{i=1}^n x_i^2 \right\} \\ &\text{s.t.} \quad (\boldsymbol{x}-\boldsymbol{x}_0)^T \boldsymbol{A} (\boldsymbol{x}-\boldsymbol{x}_0) = \textstyle\sum_{i,j=1}^n a_{ij}(x_i - x_{0,i})(x_j - x_{0,i}) = 1 \quad . \end{aligned} \tag{1}$$

where $\boldsymbol{A}$ is a symmetric positive-definite matrix and $\boldsymbol{x}_0$ is the location of the centroid of the ellipsoid. Note that if the sign (direction) of $\boldsymbol{x}$ is reversed it is still a solution (this non-uniqueness may cause some numerical problems!).

2.1 [20pts total] Analytical Solution

1. [5pts] Focus on the case $\boldsymbol{x}_0 = 0$. Consider a change of coordinates in which the matrix $\boldsymbol{A}$ is diagonal, i.e., compute the eigenvalue decomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\star$ [*Hint: It is unitarily diagonalizable and all eigenvalues are positive*]. The geometric interpretation is that the unitary matrix $\boldsymbol{U}$ is a rotation matrix and the eigenvalues are the inverse squares of the semiaxes of the ellipsoid. Express $\boldsymbol{x}$ in the orthonormal basis formed by the eigenvectors of $\boldsymbol{A}$ and transform the optimization problem to the new basis, and use this to solve it analytically.
2. [5pts] As an example, consider minimizing $x_1^2 + x_2^2$ along the ellipse
$$x_1^2 + 2x_1x_2 + 3x_2^2 = 1. \tag{2}$$
Write down the matrix $\boldsymbol{A}$. Find the solution analytically (meaning with pen-and-paper), for example, using the solution in part 1 above, or maybe in part 3 below, and evaluate it numerically. [*Hint: The exact solution is* $x_1^\star = \frac{1}{\sqrt{2}} - \frac{1}{2}$ and $x_2^\star = \frac{1}{2}$].

3. [10pts total] Write the Lagrangian function $\mathcal{L}$ and the first-order optimality conditions.
   [5pts] For the case $\boldsymbol{x}_0 = \mathbf{0}$, can you interpret the optimality conditions as an eigenvalue problem and solve it [*Hint: This is closely related to part 1 above*].

2.2 [Up to 45pts total] Numerical Solution via the Penalty Method

Write a MATLAB (or other) program that implements Newton's method for minimizing the *penalty function* (1)

$$\min_{\boldsymbol{x}} \left\{ \mathcal{L}_\alpha = f(\boldsymbol{x}) + \alpha \left[h(\boldsymbol{x}) \right]^2 \right\} \tag{3}$$

for a given penalty parameter α and some reasonable initial guess, e.g., $\boldsymbol{x}^0 = \mathbf{1}$ (all ones) or $\boldsymbol{x}^0 = randn(n, 1)$. Try to write the MATLAB code so that it works for any dimension of the problem n and any ellipsoid, i.e., for any $\boldsymbol{A}$ and $\boldsymbol{x}_0$. [*Hint: If you implemented Newton's method correctly the convergence will be quadratic, which should be easy to see if you print some convergence statistics.*]

1. [15pts] For the two-dimensional example (2) from part 2.1 above, solve the penalized problem numerically for increasing penalty parameter $\alpha = \alpha_k = 10^k$ for $k = 0, 1, \ldots$, stopping when the increment becomes too small, for example, $\left\| \boldsymbol{x}_{k+1} - \boldsymbol{x} \right\|_\infty \leq \varepsilon = 10^{-12}$ [*Hint: You may get faster convergence if you use the last known solution as an initial guess for the next* α].
   Plot the error in the solution to (3) as compared to the exact answer as a function of α. How large does α need to be before you can get a solution accurate to 6 significant digits?
2. [5pts] Apply the same approach as in part 1 above to the same matrix $\boldsymbol{A}$ but with $\boldsymbol{x}_0 = [0.1, -0.2]$ [*Hint: The solution is* $\boldsymbol{x}^\star = [0.126553521589213, 0.368363384837439]$].
   [Up to 15pts extra credit] Explore whether Newton's method is likely or guaranteed to converge for a couple of values of α, using whatever method you choose.
3. [10pts] Apply the penalty method to a randomly-generated three-dimensional problem with $\boldsymbol{x}_0 = \mathbf{0}$ [*Hints: A random matrix* $\boldsymbol{A}$ *can be generated in MATLAB using the function* $gallery('randcorr', n)$. *Verify your answer against the analytical solution from part 1.*].
4. [15pts] You will observe that the convergence of the solution to problem (3), $\boldsymbol{x}(\alpha)$, to the solution of (1), $\boldsymbol{x}^\star = \boldsymbol{x}(\alpha \to \infty)$ is slow and a rather large α is required to get an accurate answer. One can accelerate the convergence by using a technique similar to Richardson's extrapolation. The idea is to postulate how $\boldsymbol{x}(\alpha)$ depends on α and then extrapolate to $\alpha \to \infty$. Assume that

   $$\boldsymbol{x}(\alpha) \approx \boldsymbol{x}^\star - \alpha^{-1} \boldsymbol{c}$$

   for some vector $\boldsymbol{c}$. As you change the penalty α_k, for example, $\alpha = \alpha_k = 10^k$, you can estimate $\boldsymbol{c} \approx \boldsymbol{c}(\alpha_k)$ from the last two solutions, $\boldsymbol{x}(\alpha_{k-1})$ and $\boldsymbol{x}(\alpha_k)$, and use that estimate to compute an improved estimate of $\boldsymbol{x}^\star$,

   $$\boldsymbol{x}^\star(\alpha_k) = \boldsymbol{x}(\alpha_k) - \alpha_k^{-1} \boldsymbol{c}(\alpha_k).$$

   For the example (2), compute the accelerated sequence $\boldsymbol{x}^\star(\alpha)$ for $k = 1, 2, \ldots$ and plot its error versus α. How fast does the accelerated sequence converge compared to the original sequence $\boldsymbol{x}(\alpha)$ computed in part 1 above?
   Note that this process can in principle be repeated recursively to improve the estimate even further!