Homework #3, due November 8

Objective: To practice optimization and methods and robust numerical software.
Matlab Demo: Toolboxes, Optimization

(1) Explore the Line Search

(a) Write a program that attempts to minimize a non-constant function of a single variable, $\phi(t)$. The algorithm should have two phases. The first phase searches for some $0 < t_1 < t_2 < t_3$ such that $\phi(t_1) > \phi(t_2) < \phi(t_3)$. Then we take (t_1, t_3) as the starting points of the Golden section search, or take t_2 as the starting point for the Newton's method. Do this by computing $\phi(t)$ and $\phi(2t)$ for some $t > 0$, say, $t = 1$. If $\phi(t) < \phi(2t)$ and $\phi(t) < \phi(0)$, then t is the end of phase one. If $\phi(2t) < \phi(t)$, then replace t by $2t$ and repeat. If $\phi(t) > \phi(0)$, replace t by $t/2$ and repeat. If these steps do not yield a reasonable t value after a reasonable time, give up and report failure.

(b) The second phase is a Golden section search for the minimum. Try your algorithm on the function $\phi(t) = -15/(15 + (t - 3\pi)^2)$ and observe the rate of convergence.

(c) Repeat **(b)** using Newton's method instead to search for the minimum. This method should have safeguards against negative ϕ'' and against wild iterates (e.g., those that violate the criteria of phase 1). Describe the behavior of this method. In particular, does the method behave as expected in the theory?

(2) Explore the safeguards in Newton's methods.

This problem (and the next problem) asks you to write good, robust numerical software for finding the minimum of a function of 2 variables. In the real world, you would minimize a function of n variables (see problems 4-7). This problem keeps some of the flavor of the more general problem but leaves out some (for this purpose) irrelevant details.

(a) Write a program that performs the unsafe-guarded Newton's method for a differentiable function of two variables. To do this, you will need a subroutine that solves the linear system $Ax = b$ when A is a 2×2 symmetric matrix. When x is close to a minimum point, A should be positive definite, so a Cholesky factorization of A is possible. Try your program on the function

$$f(x_1, x_2) = -\frac{1}{1 + \left(x_1^2 + a(x_2 - b\sin(cx_1))^2\right)^2} \quad . \tag{1}$$

For the case when $a = 2$ and $b = c = 0$, try initial guesses close and far from the optimal solution. With a good initial guess you should observe quadratic convergence to the minimizer. With a poor initial guess you should observe wild behavior.

(b) Add two safeguards to the program from **(a)**: do a phase one line search to make sure the step is reasonable and check that the Newton step is in fact a descent direction. Use this to minimize the objective function (1) with $a = 30$, $b = 1$, and $c = 6$, starting from initial guess $(x_1, x_2) = (3, 1)$. Make a contour plot of the objective function showing the iterates.

(3) Explore the rate of convergence and MATLAB optimization toolbox.

Apply your code in **2(a)** to the following function

$$f(x, y) = 100(y - x^2)^2 + (1 - x)^2,$$

which has a (global) minimum at $x = y = 1$. (For this example, the unsafe-guarded Newton's method should be OK.)

(a) Try your code on the function $f(x, y)$ for several different initial guesses. Comment on the rate of convergence. Make a 3-D surface plot of $f(x, y)$ and plot the Newton iterates on the same graph. This can be done by evaluating f at uniformly spaced points in x and y and using the MATLAB command `mesh(x,y,f)`. Then enter the command `hold on` to keep the surface plot on the screen, and plot each of the Newton iterates by typing `plot3(xn,yn,fn,'x')`, where arrays xn and yn contain the Newton iterates and fn contains the function values at these points. You can look at your plot from different angles by using the `view` command in MATLAB. Turn in a listing of your code and one plot showing the function $f(x, y)$ and the Newton iterates.

(b) If you have access to the MATLAB optimization toolbox, run the demo `bandem` to see an interesting illustration of the performance of different methods on this problem. Nothing to turn in here, just enjoy.

(4) Explore MATLAB optimization toolbox.

We want to find a surface $u(x, y)$ for x, y in the square $[0, 1]^2$ which satisfies the boundary conditions

$$u(x, 0) = 0, \quad u(0, y) = 0, \quad u(x, 1) = \max(0, 2x - 1), \quad u(1, y) = \max(0, 2y - 1)$$

and has minimal surface area. We divide $[0, 1]^2$ in N^2 squares and each of the small squares into two triangles (using a line parallel to $x = y$). Denote $x_i = ih$ and $y_j = jh$, $h = 1/N$. The values $u(x_i, y_j)$ are approximated by $u_{i,j}$ for $i, j = 0, \cdots, N$. The function $u(x, y)$ is approximated by the piecewise linear function on the triangular mesh with nodal value $u_{i,j}$, yielding a surface of triangles. For $x \in [x_i, x_{i+1}]$, $y \in [y_j, y_{j+1}]$, the triangular mesh consists of two triangles which have a total area of

$$\frac{h^2}{2}\left(\sqrt{1 + (\frac{u_{i+1,j} - u_{i,j}}{h})^2 + (\frac{u_{i+1,j+1} - u_{i+1,j}}{h})^2} + \sqrt{1 + (\frac{u_{i,j+1} - u_{i,j}}{h})^2 + (\frac{u_{i+1,j+1} - u_{i,j+1}}{h})^2}\right)$$

Use the Matlab function `fminu` to find values $u_{i,j}$ which minimize the above approximant of the total surface area for $N = 8$. Set `opts(1)=1` to see the progress of the minimization and plot the resulting surface.

(5) Explore MATLAB optimization toolbox.

We want to find a surface $u(x, y)$ for x, y in the square $[0, 1]^2$ which satisfies

$$u(x, 0) = 0, \quad u(0, y) = 0, \quad u(x, y) \geq 1 - 2(1 - x)^2 - 2(1 - y)^2$$

and has minimal surface area. Use the Matlab function `constr` to find values $u_{i,j}$ which minimize the approximant of the total surface area in problem (4) for $N = 8$. Set `opt(1)=1` to see the progress of the minimization and plot the resulting surface. Which of the constraints are active?

(6) Explore the conjugate gradient method

Approximate the following 1-D Poisson equation with Dirichlet boundary conditions:

$$-u''(x) = f(x), \quad \text{for } 0 < x < 1, \quad u(0) = u(1) = 0, \tag{2}$$

by central difference scheme:

$$\frac{-u(x_{j+1}) + 2u(x_j) - u(x_{j-1})}{h^2} = f(x_j), \qquad u(x_0) = u(x_N) = 0 \tag{3}$$

where $x_j = jh$, $h = 1/N$ for $j = 0, 1, \cdots, N$ are the computational grids.

(a) Write (3) as a linear system $Au = b$ and show that A is positive definite.

(b) Show that the solution to the above linear system is the minimizer of the following functional:

$$\phi(u) = \frac{1}{2} u^\top A u - b^\top u \tag{4}$$

(c) Sketch the steps and write a program for the conjugate Gradient method solving the linear system (3). The program should be modular and contain two subroutines. One computes the inner product $u^\top v$. And the other one multiplies a matrix by a vector (so you are able to avoid formulating the matrix A in your code).

(d) Extend your program to a general domain $[a, b]$ and inhomogeneous boundary conditions. Observe the rate of the convergence in the residual.

(e) (optional) Extend **(a)-(d)** to 2D case and write a generic code solving the Poisson equation.

(7) Explore the Quasi-Newton methods

Instead of the conjugate gradient method in **(6)**, use the Davidon-Fletcher-Powell (DFP) method to find the minimum for (4). DFP is a popular class of quasi-Newton method. If the initial approximation H_0 is taken equal to the identity matrix, then the DFP method becomes the conjugate gradient methods.

(8) Explore the Broyden's method for systems of nonlinear equations.

(a) Carry out two iterations of Broyden's method on

$$F(x_1, x_2) = \begin{bmatrix} x_1 + x_2 - 3 \\ x_1^2 + x_2^2 - 9 \end{bmatrix},$$

starting with $x^0 = (2, 7)^\top$ and $A_0 = \nabla F(x^0)$.

(b) Continue part (a) on a computer until $\|x^{k+1} - x^k\| \leq (\text{machine eps})^{1/2}$. What is the final value of A_k? How does it compare with $\nabla F(x_*)$?