Instructor: Emre Mengi

Math 409/509 (Spring 2011)

Study Guide for Homework 2

This homework concerns the root-finding problem and line-search algorithms for unconstrained optimization. Please don't hesitate to ask for help if any of these topics is unclear. Section numbers refer to the sections of the class-text by Philip Gill and Margaret Wright.

- Newton's method for finding the roots of a univariate function (Section 2.2.2)
- Secant method (Section 2.2.3)
- Convergence of sequences and rate of convergence (Section 1.5)
- The convergence properties of Newton's method for univariate functions
- Newton's method for finding the roots of a multivariate function (Section 2.4)
- Matrix norms (Section 1.3.2)
- Convergence properties of Newton's method (Section 2.4.2; see also Theorem 11.2 in Nocedal and Wright)
- Line search methods; put emphasis on Goldstein-Armijo line search with backtracking (Section 3.5 with emphasis on Section 3.5.2)
- Steepest descent (Sections 3.4.2 and 3.4.3)
- Newton's method for optimization (Sections 3.4.4 and 3.4.5)

Homework 2 (due on March 30th, Wednesday by 14:00)

- Questions 1, 5, 11 and 12 require computations in Matlab. Please attach Matlab outputs, plots and m-files that you implemented in these questions.
- You must turn in the solutions to Questions 11 and 12. Please also turn in the solutions to any five questions out of the remaining ten questions.

1. Consider the polynomials $f(x) = (x^2 + 1)(x - 1)$ and $g(x) = (x - 1)^2$.

(a) Give the update rules of Newton's method and the secant method for the functions f and g.

(b) Suppose that the Newton sequences for both $f(x)$ and $g(x)$ converge to the root $x_* = 1$. Using the update rules from part **(a)** derive the q-rates of convergence of Newton's method for f and g.

(c) Generate an m-file `fun.m`

```
function [f,g] = fun(x)
% Task : Computes the function (x-1)^2 and its derivative.
f = (x-1)^2;
g = 2*(x-1);

return
```

which computes the function $g(x) = (x-1)^2$ and the derivative $g'(x) = 2(x-1)$ and stores these values in the output arguments f and g, respectively.

Generate a similar m-file similar to calculate the function value of $f(x) = (x^2+1)(x-1)$ and its derivative. The name of the m-file must be same as the function name.

(d) A Matlab implementation of Newton's method `Newton.m` for zero finding is provided on the course webpage. Type `help Newton` for information about its input and output arguments. Specifically to compute the root of a function $F : \mathbb{R}^n \to \mathbb{R}^m$ Newton's method needs the function value $F(x)$ and the Jacobian $F'(x)$ at every iteration. The second input argument to `Newton.m` is the name of an m-file that takes a vector $x \in \mathbb{R}^n$ as input and returns the function value $F(x)$ and the Jacobian $F'(x)$ as output. For instance type

```
>> [xr,fr] = Newton(2,'fun');
```

to retrieve the root of $g(x) = (x-1)^2$ starting from the initial guess $x_0 = 2$ after generating the m-file `fun.m` in **(c)**.

Using the Matlab routine provided compute the roots of $f(x) = (x^2+1)(x-1)$ and $g(x) = (x-1)^2$ with the initial guess $x_0 = 2$. Include your Matlab output. Do you observe the convergence rates that you derived in part **(b)**?

(e) Implement the secant method by modifying the m-file `Newton.m`. Run your implementation to find the roots of $f(x) = (x^2+1)(x-1)$ and $g(x) = (x-1)^2$ starting from $x_0 = 2$ and $x_1 = 2.2$ (recall that the secant method requires two initial guesses). How fast does the secant method converge as compared to Newton's method for these two functions?

2. Find the linear approximations used in Newton's method for the following functions about given points.

(a)

$$f : \mathbb{R}^3 \to \mathbb{R} \qquad f(x) = \frac{1}{2} x^T \begin{bmatrix} 1 & 3 & 0 \\ 3 & 1 & 3 \\ 0 & 3 & 1 \end{bmatrix} x + \begin{bmatrix} 2 & -5 & 1 \end{bmatrix} x$$

about $(1, -2)$

(b)

$$f : \mathbb{R}^2 \to \mathbb{R}^2, \qquad f(x) = \begin{bmatrix} x_2^2 - 1 \\ \sin x_1 - x_2 \end{bmatrix}$$

about $(0, 1)$

(c) $f(x) = \ln(x^T x)$ where $x \in \mathbb{R}^n$ about $x = [1\ 1\ \ldots\ 1]^T$ (vector of ones).

3. Carry out one iteration of Newton's method on the function $F : \mathbb{R}^2 \to \mathbb{R}^2$

$$F(x) = \begin{bmatrix} (x_1 + 3)(x_2^3 - 7) \\ \sin(x_2 e^{x_1} - 1) \end{bmatrix}.$$

starting with the initial guess $x_0 = (0, 1)$.

4. Let $F : \mathbb{R}^n \to \mathbb{R}^m$ be a continuously differentiable function such that there exists a positive scalar μ satisfying $\|F'(x)^{-1}\| \leq \mu$ for all $x \in \mathbb{R}^n$. Suppose also that $x_* \in \mathbb{R}^n$ is a root of $F(x)$. Show that for any point $x_k \in \mathbb{R}^n$ the inequality

$$\|x_k - x_*\| \leq \mu \|F(x_*) - L(x_*)\|$$

holds where $L : \mathbb{R}^n \to \mathbb{R}^m$ denotes the linear approximation for $F(x)$ about x_k. (Note: you can assume all norms are matrix or vector 2-norms.)

5. (Nocedal&Wright, Example 11.1) The equilibrium equations for a particular aircraft are given by a system of 5 equations in 8 unknowns of the form

$$F(x) \equiv Ax + \phi(x) = 0$$

where $F : \mathbb{R}^8 \to \mathbb{R}^5$, the matrix A is given by

$$A = \begin{bmatrix} -3.933 & 0.107 & 0.126 & 0 & -9.99 & 0 & -45.83 & -7.64 \\ 0 & -0.987 & 0 & -22.95 & 0 & -28.37 & 0 & 0 \\ 0.002 & 0 & -0.235 & 0 & 5.67 & 0 & -0.921 & -6.51 \\ 0 & 1.0 & 0 & -1.0 & 0 & -0.168 & 0 & 0 \\ 0 & 0 & -1.0 & 0 & -0.196 & 0 & -0.0071 & 0 \end{bmatrix}$$

and the nonlinear part is defined by

$$\phi(x) = \begin{bmatrix} -0.727x_2x_3 + 8.39x_3x_4 - 684.4x_4x_5 + 63.5x_4x_2 \\ 0.949x_1x_3 + 0.173x_1x_5 \\ -0.716x_1x_2 - 1.578x_1x_4 + 1.132x_4x_2 \\ -x_1x_5 \\ x_1x_4 \end{bmatrix}.$$

The first three variables x_1, x_2, x_3 represent the rates of roll, pitch and yaw, respectively, while x_4 is the incremental angle of attack and x_5 is the sideslip angle. The last three variables x_6, x_7, x_8 are controls; they represent the deflections of the elevator, aileron, and rudeer, respectively.

For the particular control values $x_6 = x_7 = x_8 = 1$ we obtain a system of 5 equations in 5 unknowns. Solve the resulting system by using the routine `Newton_backtrack.m` (provided on the course webpage) to determine the equilibrium state for these particular values of the control variables.

You should call `Newton_backtrack.m` exactly the same way as `Newton.m`. The m-file `Newton_backtrack.m` converges to a solution from any initial point, unlike `Newton.m` which converges if the initial point is sufficiently close to a solution. (This is optional, but it may be a good idea to try to understand the m-file `Newton_backtrack.m`. This is an implementation of Newton's method with backtracking; see section 2.6 in Gill&Wright. Newton's method with backtracking is not covered in class.)

6. Consider the vector-valued function $F : \mathbb{R}^2 \to \mathbb{R}^2$ such that

$$F(x) = \begin{bmatrix} x_1^2 + x_2 \sqrt[3]{x_2} - 1 \\ (x_1 - 1)^2 + (x_2 - 1)^2 - 1 \end{bmatrix}.$$

Assume a sequence $\{x_k\}$ generated by Newton's method for $F(x)$ converges to the root $x_* = (1, 0)$. Does the sequence $\{x_k\}$ converge to $(1, 0)$ q-linearly, q-superlinearly or q-quadratically? Explain. (Note: Don't try to derive the order of convergence. Rely on a theorem discussed in class.)

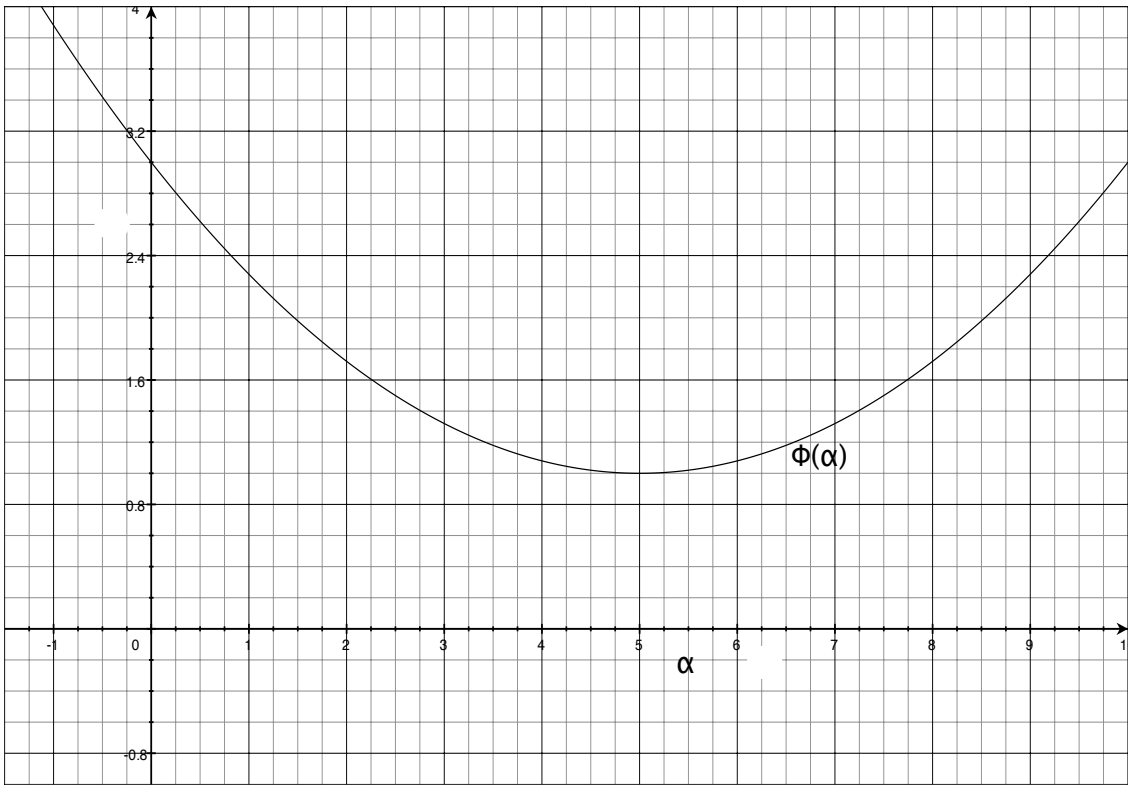
7. Given a multivariate function $f : \mathbb{R}^n \to \mathbb{R}$, a point $\bar{x}$ and a search direction $\bar{p}$. Suppose that the graph of the function $\phi : \mathbb{R} \to \mathbb{R}$

$$\phi(\alpha) = f(\bar{x} + \alpha\bar{p})$$

is as provided below. Note that $\phi'(0) = -0.8$. On the figure plot the graph of the line

$$\ell(\alpha) = f(\bar{x}) + \mu_1 \alpha \nabla f(\bar{x})^T \bar{p}$$

for $\mu_1 = 0.25$. Shade the intervals of α values satisfying the Armijo sufficient decrease condition on the horizontal axis.



8. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice continuously differentiable function. Let $x_k \in \mathbb{R}^n$ and $p_k \in \mathbb{R}^n$ be a descent direction satisfying $\nabla f(x_k)^T p_k < 0$. A point $x_k + \alpha p_k$ satisfies the Armijo sufficient decrease condition if

$$f(x_k + \alpha p_k) \leq f(x_k) + \alpha\mu \nabla f(x_k)^T p_k.$$

where μ is a positive constant.

(a) Show that the Armijo condition is satisfied for all α sufficiently small if $\mu \in (0, 1)$.

(b) Show that the Armijo condition is violated for all α sufficiently small if $\mu > 1$.

9. Suppose the function

$$f(x_1, x_2) = e^{x_1} x_2^2$$

is sought to be minimized starting from $x^{(0)} = (0, 1)$ over $\mathbb{R}^2$.

Find the steepest descent and Newton search directions at $x^{(0)}$. Is the Newton direction a descent direction? Does the quadratic model used by Newton's method have a minimizer?

10. Consider a quadratic polynomial $q : \mathbb{R}^n \to \mathbb{R}$ of the form

$$q(x) = \frac{1}{2} x^T A x + b^T x + c$$

where $A \succ 0$ and symmetric. Show that the iterate $x_1 = x_0 + p_0$ generated by the pure Newton's method (*i.e.* the step-length $\alpha_0 = 1$) is the unique local minimizer of q for all initial points $x_0 \in \mathbb{R}^n$.

11. Consider the Rosenbrock function

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

which has a unique minimizer at $x_* = (1, 1)$.

(a) Find the gradient $\nabla f(x_1, x_2)$ and the Hessian $\nabla^2 f(x_1, x_2)$ of the Rosenbrock function.

(b) A Matlab implementation `Newton_optimize.m` of Newton's method for the minimization of a smooth function is provided on the course webpage. To perform the Armijo backtracking line search `Newton_optimize.m` calls the routine `linesearch.m` which is also available on the web. Download these m-files and make sure you understand them.

The function `Newton_optimize.m` needs to be supplied with two input arguments,

- an initial point x_0, and
- the name of a Matlab m-file computing the function to be minimized, its derivative and Hessian.

It returns three output arguments, a minimizer, the function value and gradient at this minimizer. For detailed information about input-output arguments, type

```
>> help Newton_optimize
```

On the course webpage an incomplete m-file `Rosenbrock.m` is also made available. You need to fill in the lines below the comments

```
% COMPUTE THE FUNCTION
....
if nargout >= 2
         % COMPUTE THE GRADIENT
         ...
end
```

```
if nargout >= 3
        % COMPUTE THE HESSIAN
        ...
end
```

with Matlab code to compute the function value, gradient and Hessian. `Rosenbrock.m` takes an input x and is expected to return $f(x)$, $\nabla f(x)$ and $\nabla^2 f(x)$ in the variables f, g and H, respectively. When it is called only with two output arguments, `Rosenbrock.m` will not compute the Hessian. Similarly when only the function value is requested, the gradient and Hessian will not be computed.

Now run `Newton_optimize` to find the minimizer of the Rosenbrock function starting from the initial point $x_0 = (0.5, -0.5)$ by typing

```
>> [xmin,fmin,gmin] = Newton_optimize([0.5 -0.5]', 'Rosenbrock');
```

in matlab command window. What is the rate of convergence you observe for $\{\nabla f(x_k)\}$?

The default value used in the line-search for the Armijo condition parameter is $\mu = 0.25$. Run Newton's method by setting $\mu = 0.6$, $\mu = 0.7$ and $\mu = 0.9$. How does this affect the rate of convergence and number of iterations? Explain why the number of iterations is affected this way.

(c) Implement the method of steepest descent by modifying `Newton_optimize.m` and run it on the Rosenbrock function starting from $x_0 = (0.5, -0.5)$. Steepest descent should require considerably more iterations than Newton's method. Include the Matlab output only for the last few iterations. Note that all you need to change is the computation of the search direction.

12. Consider the problem of finding the shortest route between two points, on a two dimensional surface on which it is easier to move quickly on some parts of the surface than on other. You can think of the reason for this as some parts of the surface being more hilly than others, or more "sticky". Mathematically, the problem is to minimize

$$\int_0^1 \rho\left(x(t), y(t)\right) \left\{ \left(\frac{dx(t)}{dt}\right)^2 + \left(\frac{dy(t)}{dt}\right)^2 \right\} dt \tag{1}$$

over smooth functions $x(t)$ and $y(t)$. Here the parametric curve $(x(t), y(t))$ defines the path on the two dimensional surface as a function of time. The boundary conditions are given by $(x(0), y(0)) = (a, b)$ and $(x(1), y(1)) = (c, d)$. (In other words the path starts from point (a, b) at time 0 and ends at (c, d) at time 1. The boundary points $(a, b), (c, d) \in \mathbb{R}^2$ are fixed.) The function $\rho(x, y)$ describes how difficult it is to move at the given point on the surface. (Big values of ρ mean difficult to move; very sticky or high hill, depending on how you want to interpret things.) If $\rho(x, y) = 1$ for all (x, y), the surface is uniformly flat, and the unique solution is the straight line from (a, b) to (c, d). But if ρ takes large values along the straight line from (a, b) to (c, d) and smaller values off it, it may be better to "go around". That's the idea of the problem: what is the shortest route?

The continuous curve $(x(t), y(t))$ can be discretized by replacing the continuous functions $x(t)$, $y(t)$ by the corresponding vectors $X, Y \in \mathbb{R}^{N+2}$ with

- $(X_0, Y_0) = (a, b)$ and $(X_{N+1}, Y_{N+1}) = (c, d)$, and

- $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$ are unknowns. (*i.e.* divide the time interval $[0, 1]$ into $N + 1$ intervals of equal length $\Delta t = \frac{1}{N+1}$

$$\mathcal{I}_1 = [0, t_1], \;\; \mathcal{I}_2 = [t_1, t_2], \;\; \mathcal{I}_3 = [t_2, t_3], \;\; \ldots, \;\; \mathcal{I}_{N+1} = [t_N, 1]$$

so that $X_i \approx x(t_i)$ and $Y_i \approx y(t_i)$.)

The derivatives in equation (1) can be approximated by the finite difference formulas

$$\frac{dx(t_i)}{dt} \approx \frac{X_{i+1} - X_i}{\Delta t} \quad \text{and} \quad \frac{dy(t_i)}{dt} \approx \frac{Y_{i+1} - Y_i}{\Delta t}.$$

Then the integral in (1) can be replaced by the discrete objective function

$$F(X, Y) = \Delta t \sum_{i=0}^{N} \rho\left(X_i, Y_i\right) \left\{ \left(\frac{X_{i+1} - X_i}{\Delta t} \right)^2 + \left(\frac{Y_{i+1} - Y_i}{\Delta t} \right)^2 \right\}.$$

As the number of discrete points $N \to \infty$ or equivalently as $\Delta t \to 0$ the sum above converges to the integral in (1). Your task is to solve the problem

$$\text{minimize}_{X,Y \in \mathbb{R}^N} \quad F(X, Y)$$

(a) Show that if $\rho(x, y) = 1$ for all (x, y), then the solution is the vector of equally spaced points on the line between (a, b) and (c, d). (It may help to first focus on the cases when $N = 1$, $N = 2$, etc.)

(b) Let $\rho(x, y) = 1 + \alpha e^{-\beta(x^2+y^2)}$ where α, β are given constant real numbers. Compute the gradient of F with respect to the $2N$ variables $X_1, Y_1, \ldots, X_N, Y_N$, and write a Matlab function to compute F and its gradient for given X, Y. You will want to collect X and Y together into one vector of size $2N$, say z. You will need to use the chain rule. Check whether your gradient computation is correct by comparing it with the vector v of size $2N$ defined as

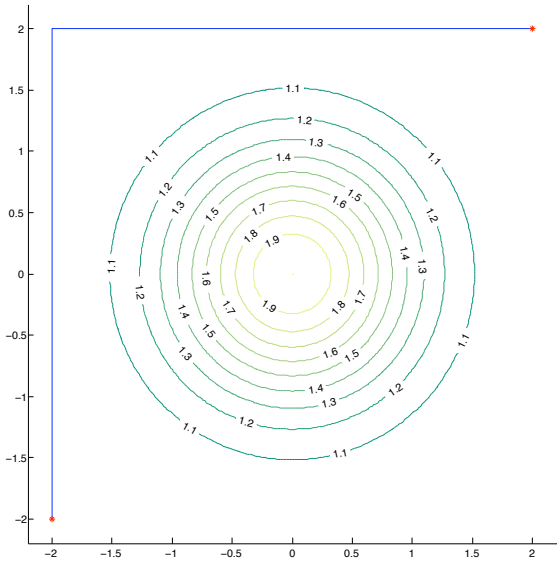
$$v_k = \frac{F(z + he_k) - F(z)}{h}, \qquad k = 1, \ldots, 2N,$$

where $e_k \in \mathbb{R}^{2N}$ is the kth column of the $2n \times 2n$ identity matrix. Choose $h = 10^{-6}$. If the discrepancy between the computed gradient and v is much bigger than h, you have probably made a mistake in your formulas. You need to fix this mistake before continuing.

(c) Use the steepest descent code that you implemented in question **11.(c)** (with the Goldstein-Armijo backtracking line search) to minimize $F(X, Y)$. Choose $N = 10$, $(a, b) = (-2, -2)$ and $(c, d) = (2, 2)$. Experiment with

- $\beta = 1$ and various values of α ranging in $[1, 10]$
- $\alpha = 1$ and various values of β ranging in $[1, 10]$.

You can also try to run your steepest descent code with various boundary conditions and N. Your initial guess z_0 is important. The function $F(X, Y)$ has a local minimizer when all discretization points are aligned to lie on the straight line from (a, b) to (c, d) (this is the case even when α, β are large; can you see how this is possible?). I suggest to start with an initial guess z_0 so that the path starts from (a, b) and goes straight up vertically, then goes to the right horizontally ending at (c, d). The plot of the suggested initial path is illustrated below



on the contour diagram of $\rho(x, y)$ (for $\alpha = \beta = 1$). The asterisks mark the boundary points.

A matlab routine `plotcont.m` is made available on the course webpage to plot a path on the contour diagram of $\rho(x, y)$. Type

```
>> help plotcont
```

to learn about the input parameters for this function. Using `plotcont` plot the optimal paths that you computed on the contour diagrams. If the minimizing path makes no physical sense, you must have made a mistake. It may help to try a smaller value of N, *e.g.* $N = 1$.