EE447: Finding the minima of functions using Powell and Conjugate Gradient

Harishankar Ramachandran
EE Dept, IIT Madras

October 6, 2011

1 Reading Assignment

The minimization chapter in Numerical Recipes, except the section on Linear Programming.

2 The given functions to minimize

The first function to be minimized is

$$f(\vec{x}) = \vec{x}^T M \vec{x}$$

where M is given by

```
array([[ 2.01057304,  1.04351422,  0.11639733,  0.15855415,  0.11737089],
       [ 1.04351422,  1.53763186,  0.93992674,  0.15855415,  0.11737089],
       [ 0.11639733,  0.93992674,  2.26846756,  1.39239668,  0.11737089],
       [ 0.15855415,  0.15855415,  1.39239668,  3.25962827,  1.76056338],
       [ 0.11737089,  0.11737089,  0.11737089,  1.76056338,  2.9342723 ]])
```

Note that the quadratic form is symmetric to make things simpler. It has simple eigenvalues and eigenvectors. Clearly its minimum is at zero, since it is a positive definite quadratic form.

The second function to be minimized is the following:

$$f_2(x,y) = u^2 + v^2 \tag{1}$$

where

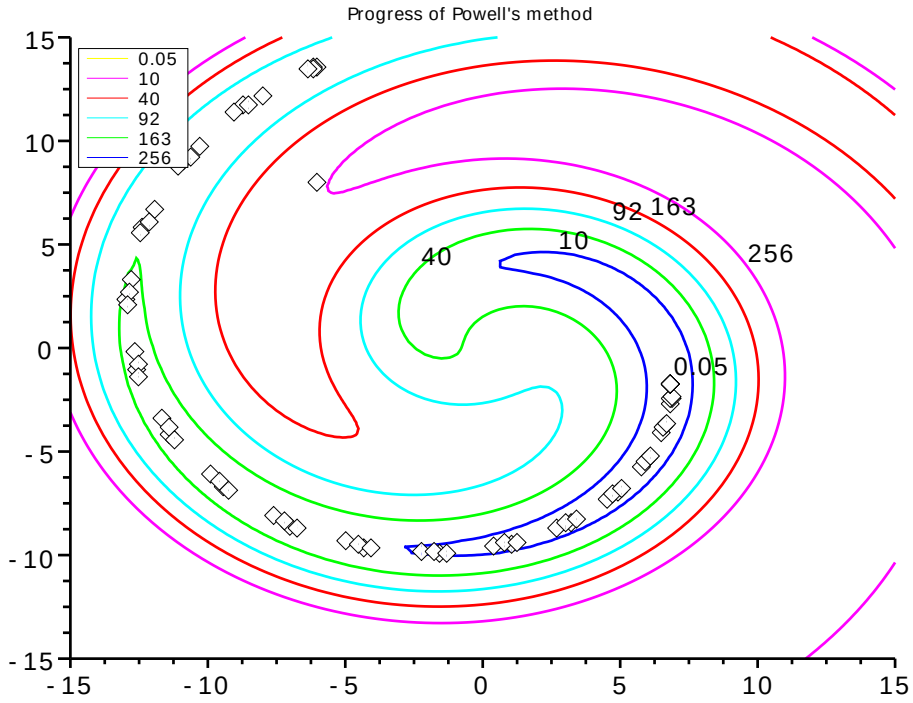
$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} 5 \\ 5 \end{pmatrix} \tag{2}$$

where $\alpha = 5\left(0.1\sqrt{x^2+y^2} - 0.5\right)$.

Note: You may implement these routines either entirely in *Scilab*, as *C* programs run from *Scilab*, or as *C functions* called from *Scilab*. I myself chose the second route (though I worked in *Fortran*).

Note: In parts 5, 6 and 7, the desired graphs should be overlaid on top of the contour plot obtained in part 1. A sample graph is shown below, for the Powell method. You can see how the solver cleverly crawls down the valley. It obtains the conjugate direction then crawls down that direction. Then finds another conjugate direction and crawls down that, etc.

Note: Animation means that having got the intermediate locations reached by the solver, you plot the points with timed delays so that we can see how the solver actually crawls its way to the minimum.



1. Implement both functions in *Python* and plot their contours. Study the structure of the minima. Notice that the first function is ellipsoidal with constant second partial derivitives while the second function has a minimum that has to be reached by crawling down valleys that go round and round.

   Note: For function $f_1(\ldots)$ you can only plot contours of projections. So do this in x, y space, setting the other three coordinates to arbitrary constants.

2. Verify that f_1 has a minimum by confirming that M is positive definite. Use the `eigh()` function of `scipy.linalg` to obtain the eigenvalues and eigenvectors of the matrix. If all the eigenvalues are positive, and M is Hermitian (i.e. symmetric for real matrix), we are guaranteed a minimum at $\vec{x} = 0$.

3. For function f_1, do simple minded minimization, **starting away from zero,** by minimizing along x, then along y and then along z, etc. and repeating till the minimum is reached. Determine why this is not a good method. For what Hessian would it be a good method? (This is a crucial question, since the better methods are based on the answer to this question)

4. For function f_1, study how Powell works. Determine the conjugate directions obtained by Powell. What is their relation to the principal axes of the ellipsoid? To answer this, determine the coordinate system in which the ellipsoid's equation looks like

$$f_1(u,v,w) = \frac{u^2}{a^2} + \frac{v^2}{b^2} + \frac{w^2}{c^2}$$

Note: Use `eigh` in Python which returns the eigen values and the eigen vector matrix for square matrices. You will get

```
[lambda,R]=eigh(M)
```

where `R` is the matrix composed of "right" eigenvectors as columns, and `lambda` is a vector consisting of the eigenvalues. Then,

$$A = R\text{diag}\{\lambda\}R^T$$

(We will learn this later in Linear Algebra) and

$$f_1 = \vec{x}^T A\vec{x} = \vec{x}^T RDR^T\vec{x} = \left(R^T\vec{x}\right)^T D\left(R^T\vec{x}\right) = \vec{z}^T D\vec{z}$$

where $\vec{z}$ is a vector in a rotated space. This gives you the coordinate transformation required to diagonalize the problem:

$$R^T\vec{x} = \vec{z}$$

We will study eigen systems next week, and this is a motivation for that.

5. **The remaining questions are about function** $f_2(x,y,z)$. Last week, you worked with this function and applied the downhill simplex method to it. Generate the contour plot again.

6. Create a solver based on the *Powell* method. The code should accept both a starting value and initial directions as arguments. Save the location of the minimizer's guess at each step and plot its evolution versus iteration, for function f_2, as an animation.

7. Create a solver based on *Conjugate Gradient* methods. The code should accept a starting value as argument. Save the location of the minimizer's guess at each step and plot its evolution versus iteration, for function f_2, as an animation.

In each case, study the effect of varying the initial guess on the method's stability and convergence rate.

Note: One thing I noticed was that you have to be very careful with your function definition. I used floats for my function and found that both the simplex and the gradient methods had trouble. I think the functions have to be computed in double precision, even if the answers are returned as floats.