

Math 224: Scientific Computing I

Homework 9.2

Assigned: Monday, Nov 22, 2004

Due: Friday, Dec 10, 2004

The Last Problem

-1. Reading: C. T. Kelley's book, Trangenstein – Chapter 8, Numerical Recipes – Chapters 9.6, 9.7, 10 (see the course web page for links to all).

0. Overview: Lets take stock of everything we've covered this term in Math 224: an introduction to programming in C/C++, analysis of iterative methods for solving nonlinear equations, numerical linear algebra (solving linear systems and least squares problems, and eigenvalue problems) via direct and iterative methods, and finally systems of nonlinear equations. Each homework set was constructed to give you some hands-on experience in computing each type of problem and evaluating accuracy and computational efficiency. If we all had more time I could pick more problems to illustrate more advanced problems and programming/computational techniques.

In particular, I've focused more on developing our own routines from the ground-up (DIY: do-it-yourself) and understanding how everything works. Another important aspect of scientific computing is making use of routines developed by other people. These may be long and complicated routines that have been carefully designed to be more accurate or efficient than codes we could write easily. These can be a great resource since if you use them, then you don't have to "re-design the wheel". All you need is the ability to find the relevant routines for your problem and to figure out how to incorporate them into your programs. So, for your future reference, some resources that I like for finding relevant codes are:

(a) NETLIB: the official web library for mathematical software

<http://www.netlib.org/liblist.html>

(b) LAPACK: the Linear Algebra PACK – THE universal standard routines for numerical linear algebra (used by MATLAB and most other programs)

<http://www.netlib.org/lapack/index.html>

(c) GAMS: The Guide to Available Mathematical Software

<http://gams.nist.gov/serve.cgi>

(d) Numerical Recipes: good insight and references to more advanced information

<http://www.library.cornell.edu/nr/cbookcpdf.html>

The final problem for the course is suggestive of some of the programming issues that come-up when you link your program to other routines, like the `blackbox()` problems, sometimes in other programming languages, here in FORTRAN.

1. [Linking FORTRAN and C++] Download the programs `fortranc.c` and `fbox.f` from the course web-page. The file `fbox.f` is a routine in FORTRAN to compute

$$f(x, y) = x^2 + y^2 - 4 \quad g(x, y) = y + 3 + e^{2x} + \sin(y)$$

and return the value in an output vector. This file should be compiled using the command:

```
f77 fbox.f -c
```

this creates `fbox.o` – an object file that can be linked to other programs. In this case, `fortranc.c` is a C++ program that treats `fbox()` as a "blackbox" routine. Compile this program using

```
g++ fortranc.c fbox.o -lm
```

Run it and convince yourself you understand how everything works. The next problem makes use of this.

2. [Problem 2, deja vu] Find all the points where the three spheres intersect:

$$(x - 1)^2 + y^2 + (z + 1)^2 = 4, \quad (x + 1)^2 + (y - 2)^2 + z^2 = 9, \quad x^2 + y^2 + (z - 2)^2 = 16.$$

Give the coordinates of the intersection points (accurate to at least 6 digits).

Do this using the following approach:

- (a) Modify `fbox.f` to yield a similar FORTRAN routine to evaluate the system of the three nonlinear equations. Everything else will be done in C++.
- (b) Write a C++ routine to compute the finite-difference approximation of the Jacobian matrix for the system by using `fbox()`:

```
void computeJfd(void (*F)(double *xk,double *Fk),double *xk,double **Jk);
```

Recall Homework 3.1 problems 3b, 4 on finite-difference approximations.
- (c) In your Newton's method loop: DO NOT use `LUfactor/solve` to solve the linear system of equations at each step. Instead, use an appropriate iterative method of your choice. Ideally, you should be able to use one of your routines from previous problem sets without modification :)¹

¹A key goal for this course is to provide you with the skills, understanding, and “your own set of multi-purpose programming tools” for solving problems that come up in your later work – most problems can be solved by using the computational techniques we have developed (alone or in combinations with each other or as building blocks in bigger codes).