MACSI One Day Graduate Course:
Numerical Solution to Differential Equations using Matlab

Part 5: Nonlinear Problems

Niall Madden and Meghan Stephens
Department of Mathematics

National University of Ireland, Galway
Ollscoil na hÉireann, Gaillimh

3rd of April 2007.

Thanks to CJ Budd

NOTE: *This section of the course is adapted (with permission) from notes by Chris Budd, Professor of Applied Mathematics at the University of Bath. The text below is essentially the same those in* `MA50174 Advanced Numerical Methods`

Why bother

Most Problems are Nonlinear!

An important class of nonlinear two point BVPs (called semilinear problems) take the form

$$u'' + b(x)u' + f(x,u) = 0;$$

and it is important to develop techniques to solve them.

Two examples of such problems are given by the differential equation

$$u'' + e^{u/(1+u)} = 0; u(a) = u(b) = 0$$

which models combustion in a chemically reacting material, and

$$u'' + (2/x)u' + k(x)u^5 = 0; u'(0) = 0; u(\infty) = 0; u > 0,$$

which describes the curvature of space by a spherical body.

Why bother

A special case is given by the equation

$$u'' + f(u) = 0; u(a) = \alpha; u(b) = \beta.$$

Discretising the differential equation using finite differences leads to a set of nonlinear equations of the form

$$\delta^2 U + f(U) = 0$$

As with most nonlinear problems, we solve this system by iteration starting with an initial guess although, be warned, the system may have one, none or many solutions.

Newton-Raphson

Perhaps the most effective such method is the *Newton-Raphson* algorithm. Suppose that $U^{(n)}$ is an approximate solution to the differential equation.

Define the *residual* $R^{(n)}$ by

$$AU^{(n)} + f(U^{(n)}) = R^{(n)}.$$

The size of the entries in $R^{(n)}$ is a measure of the quality of an approximate solution.

Newton-Raphson

Now if we define the *Jacobian* of the nonlinear function f by

$$J_{ij} = \partial f_i / \partial U_j,$$

the "linearisation" L of the problem is given by

$$L\psi \equiv A\psi + J'\psi$$

The Newton-Raphson iteration updates $U^{(n)}$ to $U^{(n+1)}$ via the iteration

$$U^{(n+1)} = U^{(n)} - L^{-1}R^{(n)}$$

Here the vector $W \equiv L^{-1}R^{(n)}$ can be found by solving the linear system

$$AW + JW = R^{(n)}$$

This algorithm converges rapidly if the initial guess $U^{(0)}$ is close to the true solution.

Newton-Raphson

Finding such an initial guess can be difficult for a general problem and often requires some *a priori* knowledge of the solution.

The code **Nonlinear.m** implements this method for problem

$$u'' + (1 + e^u + u^4) = 0 \text{ on } (0, 1)$$

and with homogeneous boundary conditions.

It includes two new constructs: **while** and **spdiags**

A While Loop

```
while( norm(R) > TOL)
     Iteration = Iteration+1;
       .
       .
end
```

spdiags

spdiags can be used to create sparse and banded matrices.

We used it in **Nonlinear.m** to create the Jacobian Matrix

$$J_{i,j} = \frac{\partial f_i}{\partial U_j}$$

In our example, $f(U) = 1 + U^2$,

$$J_{i,j} = \begin{cases} 2U_i & i = j \\ 0 & i \neq j. \end{cases}$$

So J is a sparse, diagonal matrix:

```
J = spdiags([0; df(x(2:N)); 0], 0, N+1, N+1);
```