THE UNIVERSITY OF SYDNEY

MATH3076 AND MATH3976 (ADVANCED)

Mathematical Computing: Numerical Methods 2011

Lecturer: *D. J. Ivers*

Assignment 1 — Systems of Nonlinear Equations

Due 5pm Thursday 19^{th} *May.*

Read the course messages for general information on assignments, the form your Assignment 1 file must take and how to submit it. This assignment is worth 4 marks.

1. Attached is a listing of the file `$mc3nm/ass1.inc`, which contains an incomplete main program for finding the real solution of the system of non-linear equations,

$$f_1(x_1, x_2) = x_1 x_2 - x_2^3 - 1$$
$$f_2(x_1, x_2) = x_1^2 x_2 + x_2 - 5$$

near the point (2,3) using the Newton-Raphson method with damping (see (g) below). The equations are taken from Example 5.1 of the lecture notes. In `$mc3nm/ass1.inc` a question mark `?` indicates a missing single character, three horizontal dots `...` indicates a missing part or the missing whole of a line and three vertical dots indicates a missing block of code. **Copy** the file to your directory and complete the program to have the following features:

(a) The integer variable `n` with the `parameter` attribute sets the size n of the problem.

(b) The function **f**, solution **x** and increment **p** are the real one-dimensional arrays `f`, `x`, `p` of length n, and there are additional one-dimensional arrays `x0` and `f0` for the solution and function used in the damping. The Jacobian matrix **J** is a real two-dimensional array `J` of size $n \times n$.

(c) The initial estimates of `X`, the relative error tolerance `tol` and the damping limit `kmax`, are input from the keyboard with prompts.

(d) At each iteration of the damping counter, the iteration counter `i`, the iterates, the corresponding function values and the damping counter `k` are printed. Use an `F18.7` descriptor for all `real` variables. Each column of output must have an appropriate header.

(e) The function **f** and the Jacobian matrix **J** are supplied by the subroutines `fun` and `jac`, internal to the main program `ass1`.

(f) The linear system $\mathbf{Jp} = -\mathbf{f}$ is solved using the Gaussian elimination subroutine `solve` from Exercise Set 3.

(g) The increment **p** is damped so that

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{p}/2^k$$

where $k \geq 0$ is the smallest value, such that

$$\|\mathbf{f}(\mathbf{x}_i + \mathbf{p}/2^{k-1})\|_2 \geq \|\mathbf{f}(\mathbf{x}_i)\|_2 \,, \qquad \|\mathbf{f}(\mathbf{x}_i + \mathbf{p}/2^{k})\|_2 < \|\mathbf{f}(\mathbf{x}_i)\|_2 \,.$$

The damping counter k is limited to $k \leq k_{\max}$, where $k_{\max}$ is given by the `integer` variable `kmax` in the program. If $k_{\max} = 0$, the standard Newton-Raphson method is recovered.

(h) The relative change stopping test for vectors using the norm $\| \ \|_2$ is,

$$\|\mathbf{x}_{i+1} - \mathbf{x}_i\|_2 < \epsilon \|\mathbf{x}_{i+1}\|_2 \,,$$

If this is satisfied, the Newton-Raphson iteration terminates, and the solution and function are printed with appropriate headers. The tolerance ϵ is given by the `real` variable `tol`. Note that the vector norm $\|\mathbf{x}\|_2$ is given in Fortran 90 by `sqrt(sum(x*x))`.

(i) A message is printed if there is no convergence of the (Newton-Raphson) i-loop in 24 iterations.

As a check on your program, compare your answers with those given in the notes.

2. Modify `fun` and `jac` in your program to find the solution of the equations,

$$\begin{aligned} x_1^2 - x_2 - 2.9 \ln |x_2| &= 0 \\ 2x_2^2 - x_1 x_2 - 5x_2 + 1.1 &= 0 \,, \end{aligned}$$

near (2,2).

(a) Take (2,2) as the starting approximation, a relative error tolerance $\epsilon = 5 \times 10^{-6}$ and use the two damping limits,

(i) $k_{\max} = 0$ (ii) $k_{\max} = 10$.

(b) Take (2.6,2) as the starting approximation, a relative error tolerance $\epsilon = 5 \times 10^{-6}$ and use the three damping limits,

(i) $k_{\max} = 0$ (ii) $k_{\max} = 3$

(iii) $k_{\max} = 10$.

(c) Find the other zero using a relative error tolerance $\epsilon = 5 \times 10^{-6}$.

```
program ass1
integer, parameter :: n=?
????, dimension(n) :: f,...
????, dimension(?,?) :: ?
??????? :: singular
print*,'Input starting approximation (x):'
read*,?
print*,'Input relative error tolerance (tol):'
...
print*,'Input damping limit (kmax):'
...

print*,' i          x1             ??             f1             ??     k'
???? fun

do i=1,??  ! Newton-Raphson iteration loop.
???? jac
call solve(?,-f,?,?,singular)
x0=x
f0=f
if (singular) then
   print*,'Singular matrix'
else
   do k=0,kmax  ! Damping loop.
   x = x0 + p/...
   call fun
   write(*,10) i,?,?,k
10 format(I3,??????,I3)
   if (sum(f*f)<sum(f0*f0)) exit  ! Damping criterion.
   enddo

   if (sqrt(sum((x-x0)*(x-x0)))<...) then
   print*, 'Solution:'
   print*,'x = ',?
   print*,'f = ',?
   stop
   endif
endif

enddo

print*,'No convergence ...


stop
????????
!-----------------------------------------------
subroutine fun

f(1) = ...
...

end ...
```

```
!-----------------------------------------------
subroutine jac
J(1,1) = ...
:::

...
!-----------------------------------------------
??? program ????
!===============================================
subroutine solve(A,b,x,n,singular)
:
```