

# MATH 335: Numerical Analysis

## Problem Set 10, Solutions

3.8.2 We have

$$J_F(x, y) = \begin{bmatrix} 3x^2 & -2y + 1 \\ y + 2x & x \end{bmatrix} \Rightarrow J_F(-1, 4) = \begin{bmatrix} 3 & -7 \\ 2 & -1 \end{bmatrix}.$$

3.8.5 For this problem we have

$$\left. \begin{array}{l} F_1(x, y, z) = 9 - x + \sin(x + y) \\ F_2(x, y, z) = 8y - \cos^2(z - y) - 1 \\ F_3(x, y, z) = 12z + \sin z - 1 \end{array} \right\} \Rightarrow J = \begin{bmatrix} \cos(x + y) - 1 & \cos(x + y) & 0 \\ 0 & 8 - \sin(2(z - y)) & \sin(2(z - y)) \\ 0 & 0 & 12 + \cos z \end{bmatrix}.$$

Employing the book-supplied program `newton_sys.m` program, we have

```
octave:103> function y = F(x)
> y(1) = 9 - x(1) + sin(x(1)+x(2));
> y(2) = 8*x(2) - (cos(x(3) - x(2)))^2 - 1;
> y(3) = 12*x(3) + sin(x(3)) - 1;
> y = y';      % Use to make y the correct shape
> end

octave:104> function J = JF(x)
> J = zeros(3);
> J(1,1) = cos(x(1) + x(2)) - 1;
> J(1,2) = cos(x(1) + x(2));
> J(2,2) = 8 - sin(2*(x(3) - x(2)));
> J(2,3) = sin(2*(x(3) - x(2)));
> J(3,3) = 12 + cos(x(3));
> end

octave:106> newton_sys(@F, @JF, [0.1; 0.25; 0.08], 10^(-6), 100)
x0 =
    0.100000
    0.250000
    0.080000
```

Newton's method converges after 7 iterations to  
x =

9.089109  
0.246442  
0.076929

So, the approximate solution is (9.089109, 0.246442, 0.076929).

3.8.6 Here

$$J(x, y) = \begin{bmatrix} 2x & 2ay \\ 2(x-1) & 2y \end{bmatrix} \Rightarrow J^{-1}(x, y) = \frac{1}{4xy - 4ay(x-1)} \begin{bmatrix} 2y & -2ay \\ 2(1-x) & 2x \end{bmatrix}.$$

Thus, it seems the first expression desired is

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} - \frac{1}{4x_n y_n - 4a y_n (x_n - 1)} \begin{bmatrix} 2y_n & -2a y_n \\ 2(1 - x_n) & 2x_n \end{bmatrix} \begin{bmatrix} x_n^2 + a y_n^2 - 1 \\ (x_n - 1)^2 + y_n^2 - 1 \end{bmatrix}.$$

Expanding and simplifying, we get

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \frac{1}{4x_n y_n - 4a y_n (x_n - 1)} \begin{bmatrix} 2y_n + 2x_n^2 y_n (1 - a) \\ 2a y_n^2 + 2x_n^2 - 2x_n + 2 + 2x_n y_n^2 (1 - a) \end{bmatrix}.$$

I'm not sure my routine is as short as it could be, but here it is:

```
function soln = specNewton (x0, a, tol, maxIt)
    prev_x = x0;
    x = [2*x0(2)+2*x0(1)^2*x0(2)*(1-a); ...
         2*(a*x0(2)^2+x0(1)^2-x0(1)+1+x0(1)*x0(2)^2*(1-a))] ...
         / (4*x0(2)*(x0(1)-a*(x0(1)-1)));
    k = 1;
    while ((norm(prev_x - x) >= tol) & (k < maxIt))
        k++;
        prev_x = x;
        x = [2*x(2)+2*x(1)^2*x(2)*(1-a); ...
             2*(a*x(2)^2+x(1)^2-x(1)+1+x(1)*x(2)^2*(1-a))] ...
             / (4*x(2)*(x(1)-a*(x(1)-1)));
    end
    soln = x;
    if (k >= maxIt)
        disp('Failed to converge to a solution in specified number of iterations')
    end
end
```

Trying it for  $a = 1$  with initial guess  $\mathbf{x}^{(0)} = (1, 1)$ , and  $\text{tol} = 10^{-6}$ , I get solution  $(0.50000, 0.86603)$ .

★16 By definition,

$$g(t) = \begin{bmatrix} F_1(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) \\ \vdots \\ F_n(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) \end{bmatrix} = \begin{bmatrix} F_1((x_1, \dots, x_n) + t(y_1 - x_1, \dots, y_n - x_n)) \\ \vdots \\ F_n((x_1, \dots, x_n) + t(y_1 - x_1, \dots, y_n - x_n)) \end{bmatrix}.$$

Thus,

$$g'(t) = \begin{bmatrix} (d/dt) F_1((x_1, \dots, x_n) + t(y_1 - x_1, \dots, y_n - x_n)) \\ \vdots \\ (d/dt) F_n((x_1, \dots, x_n) + t(y_1 - x_1, \dots, y_n - x_n)) \end{bmatrix}.$$

In preparation for the chain rule, let us note that we need a notation for the partial derivative of  $F_i$  with respect to its  $j$ th argument. Some people write  $D_j F_i$  for this, but in the problem statement it has been written as  $\partial F_i / \partial x_j$ . So, for each  $1 \leq i \leq n$ ,

$$\begin{aligned} & \frac{d}{dt} F_i((x_1, \dots, x_n) + t(y_1 - x_1, \dots, y_n - x_n)) \\ &= \frac{d}{dt} F_i(x_1 + t(y_1 - x_1), \dots, x_n + t(y_n - x_n)) \\ &= \frac{\partial F_i}{\partial x_1}(x_1 + t(y_1 - x_1), \dots, x_n + t(y_n - x_n)) \cdot (y_1 - x_1) + \cdots \\ & \quad + \frac{\partial F_i}{\partial x_n}(x_1 + t(y_1 - x_1), \dots, x_n + t(y_n - x_n)) \cdot (y_n - x_n) \\ &= \sum_{j=1}^n \frac{\partial F_i}{\partial x_j}(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) \cdot (y_j - x_j) \\ &= \begin{bmatrix} \frac{\partial F_i}{\partial x_1}(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) & \frac{\partial F_i}{\partial x_2}(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) & \cdots & \frac{\partial F_i}{\partial x_n}(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) \end{bmatrix} \begin{bmatrix} y_1 - x_1 \\ y_2 - x_2 \\ \vdots \\ y_n - x_n \end{bmatrix} \end{aligned}$$

Thus, putting together the various rows of  $g'$ , we have

$$\begin{aligned} g'(t) &= \begin{bmatrix} (d/dt) F_1((x_1, \dots, x_n) + t(y_1 - x_1, \dots, y_n - x_n)) \\ \vdots \\ (d/dt) F_n((x_1, \dots, x_n) + t(y_1 - x_1, \dots, y_n - x_n)) \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial F_1}{\partial x_1}(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) & \cdots & \frac{\partial F_1}{\partial x_n}(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) \\ \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial x_1}(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) & \cdots & \frac{\partial F_n}{\partial x_n}(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) \end{bmatrix} \begin{bmatrix} y_1 - x_1 \\ \vdots \\ y_n - x_n \end{bmatrix} \\ &= \mathbf{J}(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))(\mathbf{y} - \mathbf{x}). \end{aligned}$$