Step **1** of 3

To perform multivariable Newton's Method, create a Matlab M-file titled "multnewton3.m" that contains the following code:

```
function [s J] = multnewton3(f,p0,tol,MaxIter)

%function takes in multivarible inline function f

%initial vector p0, tolerance, and max number of

%iterations.

%function outputs solution vector

format long

x = sym('x'); y = sym('y'); z = sym('z');

F = f([x,y,z]);

% Compute the Jacobian matrix symbolically

J = jacobian(F);

invJ = inv(J);

s = zeros(MaxIter,3);

s(1,:) = p0;

dsnorm = inf;

iter = 1;

while dsnorm>tol && iter

ds = -subs(invJ,[x y z],s(iter,:))*f(s(iter,:));

s(iter+1,:) = s(iter,:) + ds';

dsnorm = norm(s(iter+1,:)-s(iter,:),2);

iter = iter+1;

end

%s = s(1:iter,:);

s = s(iter,:);

end
```

a) Given the system

$$f = x - 3y - z^2 = -3$$
$$g = 2x^3 + y - 5z^2 = -2$$
$$h = 4x^2 + y + z = 7$$

,

the Jacobian matrix used in Newton's method is

$$J(x,y,z) = \begin{bmatrix} f_x & f_y & f_z \\ g_x & g_y & g_z \\ h_x & h_y & h_z \end{bmatrix}$$

$$= \boxed{\begin{bmatrix} 1 & -3 & -2z \\ 6x^2 & 1 & -10z \\ 8x & 1 & 1 \end{bmatrix}}$$

Step 3

0

b) For stating vector $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$, the Jacobian is

$$J(x,y,z) = \begin{bmatrix} 1 & -3 & -2z \\ 6x^2 & 1 & -10z \\ 8x & 1 & 1 \end{bmatrix}$$

$$J(1,1,1) = \boxed{\begin{bmatrix} 1 & -3 & -2 \\ 6 & 1 & -10 \\ 8 & 1 & 1 \end{bmatrix}}$$

Step 4

0

c) Using the multivariable Newton's method, give the following Matlab commands to find a root with initial vector $[1 \quad 1 \quad 1]$:

Step 5

0

INPUT:

```
>> f=inline('[p(:,1)-3*p(:,2)-p(:,3).^2+3; 2*p(:,1).^3-
p(:,2)-5*p(:,3).^2+2;4*p(:,1).^2+p(:,2)+p(:,3)-7]','p');
>> s = multnewton3(f,[1 1 1],10^-8,10)
```

OUTPUT:

s =

1.119627579034836 1.127934568027381 0.857801769030988

Step 6

0

Using the multivariable Newton's method, give the following Matlab commands to find a root with initial vector $[1.3 \quad 0.9 \quad -1.2]$:

Step 7

0

p(:,2)-5*p(:,3).^2+2;4*p(:,1).^2+p(:,2)+p(:,3)-7]';'p');
>> s = multnewton3(f,[1 1 1],10^-8,10)
OUTPUT:
s =
1.119627579034836 1.127934568027381 0.857801769030988

Step 6

0

Using the multivariable Newton's method, give the following Matlab commands to find a root with initial vector $[1.3 \quad 0.9 \quad -1.2]$:

Step 7

0

INPUT:
>> s = multnewton3(f,[1.3 0.9 -1.2],10^-8,10)
OUTPUT:
s =
1.321147341773673 1.070985393583947 -1.052706588286513
Note: Approximated solutions may not match solutions found by built in root-finding algorithms. The reason for this is that the code implementing the multivariable newton's method is not optimized for the numerical matrix algorithms, which requires techniques from numerical linear algebra not yet introduced by the text.