```
% MultivariateNewtonP1.m
% Called by DriveP1.m

function x0 = MultivariateNewtonP1(x,y,z,d,A,B,C,t,c,steps)

format long;

x0 = [x;y;z;d];

for i=1:steps

    x = x0(1);
    y = x0(2);
    z = x0(3);
    d = x0(4);

    % Four (4) Satellite Orbit Equations (Spherical:
    %                        with GPS Receiver Clock Inaccuracy Corrections
    f1 = (x-A(1))^2 + (y-B(1))^2 +(z-C(1))^2 - (c*(t(1)-d))^2;
    f2 = (x-A(2))^2 + (y-B(2))^2 +(z-C(2))^2 - (c*(t(2)-d))^2;
    f3 = (x-A(3))^2 + (y-B(3))^2 +(z-C(3))^2 - (c*(t(3)-d))^2;
    f4 = (x-A(4))^2 + (y-B(4))^2 +(z-C(4))^2 - (c*(t(4)-d))^2;

    % Prep to populate Jacobian Matrix (Partial Derivatives)
    pf1px = 2*(x-A(1));
    pf1py = 2*(y-B(1));
    pf1pz = 2*(z-C(1));
    pf1pd = 2*(c^2)*(t(1)-d);

    pf2px = 2*(x-A(2));
    pf2py = 2*(y-B(2));
    pf2pz = 2*(z-C(2));
    pf2pd = 2*(c^2)*(t(2)-d);

    pf3px = 2*(x-A(3));
    pf3py = 2*(y-B(3));
    pf3pz = 2*(z-C(3));
    pf3pd = 2*(c^2)*(t(3)-d);

    pf4px = 2*(x-A(4));
    pf4py = 2*(y-B(4));
    pf4pz = 2*(z-C(4));
    pf4pd = 2*(c^2)*(t(4)-d);

    % Jacobian Matrix
    DF = [pf1px pf1py pf1pz pf1pd;
          pf2px pf2py pf2pz pf2pd;
          pf3px pf3py pf3pz pf3pd;
          pf4px pf4py pf4pz pf4pd];
```

```
    f = [f1;
         f2;
         f3;
         f4];

        v = -DF\f;        % Solving 4 Eqns, 4 Unknows:  Matlab "backslash"
        x0 = x0 + v;
        x0'
end
```