An Assignment for Math. 128 B due Mon. 5 Feb. 2007 :

The task is to compare two ways to solve a vector equation $\boldsymbol{f}(\mathbf{z}) = \mathbf{o}$ for its vector solution(s) $\mathbf{z}$, given a MATLAB program that computes $\boldsymbol{f}(\mathbf{x})$. The two ways are …

- Newton's Iteration $\mathbf{x}_{k+1} := \mathbf{x}_k - \boldsymbol{f}'(\mathbf{x}_k)^{-1}\cdot\boldsymbol{f}(\mathbf{x}_k)$ starting from some initial guess(es) $\mathbf{x}_0$; here $\boldsymbol{f}'(x) := \partial\boldsymbol{f}(\mathbf{x})/\partial\mathbf{x}$ is the *Jacobian* matrix of first partial derivatives.

- Solve the differential equation $d\mathbf{x}(\tau)/d\tau = -\boldsymbol{f}'(\mathbf{x}(\tau))^{-1}\cdot\boldsymbol{f}(\mathbf{x}(\tau))$ numerically starting from some initial guess(es) $\mathbf{x}(0)$ and running τ from 0 up to a sufficiently big positive number T that $\boldsymbol{f}(\mathbf{x}(T))$ is negligible. You may use MATLAB's ODE-solvers.

What evidence, if any, have you garnered to persuade you (and someone who dislikes you) that you have computed *all* the solutions $\mathbf{z}$?

Here is the MATLAB program given to define $f(x)$:

```
function  y = f(v)
%  y = f(v)  takes a column  3-vector  v  and returns the column
%  y = [v'*M*v + 2*m'*v + mu ;  v'*A*v + 2*a'*v + alpha ;  v'*T*v + 2*t'*v + theta]
%  for coefficients that are filled in here:
M = [ 0  0  0 ;  0  1  1 ;  0  1  2 ] ;
m = [ 0 ;  2 ;  6 ] ;
mu = 18 ;
A = [ 1  0  -1 ;  0  1  1 ;  -1  1  3 ] ;
a = [ -1 ;  2 ;  7 ] ;
alpha = 17 ;
T = [ 1  0  -1 ;  0  1  1 ;  -1  1  2 ] ;
t = [ -1 ;  2 ;  3 ] ;
theta = 2 ;
%
y = [v'*M*v + 2*m'*v + mu ;  v'*A*v + 2*a'*v + alpha ;  v'*T*v + 2*t'*v + theta] ;
```

You may incorporate the foregoing statements into your own program(s), which need not call the given program f(…) except to check that an alleged solution $\mathbf{z}$ makes $\boldsymbol{f}(\mathbf{z})$ negligible.

Repeat the assignment with a function $\mathbf{g}(\mathbf{x})$ in place of $\boldsymbol{f}(\mathbf{x})$ and differing from it only in that

[mu, alpha, theta] = [19 16 1] .

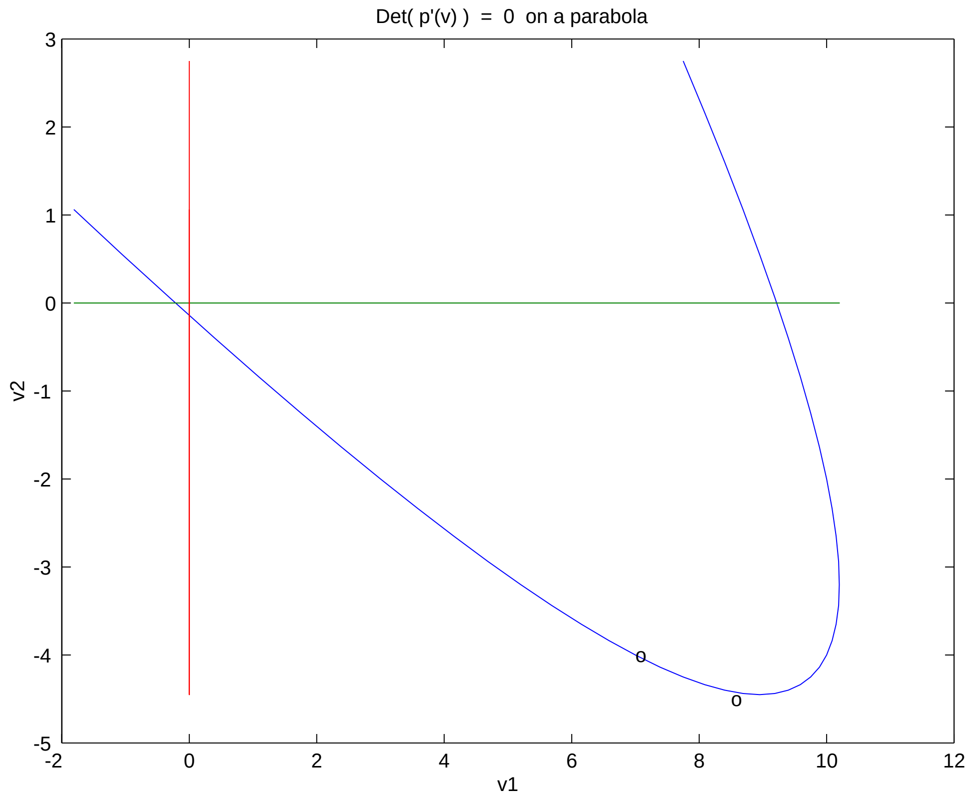
What follows are examples of MATLAB programs written to illustrate how well the foregoing ways solve a simpler equation $\mathbf{p}(\mathbf{z}) = \mathbf{o}$, and to illustrate how these two numerical ways may malfunction when $\det(\mathbf{p}'(\mathbf{x}))$ vanishes at or too near points $\mathbf{x} = \mathbf{x}_k$ or $\mathbf{x} = \mathbf{x}(\tau)$ encountered during the numerical process. In fact, $\det(\mathbf{p}'(\mathbf{x})) = 0$ on a parabola plotted below.

```
function  y = p(v)
%  y = p(v)  takes a column  2-vector  v  and computes the column
%  p(v) = a + B*V + C.v*v/2 = a + (B + 0.5*[v'*C(:,:,1); v'*C(:,:,2)])*v
%  for coefficients arrays  a, B, C  filled in below.
%
C = cat(3, [1, 2; 2, 3], [2, 3; 3, 4]) ;  % ... C  is a bilinear operator
B = [1, -2; 0, 2] ;  a = [-7.5; 11] ;
%
Cv = [v'*C(:,:,1); v'*C(:,:,2)] ;  % ... C.v*u  is a bilinear operation
y = a + (B + 0.5*Cv)*v ;


function  y = dnewtp(v, w)
%  y = dnewtp(v)  takes a column  2-vector  v  and computes the column
%  qp(v) = a + B*V + C.v*v/2 = a + (B + 0.5*[v'*C(:,:,1); v'*C(:,:,2)])*v
%  and its derivative  p1 = dp/dv ,  and returns the  Newton step  y = -p1\p
%  for coefficients arrays  a, B, C  filled in below.  If  abs(p(v))  is no
%  bigger than its roundoff bound,  a global variable  IsntNegligible  is
%  decreased by  1  and if it is negative then  y  is replaced by  [0; 0] .
%  y = dnewtp(v, w)  sets  v = w  for use as an  ODEfile  dnewtp(t, v) .
%
C = cat(3, [1, 2; 2, 3], [2, 3; 3, 4]) ;  % ... C  is a bilinear operator
B = [1, -2; 0, 2] ;  a = [-7.5; 11] ;
%
global  IsntNegligible
if (nargin > 1),  v = w ;  end
Cv = [v'*C(:,:,1); v'*C(:,:,2)] ;  % ... C.v*u  is a bilinear operation
p1 = B + Cv ;  %... = dp/dv
p = a + (B + 0.5*Cv)*v ;
%  Compute a rough error-bound for roundoff in  p :
av = abs(v) ;  aC = abs(C) ;
aCv = [av'*aC(:,:,1); av'*aC(:,:,2)] ;
ep = (abs(a) + (abs(B) + 0.5*aCv)*av)*eps ;  %...  rough error-bound
y = (abs(p) > ep) ;  %...  compare  p  with its rough roundoff bound
if ~any(y(:))  % ...  p  is  (nearly)  negligible
    IsntNegligible = IsntNegligible - 1 ;
    if (IsntNegligible < 0),  return,  end,  end  %...  y = [0; 0]
y = -(p1\p) ;  %...  the  Newton  step ...
y = y - p1\( [p1, p]*[y;1] ) ;  %... iteratively refined to reduce roundoff


function  [i, v] = iterp(v)
%  [i, v] = iterp(v)  counts iterations  v = dnewtp(v)  until
%  it converges,  if it ever does,  to a zero  v  of  p(v)
%  up to a maximum of,  say,  100  iterations.  Meanwhile it
%  displays each iterate's  v' and the final  residual = p(v)' .
global IsntNegligible
IsntNegligible = 3 ;
i = 0 ;
while  (i < 100)&IsntNegligible
    v = v + dnewtp(v) ;  i = i+1 ;  V = v' ,  end
residual = p(v)'


function  v = odep45(vo)
%  v = odep45(vo)  solves  dv/dt = dnewtp(t, v)  for a
%  column 2-vector  v(t)  starting from  v(0) = vo  and
%  ending at  v = v(20) .  The trajectory of  v(t)  is plotted.
global IsntNegligible
IsntNegligible = 2 ;
options = odeset( 'OutputFcn', 'odephas2' ) ;
[T, V] = ode45('dnewtp', [0, 20], vo, options) ;
v = V(length(T),:)' ;
```



Try a variety of initial guesses vo , like vo = [0; 0] , and then see what happens when programs [i, v] = iterp(vo) and v = odep45(vo) are run.