```
% ------------ Function MY_NYSTROM.M ----------------------
% This function implements Nystrom's 3 step implicit
algorithm for solving
%  ODE-IVP over interval [tn, tn+h]
%  Note: Variable xn in this program is a matrix that stores
vectors x(n), x(n-1)
%  and x(n-2).
%  Similarly, Fn is a matrix that contains derivative vectors
F(n), F(n-1) and F(n-2).
% fun_name: Name of function that returns derivatives
vector F(x,t) given x and t
% h : integration step size
% eps:  tolerence
% maxiter: maximum number of iterations
% ------------------------------------------------------------

function [xnplus1, converged ] = my_nystrom( fun_name, xn,
tn, h, eps, maxitr )

% To generate initial guess for x(n+1), 3 step explicit
algorithm is used

coeff1 = [ 7/3 -2/3 1/3]';
Fn(:,1) = feval( fun_name, tn, xn(:,1) ) ;
Fn(:,2) = feval( fun_name, tn, xn(:,2) ) ;
Fn(:,3) = feval( fun_name, tn, xn(:,3) ) ;
xt0 = xn(:,1) + h * ( coeff1(1) * Fn(:,1) + coeff1(2) * Fn(:,2) +
coeff1(3) * Fn(:,3)) ;

% Note: The last line can be replaced by following compact
```

```
%  form :
%   xt0 = xn(:,1) + h * Fn* coeff1 ;

% Iterations for implicit algorithm

coeff2 = [ 1/3 4/3 -1/3]';  % Coefficients of implicit algorithm
xinit  = xt0 ;
delta = 100 ; itr = 0 ;

while ( (delta > eps) && (itr < maxitr ))
   Fn1 = feval( fun_name, tn, xinit ) ;
   xnew = xn(:,1) + h * [Fn1 Fn(:,1:2)] * coeff2 ;
   delta = norm( xnew - xinit ) / norm(xnew) ;
   xinit = xnew ;
   itr = itr + 1 ;
end

if (itr < maxitr )
   converged = 1 ;
else
   converged = 0 ;
end

% create solution matrix by using 'xnew' and part of 'xn'

xnplus1 = [ xnew xn(:,1:2) ]  ;
```