All code generated with Matlab® Software

```
% approx_Jacobian_FD.m
%
% function [Jac,iflag] = approx_Jacobian_FD(x,Options,Param);
%
% This MATLAB m-file contains a function that uses finite
% differences to approximate a Jacobian using finite differences.
% The input to the routine is :
%
% x - a column vector of the N unknown variables
% f - the column vector of the function values
% calc_f - the name of the routine that evaluates the function
% Options - a data structure containing the following parameters
%     .epsilon = if non-zero, user-specified offset for each state variable
%                used in the finite difference method.
%     .use_sparse = if non-zero, use sparse matrix format for Jacobian
%     .S = a matrix containing non-zero elements only at those positions
%          for which the Jacobian is sparse
% Param - a data structure containing system parameters to be passed
%         to the function evaluator.
%
% K. Beers. MIT ChE. 10/18/2001


function [Jac,iflag] = approx_Jacobian_FD(x,f,calc_f,Options,Param);


% signify sucessful completion not yet attained.
iflag = 0;

% extract the number of state variables
Nvar = length(x);


% Allocate space for the Jacobian in memory based on whether one
% uses the full or sparse matrix format.

if(Options.use_sparse)

   % extract the number of non-zero elements from S.
   nz_Jac = nnz(Options.S);

   % allocate space for Jac using the sparse matrix format
   Jac = spalloc(Nvar,Nvar,nz_Jac);

else   % use full matrix format

   Jac = zeros(Nvar,Nvar);

end
```

```
% We now set the offset used in finite differences for each state variable.

if(Options.epsilon==0)
   epsilon = sqrt(eps);
   epsilon_is_vector = 0;

elseif (length(Options.epsilon)==1)
   epsilon = Options.epsilon;
   epsilon_is_vector = 0;

else
   epsilon = Options.epsilon;
   epsilon_is_vector = 1;

end


% Begin iterations over each state variable to estimate corresponding
% elements of the Jacobian by finite differences.


for ivar = 1:Nvar

   % Get magnitude of offfset of unknown ivar
   if(epsilon_is_vector)
      delta_ivar = epsilon(ivar);
   else
      delta_ivar = epsilon;
   end

   % Get offset state vector.
   x_offset = x;
   x_offset(ivar) = x_offset(ivar) + delta_ivar;

   % Calculate function vector for offset state vector.
   f_offset = feval(calc_f,x_offset,Param);

   % Calculate the Jacobian elements in column ivar.
   if(Options.use_sparse==0)
      Jac(:,ivar) = (f_offset - f)/delta_ivar;
   else
      list_nz = find(Options.S(:,ivar));
      for count=1:length(list_nz)

         % Get row number of non-zero element.
         k = list_nz(count);

         % calculate Jacobian element at this position
         Jac(k,ivar) = (f_offset(k) - f(k))/delta_ivar;

      end
```

```
    end
end
iflag = 1;
return;
```