TR_1D_model1_SS\calc_epsilon.m

```
% TR_1D_model1_SS\calc_epsilon.m
%
% function [epsilon,iflag] = calc_epsilon( ...
%    num_pts,imask_int,num_fields);
%
% This procedure takes as input the number of fields and an
% integer mask that is non-zero only at the interior points,
% and returns the epsilon vector that identifies in the full
% DAE set which are the ordinary differential equations
% (epsilon(k)=1) and which are the algebraic equations
% (epsilon(k)=0) arising from the boundary conditions.
%
% INPUT :
% =======
% num_pts              INT
%                       the number of grid points
% imask_int            INT(num_pts)
%                       This is an integer mask that is 1 for
%                       every interior point and is 0 for the
%                       boundary points.
% num_fields            INT
%                       This is the number of fields that are
%                       discretized in the set of PDE's.
%
% OUTPUT :
% ========
% epsilon              INT(num_DOF=num_fields*num_pts)
%                          this is a 1-D array of integers of the same size
%                          as x_state.  It contains a 1 at position k for
%                          very equation that is an ordinary differential
%                          equation, and a 0 for every algebraic equation,
%                          which in this problem arise from the boundary
%                          conditions
%
% Kenneth Beers
% Massachusetts Institute of Technology
% Department of Chemical Engineering
% kbeers@mit.edu
% 7/2/2001
%
% Version as of 7/21/2001


function [epsilon,iflag] = calc_epsilon( ...
   num_pts,imask_int,num_fields);

iflag = 0;
```

```
func_name= 'calc_epsilon';


% This flag controls what action to take in case of
% an assertion failure.  See the assertion routines
% for further details.
i_error = 2;


% First, check the input data.

% num_pts
check_real=1; check_sign=1; check_int=1;
assert_scalar(i_error,num_pts,'num_pts', ...
   func_name,check_real,check_sign,check_int);

% num_fields
check_real=1; check_sign=1; check_int=1;
assert_scalar(i_error,num_fields,'num_fields', ...
   func_name,check_real,check_sign,check_int);

% check to make sure imask_int is vector of
% proper length
dim=num_pts; check_column=0;
check_real=1; check_sign=2; check_int=1;
assert_vector(i_error,imask_int,'imask_int', ...
   func_name,dim,check_real,check_sign, ...
   check_int,check_column);


%PDL> Initialize epsilon to all zeros

% calculate total number of degrees of freedom of
% DAE sytem

num_DOF = num_fields*num_pts;

epsilon = linspace(0,0,num_DOF)';


%PDL> FOR ifield FROM 1 TO num_fields

for ifield = 1:num_fields


%PDL> pos_offset = (ifield-1)*num_pts
% set integer offset to the beginning of
% the current field in the master array

   pos_offset = (ifield-1)*num_pts;
```

```
%PDL> Copy integer mask of interior points to proper
%   locations in epsilon for the current field :
%   epsilon(pos_offset+1:pos_offset+num_pts) = imask

   epsilon(pos_offset+1:pos_offset+num_pts)= imask_int;


%PDL> ENDFOR

end


iflag = 1;

return;
```