TR_1D_model1_SS\stack_state.m

```
% TR_1D_model1_SS\stack_state.m
%
% function [x_state,iflag] = stack_state(State, ...
%    num_species,num_pts);
%
% This procedure stacks the concentration and temperature
% profiles into a single master array for use with the
% general solver.
%
% INPUT :
% =======
% State                 see TR_1D_model1_SS.m for details
%                         data structure containing concentration
%                         and temperature profiles
% num_species               INT
%                          the number of species in the system
% num_pts                  INT
%                          the number of grid points
%
% OUTPUT :
% ========
% x_state                   REAL(num_DOF= (num_species+1)*num_pts)
%                          this is the 1-D master array of state
%                          variables
%
% Kenneth Beers
% Massachusetts Institute of Technology
% Department of Chemical Engineering
% 7/2/2001
%
% Version as of 7/25/2001



function [x_state,iflag] = stack_state(State, ...
   num_species,num_pts);

iflag = 0;

func_name = 'stack_state';


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

```
% check for errors in input parameters


% check dimensions
assert_scalar(1,num_species,'num_species', ...
   func_name,1,1,1);
assert_scalar(1,num_pts,'num_pts',func_name,1,1,1);


% check if State is proper structure type
StateType.num_fields = 2;
% .conc
ifield = 1;
FieldType.name = 'conc';
FieldType.is_numeric = 1;
FieldType.num_rows = num_pts;
FieldType.num_columns = num_species;
FieldType.check_real = 1;
FieldType.check_sign = 2;
FieldType.check_int = 0;
StateType.field(ifield) = FieldType;
% .Temp
ifield = 2;
FieldType.name = 'Temp';
FieldType.is_numeric = 1;
FieldType.num_rows = num_pts;
FieldType.num_columns = 1;
FieldType.check_real = 1;
FieldType.check_sign = 1;
FieldType.check_int = 0;
StateType.field(ifield) = FieldType;
% call assertion routine for structure
assert_structure(i_error,State,'State', ...
   func_name,StateType);


% allocate x_state column vector and
% initialize to zeros
num_DOF = (num_species+1)*num_pts;
x_state = linspace(0,0,num_DOF)';


%PDL> First, we stack the concentration profiles

%PDL> Set pos_counter to zero

pos_counter = 0;


%PDL> FOR ispecies FROM 1 TO ProbDim.num_species
```

```
for ispecies = 1:num_species


%   PDL> x_state(pos_counter+1:pos_counter+num_pts) =
%        state_data:conc(:,ispecies)

    x_state(pos_counter+1:pos_counter+num_pts) = ...
       State.conc(:,ispecies);


%   PDL> Increment pos_counter by num_pts

    pos_counter = pos_counter + num_pts;


%PDL> ENDFOR

end


%PDL> Next, we stack the temperature profile

%PDL> x_state(pos_counter+1:pos_counter+num_pts) =
%       state_data:Temp

x_state(pos_counter+1:pos_counter+num_pts) = ...
    State.Temp;


iflag = 1;

return;
```