TR_1D_model1_SS\read_solver_input.m

```
% TR_1D_model1_SS\read_solver_input.m
%
% function [Solver,iflag] = read_solver_input();
%
% This procedure reads in from the screen the simulation
% parameters that control the solver operation.  New values
% of these parameters are read every time the program runs,
% even on restarts.
%
% Kenneth Beers
% Massachusetts Institute of Technology
% Department of Chemical Engineering
% 7/2/2001
%
% Version as of 7/25/2001



function [Solver,iflag] = read_solver_input();

iflag = 0;

func_name = 'read_solver_input';


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


disp(' ');
disp(' ');
disp('Enter the parameters for the steady state solver.');


% PDL> Input Solver.max_iter_time

disp(' ');
disp('First, enter the maximum number of iterations of the');
disp('implicit Euler method that is used to approach the');
disp('vicinity of the steady state solution.  If a value of');
disp('0 is entered, then no implicit Euler steps are performed');
disp('and the solver goes directly to Newtons method.');
disp( ' ');

% Solver.max_iter_time
```

```
check_real=1; check_sign=2; check_int=1;
prompt = 'Enter max. # of time iterations : ';
Solver.max_iter_time = get_input_scalar( ...
   prompt,check_real,check_sign,check_int);


% PDL> If Solver.max_iter_time IS NOT 0 THEN

if(Solver.max_iter_time ~= 0)

   disp(' ');
   disp('Enter data for the time integration stage.');


%        PDL> Input Solver.dt

   check_real=1; check_sign=1; check_int=0;
   prompt = 'Enter the time step dt (t) : ';
   Solver.dt = get_input_scalar(prompt, ...
      check_real,check_sign,check_int);


%        PDL> Input Solver.atol_time

   check_real=1; check_sign=1; check_int=0;
   prompt = 'Enter the abs. tolerance for the time integration : ';
   Solver.atol_time = get_input_scalar(prompt, ...
      check_real,check_sign,check_int);


% Otherwise, set dummy values

else

   Solver.dt = 1;
   Solver.atol_time = 1;


%PDL> ENDIF

end


%PDL> Input data for Newton's method solver,
%          Solver.max_iter_Newton, Solver.atol_Newton

disp(' ');
disp('Now enter parameters for Newtons method solver.');

% Solver.max_iter_Newton
check_real=1; check_sign=2; check_int=1;
```

```
prompt = 'Enter max. # of Newtons method iterations : ';
Solver.max_iter_Newton = get_input_scalar(...
   prompt,check_real,check_sign,check_int);

% Solver.atol_Newton
check_real=1; check_sign=1; check_int=0;
prompt = 'Enter abs. tolerance for Newtons method solver : ';
Solver.atol_Newton = get_input_scalar(...
   prompt,check_real,check_sign,check_int);


% PDL> Set Solver.iflag_Adepend to 0 to signify that
%   the A matrix obtained by discretizing the system
%   is not state-dependent.

Solver.iflag_Adepend = 0;


% PDL> Set Solver.iflag_nonneg to 1 to signify that
%   the components of the state vector should be
%   enforced to be non-negative at every iteration
%   of the solution procedure.

Solver.iflag_nonneg = 1;


% PDL> Input desired value for Solver.iflag_verbose

disp(' ');
check_real=1; check_sign=0; check_int=0;
prompt = 'Solver to be verbose (enter 1) or silent (other value) : ';
Solver.iflag_verbose = get_input_scalar( ...
   prompt,check_real,check_sign,check_int);


iflag = 1;

return;
```