TR_1D_model1_SS\assert_scalar.m

```
% TR_1D_model1_SS\assert_scalar.m
%
% function [iflag_assert,message] = assert_scalar( ...
%   i_error,value,name,func_name, ...
%   check_real,check_sign,check_int,i_error);
%
% This m-file contains logical checks to assert than an
% input value is a type of scalar number.  This function is
% passed the value and name of the variable, the name of
% the function making the assertion, and four integer
% flags that have the following usage :
%
% i_error : controls what to do if test fails
%       if i_error is non-zero, then use error()
%       MATLAB command to stop execution, otherwise
%       just return the appropriate negative number.
%       if i_error > 1, then dump current state to
%       dump_error.mat before calling error().
%
% check_real : check to examine whether input number is
% real or not.  See table after function header for set
% values of these case flags
% check_real = i_real (make sure that input is real)
% check_real = i_imag (make sure that input is
%           purely imaginary)
% any other value of check_real (esp. 0) results
%       in no check
%
%   check_real
%       i_real = 1;
%       i_imag = -1;
%
% check_sign : check to examine sign of input value
% see table after function header for set values
% of these case flags
% check_sign = i_pos (make sure input is positive)
% check_sign = i_nonneg (make sure input is non-negative)
% check_sign = i_neg (make sure input is negative)
% check_sign = i_nonpos (make sure input is non-positive)
% check_sign = i_nonzero (make sure input is non-zero)
% check_sign = i_zero (make sure input is zero)
% any other value of check_sign (esp. 0)
%       results in no check
%
% check_sign
%       i_pos = 1;
%       i_nonneg = 2;
%       i_neg = -1;
%       i_nonpos = -2;
%       i_nonzero = 3;
%       i_zero = -3;
%
% check_int : check to see if input is an integer
% if = 1, then check to make sure input is an integer
% any other value, perform no check
%
%
% Kenneth Beers
% Massachusetts Institute of Technology
% Department of Chemical Engineering
% 7/2/2001
%
% Version as of 7/21/2001


function [iflag_assert,message] = assert_scalar( ...
    i_error,value,name,func_name, ...
    check_real,check_sign,check_int);


iflag_assert = 0;
message = 'false';


% First, set case values of check integer flags.

% check_real
i_real = 1;
i_imag = -1;

% check_sign
i_pos = 1;
i_nonneg = 2;
i_neg = -1;
i_nonpos = -2;
i_nonzero = 3;
i_zero = -3;

iflag_assert = 0;


% Check to make sure input is numerical and not a string.
if(~isnumeric(value))
    message = [ func_name, ': ', ...
            name, ' is not numeric'];
    iflag_assert = -1;
    if(i_error ~= 0)
        if(i_error > 1)
```

```
         save dump_error.mat;
      end
      error(message);
   else
      return;
   end
end


% Check to see if it is a scalar.

if(max(size(value)) ~= 1)
   message = [ func_name, ': ', ...
            name, ' is not scalar'];
   iflag_assert = -2;
   if(i_error ~= 0)
      if(i_error > 1)
         save dump_error.mat;
      end
      error(message);
   else
      return;
   end
end


% Then, check to see if it is real.

switch check_real;

case {i_real}
   if(~isreal(value))
      message = [ func_name, ': ', ...
               name, ' is not real'];
      iflag_assert = -3;
      if(i_error ~= 0)
         if(i_error > 1)
            save dump_error.mat;
         end
         error(message);
      else
         return;
      end
   end

case {i_imag}
   if(real(value))
      message = [ func_name, ': ', ...
               name, ' is not imaginary'];
      iflag_assert = -3;
      if(i_error ~= 0)
```

```
         if(i_error > 1)
            save dump_error.mat;
         end
         error(message);
      else
         return;
      end
   end

end


% Next, check sign.

switch check_sign;

case {i_pos}
   if(value <= 0)
      message = [ func_name, ': ', ...
               name, ' is not positive'];
      iflag_assert = -4;
      if(i_error ~= 0)
         if(i_error > 1)
            save dump_error.mat;
         end
         error(message);
      else
         return;
      end
   end

case {i_nonneg}
   if(value < 0)
      message = [ func_name, ': ', ...
               name, ' is not non-negative'];
      iflag_assert = -4;
      if(i_error ~= 0)
         if(i_error > 1)
            save dump_error.mat;
         end
         error(message);
      else
         return;
      end
   end

case {i_neg}
   if(value >= 0)
      message = [ func_name, ': ', ...
               name, ' is not negative'];
      iflag_assert = -4;
```

```
        if(i_error ~= 0)
            if(i_error > 1)
                save dump_error.mat;
            end
            error(message);
        else
            return;
        end
    end

case {i_nonpos}
    if(value > 0)
        message = [ func_name, ': ', ...
                    name, ' is not non-positive'];
        iflag_assert = -4;
        if(i_error ~= 0)
            if(i_error > 1)
                save dump_error.mat;
            end
            error(message);
        else
            return;
        end
    end

case {i_nonzero}
    if(value == 0)
        message = [ func_name, ': ', ...
                    name, ' is not non-zero'];
        iflag_assert = -4;
        if(i_error ~= 0)
            if(i_error > 1)
                save dump_error.mat;
            end
            error(message);
        else
            return;
        end
    end

case {i_zero}
    if(value ~= 0)
        message = [ func_name, ': ', ...
                    name, ' is not zero'];
        iflag_assert = -4;
        if(i_error ~= 0)
            if(i_error > 1)
                save dump_error.mat;
            end
            error(message);
        else
```

```
            return;
        end
    end

end


% Finally, check to make sure it is an integer.

if(check_int == 1)
    if(round(value) ~= value)
        message = [ func_name, ': ', ...
                    name, ' is not an integer'];
        iflag_assert = -5;
        if(i_error ~= 0)
            if(i_error > 1)
                save dump_error.mat;
            end
            error(message);
        else
            return;
        end
    end
end


% set flag for succesful passing of all checks

iflag_assert = 1;
message = 'true';

return;
```