TR_1D_model1_SS\assert_structure.m

```
% TR_1D_model1_SS\assert_structure.m
%
% function [iflag_assert,message] = assert_structure(...
%    i_error,Struct,struct_name,func_name,StructType);
%
% This MATLAB m-file performs assertions on a data
% structure. It makes use of assert_scalar,
% assert_vector, and assert_matrix for the
% fields.
%
% INPUT :
% =======
% 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().
% Struct    This is the structure to be
%           checked
% struct_name   the name of the structure
% func_name     the name of the function making the
%               assertion
% StructType    this is a structure that contains the typing
%               data for each field.
%   .num_fields is the total number of fields
%   Then, for i = 1,2, ..., StructType.num_fields, we have :
%   .field(i).name          the name of the field
%   .field(i).is_numeric    if non-zero, then field is numeric
%   .field(i).num_rows      # of rows in field
%   .field(i).num_columns   # of columns in field
%   .field(i).check_real    value of check_real passed to assertion
%   .field(i).check_sign    value of check_sign passed to assertion
%   .field(i).check_int     value of check_int passed to assertion
%
% OUTPUT :
% =======
% iflag_assert  an integer flag telling of outcome
% message       a message passed that describes
%               the result of making the assertion
%
% Kenneth Beers
% Massachusetts Institute of Technology
% Department of Chemical Engineering
% 7/2/2001
%
```

```
% Version as of 7/25/2001


function [iflag_assert,message] = assert_structure(...
   i_error,Struct,struct_name,func_name,StructType);

iflag_assert = 0;
message = 'false';


% first, check to make sure Struct is a structure

if(~isstruct(Struct))
   iflag_assert = -1;
   message = [func_name, ': ', struct_name, ...
         ' is not a structure'];
   if(i_error ~= 0)
      if(i_error > 1);
         save dump_error.mat;
      end
      error(message);
   else
      return;
   end
end


% Now, for each field, perform the required assertion.

for ifield = 1:StructType.num_fields

   % set shortcut to current field type
   FieldType = StructType.field(ifield);

   % check if it exists in Struct
   if(~isfield(Struct,FieldType.name))
      iflag_assert = -2;
      message = [func_name, ': ', struct_name, ...
            '  does not contain ', FieldType.name];
      if(i_error ~= 0)
         if(i_error > 1)
            save dump_error.mat;
         end
         error(message);
      else
         return;
      end
   end

   % extract value of field
   value = getfield(Struct,FieldType.name);
```

```
% if the field is supposed to be numeric
if(FieldType.is_numeric ~= 0)

   % check to make sure field is numeric
   if(~isnumeric(value))
      iflag_assert = -3;
      message = [func_name, ': ', ...
            struct_name, '.', FieldType.name, ...
            ' is not numeric'];
      if(i_error ~= 0)
         if(i_error > 1)
            save dump_error.mat;
         end
         error(message);
      else
         return;
      end
   end


   % decide which assertion statement to use based on
   % array dimension of field value

   % If both num_rows and num_columns are set equal
   % to zero, then no check of the dimension of this
   % field is made.

   if(and((FieldType.num_rows == 0), ...
        (FieldType.num_columns == 0)))

      message = [func_name, ': ', ...
            struct_name,'.',FieldType.name, ...
            ' is not checked for dimension'];
      if(i_error ~= 0)
         disp(message);
      end


   % else, peform check of dimension to make sure
   % it is a scalar, vector, or matrix (i.e. a
   % two dimensional array).

   else

      % check that is is not a
      % multidimensional array
      if(length(size(value)) > 2)
         iflag_assert = -4;
         message = [func_name, ': ', ...
            struct_name,'.',FieldType.name, ...
        ' is multidimensional array'];
    if(i_error ~= 0)
        if(i_error > 1)
            save dump_error.mat;
        end
        error(message);
    else
        return;
    end

% else if scalar
elseif(and((FieldType.num_rows == 1), ...
           (FieldType.num_columns == 1)))
    assert_scalar(i_error,value, ...
        [struct_name,'.',FieldType.name], ...
        func_name,FieldType.check_real, ...
        FieldType.check_sign,FieldType.check_int);

% else if a column vector
elseif (and((FieldType.num_rows > 1), ...
            (FieldType.num_columns == 1)))
    dim = FieldType.num_rows;
    check_column = 1;
    assert_vector(i_error,value, ...
        [struct_name,'.',FieldType.name], ...
        func_name,dim,FieldType.check_real, ...
        FieldType.check_sign,FieldType.check_int, ...
        check_column);

% else if a row vector
elseif (and((FieldType.num_rows == 1), ...
            (FieldType.num_columns > 1)))
    dim = FieldType.num_columns;
    check_column = -1;
    assert_vector(i_error,value, ...
        [struct_name,'.',FieldType.name], ...
        func_name,dim,FieldType.check_real, ...
        FieldType.check_sign,FieldType.check_int, ...
        check_column);

% otherwise, a matrix
else
    assert_matrix(i_error,value, ...
        [struct_name,'.',FieldType.name], ...
        func_name, ...
        FieldType.num_rows,FieldType.num_columns, ...
        FieldType.check_real,FieldType.check_sign, ...
        FieldType.check_int);

end % selection of assertion routine
```

```
        end     % if perform check of dimension

    end         % if (FieldType.is_numeric ~= 0)

end             % for loop over fields


% set return results for succesful assertion

iflag_assert = 1;
message = 'true';

return;
```