Introduction

It is recommended to be familiar with how .fem or .bdf files are generated before using these functions. There are six functions that can be used for editing bdf and fem files. These functions are listed below:

Name	Description
rea_fem()	Uploads .fem file into MATLAB
read_bdf()	Uploads .bdf file into MATLAB
fem_cell2num()	Turns cellstring format of the bulk data into matrices.
fem_num2cell()	Turns matrices format of the bulk data into cellstrings.
print_fem()	Prints out a .fem file given cellstring format of the both the bulk data and the output requests.
Print_bdf()	Prints out a .bdf file given cellstring format of the both the bulk data and the output requests.

The general process is to first upload the file into MATLAB as a structure made from cell strings. These strings cut the file into 8-character chunks. Each chunk represents a cell in a card that would be read in a bdf of fem file. For example, a CHEXA in a bdf looks like:

1	2	3	4	5	6	7	8	9	10
CHEXA	EID	PID	G1	G2	G3	G4	G5	G6	
	G7	G8	G9	G10	G11	G12	G13	G14	
	G15	G116	G17	G18	G19	G20			

CHEXA	71	4	3	4	5	6	7	8	
	9	10	30	31			53	54	
	55	56	57	58	59	60			

The template has 27 cells for inputting information, 9 column cells and 3 rows.

In cell string format, cards are written using a cell string matrix. Each cell has a string less than 8 characters long. Empty cells will be written and 8 whitespace characters when sent to print. An example of the above CHEXA element in string format would look like:

chex_elm = {'71','4','3','4','5','6','7','8','';'9','10','30','31','','','53','54','';'55','56','57','58','59','60','','',''};

When printed onto the MATLAB console:

```
>>>    {'71'}  {'4' }  {'3' }  {'4' }  {'5'    }  {'6'    }  {'7'    }  {'8'    }  {0×0 char}
       {'9' }  {'10'}  {'30'}  {'31'}  {0×0 char}  {0×0 char}  {'53'    }  {'54'    }  {0×0 char}
       {'55'}  {'56'}  {'57'}  {'58'}  {'59'    }  {'60'    }  {0×0 char}  {0×0 char}  {0×0 char}
```

Hence, knowledge of each card's format and information (.bdf or .fem) are needed to take full advantage of the previously mentioned MATLAB functions.

Appending new Cards to the .bdf or .fem deck

New and old cards are organized by fields. For example; to access or create a CHEXA element write;

```
output.BULK.CHEXA{1} = chex_elem; %chex_elem is a custom card for a cubic element
```

This says that the file data in output has bulk data in BULK and then overwrites the first CHEXA card with the new element chex_elem.

For faster editing, the cellstrings can be formatted into matrices with only the numerical values being preserved. To do this use the function fem_cell2num() on 'output'. All fields will be translated into matrices. The first element will then be formatted as:

```
output.CHEXA_MAT(1,:)
```

```
>>> 71 4 3 4 5 6 7 8 9 10 30 31 nan nan 53 54 55 56 57 58 59 60
```

Where the empty spaces are treated as nans.

Similarly, matrices can be translated back into cellstring format using fem_num2cell(). This is done because the functions which write out the bdf and fem files only accept cellstring format.

The field names for the cellstring format are given in the table below. All fieldnames must be in capital letters only.

Table 1 : OPTISTRUCT FEM Template

Field Name	Description
IO	Input and output arguments Is a cell list of strings (dimension is nx1 where n is the number of requests). Contain of the strings must be the commands you would append to the top of the fem file. These include output requests and other type of included files or requests.

OBJECTIVES	Field of Objective information. Contains subfields of DESGLB and DESOBJ.
	To define a DESOBJ write:
	`output.OBJECTIVES.DESOBJ{i} = {'MIN', '1'};`
	To define a DESGLB write the same as a DESOBJ:
	`output.OBJECTIVES.DESGLB{i} = '1';`
	Where i is the objective number. Most of the time it is i=1.

	If additional objective information is needed such as DESSUB, then it is added to the subcase.
TITLE	Title information. Titles are added as strings. To define a title, write: `output.TITLE{i}='TITLE';` Where I is the title number .
SUBCASE	Contains information about the subcases. It is the same format as the BDF files/function. Each subcase is a structure defined as: “`output.SUBCASE(i)`” where “i” is the subcase number. The SUBCASE field contains additional fields which take string data as an input. These filed include: LABEL, ANALYSIS, SPC, LOAD, DESSUB, WEIGHT and etc.. All inputs must be strings.
BULK	Contains bulk data card names Uses template described previously at the beginning of this section of the manual. Additional control cards such as PARAM can be added inside the bulk using the field: `output.BULK.CONTROL_CARDS` The inputs to this field is a cellstring list.

When NASTRAN files are being read or built, the overall data structure is unchanged when compared to OPTISTRUCT/HYPERMESH. However, there are a few additions.

Table 2 : NASTRAN file Template

Field Name	Description
STATEMENTS	Is a cell list of strings for the NASTRAN STATEMENTS. To define a statement, write out the text as you would in a bdf but exclude the NASTRAN STATEMENT part. Ex; output.STATEMENTS = `{' BUFFSIZE=65537';' Q4TAPER=0.9';' Q4SKEW=10.0'; ' T3SKEW=1.3'}`
FILE_MANAGEMENT	Input is a cell list of strings for the File management of a bdf file.
EXECUTIVE_CONTROL	Input is a structure with fields ID, SOL for the solution type and TIME. Inputs to the subfields must be strings. Example: output.EXECUTIVE_CONTROL.SOL = `'101';`
CASE_CONTROL	A cell list of strings for the case control. Includes titles, label and other output requests. For unsupported requests (in the function), add them as a string in CASE_CONTROL. For fatigue analysis, the requests must be made in this section. (as cellstrings, the MATLAB code will translate the fatigue requests into OPTISTRUCT template)
OBJECTIVES	NOTE: Objectives not supported for NASTRAN bdf files. If they are required, then update the function with the code from the OPTISTRUCT function. Or use CASE_CONTROL to define the objective information as strings.

	Field of Objective information. Contains subfields of DESGLB and DESOBJ.
	To define a DESOBJ write:
	
	`output.OBJECTIVES.DESOBJ{i} = {'MIN', '1'};`
	
	To define a DESGLB write the same as a DESOBJ:
	
	`output.OBJECTIVES.DESGLB{i} = '1';`
	
	Where i is the objective number. Most of the time it is i=1.
	
	If additional objective information is needed such as DESSUB, then it is added to the subcase.

SUBCASE	Contains information about the subcases. It is the same format as the FEM files/function. Each subcase is a structure defined as: "`output.SUBCASE(i)`" where "i" is the subcase number. The SUBCASE field contains additional fields which take string data as an input. These filed include: LABEL, ANALYSIS, SPC, LOAD, DESSUB, WEIGHT etc... All inputs must be strings.
BULK	Contains bulk data card names Uses template described previously at the beginning of this section of the manual. Additional control cards such as PARAM can be added inside the bulk using the field: `output.BULK.CONTROL_CARDS` The input to this field is a cellstring list.

Another example:

Code example of writing and printing out a .fem file

```
%path name
path_name = "RectangleB_div2.fem"; % arbitrary filename

%upload the file into matlab
fprintf('Uploading .fem File into Console\n')
output = read_fem(path_name);

%output is in cellstring format

% output.IO : Input/output requests for fem file
%     {'DISPLACEMENT(SORT1,PUNCH) = ALL'}
%     {'ESE(PUNCH) = ALL'              }
%     {'STRAIN(SORT1,PUNCH) = ALL'     }
%     {'STRESS(SORT1,PUNCH) = YES'       }

%output.SUBCASE(1) : 1st Subcase information, is a structure, takes string
%input
%        LABEL: "Forces"
%     ANALYSIS: "STATICS"
%          SPC: "1"
%         LOAD: "2"

%output.BULK : is the bulk data as a structure with fieldnames equal to the
%card names
%      CHEXA: {3360×1 cell}
%      FORCE: {9×1 cell}
%       GRID: {4305×1 cell}
%       MAT1: {3×1 cell}
%     PSOLID: {4×1 cell}
%        SET: {{2×9 cell}}
%        SPC: {32×1 cell}

%access the first cell for CHEXA element
%output.BULK.CHEXA{1}
%
%     {'561' }    {'1'   }    {'123'   }    {'1200'  }    {'2966'  }
{'1211'  }    {'1212'  }    {'2968'  }    {0×0 char}
%     {'2965'}    {'2971'}    {0×0 char}    {0×0 char}    {0×0 char}    {0×0
char}    {0×0 char}    {0×0 char}    {0×0 char}

%To convert into matrix format:
fprintf('Convert bulk data into matrix data bulk structure\n');
new_output = fem_cell2num(output);

%the new_output
%      CHEXA_MAT: [3360×10 double]
%      FORCE_MAT: [9×7 double]
%       GRID_MAT: [4305×9 double]
%       MAT1_MAT: [3×9 double]
%     PSOLID_MAT: [4×9 double]
%        SET_MAT: {[1×421 double]}
%        SPC_MAT: [32×9 double]

%SETS and list are converted into a cell structure of vectors.
%new_output.SET_MAT
%{1×421 double}
%the first number is the id of the set. Additional edits for the tpe of set
%must be don in cellstring format.



%Convert back into cellstring format:
fprintf('Convert bulk data into cellstring data bulk structure\n');
new_output2 = fem_num2cell(new_output,'fem'); %could also convert bulk to
bdf

%the bulk data of new_output2
%      CHEXA_MAT: {3360×1 cell}
%      FORCE_MAT: {9×1 cell}
%       GRID_MAT: {4305×1 cell}
%       MAT1_MAT: {3×1 cell}
%     PSOLID_MAT: {4×1 cell}
%        SET_MAT: {{54×8 cell}}
%        SPC_MAT: {32×1 cell}

%to remove the _MAT from the fieldnames
old_fieldnames = fieldnames(new_output2);
new_fieldnames = strrep(old_fieldnames,'_MAT','');
map = containers.Map(old_fieldnames,new_fieldnames);
tmp = reshape(values(map,fieldnames(new_output2)),1,[]);
tmp(2,:) = num2cell(permute(struct2cell(new_output2),[3,1,2]),1);
new_output3 = reshape(struct(tmp{:}),size(new_output2));

%new_output3
%      CHEXA: {3360×1 cell}
%      FORCE: {9×1 cell}
%       GRID: {4305×1 cell}
%       MAT1: {3×1 cell}
%     PSOLID: {4×1 cell}
%        SET: {{54×8 cell}}
%        SPC: {32×1 cell}

%append the new fieldnames to the bulk of output
final_output = output;
final_output.BULK = new_output3;

%new path
new_path = "new_file.fem";
%write out the file to a new path
% 0, print Begin BULK and END DATA
% 1, print Begin BULK
% 2, print END DATA
print_fem(final_output, new_path,0)
```