# Capacitance Extraction Script
## *User Manual*

Stav Zaitsev[*]
Electrical Engineering Department,
Technion - Israel Institute of Technology,
Haifa, Israel

October 26, 2009

## 1 Introduction

The Matlab script `capacitance_extraction_gui.m` loads a graphical user interface (GUI) which allows an interactive extraction of capacitance matrices for two conductors of virtually arbitrary geometry, as described in [1]. The main window of the GUI can be seen in Fig. 1.

First, the two conductors are defined as two rectangular right angle prisms. After the meshing, an arbitrary form function can be applied to the mesh nodes of each conductor, as explained below. As a result, it is possible to model many different device geometries. In the example used here, the first conductor (mechanical beam resonator) is bent by using a cosine form function, and the second conductor (stationary electrode) is simply shifted in $y$ direction.

## 2 Capacitance extraction flow

### 2.1 Parameter table

In order to define all the parameters of the original rectangular beams and the original triangular mesh, the parameter table on top of the GUI window is used. Note that changing any parameter in this table invalidates any previous extraction results.

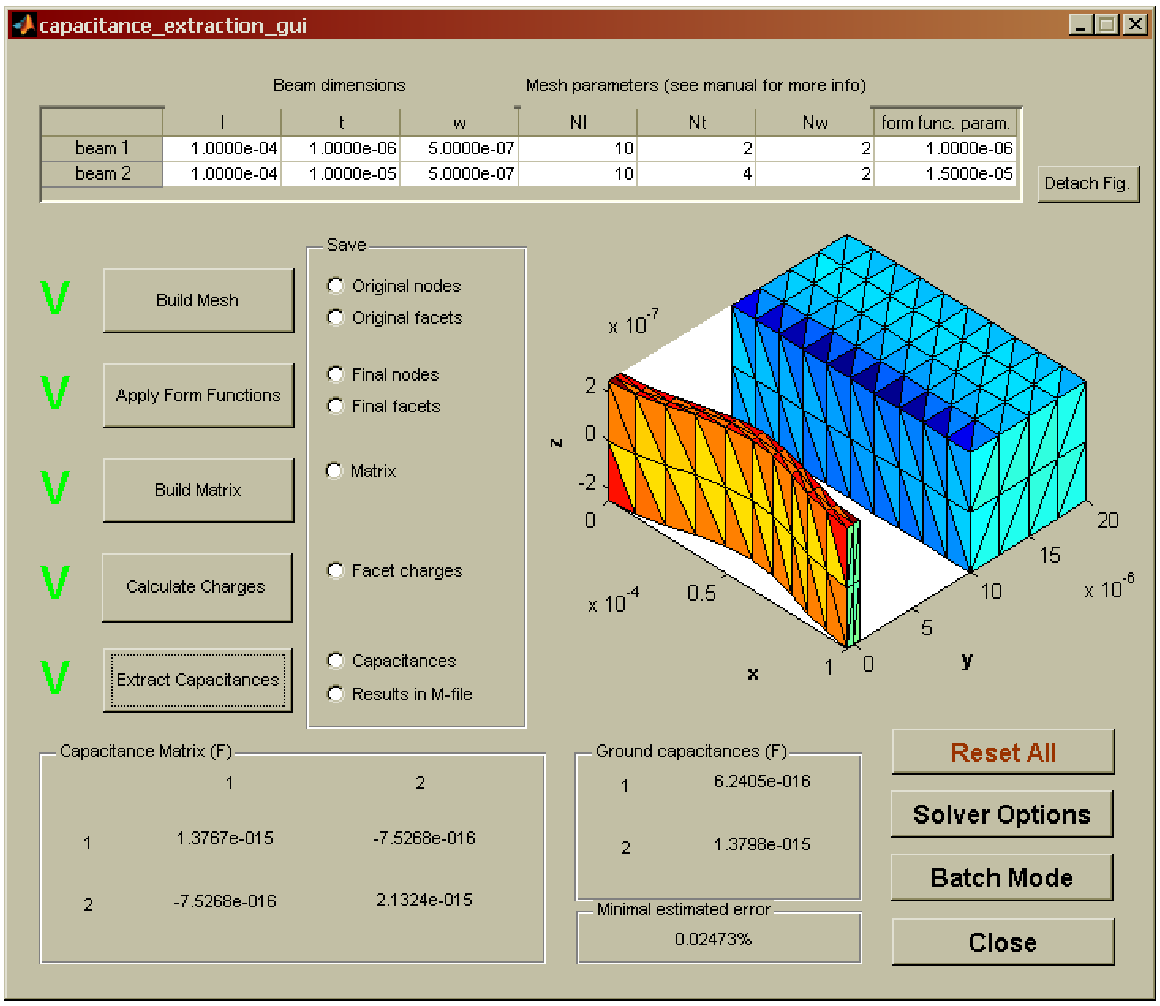
In addition to conductor parameters, two additional parameters are defined in this table. Parameters `form func. param` are passed to the two form functions which are employed for the first and second conductor, respectively. In our example, the form function of the first conductor receives the value $10^{-6}$, and the form function of the second conductor receives the value $1.5 \times 10^{-5}$. The exact syntax of the form function calls will be explained in Sec. 2.3. In our case, the value for the first form function defines the maximum deflection value of the doubly clamped beam, while the value for the second form function defines the amount of shift in $y$ direction.

### 2.2 Solver parameters

As explained in Ref. [1], operator matrix calculation requires evaluation of multiple integrals over triangular facets. In order to increase calculation speed and efficiency, several integration methods are used according to the distance between the test point and the source facet of the integral. The distances are called far field radius and close field radius. Usually, the script calculates these distances based on the size of the facets in the mesh. By default, far field radius is equal to 30 times the length of the largest side of any facet, while the close field radius is equal twice the length of the smallest side.

These values are automatically recalculated after any change to the parameter table. However, the user can set these parameters manually using the "Solver Options" button. Then, the button text color changes to green. This behavior is reset after "Reset All" button is pressed.

[*]e-mail: zzz@tx.technion.ac.il



Beam dimensions | Mesh parameters (see manual for more info)

| | l | t | w | Nl | Nt | Nw | form func. param. |
|---|---|---|---|---|---|---|---|
| beam 1 | 1.0000e-04 | 1.0000e-06 | 5.0000e-07 | 10 | 2 | 2 | 1.0000e-06 |
| beam 2 | 1.0000e-04 | 1.0000e-05 | 5.0000e-07 | 10 | 4 | 2 | 1.5000e-05 |

Capacitance Matrix (F)

| | 1 | 2 |
|---|---|---|
| 1 | 1.3767e-015 | -7.5268e-016 |
| 2 | -7.5268e-016 | 2.1324e-015 |

Ground capacitances (F)

| | |
|---|---|
| 1 | 6.2405e-016 |
| 2 | 1.3798e-015 |

Minimal estimated error
0.02473%

Figure 1: Main window of `capacitance_extraction_gui` script.

## 2.3 Extraction flow buttons

The extraction flow follows six steps, which are represented in the main GUI window by six buttons on the left. After sucessful completion of each step, the black circle near the corresponding button changes to a green "V" sign.

The first button creates a mesh covering the original conductors by using an external function `mesh_rectangular_prism.m`. The function call syntax is

```
function [node_list facet_list]=mesh_rectangular_prism(l,t,w,Nl,Nt,Nw),
```

where the first three arguments are the dimensions of a given prism, and the last three arguments are the grid division numbers in $x$, $y$, and $z$ directions, respectively.

The `mesh_rectangular_prism.m` function returns a node list and a facet list for the prism. These original node and facet lists can be saved using the corresponding radio buttons into files named `orig_node_list1.txt`, `orig_node_list2.txt`, `facet_list1.txt`, and `facet_list2.txt`. Note that nodes and facets for each rectangular prism are saved separately.

The exact syntax of node list and facet list is shown below. Node list:

```
0.0000000000000000e+000   4.9999999999999998e-007  -2.4999999999999999e-007
1.0000000000000001e-005   6.9098300562505262e-007  -2.4999999999999999e-007
2.0000000000000002e-005   1.1909830056250527e-006  -2.4999999999999999e-007
3.0000000000000004e-005   1.8090169943749472e-006  -2.4999999999999999e-007
4.0000000000000003e-005   2.3090169943749475e-006  -2.4999999999999999e-007
5.0000000000000002e-005   2.4999999999999998e-006  -2.4999999999999999e-007
6.0000000000000008e-005   2.3090169943749470e-006  -2.4999999999999999e-007
7.0000000000000007e-005   1.8090169943749467e-006  -2.4999999999999999e-007
8.0000000000000007e-005   1.1909830056250523e-006  -2.4999999999999999e-007
...
```

Here, in each row the three coordinates $x$, $y$, and $z$ of a node are listed.

In facet list

```
1 12 13
1 2 13
2 13 14
2 3 14
3 14 15
... ,
```

each row represents a single triangular facet. The numbers of the nodes constituting this facet are given in the row. The coordinates of each node can be found in the node list described above.

All intermediate results are saved in simple text format with tab delimiters.

The second button, "Apply Form Functions", applies form functions to the node lists of the two conductors, and combines the resulting node lists and facet lists. The form functions are defined by user in files `beam1_form_function.m` and `beam2_form_function.m`. The syntax of the form function for the first conductor in our example is

```
function [node_list_out]=beam1_form_function(node_list,l,t,w,form_func_param)
node_list_out=node_list;
node_list_out(:,2)=node_list(:,2)+form_func_param*(1-cos(2*pi()*(node_list(:,1)/l-1)));
```

Here, `form_func_param` is the deflection (bending) amplitude. The above function creates a doubly clamped beam with a cosine form and bending amplitude `form_func_param` from a rectangular prism. The bending is in $y$ direction, hence the index 2 in the `node_list` column references.

The default form function for the second conductor shifts all the nodes in $y$ direction by the distance defined by the second `form func. param`. Figure 2 shows the resulting meshes of the two conductors after the form functions were applied. Note that scaling is not uniform along different axes.

The modified node list and facet list can be saved in files named `node_list.txt` and `facet_list.txt` by checking the respective radio buttons to the right of the "Apply Form Functions" button.

The most computationally intensive step is the calculation of the operator matrix. This is done by pressing the "Build Matrix" button. As the calculation progresses, a countdown from 9 to 0 is carried out to the left of the button.

Two external functions are employed to calculate the operator matrix [1]. Both functions calculate the value of the integral

$$I = \oint \frac{dS}{|r_0 - r|}, \tag{1}$$

where $S$ is the area of a triangular facet, $r$ is the position vector which belongs to $S$, and $r_0$ is the test point, which can lay either on the facet itself (self-term) or outside the facet (cross-terms). The first function returns the exact analytical value of (1), and its syntax is

```
function I=triangle_analytical_integration(a,b,c,area,r0),
```

where `a`, `b`, and `c` are the vectors of three nodes defining the facet, `area` is the area of the facet, and `r0` is the test point position. Another function calculates the value of (1) by means of seven point Gaussian quadrature approximation. The syntax of this function is

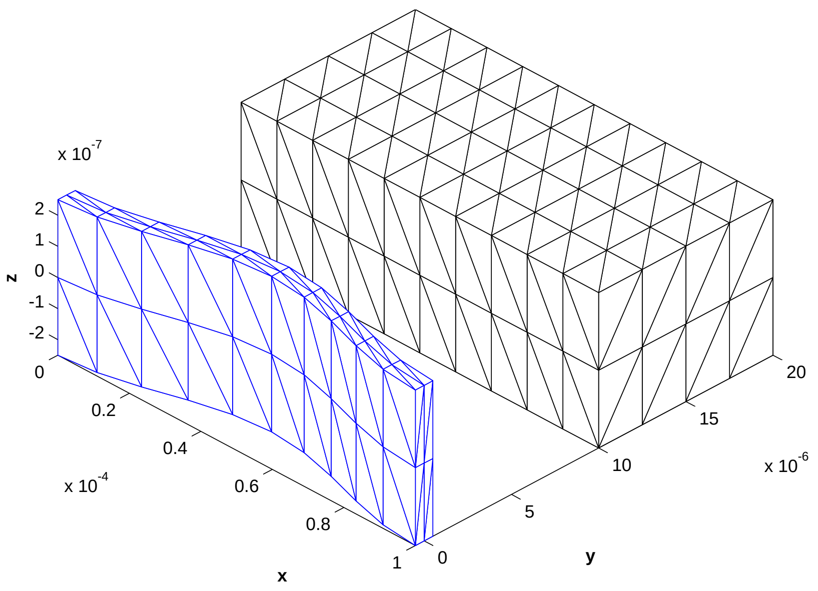


Figure 2: Triangular mesh after applying form functions.

```
function I=triangle_Gaussian_quadrature(a,b,c,area,r0),
```

where all the input definitions coincide with those given above.

After this step is completed, a graphical representation of the resulting matrix is presented, as shown in Fig. 3. With some experience, this representation can be used to diagnose serious problems in the capacitance extraction process. Bear in mind that elements on the main diagonal are self-terms [2], and should have relatively large values. Also, there are two blocks on the diagonal, which correspond to cross elements on the same conductor. If the conductors are well separated, these cross-elements should be larger than the cross-elements involving test point on one conductor and source facet on the other. These far cross-elements are located at the left lower and right upper corners of the matrix.

The resulting matrix in text format with tab delimiters can be saved in file named `matrix.txt` using the radio button to the right of the "Build Matrix" button.

Finally, the charge distributions can be calculated and the capacitance values extracted by pressing "Calculate Charges" and "Extract Capacitances", respectively. As explained in Ref. [1], two charge distributions are calculated, one with a unity voltage applied to the first conductor and zero voltage applied to the second conduct, and the other with voltages inverted. The results of the first case are shown in Fig. 4. It is important to make a "sanity check" by insuring that a conductor with positive voltage applied has positive charge distribution, which is represented by yellow and red colors. Negative charges are represented by blue color.

As usual, the charge distributions can be saved to text files named `charges_V1=1_V2=0.txt` and `charges_V1=0_V2=1.txt` by checking the appropriate radio button. The charges saved are total charges per each facet. In order to calculate the charge densities, the saved values should be divided by facet areas.

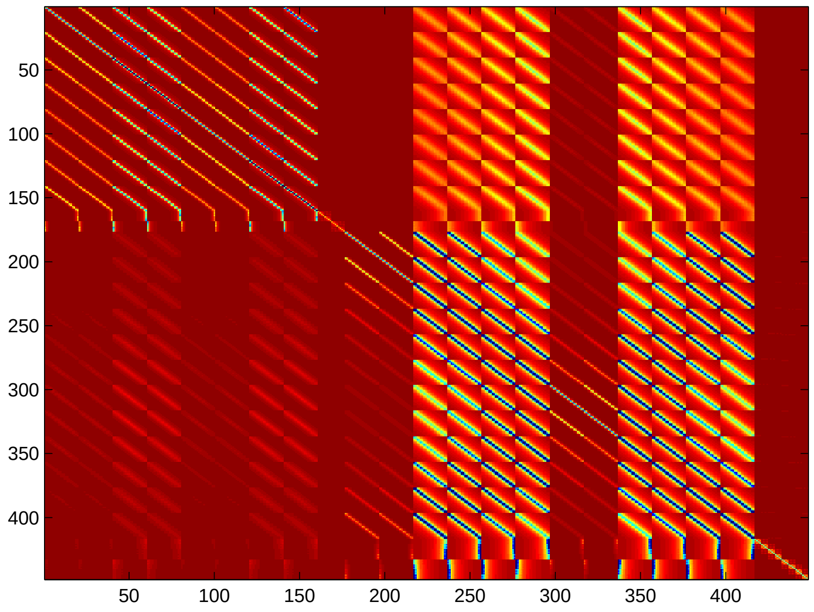


Figure 3: Graphical representation of the operator matrix. The values of the elements are represented by color, blue is close to zero, red is largest value. Note that the self elements on the main diagonal are most significant.

## 2.4 Reading and saving results

The final capacitance matrix and ground capacitance values are shown at the lower part of the main window. They can be also saved to a text file named `capacitances.txt`. Below the contents of a typical `capacitances.txt` are presented:

```
Capacitance matrix (F)
1.37673e-015 -7.52684e-016
-7.52684e-016 2.13245e-015

Ground capacitances (F)
C10 6.24051e-016
C20 1.37976e-015

Minimal estimated error: 0.0247296
```

Minimal estimated error is calculated as the relative error between the extracted values of $C_{12}$ and $C_{21}$. Ideally, these two values should be equal. We use their average value in the capacitance matrix.

All the results of a single script run can be saved to a binary Matlab file called `results.mat` using the lowest radio button. The file contains the `handles` structure of the main window figure. The structure fields containing important information are:

**table_beam_param** The parameter table handle, use `get(handles.table_beam_param,'Data')` to read the table values

**P** The operator matrix

**facet_centers** List, in which each row represents the position of the center of the corresponding facet

**facet_areas** List, in which each row represents the area of the corresponding facet

**node_list** Final node list, containing both conductors

**facet_list** Final facet list

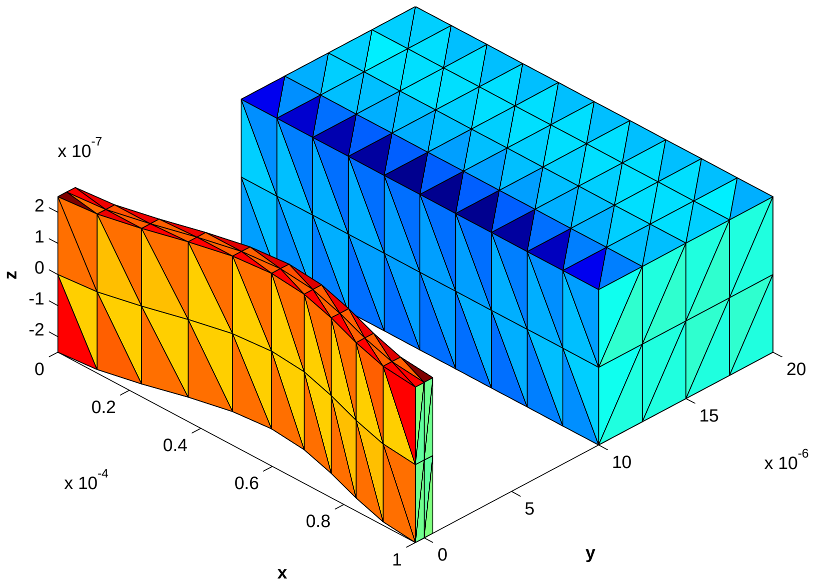


Figure 4: Charge distribution on the triangular mesh. Yellow and red color represent positive charges, blue color represent negative charge. Positive voltage is applied to the bent beam, while zero voltage is applied to the larger stationary electrode.

**node1_list, node2_list** Node lists of each conductor after application of the form functions

**facet1_list, facet2_list** facet lists of each conductor after application of the form functions

**sigma1** Facet charges (not charge densities) for the case in which the potential of the first conductor is 1 V and the potential of the second conductor is 0 V

**sigma2** Facet charges (not charge densities) for the case in which the potential of the first conductor is 0 V and the potential of the second conductor is 1 V

**custom_field_radii** A flag which is equal to 1 if custom far and close field radii were used, 0 otherwise

**close_field_radius** The close field radius

**far_field_radius** The far field radius

**orig_node_list1, orig_node_list2** Original node lists

**facet_list1, facet_list2** Original facet lists

**number_of_nodes1, number_of_nodes2** Number of nodes in the mesh of the first and second conductor, respectively

**number_of_facets1, number_of_facets2** Number of facets in the mesh of the first and second conductor, respectively

**number_of_facets** Total number of facets

**number_of_nodes** Total number of nodes

**C12, C21** The extracted cross-capacitance values

**C11, C22, C12_avg** The capacitance values used in the final capacitance matrix

**C10, C20** Ground capacitance values

**C_error** Minimal estimated error

## 2.5 Batch mode

The user can run the script in batch mode, in which the form function parameters are swept, and all results are saved in tabular form in `batch_results.txt` file. The sweep ranges can be defined using standard Matlab syntax by pressing the "Batch Mode" button. For the duration of the batch run, all irrelevant GUI buttons are disabled and the "Batch Mode" button text is changed to "Press to stop", displayed in green color.

Contents of a typical `batch_results.txt` file are shown below.

```
Beam parameters:
l t w
0.0001 1e-006 5e-007
0.0001 1e-005 5e-007

Mesh parameters:
Nl Nt Nw
20 2 2
20 4 2

param1 param2 C11(F) C22(F) C12_avg(F) C10(F) C20(F) C_error(%)
-3e-006 1e-005 1.43037e-015 2.19788e-015 8.25386e-016 6.04982e-016 1.3725e-015 0.0212044
-3e-006 1.2e-005 1.3656e-015 2.10944e-015 7.28375e-016 6.37225e-016 1.38107e-015 0.0167914
-3e-006 1.4e-005 1.32185e-015 2.0481e-015 6.57915e-016 6.63931e-016 1.39019e-015 0.0140866
-3e-006 1.6e-005 1.28987e-015 2.0025e-015 6.0314e-016 6.86733e-016 1.39936e-015 0.0122349
-3e-006 1.8e-005 1.26533e-015 1.96706e-015 5.58716e-016 7.06615e-016 1.40835e-015 0.0108745
-3e-006 2e-005 1.24584e-015 1.93867e-015 5.21618e-016 7.24225e-016 1.41705e-015 0.00982372
-3e-006 2.2e-005 1.22997e-015 1.91538e-015 4.89963e-016 7.40012e-016 1.42542e-015 0.00898136
-3e-006 2.4e-005 1.2168e-015 1.89594e-015 4.62499e-016 7.543e-016 1.43344e-015 0.00828629
-3e-006 2.6e-005 1.20569e-015 1.87947e-015 4.38354e-016 7.67334e-016 1.44111e-015 0.00769943
-3e-006 2.8e-005 1.1962e-015 1.86535e-015 4.16898e-016 7.79301e-016 1.44845e-015 0.00719463
...
```

The results are arranged into tab delimited eight columns, each row representing a single combination of two `form_func_param` parameters.

## References

[1] Stav Zaitsev, *Capacitance of a doubly clamped beam mechanical resonator.* Final project in Introduction to Computational Physics course 118094, Electrical Engineering, Technion, Spring 2009.

[2] W. C. Gibson, *The Method of Moments in Electromagnetics.* Chapmann & Hall/CRC, 2008.