F^1-Transform Matlab Viewer

User Manual

Marek Vajgl, Petra Hoďáková - marek.vajgl@osu.cz, petra.hodakova@osu.cz
Institute for Research and Applications of Fuzzy Modelling – http://irafm.osu.cz
University of Ostrava
30. dubna 22, 701 03 Ostrava, Czech Republic

Table of content

Introduction

The aim of the application is to demonstrate behavior of F^1-transform in in the field of discrete 2D data.

Application is based on the F^1-transform, what is fuzzy transformation of first degree introduced during research at the Institute of Research and Application of Fuzzy Modeling at University of Ostrava. This transformation is integral transform mapping input data space into the space of components, which can be used as an approximation of the values of the original function in following processing. Finally, when inverse F-transform is applied over the components, the reconstruction of the original data is obtained.

Created application takes a 2D array of values as an input. The input is visualized using graphs (figures) in the Matlab environment. One component or all components can be selected to be shown.

Installation and startup requirements

Startup requirements

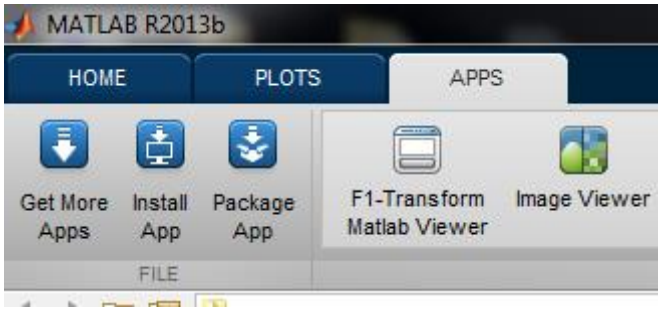
Application was created and tested in the Matlab R2013 environment with *Image Processing Toolbox*, but it does not use any specific or special functions. Therefore it should be executed on the previous versions of Matlab environment. However, tested minimal required environment is:

- Matlab 2013b
- Image Processing Toolbox - http://www.mathworks.com/products/image/?refresh=true

Postup instalace

Installation is started by double click over the install package file *F1-Transform Matlab Viewer.mlappinstall*.

When the installation process is started, Matlab environment is executed asking for confirmation of the installation. After the confirmation the application is installed and finally the *F1-Transform Matlab Viewer* icon is shown at the main ribbon *Apps* of the Matlab Environment.



Execution of the installed application

Application is executed by doubleclick over the *F1-Transform Matlab Viewer* icon in *Apps* ribbon in Matlab environment.

Application usage

Preparation of input data

For an input of the application the prepared data in the file in **.mat* format are used, input is save in some variable in this file. When application is started, you need to specify the *.mat* file and the variable name to continue.

The variable will contain two-dimensional array of values of *double* data type. First dimension of the array represents first dimension of the function (or X coordinate), second dimension represents second dimension of the function (or Y coordinate). Exact value for [x,y] coordinate is stored in the input array at this coordinate. So, at [1,1], [1,2], …, [1,n], [2,1], [2,2], …, [m,n] should be values of *f(1,1), f(1,2), …, f(1,n), f(2,1), f(2,2), …, f(m,n)*.

Simple example of the stored data in the variable *d* in the matrix 4x6x1:

```
d =

    1.5000    3.0000    4.5000    6.0000    7.5000    9.0000
    1.3500    2.7000    4.0500    5.4000    6.7500    8.1000
    1.2000    2.4000    3.6000    4.8000    6.0000    7.2000
    1.0500    2.1000    3.1500    4.2000    5.2500    6.3000
```

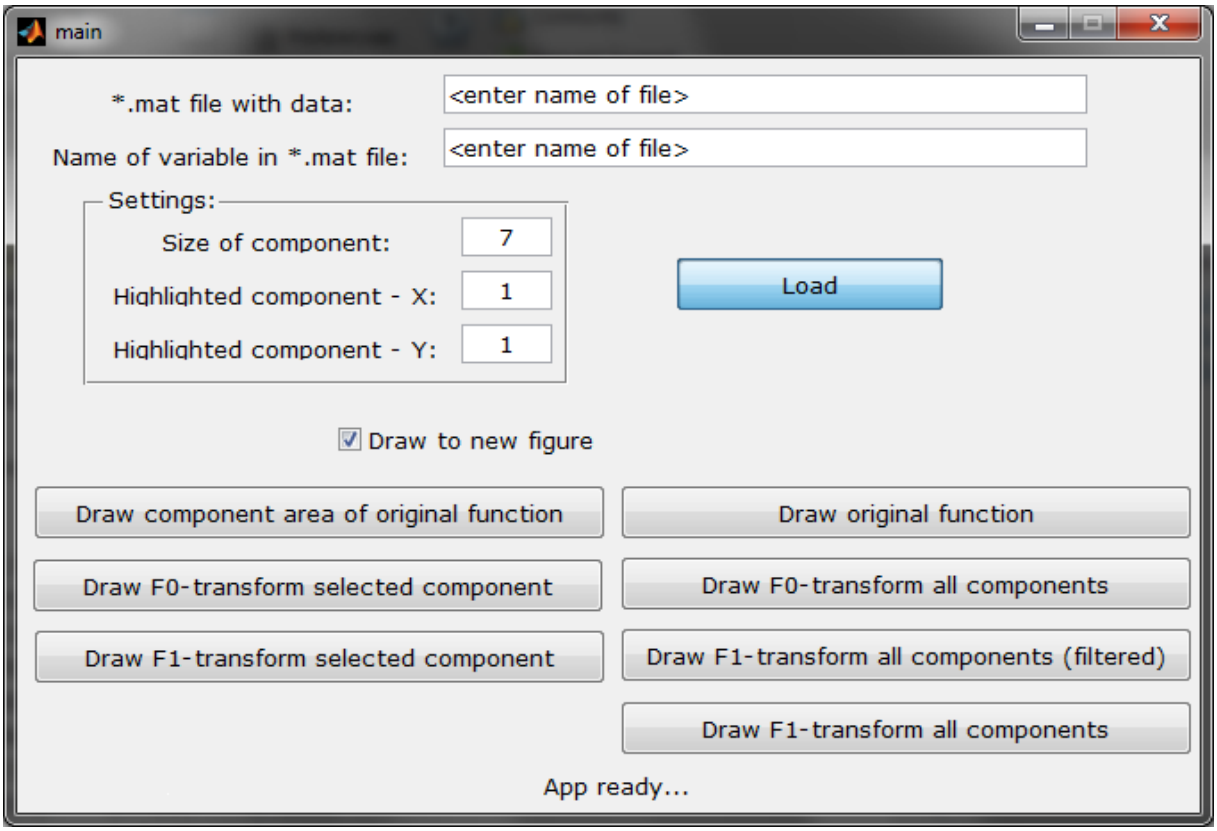
Data are evaluated only using integers over X and Y axis, therefore all data have to had the same scale at both axes (with step 1).

Execution and main window description

Application is executed by command

```
main
```

. After execution the main window of the application is shown.



The main form of the application offers a following user controls to interact:

- **.mat file with data* – name of the file which contains the variable with input data. This file will be loaded when *Load* button is pressed.
- *Name of variable in *.mat file* – name of the variable containing input data in the input file in correct format. This variable is analysed when *Load* button is pressed.
- *Settings*
    - *Size of component* – number of points covered by one component. Higher value produces lower total number of components and therefore faster calculation. Lower value will cause higher total number of components, what means that approximation of the original function will be more precise. Allowed values are odd values greater than three.
    - *Highlighted component X* – for specific operations (see following explanation of buttons), this value defines which *x* component is selected to be visualized.
    - *Highlighted component Y* – for specific operations (see following explanation of buttons), this value defines which *y* component is selected to be visualized.
- *Draw to new figure* – if checked, the drawing operations draw result in the new figure (new image). If unchecked, the drawing operations draw result in the last used figure. Using this feature you can draw multiple items in the same figure to join multiple visualisations.

The remaining buttons on the form represents operations, how the data can be evaluated and viewed. Buttons on the left side are used to work with selected component only (see *highlighted component* options above), buttons on the right side work displays all data:

- *Draw component area as original function* – draws only such area of the original function which is covered by basis functions of the selected component.
- *Draw F0-transform selected component* – draws only selected F^0 component (scalar value) as dot in the graph.
- *Draw F1-transform selected component* – draws only selected F^1-component (vector) as square area in the graph.
- *Draw original function* – draws whole original function as a graph.
- *Draw F0-transform all components* – draw all F^0-components (scalar values) as a dots in the graph.
- *Draw F1-transform all components* – draw all F^1-components as a set of multiple areas in the graph.
- *Draw F1-transform all components (filtered)* – draw every second F^1-component as a set of multiple areas in the graph.

Example of usage

This simple example will show usage of the application in step-by-step tutorial.

Step 1 – input data preparation

At the beginning we need some function we would like to represent as an input. We can create two vectors of two variables. The first command for Matlab will be:

```
a = sin(0:pi/8:3*pi)
```

We will obtain vector:

```
a =
  Columns 1 through 7
0    0.3827    0.7071    0.9239    1.0000    0.9239    0.7071

  Columns 8 through 14
0.3827     0.0000    -0.3827    -0.7071    -0.9239    -1.0000    -
0.9239

  Columns 15 through 21
-0.7071    -0.3827    -0.0000     0.3827     0.7071     0.9239
1.0000

  Columns 22 through 25
0.9239    0.7071    0.3827    0.0000
```

Second command for Matlab will be:

```
b = (0:0.25:5)'
```

The result will be:

```
b =

           0
```

```
    0.2500
    0.5000
    0.7500
    1.0000
    1.2500
    1.5000
    1.7500
    2.0000
    2.2500
    2.5000
    2.7500
    3.0000
    3.2500
    3.5000
    3.7500
    4.0000
    4.2500
    4.5000
    4.7500
    5.0000
```

The whole function will be obtained using command

```
c = b * a
```

The result (trimmed):

```
c =

  Columns 1 through 7

0         0         0         0         0         0         0
0    0.0957    0.1768    0.2310    0.2500    0.2310    0.1768
0    0.1913    0.3536    0.4619    0.5000    0.4619    0.3536
0    0.2870    0.5303    0.6929    0.7500    0.6929    0.5303
0    0.3827    0.7071    0.9239    1.0000    0.9239    0.7071
0    0.4784    0.8839    1.1548    1.2500    1.1548    0.8839
0    0.5740    1.0607    1.3858    1.5000    1.3858    1.0607
0    0.6697    1.2374    1.6168    1.7500    1.6168    1.2374
0    0.7654    1.4142    1.8478    2.0000    1.8478    1.4142
0    0.8610    1.5910    2.0787    2.2500    2.0787    1.5910
0    0.9567    1.7678    2.3097    2.5000    2.3097    1.7678
0    1.0524    1.9445    2.5407    2.7500    2.5407    1.9445
0    1.1481    2.1213    2.7716    3.0000    2.7716    2.1213
0    1.2437    2.2981    3.0026    3.2500    3.0026    2.2981
0    1.3394    2.4749    3.2336    3.5000    3.2336    2.4749
0    1.4351    2.6517    3.4645    3.7500    3.4645    2.6517
0    1.5307    2.8284    3.6955    4.0000    3.6955    2.8284
0    1.6264    3.0052    3.9265    4.2500    3.9265    3.0052
0    1.7221    3.1820    4.1575    4.5000    4.1575    3.1820
```

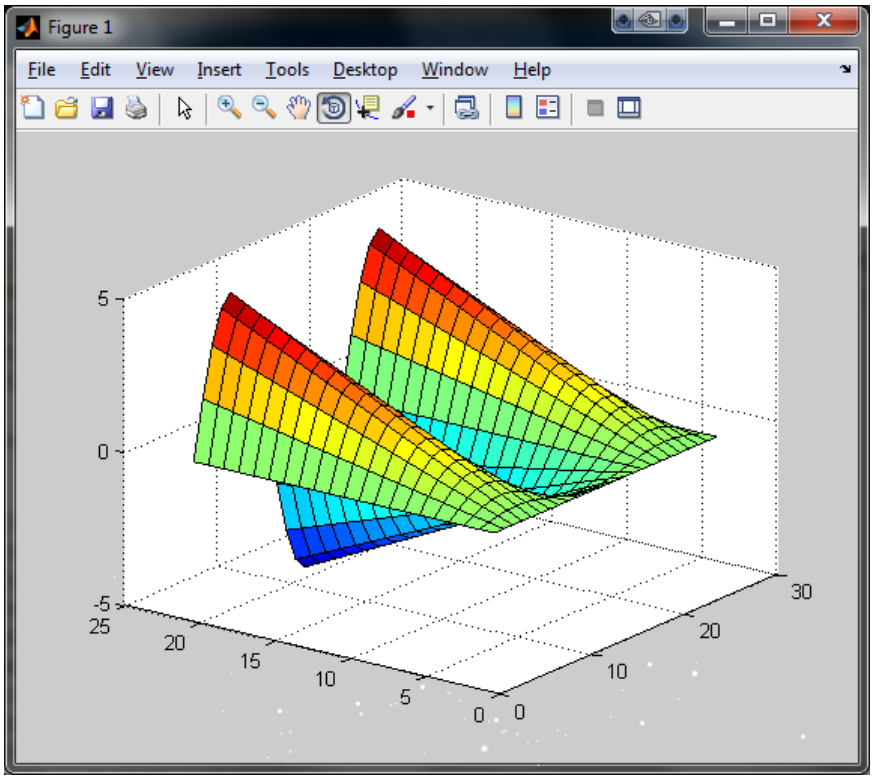
```
0    1.8177    3.3588    4.3884    4.7500    4.3884    3.3588
0    1.9134    3.5355    4.6194    5.0000    4.6194    3.5355

  Columns 8 through 14
...
  Columns 15 through 21
...
  Columns 22 through 25
...
4.3884    3.3588    1.8177    0.0000
4.6194    3.5355    1.9134    0.0000
```

Alternatively, we can show the resulting function using command:

```
surf(c)
```

Resulting figure:



Now we need to store the variable into the file. We can store only the variable "c" using command:

```
save ('myMat', 'c')
```

This command will save variable "c" into the file "myMat.mat". For the further operation, localize the full path of the "myMat.mat" file. You will need something like "C:\....\...\myMat.mat". This value will be referred as *full file name*.

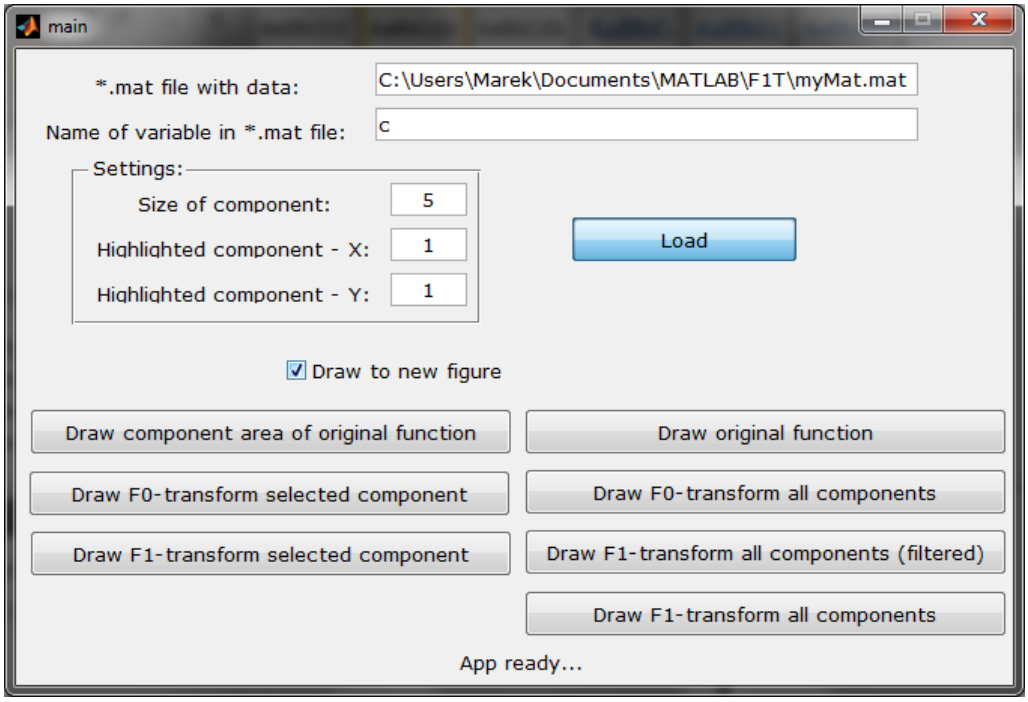
Step 2 – loading file into the application

At the beginning, we need to start the application using command:

```
main
```

The main window of the application is shown.

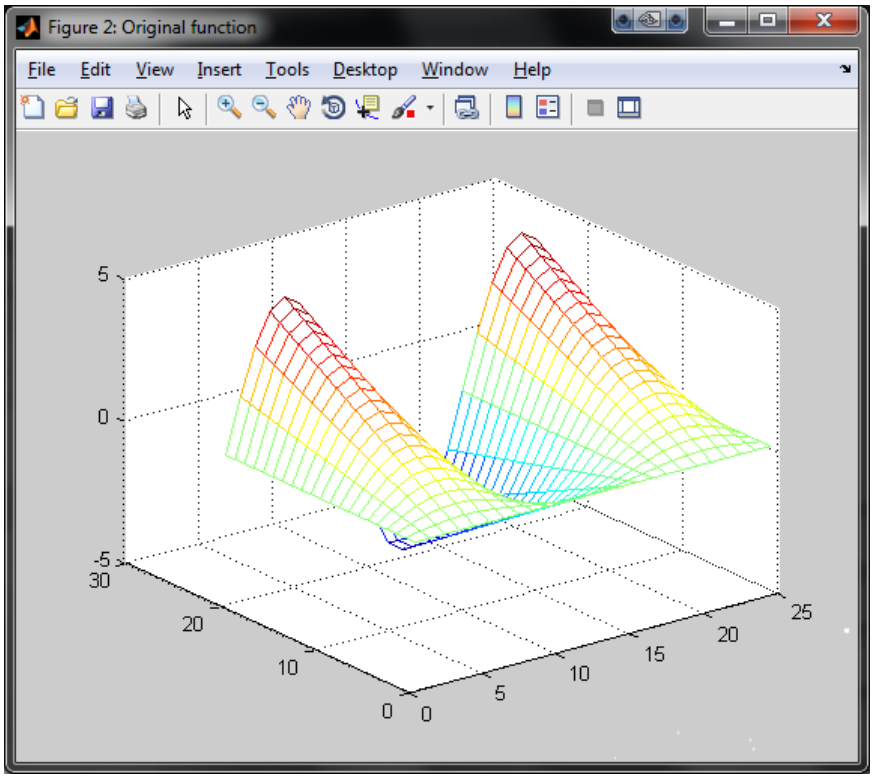
1. Enter the *full file name* into the first text field (**.mat file with data*).
2. Enter the name of the variable "c" into the second text field (*Name of variable in *.mat file)*.
3. Adjust number of components using *Size of component* field set to 5.
4. Press the *Load* button.



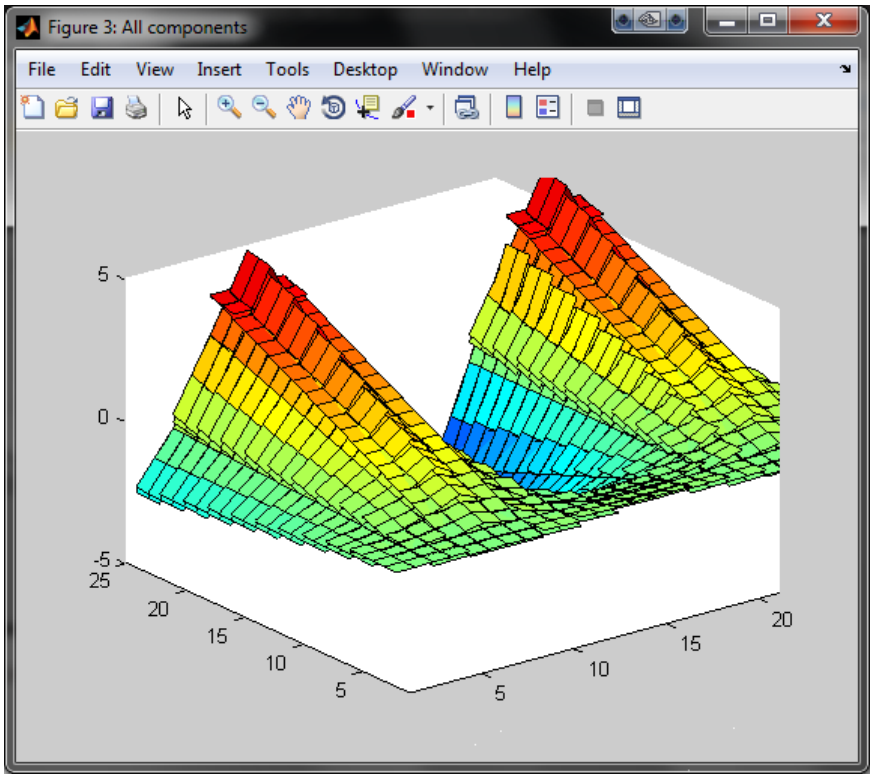
If everything loading is done successfully, the last line of the window will change from *Static Text* to *Loaded and ready*. Otherwise some error will be shown.

Step 3 – printing required figures A

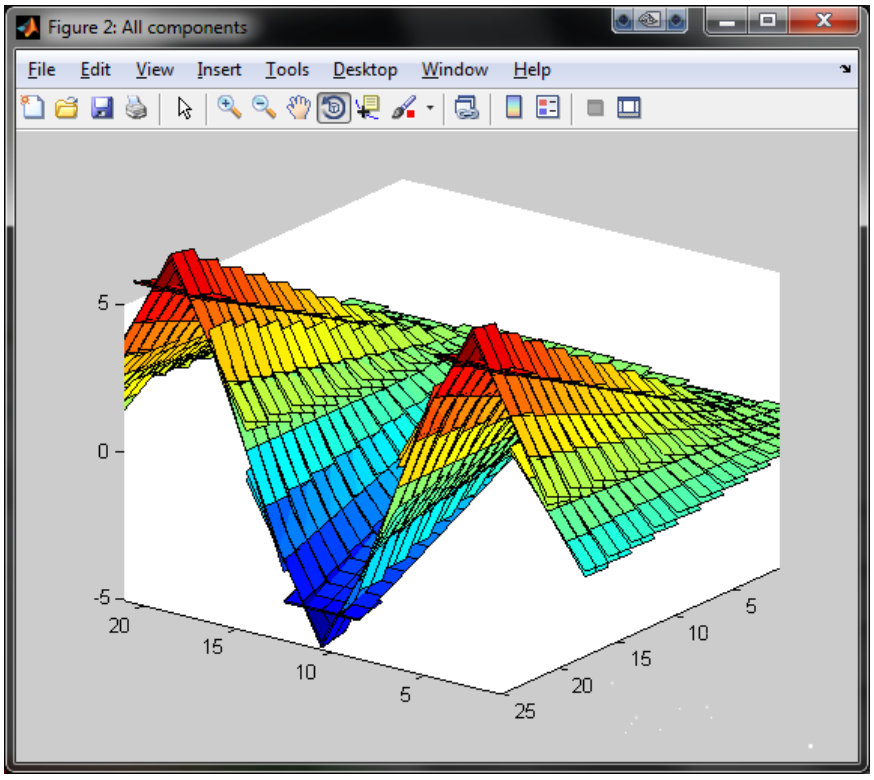
If you would like to print the original function, just press *Draw original function* button. The result will be:



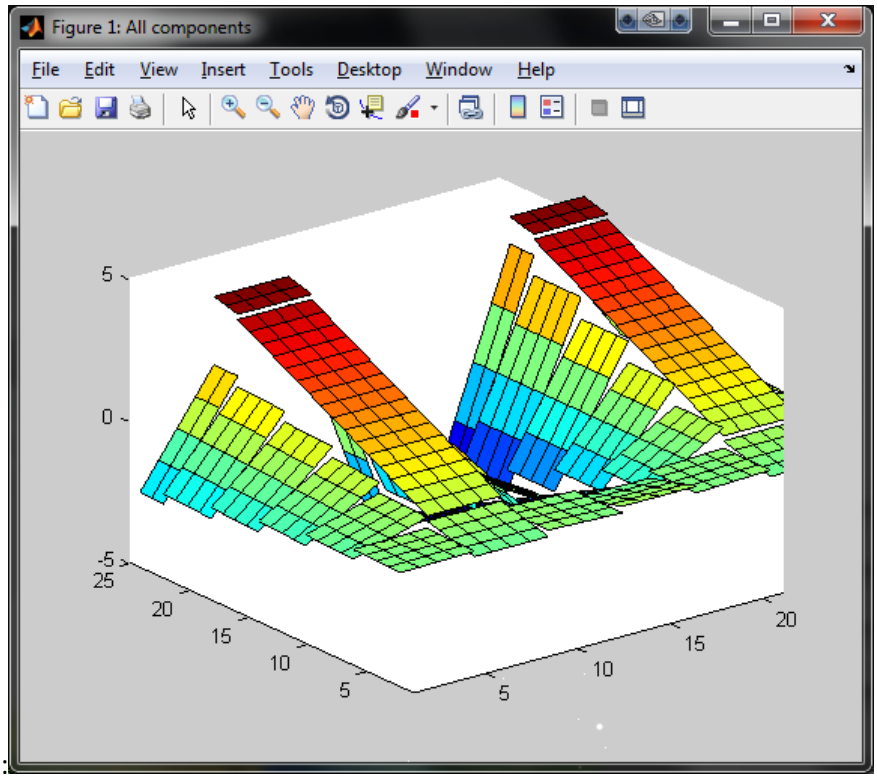
If you would like to print all F^1-components, press *Draw F1-transform all components* button. The result:



, and after rotation:

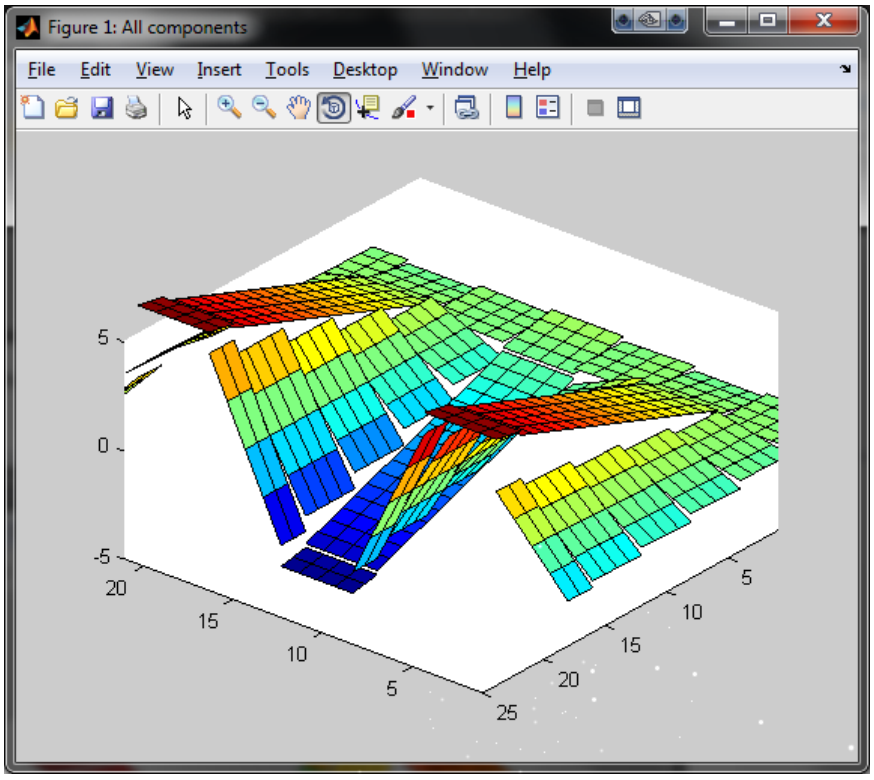


We can also print filtered F^1 components using *Draw F1-transform all components (filtered)* button with the result:



R:

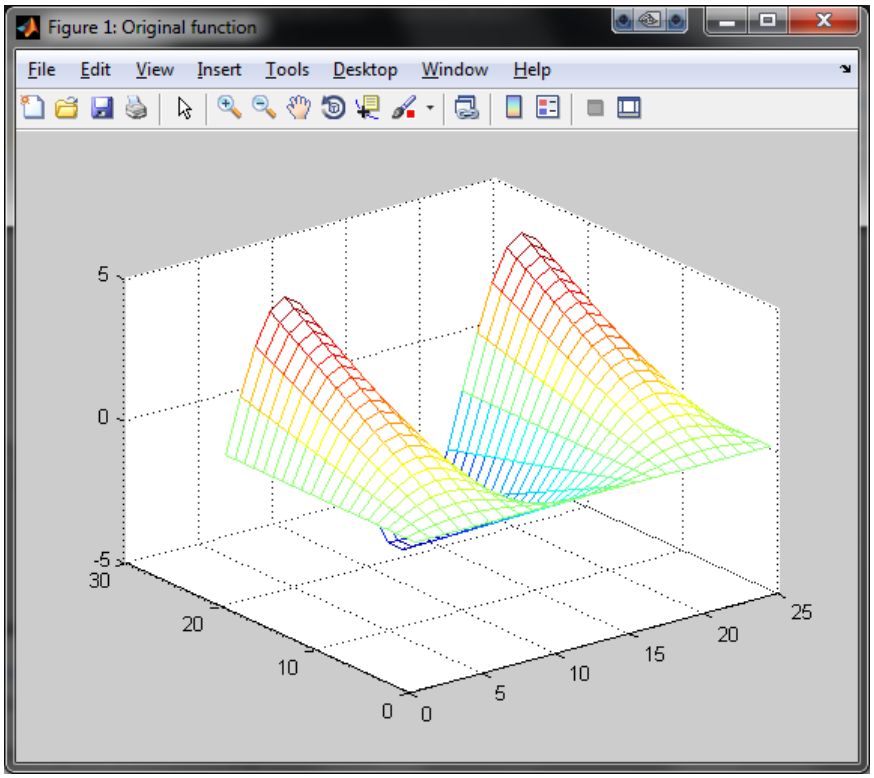
Here, every second component is drawn. Therefore the components creates visually one big separated “plate” (after rotation):



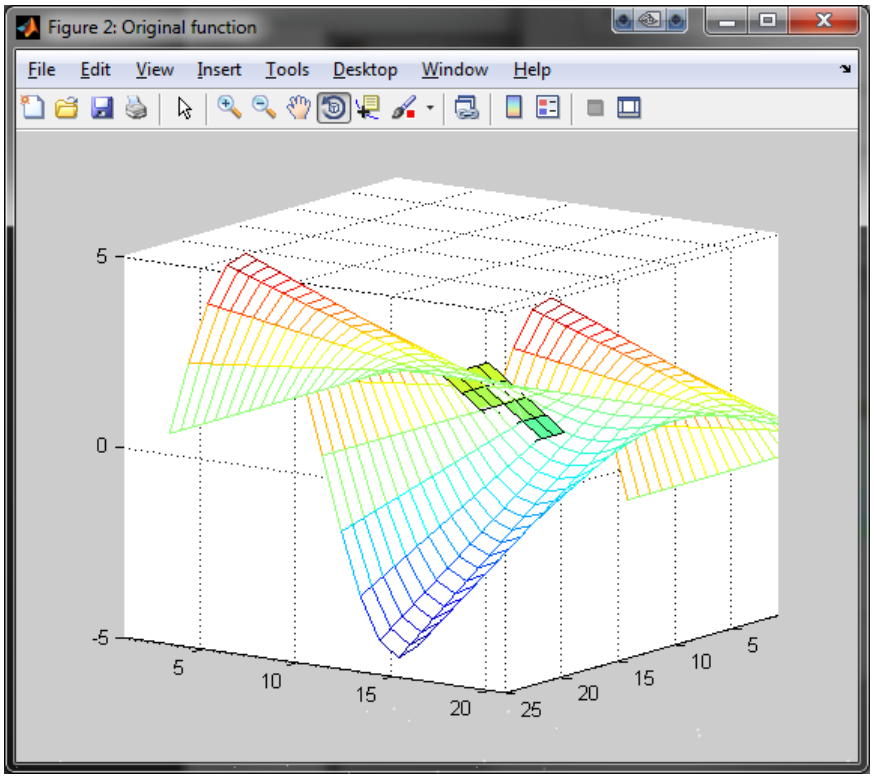
Step 3 – printing required figures B

Now we would like to print only one selected component at coordinate [3,5] in the original function.

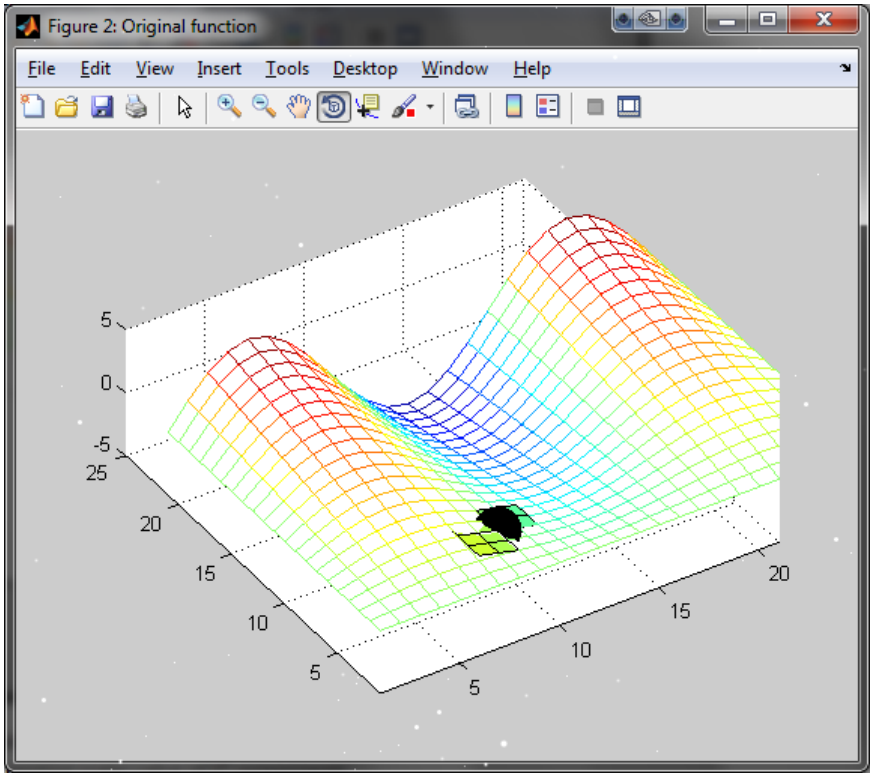
1. Ensure you have option *Draw to new figure* checked.
2. Set *Highlighted component – X* to 3 and *Highlighted component – Y* to 5.
3. Press *Draw original function*. The result will be:



4. Now let's add F^1 component. Uncheck *Draw to new figure*.
5. After that, press *Draw F1-Transform selected component*. You will obtain (after rotation) filled area inside the original figure. Filled area represents selected component.



6. Finally, we will add F^0 component. Press *Draw F0-transform selected component*. You will obtain dot representing scalar value of F^0 component:



Created figure can be processed using typical Matlab approaches – saved to file, changed using properties, etc.

Other information

According to continuous research in this area, the newer version of this or similar software may be available. For any further information or comments contact IRAFM staff.

The research has been supported by the European Regional Development Fund in the IT4Innovations Centre of Excelence project (CZ.1.05/1.1.00/02.0070) and SGS18/PrF/2014 (Research of the F-transform method and applications in image processing based on soft computing) project.

Contact

Institute for Research and Applications of Fuzzy Modeling

University of Ostrava in Ostrava

30. dubna 22

701 03 Ostrava 1

Czech Republic

http://irafm.osu.cz