E7: Introduction to Computer Programming for Scientists and Engineers
University of California at Berkeley, Spring 2017
Instructor: Lucas A. J. Bastien

Lab Assignment 06: Data Visualization; Complexity of Algorithms
Version: release

Due date: Friday March 3rd 2017 at 12 pm (noon).

General instructions, guidelines, and comments:

- For each question, you will have to write and submit one or more Matlab functions. We provide a number of test cases that you can use to test your function. The fact that your function works for all test cases provided does not guarantee that it will work for all possible test cases relevant to the question. It is your responsibility to test your function thoroughly, to ensure that it will also work in situations not covered by the test cases provided. During the grading process, your function will be evaluated on a number of test cases, some of which are provided here, some of which are not.
- Submit on bCourses one m-file for each function that you have to write. The name of each file must be the name of the corresponding function, with the suffix `.m` appended to it. For example, if the name of the function is `my_function`, the name of the file that you have to submit is `my_function.m`. **Carefully check the name of each file that you submit.** Do not submit any zip file. If you re-submit a file that you have already submitted, bCourses may rename the file by adding a number to the file's name (*e.g.,* rename `my_function.m` into `my_function-01.m`). This behavior is okay and should be handled seamlessly by our grading system. Do not rename the file yourself as a response to this behavior.
- A number of optional Matlab toolboxes can be installed alongside Matlab to give it more functionality. All the functions that you have to write to complete this assignment can, however, be implemented without the use of any optional Matlab toolboxes. We encourage you to not use optional toolboxes to complete this assignment. All functions of the Matlab base installation will be available to our grading system, but functions from optional toolboxes may not. If one of your function uses a function that is not available to our grading system, you will loose all points allocated to the corresponding part of this assignment. To guarantee that you are not using a Matlab function from an optional toolbox that is not available to our grading system, use one or both of the following methods:
  - ◇ Only use functions from the base installation of Matlab.
  - ◇ Make sure that your function works on the computers of the 1109 Etcheverry Hall computer lab. All the functions available on these computers will be available to our grading system.

- For this assignment, the required submissions are:
  - ◇ `my_big_o_notation.m`
  - ◇ `my_degree_trends.m`
  - ◇ `my_plot_3d_data.m`

1. Complexity of Algorithms

In this problem, you will have to determine the complexity of four sample functions by specifying the corresponding big-O notation. More precisely, write a function with the following header:

```
function [answers] = my_big_o_notation()
```

where:

- `answers` is a 1×4 `cell` array whose values are your answers to the four following questions, in the same order. Each answer (*i.e.* each element of the `cell` array `answers`) should be a 1×1 array of class `char` that indicates the letter representing the answer of your choice. This letter **must be in lower case**. For example, if you pick answer (b) for question 1, answer (c) for question 2, answer (a) for question 3, and answer (d) for question 4, then `answers` should be the `cell` array:

  `{'b', 'c', 'a', 'd'}`.

Unless specified otherwise, you can assume that all the input arguments to the functions `my_first_function`, `my_second_function`, `my_third_function`, and `my_fourth_function` (see below) are scalars of class `double` that represent integers greater than zero. The four questions are:

1. What is the time complexity (in big-O notation) of the following function?

```
function [a] = my_first_function(m, n)
a = zeros(m, n);
for i1 = 1:m
    for i2 = 1:n
        a(i1,i2) = 2*i1 + 5*i2;
    end
end
end
```

(a) $\mathcal{O}(\texttt{i1} \times \texttt{i2})$
(b) $\mathcal{O}(2 \times \texttt{i1} + 5 \times \texttt{i2})$
(c) $\mathcal{O}(\texttt{i1} + \texttt{i2})$
(d) $\mathcal{O}(\texttt{m} \times \texttt{n})$
(e) $\mathcal{O}(2 \times \texttt{m} + 5 \times \texttt{n})$
(f) $\mathcal{O}(\texttt{m} + \texttt{n})$

2. What is the time complexity (in big-O notation) of the following function?

```
function [a] = my_second_function(p, q)
a = zeros(p,q);
for i = 0:p-1
    j = 0;
    while j < q
        j = j + 1;
        a(i+1,j) = sqrt(i^2 + j^2);
    end
end
end
```

(a) $\mathcal{O}(\mathtt{p} \times \log(\mathtt{j}))$
(b) $\mathcal{O}(\mathtt{p} \times \log(\mathtt{q}))$
(c) $\mathcal{O}(\mathtt{p} \times \mathtt{q})$
(d) $\mathcal{O}(\mathtt{p}^2 \times \mathtt{q})$
(e) $\mathcal{O}(2 \times \mathtt{p} + \mathtt{q})$
(f) $\mathcal{O}(\mathtt{j} \times \mathtt{p} \times \mathtt{q})$

3. What is the time complexity (in big-O notation) of the following function? You can assume that the input parameter `g` to the function `my_third_function` is a function handle to a function that takes a scalar of class `double` as its only input argument, and that executes in square time with the magnitude of its input (*i.e.* the time complexity of `g(x)` is $\mathcal{O}(\mathtt{x}^2)$).

```
function [] = my_third_function(g, n, m)
for i = 1:n
    g(n)*m
end
end
```

(a) $\mathcal{O}(\mathtt{n})$
(b) $\mathcal{O}(\mathtt{n}^2)$
(c) $\mathcal{O}(\mathtt{n}^3)$
(d) $\mathcal{O}(\mathtt{m} \times \mathtt{n})$
(e) $\mathcal{O}(\mathtt{m} \times \mathtt{n}^2)$
(f) $\mathcal{O}(\mathtt{m} \times \mathtt{n}^3)$

4. What is the time complexity (in big-O notation) of the following function?

```
function [a] = my_fourth_function(n, m)
a = 0;
for var = 1:m^2
    x = abs(n);
    while 2*pi > pi
        x = x / 10;
        if x < 1
            break
        end
    end
end
end
```

(a) $\mathcal{O}(n \times m)$
(b) $\mathcal{O}(n \times m^2)$
(c) $\mathcal{O}(\log(n) \times m)$
(d) $\mathcal{O}(\log(n) \times m^2)$
(e) $\mathcal{O}(\log(n \times m))$
(f) $\mathcal{O}(\log(n \times m^2))$

2. Degree trends

In this question, you will write a function that creates a figure showing the number of degrees awarded by different UC Berkeley undergraduate "Units" (Units can be Schools, Colleges, and Divisions). More precisely, write a function with the following header:

```
function [] = my_degree_trends(how, to_plot)
```

where:

- `how` is a character string (*i.e.* a row vector of class `char`) that can take one the following two values:
  - ◇ `'numbers'`. In this case, the figure should show the number of degrees awarded versus time in year (separately for each Unit). For example, if the School of Engineering awarded 200, 300, and 400 degrees in year 1, year 2, and year 3, respectively, the figure should show a line connecting the points of coordinates (1, 200), (2, 300), and (3, 400), in this order.
  - ◇ `'change'`. In this case, the figure should show the percent change of numbers of degrees awarded between the previous year and the current year, versus time in year (separately for each Unit). This quantity can be calculated using the following formula:

$$100 \times \frac{(\text{degrees awarded in the current year}) - (\text{degrees awarded in the past year})}{(\text{degrees awarded in the past year})}$$

- `to_plot` is a $1 \times n$ cell array where each element is a character string (*i.e.* a row vector of class `char`) that represents the abbreviated name of a UC Berkeley Unit. The list of possible abbreviated names is shown in Table 1.

Your function should not return any output. Rather, it should create a Matlab figure that follows the guidelines and instructions described below:

1. Your function should load the `mat` file named `degrees_awarded.mat` (available on bCourses) by using the command: `load degrees_awarded.mat`. This command will create a 1×1 `struct` array named `degrees_awarded` in your function's workspace. One of the fields in this `struct` array is named `years`. Additionally, the `struct` array has one field for each of the UC Berkeley Units listed in Table 1; the names of these fields are the abbreviated names of the corresponding Units. The values of all the fields of the `struct` array are 1×10 arrays of class `double`. The field `years` represents each year from 2007 to 2016 (both of these years included) and the other fields represent the numbers of degrees awarded in each of these years (in the same order) by the corresponding Unit. You can load this `struct` array in the command window's workspace if you want to explore its contents.

   The file `degrees_awarded.mat` should be somewhere where Matlab can find it (*e.g.,* in your working directory) when you try to load it. This file will be available to our grading system and you do not need to submit it for this lab assignment. Do not rename this file: the only `mat` file that our grading system will have access to when grading your `my_degree_trends` function will be the file named `degrees_awarded.mat`.

   Loading this `mat` file will also create a variable named `origin_of_the_data` in the workspace where it is loaded. This variable will be a character string (*i.e.* a row vector of class `char`) that describes the origin of the data provided in the `mat` file. You can ignore this variable for this lab assignment.
2. Use Matlab's built-in function `plot` to plot the numbers of degrees awarded versus time in years for years 2007 through 2016 (if `how` is `'numbers'`) or to plot the percent change of numbers of degrees awarded between the previous year and the current year versus time in years for years 2008 through 2016 (if `how` is `'change'`). Note that the percent change data only span years 2008 through 2016 because calculating the percent change for a given year requires the knowledge of the numbers of degrees awarded during the previous year. The figure should show the data for each of the Units listed in the input parameter `to_plot`, using a solid line and a different symbol (of your choice) for each of them. The choice of colors is up to you (you can let Matlab use its default colors).
3. If an element of `to_plot` is not one of the abbreviated names listed in Table 1 (the comparisons should be case-sensitive), your function should ignore it. You can assume that at least one of the elements of `to_plot` will be one of the abbreviated names listed in Table 1, and that all elements of `to_plot` will be different from each other.
4. The x-axis should be labeled "Year" (without the quotes) with a font size of 20.
5. If `how` is `'numbers'`, the y-axis should be labeled "Number of degrees" (without the quotes). If `how` is `'change'`, the y-axis should be labeled "Annual percent change" (with-

out the quotes). Use a font size of 20 in either case.

6. If `how` is `'numbers'`, the title of the figure should be "Number of degrees awarded per year" (without the quotes). If `how` is `'change'`, the title of the figure should be labeled "Change in number of degrees awarded" (without the quotes). Use a font size of 20 in either case.
7. Axis labels and titles can span one or multiple lines (your choice).
8. Let Matlab set the range of the x- and y-axes automatically.
9. Grid lines should be shown on the figure.
10. The figure should include a legend box that indicates which dataset is represented by each line on the figure. Use the abbreviated names in the legend box. If the abbreviated names to display in the legend box contain underscores, Matlab will not show the underscores, but will format the characters directly following the underscores as subscripts. To prevent this issue from happening, you can replace all the underscores in the abbreviated names to display by `\_` (*i.e.* a backslash followed by an underscore). If the variable `s` is a character string (*i.e.* a row vector of class `char`), you can replace all the underscores in `s` by `\_` using the following command:

```
s = replace(s, '_', '\_')
```

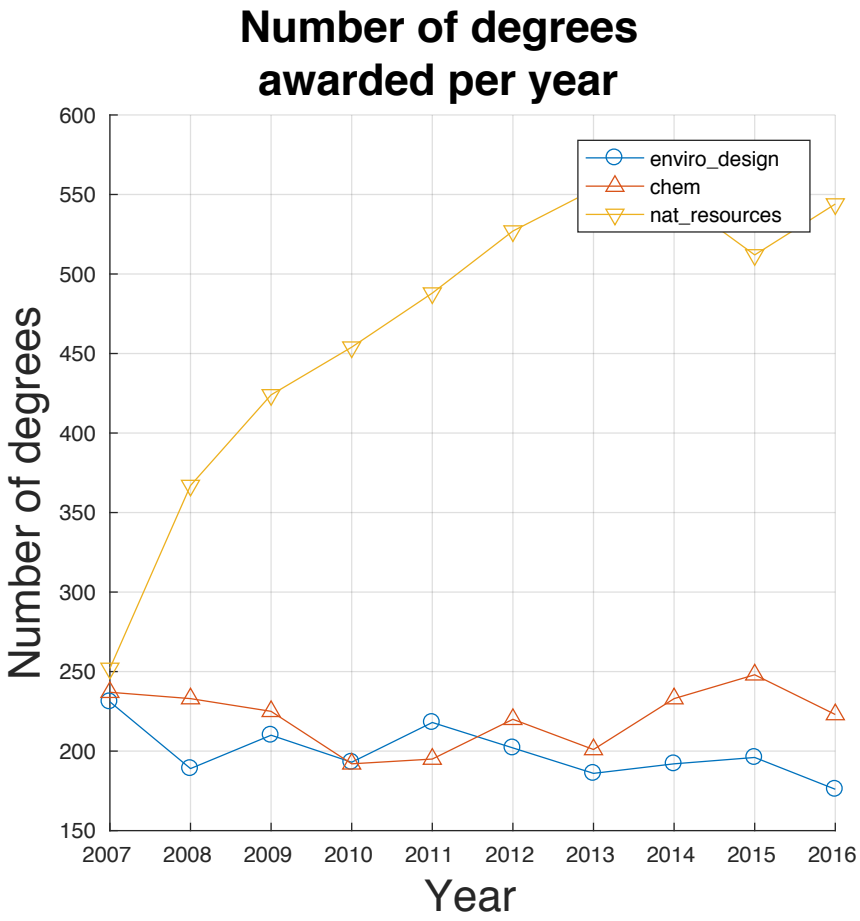
The data provided in the file `degrees_awarded.mat` were retrieved on February 23rd 2017 from the website of the Office of Planning and Analysis of the University of California at Berkeley, at the following URL: http://opa.berkeley.edu/degrees_awarded. In this `mat` file, each number in the struct array named `degrees_awarded` has been rounded to the nearest integer (some numbers in the original data had a non-zero decimal part). Note that here, degrees awarded in the academic year 2006–2007 are counted as awarded in the year 2007. As a consequence, degrees awarded in the fall of 2006 are counted as awarded in year 2007. A similar approximation is used for each year covered by these data.

Table 1: List of UC Berkeley "Units" (Units can be Schools, Colleges, and Divisions). The right column shows the full names, and the left column shows the corresponding abbreviated names. The list of Units was retrieved on February 23rd 2017 from the website of the Office of Planning and Analysis of the University of California at Berkeley, at the following URL: http://opa.berkeley.edu/degrees_awarded

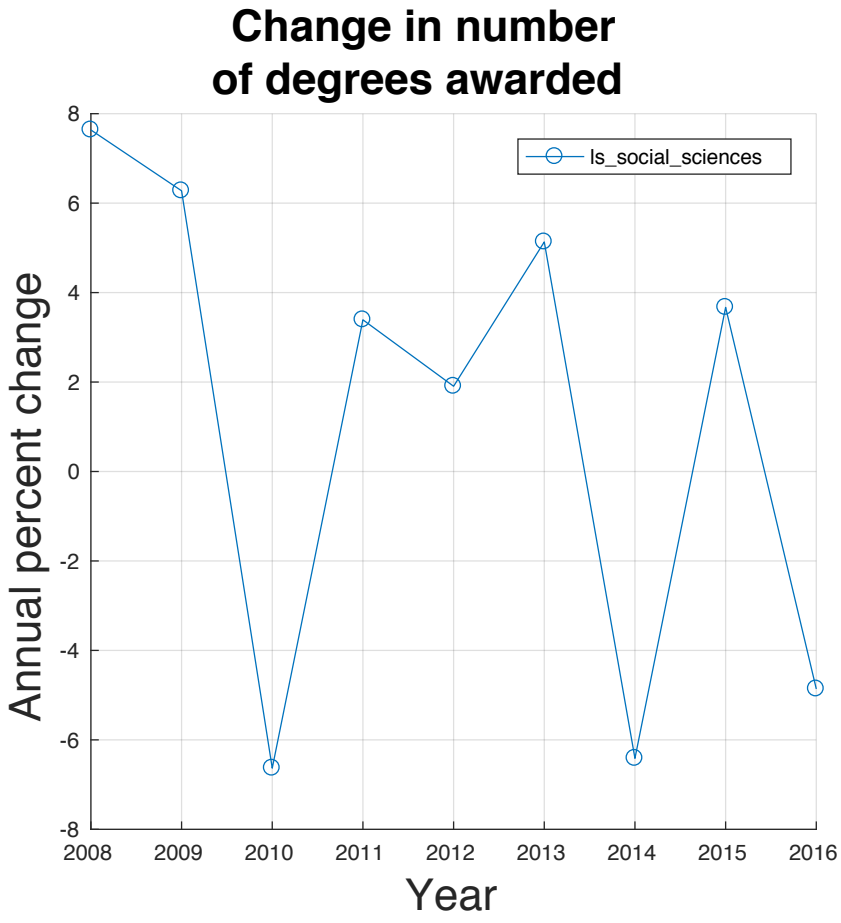
Abbreviated name	Full name
chem	College of Chemistry
eng	College of Engineering
enviro_design	College of Environmental Design
ls_admin_programs	College of Letters and Sciences, Administered Programs
ls_arts_humanities	College of Letters and Sciences, Arts and Humanities Division
ls_bio_sciences	College of Letters and Sciences, Biological Sciences Division
ls_math_physics	College of Letters and Sciences, Mathematical and Physical Sciences Division
ls_social_sciences	College of Letters and Sciences, Social Sciences Division
ls_undergrad_studies	College of Letters and Sciences, Undergraduate Studies Division
nat_resources	College of Natural Resources
business	Haas School of Business

Test cases:

```
>> my_degree_trends('numbers', {'enviro_design', 'chem', 'nat_resources'})
```



```
>> my_degree_trends('change', {'ls_social_sciences', '-'})
```



3. Plotting 3-dimensional data

Consider a real-valued mathematical function defined on $\mathbb{R}^2$:

$$\begin{aligned} f : \mathbb{R}^2 &\to \mathbb{R} \\ x, y &\mapsto f(x, y) \end{aligned}$$

For example:

$$\begin{aligned} f : \mathbb{R}^2 &\to \mathbb{R} \\ x, y &\mapsto \sqrt{x^2 + y^2} \end{aligned}$$

In this question, you will explore a variety of methods that can be used to plot 3-dimensional data such as represented by mathematical functions of the form defined above. More precisely, write a function with the following header:

```
function [] = my_plot_3d_data(fh, xmin, xmax, ymin, ymax, n, xslice, yslice)
```

where:

- `fh` is a function handle to a function that takes two $m \times n$ arrays of class `double` as input arguments (let us call these arrays `x` and `y`, respectively), and returns a $m \times n$ array of class `double` (let us call this array `z`). This function handle represents a real-valued mathematical function f defined on $\mathbb{R}^2$ (such as defined above), such that

  $$\texttt{z(i,j)} = f(\texttt{x(i,j), y(i,j)})$$

  See the test cases below for examples. The three arrays `x`, `y`, and `z` have the same size and, when considered together, represent 3-dimensional data. In other words, to each point of coordinates (`x(i,j), y(i,j)`), the function handle `fh` associates a value `z(i,j)`.
- `n` is a scalar of class `double` that represents a positive integer. See below for its description.
- `xmin`, `xmax`, `ymin`, `ymax`, `xslice`, and `yslice` are scalars of class `double`. See below for their descriptions.

Your function should not return any output. Rather, it should create a Matlab figure which has four subplots, arranged in a 2×2 grid. The content and format of each subplot is described below.

Subplot 1 (top-left): this subplot should show a surface plot of $z = f(x, y)$ over the domain defined by $x \in$ [`xmin`,`xmax`] and $y \in$ [`ymin`,`ymax`]. For this subplot, use data calculated at n equally spaced points in each of these intervals. Format this subplot as follows:

- Set the title of the subplot to be "Surface plot" (without the quotes).
- Set the labels of the x-, y-, and z-axes to be "x", "y", and "z", respectively (without the quotes).

Subplot 2 (top-right): this subplot should show a filled contour plot of $z = f(x, y)$ over the domain defined by $x \in$ [`xmin`,`xmax`] and $y \in$ [`ymin`,`ymax`]. For this subplot, use data calculated at n equally spaced points in each of these intervals. Format this subplot as follows:

- Set the title of the subplot to be "Filled contour plot" (without the quotes).
- Set the labels of the x- and y-axes to be "x" and "y", respectively (without the quotes).
- Add a colorbar with label "z" (without the quotes) on the right side of the subplot, outside the border.

Subplot 3 (bottom-left): this subplot should show a 2-dimensional representation of the "slice" of the function f at $x =$ `xslice`. In other words, plot the function:

$$f : \mathbb{R} \to \mathbb{R}$$
$$y \mapsto f(x = \text{xslice}, y)$$

This subplot should show data over the domain defined by $y \in$ [`ymin`,`ymax`], using data calculated at n equally spaced points in this interval. Format this subplot as follows:

- Set the title of the subplot to be "Slice at x = ???" (without the quotes), where ??? should be replaced by the value of `xslice`, with three digits after the decimal point.
- Set the labels of the x- and y-axes to be "y" and "z", respectively (without the quotes).
- Display a grid.

Subplot 4 (bottom-right): this subplot should show a 2-dimensional representation of the "slice" of the function f at $y =$ `yslice`. In other words, plot the function:

$$f : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto f(x, y = \text{yslice})$$

This subplot should show data over the domain defined by $x \in$ [`xmin`,`xmax`], using data calculated at n equally spaced points in this interval. Format this subplot as follows:

- Set the title of the subplot to be "Slice at y = ???" (without the quotes), where ??? should be replaced by the value of `yslice`, with three digits after the decimal point.
- Set the labels of the x- and y-axes to be "x" and "z", respectively (without the quotes).
- Display a grid.

In all subplots:

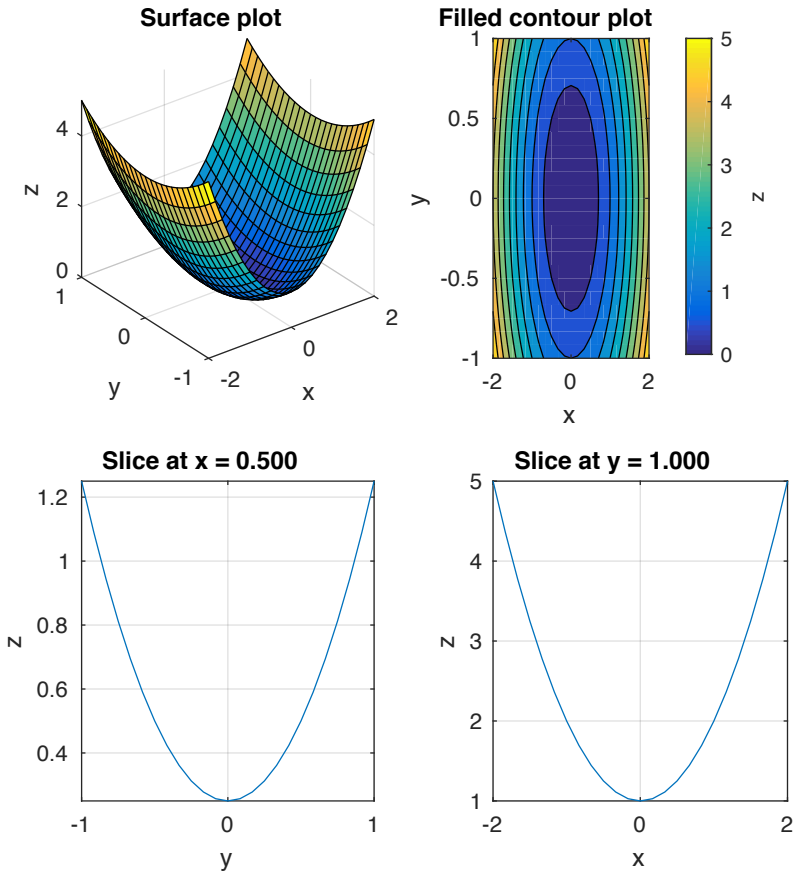
- Set the limits of all the axes such that all the data fits exactly within the corresponding ranges, without extra space.
- Set the font size of all written text so that the text is clearly legible.
- You can choose the colors used to display the plotted elements, provided that these elements are still clearly visible.

Hint: if `a` and `b` are scalars of class `double` such that `a < b`, and if `n` is a scalar of class `double` that represents an integer such that `n>1`, then the command `linspace(a, b, n)` returns a row vector of class `double` that contains `n` equally spaced values from `a` to `b` (these two boundaries included). For example:

```
>> vector = linspace(1, 2, 11)
vector =
  Columns 1 through 7
    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000
  Columns 8 through 11
    1.7000    1.8000    1.9000    2.0000
```

Test cases:

```
>> fh = @(x, y) x.^2 + y.^2;
>> my_plot_3d_data(fh, -2, 2, -1, 1, 25, 0.5, 1)
```



```
>> fh = @(x, y) x.^3+sin(abs(y)+y);
>> my_plot_3d_data(fh, -2, 2, -10, 10, 50, -1.5, 1.5)
```