CME 192: Introduction to MATLAB
Lecture 4

Stanford University

January 24, 2019

Outline

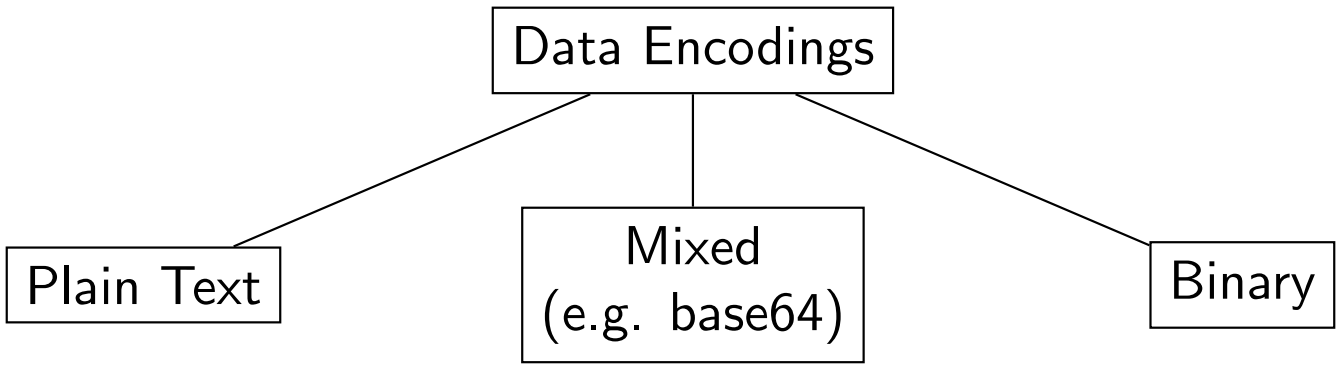
Review

Lecture 3

- Data Structures (Structs)
- Plotting
  - 2D plotting
  - 3D plotting
  - styling by modifying properties

Outline

Fundamentals of Data Encoding



Plain Text Encoding

- used for all code
- uses mostly characters a-z, A-Z, 0-9
- `.txt`, `.m`, `.cpp`, `.csv`, `.py`, `.jl`, etc.

Example 1 (`.m`)

```
function my_abs(x)
  % a helpful comment
  if x < 0.0
    x = -x
  end
end
```

Example 2 (`.csv`)

```
0.000e+00 5.000e+00 0.000e+00
1.000e-04 5.000e+00 -5.000e-04
2.000e-04 5.000e+00 -1.000e-03
3.000e-04 5.000e+00 -1.000e-03
4.000e-04 5.000e+00 -2.000e-03
5.000e-04 4.999e+00 -2.000e-03
6.000e-04 4.999e+00 -3.000e-03
7.000e-04 4.999e+00 -3.000e-03
8.000e-04 4.998e+00 -4.000e-03
9.000e-04 4.998e+00 -4.999e-03
1.000e-03 4.998e+00 -4.999e-03
```

Mixed Encoding

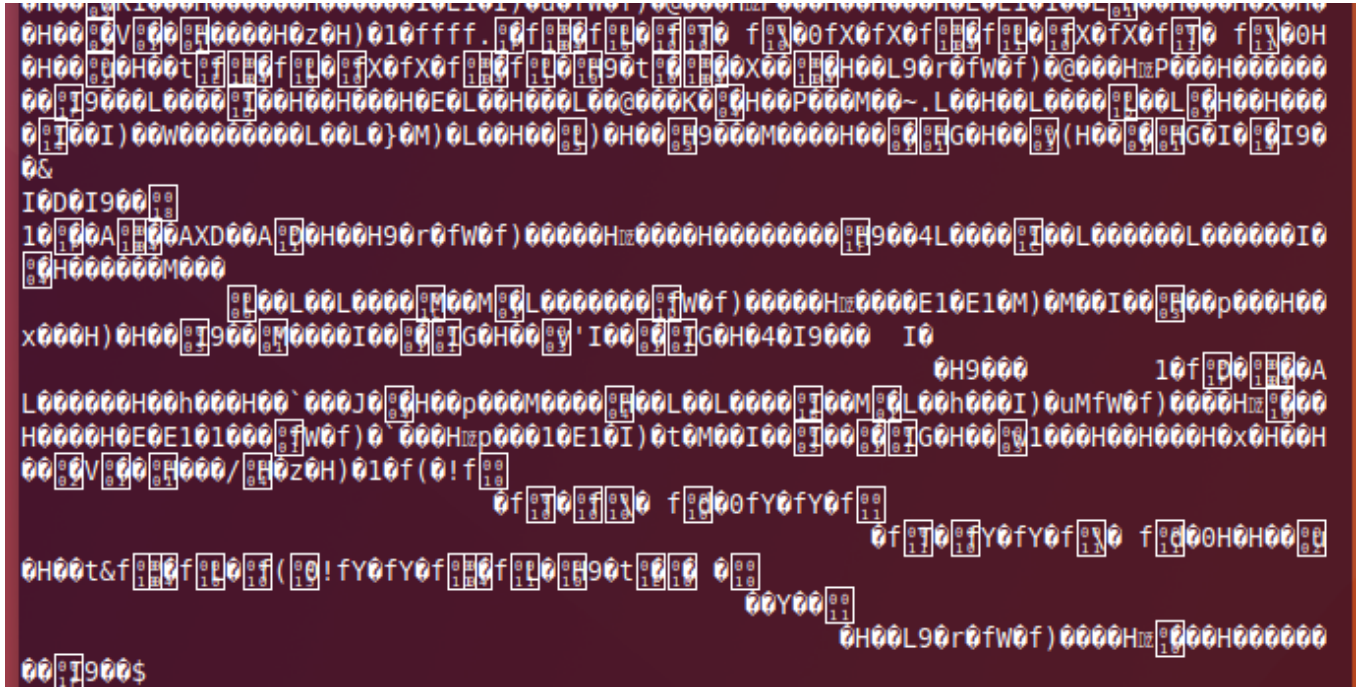
- used for some data compression with reserved symbols
- uses 64 symbols for numbers
- rarely used, mostly in web applications

Man is distinguished, not only by his reason, but by this singular passion from other animals, which is a lust of the mind, that by a perseverance of delight in the continued and indefatigable generation of knowledge, exceeds the short vehemence of any carnal pleasure.

```
TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlz
IHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2Yg
dGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGlu
dWVkIGFuZCBpbmRlZmF0aWdhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRo
ZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4=
```

Binary Encoding

- pure data, very compact
- a sequence of numbers
- has no meaning unless interpreted
- most files stored this way (`.pdf`, `.jpeg`, `.docx`)
- most efficient space use



Outline

Saving and Loading Workspace

- quickest way to save data in Matlab
- only Matlab/Octave compatible
- binary encoding, `.mat`

Saving

```
1 a = 2;
2 b = 2;
3 % to save entire workspace
4 save('my_workspace.mat');
5
6 % to save individual variables
7 % notice strings
8 save('my_workspace.mat', 'a');
```

Loading

```
1 load('my_workspace.mat');
```

Outline

Comma Separated Values

- very common for data storage
- Excel-like
- plain text encoding
- format
  - values in each line separated by commas
  - same number of values per line
  - first line contains descriptions
  - matrix-like

Delimited Files in Scientific Computing

- usually just numbers
- not necessarily comma delimited
- typical delimiters: comma, space, tab, semicolon

```
0.000000e+00 5.000000e+00 0.000000e+00
1.000000e-04 5.000000e+00 -5.000000e-04
2.000000e-04 5.000000e+00 -1.000000e-03
3.000000e-04 5.000000e+00 -1.500000e-03
4.000000e-04 5.000000e+00 -2.000000e-03
5.000000e-04 4.999999e+00 -2.500000e-03
6.000000e-04 4.999999e+00 -3.000000e-03
7.000000e-04 4.999999e+00 -3.500000e-03
8.000000e-04 4.999998e+00 -4.000000e-03
9.000000e-04 4.999998e+00 -4.499999e-03
1.000000e-03 4.999998e+00 -4.999999e-03
1.100000e-03 4.999997e+00 -5.499999e-03
1.200000e-03 4.999996e+00 -5.999999e-03
1.300000e-03 4.999996e+00 -6.499998e-03
1.400000e-03 4.999995e+00 -6.999998e-03
```

Delimited Files in Matlab

Reading

```
% automatically detect delimiter
D = dlmread('data.txt');

% delimiter as second argument
D = dlmread('data.txt', '\t');
```

Writing

```
% usually #rows >> #columns
A = rand(5000, 5);
dlmwrite('data.txt', A);

% choosing the delimiter
dlmwrite('data.txt', A, 'delimiter', '\t');
```

Outline

Custom Writing

- plain text
- make use of `fprintf`
- specify printing to file instead of console
- `<file_id> = fopen(<filename>, <permission>)`
- `fclose(<file_id>)`

```
% open the file for writing
% 'r' - reading
% 'w' - writing
% 'a' - appending (writing)
mol = 42;
fid = fopen('data.txt', 'w');
fprintf(fid, 'The Meaning of life is %i\n', mol);
fclose(fid);

A = rand(50, 5);
fid = fopen('data.txt', 'w');
fprintf(fid, '%f %f %f %f %f', A);
fprintf(fid, '\n');
fprintf(fid, '%f %f %f %f %f', A);
fclose(fid);
```

Custom Reading

- plain text
- make use of `textscan(<file_id>, <format_string>)`

```
String 42 23.0 c
Name 2 -3.0 f
Word -22312 17.0 h
Characters 00234 Inf z
```

```
1 % open the file for reading
2 fid = fopen('data.txt', 'r');
3 data = textscan(fid, '%s %d %f %s');
4 fclose(fid);
```

Outline

Java Script Object Notation

- popular format for representing data structures
- structured same as data structure in Matlab
- `<json_string> = fileread(<filename>)`
- `<my_struct> = jsondecode(<json_string>)`
- support in Octave only through external libraries
- plain text

```
{"widget": {
    "debug": "on",
    "window": {
        "name": "main_window",
        "width": 500,
        "height": 500
    },
    "image": {
        "src": "Images/Sun.png",
        "alignment": "center"
    },
    "text": {
        "data": "Click Here",
        "alignment": "center",
        "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
}}
```

Outline

Interpolation

- way of filling in missing spots in the data
- `<new_y> = interp1(<data_x>, <data_y>, <new_x>)`
- can be 1D, 2D, 3D, ...
- various methods: linear, nearest, cubic, spline
- not magic, but very helpful

Basic Filtering

- way to smooth noisy data
- signal processing a field on its own
- a digital filter can be described by
  - `a` coefficient vector
  - `b` coefficient vector
- for moving average (smoothing)
  - `a = 1`
  - `b = [1/N, 1/N, ..., 1/N]`
- `<smooth_y> = filtfilt(b, a, <data_y>)`
- not magic, but very helpful

Polynomial Function Approximation

- function approximation
- much easier to store a vector of coefficients than data
- easy calculus (integration, differentiation) on polynomials
- Taylor Series says that every function has a polynomial expansion
- `<p_coeff> = polyfit(<data_x>, <data_y>, <p_order>)`
- not magic, but very helpful