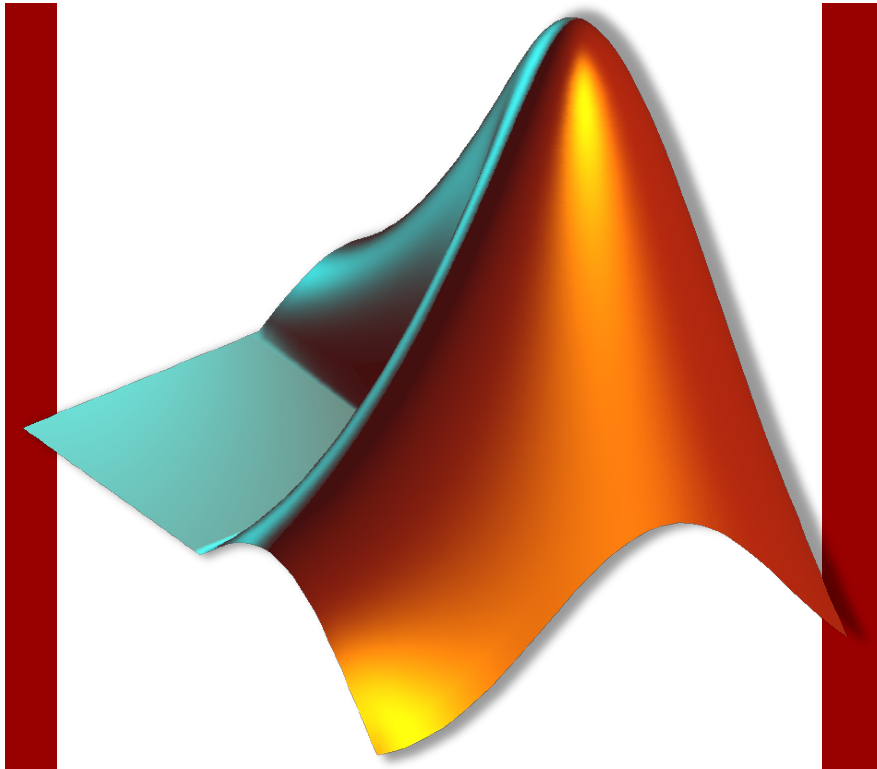


MATLAB®

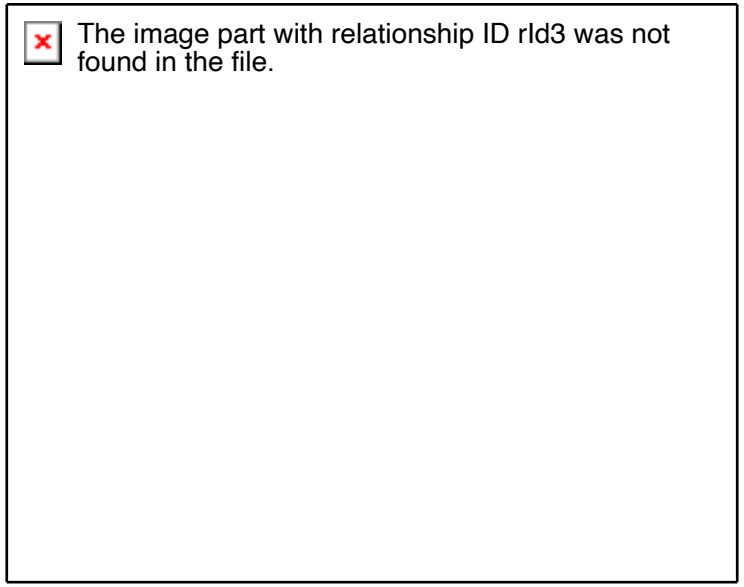
The Language of Technical Computing

Audio Signals

Create a simple signal

- Let's try to create a sawtooth:
  - https://it.mathworks.com/help/signal/ref/sawtooth.html

```
t = [0:.1:2*pi];
x = sawtooth(2*t);
```



Plot a signal

- Let's try to plot a signal
  - https://it.mathworks.com/help/matlab/ref/plot.html

    plot(x, y)

    plot(x, y, lineSpec)

Add a legend to the plot

- Let's try to add a legend
  - https://it.mathworks.com/help/matlab/ref/legend.html

    legend

    legend(label1,...,labelN)

Create another signal

- Let's try to create a sin:
  - https://it.mathworks.com/help/matlab/ref/sin.html

```
t = [0:.01:2*pi];
x = sin(t);
plot(x,t);
```

Plot two signals

```
t = [0:.01:4*pi];
x = sawtooth(2*t);

u = [0:.01:4*pi];
y = sin(u);

plot(t,x, '--', u, y, ':')
legend('Sawtooth','Sin','Location','NorthOutside');
```

DPCM Encode

- Differential pulse code modulation
- *dpcmenco* function
  - https://it.mathworks.com/help/comm/ref/dpcmenco.html

```
% Quantize x using DPCM
% codebook prescribes a value for each partition in the quantization
% partition is a vector whose entries give the endpoints of the partition intervals
% predictor specifies the predictive transfer function.
encodedx = dpcmenco(x, codebook, partition, predictor);
```

DPCM Decode

- Differential pulse code modulation
- *dpcmdeco* function
  - https://it.mathworks.com/help/comm/ref/dpcmdeco.html

```
decodedx = dpcmdeco(encodedx,codebook,predictor);
```

Exercise 1

- Using a y(k)=x(k-1) predictor, try to:
  - Create a sawtooth signal
  - Quantize it using DPCM
  - Recover it from the modulated signal
  - Plot the sawtooth and the recovered signal, with a legend
  - Calculate and print the mean square error

$$e = \frac{\sum(x - \bar{x})^2}{len(x)}$$

Exercise 2

- Using the dpcmopt function modify Exercise 1 to obtain optimized differential pulse code parameters
  - https://it.mathworks.com/help/comm/ref/dpcmopt.html

- Hint: use an initial codebook

```
initcodebook = [-1:.1:1];
```