Convolution of Signals

Practical seminar on 'Signals and Systems 1'

Gerhard-Mercator-Universität Duisburg
Nachrichtentechnische Systeme

http://fb9nt.uni-duisburg.de

Overview

- 1 Introduction
- 2 Basic Language Elements and Concepts
- 3 Handling of Matrices
- 4 Plotting
- 5 Flow control
- 6 Other frequently used Functions
- 7 Conclusion

1 Introduction

- **Matlab is a language suitable for:**
  - General technical computing
  - Mathematical and engineering applications
  - Development of algorithms
  - Modeling and simulation
  - Scientific and engineering graphical analysis
- **Main features**
  - Interactive programming system
  - Basis is matrix computation
  - Comprehensive set of toolboxes
  - Availability of many textbooks on Matlab excercises
  - Availability of a student version

2 Basic Language Elements and Concepts

- **Variables**
  - No declarations or storage specification needed
  - Variables are case sensitive and can address an entire matrix
- **Number representation**
  - Conventional decimal notation
  - Scientific notation using the letter ‘e’. Example: -12.34e+05
  - Letter ‘i’ or ‘j’ for imaginary numbers. Example: 1+5j
- **Operators**
  - All conventional operators (including ^ for power operations)
  - Symbol ‘ for transform of row to line vectors and vice versa
- **Functions**
  - Comprehensive set of advanced mathematical functions
  - Complex arguments allowed in many cases

3 Handling of Matrices I

- **Entering matrices by specification of all elements**
  - Separation of elements using the blank character
  - Separations of rows using a semicolon
  - Example: A=[1 2 3;4 5 6]
- **Addressing matrix elements**
  - Usual form for single elements: A(i,j)
  - Sub-matrix addressing using colons: A(k:l,m:n)
  - Addressing of a whole row or a whole line: A(i,:) or A(:,i)
- **Generation of matrix elements**
  - Linearly spaced elements: A=linspace(0,2*pi,101)
  - Second possibility: A=(0:1/100:1)*2*pi
  - Logarithmically spaced elements: A=logspace(-2,0,101)* 2*pi

3 Handling of Matrices II

- **Generation of special matrices** (Dimensions: N x M)
  - Matrix containing only zeros: A=zeros(N,M)
  - Matrix containing only ones: A=ones(N,M)
  - Matrix with uniformly distributed elements: A=rand(N,M)
  - Matrix with Gaussian (normally) distributed elements: A=randn(N,M)
- **Concatenation and deletion of matrices**
  - Specification of entire matrix as part of another: C=[A B]
  - Deletion operation for a row or a column: A(i,:)=[]
- **Rearranging a matrix**
  - Usage of transpose operator: B=A'
  - Thereby rows are converted to columns and vice versa

3 Handling of Matrices III

- **Scalar-array mathematics**
  - Matrices can be scaled in a simple manner. Example: B=2*A-1
- **Element-by-element operations**
  - Usual notation for addition and substraction: D=A+B-C
  - Dot usage for multiplication and division: D=A.*B./C
  - Dot usage for specifying the squaring operation: B=A.^2
  - Example for application of functions: B=sin(A)
- **Notation for normal matrix operations**
  - No dot is used here: C=A*B
  - Essential restriction: Number of columns and rows must match

4 Plotting

- **Typical sequence for line plotting:**

```
t=0:0.001:1        %Definition range set for x from 0 to 1
plot (t,sin(2*pi*t))    %Plots one period of sine function
zoom xon
grid on
title('signal s(t)')
xlabel('Time t')
```

- **Other plotting control elements**
  - Subplots are possible using the subplot() command
  - Forcing the scale can be performed using the axes() command
- **Additional plotting tools**
  - Hidden line and surface plots
  - Special colouring of surface plots and line plots is possible

5 Flow control

- **Usual set of flow control commands just as in other languages**
- **Example for the notation of the ‘for-loop’:**

```
for i=1:10
  command1
  command2
end
```

- **Notation for the ‘while-loop’:** Similar to ‘for-loop’
- **Notation for the ‘if-then-else’ structure:**

```
if expression
  command
else
  command
end
```

6 Other frequently used Functions

- **Semicolon for display suppression**
- **Character ‘%’ for commenting in line**
- **min() and max() operations work on an entire vector or matrix**
- **length() function determines numbers of vector elements**
- **FFT and inverse FFT operations are available**
- **abs() and angle() functions can be used on complex values**
- **real() and imag() functions are also available**
- **Comprehensive help function gives detailed description**
- **This description explains the use, the applied algorithm and includes examples**

7 Conclusion

- Matlab exhibits elements of interpreter-type language
- Use of matrix notation needs some training
- Powerful set of mathematical and engineering functions
- High degree of support available in the web
- This results in very efficient handling of engineering and mathematical problems!