Fondamenti di Informatica

Getting started with Matlab and basic concept of programming

Prof. Emiliano Casalicchio
http://www.ce.uniroma2.it/courses/FOI/
Avvisi

- L’esame sarà in lingua italiana
- L’esame consisterà in
  - una prova scritta
    - aspetti metodologici e teorici del corso
  - una prova al calcolatore
    - realizzazione di un programma (funzionante) in matlab
- La sufficienza alla prova scritta è condizione necessaria allo svolgimento della prova al calcolatore
- L’esame sarà superato se e solo se lo studente ha raggiunto la sufficienza ad entrambe le prove
- Il voto finale sarà dato dalla somma dei punteggi conseguiti alla prova scritta e orale
Matrix Laboratory

- create, manipulate and visualize matrices
- rectangular array of numerical value

\[
\begin{bmatrix}
3 & 2 \\
1 & 0 \\
\end{bmatrix}
\quad \begin{bmatrix}
23 & 27 & 30 & \ldots
\end{bmatrix}
\]

what I can do with a matrix?
Examples

- $Ax=b \Rightarrow x=A^{-1}b$

- $A=[1 \ 2;3 \ 4]; \ b=[1;5]; \ x=\text{inv}(A)*b;$

- Image processing

  - read the example “Load Integer Image Data” available at

  - Download the images street1 and street2 from the website and repeat, by yourself, the sequence of commands
Objectives - Programming Languages

The fundamentals of programming include:

- How to use abstraction to think in a general way about a collection of data and procedural steps
- How to describe the solution of a problem as an algorithm
- The three paradigms of computing and the position of MATLAB in that spectrum
- Three aspects of the apparently simple task of assigning a value to a variable
Programming Language Background

- **Abstraction**
  
  “expressing a quality apart from a particular implementation.”
  - “To convert from degrees Celsius to Kelvin, you add 273 to the temperature.”
  - “He drove home from the office.”

- **Algorithm**
  
  a sequence of instructions for solving a problem

- **Programming Paradigms**
  
  “a set of assumptions, concepts, values, and practices that constitutes a way of viewing reality.”
Abstraction

- Data abstraction
  - “To convert from degrees Celsius to Kelvin, you add 273 to the temperature.”

- Procedural abstraction
  - “He drove home from the office.”
  - “Baking cookies”
Baking some cookies

- buy a cookies box
- follow the directions

....Or a more detailed algorithm
Changing the order of the instructions can reduce execution time!!!!!
Programming paradigms

“a set of assumptions, concepts, values, and practices that constitutes a way of viewing reality.”

- **Functional programming** – every programming operation is actually implemented as a function call with no side effects.

- **Procedural programming** – the basic programs or subprograms are sequences of operations on data items.

- **Object-oriented programming (OOP)** – characterized by the concept of encapsulating data items with the methods or functions that manipulate those data items.
Procedural vs Functional

- Program: a sequence of instructions for a von Neumann m/c.
- Computation by instruction execution.
- Iteration.
- Modifiable or updateable variables.

- Program: a collection of function definitions (m/c independent).
- Computation by term rewriting.
- Recursion.
- Assign-only-once variables.
<table>
<thead>
<tr>
<th>Paradigm vs Language</th>
<th>Procedural style</th>
<th>Functional Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>i := 0; sum := 0;</td>
<td></td>
<td>func sum(n:int) : int;</td>
</tr>
<tr>
<td>while (i &lt; n) do</td>
<td></td>
<td>if n = 0</td>
</tr>
<tr>
<td>i := i + 1;</td>
<td></td>
<td>then 0</td>
</tr>
<tr>
<td>sum := sum + i</td>
<td></td>
<td>else n + sum(n-1)</td>
</tr>
<tr>
<td>end;</td>
<td></td>
<td>end;</td>
</tr>
</tbody>
</table>
Functional Style : Illustration

- **Definition : Equations**
  
  \[ \text{sum}(0) = 0 \]
  \[ \text{sum}(n) = n + \text{sum}(n-1) \]

- **Computation : Substitution and Replacement**
  
  \[ \text{sum}(2) \]
  \[ = 2 + \text{sum}(2-1) \]
  \[ = \ldots \]
  \[ = 3 \]
Procedural vs Object-Oriented

- Emphasis on procedural abstraction.
- Top-down design; Step-wise refinement.
- Suited for programming in the small.

- Emphasis on data abstraction.
- Bottom-up design; Reusable libraries.
- Suited for programming in the large.
Assigning Values to Variables

Syntax similar to conventional algebra; different meaning.

\[ z = x + y \]
\[ z = 4x - y \]

**Algebraic interpretation**: implied relationship between \( x \) and \( y \) by addition and simplification:

\[ 2y = 3x \]

**Programming**: I changed my mind about what I wanted to store in the variable \( z \). (\( x \) and \( y \) must have values)
Variable names

- any combination of uppercase and lower case alphabetic letters, numbers and the special characters _ and $

- myfirstvariable
- myFirstVar
- my_1st_var
- 1stvarofmine
Data Typing

- How a programming language treats data stored in a variable
  - **untyped**, e.g. Matlab
  - **typed**, e.g. C, Fortran, Java
    - must be specified both name and type
    - weak typing
    - strong typing

```plaintext
a Meter;
b Second;
c, d MeterPerSecond;
a + b, b + a, c + a **KO**
c = a * b + d **OK**
```

**NASAs metric confusion caused Mars orbiter loss**

September 30, 1999

CNN NASA lost a 125 million Mars orbiter because one engineering team used metric units while another used English units for a key spacecraft operation, according to a review finding released Thursday.

For that reason, information failed to transfer between the Mars Climate Orbiter spacecraft team at Lockheed Martin in Colorado and the mission navigation team in California. Lockheed Martin built the spacecraft.
Untyped languages

```plaintext
>> letter = 'A'
letter = A
>> letter = letter + 1
letter = 66

fx >>

>> radius=49
radius = 49
>> radius + 1
ans =
  50
>> radius='radius of a circle'
radius =
  radius of a circle
>> radius + 1
ans =
  Columns 1 through 16
  115 98 101 106 118 116 33 112 103 33 98 33 100 106 115 100
  Columns 17 through 18
  109 102
```
Additional reading: the ascii code

- ASCII stands for American Standard Code for Information Interchange. Computers can only understand numbers, so an ASCII code is the numerical representation of a character such as 'a' or '@' or an action of some sort. ASCII was developed a long time ago and now the non-printing characters are rarely used for their original purpose. Below is the ASCII character table and this includes descriptions of the first 32 non-printing characters. ASCII was actually designed for use with teletypes and so the descriptions are somewhat obscure. If someone says they want your CV however in ASCII format, all this means is they want 'plain' text with no formatting such as tabs, bold or underscoring - the raw format that any computer can understand. This is usually so they can easily import the file into their own applications without issues. Notepad.exe creates ASCII text, or in MS Word you can save a file as 'text only'
### Additional reading: The ascii table

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hx Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hx Oct</th>
<th>Html Chr</th>
<th>Dec</th>
<th>Hx Oct</th>
<th>Html Chr</th>
<th>Dec</th>
<th>Hx Oct</th>
<th>Html Chr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>NUL (null)</td>
<td>32</td>
<td>20</td>
<td>040</td>
<td>64</td>
<td>40</td>
<td>100</td>
<td>96</td>
<td>60</td>
<td>140</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>SOH (start of heading)</td>
<td>33</td>
<td>21</td>
<td>041</td>
<td>65</td>
<td>41</td>
<td>101</td>
<td>97</td>
<td>61</td>
<td>141</td>
</tr>
<tr>
<td>2</td>
<td>002</td>
<td>STX (start of text)</td>
<td>34</td>
<td>22</td>
<td>042</td>
<td>66</td>
<td>42</td>
<td>102</td>
<td>98</td>
<td>62</td>
<td>142</td>
</tr>
<tr>
<td>3</td>
<td>003</td>
<td>ETX (end of text)</td>
<td>35</td>
<td>23</td>
<td>043</td>
<td>67</td>
<td>43</td>
<td>103</td>
<td>99</td>
<td>63</td>
<td>143</td>
</tr>
<tr>
<td>4</td>
<td>004</td>
<td>EOT (end of transmission)</td>
<td>36</td>
<td>24</td>
<td>044</td>
<td>68</td>
<td>44</td>
<td>104</td>
<td>100</td>
<td>64</td>
<td>144</td>
</tr>
<tr>
<td>5</td>
<td>005</td>
<td>ENQ (enquiry)</td>
<td>37</td>
<td>25</td>
<td>045</td>
<td>69</td>
<td>45</td>
<td>105</td>
<td>101</td>
<td>65</td>
<td>145</td>
</tr>
<tr>
<td>6</td>
<td>006</td>
<td>ACK (acknowledge)</td>
<td>38</td>
<td>26</td>
<td>046</td>
<td>70</td>
<td>46</td>
<td>106</td>
<td>102</td>
<td>66</td>
<td>146</td>
</tr>
<tr>
<td>7</td>
<td>007</td>
<td>BEL (bell)</td>
<td>39</td>
<td>27</td>
<td>047</td>
<td>71</td>
<td>47</td>
<td>107</td>
<td>103</td>
<td>67</td>
<td>147</td>
</tr>
<tr>
<td>8</td>
<td>010</td>
<td>BS (backspace)</td>
<td>40</td>
<td>28</td>
<td>050</td>
<td>72</td>
<td>48</td>
<td>110</td>
<td>104</td>
<td>68</td>
<td>150</td>
</tr>
<tr>
<td>9</td>
<td>011</td>
<td>HT (horizontal tab)</td>
<td>41</td>
<td>29</td>
<td>051</td>
<td>73</td>
<td>49</td>
<td>111</td>
<td>105</td>
<td>69</td>
<td>151</td>
</tr>
<tr>
<td>10</td>
<td>012</td>
<td>LF (NL line feed, new line)</td>
<td>42</td>
<td>2A</td>
<td>052</td>
<td>74</td>
<td>4A</td>
<td>112</td>
<td>106</td>
<td>6A</td>
<td>152</td>
</tr>
<tr>
<td>11</td>
<td>013</td>
<td>VT (vertical tab)</td>
<td>43</td>
<td>2B</td>
<td>053</td>
<td>75</td>
<td>4B</td>
<td>113</td>
<td>107</td>
<td>6B</td>
<td>153</td>
</tr>
<tr>
<td>12</td>
<td>014</td>
<td>FF (NF form feed, new page)</td>
<td>44</td>
<td>2C</td>
<td>054</td>
<td>76</td>
<td>4C</td>
<td>114</td>
<td>108</td>
<td>6C</td>
<td>154</td>
</tr>
<tr>
<td>13</td>
<td>015</td>
<td>CR (carriage return)</td>
<td>45</td>
<td>2D</td>
<td>055</td>
<td>77</td>
<td>4D</td>
<td>115</td>
<td>109</td>
<td>6D</td>
<td>155</td>
</tr>
<tr>
<td>14</td>
<td>016</td>
<td>SO (shift out)</td>
<td>46</td>
<td>2E</td>
<td>056</td>
<td>78</td>
<td>4E</td>
<td>116</td>
<td>110</td>
<td>6E</td>
<td>156</td>
</tr>
<tr>
<td>15</td>
<td>017</td>
<td>SI (shift in)</td>
<td>47</td>
<td>2F</td>
<td>057</td>
<td>79</td>
<td>4F</td>
<td>117</td>
<td>111</td>
<td>6F</td>
<td>157</td>
</tr>
<tr>
<td>16</td>
<td>020</td>
<td>DLE (data link escape)</td>
<td>48</td>
<td>30</td>
<td>060</td>
<td>80</td>
<td>50</td>
<td>120</td>
<td>112</td>
<td>70</td>
<td>160</td>
</tr>
<tr>
<td>17</td>
<td>021</td>
<td>DC1 (device control 1)</td>
<td>49</td>
<td>31</td>
<td>061</td>
<td>81</td>
<td>51</td>
<td>121</td>
<td>113</td>
<td>71</td>
<td>161</td>
</tr>
<tr>
<td>18</td>
<td>022</td>
<td>DC2 (device control 2)</td>
<td>50</td>
<td>32</td>
<td>062</td>
<td>82</td>
<td>52</td>
<td>122</td>
<td>114</td>
<td>72</td>
<td>162</td>
</tr>
<tr>
<td>19</td>
<td>023</td>
<td>DC3 (device control 3)</td>
<td>51</td>
<td>33</td>
<td>063</td>
<td>83</td>
<td>53</td>
<td>123</td>
<td>115</td>
<td>73</td>
<td>163</td>
</tr>
<tr>
<td>20</td>
<td>024</td>
<td>DC4 (device control 4)</td>
<td>52</td>
<td>34</td>
<td>064</td>
<td>84</td>
<td>54</td>
<td>124</td>
<td>116</td>
<td>74</td>
<td>164</td>
</tr>
<tr>
<td>21</td>
<td>025</td>
<td>NAK (negative acknowledge)</td>
<td>53</td>
<td>35</td>
<td>065</td>
<td>85</td>
<td>55</td>
<td>125</td>
<td>117</td>
<td>75</td>
<td>165</td>
</tr>
<tr>
<td>22</td>
<td>026</td>
<td>SYN (synchronous idle)</td>
<td>54</td>
<td>36</td>
<td>066</td>
<td>86</td>
<td>56</td>
<td>126</td>
<td>118</td>
<td>76</td>
<td>166</td>
</tr>
<tr>
<td>23</td>
<td>027</td>
<td>ETB (end of trans. block)</td>
<td>55</td>
<td>37</td>
<td>067</td>
<td>87</td>
<td>57</td>
<td>127</td>
<td>119</td>
<td>77</td>
<td>167</td>
</tr>
<tr>
<td>24</td>
<td>030</td>
<td>CAN (cancel)</td>
<td>56</td>
<td>38</td>
<td>070</td>
<td>88</td>
<td>58</td>
<td>130</td>
<td>120</td>
<td>78</td>
<td>170</td>
</tr>
<tr>
<td>25</td>
<td>031</td>
<td>EM (end of medium)</td>
<td>57</td>
<td>39</td>
<td>071</td>
<td>89</td>
<td>59</td>
<td>131</td>
<td>121</td>
<td>79</td>
<td>171</td>
</tr>
<tr>
<td>26</td>
<td>032</td>
<td>SUB (substitute)</td>
<td>58</td>
<td>3A</td>
<td>072</td>
<td>90</td>
<td>5A</td>
<td>132</td>
<td>122</td>
<td>7A</td>
<td>172</td>
</tr>
<tr>
<td>27</td>
<td>033</td>
<td>ESC (escape)</td>
<td>59</td>
<td>3B</td>
<td>073</td>
<td>91</td>
<td>5B</td>
<td>133</td>
<td>123</td>
<td>7B</td>
<td>173</td>
</tr>
<tr>
<td>28</td>
<td>034</td>
<td>FS (file separator)</td>
<td>60</td>
<td>3C</td>
<td>074</td>
<td>92</td>
<td>5C</td>
<td>134</td>
<td>124</td>
<td>7C</td>
<td>174</td>
</tr>
<tr>
<td>29</td>
<td>035</td>
<td>GS (group separator)</td>
<td>61</td>
<td>3D</td>
<td>075</td>
<td>93</td>
<td>5D</td>
<td>135</td>
<td>125</td>
<td>7D</td>
<td>175</td>
</tr>
<tr>
<td>30</td>
<td>036</td>
<td>RS (record separator)</td>
<td>62</td>
<td>3E</td>
<td>076</td>
<td>94</td>
<td>5E</td>
<td>136</td>
<td>126</td>
<td>7E</td>
<td>176</td>
</tr>
<tr>
<td>31</td>
<td>037</td>
<td>US (unit separator)</td>
<td>63</td>
<td>3F</td>
<td>077</td>
<td>95</td>
<td>5F</td>
<td>137</td>
<td>127</td>
<td>7F</td>
<td>177</td>
</tr>
</tbody>
</table>

Source: [www.LookupTables.com](http://www.LookupTables.com)
Classes and Objects in Matlab

```
>> thisNumber = 42.0
```

- **Value**
- **Class double**
- **instance of a class i.e. an object**
Starting with Matlab

- Do by yourself the examples in sections 2.2, 2.3 and 2.4
The first problem

Create a script derived from the Pythagorean theorem to compute the hypotenuse of a right triangle:

\[ H^2 = A^2 + B^2 \]

where \( A \) and \( B \) are the sides adjacent to the right angle, and \( H \) is the hypotenuse opposite.

```matlab
clear
clc

A = 3; % the first side of a triangle
B = 4; % the second side of a triangle
hypSq = A^2 + B^2; % the square of the
% hypotenuse
H = sqrt(hypSq) % the answer
```
Another example: Spaceship 1

Problem: Assuming that the spacecraft uses all its fuel to achieve a vertical velocity $u$ at 25000 feet, what is the value of $u$ for the spacecraft to reach outer space (100 Km)?

Solution: $v^2 = u^2 - 2gs \Rightarrow u = \sqrt{2gs}$
Engineering Example—Spacecraft Launch

clear
clc

cmPerInch = 2.54; \% general knowledge
inchesPerFt = 12; \% general knowledge
metersPerCm = 1/100; \% general knowledge
MetersPerFt = metersPerCm * cmPerInch * inchesPerFt;
startFt = 25000; \% ft - given
startM = startFt * MetersPerFt;
g = 9.81; \% m/sec^2
top = 100; \% km - given
s = (top*1000) - startM; \% m
initial_v = (2*g*s)^0.5 \% the final answer
An open source alternative to Matlab

- OCTAVE
  - http://www.octave.org

GNU Octave is a high-level interpreted language, primarily intended for numerical computations. It provides capabilities for the numerical solution of linear and nonlinear problems, and for performing other numerical experiments. It also provides extensive graphics capabilities for data visualization and manipulation. Octave is normally used through its interactive command line interface, but it can also be used to write non-interactive programs. **The Octave language is quite similar to Matlab so that most programs are easily portable.**