

# CHARACTERS AND STRINGS

## Going Beyond Numbers

## WHAT DO WE WANT TO DO?

- ◆ MATLAB is first and foremost a “numerical” computation platform
- ◆ In practice, however, there are many occasions where one has to work with characters or alpha-numeric quantities

## EXAMPLES

- ◆ Here are a few examples:
  - placing titles and labels on graphs
  - working with file names such as day1.dat, day2.dat, etc.
  - comparing two strings, one user supplied and one in the library
  - decimal to binary string conversion

## CHARACTER ARRAY

- ◆ How does MATLAB define a character array?
- ◆ Any expression enclosed inside single quotes is converted to a character array
  - `u='villanova'`
  - `u` is defined as a 1x9 character array taking up 18 bytes

## OTHER FORMS

- ◆ All the usual MATLAB conventions apply
  - `u=['villanova','university']` is a 1x19 array
  - `u=['villanova';'university']`. Does this work?

## STRING COMPARISON

- ◆ One of the key operations performed on character arrays is string comparison
- ◆ Take
  - `s1='ece 2290'`
  - `s2='ece 4790'`
  - `c=strcmp(s1,s2)`
- ◆ `c=0` means `s1` and `s2` are not equal

## PARTIAL COMPARISON

- ◆ It is possible to do partial matching, say, the first 3 characters
  - `c=strncmp(s1,s2,4)`
- ◆ The result of this comparison will be `c=1`, indicating that the first 4 positions are the same

## RELATIONAL OPERATORS

- ◆ Relational operators (`==`, `<=`, `>=`) can be applied to character arrays
  - `s1='james'`
  - `s2='jenny'`
  - `s1==s2` returns a binary array with 1's pointing to positions of equality
- ◆ What does `s2 > s1` mean?

## SEARCHING FOR A STRING

- ◆ We can find the occurrence of a string by using *findstr*
  - label='villanova'
  - position=findstr('nova',label)
- ◆ findstr points to the beginning position of the string

## REPLACING A STRING

- ◆ We can replace the occurrence of a string using *strrep*
  - strrep(label,'vill','Vill')

## NUMBER/STRING CONVERSION

- ◆ Sometimes we need control over individual digits
- ◆ Take
  - n=2290
  - m=*int2str*(n)
- ◆ Now, m is a 1x4 array with m(1)=2. You can set, for example, m(2)=3

## LABELING PLOTS AT RUNTIME

- ◆ Sometimes plot labels are not known before execution
- ◆ For example, the label should say...  
*voltage ranges from 5 to 9 volts*
- ◆ How do we pass 5 and 9 to the *title* command at runtime?

## MIXING STRINGS AND NUMBERS

- ◆ Let's say we want to create a text string that says

temperature plot:4pm to 6pm

- ◆ Call the string s

```
s=['temperature plot:',int2str(4),'pm to ',int2str(6),'pm']
```

## TITLING A PLOT

- ◆ The following code inserts a parameter, determined at runtime, in a graph title

```
for i=1:4
    f=i;% pick a frequency
    x=cos(2*pi*f*t); %signal to plot
    figure(i);%number figures
    plot(t,x);grid;
    s=['a sinusoid of frequency ',int2str(f),' Hz'];%construct title
    title(s)
end
```

## SAVING LARGE NUMBER OF FILES AT ONCE

- ◆ In many situations your code generates a large number of files which you want them saved, each under a different name, like

- datafile1.mat
- datafile2.mat
- etc


## ILLUSTRATING WHAT WE WANT

- ◆ Say we want to generate 10 sinusoids each at a different frequency then save them to individual files
  - t=0:0.01:1;
  - for f=1:10
  - x=cos(2\*pi\*t\*f);
  - save datafile x
  - end
- ◆ What we really want save statement to do is
  - save datafile1 x
  - save datafile2 x
  - save datafile3 x
  - ...

## SOLUTION: *eval*

- ◆ *eval(s)* evaluates string *s* as if it was an explicit MATLAB statement typed in the command window

- ◆ For example,

  
`eval('plot(x)')`

– is equivalent to `plot(x)`

## CUSTOMIZING STRING AT RUNTIME

- ◆ First put together the string piece by piece. Here we want the filename appended by the loop index, i.e. `datafile1`, `datafile2` etc
- ◆ Then insert the string inside *eval*

## ANATOMY OF A STRING

- ◆ Here is what you need to do

blank

`s=['save datafile',int2str(i),' x']`



digit to be appended  
to file name

- ◆ then do *eval(s)*

## LOADING MULTIPLE DATA FILES

- ◆ The inverse of saving many files at once is loading them
- ◆ If you are given hundreds of files, how would you read them?

## EXAMPLE: LOADING DIGITAL VIDEO

- ◆ Video is a collection of thousands of still frames that are saved in files named *frame1, frame2, frame3* etc.
- ◆ Question is how do you import them into MATLAB frame by frame, but not manually?

## COMPOSING THE LOAD STRING

- ◆ To load file *frame* we write
  - load frame
- ◆ Again, we need to load *frame1, frame2, etc.* instead of just *frame*

## HOMEWORK

- ◆ Do the problem just stated, i.e. write a code to load the 4 files stored on the Mac server, *frame1.mat* all the way to *frame4.mat*. Then plot the recovered data, one plot for each.
- ◆ Title each plot. Titles should read
  - plot of frame1
  - plot of frame2
  - etc.