MATLAB for the Sciences
Error Checking, Easier Programming, and MATLAB Movies

Jon M. Ernstberger

January 26, 2009
Making Sure Things Work Correctly

- All through each program we’ve *assumed* that inputs are of a certain type and dimension.
- What if someone stumbles upon your (or my!) code and doesn’t read the internal comments?
- Inputting certain “checks” into your code can do several things:
  - Makes sure you program correctly,
  - Make sure the users input the correct type/number of parameters, and
  - Make your code appear more professional.
- MATLAB builds tools into the language that help things work out well.
Basic Tests

- Examine the Matrix Multiply Code
- We presumed the two matrices were $n \times n$.
- What if someone inputs matrices of different sizes that are not multiplicable?
- Use the size command.

```matlab
function C=matrix_multiply(A,B)
[m1,n1]=size(A);
[m2,n2]=size(B);

if n1~=m2
    error('Matrices Are Not of Appropriate Dimension');
end;
```

Jon M. Ernstberger
MATLAB for the Sciences
error('Matrices Are Not of Appropriate Dimension');

- MATLAB Error Messages
  - Flash an error to screen
  - Abort the Program!

- Ensures that problems get fixed with variable passing.
- May also ensure that your code is correct!
- What does the next piece do?

```
if n1~=m2
    error('Matrices Are Not of Appropriate Dimension');
end;
```
- This built-in command returns the number of variables passed to the function.
- This does not return the number of inputs that the function!
- Why is this useful?
This built-in command returns the number of variables passed to the function.

This does not return the number of inputs that the function!

Why is this useful? What if the user doesn’t input two matrices into our multiply function?

```matlab
if nargin~=2
    error('Two matrices must be supplied');
end;
```
Another nargin Use?

- You can use nargin to set defaults.
- You could say

```matlab
function C = matrix_multiply(A,B)

if nargin == 1
    [m1,n1] = size(A);
    B = eye(n1);
end;
```

- What does that code snippet imply?
You can use `nargin` to set defaults.

You could say

```matlab
function C = matrix_multiply(A,B)
  if nargin==1
    [m1,n1] = size(A);
    B = eye(n1);
  end;
```

What does that code snippet imply?

This would ensure that *A always* can be multiplied.
What do you think `nargout` means?
Are there cases where I specific output?
Revisit the random sum program and change it to a function

```matlab
function [count,sum]=random_sum

Further, add (at the bottom)

if nargout==1
    s=['The number of iterations required is ','... num2str(count),'.' ];
    disp(s);
elseif nargout==2
    s=['There were ',num2str(count),... ' iterations required to sum to ','... num2str(sum),'.' ];
    disp(s);
end;
```
A cell array provides a storage mechanism for dissimilar kinds of data. You can store arrays of different types and/or sizes within the cells of a cell array.¹

- For instance \( C = \{15, 'Stacey', \text{rand}(3)\} \); is a \( 1 \times 3 \) cell array.
- I can then reference the 3rd entry of the cell array by \( C\{3\} \).
- I can reference the (1, 2) entry of the matrix in \( C\{3\} \).

¹MATLAB Cell Online Documentation
Cells, cont.

- You can also stick things in a cell array entry-wise:
  ```
  D{1}='dog';
  D{2,2}=rand(15);
  D{3,2}=15;
  ```

- I could reference the (6, 7) entry in the random matrix in the {2,2} cell entry. i.e.,
  ```
  D{2,2}(6,7)
  ```
  
  ans =
  0.8765

- Why did we talk about all of this?
varargin implies a variable number of inputs once that variable is in the function call.

Further, varargin must be a cell so all manner of dissimilar variables can be passed.

For example:
function C=matrix_multiply2(varargin)
if length(varargin)==0 | length(varargin)==1
    error('Not Enough Inputs');
elseif length(varargin)>=2
    if length(varargin)>2
        disp('Disregarding additional inputs');
    end;
    [m1,n1]=size(varargin{1});
    [m2,n2]=size(varargin{2});

    if n1~=m2
        error('Inappropriate matrix dimension');
    end;
end;
How to Make a Movie

- You can put a Figure into an array using the `getframe` command.
- Then, you can either play the movie directly in MATLAB using the `movie` command or
- you can convert the movie to a `.avi` file.
- \( M(i) = \text{getframe}(1); \)
Fibonacci Sequence

```matlab
fib=zeros(100,1);
fib(1)=1;fib(2)=1;

for i=3:100
    fib(i)=fib(i-1)+fib(i-2);
end;

figure(1)
plot(1:i,fib(1:i))
axis([0 100,0,4e20])

M(i-2)=getframe(1);
end;

movie2avi(M,'fibmovie');
```
Fibonacci Sequence

```matlab
fib=zeros(100,1);
fib(1)=1;fib(2)=1;

for i=3:100
    fib(i)=fib(i-1)+fib(i-2);

figure(1)
plot(1:i,fib(1:i))
axis([0 100,0,4e20])

M(i-2)=getframe(1);
end;

movie2avi(M,'fibmovie');
```

Note that I plot in Figure 1.
Fibonacci Sequence

fib=zeros(100,1);
fib(1)=1;fib(2)=1;

for i=3:100
    fib(i)=fib(i-1)+fib(i-2);

figure(1)
plot(1:i,fib(1:i))
    axis([0 100,0,4e20])

M(i-2)=getframe(1);
end;

movie2avi(M,'fibmovie');

Note that I plot in Figure 1.

Note that I always getframe(1).
Fibonacci Sequence

```matlab
fib=zeros(100,1);
fib(1)=1;fib(2)=1;

for i=3:100
    fib(i)=fib(i-1)+fib(i-2);

figure(1)
plot(1:i,fib(1:i))
axis([0 100,0,4e20])

M(i-2)=getframe(1);
end;

movie2avi(M,'fibmovie');
```

- Note that I plot in Figure 1.
- Note that I always `getframe(1)`.
- Why am I fixing the size axes with the `axis` command?
Fibonacci Sequence

```matlab
fib=zeros(100,1);
fib(1)=1;fib(2)=1;
for i=3:100
    fib(i)=fib(i-1)+fib(i-2);
end;
figure(1)
plot(1:i,fib(1:i))
axis([0 100,0,4e20])
M(i-2)=getframe(1);
end;
movie2avi(M,'fibmovie');
```

- Note that I plot in Figure 1.
- Note that I always getframe(1).
- Why am I fixing the size axes with the axis command?
- Why M(i-2)?
Fibonacci Sequence

fib=zeros(100,1);
fib(1)=1;fib(2)=1;

for i=3:100
    fib(i)=fib(i-1)+fib(i-2);

figure(1)
plot(1:i,fib(1:i))
axis([0 100,0,4e20])

M(i-2)=getframe(1);
end;

movie2avi(M,’fibmovie’);

- Note that I plot in Figure 1.
- Note that I always getframe(1).
- Why am I fixing the size axes with the axis command?
- Why M(i-2)?
- movie2avi(M,’fibmovie’); converts to .avi format.
Why are we spending this time on MATLAB movies.

- Often, a movie is priceless in terms of relaying information.
- Putting these in presentations can make quite an impact if used correctly.

MATLAB Movie Tutorial - Example 1
Why are we spending this time on MATLAB movies.

- Often, a movie is priceless in terms of relaying information.
- Putting these in presentations can make quite an impact if used correctly.

Are there any downsides to the use of movies?

- If you make a movie that looks anything less than superbly professional, it looks horrible.
- Presentations and websites get large!