Matlab Tweaks II - The Profiler

- tools for optimizing your code -

Benjamin Unger

Tools Seminar - TU Berlin

May 11, 2015

Motivation

1. Measuring Performance in MATLAB

2. Code Optimization

Measuring Performance - tic/toc

Syntax

```
tic;
   code of interest;
toc
```

```
Elapsed time is 0.003820 seconds.
```

Nested tic/toc

```
t1 = tic;
maxT = 0;
   for i=1:100
     t2 = tic;
     code of interest;
     maxT = max(maxT,toc(t2));
   end
toc(t1)
```

Measuring Performance - cputime

Syntax

```
t = cputime;
    code of interest;
el = cputime-t;
```

cputime returns the CPU time in seconds that has been used by the MATLAB process since MATLAB started.

Measuring Performance - The Profiler I

Definition (Profiling)

Profiling is a form of *dynamic program analysis* that measures the performance of function calls.

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
28	`C(j,k) = C(j,k) + A(j,i)*B(i,k...`	1000000	5.843 s	34.0%	
29	`end`	1000000	4.920 s	28.6%	
137	`print -dpng heart`	1	2.317 s	13.5%	
111	`pause(.1)`	18	0.738 s	4.3%	
84	`pause(.1)`	11	0.718 s	4.2%	
All other lines			2.665 s	15.5%	
Totals			17.200 s	100%	

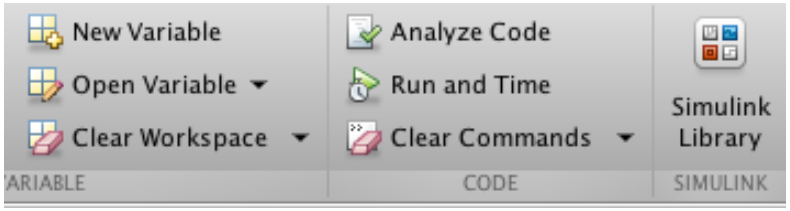
Measuring Performance - The Profiler II

Syntax

```
profile clear     % clears the view contents
profile on        % start profiling
doFunction();     % or script
profile off       % stop profiling
profile viewer    % view results
```

Profilers are available for many programming languages:

- Python `cProfile`
- Java `JConsole`, `JProfiler`
- R `Rprof()`



Measuring Performance - The Profiler III

Additional Options

-timer `cpu` measurements are taken as cpu time
`real` measurements are taken as wallclock time

-history If this option is specified, MATLAB records the exact sequence of function calls so that a function call history report can be generated. (`-nohistory`)

-memory Turns off memory stats in the profile report (*undocumented*)

1 Measuring Performance in MATLAB

2 Code Optimization

Array Preallocation I

Preallocate arrays if they are created step-by-step.

```
data = zeros(10,1);
for i=1:10
   data(i) = i^2;
end
```

If `data` is not preallocated, each time we concatenate the array, the following steps happen:

- Find a contiguous block that is large enough to store the new array of length $+1$.
- Copy old array into the new array.
- Add the new value to the new array.
- Delete the old array.

Array Preallocation I

Preallocate arrays if they are created step-by-step.

```
data = zeros(10,1);
for i=1:10
   data(i) = i^2;
end
```

If `data` is not preallocated, each time we concatenate the array, the following steps happen:

- Find a contiguous block that is large enough to store the new array of length $+1$.
- Copy old array into the new array.
- Add the new value to the new array.
- Delete the old array.

Array Preallocation II

Do not allocate in general.

```
mat = zeros(100,200);
mat = rand(100,200);
```

Unknown variable size:

- Estimate maximal size n and allocate with n
- If size grows bigger than n, allocate additional size, e. g. $n \leftarrow 2n$
- Remove unused items after computation: `data(57:end) = [];`

Vectorization[1] I

Definition (Vectorization)

The process of revising loop-based, scalar-oriented code to use MATLAB matrix and vector operations is called *vectorization*.

Appearance Vectorized mathematical code appears more like the mathematical expressions found in textbooks, making the code easier to understand.

Less Error Prone Without loops, vectorized code is often shorter. Fewer lines of code mean fewer opportunities to introduce programming errors.

Performance Vectorized code often runs much faster than the corresponding code containing loops.

[1] from `http://de.mathworks.com/help/matlab/matlab_prog/vectorization.html`

Vectorization II

Commonly used functions

all	Test to determine if all elements are nonzero
any	Test for any nonzeros
diff	Find differences and approximate derivatives
meshgrid	Generate X and Y arrays for 3-D plots
repmat	Replicate and tile an array
squeeze	Remove singleton dimensions from an array
sum	Compute the sum of array elements

Word of Warning

Shorter code is not always better code!

Pass by Value vs. Pass by Reference

Pass by value The local parameters are copies of the original arguments.

Pass by reference The local parameters are references to the storage location of the original arguments.

Example

Say I want to share a web page with you:
Pass by reference: I give you the url.
Pass by value: I print out the web page and give you the printout.

MATLAB's default behavior

- If you pass an input argument to a function that is not modified in the function $\Longrightarrow$ pass by reference
- If you modify the input argument in the function $\Longrightarrow$ pass by value

Data Accessing

```
A = rand(n,n);
for i=1:nLoops
   a = A(1,:);
end
```

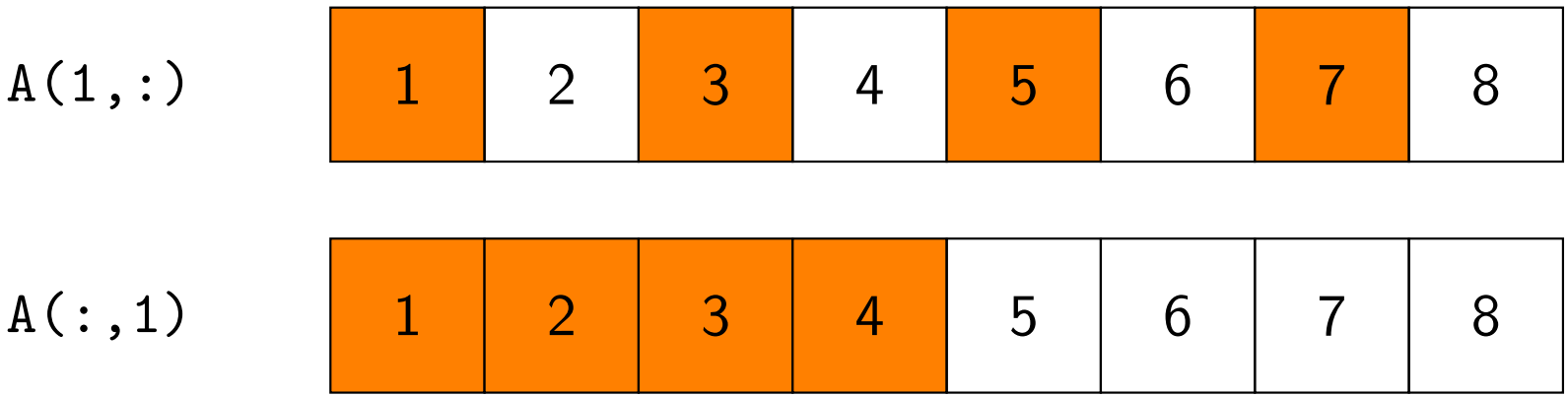
```
A = rand(n,n);
for i=1:nLoops
   a = A(:,1);
end
```

Data Accessing

```
A = rand(n,n);
for i=1:nLoops
    a = A(1,:);
end
```

```
A = rand(n,n);
for i=1:nLoops
    a = A(:,1);
end
```

MATLAB's 2D arrays are stored as sequential 1D arrays

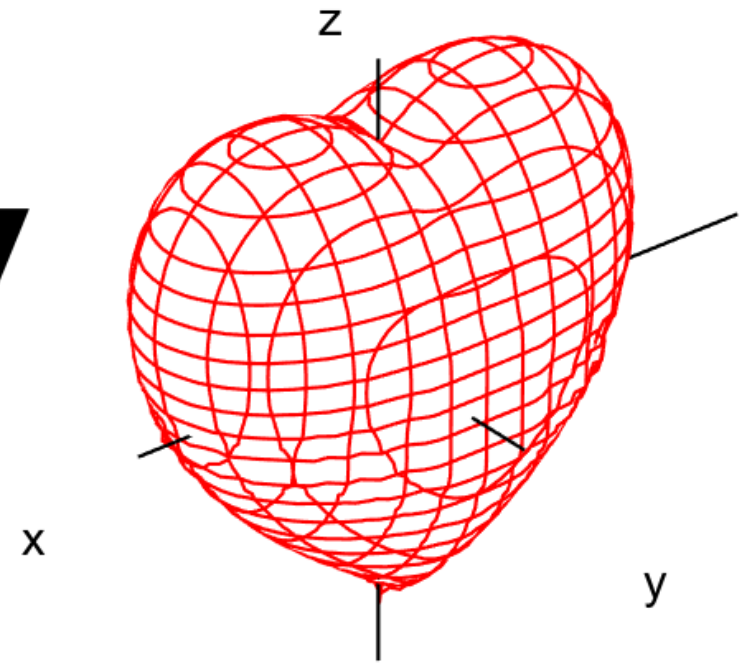


References and Further Reading

- http://undocumentedmatlab.com/blog/undocumented-profiler-options
- http://blogs.mathworks.com/loren/2012/11/29/understanding-array-preallocation/
- http://de.mathworks.com/help/matlab/matlab_prog/vectorization.html
- http://blogs.mathworks.com/loren/2006/05/10/memory-management-for-functions-and-variables/

Image credits:

- http://images.christianpost.com/blog/full/13437/way.jpg?w=288
- https://realtheatre.files.wordpress.com/2013/01/maze1.jpg

I 

Math

$$(x^2 + 9/4y^2 + z^2 - 1)^3 - x^2z^3 - 9/80y^2z^3 = 0$$

$$-3 \leq x, y, z \leq 3$$

http://stackoverflow.com/questions/1526898/how-do-i-reproduce-this-heart-shaped-mesh-in-matlab