Speeding Up MATLAB

MathWorks Japan

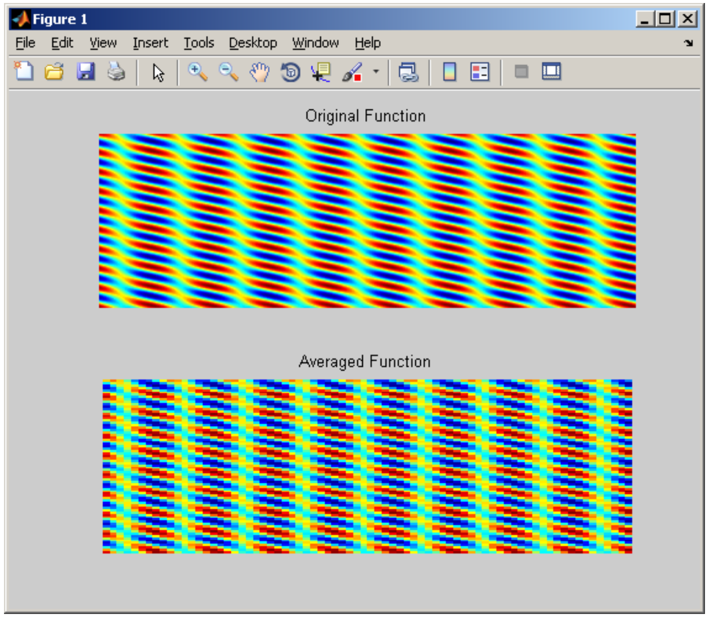
Teja Muppirala

Contents

- Techniques for speeding up code
  - Memory preallocation
  - Vectorization
  - Special functions
- Examples
  - 1:Block averaging on an image
  - 2:Generating a random walk

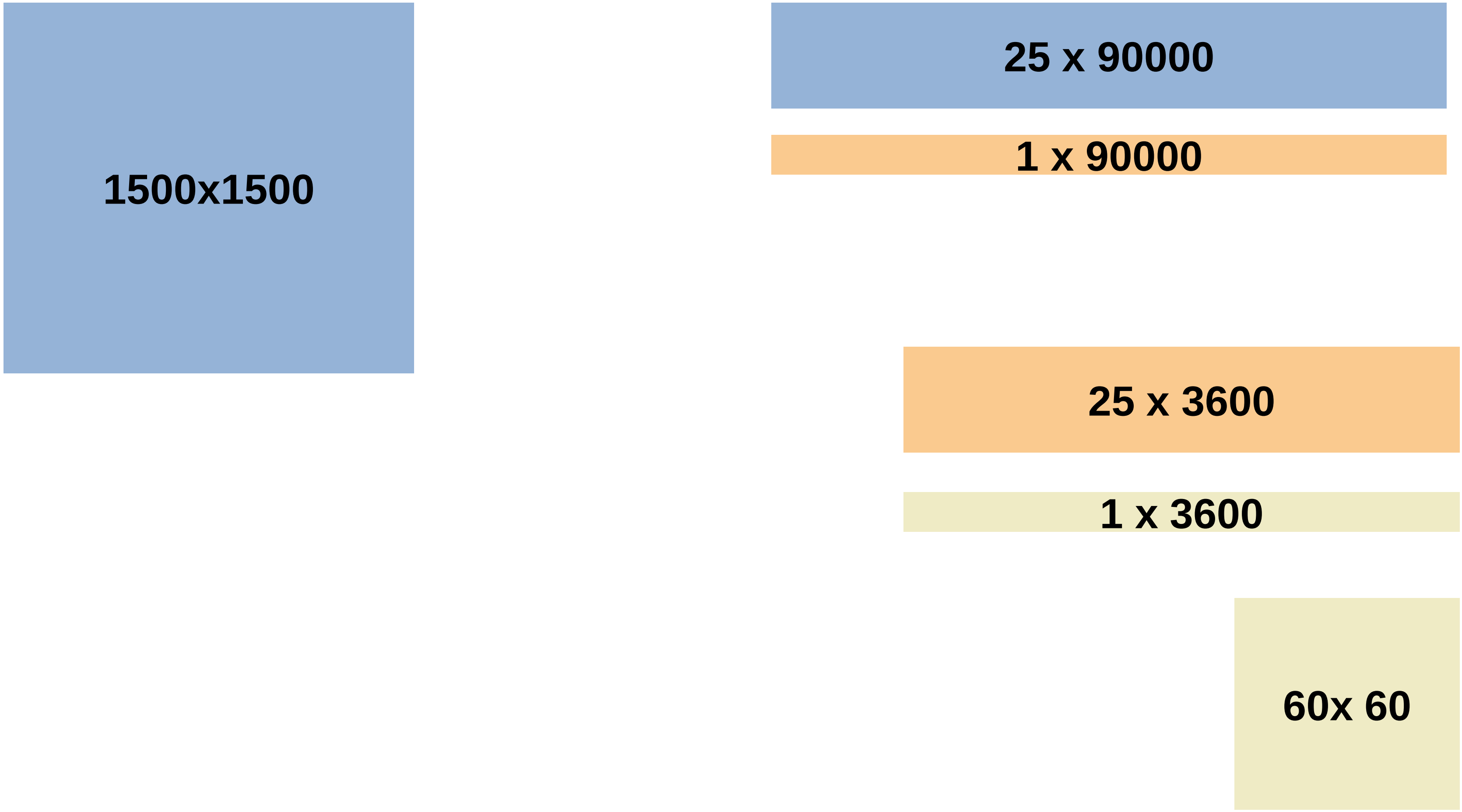
Example 1: Block averaging

$$f(x,y) =$$
$$5\sin(x+y) + 2\sin(x) + 2\cos(x)$$



- Evaluate on a 1500x1500 grid
- Average over 25x25 pixel blocks
- Compare the results
- Compare the code

Example 1: Block averaging
(RESHAPE & SUM)



1500x1500

25 x 90000

1 x 90000

25 x 3600

1 x 3600

60x 60

Example 1: Block averaging - Summary

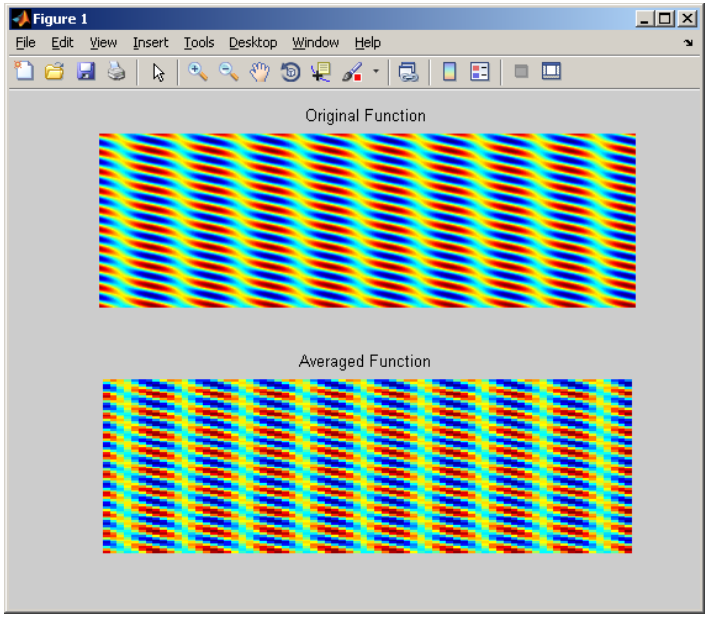
- Evaluate elapsed time

```
>> tic
>> toc
```

- Use the profiler

- Preallocate

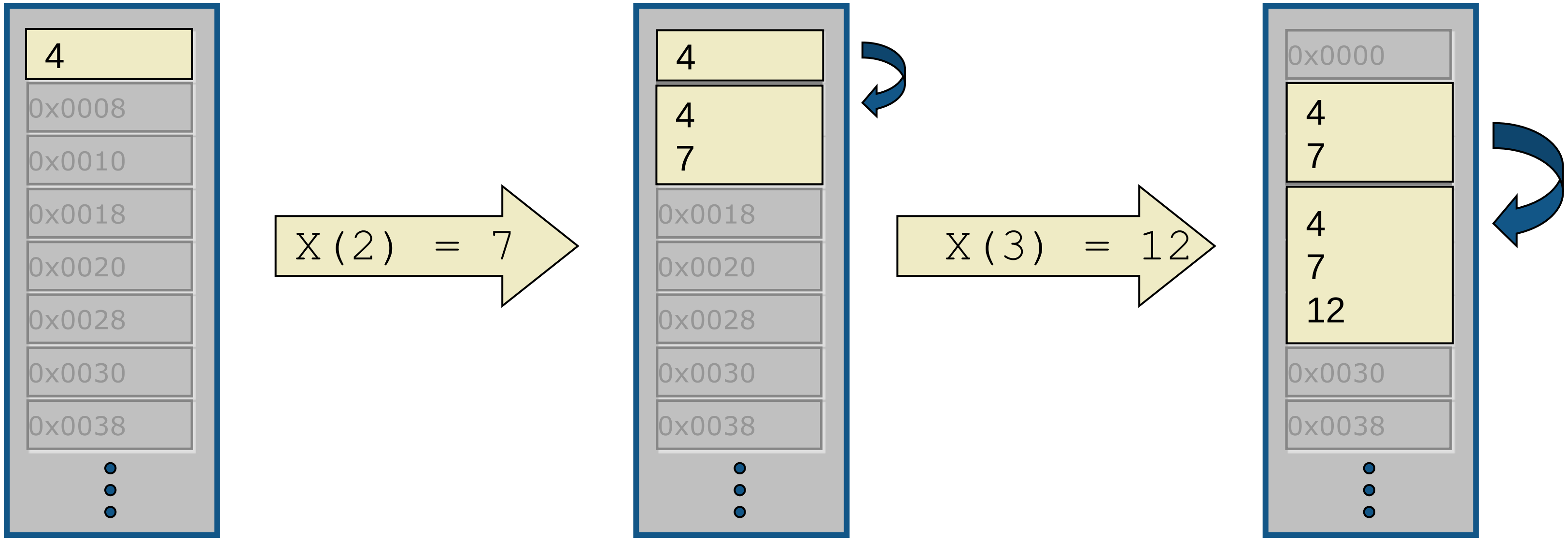
- Vectorize code



If you do not preallocate…

```
>> x = 4
>> x(2) = 7
>> x(3) = 12
```

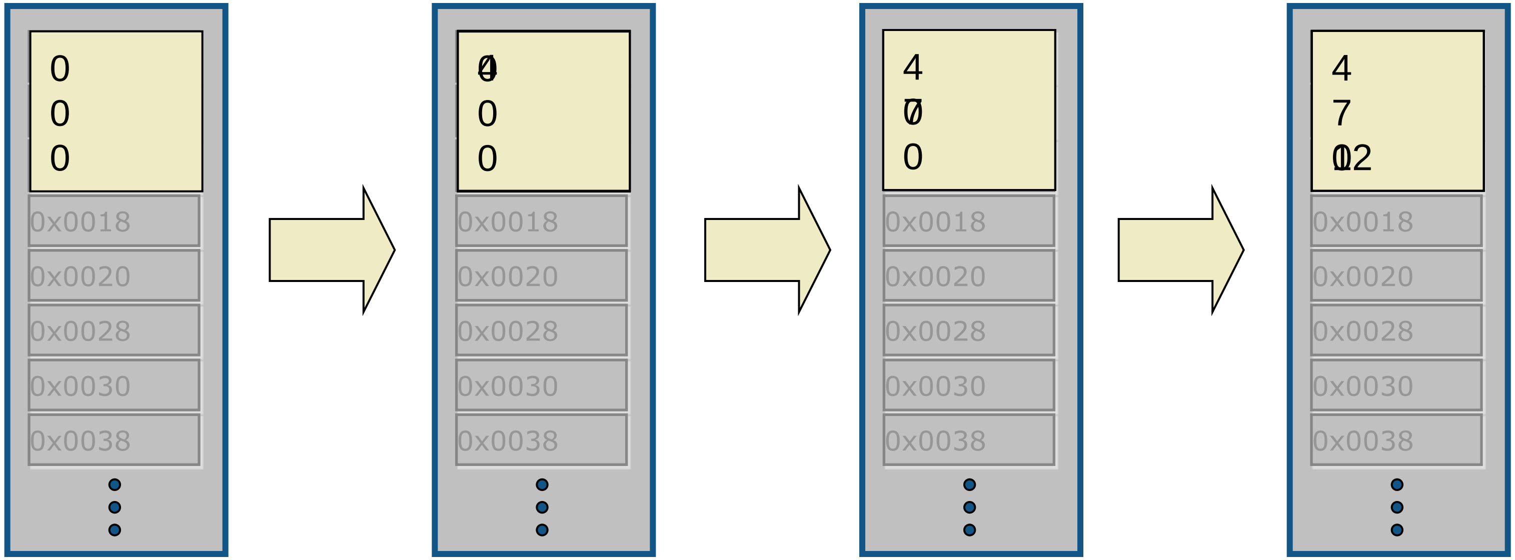
Resizing arrays is expensive



If you DO preallocate…

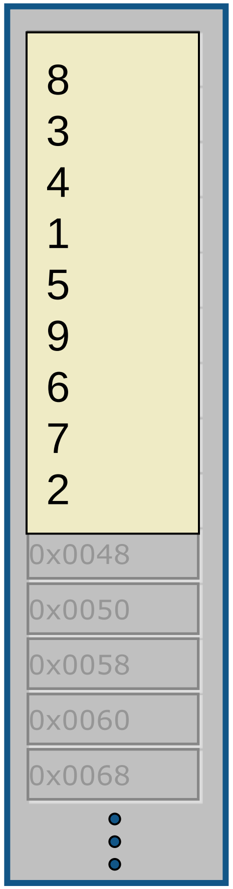
```
>> x = zeros(3,1)
>> x(1) = 4
>> x(2) = 7
>> x(3) = 12
```

No unnecessary copying



How MATLAB stores matrices

```
>> x = magic(3)
x =
     8     1     6
     3     5     7
     4     9     2
```

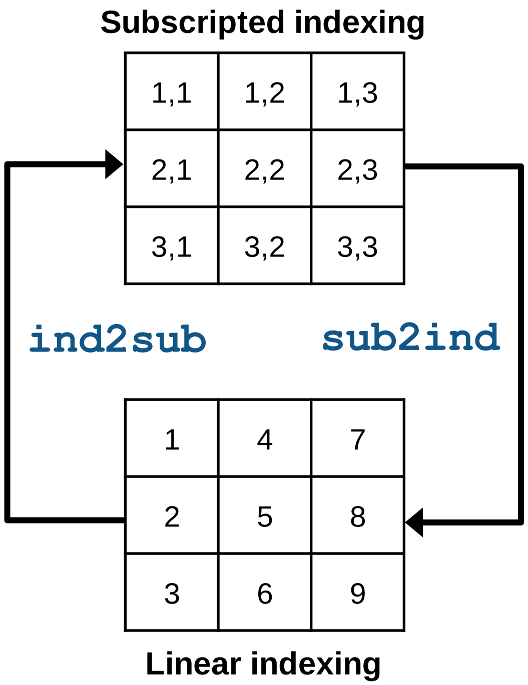


Column-major

See the June 2007 article in “The MathWorks News and Notes”:
http://www.mathworks.com/company/newsletters/news_notes/june07/patterns.html

Ways to access MATLAB arrays

- Subscript indexing
  - Specify row and column numbers
- Linear indexing
  - Specify only a single number
- Logical indexing
  - Use logical expressions to select elements



Subscripted indexing

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3

`ind2sub` `sub2ind`

1	4	7
2	5	8
3	6	9

Linear indexing

MATLAB Math Libraries

- Basic Math and Linear Algebra
  - BLAS: Basic Linear Algebra Subroutines
  - LAPACK: Linear Algebra Package
  - etc.

BLAS, LAPACK require contiguous memory

- JIT/Accelerator
  - Accelerate loops（FOR, WHILE）
  - Compiles code on the fly
  - Always improving

Things to keep in mind about the JIT:

- Changing variable size inside the loop

```
>> x = 1;
>> x = [1 2; 3 4];
```

- Changing datatype inside the loop

```
>> x = 1;
>> x = 'hello';
```

- Nonlinear loop indices

```
for  n = (1:1000).^2
   ...
end
```

Things to keep in mind about the JIT:

- Write IF statements in order of ease-of-evaluation

```
if A || B || C
   ...
end
```

- Reduce function call count by replacing simple functions with explicit code

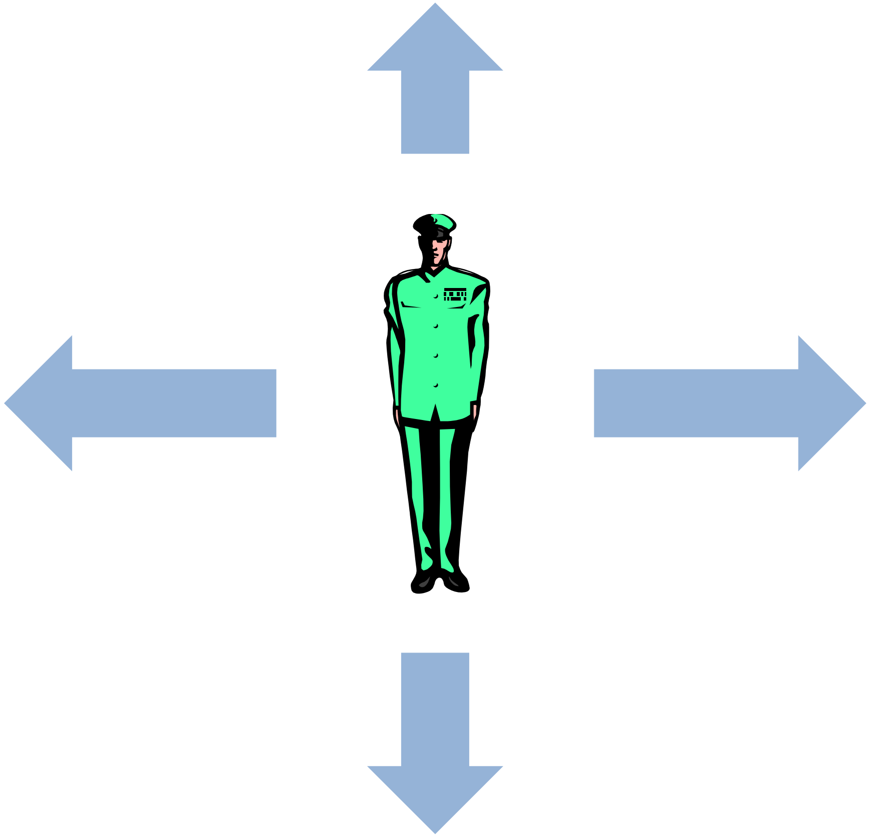
```
if a(1) == max(a) %% Function call to MAX()
   ...
end

if a(1) >= a(2) && a(1) >= a(3) % Possibly faster
   ...
end
```

Contents

- Techniques for speeding up code
  - Memory preallocation
  - Vectorization
  - Special functions
- Examples
  - 1: Block averaging on an image
  - 2: Generating a random walk

Example 2: Random walk

- Proceed randomly, North/South/East/West at each step



Example 2: Random walk

- Initial： 6.9sec
- Preallocation：0.4 sec
- Vectorization： 0.06 sec
- Redesigned algorithm using special functions：
- 0.005sec

Example 2: Random walk

- Preallocation and vectorization
- In general, hardware rendering is fast
  - `set(gcf,'renderer','opengl');`
- For special operations use special functions
  - Cumulative summation → **`cumsum`**
  - Many others:

`bsxfun`, `reshape`, `accumarray`, `histc`, `diff`, `repmat`, `permute`, **`sparse`**

BSXFUN (binary singleton expansion function)

- Expand a vector to be the same size as a matrix
  - Ex.1 Subtract the mean from each column of A

$$A = \begin{bmatrix} 2 & 5 & -2 \\ 4 & 1 & 3 \\ 0 & 6 & 2 \end{bmatrix}$$

$$mean(A) = \begin{bmatrix} 2 & 4 & 1 \end{bmatrix}$$

```
bsxfun(@minus, A, mean(A))

ans =

     0     1    -3
     2    -3     2
    -2     2     1
```

BSXFUN – Another example

- Ex.2 Compare the rows of A to the values in P

$$A = \begin{bmatrix} -1 & 0 & 3 \\ 6 & 8 & 7 \\ 11 & 7 & 13 \end{bmatrix}$$

$$P = \begin{bmatrix} 1 \\ 5 \\ 10 \end{bmatrix}$$

```
bsxfun(@gt, A, P)

ans =

     0     0     1
     1     1     1
     1     0     1
```

Summary

- Use the profiler, and find where the code is slow
- Use preallocation, vector/matrix operations, and other special MATLAB functions to speed up code

Thank you for listening