How to speed up MATLAB

Isaac Noh
Application Engineer
Agenda

- Two coding tips to speed up MATLAB code
- Leveraging multiple cores with MATLAB
- MATLAB to C:
  - Generating C code from MATLAB
  - Integrate C functions in MATLAB
Two Coding Tips for Speeding Up MATLAB

- Pre-Allocation of Memory
- Vectorization
Pre-Allocation of Memory

```matlab
for id = 1:10000
    a(id) = rand();
end
```

vs.

```matlab
a = nan(1, 10000);
for id = 1:10000
    a(id) = rand();
end
```
Effect of Not Preallocating Memory

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

![Memory Layout Diagram](image)
Benefit of Preallocation

```matlab
>> x = zeros(3,1)
>> x(1) = 4
>> x(2) = 7
>> x(3) = 12
```
Data Storage of MATLAB Arrays

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

See the June 2007 article in “The MathWorks News and Notes”:
Two Coding Tips for Speeding Up MATLAB

- Pre-Allocation of Memory
- Vectorization
Vectorization

- MATLAB is a matrix (vector)-based language
  → supports vectorized operations (faster in general)

```matlab
for id1 = 1:size(a,1)
    for id2 = 1:size(a,2)
        b(id1, id2) = a(id1, id2)^a(id1, id2);
    end
end
```

- Process in column blocks, rather than row blocks
Agenda

- Some coding techniques to speed up MATLAB code

- Leveraging Parallel Computing with MATLAB

- MATLAB to C:
  - Generating C code from MATLAB
  - Integrate C functions in MATLAB
Going Beyond Serial MATLAB Applications
Performance Gain with More Hardware

Using More Cores (CPUs)

Using GPUs

GPU cores

Device Memory
Independent Tasks or Iterations

- Ideal problem for parallel computing
- No dependencies or communications between tasks
- Examples: parameter sweeps, Monte Carlo simulations
Agenda

- Some coding techniques to speed up MATLAB code
- Leveraging multiple cores with MATLAB

MATLAB to C:
- Generating C code from MATLAB
- Integrate C functions in MATLAB
<table>
<thead>
<tr>
<th>File Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.c</td>
<td><strong>Implement</strong> C code on processors or hand off to software engineers</td>
</tr>
<tr>
<td>.lib, .dll</td>
<td><strong>Integrate</strong> MATLAB algorithms with existing C environment using source code and static/dynamic libraries</td>
</tr>
<tr>
<td>.exe</td>
<td><strong>Prototype</strong> MATLAB algorithms on desktops as standalone executables</td>
</tr>
<tr>
<td>MEX</td>
<td><strong>Accelerate</strong> user-written MATLAB algorithms</td>
</tr>
</tbody>
</table>
With MATLAB Coder, design engineers can:

- Maintain one design in MATLAB
- Design faster and get to C quickly
- Test more systematically and frequently
- Spend more time improving algorithms in MATLAB
Acceleration using MEX (MATLAB Executable)

- Call C or Fortran code directly from MATLAB
  - Integrate existing code using MEX API
  - Auto-generate C-based MEX files from MATLAB code using MATLAB Coder

- Speed-up factor will vary
  - May see speedup for state-based for-loops
  - May not see a speedup when MATLAB code is
    - Using multithreaded computations
    - Using optimized libraries (BLAS, FFTW, etc.)

```c
myFcn.c
void mexFunction(
    int nlhs, mxArray *plhs[],
    int nrhs, const mxArray *prhs[])
{
    /* more C code ... */
}
```
Questions?