Automatic C Code Generation from MATLAB
Marc Barberis – Application Engineering Group, MathWorks Inc.
AGENDA

- Quick Demo
- Benefits of Automatic C Code Generation
- In-Depth Example
- Comparison between MATLAB Coder and MATLAB Compiler
- Fixed-Point Design
- Conclusion
Demo: Using Generated C Code in a Stand-Alone C Project
Why translate MATLAB to C?

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="lib" /></td>
<td>Integrate MATLAB algorithms w/ existing C environment using source code or static libraries</td>
</tr>
<tr>
<td><img src="image" alt="exe" /></td>
<td>Prototype MATLAB algorithms on desktops as standalone executables</td>
</tr>
<tr>
<td><img src="image" alt="MEX" /></td>
<td>Accelerate user-written MATLAB algorithms</td>
</tr>
<tr>
<td><img src="image" alt="c,cpp" /></td>
<td>Implement C/C++ code on processors or hand-off to software engineers</td>
</tr>
</tbody>
</table>
Challenges with Manual Translation from MATLAB to C/C++

- Separate functional and implementation specification
  - Leads to multiple implementations that are inconsistent
  - Hard to modify requirements during development
  - Difficult to keep reference MATLAB code and C code in-sync
- Manual coding errors
- Time consuming and expensive
Automatic Translation of MATLAB to C

With MATLAB Coder, design engineers can

• Maintain one design in MATLAB
• Design faster and get to C/C++ quickly
• Test more systematically and frequently
• Spend more time improving algorithms in MATLAB
Implementation Constraints

Element by element multiply
Dot product
Matrix multiply

function a = foo(b, c)
a = b * c;

void foo(const double b[15],
    const double c[30], double a[18])
{
    int i0, i1, i2;
    for (i0 = 0; i0 < 3; i0++) {
        for (i1 = 0; i1 < 6; i1++) {
            a[i0 + 3 * i1] = 0.0;
        }
    }
    for (i2 = 0; i2 < 5; i2++) {
        a[i0 + 3 * i1] += b[i0 + 3 * i2] * c[i2 + 5 * i1];
    }
}

double foo(double b, double c)
{
    return b * c;
}
Implementation Constraints

- Polymorphism
- Memory allocation
- Processing matrices & arrays
- Fixed-point data types

```matlab
function [x_est p_est] = kalman_estimate(R,H,x_prd,p_prd,z)
    S = H * p_prd' * H' + R;
    B = H * p_prd;
    klm_gain = (S \ B)';
    x_est = x_prd + klm_gain * (z - H * x_prd);
    p_est = p_prd - klm_gain * H' * p_prd;
```

7 Lines of MATLAB

107 Lines of C
In-Depth Demo of MATLAB Coder

- Coder UI
- Code Generation options
- Generate code
- Browse through report
Supported MATLAB Language Features and Functions

- Broad set of language features and functions/system objects supported for code generation

<table>
<thead>
<tr>
<th>Matrices and Arrays</th>
<th>Data Types</th>
<th>Programming Constructs</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Matrix operations</td>
<td>• Complex numbers</td>
<td>• Arithmetic, relational, and logical operators</td>
<td>• MATLAB functions and sub-functions</td>
</tr>
<tr>
<td>• N-dimensional arrays</td>
<td>• Integer math</td>
<td>• Program control (if, for, while, switch)</td>
<td>• Variable length argument lists</td>
</tr>
<tr>
<td>• Subscripting</td>
<td>• Double/single-precision</td>
<td></td>
<td>• Function handles</td>
</tr>
<tr>
<td>• Frames</td>
<td>• Fixed-point arithmetic</td>
<td></td>
<td>Supported algorithms</td>
</tr>
<tr>
<td>• Persistent variables</td>
<td>• Characters</td>
<td></td>
<td>• &gt; 400 MATLAB operators and functions</td>
</tr>
<tr>
<td>• Global variables</td>
<td>• Structures</td>
<td></td>
<td>• &gt; 200 System objects for</td>
</tr>
<tr>
<td></td>
<td>• Numeric classes</td>
<td></td>
<td>• Signal processing</td>
</tr>
<tr>
<td></td>
<td>• Variable-sized data</td>
<td></td>
<td>• Communications</td>
</tr>
<tr>
<td></td>
<td>• System objects</td>
<td></td>
<td>• Computer vision</td>
</tr>
<tr>
<td></td>
<td>• Classes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Code Generation Readiness Tool

Instant feedback on code generation compliance of your MATLAB code

- Provides estimate of effort needed to generate C code from your MATLAB code on a scale of 1 to 5
- Provides a list of issues that need to be resolved in one report
- Gives detailed information on unsupported functions
Other Deployment Options
Deploying Applications with MATLAB Compiler

• Share applications
  – Desktop or Web software components
  – Supports full MATLAB language and most toolboxes
  – Requires MCR
    • Free run-time library
    • Royalty-free deployment

Supports full MATLAB language and most toolboxes
Requires MCR
Free run-time library
Royalty-free deployment
Choosing the Right Deployment Solution
MATLAB Coder and MATLAB Compiler

<table>
<thead>
<tr>
<th></th>
<th>MATLAB Coder</th>
<th>MATLAB Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output</strong></td>
<td>Portable and readable C source code</td>
<td>Executable or software component/library</td>
</tr>
<tr>
<td><strong>MATLAB support</strong></td>
<td>Subset of language Some toolboxes</td>
<td>Full language Most toolboxes Graphics</td>
</tr>
<tr>
<td><strong>Runtime requirement</strong></td>
<td>None</td>
<td>MATLAB Compiler Runtime (MCR)</td>
</tr>
<tr>
<td><strong>License model</strong></td>
<td>Royalty-free</td>
<td>Royalty-free</td>
</tr>
</tbody>
</table>
## Fixed Point Design: Motivation

<table>
<thead>
<tr>
<th>Consideration</th>
<th>Fixed Point</th>
<th>Floating Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM and ROM consumption</td>
<td>Small</td>
<td>Large</td>
</tr>
<tr>
<td>Execution time</td>
<td>Faster</td>
<td>Slower</td>
</tr>
<tr>
<td>Hardware power consumption</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Development time</td>
<td>Long</td>
<td>Short</td>
</tr>
<tr>
<td>Implementation complexity</td>
<td>More complex. Control of word length, rounding mode, saturation...</td>
<td>Less</td>
</tr>
<tr>
<td>Error Prone</td>
<td>Harder to develop. More prone to programming errors</td>
<td>Easier to develop</td>
</tr>
</tbody>
</table>
Fixed Point Design: Pitfalls

- Arithmetic Pitfalls
  - Introduces quantization errors
  - Word length and Fraction Length must be specified
    - For every variable
  - Degradation must be analyzed
Fixed Point Design: Pitfalls

- Fixed Point C Pitfalls
  - No native fixed-point math libraries
  - No built-in overflow/underflow checks
  - No tools to determine optimal integer and fractional bits
  - No visualization of floating and fixed-point representations
Fixed-Point Toolbox: MATLAB Fixed-Point Object

- Signed: true
- WordLength: 16
- FractionLength: 13

RoundMode: round
OverflowMode: saturate
ProductMode: FullPrecision
MaxProductWordLength: 128
SumMode: FullPrecision
MaxSumWordLength: 128
CastBeforeSum: true

1. Controls output type of operations
2. Allows natural operator syntax

A*B, A+B, pow2(A,3)
Fixed Point Design in MATLAB

- Collect histograms for signals
- Run MATLAB code with floating point data types
- Simulation results for all variables
- Analyze simulation min/max
Demo: Fixed Point Design in MATLAB

- Determine best fixed-point settings
- Simulate the fixed-point code
- Generate fixed-point C code
Benefits of C Code Generation with MATLAB Coder

• Generate C code directly
  – Automatically generated C code is correct by construction
  – Reduce verification effort and cost

• Maintain floating and fixed-point designs in a unified environment
  – Run simulations in double precision or fixed-point as needed
  – Validate fixed-point effects during system design phase