Automatically Convert MATLAB code to C code

Generate readable and portable C code from your MATLAB algorithms

Daryl Ning
Applications Engineer

MathWorks Australia
Level 5, Tower 1
495 Victoria Ave
CHATSWOOD NSW 2067

© 2014 The MathWorks, Inc.
Example: Euclidean distance measure

```matlab
function [y,idx,distance] = euclidean(x,codebook) %#codegen
    % Initialize minimum distance as first element of codebook
    idx=1;
    distance=norm(x-codebook(:,1));

    % Find the vector in codebook with minimum distance to x
    for index=2:size(codebook,2)
        d=norm(x-codebook(:,index));
        if d < distance
            distance=d;
            idx=index;
        end
    end

    % Output the minimum distance vector
    y=codebook(:,idx);
end
```

```c
void euclidean(const float x[20], const float codebook[2000], float y[20],
               double *idx, float *distance)
{
    float b_x[20];
    int i0;
    int b_index;
    float d;
    *idx = 1.0;
    for (i0 = 0; i0 < 20; i0++) {
        b_x[i0] = x[i0] - codebook[i0];
    }

    *distance = norm(b_x);

    /* Find the vector in codebook with minimum distance to x */
    for (b_index = 0; b_index < 99; b_index++) {
        for (i0 = 0; i0 < 20; i0++) {
            b_x[i0] = x[i0] - codebook[i0 + 20 * (b_index + 1)];
        }
        d = norm(b_x);
        if (d < *distance) {
            *distance = d;
            *idx = 2.0 + (double)b_index;
        }
    }

    /* Output the minimum distance vector */
    memcpy(&y[0], &codebook[20 * ((int)*idx - 1)1], 20U * sizeof(float));
}
```
Challenge
Develop and implement an acoustic respiratory monitoring system for wheeze detection and asthma management

Solution
Develop algorithms for detecting wheeze and ambient noise in MATLAB, and use MATLAB Coder to generate code from the algorithms for mobile devices and a web server

Results
- Manual coding effort reduced
- Algorithm development iterations accelerated
- Code maintenance overhead reduced

“MATLAB enables us to rapidly develop, debug, and test sound-processing algorithms, and MATLAB Coder simplifies the process of implementing those algorithms in C. There’s no other environment or programming language that we could use to produce similar results in the same amount of time.”

Mark Mulvey
iSonea

Link to user story
Baker Hughes
Oilfield Services Company

- Deployed a real time algorithm that optimizes the drilling process and lowers the cost of operations

- “This workflow shortened the development process by eliminating the need for maintaining and testing the same algorithm in two languages.” Dr. Christian Hansen, Baker Hughes

Agenda

- **Motivation**
  - Why translate MATLAB to C?
  - Challenges of manual translation

- **Using MATLAB Coder**
  - Three-step workflow for generating code

- **Use cases**
  - Integrate algorithms with external C code
  - Accelerate through MEX
  - Prototype by generating EXE
  - Integration with Simulink and Embedded Coder
  - Other deployment solutions

- **Summary**
## Why Engineers Translate MATLAB to C

<table>
<thead>
<tr>
<th>Extension</th>
<th>Task</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>
Algorithm Development Process

- Explore and discover
- Gain insight into problem
- Evaluate options, tradeoffs

Requirements

Research and Development

- Design
- Test
- Elaborate

Implementation

- Desktop
  - .dll
  - .exe
  - .c, .cpp
- Embedded
  - C
  - VHDL/Verilog
  - Structured Text

Test and Verification
Introductory Demo

\[ c = a \times b \]

- MATLAB Coder app
- Autodefine input type
- Code generation report

>> Demo
Challenges with Manual Translation from MATLAB to 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 process
Challenges with Manual Translation

Implementation Considerations

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

double foo(double b, double c)
{
    return 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];
            }
        }
    }
}
Challenges with Manual Translation

Implementation Considerations

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

![MATLAB and C code snippets]

7 Lines of MATLAB
105 Lines of C
Algorithm Design and Code Generation in MATLAB

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
Agenda

- Motivation
  - Why translate MATLAB to C?
  - Challenges of manual translation

- Using MATLAB Coder
  - Three-step workflow for generating code

- Use cases
  - Integrate algorithms with external C code
  - Accelerate through MEX
  - Prototype by generating EXE
  - Integration with Simulink and Embedded Coder
  - Other deployment solutions

- Summary
Using MATLAB Coder: Three-Step Workflow

**Prepare** your MATLAB algorithm for code generation

- Make implementation choices
- Use supported language features

**Test** if your MATLAB code is compliant

- Validate that MATLAB program generates code
- Iterate your MATLAB code to optimize (speed, memory, etc.)
- Verify generated code against testbench using MEX

**Generate** source code or MEX for final use

- Implement as source, executable, or library
Example: Newton/Raphson Algorithm

Preparing your MATLAB code

- Code generation readiness tool
- Pre-allocate
- Identify more efficient constructs
- Select code generation options

\[ x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}. \]

```
function [x,h] = newtonSearchAlgorithm(b,n,tol)
    % Given, "a", this function finds the nth root of a
    % number by finding where: x^n-a=0.

    notDone = 1;
    aNew   = 0; %Refined Guess Initialization
    a      = 1; %Initial Guess
    cnt    = 0;
    h(1)=a;
    while notDone
        cnt = cnt+1;
        [curVal,slope] = f_and_df(a,b,n); %square
        yint = curVal-slope*a;
        aNew = -yint/slope; %The new guess
        h(cnt)=aNew;
        if (abs(aNew-a) < tol) %Break if it's converged
            notDone = 0;
        elseif cnt>49 %after 50 iterations, stop
            notDone = 0;
            aNew = 0;
    end
end
```
MATLAB Language Support for Code Generation

Java
nested functions
cell arrays
graphics
sparse
variable-sized data
functions
numeric
arrays
System objects
fixed-point
persistent
classes
global
complex
malloc
struct
visualization
## 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 subfunctions</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></td>
</tr>
<tr>
<td>• Persistent variables</td>
<td>• Characters</td>
<td></td>
<td>Supported algorithms</td>
</tr>
<tr>
<td>• Global variables</td>
<td>• Structures</td>
<td>• More than 700 MATLAB operators and functions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Numeric class</td>
<td>• More than 300 System objects for:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Variable-sized data</td>
<td>• Signal processing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• MATLAB Class (MCOS)</td>
<td>• Communications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• System objects</td>
<td>• Computer vision</td>
<td></td>
</tr>
</tbody>
</table>

### Matrices and Arrays
- Matrix operations
- N-dimensional arrays
- Subscripting
- Frames
- Persistent variables
- Global variables

### Data Types
- Complex numbers
- Integer math
- Double/single-precision
- Fixed-point arithmetic
- Characters
- Structures
- Numeric class
- Variable-sized data
- MATLAB Class (MCOS)
- System objects

### Programming Constructs
- Arithmetic, relational, and logical operators
- Program control (if, for, while, switch)

### Functions
- MATLAB functions and subfunctions
- Variable-length argument lists
- Function handles

### Supported algorithms
- More than 700 MATLAB operators and functions
- More than 300 System objects for:
  - Signal processing
  - Communications
  - Computer vision
Supported Functions & System objects

- Aerospace Toolbox
- Communications System Toolbox
- Computer Vision System Toolbox
- DSP System Toolbox
- Image Processing Toolbox
- Phased Array System Toolbox
- Signal Processing Toolbox
- Statistics Toolbox
- Optimisation Toolbox

**Supported System objects**

- **300+** functions
- **700+** total supported functions
# Code Generation Support for Statistics Toolbox functions

Use 100+ Statistics Toolbox functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>betacdf</td>
<td>evstat</td>
<td>geornd</td>
<td>kurtosis</td>
<td>nbinrnd</td>
<td>pdf</td>
<td>tstat</td>
</tr>
<tr>
<td>betainv</td>
<td>expcdf</td>
<td>geostat</td>
<td>logncdf</td>
<td>nbinstat</td>
<td>poisscdf</td>
<td>unidcdf</td>
</tr>
<tr>
<td>betapdf</td>
<td>expinv</td>
<td>gevcdf</td>
<td>logninv</td>
<td>ncfcdf</td>
<td>poissinv</td>
<td>unidinv</td>
</tr>
<tr>
<td>betarnd</td>
<td>exppdf</td>
<td>geinv</td>
<td>lognpdf</td>
<td>ncfinv</td>
<td>poisspdf</td>
<td>unidpdf</td>
</tr>
<tr>
<td>betastat</td>
<td>exprnd</td>
<td>gevpdf</td>
<td>lognrnd</td>
<td>ncfpdf</td>
<td>poissrnd</td>
<td>unidrnd</td>
</tr>
<tr>
<td>binocdf</td>
<td>expstat</td>
<td>gevrnd</td>
<td>lognstat</td>
<td>ncfrnd</td>
<td>poisstat</td>
<td>unidstat</td>
</tr>
<tr>
<td>binoinv</td>
<td>fcdf</td>
<td>gevstat</td>
<td>mad</td>
<td>ncfstat</td>
<td>prctile</td>
<td>unifcdf</td>
</tr>
<tr>
<td>binopdf</td>
<td>finv</td>
<td>gpcdf</td>
<td>mnpdf</td>
<td>nctcdf</td>
<td>quantile</td>
<td>unifinv</td>
</tr>
<tr>
<td>binornd</td>
<td>fpdf</td>
<td>gpninv</td>
<td>moment</td>
<td>nctinv</td>
<td>randg</td>
<td>unifpdf</td>
</tr>
<tr>
<td>binostat</td>
<td>frnd</td>
<td>gppdf</td>
<td>nancov</td>
<td>nctpdf</td>
<td>random</td>
<td>unifrnd</td>
</tr>
<tr>
<td>cdf</td>
<td>fstat</td>
<td>gprnd</td>
<td>nanmax</td>
<td>nctpdf</td>
<td>raylcdf</td>
<td>unifstat</td>
</tr>
<tr>
<td>chi2cdf</td>
<td>gamcdf</td>
<td>gpstat</td>
<td>nanmean</td>
<td>nctrnd</td>
<td>raylinv</td>
<td>wblcdf</td>
</tr>
<tr>
<td>chi2inv</td>
<td>gaminv</td>
<td>harmmean</td>
<td>nanmedian</td>
<td>nctstat</td>
<td>raylpdf</td>
<td>wblinv</td>
</tr>
<tr>
<td>chi2pdf</td>
<td>gampdf</td>
<td>hygecdf</td>
<td>nanmin</td>
<td>ncx2cdf</td>
<td>raylrnd</td>
<td>wblpdf</td>
</tr>
<tr>
<td>chi2rnd</td>
<td>gamrnd</td>
<td>hygeinv</td>
<td>nanstd</td>
<td>ncx2rnd</td>
<td>raylstat</td>
<td>wblrnd</td>
</tr>
<tr>
<td>chi2stat</td>
<td>gamstat</td>
<td>hygepdf</td>
<td>nansum</td>
<td>ncx2stat</td>
<td>skewness</td>
<td>wblstat</td>
</tr>
<tr>
<td>evcdf</td>
<td>geocdf</td>
<td>hygernd</td>
<td>nanvar</td>
<td>normcdf</td>
<td>tcdf</td>
<td>zscore</td>
</tr>
<tr>
<td>evinv</td>
<td>geoinv</td>
<td>hygestat</td>
<td>nbincdf</td>
<td>norminv</td>
<td>tinv</td>
<td></td>
</tr>
<tr>
<td>evpdf</td>
<td>geomean</td>
<td>icdf</td>
<td>nbininv</td>
<td>normpdf</td>
<td>tpdf</td>
<td></td>
</tr>
<tr>
<td>evrnd</td>
<td>geopdf</td>
<td>iqr</td>
<td>nbinpdf</td>
<td>normrnd</td>
<td>trnd</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>nansum</td>
<td>normstat</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Code Generation Support for Phased Array System Toolbox**

**Use 80+ functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>aictest</td>
<td>dop2speed</td>
<td>pol2circpol</td>
<td>roty</td>
<td>val2ind</td>
</tr>
<tr>
<td>albersheim</td>
<td>dopsteeringvec</td>
<td>polellip</td>
<td>rotx</td>
<td></td>
</tr>
<tr>
<td>ambgfun</td>
<td>effearthisphere</td>
<td>polloss</td>
<td>sensorcov</td>
<td></td>
</tr>
<tr>
<td>aperture2gain</td>
<td>espritdoa</td>
<td>polratio</td>
<td>sensorsig</td>
<td></td>
</tr>
<tr>
<td>az2broadside</td>
<td>fspl</td>
<td>polsignature</td>
<td>shnidman</td>
<td></td>
</tr>
<tr>
<td>azel2phitheta</td>
<td>gain2aperture</td>
<td>pulsint</td>
<td>speed2dop</td>
<td></td>
</tr>
<tr>
<td>azel2phithetapat</td>
<td>global2localcoord</td>
<td>radareqpow</td>
<td>sph2cartvec</td>
<td></td>
</tr>
<tr>
<td>azel2uv</td>
<td>grazingang</td>
<td>radareqrng</td>
<td>spsmooth</td>
<td></td>
</tr>
<tr>
<td>azel2uvpat</td>
<td>horizonrange</td>
<td>radareqsnr</td>
<td>steevec</td>
<td></td>
</tr>
<tr>
<td>azelaxes</td>
<td>lcmvweights</td>
<td>radarvcd</td>
<td>stokes</td>
<td></td>
</tr>
<tr>
<td>beat2range</td>
<td>local2globalcoord</td>
<td>radialspeed</td>
<td>stretchfreq2rng</td>
<td></td>
</tr>
<tr>
<td>billingsleyicm</td>
<td>mdltest</td>
<td>range2beat</td>
<td>surfacegamma</td>
<td></td>
</tr>
<tr>
<td>broadside2az</td>
<td>mvdrweights</td>
<td>range2bw</td>
<td>surfclutterrcs</td>
<td></td>
</tr>
<tr>
<td>cart2sphvec</td>
<td>noisepow</td>
<td>range2time</td>
<td>systemp</td>
<td></td>
</tr>
<tr>
<td>cbfweights</td>
<td>npwgntresh</td>
<td>rangeangle</td>
<td>time2range</td>
<td></td>
</tr>
<tr>
<td>circpol2pol</td>
<td>phitheta2azel</td>
<td>rdcoupling</td>
<td>unigrid</td>
<td></td>
</tr>
<tr>
<td>dechirp</td>
<td>phitheta2azelpat</td>
<td>rocpfa</td>
<td>uv2azel</td>
<td></td>
</tr>
<tr>
<td>delayseq</td>
<td>phitheta2uv</td>
<td>rocsnr</td>
<td>uv2azelpat</td>
<td></td>
</tr>
<tr>
<td>depressionang</td>
<td>phitheta2uvpat</td>
<td>rootmusicdoa</td>
<td>uv2phitheta</td>
<td></td>
</tr>
<tr>
<td></td>
<td>physconst</td>
<td>rotx</td>
<td>uv2phithetapat</td>
<td></td>
</tr>
</tbody>
</table>
Code Generation Support for Phased Array System Toolbox

Use 50+ System objects

- phased.CosineAntennaElement
- phased.CrossedDipoleAntennaElement
- phased.CustomAntennaElement
- phased.CustomMicrophoneElement
- phased.IsotropicAntennaElementIsotropic
- phased.OmnidirectionalMicrophoneElement
- phased.ShortDipoleAntennaElement
- phased.ULA
- phased.URA
- phased.ConformalArray
- phased.PartitionedArray
- phased.ReplicatedSubarray
- phased.SteeringVector
- phased.ArrayGain
- phased.ArrayResponse
- phased.ElementDelay
- phased.Collector
- phased.Radiator
- phased.WidebandCollector
- phased.LinearFMWaveformLinear
- phased.PhaseCodedWaveform
- phased.RectangularWaveform
- phased.SteppedFMWaveform
- phased.FMCWWaveform
- phased.MatchedFilter
- phased.Transmitter
- phased.ReceiverPreamp
- phased.PhaseShiftBeamformer
- phased.LCMVBeamformer
- phased.MVDRBeamformer
- phased.SubbandPhaseShiftBeamformer
- phased.FrostBeamformer
- phased.TimeDelayBeamformer
- phased.TimeDelayLCMVBeamformer
- phased.SteeringVector
- phased.SumDifferenceMonopulseTracker
- phased.SumDifferenceMonopulseTracker2D
- phased.BeamscanEstimator
- phased.BeamscanEstimator2D
- phased.MVDREstimator
- phased.MVDREstimator2D
- phased.RootMUSICEstimator
- phased.RootWSFEstimator
- phased.ESPRITEstimator
- phased.BeamspaceESPRITEstimator
- phased.STAPSMIBeamformer
- phased.DPCACanceller
- phased.ADPCACanceller
- phased.AngleDopplerResponse
- phased.CFARDetector
- phased.MatchedFilter
- phased.RangeDopplerResponse
- phased.StretchProcessor
- phased.TimeVaryingGain
- phased.FreeSpace
- phased.RadarTarget
- phased.ConstantGammaClutter
- phased.BarrageJammer
- phased.Platform
Agenda

- **Motivation**
  - Why translate MATLAB to C?
  - Challenges of manual translation

- **Using MATLAB Coder**
  - Three-step workflow for generating code

- **Use cases**
  - Integrate algorithms with external C code
  - Accelerate through MEX
  - Prototype by generating EXE
  - Integration with Simulink and Embedded Coder
  - Other deployment solutions

- **Summary**
MATLAB Coder Use Cases

- **Integrate**
  - algorithms with custom software

- **Prototype**
  - algorithms on PCs

- **Accelerate**
  - algorithm execution

- **Implement**
  - algorithms on embedded processors

Extensions:
- .lib
- .dll
- .exe
- .mex
Example: Code Integration with Visual Studio Parent Project

MATLAB

Visual Studio C/C++

>> Demo
Example: External Code Integration

coder.ceval

Integrate third-party libraries with generated code

- Uses coder.target to distinguish between MATLAB simulation and code generation
- Integrates custom C code to replace automatically generated C code
- Just need to specify additional files and include path

```matlab
function y = custom_conv_ceval(x,h)     %#codegen
% This example uses the coder.target and coder.ceval functions to
% explicitly define what should be run during simulation, and what should
% be used when generating C code.

% Initialise size and type of output
y = zeros(length(x)+length(h)-1,1);

if coder.target('MATLAB')
    % Executing in MATLAB
    y = conv(x,h);
else
    % Call to C function 'custom_conv.c' using coder.ceval
    coder.ceval('custom_conv', coder.rref(x), uint32(length(x)), ...
                coder.rref(h), uint32(length(h)), ...
                coder.wref(y));
end
```

```c
void custom_conv_ceval(const double x[10], const double h[3], double y[12])
{
    /* Initialise size and type of output */
    /* Call to C function 'custom_conv.c' using coder.ceval */
    custom_conv(x, 10U, h, 3U, y);
}
```
External Code Integration using
\texttt{coder.ExternalDependency}

Integrate third-party libraries with generated code

- Encapsulates API to an external library, object file, or C/C++ source code
- Integrates with external libraries without user intervention
- Automatically adds necessary compiler and linker flags and objects
Acceleration Strategies

- Better algorithms
  Matrix inversion vs. QR or SVD
  - Different approaches to solving the same problem

- More efficient implementation
  Hand-coded vs. optimized library (e.g. BLAS and LAPACK)
  - Different optimization of the same algorithm

- More computational resources
  Single-threaded vs. multithreaded (multithreaded BLAS)
  - Leveraging additional processors, cores, GPUs, FPGAs, etc.
Accelerating Algorithm Execution

User’s code

```
for k=1:max
  x = fft(dat)
  y = 20*log1
```

Optimize MATLAB code

Parallel computing

System objects

Custom code using MEX

MATLAB to C
Example: Newton-Raphson

```matlab
%% Script to compare execution time of MATLAB code to generated MEX code
iter = 1000;

%% Time MATLAB code
e1Time = zeros(iter,1);
for i = 1:iter
tic
  nrt(1e8,12,1e-9);
t = toc;
e1Time(i) = t;
end
matTime = mean(e1Time);
disp([\'Mean MATLAB time is: '' num2str(matTime) ' seconds\']);

%% Time MEX Code
e2Time = zeros(iter,1);
for i = 1:iter
tic
  nrt_mex(1e8,12,1e-9);
t = toc;
e2Time(i) = t;
end
mexTime = mean(e2Time);
disp([\'Mean MEX time is: '' num2str(mexTime) ' seconds\']);

%% Speed Up Factor
speedUp = matTime/mexTime;
disp([\'Speed up factor is: '' num2str(speedUp) 'X\']);
```

>> speedTest
Mean MATLAB time is: 0.00033502 seconds
Mean MEX time is: 2.8008e-05 seconds
Speed up factor is: 11.9615X

>> Demo
Acceleration Using MEX

- Speed-up factor will vary

When you **may** see a speedup:
  - Often for communications and signal processing
  - Always for fixed point
  - Likely for loops with states or when vectorisation isn’t possible

When you **may not** see a speedup:
  - MATLAB implicitly multithreads computation.
  - Built-in functions call IPP or BLAS libraries.
Multicore `parfor` Support in MEX Functions

Run MATLAB faster by generating MEX functions that execute on multiple cores

- Relies on OpenMP technology to parallelize `parfor` loops
- OpenMP supported by Microsoft, Intel, and GCC C compilers
Multicore `parfor` Support for Standalone Code Generation

Use `parfor` to generate parallel C/C++ code using OpenMP

- Requires C/C++ compiler supporting OpenMP

```matlab
parfor plane = 1:sz(3)
    planeHist = zeros(1,L);
    for y = 1:N
        for x = 1:M
            r = originalImage(y,x,plane);
            planeHist(r+1) = planeHist(r+1) + 1;
        end
    end
    originalHist(plane,:) = planeHist;
end
```

```c
#pragma omp parallel for
num_threads(omp_get_max_threads()) \ 
private(planeHist_data,loop_ub,10,y,x,il)|
for (plane = 0; plane < 3; plane++) {
    loop_ub = (int)L;
    for (i0 = 0; i0 < loop_ub; i0++) {
        planeHist_data[i0] = 0.0;
    }
}
```
Example: Standalone Executable

Video Stabilisation

- Need to provide `main.c` for entry point
- Use System objects from Computer Vision System Toolbox to stream, process, and display video

>> Demo
Working with Embedded Coder

- Advanced support for MATLAB Coder, including:
  - Speed
  - Memory
  - Code appearance
  - Hardware-specific optimization
Working with Embedded Coder

Software-in-the-Loop Verification

Verify numerical behavior of generated source code through software-in-the-loop testing

- Reuse MATLAB tests to exercise standalone source code compiled for host computer

- Integrate SIL verification with the existing Project verification tool and command-line utility `coder.runTest`

- Step through generated code in Microsoft Visual Studio debugger during SIL testing when using `coder.runTest`
Working with Simulink

MATLAB Function block in Simulink

---

```matlab
persistent F;
persistent xhat;
if isempty(F)
    xhat = [0.001; 0.01; 0.001; 400];
    P = zeros(4);
end

% Radar update time deltat is inherited from model

% 1. Compute Phi, Q, and R
Phi = [1 deltat 0 0; 0 1 0 0; 0 0 1 deltat; 0 0 0 0];
Q = diag([0.005 0.005]);
R = diag([300^2 0.001^2]);

% 2. Propagate the covariance matrix:
P = Phi'P*Phi + Q;

% 3. Propagate the track estimate:
   xhat = Phi*xhat;

% 4a. Compute observation estimates:
   % ...
```
Other Desktop Deployment Options

- Explore and discover
- Gain insight into problem
- Evaluate options, tradeoffs

Desktop
- .dll
- .exe
- .c, .cpp

Embedded
- C
- VHDL/Verilog
- Structured Text

Requirements

Research and Design

Design
Test
Elaborate
MATLAB Compiler

MATLAB Coder

MATLAB

for k=1:max
  x = fft(dat);
  y = 20*log10;
end

Matlab Compiler Runtime (MCR)
Other Deployment Options

**MATLAB Compiler**

- **Share applications**
  - Creates desktop or web software components
  - Supports full MATLAB language and most toolboxes
  - Requires MATLAB Compiler Runtime
    - Free run-time library
    - Royalty-free deployment
## Choosing the Right Deployment Solution

### MATLAB Coder or 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</td>
<td>Full language</td>
</tr>
<tr>
<td></td>
<td>Some toolboxes</td>
<td>Most toolboxes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Graphics</td>
</tr>
<tr>
<td><strong>Runtime requirement</strong></td>
<td>None</td>
<td>MATLAB Compiler Runtime</td>
</tr>
<tr>
<td><strong>License model</strong></td>
<td>Royalty-free</td>
<td>Royalty-free</td>
</tr>
</tbody>
</table>
VivaQuant Accelerates Development and Validation of Embedded Device for Ambulatory ECG Sensing

Challenge
Design and implement an embedded system for extracting accurate information from noisy electrocardiogram signals

Solution
Use MATLAB to develop an algorithm for removing in-band noise, and use Fixed-Point Designer and MATLAB Coder to implement it on an ARM Cortex-M series processor

Results
- Development accelerated by 300%
- Power and memory consumption minimized
- Rigorous testing enabled

“MATLAB, MATLAB Coder, and Fixed-Point Designer enabled our small team to develop a complex real-time signal processing algorithm, optimize it to reduce power and memory requirements, accelerate embedded code implementation, and perform the rigorous testing required for medical device validation.”

Marina Brockway
VivaQuant

Link to user story
Agenda

- Motivation
  - Why translate MATLAB to C?
  - Challenges of manual translation

- Using MATLAB Coder
  - Three-step workflow for generating code

- Use cases
  - Integrate algorithms with external C code
  - Accelerate through MEX
  - Prototype by generating EXE
  - Integration with Simulink and Embedded Coder
  - Other deployment solutions

- Summary
Summary

Access
- Files
- Software
- Hardware

Explore & Discover
- Data Analysis & Modeling
- Algorithm Development
- Application Development

Share
- Reporting and Documentation
- Outputs for Design
- Deployment

Automate
Automatic C Code Generation

Accelerates Development

MathWorks

Develop algorithm  Test  Convert to C/C++  Test  Iterate  Test  Deploy

Algorithm Design and Code Generation in MATLAB

Iterate

verify / accelerate

MEX

.c, cpp

.exe

.lib

Time savings

Generate code
Takeaways

- MATLAB provides a direct path to C code
  - Both floating-point and fixed-point

- Suitable for applications where
  - Source code is required
  - Small memory footprint is required

- Automatic Code Generation
  - accelerates design iterations
  - reduces verification effort
More Information

- To learn more, visit the product page: 
  mathworks.com/products/matlab-coder

- To request a trial license:
  - Talk to your MathWorks account manager to request a trial license and set up a guided evaluation with an application engineer

- Contact us
  - info@mathworks.com.au
  - 02 8669 4700
# Training courses - Sydney

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Code</th>
<th>Start Date</th>
<th>End Date</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>5-Aug-14</td>
<td>7-Aug-14</td>
<td>Sydney</td>
</tr>
<tr>
<td>Stateflow for Logic Driven System Modeling</td>
<td>SLSF</td>
<td>14-Aug-14</td>
<td>15-Aug-14</td>
<td>Sydney</td>
</tr>
<tr>
<td>Embedded Coder for Production Code Generation</td>
<td>SLEC</td>
<td>16-Sep-14</td>
<td>18-Sep-14</td>
<td>Sydney</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>30-Sep-14</td>
<td>2-Oct-14</td>
<td>Sydney</td>
</tr>
<tr>
<td>MATLAB Programming Techniques</td>
<td>MLPR</td>
<td>14-Oct-14</td>
<td>15-Oct-14</td>
<td>Sydney</td>
</tr>
<tr>
<td>Statistical Methods in MATLAB</td>
<td>MLST</td>
<td>11-Nov-14</td>
<td>12-Nov-14</td>
<td>Sydney</td>
</tr>
<tr>
<td>Image Processing with MATLAB</td>
<td>MLIP</td>
<td>13-Nov-14</td>
<td>14-Nov-14</td>
<td>Sydney</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Code</th>
<th>Start Date</th>
<th>End Date</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB Programming Techniques</td>
<td>MLPR</td>
<td>12-Aug-14</td>
<td>13-Aug-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>2-Sep-14</td>
<td>4-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>Simulink for System and Algorithm Modeling</td>
<td>SLBE</td>
<td>11-Sep-14</td>
<td>12-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>9-Dec-14</td>
<td>11-Dec-14</td>
<td>e-learning</td>
</tr>
</tbody>
</table>
# Training courses - Adelaide

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Code</th>
<th>Start Date</th>
<th>End Date</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulink for System and Algorithm Modeling</td>
<td>SLBE</td>
<td>21-Aug-14</td>
<td>22-Aug-14</td>
<td>Adelaide</td>
</tr>
<tr>
<td>Image Processing with MATLAB</td>
<td>MLIP</td>
<td>9-Sep-14</td>
<td>10-Sep-14</td>
<td>Adelaide</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>4-Nov-14</td>
<td>6-Nov-14</td>
<td>Adelaide</td>
</tr>
<tr>
<td>Parallel Computing with MATLAB</td>
<td>MLPC</td>
<td>18-Nov-14</td>
<td>19-Nov-14</td>
<td>Adelaide</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Code</th>
<th>Start Date</th>
<th>End Date</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB Programming Techniques</td>
<td>MLPR</td>
<td>12-Aug-14</td>
<td>13-Aug-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>2-Sep-14</td>
<td>4-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>Simulink for System and Algorithm Modeling</td>
<td>SLBE</td>
<td>11-Sep-14</td>
<td>12-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>9-Dec-14</td>
<td>11-Dec-14</td>
<td>e-learning</td>
</tr>
</tbody>
</table>
Training courses - Brisbane

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Code</th>
<th>Start Date</th>
<th>End Date</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimization Techniques in MATLAB</td>
<td>MLOP</td>
<td>24-Oct-14</td>
<td>24-Oct-14</td>
<td>Brisbane</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>11-Nov-14</td>
<td>13-Nov-14</td>
<td>Brisbane</td>
</tr>
<tr>
<td>MATLAB Programming Techniques</td>
<td>MLPR</td>
<td>12-Aug-14</td>
<td>13-Aug-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>2-Sep-14</td>
<td>4-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>Simulink for System and Algorithm Modeling</td>
<td>SLBE</td>
<td>11-Sep-14</td>
<td>12-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>9-Dec-14</td>
<td>11-Dec-14</td>
<td>e-learning</td>
</tr>
</tbody>
</table>
Training courses - Melbourne

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Code</th>
<th>Start Date</th>
<th>End Date</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>19-Aug-14</td>
<td>21-Aug-14</td>
<td>Melbourne</td>
</tr>
<tr>
<td>Building Interactive Applications in MATLAB</td>
<td>MLGU</td>
<td>22-Aug-14</td>
<td>22-Aug-14</td>
<td>Melbourne</td>
</tr>
<tr>
<td>Physical Modeling of Multidomain Systems with Simscape</td>
<td>SLMPM-S</td>
<td>22-Sep-14</td>
<td>22-Sep-14</td>
<td>Melbourne</td>
</tr>
<tr>
<td>MATLAB Programming Techniques</td>
<td>MLPR</td>
<td>23-Sep-14</td>
<td>24-Sep-14</td>
<td>Melbourne</td>
</tr>
<tr>
<td>Statistical Methods in MATLAB</td>
<td>MLST</td>
<td>25-Sep-14</td>
<td>26-Sep-14</td>
<td>Melbourne</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>14-Oct-14</td>
<td>16-Oct-14</td>
<td>Melbourne</td>
</tr>
<tr>
<td>Optimization Techniques in MATLAB</td>
<td>MLOP</td>
<td>17-Oct-14</td>
<td>17-Oct-14</td>
<td>Melbourne</td>
</tr>
<tr>
<td>Parallel Computing with MATLAB</td>
<td>MLPC</td>
<td>28-Oct-14</td>
<td>29-Oct-14</td>
<td>Melbourne</td>
</tr>
<tr>
<td>Signal Processing with MATLAB</td>
<td>MLSG</td>
<td>30-Oct-14</td>
<td>31-Oct-14</td>
<td>Melbourne</td>
</tr>
<tr>
<td>Signal Processing with Simulink</td>
<td>SLBE-G</td>
<td>18-Nov-14</td>
<td>20-Nov-14</td>
<td>Melbourne</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Code</th>
<th>Start Date</th>
<th>End Date</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB Programming Techniques</td>
<td>MLPR</td>
<td>12-Aug-14</td>
<td>13-Aug-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>2-Sep-14</td>
<td>4-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>Simulink for System and Algorithm Modeling</td>
<td>SLBE</td>
<td>11-Sep-14</td>
<td>12-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>9-Dec-14</td>
<td>11-Dec-14</td>
<td>e-learning</td>
</tr>
</tbody>
</table>
## Training courses - Perth

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Code</th>
<th>Start Date</th>
<th>End Date</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistical Methods in MATLAB</td>
<td>MLST</td>
<td>26-Aug-14</td>
<td>27-Aug-14</td>
<td>Perth</td>
</tr>
<tr>
<td>Simulink for System and Algorithm Modeling</td>
<td>SLBE</td>
<td>28-Aug-14</td>
<td>29-Aug-14</td>
<td>Perth</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>23-Sep-14</td>
<td>25-Sep-14</td>
<td>Perth</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Code</th>
<th>Start Date</th>
<th>End Date</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB Programming Techniques</td>
<td>MLPR</td>
<td>12-Aug-14</td>
<td>13-Aug-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>2-Sep-14</td>
<td>4-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>Simulink for System and Algorithm Modeling</td>
<td>SLBE</td>
<td>11-Sep-14</td>
<td>12-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>9-Dec-14</td>
<td>11-Dec-14</td>
<td>e-learning</td>
</tr>
</tbody>
</table>
## Training courses - New Zealand

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Code</th>
<th>Start Date</th>
<th>End Date</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Processing with Simulink</td>
<td>SLBE-G</td>
<td>3-Sep-14</td>
<td>5-Sep-14</td>
<td>Christchurch</td>
</tr>
<tr>
<td>Statistical Methods in MATLAB</td>
<td>MLST</td>
<td>7-Oct-14</td>
<td>8-Oct-14</td>
<td>Wellington</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Code</th>
<th>Start Date</th>
<th>End Date</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB Programming Techniques</td>
<td>MLPR</td>
<td>12-Aug-14</td>
<td>13-Aug-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>2-Sep-14</td>
<td>4-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>Simulink for System and Algorithm Modeling</td>
<td>SLBE</td>
<td>11-Sep-14</td>
<td>12-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>9-Dec-14</td>
<td>11-Dec-14</td>
<td>e-learning</td>
</tr>
</tbody>
</table>

## Training courses - OnLine

<table>
<thead>
<tr>
<th>Course Name</th>
<th>Course Code</th>
<th>Start Date</th>
<th>End Date</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB Programming Techniques</td>
<td>MLPR</td>
<td>12-Aug-14</td>
<td>13-Aug-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>2-Sep-14</td>
<td>4-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>Simulink for System and Algorithm Modeling</td>
<td>SLBE</td>
<td>11-Sep-14</td>
<td>12-Sep-14</td>
<td>e-learning</td>
</tr>
<tr>
<td>MATLAB Fundamentals</td>
<td>MLBE</td>
<td>9-Dec-14</td>
<td>11-Dec-14</td>
<td>e-learning</td>
</tr>
</tbody>
</table>


MathWorks®