Generate C/C++ Code from a MATLAB Function

This example shows the recommended workflow for generating C/C++ code from a MATLAB® function. The steps in this workflow are:

Copy Command

1. Prepare MATLAB code for code generation.
2. Generate and test MEX function.
3. Generate and inspect C/C++ code.

This example generates C/C++ code at the command line using the `codegen` command. To learn how to generate code using the MATLAB Coder app, see Generate C Code by Using the MATLAB Coder App.

Create MATLAB Code and Sample Data

This step is necessary for the purposes of this example and is not a typical step in the code generation workflow.

Create a MATLAB function `averagingFilterML` that acts as an averaging filter on an input signal. This function takes an input vector of signal values and returns an output vector of filtered values. The output vector is the same size as the input vector. The `averagingFilterML` function uses the variable `slider` to represent a sliding window of 16 signal values and calculates the average signal value for each window position.

```
type averagingFilterML
```

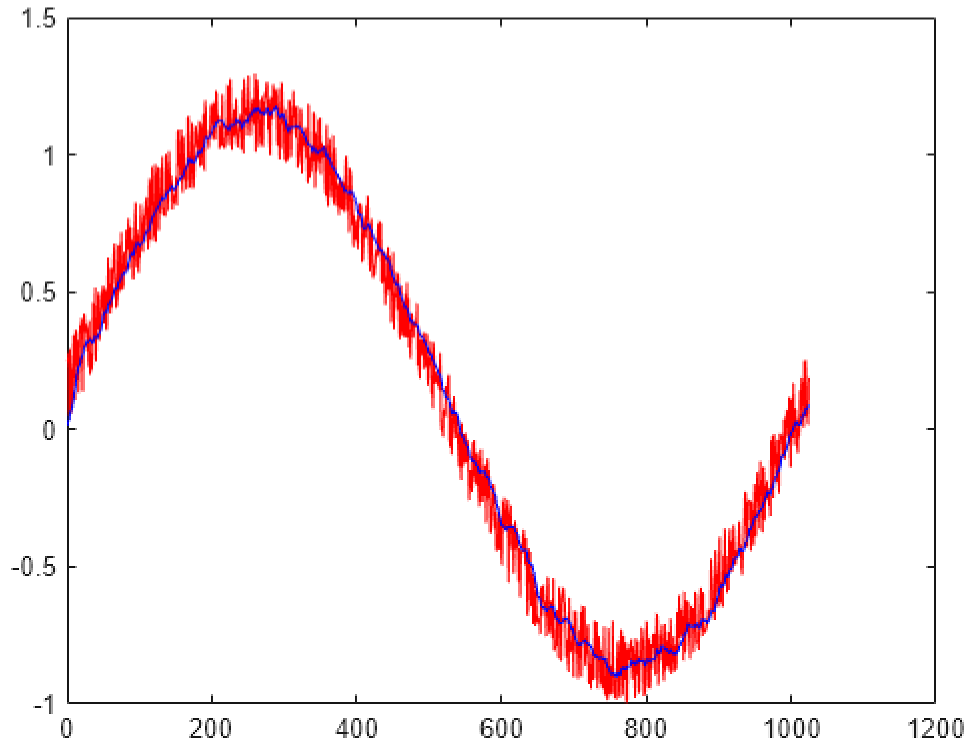
Get

```
function y = averagingFilterML(x)
slider = zeros(16,1);
y = zeros(size(x));
for i = 1:numel(x)
    slider(2:end) = slider(1:end-1); % move one position in the buffer
    slider(1) = x(i); % Add a new sample value to the buffer
    y(i) = sum(slider)/numel(slider); % write the average of the current window to y
end
end
```

Generate a noisy sine wave as sample data, and use `averagingFilterML` to filter the noisy data. Plot the noisy data and the filtered data in the same figure window.

Get

```
v = 0:0.00614:2*pi;
x = sin(v) + 0.3*rand(1,numel(v));
y = averagingFilterML(x);
plot(x,"red");
hold on
plot(y,"blue");
hold off;
```



Step 1: Prepare MATLAB Code for Code Generation

Rename the `averagingFilterML` function to `averagingFilterCG`. Add the `%#codegen` directive to `averagingFilterCG` to prompt the MATLAB Code Analyzer to identify warnings and errors specific to code generation. For code generation, input variable types must be defined. Specify the input as an unbounded vector of `doubles` using an `arguments` block.

```
type averagingFilterCG
```

Get

```
function y = averagingFilterCG(x) %#codegen
arguments
    x (1,:) double
end
slider = zeros(16,1);
y = zeros(size(x));
for i = 1:numel(x)
    slider(2:end) = slider(1:end-1); % move one position in the buffer
    slider(1) = x(i); % Add a new sample value to the buffer
    y(i) = sum(slider)/numel(slider); % write the average of the current window to y
end
end
```

Step 2: Generate and Test MEX Function

It is important to generate and test a MEX function before you generate C/C++ code. Running the MEX function in MATLAB before generating C/C++ code enables you to detect and fix run-time errors that are much harder to diagnose in the generated code. In addition, you can use the MEX function to verify that your generated code functions similarly to your original MATLAB code.

Use the `codegen` command to generate a MEX function from `averagingFilterCG`. Test the MEX function with the same input that you passed to the original MATLAB function and compare the results. The MEX function produces the same output.

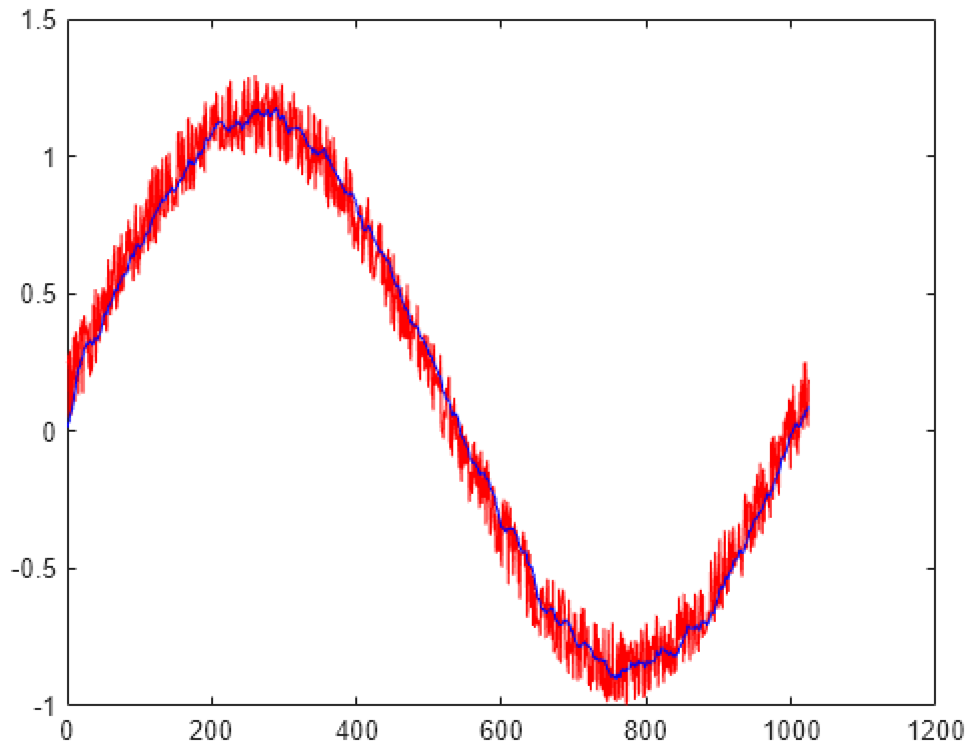
```
codegen averagingFilterCG
```

Get

```
Code generation successful.
```

```
z = averagingFilterCG_mex(x);
plot(x,"red");
hold on
plot(z,"blue");
hold off;
```

Get



Step 3: Generate and Inspect C/C++ Code

Use the `codegen` command with the `-config:lib` option to generate a standalone C library. Inspect the `averagingFilterCG` function in the generated C code.

```
codegen -config:lib averagingFilterCG
```

Get

```
Code generation successful.
```

```
type(fullfile("codegen","lib","averagingFilterCG","averagingFilterCG.c"))
```

Get

```
/*
 * Prerelease License - for engineering feedback and testing purposes
 * only. Not for sale.
 * File: averagingFilterCG.c
 *
 * MATLAB Coder version            : 24.2
 * C/C++ source code generated on  : 20-Jul-2024 12:20:51
 */

/* Include Files */
#include "averagingFilterCG.h"
#include "averagingFilterCG_emxutil.h"
#include "averagingFilterCG_types.h"
#include <string.h>

/* Function Definitions */
/*
 * Arguments    : const emxArray_real_T *x
 *                emxArray_real_T *y
 * Return Type  : void
 */
void averagingFilterCG(const emxArray_real_T *x, emxArray_real_T *y)
{
  double slider[16];
  double b_slider[15];
  const double *x_data;
  double *y_data;
  int i;
  int k;
  int loop_ub;
  x_data = x->data;
  memset(&slider[0], 0, 16U * sizeof(double));
  i = y->size[0] * y->size[1];
  y->size[0] = 1;
  y->size[1] = x->size[1];
  emxEnsureCapacity_real_T(y, i);
  y_data = y->data;
  loop_ub = x->size[1];
```

```
    for (i = 0; i < loop_ub; i++) {
      y_data[i] = 0.0;
    }
    i = x->size[1];
    for (loop_ub = 0; loop_ub < i; loop_ub++) {
      double b_y;
      memcpy(&b_slider[0], &slider[0], 15U * sizeof(double));
      /*  move one position in the buffer */
      b_y = x_data[loop_ub];
      slider[0] = b_y;
      /*  Add a new sample value to the buffer */
      for (k = 0; k < 15; k++) {
        double d;
        d = b_slider[k];
        slider[k + 1] = d;
        b_y += d;
      }
      y_data[loop_ub] = b_y / 16.0;
      /*  write the average of the current window to y */
    }
  }

  /*
   * File trailer for averagingFilterCG.c
   *
   * [EOF]
   */
```

Alternatively, use the codegen command with the `-config:lib` and `-lang:C++` options to generate a standalone C++ library. Compare the `averagingFilterCG` function in the generated C++ code to that in the generated C code.

```
codegen -config:lib -lang:c++ averagingFilterCG
```

Get

```
Code generation successful.
```

```
type(fullfile("codegen","lib","averagingFilterCG","averagingFilterCG.cpp"))
```

Get

```
//
// Prerelease License - for engineering feedback and testing purposes
// only. Not for sale.
// File: averagingFilterCG.cpp
//
// MATLAB Coder version            : 24.2
// C/C++ source code generated on  : 20-Jul-2024 12:20:55
//

// Include Files
#include "averagingFilterCG.h"
#include "coder_array.h"
#include <algorithm>
#include <cstring>

// Function Definitions
//
// Arguments    : const coder::array<double, 2U> &x
//                coder::array<double, 2U> &y
// Return Type  : void
//
void averagingFilterCG(const coder::array<double, 2U> &x,
                       coder::array<double, 2U> &y)
{
  double slider[16];
  double b_slider[15];
  int i;
  int loop_ub;
  std::memset(&slider[0], 0, 16U * sizeof(double));
  y.set_size(1, x.size(1));
  loop_ub = x.size(1);
  for (i = 0; i < loop_ub; i++) {
    y[i] = 0.0;
  }
  i = x.size(1);
  for (loop_ub = 0; loop_ub < i; loop_ub++) {
    double b_y;
    std::copy(&slider[0], &slider[15], &b_slider[0]);
```

```
      //  move one position in the buffer
      b_y = x[loop_ub];
      slider[0] = b_y;
      //  Add a new sample value to the buffer
      for (int k{0}; k < 15; k++) {
        double d;
        d = b_slider[k];
        slider[k + 1] = d;
        b_y += d;
      }
      y[loop_ub] = b_y / 16.0;
      //  write the average of the current window to y
    }
  }

  //
  // File trailer for averagingFilterCG.cpp
  //
  // [EOF]
  //
```

See Also

codegen

Related Examples

- Generate C Code by Using the MATLAB Coder App
- Generate C Code at the Command Line

More About

- Code Generation Guide: Generate Deployable C/C++ Code
- Generate C/C++ Code From MATLAB Code Using MATLAB Coder