CODEGEN: CONSTANT INPUTS

BACKGROUND

When a computer stores space for a variable of unknown size, it cannot allocate a specific allotment of space on the disk. Meaning that the computer has to be prepared to store a variable of unknown size and type. By pre-defining these inputs, the computer can fix the space allotted to each value. Thereby, reducing at least one element of computation. These pre-defined inputs are known as *constant* inputs. Algorithms use them as flags during execution to minimize overhead.

In the culmination of the examples provided, you will create a custom function using MEX for code generation. When generating code, you must specify that the first input has the same type as the second. Then using coder.Constant, specify the value of the second input.

CREATING CUSTOM FUNCTION USING CONSTANT INPUTS VIA MEX FUNCTION

Using a custom MATLAB function, define the default constant inputs for that function within the Commandline interface.

1. Using the **codegen** command in the Commandline interface, utilize the **-args** option to define the constants.

   Example Input:

   Write a MATLAB function, *myadd.m*

   ```
   function y = mcadd(u,v) %#codegen
       y = u + v;
   % mcadd has two inputs, u and v.
   ```

2. Create a configuration object for MEX code generation, using command cfg = coder.config(‘mex’) in the Commandline interface.

   Example Input:

   ```
   cfg = coder.config(‘mex’)
   codegen -config cfg -args {2, coder.Constant(42)} mcadd
   mcadd(2,42)
   ```

   Example Output:

   ```
   ans =
      46
   ```

CONTROLLING MEX FUNCTION

You can control whether a generated MEX function signature will include constant inputs. If you will be using the same test file to run the original MATLAB® function and the MEX function, then the MEX function signature must contain the constant inputs. You can also control whether the run-time values of the constant inputs must match the compile-time values. Checking for matching values can slow down execution speed.

The configuration object property, *ConstantInputs*, controls whether you will have to provide a value for the input that is constant when you call the MEX function.

1. Set *ConstantInputs* to *CheckValues* to force the values to be the default values.

   Example Input:

   ```
   cfg.ConstantInputs = ‘CheckValues';
   mcadd_mex(4, 42)
   ```

   Example Output:

   ```
   ans =
   46
   ```

2. To allow the default inputs to be overridden, set *ConstantInputs* to *IgnoreValues.*

   Example Input:

   ```
   cfg.ConstantInputs = ‘IgnoreValues';
   mcadd_mex(4, 142)
   ```

   Example Output:

   ```
   ans =
   146
   ```

3. To avoid having to include the constant values every time the MEX generated function is used, set *ConstantInputs* to *Remove*.

   Example Input:

   ```
   cfg.ConstantInputs = ‘Remove';
   mcadd_mex(4)
   mcadd_mex(3)
   ```

   Example Output:

   ```
   ans =
   46
   ans =
   45
   ```

OBSERVATIONS & QUESTIONS

OBSERVATIONS

1. The notes are repetitive, not very meaningful, and at times require the writer to make assumptions on the coding techniques required.
2. The whole document assumes we are working within the MATLAB Commandline interface.
3. From the perspective of a Software Engineer, it seems like the engineer described a bunch of features at random without explaining what the system does. I would suspect that the engineer did not comment out their code nor construct the customary text explaining in detail how that part of the code works.
4. Often properties are provided in tables or a similar format to allow developers to quickly understand the scope of what a function can do.

QUESTIONS

1. What are MathWorks' style guide guidelines for defining properties of a function?
2. The document explains how to avoid *not* redefining constant values and then teaches the user how to turn off that behavior in the functions. Why?
3. What are the specific interactions of the MEX code generating function?
4. We assume that after entering the **codegen** code into the Commandline with a custom MATLAB function that the generated function is [custom_code_name]_mex (i.e. mcadd becomes mcadd_mex). Why is this? What is the behavior that allows this? Are there exceptions?
5. How does this functionality influence the system as a whole? What's the bigger picture? How does it impact calculations involving big data?
6. Once we have implemented this task, happens next?
7. Are there any other values for cfg.*ConstantInputs*?