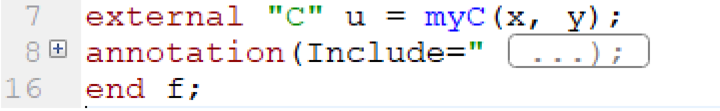
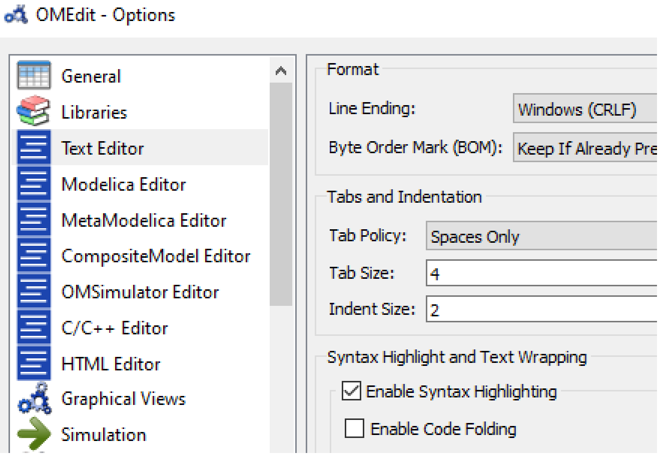
External Functions – Exercises

- Open ExternalFunctions.mo
- Complete the C definition of function f returning addition of inputs x and y
  Hint: add C code line: `return x+y;`



```
external "C" u = myC(x, y);
annotation(Include=" ...);
end f;
```

- Hint: to see the C code click on the +
  or in Tools->Options disable code folding



OMEdit - Options

Format

Line Ending: Windows (CRLF)

Byte Order Mark (BOM): Keep If Already Pre

Tabs and Indentation

Tab Policy: Spaces Only

Tab Size: 4

Indent Size: 2

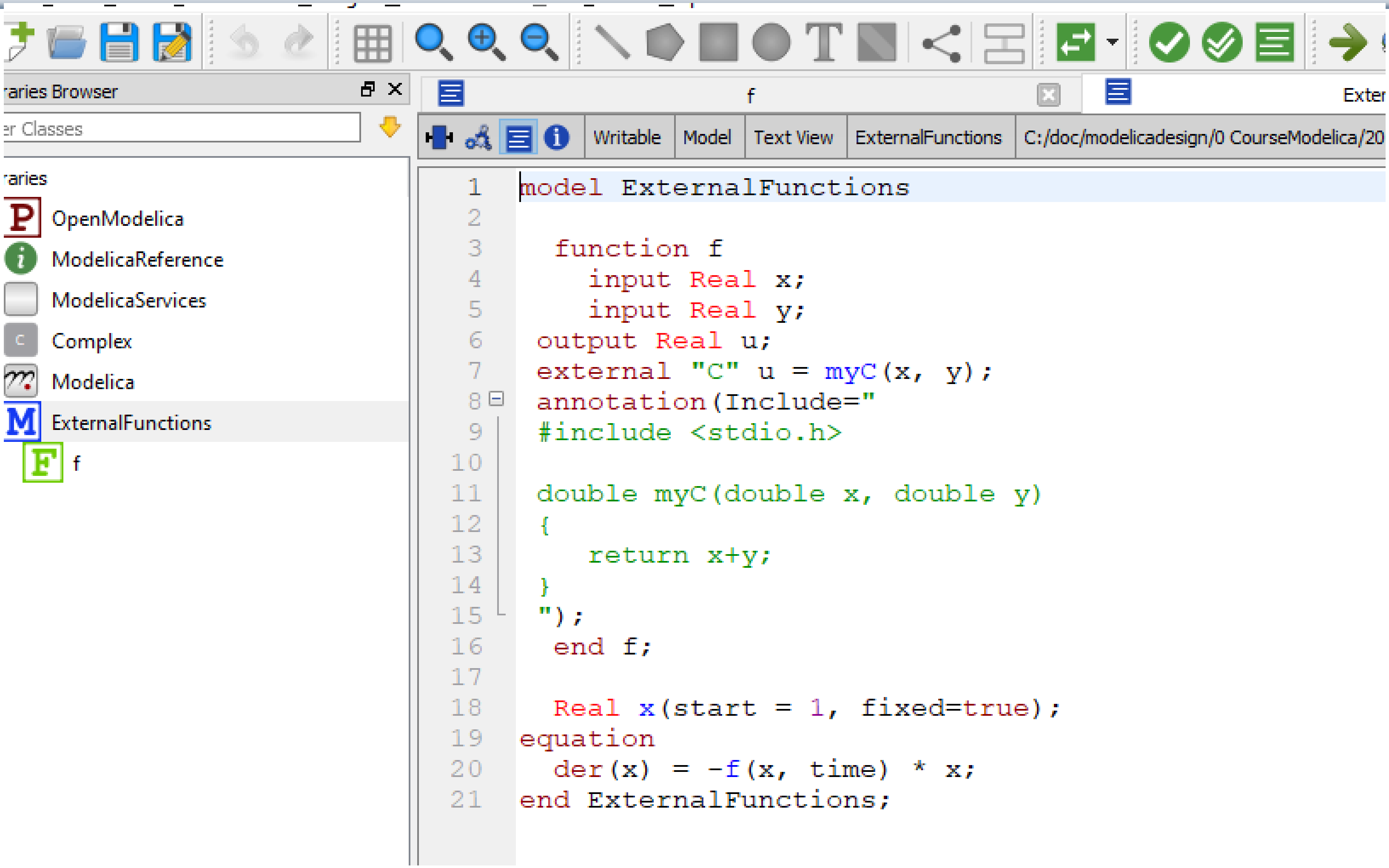
Syntax Highlight and Text Wrapping

☑ Enable Syntax Highlighting

☐ Enable Code Folding

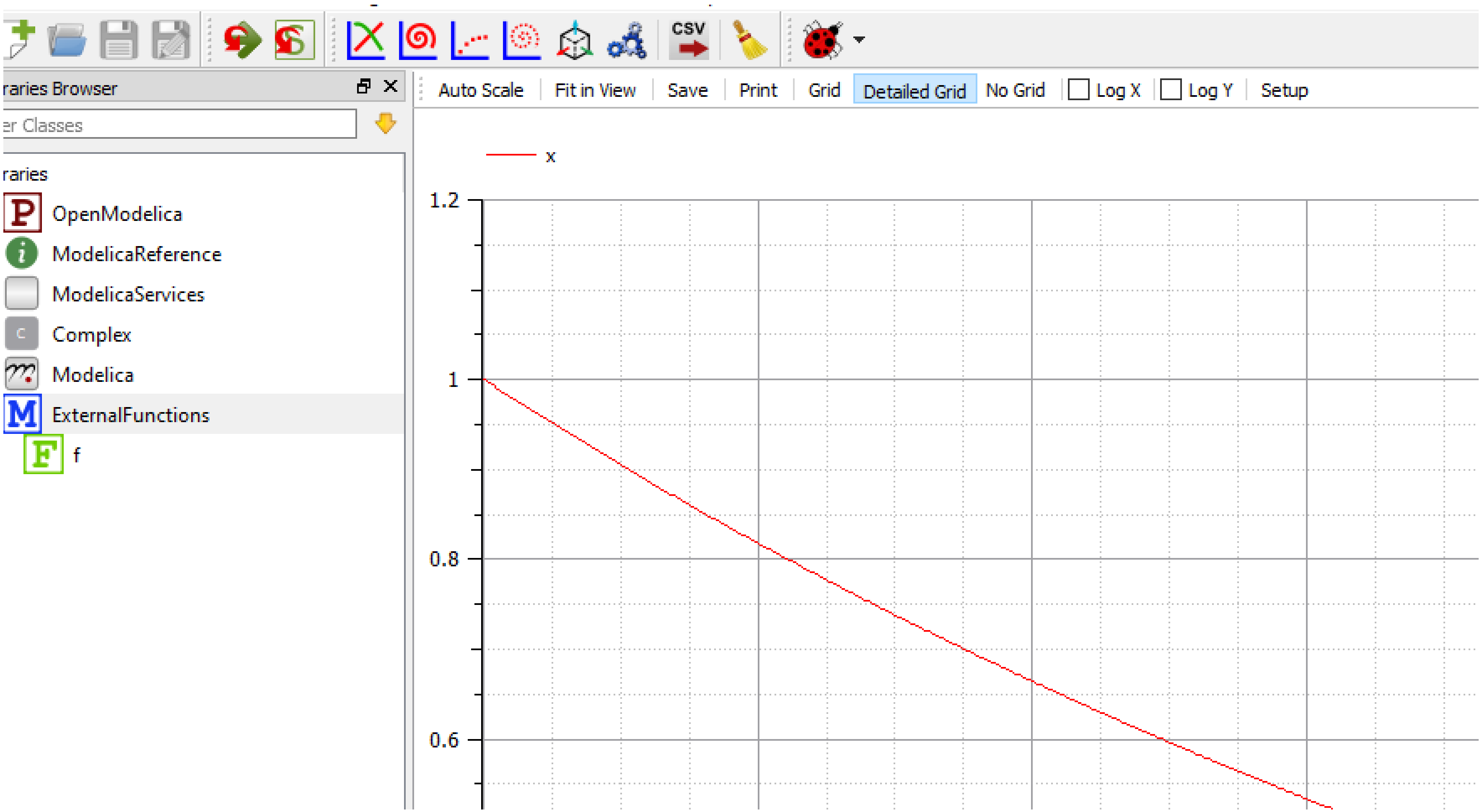
- Simulate the model that uses function f
- Optional change the external definition of function f to pass output as argument to the external C function

Solution



```
model ExternalFunctions

  function f
    input Real x;
    input Real y;
 output Real u;
 external "C" u = myC(x, y);
 annotation(Include="
 #include <stdio.h>

 double myC(double x, double y)
 {
    return x+y;
 }
 ");
  end f;

  Real x(start = 1, fixed=true);
equation
  der(x) = -f(x, time) * x;
end ExternalFunctions;
```

External function plot



External Objects

- Used for C functions that need to maintain an internal state (opaque to Modelica)
- They are good for making sure things are initialized once (functions can be called multiple time during initialization and simulation). The external object constructor is called only once before the first use of the object

External Objects – Declaration

- Extend from ExternalObject and provide two functions: constructor and destructor

```
class MyTable
  extends ExternalObject;
  function constructor
    input String fileName := "";
    input String tableName := "";
    output MyTable table;
    external "C" table = initMyTable(fileName, tableName);
  end constructor;
  function destructor "Release storage of table"
    input MyTable table;
    external "C" closeMyTable(table);
  end destructor;
end MyTable;
```

External Objects – Exercises

- Open ExternalObjects.mo
- Fill in the external C code for `constructor` and `destructor` for the external object
- Simulate ExternalObjects.Test

```
int i=0; // Hint: you can Insert this code in Constructor

  double *extObj = (double*)malloc(size*sizeof(double));
  if(extObj == NULL)
    printf(\"Not enough memory\");

  for(i=0; i<size; i++)
    if(i < 2)
      extObj[i] = 1.0;
    else
      extObj[i] = extObj[i-1]+extObj[i-2];

  return (void*)extObj;
```

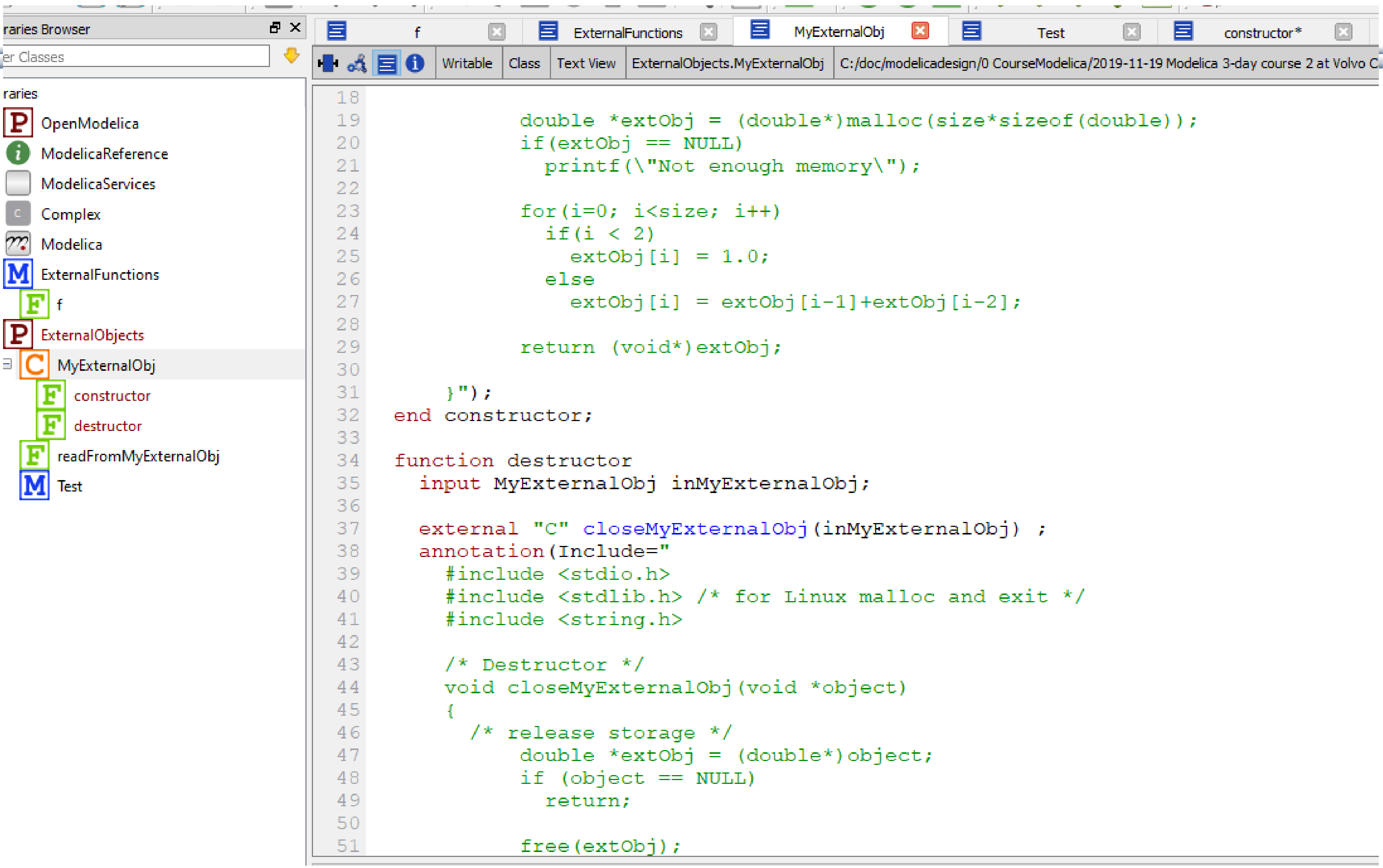
Hint, code
for constructor

Index 0 in C array
corresponds to index 1
in Modelica.

```
/* Release storage */
       double *extObj = (double*)object;
       if (object == NULL)
         return;

       free(extObj);
```

Hint, code
for destructor

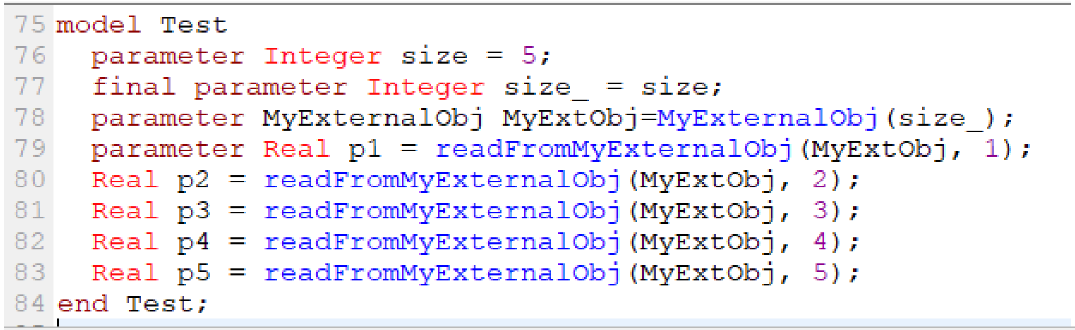
External objects solution



```

          double *extObj = (double*)malloc(size*sizeof(double));
          if(extObj == NULL)
            printf(\"Not enough memory\");

          for(i=0; i<size; i++)
            if(i < 2)
              extObj[i] = 1.0;
            else
              extObj[i] = extObj[i-1]+extObj[i-2];

          return (void*)extObj;

    }");
end constructor;

function destructor
  input MyExternalObj inMyExternalObj;

  external "C" closeMyExternalObj(inMyExternalObj) ;
  annotation(Include="
    #include <stdio.h>
    #include <stdlib.h> /* for Linux malloc and exit */
    #include <string.h>

    /* Destructor */
    void closeMyExternalObj(void *object)
    {
      /* release storage */
          double *extObj = (double*)object;
          if (object == NULL)
            return;

          free(extObj);
```

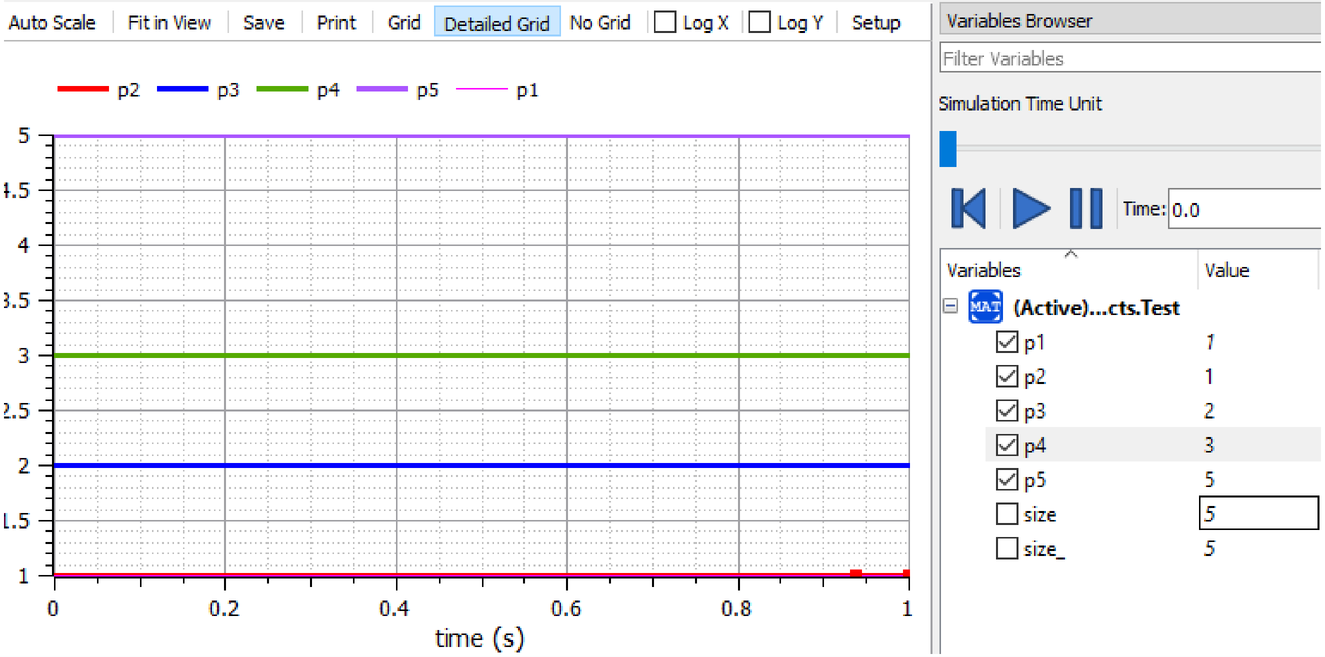
Test simulation of External Objects Example

- Test model simulation and plots. Change plot line width via plot Setup.



```
model Test
  parameter Integer size = 5;
  final parameter Integer size_ = size;
  parameter MyExternalObj MyExtObj=MyExternalObj(size_);
  parameter Real p1 = readFromMyExternalObj(MyExtObj, 1);
  Real p2 = readFromMyExternalObj(MyExtObj, 2);
  Real p3 = readFromMyExternalObj(MyExtObj, 3);
  Real p4 = readFromMyExternalObj(MyExtObj, 4);
  Real p5 = readFromMyExternalObj(MyExtObj, 5);
end Test;
```

According to the algorithm
the first two elements are 1
The next are the sum of the two
previous elements



Time: 0.0

Variables	Value
☑ p1	1
☑ p2	1
☑ p3	2
☑ p4	3
☑ p5	5
☐ size	5
☐ size_	5