Tips for Code Re-Use

Rejwan Ali, Marketing Engineer
NI Africa and Oceania
Graphical System Design

A platform-based approach for measurement and control
Calling Compiled Code From LabVIEW
C DLL vs. .NET Assembly
The Secret to Calling External Code from LabVIEW

You must have some knowledge about the external code you are calling!

✓ What language was used to develop the code?

✓ Is there a run-time present on the system for which you are calling the compiled code?

✓ What functionality is available in the external code?

✓ What data types are being handled in the external code?
Ways to Call C DLLs from LabVIEW

1. Call Library Function Node

2. Import Shared Library Wizard
Call Library Function Node (CLFN)

What is it?
- Function for directly calling a DLL or shared library function

How do you use it?
1. Point to the library file on disk (.dll)
2. Specify the name of the function
3. Specify the arguments types the function requires
4. Specify the calling convention
Import Shared Library Wizard

What is it?
- Utility to wrap DLL calls into LabVIEW VIs

How do you use it? (Tools » Import » Shared Library)
1. Point to the library file and header file on disk (.dll, .h)
2. Select functions to convert
3. Configure the VIs and Controls
Import Shared Library Wizard

What’s so great? It’s automatic!
• Parse the header file
• List the functions in the shared library
• Convert shared library data types into LabVIEW data types
• Generates a wrapper VI for each function
• Saves the VIs in a LabVIEW project library
WARNING! Caveats of Calling DLLs in LabVIEW

• Requires knowledge of C prototypes and DLL contents
  • Familiarity yields better optimization of LabVIEW code
  • Difficult to debug application if DLL throws an error

• Must have a header file or documentation

• Some complex parameter types not supported
  • Examples: array of structures, structs as return values

• Shared Libraries execute synchronously

• Shared Libraries are platform specific
Ways to Call .NET Assemblies from LabVIEW

1. .NET Connectivity Nodes (Programmatic)

2. .NET Container (UI)
A Little Bit About .NET...

What is the .NET Framework?
- Common Language Runtime (CLR) - Execution Engine
- Base Class Library - Standard functionality
- Different versions - 3.0, 3.5, 4.0

What is a .NET Assembly?
- Unit of deployment built using the .NET compiler
- Includes a manifest that contains information about the assembly
  - Name, version, publisher security
  - Files that make up the assembly, dependent assemblies
  - Resources, exported data types
Characteristics of .NET Code

• Object Oriented

• Inheritance (from any number of other objects)

• Data Protection – (various levels of access to data)
  • **Public** – accessed anywhere (inside or outside the class)
  • **Private** – cannot be accessed outside the class
  • **Protected** - can be used within the class as well as classes that inherits from this class
.NET Connectivity Nodes

- **Constructor Node** – Creates an instance of a .NET object
**.NET Connectivity Nodes**

- **Constructor Node** – Creates an instance of a .NET object

- **Property Node** – Gets (reads) and/or sets (writes) properties of a reference
.NET Connectivity Nodes

- **Constructor Node** – Creates an instance of a .NET object

- **Property Node** – Gets (reads) and/or sets (writes) properties of a reference

- **Invoke Node** – Invokes a method or action on a reference
WARNING! Caveats of Calling .NET Assemblies in LabVIEW

• Must be compatible with the Common Language Runtime (CLR)

• Must have compatible Class Libraries

• LabVIEW does not automatically detect changes to Assemblies already loaded into memory
Using the .NET Container

What is it?
• LabVIEW control for hosting .NET UI components

How do you use it?
• Load UI components in .NET Assembly
• Use .NET connectivity nodes to control

What are the Benefits?
• Add functionality that does not exist natively in LabVIEW
• Develop UIs that more closely resemble Windows OS
• Create UIs dynamically at run-time
Using Textual Nodes in LabVIEW
Add-on Modules With LabVIEW Real-Time Capability

MathScript RT Module
Re-use .m scripts that you developed in MathWorks Inc. MATLAB® or GNU Octave

1. A = [0 1; -2 -4];
2. eigs = eig(A);
3. detA = det(A);

- Now run your algorithms and models on NI Linux Real-Time targets
- Native LabVIEW solution
- No code generation or 3rd party software required

Control Design & Simulation Module
Move beyond PID for advanced control and dynamic system simulation

MATLAB® is a registered trademark of The MathWorks, Inc.
Octave vs MathScript

Octave:

```octave
octave:1> A=[0,2,0,1;2,2,3,2;4,-3,0,1;6,1,-6,-5]
A =
    0 2 0 1
    2 2 3 2
    4 -3 0 1
    6 1 -6 -5
```

```octave
octave:14> det(A)
ans = -234
```

```octave
octave:11> cond(A)
ans = 9.7626
```

MathScript:

```mathscript
>>A=[0,2,0,1;2,2,3,2;4,-3,0,1;6,1,-6,-5]
A =
    0 2 0 1
    2 2 3 2
    4 -3 0 1
    6 1 -6 -5
```

```mathscript
>>det(A)
ans =
    -234
```

```mathscript
>>cond(A)
ans =
    9.7626
```
LabVIEW MathScript RT Module

- Text-based controls, signal processing, analysis, and math
  - 900 built-in functions / user-defined functions
  - Reuse many of your .m file scripts created with The MathWorks, Inc. MATLAB® software and others
- Based on original math from NI MATRIXx software
- A native LabVIEW solution
  - Interactive and programmatic interfaces
  - Does not require 3rd-party software
  - Enables hybrid programming

MATLAB® is a registered trademark of The MathWorks, Inc.
LabVIEW MathScript RT Background

Textual node & interactive window

Added plug-ins:
- LabVIEW Control Design & Simulation Module
- LabVIEW Digital Filter Design Toolkit

Compatible with The MathWorks, Inc.:
- MATLAB®
- Signal Processing Toolbox™
- Control System Toolbox™
- DSP System Toolbox™

Useful for desktop or real-time applications with hardware

MATLAB® and Simulink® are registered trademarks of The MathWorks, Inc.
Don’t Get Confused About the Nodes

• Native LabVIEW code
• Works with:
  • LabVIEW on Windows, Mac or Linux
  • LabVIEW Real-Time

• Requires MATLAB
• Only works with LabVIEW on Windows
• Does not work with LabVIEW Real-Time
Demo: MathScript Interactive Environment
LabVIEW MathScript DEMO

Graphical and textual programming
Interactive user-interface
Heat Equation DEMO

C:\Program Files (x86)\National Instruments\LabVIEW 2013\examples\MathScript\MathScript - Heat Equation
Importing your .m file and adding interactivity DEMO

Fs = 150; % Sampling frequency
t = 0:1/Fs:1; % Time vector of 1 second
f = 5; % Create a sine wave of f Hz.
x = sin(2*pi*t*f);
nfft = 1024; % Length of FFT
% Take fft, padding with zeros so that length(X) is equal to nfft
X = fft(x,nfft);
% FFT is symmetric, throw away second half
X = X(1:nfft/2);
% Take the magnitude of fft of x
mx = abs(X);
% Frequency vector
f = (0:nfft/2-1)*Fs/nfft;
% Generate the plot, title and labels.
figure(1);
plot(t,x);
title('Sine Wave Signal');
xlabel('Time (s)');
ylabel('Amplitude');
figure(2);
plot(f,mx);
title('Power Spectrum of a Sine Wave');
xlabel('Frequency (Hz)');
ylabel('Power');

http://www.utdallas.edu/~dlm/3350%20comm%20sys/FFTandMatLab-wanjun%20huang.pdf

ni.com
Recommended options for analyzing data with MathScript

DAQ Assistant plus MathScript node
Instrument Drivers plus MathScript node
I/O Asst plus MathScript node
Working with LabVIEW MathScript

- Develop scripts interactively with the MathScript Window
- Move to the MathScript Node to “Instrument your Algorithms”
- Move back and forth as necessary to complete your work
Data Types in MathScript

Real and complex numerics & matrices
Booleans and strings
Structs (can connect to LabVIEW clusters)
Cell arrays
Plot objects
Debugging A MathScript Node

- Execution highlighting and single-stepping
- Probe tool for:
  - Variables in each node
  - Output defined within node
- Syntax error indication for each line
- Error indicator for node
Additional MathScript Capabilities

User-defined functions
Global variables shared between interactive window and node
Special MathScript probe for debugging node
Warning glyphs and guidance to improve run-time performance
Nodes can run in parallel and on multi-core computers
Dynamic System Simulation DEMO

- Quadcopter shipping example available w/ evaluation software
- Full state feedback control implementation and simulation w/ LabVIEW Control Design and Simulation Module
- Equations of motion in LabVIEW MathScript node – 6 DOF
Real-Time Algorithm Examples

**Open** algorithms for microprocessor targets:
- Optimization w/ constraints
- Curve fitting
- Model predictive control
- System identification

Solutions for MIMO & nonlinear systems

**Online** design & adaptation
Dr. Andy Clegg, Lead Engineer & ISC Managing Director:
“We used the LabVIEW MathScript RT Module to run a textual node containing our m-file code, developed with MATLAB on a desktop, for kinematics on the real-time CompactRIO controller. MathScript has significant benefits for real-time deployment on the CompactRIO controller including determinism, easy debugging, and no extra compilation steps.”
MathScript Industrial Case Study Quotes

**BPP-TECH (UK) Riser Management:**
LabVIEW MathScript and formula nodes enabled our pre-existing IP developed in text-based languages to be included within the software, allowing for more effective code reuse.

**ALSTOM (UK) Condition Monitoring:**
“The original Novus tool developed using MATLAB software has been successfully implemented within the LabVIEW platform. We chose LabVIEW over other programming languages because it helped us decrease development time, develop familiar Windows-style GUIs, use LabVIEW MathScript functions to embed our .m file scripts in the solution, and adapt system functionality for future releases.”
Deploy to Hardware Through LabVIEW

MathScript RT Module

Control Design & Simulation Module

The MathWorks Inc. software development environment

Your .m code

Your .mdl code

MATLAB®

Simulink®

Simulink Coder™

LabVIEW Real-Time

NI VeriStand

CompactRIO, Single-Board RIO, PXI, or desktop

MATLAB® and Simulink® are registered trademarks of The MathWorks, Inc.
Simulation Model Converter DEMO

http://www.library.cmu.edu/ctms/ctms/simulink/basic/imatlab.htm
LabVIEW Control Design and Simulation Module

- Both signal flow and .m file development
- Single environment for:
  - Simulation of dynamic systems
  - Real-time implementation for rapid control prototyping or hardware-in-the-loop simulation
F14 DEMO
Standard Notation Reduces Learning Curve

LabVIEW Control Design & Simulation Module

The Simulink® software environment
With both an FPGA and a PowerPC processor, the Single-Board RIO platform has the flexibility to handle a variety of programming tasks and algorithms in an efficient manner. Onboard Ethernet, combined with a Wi-Fi adapter, enables real-time wireless communication, debugging, and deployment of software.

– Nick Morozovsky, Mechanical Engineering PhD student, Switchblade project leader
Video “UCSD Switchblade Robot”

http://www.youtube.com/watch?v=Dw0WxPlyWII
LabVIEW MathScript RT and Control Design and Simulation Review

- Quickly take code to hw targets
- Multicore ease-of-use
- Instrumenting your algorithm
- Graphical/textual combination
- LabVIEW includes full programming capability
- Integration of 3rd party IP – including .m and .mdl
- LabVIEW is a complete graphical system design environment w/ all of the MoC’s that you need
Summary

- **LabVIEW is an open software platform**
  - No reason to rewrite code
  - Save development time and money by integrating external code

- **Basic understanding of external code required**
  - Language and OS used to compile, run-time present
  - Functionality and data types implemented

- **Multiple ways to integrate external code**
  - Call Library Function Node (C DLL)
  - Import Shared Library Wizard (C DLL)
  - .NET Connectivity Nodes and Container (.NET Assembly)
  - .MathScript interactive window and nodes
  - Simulation Model Converter
Stay Connected During and After NIWeek

ni.com/niweekcommunity
facebook.com/NationalInstruments
twitter.com/niglobal or #NIWeek
youtube.com/nationalinstruments

Provide your feedback on this session via the NIWeek App