Analysis Plugin Training

The analysis plugin C++ API allows users to implement failure analyses (strength, crippling, buckling, etc.) in HyperSizer. Using these custom analyses, stress engineers can leverage the powerful automation capabilities of HyperSizer - FEM import, load case management, detailed sizing, stress reports, 3D graphics, FEM update - while getting the correct answers.

Objectives

- Build, compile, and execute an analysis plugin
- Navigate the analysis plugin API to extract loads, geometry, stresses, strains, and material allowables
- Store intermediate analysis details for stress report traceability
- Create user-defined constants for custom analysis inputs
- Design your analysis plugin code so that the core engineering logic can be called from non-HyperSizer applications (e.g. Excel or MATLAB)
- Unit test your analysis code to ensure the reliability of results

Who Should Attend

Stress engineers with an interest in tool development and automation.

Prerequisites

Programming Background
Students should have experience implementing engineering methods in programming languages such as Excel VBA, MATLAB, Python, C++, Fortran, etc.

C++ Background
Students should have a beginner level understanding of C++. Advanced proficiency in C++ and object oriented programming is not required.

Before coming to the course, students should be able to:

- Create a "Hello World" console application using Visual Studio or Eclipse.
- Understand how to create and consume header files.
- Create a simple C++ class using a constructor.
- Implement basic control flow constructs such as if-then, for, for-each, etc.

Target Application
Students should come prepared with some failure analyses that they would like to implement as a plugin.

Course Overview

Day 1

- Build the plugin using Visual Studio or Eclipse.
- Debug the plugin using the IDE.
- Navigate the plugin API and documentation.
- Extend the sample plugin.
    - Add analysis details for stress reporting.
    - Add custom correction factors via user-defined constants.

Day 2

- Design your analysis plugin for reuse - patterns and practices.
- Develop an automated test suite for your plugin.
- Incorporate 3rd party libraries into your plugin. Example using Eigen for matrix mathematics.

Day 3

- Workshop to implement your analysis plugin.

Contact Us

For more information including upcoming training classes, pricing, or trainer availability, please contact us.