MATLAB Implementation of First-Order Sizing Tool for Supercritical CO_2 Compressors

Prepared by: Iggy Matheson
2016 University Turbine System Research Gas Turbine Industrial Fellowship Program
BS Mechanical Engineering 2016, Brigham Young University, Provo, Utah

Prepared for: Timothy C. Allison
Manager, Section 17 (Rotating Machinery Dynamics)
Division 18 (Mechanical Engineering)
Southwest Research Institute, San Antonio, Texas

Executive Summary

As the host of the Supercritical Carbon Dioxide (sCO_2) Power Cycles Symposium, the possessor of the world's largest prototype sCO_2 turbine test loop, and the recent recipient of $13.1 million in sCO_2 research grants, Southwest Research Institute (SwRI) is an internationally-recognized leader in sCO_2 research. Despite its prominence, SwRI lacks a systematic method to conduct parameter sweeps when optimizing first-order sCO_2 turbomachinery designs for clients, instead relying on a cumbersome spreadsheet program to compute single cases. The goal of this fellowship was to produce a command-line program capable of computing sCO_2 compressor dimensions and efficiencies for specified ranges of operating conditions and exporting the calculated data in a format useful for design decisions and later detail design work. Due to the author's inexperience with object-oriented programming and computational challenges with compressor operating conditions near the two-phase region, this program was not completed by the fellowship's end date. Instead, a command-line function to optimize compressor sizing at a single operating point within the MATLAB environment allows end users to write their own wrapper scripts as desired for sweeps.

Introduction to Supercritical Carbon Dioxide

Supercritical carbon dioxide is an attractive process fluid for turbomachinery power cycles due to its high density, high specific heat capacity and thermal conductivity, relatively benign chemical properties, and resource abundance. These allow compressors and turbines to change the specific enthalpy of the process fluid by large amounts without large changes in bulk properties and therefore with little entropy increase. The theoretical efficiencies of most Brayton cycles with sCO_2 are several percentage points above those of high-performing Ranking cycles with steam, and the high density of the working fluid allows sCO_2 turbomachinery to achieve comparable performance with components over ten times smaller by length, and over a thousand times smaller by volume. Commercialization of sCO_2 technology is hindered by high dry gas seal leakage rates, windage losses from viscous interactions between rotating components and their housings, rotordynamic instabilities, poor understanding of contaminant effects, and adequate heat exchangers for high-pressure, high-density, high-temperature flow. Also, although each stage of a sCO_2 compressor may be separately optimized for its flow conditions, the differing hub and tip diameters in each stage may make the complete machine dynamically unstable and difficult to build.

Pre-existing Spreadsheet-Based First-Order Sizing Tool

The existing tool used by SwRI engineers for first-order compressor sizing on client projects is the Excel spreadsheet seen below. Operating conditions are entered in the blocks on the left, and stage parameters are calculated in the columns of the main block and summarized in the bottom block. Users must manually add and delete columns as needed for the desired stage count, and the resulting data and plots cannot be easily exported to other programs and spreadsheets. The spreadsheet template is built for a single machine at a single operating point, making it difficult to chart performance trends as operating conditions or the stage count change. The order of calculations is difficult to follow, and the dependency is fixed. This sometimes leads to first-stage calculation failure, requiring end users to manually enter reasonable values for stage dimensions (usually the hub diameter) instead of allowing the tool to compute them. Users must manually adjust one parameter, the "enthalpy change multiplier" to produce the same tip diameter for each stage. It was felt that a command-line program using the same calculation sequence would increase the flexibility and accuracy of first-order sizing efforts and allow optimization earlier in the design process.

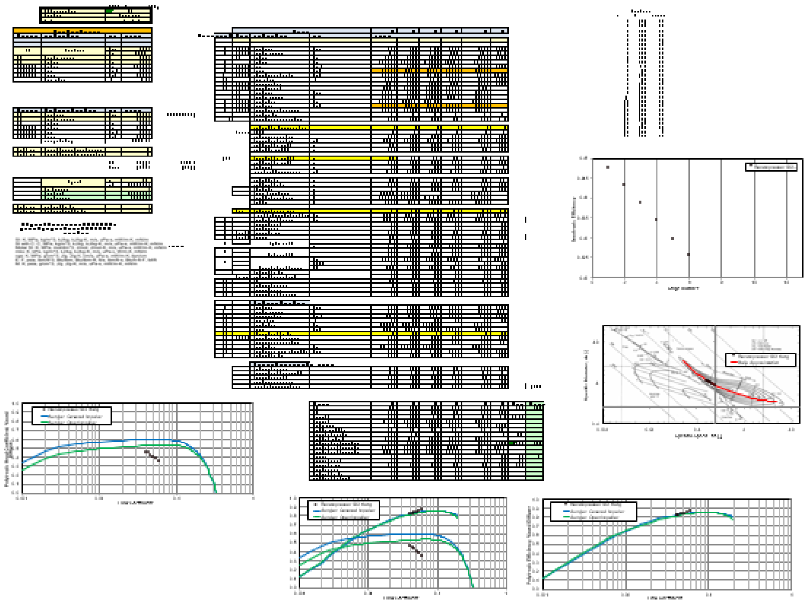
Figure 1. Existing first-order sizing tool.

Program Structure

The first-order sizing tool was written iteratively, as the author's understanding of the problem and MATLAB code improved. The first iteration was a single script to replicate the results of the Excel spreadsheet for a single case. Next came a package of separate scripts operating on the same workspace to compute stage properties, for a series of machines with the same operating conditions but different stage counts. To run a compressor analysis, a **Main** script was called, which called separate scripts to read in machine definitions, compute boundary conditions, initialize variables, compute stage properties and optimize them for uniform hub and tip diameter, compute overall machine performance, and generate various plots. Hub and tip diameter were controlled using simple proportional feedback loops that applied relative diameter errors to each stage's two degrees of freedom: the enthalpy change multiplier and the suction Mach number. All properties were saved to cell variables in the workspace with one cell entry for each machine definition, each cell entry consisting of a vector with the values of the applicable variable for each stage of the machine. All plots were generated in separate figures and remained on the desktop.

After the initial script package was working for the thirty-six test cases (four stage counts at each of six operating conditions, and two stage counts at each of six more operating conditions), it was felt that the code was complex enough to merit a graphical user interface (GUI) for easy management of inputs and outputs. It was also felt that the input and output file structure of the GUI would strongly influence the class and method structure of the desired object-oriented, parameter-sweeping final app, so the GUI was written first with the intention of integrating the compressor algorithms later. Due to the author's lack of experience with event handlers and callback functions, the GUI was not complete by the end of the fellowship, and work on the object-oriented conversion and parameter sweeps had not yet begun. A sample window from the GUI is displayed at right. Users can save analysis configurations consisting of unique combinations of operating case, machine definition, parameter sweep, and plot output. These definitions are stored as data files in a subfolder and can be added, edited, and deleted in other GUI windows (not pictured).

Once it was clear that not all desired structure and functionality of the application would be ready by the end of the fellowship, the **Main** script was modified to run as a function that operates on a single machine definition and operating condition and saves all stage information to a single data structure variable and all plots to a user-specified folder. Because the single-machine **Single** function leaves the desktop and workspace clean, it can be called within a user-defined wrapper script to run parameter sweeps for optimization purposes even though it depends on the same set of interacting scripts as the multi-machine-capable **Main** script.

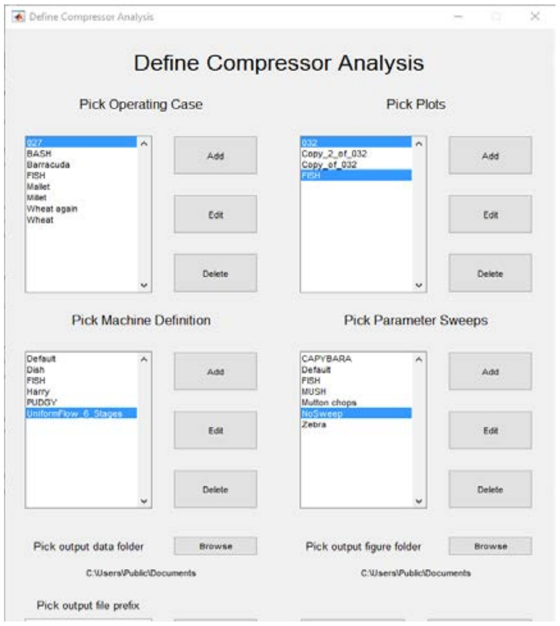

Figure 2. Sample window from incomplete GUI.

Sequence of Calculations

The calculations implemented in the MATLAB code closely followed those in the original spreadsheet, but with some differences. Because sCO_2 cycles operate very close to the two-phase region, the industry-standard REFPROP program was used to calculate thermodynamic properties of the process fluid. This program is implemented slightly differently in Excel and MATLAB, with different results near the critical lines. Because any two thermodynamic properties can be used to obtain any other thermodynamic property of a simple compressible system, the order of property calculations could usually be changed to prevent divergence. The optimization loop frequently entered the two-phase region during code testing, which also required changes in the calculation sequence.

Further Work

Further work should focus on developing a robust calculation algorithm that avoids the two-phase region and other divergence errors, perhaps by a path-finding routine that explores all possible calculation sequences before returning an error. Several dozen possible calculation sequences are estimated to exist, too many to rigorously test during the span of the fellowship. Although a calculation sequence was found that worked for all thirty-six test cases, it produced a two-phase error when the code was first used on a client project, during the last week of the fellowship. This left too little time to find a new calculation sequence that worked for the thirty-seventh case. Due to the wide variety of operating conditions which may be expected to occur in production sCO_2 turbomachinery and the different ways the optimization loop may respond to these, it is probable that no single calculation sequence exists which will return results for all physically reasonable combinations of operating condition and stage count. Since a software tool that requires extensive rewriting and testing for each new case is of little or no practical use, an adaptive algorithm is crucial. A routine to save and plot stage data from each iteration of the optimization routine, overlaid on a phase diagram, would help diagnose convergence failures.

Besides an adaptive algorithm, the code lacks the originally requested parameter sweep capability. End users may easily write MATLAB scripts to run the single-condition code for any desired range of parameters and process the saved data from each run as needed, but the first-order sizing tool would be more complete and user-friendly if this capability was native to the app package.

Although the sizing tool is meant for use from the command line, it is desirable to finish the graphical user interface to encourage use by engineers who are less familiar with the command line.

Finally, the code should be rewritten from an object-oriented perspective for greater flexibility, translated to Python, and compiled as an executable for command-line compatibility outside the MATLAB environment, since MATLAB licenses are expensive.

Other Projects: Pump Vibration Analysis

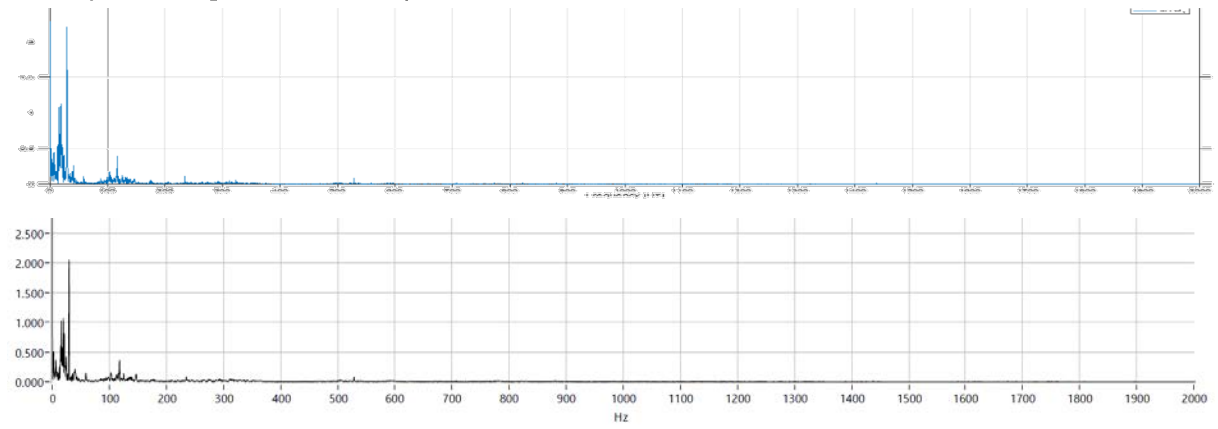

Figure 3. MATLAB FFT (above) and client FFT (below) for validation data set.

A secondary project involved helping two engineers in Section 17 conduct a root-cause failure analysis for a pumping company in Nevada. Fast Fourier Transforms (FFTs) were applied to raw accelerometer data from pump vibration tests with and without failure, in order to identify the dominant vibrational frequencies. A MATLAB routine was written to automate FFT analysis of large folders of vibration tests. The FFT routine itself was validated using the single pumping test for which the client provided both the raw data and its own FFT plot. However, enough uncertainty remained in the FFT results and in the interpretation of FFT plots provided by the client that further work was postponed until the client returned some requests for information, which did not occur before the end of the fellowship.

Other Projects: Miscellaneous

As support for ongoing Division 17 projects, other fellowship work included the preparation of engineering drawings for a set of coupling guards to be installed on a new CO_2 test loop, minor changes to piping and instrumentation diagrams in support of the new loop, identification of a polynomial fit function to describe corrosion growth with time and temperature on various alloys, conversion of archived Mathcad worksheets from prior projects to MATLAB, and labor to reconfigure the valves and instrumentation of a reciprocating compressor test cell for a new series of tests.

Acknowledgements

The UTSR fellowship program at SwRI was an excellent opportunity to learn about turbomachinery operating and design principles in the supercritical regime, and the thermodynamics of supercritical fluids, as well as a good place to practice programming and computer-aided engineering in MATLAB, ANSYS, and SolidWorks. This summer’s lessons in a professional, rather than academic, research environment will be useful in later career development. Living in a new city, state, and section of the country also provided valuable cultural and geographic perspective. Special thanks are due to Timothy Allison, Natalie Smith, Jonathan Wade, Hector Delgado, Justin Hollingsworth, Meera Day, Grant Musgrove, Ryan Cater, Aaron McClung, Brian Moreland, Herman Monroe, Jeremy Johnson, Larry Miller, Aaron Rimpel, and Stefan Cich for giving projects to work on and answering questions about thermodynamics, mathematics, coding, software in general, and San Antonio in general.