2016

Software Development

Champion, PA

LinExtractor

Table of Contents

Research

Problems in advanced physics and mathematics can often be simplified to linear equations. Examples include single and multi-spring systems, pendula, wave motion, quantum potential systems, or even basic multi-variable algebra. Even complex chaotic systems can undergo linearization by using Jacobian matrices at critical values, being reduced to an eigenvalue equation.

Though definitions of determinants, inverses, conjugates, transposes, and classes of matrices can be found in a list in the back of any linear algebra textbook (or google search), their computation by hand may be extremely intensive. For example, a determinant on an n x n matrix takes multiplies n values by n determinants of n-1 x n-1 matrices, meaning the number of operations required grows by n^3. This means while a 2x2 can be found in 3 seconds by an experienced mathematician, a 13x13 may take a number of days.

The current computation options are to either buy (or have one's school buy) an expensive mathematical package which may be bulky to load or require waiting for computation time or to write your own program with an obfuscated python or lua library (which may be required for more advanced functions like neural network processing).

Description of Project

Problem: While matrices are easy to understand, they are difficult to use without bulky arithmetic or software.

Solution: Develop a program to more easily reach linear algebraic solutions, utilizing all necessary calculations.

A user of the program will likely have some interest in the fields they will use the program for (physics, engineering, or purely mathematics). They will surely be able to check their work for advanced problem sets and benefit in cases such as finding 0 and stating no solution, where they would otherwise be incorrect due to the quantity of arithmetic involved. Others may develop a better grasp of the numerical importance of these values beyond a grueling process. New mathematical insights may be revealed which have not even been given by the program (classes of matrices obeying certain properties).

The program simply reads inputs when told do so and draws calculation results on the screen. They allow for finding basic information which may be a straining piece of a complex multi-step process such as:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & i & 3 \\ 0 & -1 & 0 & -i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3i & 0 & 0 \\ 0 & i & 0 & 0 & 0 & 0 & -5i-7 \\ 0 & 0 & 3i & 0 & 0 & 0 & 0 \\ -i & 0 & 0 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 5i-7 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(Is the resulting 7x7 matrix a Hermitian self-adjoint operator?)[1]

[1] Yes

TECHNOLOGY STUDENT ASSOCIATION
PLAN OF WORK

Date	Task	Time involved	Team member responsible	Comments
12/22/2015-12/29/2015	Decided on an idea and began documentation and development	3h-ideas 4h-program	2-ideas 1-program	Other ideas much more bizarre or impractical to solving a problem
1/11/2016	Updated documentation to new simple data calculation	3h	1,2	Plan for high-level software design and layout; Class structure
1/12/2016	GUI tweaked from template	4h	1	Empirical alignment and abstract classes
1/12/2016	Updated documentation and presentation	3h	2	Added simple graphics and matrix formatting
1/13/2016	Full matrix functionality implemented	4h	1,2	Combined GUI and Matrix classes via conversion and a display event
2/18/2016	Planned improvements with new group	2h	1,2,3,4,5	Rework functionality, describe more precisely
2/23/2016-3/3/2016	Implemented full linear algebra system	8h	1,2	Fixed order and matrix size errors; Implemented notation
3/24/2016-4/2/2016	Updated documents Added features	4h	2,4,5-document 1,3-program	Better diagrams, animations Reference values, aligned boxes
4/5/2016	Final documents and burn	2h	2,4,5	Documentation, presentation, and program on disk
Advisor signature				

Project Requirements

For a user to perform calculations accessibly, a Graphical User Interface must be developed and interact with the calculation and abstracted portions of the program. An API must be combined to handle the overhead load of displaying this. All calculations must be reduced to functions with a set of inputs, but with an individual printable output. This should consider what the user will know, how they would most easily represent it, and how they could most easily read or reuse it (convert to an answer or put into another program by no more than a simple replace macro).

The GUI must be simple and consistent, and the program's calculations and algorithms must be efficient as to be a fast alternative to other methods of calculation (math packages, hand methods, agreeing with one's textbook). A majority of the problem must be visible to allow the user to modify their input.

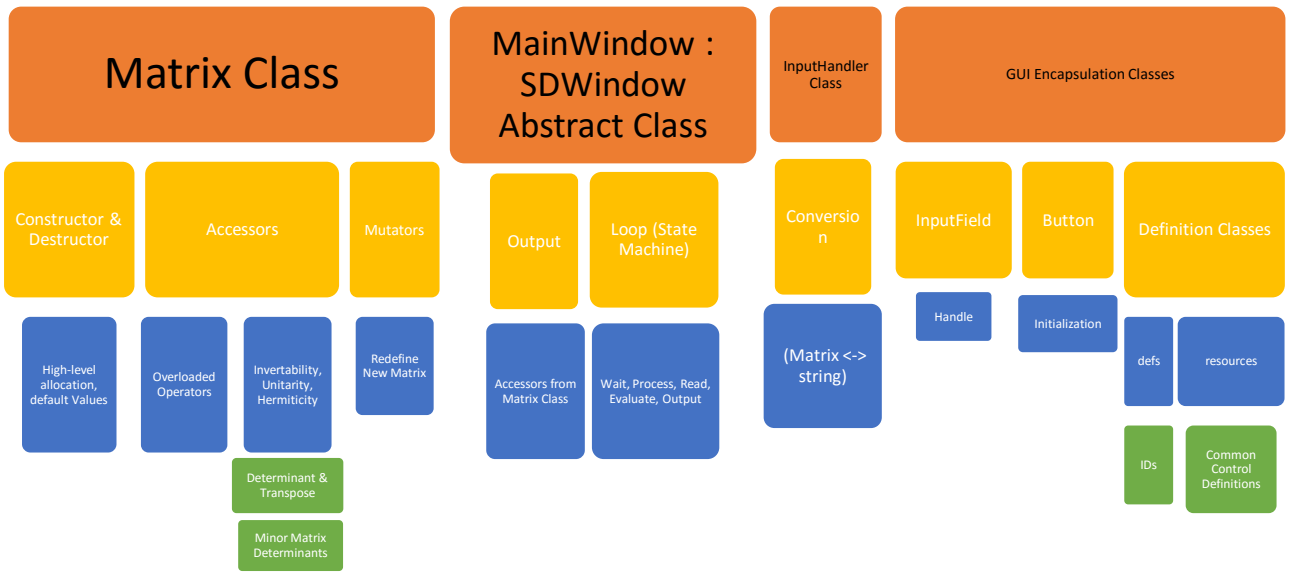
To interface, algorithms to convert between string and Matrix objects must be implemented along with error correction. This should have well-defined delimiters and allow for complex inputs such as **4.2526255e23+3.355e-45i**.

High-Level Software Design

The matrices are instantiated from a Matrix class using stdc++ vectors in their implementation. The values are private with accessors and operators overridden (a substantial benefit of C++) for common functions which may be necessary for computation.

Beyond a portable linear algebra library (which are often rewritten for individual programs), higher-level stringstreams and conversion functions are written to interface between a high-level object implementation and low-level API access (stringstream -> string -> char[] ~ char* ~ long).

The built-in Microsoft Windows API is wrapped by the windows.h header for a practical interface (could be easily copied to a multi-platform library). The GUI will be called through the functions WinMain (an overridden main) and WindowProcedure (The state event handler). Inputs are added as "EDIT" windows and are returned as handles. The GetWindowText(HWND,char*,int) and SetWindowText(HWND,char*) functions are utilized to read and write values when a button's ID is returned as a parameter in an event (WM_COMMAND). All components are given unique ids for this process.



Testing

Matrix printout formatted properly when converting from input to matrix and back.

Solution accuracy:

2+2 = $[4]$ ✓

4x4 determinants match. ✓

Determinants correspond to Boolean functions. ✓

Even $\begin{bmatrix} 12 & 0 & 9-5\times10^5 i & 0 \\ 0 & 1 & 0 & 0 \\ 9+5\times10^5 i & 0 & 1 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix}$ properly returns self-adjoint. ✓

Identities of size NxN * a matrix of size NxM gives back the NxM matrix. ✓

An NxN incremental matrix has determinant 0. ✓

Module unit testing:

- With test matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$, all minors are found correctly. ✓
- All acceptable forms of input even with minor omissions for simplicity are read accurately by the program. ✓
- The GUI navigates without errors in display, overlap, or stretching off screen. ✓
- The Matrix and GUI classes function independently. ✓
- Input fields, dialog boxes, and other controls display properly without intersection. ✓
- Format conversion functions provide recognizable input. ✓

Testing

Problems/Solutions:

- Arrays are size-immutable, but easy to address
    - Use name-inspired std::vector class with overridden addressing methods
- String methods were inconsistent in memory (delete[] necessary for buffers)
    - Used std::ostringstream class over cstdlib char* convert functions
- Outputs are too long
    - Formatted and stress-tested to allow numbers as long in display as, for example, 4.56746e112 to fit on the screen
- Crashes on inputting " [a,b;c]"
    - Check to ensure column sizes consistent and return ERR_MATRIX if incompatible matrices constructed or operated on
- $\prod_{k=0}^{N} A_k \neq A_0 A_1 \dots A_{N-1} A_N$
    - $\prod_{k=0}^{N} A_k = A_0(A_1(\dots(A_{N-1}A_N)\dots))$
- OPEN: WIN32 API still uses low-level long character pointers for strings with obfuscated typedefs. (Learn a better GUI library)
- OPEN: Leading constants read as 1x1 matrix (i.e. 3*[1,0;0,1] crashes, while 3*[1,0] returns [3,0]) (Add contextual conversion)

End-User Product Documentation:

LinExtractor

Beginning:

After starting the program, **`LinExt.exe`**, you will be met with the main menu window.

Calculating:

After entering the program, you will be met with an input box with a default value and a calculate button.

Type in the value of your input into the input text box, and then press *Calculate* to update the outputs.

Formatting:

Inputs in the form [a,b,c;d,e,f;g,h,i] -> $\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ (i.e. ',' and ';' to delimit)

One may use ***** between matrices for multiplication and/or **+** for addition. (Remember, the group of $n\ x\ by\ x$ matrices under these operations is non-abelian, therefore **order matters**)

Values in these must be between $2.2250738585072014 \times 10^{-308}$ and $1.7976931348623158 \times 10^{308}$ in one of these **3** forms:

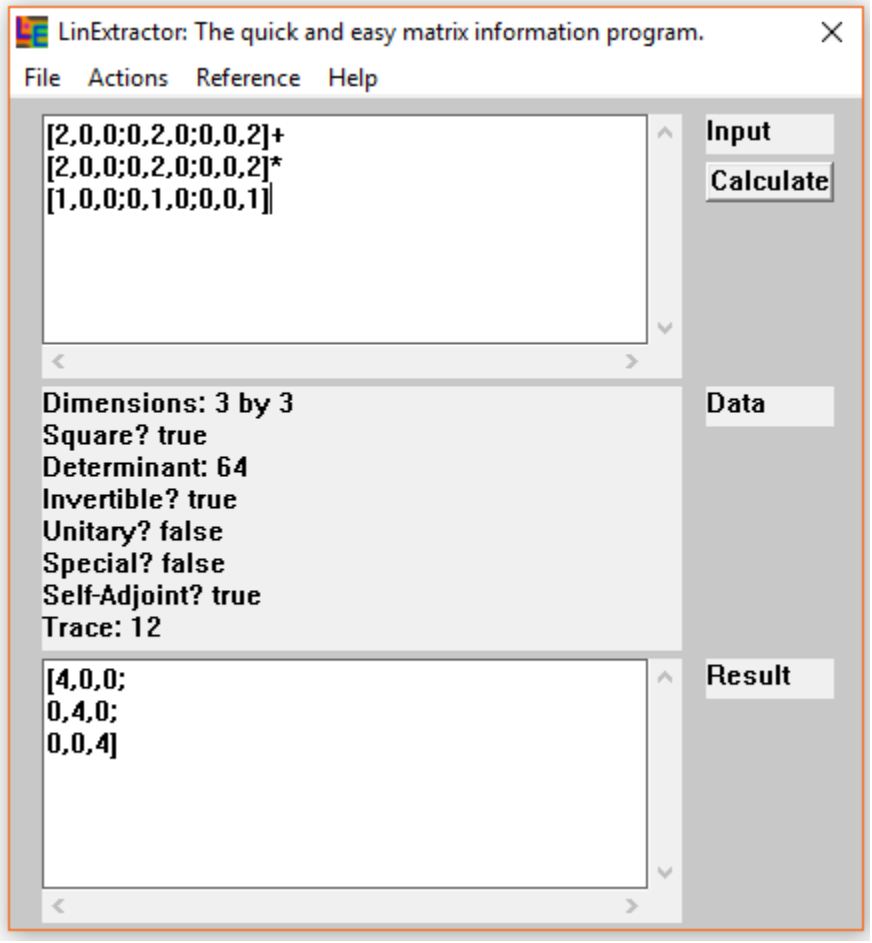
- 25463525
- 2.5463525e7
- 2.5463525e+7

Or Complex (real|imaginary) (**pure imaginary must include 0|**):

- $3|425 = 3 + 425i$
- $0|\text{-}204 = -204i$
- $1944e345|\text{-}194e\text{-}6 = 1944 \times 10^{345} - 194 \times 10^{-6}i$

Besides a 2147483647 limit to each dimension, computational power and memory are the only limits.

End-User Product Documentation- Graphical Breakdown



To use the program, simply:

1. Type a valid Matrix expression utilizing valid MxN matrices (do not mix) and both + and * operators to analyze in the input field.
2. Click on **Calculate**.
3. Observe and interpret your results (if an expression contains more than one matrix, the result used for analysis will be displayed).

Outputs:

- Dimensions – Dimensions of the matrix (rows by columns)
- Square – true if the matrix has an equal number of rows and columns
- Determinant – Useful measure of the matrix; found recursively; used to determine whether solutions exist or in eigenvalue analysis
- Invertible – true if there exists a matrix that when multiplied by this one, one achieves the identity matrix

- Unitary – Whether the matrix preserves inner products when applied to vectors
- Special – Whether the determinant is 1, leading to useful mathematical symmetries
- Hermitian Self-Adjoint – Whether the complex conjugate of the transpose is equal to the original matrix (useful as for a Hermitian matrix H, $\langle\psi|H|\psi\rangle$ is strictly real i.e. an observable value, allowing the representation of complex systems)
- Trace – the sum of the diagonals; part of simplifying characteristic equations
- Result – a representation of the final matrix used to calculate properties (result of an expression)

Menus:

- File
  - Exit – closes programs
- Actions
  - Generate Identity – opens dialog to create an NxN trace identity or fill a matrix with values
  - Generate Incremental – fills a single input matrix with an increasing series of values
  - Find Wolfram String… - converts output matrix to the format used in common Wolfram Products (Mathematica)
  - Find MATLAB String… - converts output matrix to the format used in the common MATLAB program
- Reference
  - Trig Values and Pi Ratios – prints numerical results to common "unit circle" values and whole number ratios of pi
  - Common Constants – prints common mathematical constants (π,e,√2)
- Help
  - About – view abbreviated program documentation

Evaluation

The program development went quickly and logically after refreshing on some algorithms and generalized C++ libraries. Having experience with hand-written Windows API code, it may have been the easiest part. Now, we've figured out a simple general method for implementing mutable matrices along with how to turn hand algorithms into functional steps. A few steps involved investigating for half an hour to realize a single line was giving a bizarre output. Otherwise, most methods functioned properly on the first attempt.

Rather than using our current method of going to Wolfram Alpha, searching, "determinant {{3,1,5,4},{9,7,4,2},{4,5,8,8},{1,4,5,3}}," and waiting for computation time to be exceeded, we can quickly and cleanly click, plug the matrix in, and our determinant/trace/etc. appears (sure without alternative forms or step-by-step solutions). Some problems such as guessing whether a matrix is in a Special Unitary group becomes tedious as the list of elements grows with dimension. This program could easily tell us. With a scalable design and success with relatively miniature dimensions, one should surely trust the program on even larger matrices.

This program does successfully introduce theoretical functionality such as determining whether a system of 10 to 32768 linear equations has one, none, or infinitely many solutions which is beyond the reach of the average scientific graphing calculator or free math package. It was designed to be small, avoiding overhead, while leaving behind a reusable, simplistic, and practical linear algebra library. To this goal, we find this project extremely successful.

Eventually, the program could be expanded into multiple matrix operations. For example, commutators, accessing matrix operators, returning matrices from expressions ($A^{-1} = A^{T}$), or eigenvalues/vectors could be implemented. A more user-friendly input and perhaps matrix output would be another simple addition. In the end, a graphical visualization of vectors undergoing a transformation may be a much more useful and original idea.

References

Adams, Allan. “Lecture 7: More on Energy Eigenstates.” MIT OCW: 8.04 Quantum Physics I. MIT, Spring 2013. Lecture.

“Complex number class.” *C++ Reference*. cplusplus.com, 2015 Web. 12 Jan. 2016

Mattuck, Arthur. “Lecture 25: Homogeneous Linear Systems with Constant Coefficients: Solution via Matrix Eigenvalues (Real and Distinct Case).” MIT OCW: 18.03 Differential Equations. MIT, Spring 2003. Lecture.

Strang, Gilbert. MIT OCW: 18.06 Linear Algebra. MIT, Spring 2010. Lecture.

“Windows API Index.” *Microsoft Developer Network*. Microsoft, n.d. Web. 29 Dec. 2015.