2015

Software Development

Champion, PA

PHYSICS SIMULATOR COLLECTION

Table of Contents

Research

Physics problems, though painstaking to work out by hand, have been reduced through mathematical models and theorems to simple equations.

In orbital mechanics, Newton's Universal Law of Gravitation, $\vec{F} = \frac{mMG}{r^2}\hat{r}$, leads to expressions for an entire elliptical orbit with only a point-mass abstraction. Kepler's Third Law, $T^2 \alpha a^3$, can then be used to find periods of oscillation. After a fair bit of linear algebra and vector calculus, most parameters can be calculated. For launches and transfers, however, another expression is required. A derivation from changes in momentum, mass, and impulse/exhaust velocity yields the rocket equation, $\Delta v = v_e \ln \frac{m_i}{m_f}$.

Oscillator mechanics, or the dynamics of simple harmonic oscillators, are physical systems reduced to the ODE $m\ddot{x} + c\dot{x} + kx = 0$ with ideal/no-damping solution, $x = A\cos\omega t + \varphi$. For springs, $\omega = \sqrt{\frac{k}{m}}$, but for penduī, $\omega = \sqrt{\frac{g}{L}}$. Most oscillatory parameters, such as frequency, angular frequency, period, wavelength, and wavenumber, can be found through multiplication, division, and 2π coefficients. The wavelength and wavenumber additionally require the propagation velocity v, where $y = A\cos(k(x - vt) + \varphi)$.

Two simple forces are documented caused by electric and magnetic fields on charged particles: the Coulomb and Lorentz. The Coulomb force between two particles of charge q is $\vec{F} = \frac{q_1 q_2}{4\pi\varepsilon_0 r^2}\hat{r} = q\vec{E}$ and the Lorentz Force in a magnetic field B is $\vec{F} = q\vec{v} \times \vec{B}$. The Lorentz Force requires a vector cross-product, which must be evaluated with either a matrix determinant or sin of the angle between the velocity and B-field vectors.

More complex problems in many physics fields involve solving ODEs beyond $m\ddot{x} = -kx$. Thankfully, most are linear with constant coefficients or can be linearized into a system to solve. The solutions are simple to calculate and analyze, but require a plethora of special cases which lead to long acronyms such as SOLHCCODE (2nd-order linear homogeneous constant-coefficient ordinary differential equation).

Finally, the Hydrogen Atom energy eigenstates can be found by solving the 2nd-order in 3-D space, 1st –order in time partial differential equation $i\hbar\frac{\partial\psi}{\partial t} = \left(\frac{-\hbar^2}{2m}\nabla^2 + \frac{e^2}{4\pi\varepsilon_0 r}\right)\psi = E\psi$. The solutions have energy differences, where when put in the equation E=hf with relativistic corrections, produce the light emission spectrum of hydrogen. Plugging in quantum numbers for energy level, orbital angular momentum, azimuthal angular momentum, and spin into a resulting equation reveal individual spectral lines.

Most importantly, putting values in any of these equations would predict further meaningful values, forming a **practical application**.

Description of Project

Problem: It is difficult to calculate necessary parameters in physics problems by hand.

Solution: Develop a program to more easily reach numerical solutions, utilizing all necessary equations.

A user of the program will likely have some interest in the fields they will use the program for, and may accelerate their exploration using the calculation features in the program. They may develop a better grasp of the physical meanings of the parameters when calculating. Finally, continuous calculation will reveal numerically the importance of input parameters and initial conditions in the trajectories, energies, and forces acting on particles and more massive objects.

The program simply reads inputs when told do so and draws calculation results on the screen. Among the ~25 possible calculations are escape velocity (from mass/radius of a body), force between two charges (from each charge and displacement), and solutions to second order linear homogeneous ordinary differential equations with constant coefficients.

TECHNOLOGY STUDENT ASSOCIATION PLAN OF WORK				
Date	**Task**	**Time involved**	**Team member responsible**	**Comments**
12/8/2014	Find problem; Derive equations; Begin process	5h	1,2	Compromised and computed; Implemented OM window/input
12/11/2014-12/18/2014	Start documentation, Continue implementation	4h	1	Added orbit display, presets; Documented progress
12/29/2014	Redesign GUI	3h	2	New images (orbits, logos, and icons)
1/8/2015-1/12/2015	Begin Presentation, Document	1h	1,2	Create, convert, and shorten process
2/15/2015-2/19/2015	Extend Program to more physics	16h	1	OSC,EM,ODE,and HATOM classes
2/20/2015	Add Help, Presets	4h	1	Added UNICODE support
3/5/2015	Update Documentation	2h	1,2	Modified processes and end user documentation to reflect new menus
3/13/2015	Compile Project	1h	1,2	Converted src/docs/presenation to pdf; burned.
Advisor signature				

Project Requirements

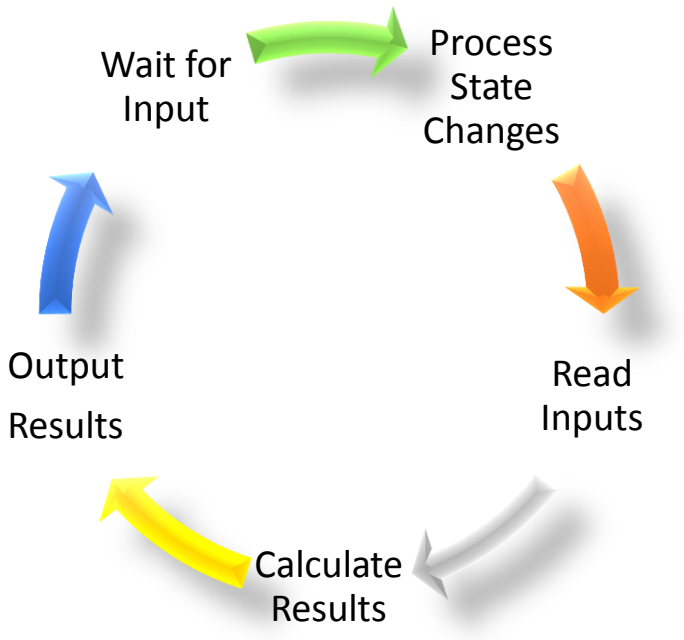
For a user to perform calculations accessibly, a Graphical User Interface must be developed and interact with the calculation portion of the program. An Application Programming Interface must be acquired, utilized, and available at runtime to display a GUI. All calculations must be reduced to functions with a series of inputs and an individual output. This derivation, though easy, should consider what information the user would know before consulting the program as to not require prior calculation from the user.

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.

High-Level Software Design

The built-in Microsoft Windows API is wrapped by the windows.h header file for the user interface. The program will be called by the functions int WINAPI WinMain (…) and LRESULT CALLBACK WindowProcedure(…). 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).

Functions of the form recalc() are called to perform derived arithmetical operations. The C++ standard library is used for math and type conversion functions. Memory is dynamically allocated or buffered where possible by implementations such as: wchar_t* str = new wchar_t[len]; … delete[] str;. Logical structures (if, switch, state machine architecture), loops (for, while), and pointers (LPCWSTR, handles) are utilized to optimize program flow.



Testing

Solution accuracy:

- Using the default values for the mass/radius of Earth, the program calculates the accurate minimum orbital (7921.3 m/s) and escape (11202 m/s) velocities.
- Given a delta v of the default rocket (3408.6m/s), and an initial velocity of roughly 8000 m/s, the rocket should be able to escape the influence of the planet, as indicated by the display.
- The ODE solver solutions match those given by Wolfram Alpha for the given inputs.
- The frequencies of light given off by differences between high energy eigenstates and n=2 states correspond accurately to within the visible spectrum.

Module unit testing:

- All three forms of input are read accurately by the program.
- All presets fill in the proper inputs.
- The GUI navigates without errors in display/overlap.

Problems/Solutions:

- UNICODE characters are converted to obfuscated ANSI
    - Implemented wide characters for dynamic support
- 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
- Module windows all shown at the same time
    - Use ShowWindow and CreateWindow to hide/redraw windows on demand.

End-User Product Documentation:
Physics Simulator Collection

Beginning:

After starting the program, PSC.exe, you will be met with the main menu window (see next page).

Select a button corresponding to the type of problem you wish to solve.

Calculating:

After entering a program window, you will be met with a series of input boxes with default values and a calculate button.

Type in the values of the input into the text boxes, and then press *Calculate* to update the outputs. (See Formatting)

Note: If there are headings, only values under the heading are relevant to calculation.

Formatting:

Inputs must be numbers from $2.2250738585072014 \times 10^{-308}$ to $1.7976931348623158 \times 10^{308}$ in one of the **3** forms:

- 25463525
- 2.5463525e7
- 2.5463525e+7

Units:

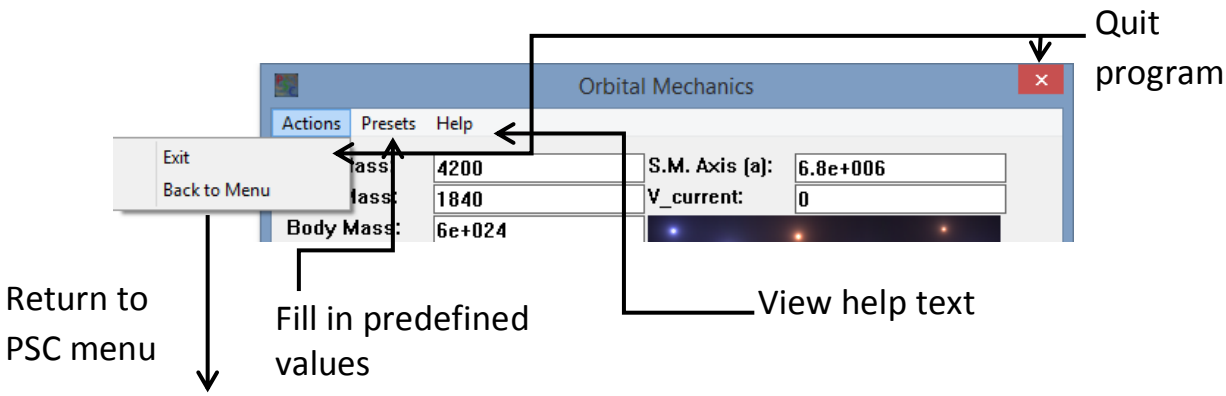
For their international recognition and functionality, all values are expressed in SI (Système International d'Unités) units.

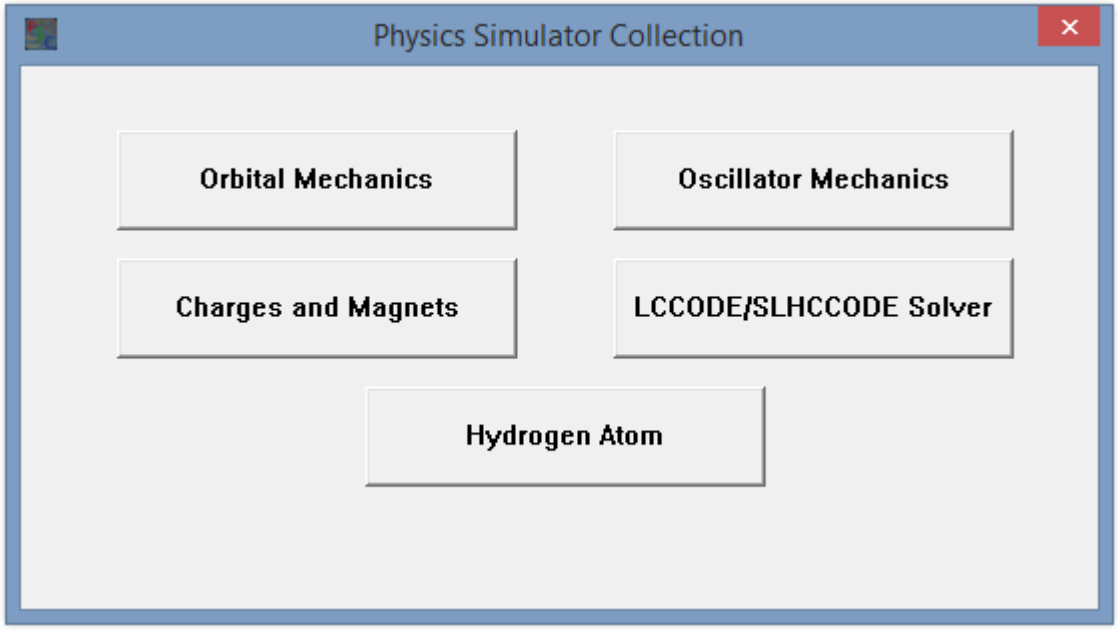
The rest of this documentation contains the meanings of each input and output, for your convenience.

Further information as to equations can be found on Google.

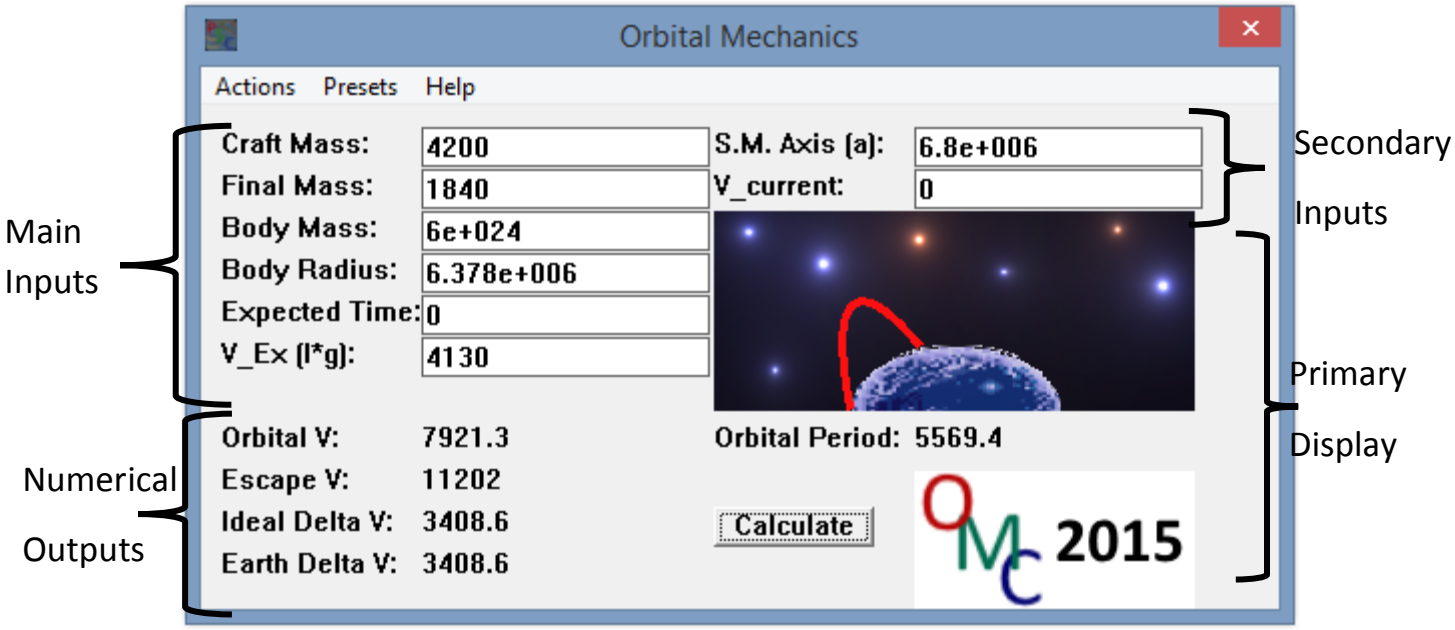
End-User Product Documentation- Graphical Breakdown

General:





Menu: Select an individual calculator or simulator



Main Inputs:

- Craft Mass: Total mass of the craft (in kg)
- Final Mass: Dry mass of the craft/mass after all fuel spent (in kg)
- Planet Mass: Total mass of the body being orbited (in kg)
- Planet Radius: Radius of the body being orbited (in m)
- Expected Time: Time expected to reach orbit/for ΔV correction (in s)
- Exhaust Velocity (Specific Impulse*g): Velocity of the exhaust relative to the craft (in m/s)

Secondary Inputs:

- Semi-Major Axis: a of orbital trajectory/for orbital period calculation (in m)
- Current Velocity: Velocity relative to orbital body for orbit classification using V+ ΔV for highest achievable speed (in m/s)

Numerical Outputs:

- Orbital Velocity: Orbital velocity of the main body (in m/s)
- Escape Velocity: Escape velocity of the main body (in m/s)
- Ideal ΔV: ΔV (highest achievable change in velocity for maneuvering) (in m/s)

Primary Display:

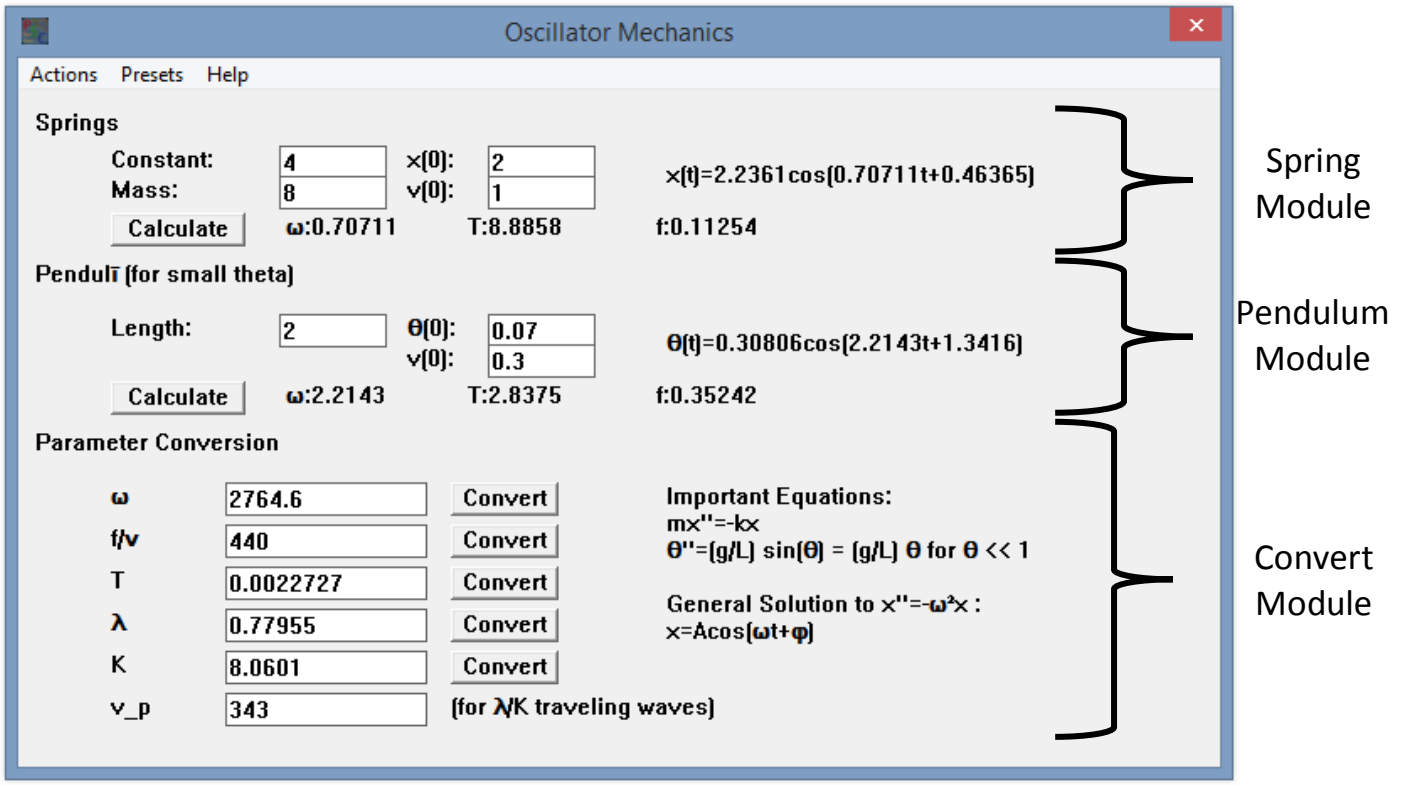
- Image: Shows type of orbit (Sub-Orbital/Orbital/Escape)
- Orbital Period: Period of the current orbit given the Semi-Major Axis of the current trajectory

Calculate Button:

Press to update display with new inputs.

Presets Tab:

Select from among various celestial bodies to fill in masses/radii.



Spring Module:

- Constant k: Spring Constant / Material Dependency (in N/m)
- Mass: Mass on Spring (in kg)
- x(0): Initial displacement from equilibrium of end of spring (in m)
- v(0): Initial rate of change of displacement from equilibrium of end of spring (in m/s)
- **Calculate Button:** Press to update display with new inputs.
- **Display:** ω: Angular Frequency (in rad/s) | T: Period of oscillation (in s) | f: Frequency (in Hz)

Pendulum Module:

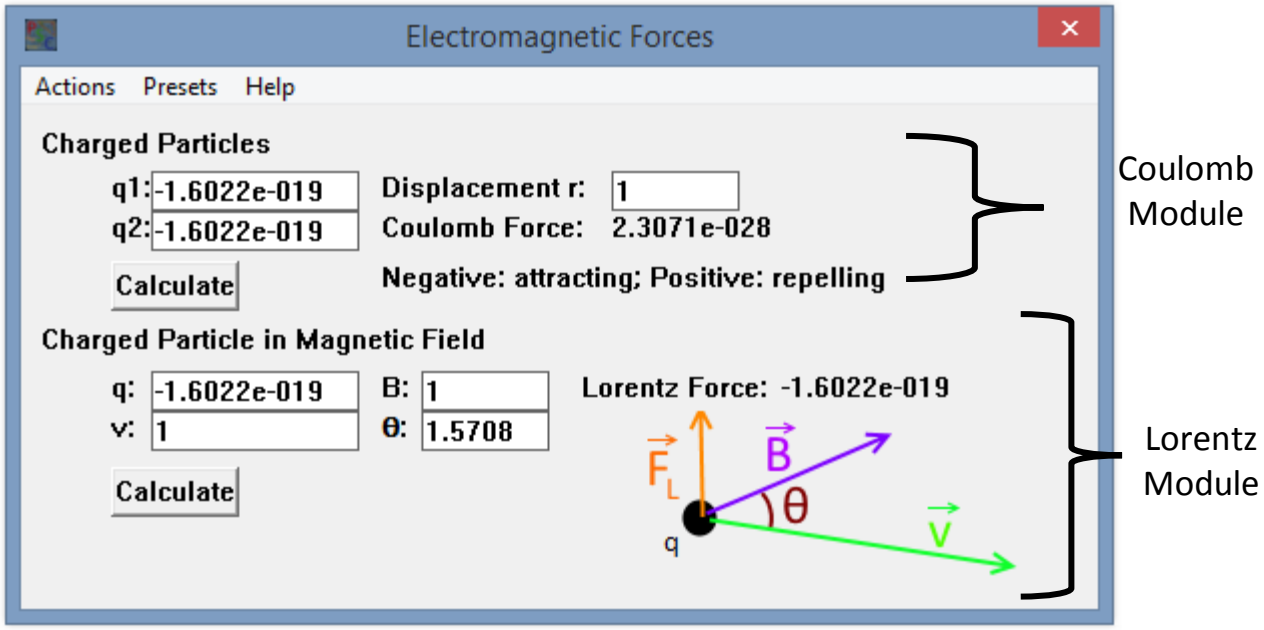
- Length l: Length of string on which mass is suspended (in m)
- θ(0): Initial angle displaced from equilibrium of end of spring (in rad)
- v(0): Initial rate of change of angle displaced from equilibrium of end of spring (in rad/s)
- **Calculate Button:** Press to update display with new inputs.
- **Display:** ω: Angular Frequency (in rad/s) | T: Period of oscillation (in s) | f: Frequency (in Hz)

Pendulum Module:

- ω: Angular Frequency (in rad/s) | f: Frequency (in Hz) | T: Period of oscillation (in s)
- λ: Wavelength (in m) | K: Wavenumber (in rad/m) | v_p Propagation velocity (in m/s)
- **Convert:** Convert value to left into the remaining parameters

Presets Tab:

- Select from among various velocities to fill in propagation velocity.



Coulomb Module:

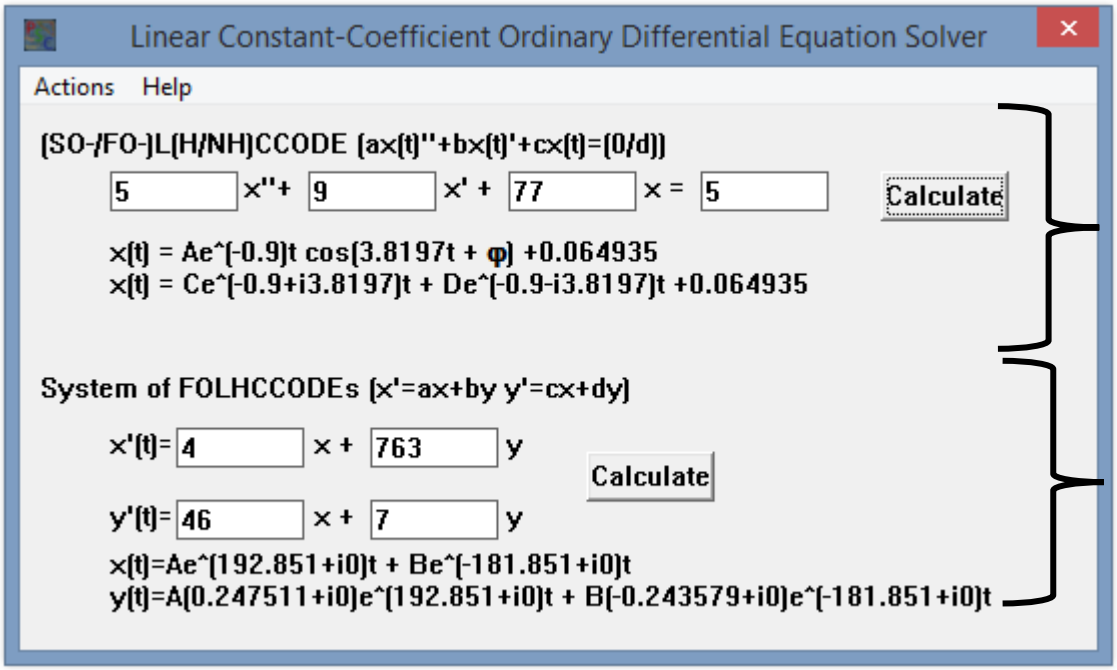
- q1: Charge of stationary particle (in C)
- q2: Charge of interacting particle (in C)
- Displacement r: Distance between the charges (in m)
- **Calculate Button:** Press to update display with new inputs.
- **Display:** Coulomb Force: Magnitude of force on each particle (in N)

Pendulum Module:

- q: Charge of interacting particle (in C)
- v: Speed of interacting particle (in m/s)
- B: Magnitude of magnetic field the particle is moving through (in T)
- θ: Angle from velocity vector to magnetic field vector (in rad)
- **Calculate Button:** Press to update display with new inputs.
- **Display:** Lorentz Force: Magnitude of force on charged particle from magnetic field (in N)

Presets Tab:

- Select from among various charges (+/- elementary charge) and common angles to fill in inputs.



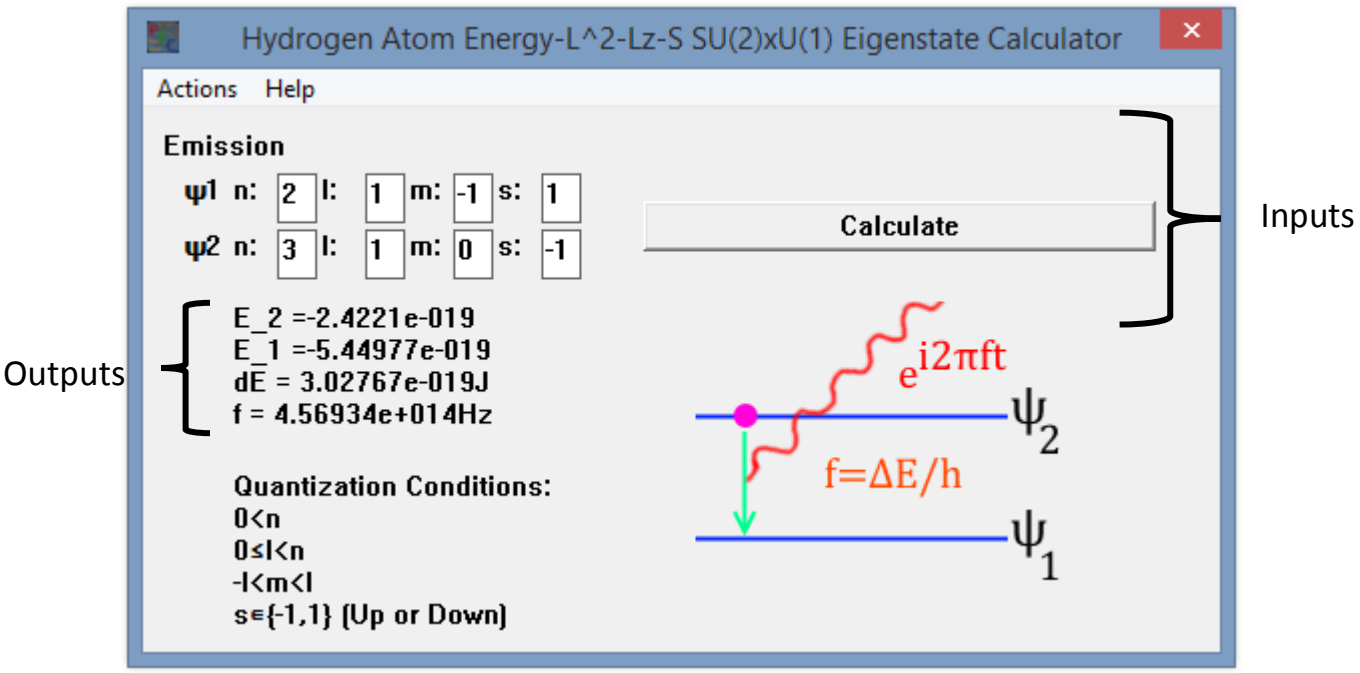
ODE Module:

- Coefficients: Constants to fill in differential equation (may be 0 for First-Order/0-Order)
- **Calculate Button:** Press to update display with new inputs.
- **Display:** Solution, x(t) = $Ae^{at} + Be^{bt}$; If r is complex, cosine is used in addition to exponential

System Module:

- Coefficients: Constants to fill in differential equations
- **Calculate Button:** Press to update display with new inputs.
- **Display:** Solution, x(t) = $Ae^{at} + Be^{bt}$, y(t) = $Ace^{at} + Bde^{bt}$
    - **Achieved using Eigenvalue Analysis algorithm**

See in-program "Help" for further understanding.



Inputs:

- n : Principle quantum number $\hat{E}\psi = \frac{E_0}{n^2}\psi$
- l : Azimuthal quantum number $\hat{L^2}\psi = \hbar^2 l(l+1)\psi$
- m : Magnetic quantum number $\widehat{L_z}\psi = \hbar m\psi$
- s : Spin projection quantum number $\hat{S_z}\psi = \frac{\hbar}{2}s\psi$
- **Calculate Button:** Press to update display with new inputs.

System Module:

- E_n: Energy of each energy eigenstate (in J)
- dE : Difference in energy between states (in J)
- f : Frequency of Light/Photon emitted in transition from ψ_2 to ψ_1 (in Hz)

Evaluation

The program development went quickly and logically after refreshing on some of the physics and API. It came out as a useful program (with only orbital mechanics), making calculations for playing Kerbal Space Program, launching amateur rockets, and checking others' math on internet forums. After additions (beyond orbital mechanics), it is now more accessible than writing a python script or looking up formulas and using a high-precision calculator (TI-83/WolframAlpha).

Eventually, the program could be expanded into a more interactive drag-and-drop simulator, which would require much more calculation and some large graphics library. This, though, wouldn't add much if the problem/terminology is already understood. An integrator-like simulation of an orbit could be implemented, but again would remove the convenience of a calculator. Further development would require extensive research in integration algorithms and plotting methods.

References

Adams, Allan. “Lecture 17: More on Central Potentials.” MIT OCW: 8.04 Quantum Physics I. MIT, Spring 2013. Lecture.

Benson, Tom J. “Ideal Rocket Equation.” *Beginner's Guide to Rockets.* NASA, n.d. Web. 11 Dec. 2014.

“Flight Dynamics.” *How Things Fly*. Smithsonian National Air and Space Museum, n.d. Web. 22 Dec. 2014.

Lewin, Walter. “Lecture 1: Electric Charges-Polarization-Electric Force - Coulomb's Law.” MIT OCW: 8.02 Electricity and Magnetism. MIT, Spring 2002. Lecture.

Lewin, Walter. “Lecture 11: Magnetic field - Lorentz Force - Torques - Electric Motors (DC) - Oscilloscope.” MIT OCW: 8.02 Electricity and Magnetism. MIT, Spring 2002. Lecture.

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.

Stern, David P. “Newton's theory of ‘Universal Gravitation’.” *From Stargazers to Starships*. NASA, 24 Mar. 2006. Web. 11 Dec. 2014.

“Windows API Index.” *Microsoft Developer Network*. Microsoft, n.d. Web. 11 Dec. 2014.

Wigand, Rob. “Calculate Escape Velocity.” NASA/JPL/MGS Education Outreach Program, 5 Feb. 1998. Web. 22 Dec. 2014.