Mystic: Implementation of the Static Dynamic Optimal Control Algorithm for High-Fidelity, Low-Thrust Trajectory Design

Gregory J. Whiffen*

Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California

Mystic software is designed to compute, analyze, and visualize optimal high-fidelity, low-thrust trajectories. The software can be used to analyze inter-planetary, planetocentric, and combination trajectories. Mystic also provides utilities to assist in the operation and navigation of low-thrust spacecraft. Mystic will be used to design and navigate the NASA’s Dawn Discovery mission to orbit the two largest asteroids. The underlying optimization algorithm used in the Mystic software is called Static/Dynamic Optimal Control (SDC). SDC is a nonlinear optimal control method designed to optimize both “static variables” (parameters) and dynamic variables (functions of time) simultaneously. SDC is a general nonlinear optimal control algorithm based on Bellman’s principal.

I. Introduction

Mystic software is designed to compute, analyze, and visualize optimal high-fidelity, low-thrust trajectories. The software can be used to analyze inter-planetary, planetocentric, and combination trajectories. Mystic also provides utilities to assist in the operation and navigation of low-thrust spacecraft. The underlying optimization algorithm used in the Mystic toolset is called Static/Dynamic optimal Control (SDC). SDC is a nonlinear optimization method designed to optimize both “static variables” (parameters) and dynamic variables (functions of time) simultaneously. SDC is a general nonlinear optimal control algorithm based on Bellman’s principal that was developed by the author. The development of SDC was inspired by the dual dynamic and parametric nature of multi-body propagated, low-thrust trajectory optimization.

Mystic software was used successfully to design NASA’s cancelled Jupiter Icy Moon Orbiter (JIMO) mission’s reference trajectory. The JIMO mission had proposed to send a single electric propulsion spacecraft to orbit Callisto, Ganymede, and Europa in succession. The JIMO trajectory is likely the most complex trajectory ever designed for a space mission. Currently, Mystic is being used to build the flight trajectory and will assist navigation of NASA’s DAWN Discovery mission that will orbit the two largest asteroids in the solar system after getting a gravity assist from Mars. Results from Mystic have been published in numerous papers.

The tool described in this paper is designed to compute optimal low-thrust trajectories by maximizing final spacecraft net mass or by minimizing a user defined infeasibility. Infeasibility in this context is measured by the magnitude of constraint violation. Trajectories that are over-constrained have no feasible solution. Mystic will minimize constraint violation when a trajectory is over-constrained. Trajectories are fully integrated using finite burns, multi-body propagation, gravitational harmonics, and solar radiation pressure. Any or all major planets, natural satellites, asteroids, and comets can be included in the gravitating body set. Mystic trajectories are continuous from beginning to end, there are no “matching” points at intermediate times.

As a result of the multi-body formulation, Mystic can, in many cases, identify advantageous intermediate flybys and multi-body interactions on its own. For example, if the user specifies a final state of a rendezvous with Mercury, but does not specify any intermediate flybys, Mystic often locates favorable flyby(s) of Venus.

*Senior Engineer, Navigation and Mission Design Section

American Institute of Aeronautics and Astronautics
and/or Mercury at intermediate times. Mystic is of great utility when multi-body or non-spherical body gravity effects cannot be neglected. Often Mystic exploits complex gravity for performance improvements. Example of trajectories for which Mystic provides great advantage include: Earth orbit to Moon orbit transfers, planetary capture, Earth capture or escape using the Moon, planet or asteroid centered spiraling, and libration point maneuvers.

The SDC algorithm in the Mystic optimization engine can optimize trajectories that have several regimes with widely varying length and time scales. Complex, multi-regime trajectories do not need to be broken up into separate optimization problems. Specifically, SDC can optimize trajectories beginning in orbit around one planet and ending in orbit around another planet or another planet’s moon.

The general limitations on the types of trajectories Mystic can optimize correspond directly to the stability and accuracy limitations for multi-body propagation. This is because SDC trajectories are modeled as continuous from beginning to end. For example, there is a limit of approximately 3 close flybys of massive bodies separated by interplanetary distances or 6 close flybys of massive moons around a single planet in a single trajectory. More flybys cannot be accurately propagated end-to-end and hence optimized by this tool. Also, since trajectories are modeled as continuous from beginning to end, it can take time to set up a meaningful initial guess for some multi-flyby problems. Trajectory problems with more flybys need to optimized in a piece-wise fashion. For spiraling problems, fully optimized trajectories are limited to less than 250 revolutions for practical run-times. However, Mystic can design sub-optimal trajectories of 300 or more revolutions by using sophisticated control laws developed by Petropoulos.

Mystic software was designed primarily for Solar Electric Propulsion (SEP) and Nuclear Electric Propulsion (NEP) design. No attempt has been made to accommodate solar sail trajectory design.

II. Architecture of Mystic

Mystic tools can be divided into 5 categories. The first is the Optimization Engine which iteratively computes an optimal trajectory given a problem specification and an initial guess. The second category is the Guesstool graphical user interface which can perform high precision propagation and be used to construct a problem specification and initial guess file for the Optimization Engine. The third category is trajectory visualization software including high fidelity trajectory animation rendering. The fourth category is high- and low-level post- and pre-processors for analysis and data entry called the “Analysis Toolkit.” The fifth category is the Scripting Interface which is a set of functions that allow any or all of Mystic functions to be scripted for complex or repetitive analysis tasks.

Mystic software tool set consists of 80,000 lines of Fortran 95 code and 36,000 lines of Matlab code. All numerically intensive operations (like high-fidelity propagation and optimization) are performed in compiled Fortran 95 code. Matlab provides advanced graphics, graphical user interfaces, and a scripting environment.

![Diagram](image.png)

**Figure 1. Illustration of the typical use of Mystic tools for ordinary trajectory optimization**

The standard usage of the Mystic tool set for trajectory optimization is illustrated in Figure 1. Beginning
at the left side of Figure 1, either a “cold” or a “warm” start is used for setting up new trajectory optimization problems. A cold start involves using one of the supplied template files. The user selects a template file from a menu on the main Guess tool control panel that is as close to the desired problem as possible. Generally, template files require a fair amount of modification through the Guess tool GUI to adapt to a new problem’s requirements and models. A warm start involves loading an existing problem input file into the Guess tool GUI that is very similar to the new problem to be solved. A warm start implies that the file that is loaded requires few if any fundamental modifications to proceed to solve the new problem of interest. All brand new analysis must use a cold start (either using a supplied template or an unrelated input file from some very different analysis). In practice, most problems are set up using a warm start, since most analysis involves repeatedly solving problems with small modifications.

Once either a warm (related input file) or cold start (template or unrelated input file) is selected, the Guess tool GUI is used to modify the input file for the new purposes. All models, constraints and the initial guess for all optimization variables can be constructed with Guess tool. The initial trajectory propagation can be viewed and analyzed while using Guess tool. After propagating a trajectory with Guess tool, the full functionality of the Analysis toolkit and plotting tools are available to the user. Figure 1 illustrates the connection between Guess tool and the Analysis toolkit and plotting tools. After Guess tool is used to build a new input file for the Optimization Engine, the input file can be used by the Optimization Engine to compute a new optimal trajectory. The Optimization Engine can be invoked with either a simple command line expression at a shell prompt, a matlab shell escape command, or through the use of the Mystic Scripting Interface. Once the Optimization Engine has begun to work on the problem, the full functionality of the Analysis toolkit and plotting tools are available to the user to analyze the progress of the optimization or analyze the final results in detail.

When the Optimization Engine completes its work, a new set of values for every optimization variable is written to a file that can be loaded back into Guess tool. Guess tool can be used to analyze the new optimal trajectory in detail or modify the solution for a new purpose (a “warm start”).

II.A. The Mystic Scripting Interface

The standard usage of the Mystic tool set for trajectory optimization is illustrated in Figure 1. The arrows in Figure 1 represent a human in the loop either making decisions, pressing a GUI button, or issuing commands at a command prompt. An alternative to having a human in the loop is provided by the Mystic Scripting Interface. The interface allows complete automation of all of the processes illustrated in Figure 1. All Guess tool, Analysis Toolkit, Plotting tools, and Mystic Optimization Engine features can be controlled with convenient Matlab script commands that can make decisions, push GUI buttons and issue commands. Repetitive analysis like parametric trade studies can be carried out in a fully automated way. In addition, Monte-Carlo analysis that involve propagation and optimization, or propagation only can be conveniently scripted.

The full mathematical and graphical power of Matlab can be used in the Mystic Scripting Interface to analyze results and make decisions for further analysis. Complex trade study results can be plotted in any way the user wants through the use of the interface. Some knowledge of Matlab scripting is needed to obtain the full power of the Mystic Scripting Interface.

II.B. The Guess tool Graphical User Interface

“Guess tool” refers to a Matlab based graphical user interface and a set of compiled Fortran programs that are designed to work together help a user to set up input files for the Optimization Engine, provide a high-fidelity propagator, create high quality graphics, and provide a stand alone rapid sub-optimal trajectory design capability. Guess tool can be used to rapidly set up and carry out trajectory propagations (ballistic or feedback law controlled). Guess tool links to and uses the same trajectory propagator as is used by the Optimization Engine. Guess tool provides the ability to rapidly design high-fidelity sub-optimal orbital transfers around a single body by using simple feedback laws or the advanced feedback law called Q-Law. Guess tool allows full access to all of Mystic’s plotting tools and analysis tools in the Analysis Toolkit. All of the capabilities of Guess tool can be incorporated into user written scripts for automated analysis.

The original (and still main) function of Guess tool is to create initial guesses for all optimization variables,
provide an easy way to specify all models, constraints, and objectives, and write what is called the “Standard Input File”. Guesstoool helps tremendously when it is necessary to generate complete trajectory guess from scratch, or edit all or part of an existing solution for new purposes. Guesstoool allows the user to manipulate most of the important parameters in the input file. Once a change is made, the resulting trajectory guess can be immediately graphically inspected and it can be analyzed using any of the tools in the Analysis Toolkit. The standard input file written by Guesstoool is used “as is” with the Optimization Engine.

To make finding particular functions on the control panel easier, controls (buttons, menus, etc.) are given the same color if they have related functions. If the mouse is placed over an active control or button, after a few seconds a “tool-tip” message will appear with a short description of the button or control’s purpose. Figure 2 is an example of the appearance of the main control panel of Guesstoool. The main panel can be used to launch 16 additional auxiliary control panels for model, constraint, targeting, and convergence parameter inputs. There are a total of more than 1000 functions available in the Guesstoool graphical interface.

II.C. Trajectory Visualization

Mystic provides highly customizable graphics output for trajectory analysis and high quality presentation style graphics. Static plots permit any perspective in inertially fixed frames, instantaneously inertial fixed frames, and rotating frames. Plots can be automatically annotated with trajectory events like flybys, escape, capture, penumbra and umbra crossing, and initial and final state conditions. Examples of trajectory visualizations are given in Figures 3 and 4.

The “3D Movie” button (bottom middle of Figure 2) will create a fly-through trajectory animation displaying what a spacecraft’s camera will see given the most recently constructed trajectory. A good Open GL graphics card is necessary to get the full benefit of the 3D movie feature. Figures 6 through 7 are example frames generated using the 3D Movie animation capability of Mystic.

II.D. The Analysis Toolkit

The Analysis Toolkit is a set of pre- and post-processors available at the Matlab command line. There are more than 100 different tools to choose from at present. The functionality of the tools include: plotting and fitting spacecraft models, spacecraft ranges and body relative angles, spacecraft orbital elements, thrust beam angles, thrust beam angular velocity and acceleration, creating tables of trajectory data, and a host of low level functions that can be used in custom scripts. High level tools (usually interactive) are named a capital first letter and with there function like “Plot” or “Fit” first. Low-level (usually non-interactive) functions are named beginning with a lower case letter. Low-level functions are useful when using the Mystic Scripting Interface. Help messages are available for each tool.

Figure 8 is an example of using the Plot_body_body_angle analysis tool on an optimal trajectory from Earth to Mars flyby to Vesta rendezvous. The plot in Figure 8 is centered on the Mars gravity assist, showing the Earth and Mars angles with an overlay of the Mars range.

III. Models and Constraints

Mystic’s central function is to optimize low-thrust trajectories. The optimization variables include thrust as a function of time, launch date, flight time (or equivalently arrival date), up to six initial condition parameters, and the initial mass or the “injected mass increment”. Injected mass increment is an optimizable deviation from a given launch vehicle’s maximum performance curve. State variables that are optimized include position, velocity, and mass of a spacecraft as a function of time. The general problem that Mystic’s Optimization Engine solves is

\[
J(v(t), w) = \min_{v(t), w} \int_{t_0}^{t_N} F(x(t), v(t), w, t) dt + \sum_{i=1}^{N} H_i(x(t_i), u_i, w, t_i) + G(x(t_N), u_N, w, t_N),
\]

subject to a state equation of the form

\[
\frac{dx(t)}{dt} = T(x(t), v(t), w, t),
\]
Figure 2. The main control panel for the graphical user interface Guesser.
and an initial condition of the form

\[ x(t_0) = \Gamma(w), \]  

(3)

where the functions \( F \) and \( G \) are user selectable, once continuously differentiable objective functions; the function \( T \) represents the physical interactions; and the function \( \Gamma \) returns the initial system state. There are three classes of variables: state variables \( x(t) \), dynamic control variables \( v(t) \), and static control variables \( w \). In the current Mystic implementation, the components of the state vector \( x(t) \) are

\[ x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \\ x_5(t) \\ x_6(t) \\ x_7(t) \end{bmatrix} = \begin{bmatrix} x \text{ coordinate of spacecraft} \\ y \text{ coordinate of spacecraft} \\ z \text{ coordinate of spacecraft} \\ x \text{ velocity of spacecraft} \\ y \text{ velocity of spacecraft} \\ z \text{ velocity of spacecraft} \\ \text{mass of the spacecraft.} \]  

(4)
The dynamic and static control vectors $v(t)$ and $w$ are defined to be

$$v(t) = \begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} = \begin{bmatrix} x \text{ thrust} \\ y \text{ thrust} \\ z \text{ thrust} \end{bmatrix}$$

$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \end{bmatrix} = \begin{bmatrix} \text{transfer start} \\ \text{total flight time.} \\ \text{initial} V_x \text{ or} \Omega \\ \text{initial} V_y \text{ or} \omega \\ \text{initial} V_z \text{ or} \nu \\ \text{initial} X, a, \text{ or} C_3 \\ \text{initial} Y, e, \text{ or} R_p \\ \text{initial} Z \text{ or} i \\ \text{initial mass or increment} \end{bmatrix}$$

(5)

where $V_x, V_y, V_z, X, Y, and Z$ are the spacecraft initial velocity and position; $\Omega, \omega, a, e, i$ are initial spacecraft classical orbital elements; $C_3$ is the initial spacecraft orbital energy; and $R_p$ is the initial spacecraft orbital periapsis distance.

The equation used to describe the time evolution of the state is

$$\frac{dx}{dt} = T(x, v, w, t) = \begin{bmatrix} x_4(t) \\ x_5(t) \\ x_6(t) \\ x_7(t) \end{bmatrix} + \sum_{i=1}^{N_i} a_x(\text{body} = i, t, x_{1:3}) + \text{SRP}_x$$

$$+ \sum_{i=1}^{N_i} a_y(\text{body} = i, t, x_{1:3}) + \text{SRP}_y$$

$$+ \sum_{i=1}^{N_i} a_z(\text{body} = i, t, x_{1:3}) + \text{SRP}_z$$

$$- \dot{m}(||v||, x_{1:3}, t) = \text{mass flow rate.}$$

(6)

The propellant mass flow rate $\dot{m}$ is a function of the magnitude of the thrust control vector $||v||$. The power available to the thrusters can be a function of the distance from the Sun, the non-thruster spacecraft power consumption, and time due to power generation degradation. The parameter $N_i$ is the number of gravitating
bodies that are modeled. The gravitational acceleration vector \( a \equiv (a_x, a_y, a_z) \) resulting from each body is given by a time dependent matrix times the gradient of a harmonic expansion of the gravitational potential,

\[
a(r, \lambda, \phi, t) = [Q_{body}](t) \cdot \nabla \left\{ \mu_{body} \sum_{n=0}^{\infty} \frac{P_{nm}}{r^n} P_{nm}(\sin(\phi))(C_{nm}\cos(m\lambda) + S_{nm}\sin(m\lambda)) \right\}
\]  

(7)

where \([Q_{body}](t)\) is a rotation matrix from the body fixed rotating frame of the oblate body to an inertial (non-rotating) frame, \(\mu_{body}\) is the gravitational constant of the body, \(r\) is the scalar radius vector from the body center of mass to the spacecraft, \(R_{body}\) is the semi-major axis of the oblate body’s ellipsoid used in the expansion, \(P_{nm}\) are the associated Legendre polynomials of degree \(n\) and order \(m\), \(\lambda\) is the body fixed longitude of the spacecraft, and \(\phi\) is the body fixed latitude of the spacecraft. \(C_{nm}\) and \(S_{nm}\) are the harmonic expansion constants. “SRP” is the solar radiation force.

The SDC algorithm was implemented using full second order analytic derivatives (in both time and space) of the harmonic expansion (7) and all constraints. Optimal solutions generated by SDC satisfy both the necessary and sufficient conditions of optimality.

Mystic has numerous optional constraints on trajectory geometry, propellant usage, intermediate targeting, and the dynamic range and rate of the control. For example, many different final spacecraft states can be specified. The final states constraints available include 19 different low-level constraints and 8 high-level constraints. The high-level constraints are commonly useful combinations of low-level constraints. For example general orbital element targeting is one of the high level constraints. Any subset of orbital elements can be fixed or free for optimization. The low-level constraints can be combined by the user in novel ways to achieve a large number of different possible custom terminal conditions.

Mystic provides constraints on propellant mass when there is a launch vehicle performance curve define or the initial mass is free in the problem. Mystic also provides thrust beam cone angle constraints. Thrust beam constraints can be necessary for communication sessions and to avoid Sun light from entering a thruster and causing it to overheat.

Constraints on trajectory geometry include: minimum flyby distances, forced flybys, launch asymptote and energy constraints, and intermediate targeting constraints (for example intermediate rendezvous.)

Mystic provides spacecraft models including solar array performance, solar array or nuclear electric power degradation, spacecraft bus (non-electric thrusting) power consumption, thruster mass flow as a function
of input power, and thruster thrust magnitude as a function of input power. In addition, thrusters can be modeled using a “throttle table” which is a list of discrete operating points for a given thruster. The ability to design trajectories with a throttle table is essential for operational support.

Mystic also provides the ability to alter the initial state of an existing sensitive trajectory to a new user specified initial state and re-optimize the trajectory quickly. This ability is designed to incorporate new orbit determinations into the optimal reference trajectory quickly during flight.

IV. Conclusion

Mystic software is designed to compute, analyze, and visualize optimal high-fidelity, low-thrust trajectories. The software can be used to analyze inter-planetary, planetocentric, and combination trajectories. Mystic also provides utilities to assist in the operation and navigation of low-thrust spacecraft. The underlying optimization algorithm used in the Mystic toolset is called Static/Dynamic optimal Control. The Static/Dynamic optimal Control algorithm (SDC) is a nonlinear optimization method designed to optimize both “static variables” (parameters) and dynamic variables (functions of time) simultaneously.

Mystic software was used successfully to design NASA’s cancelled Jupiter Icy Moon Orbiter (JIMO) mission’s reference trajectory. The JIMO trajectory is likely the most complex trajectory ever designed for a space mission. Currently, Mystic is being used to perform mission design and build the thrust command sequence for the flight trajectory for NASA’s DAWN Discovery mission that will orbit the two largest asteroids in the solar system.

Acknowledgments

The software development described in this paper was carried out at Optimal Dynamics, Inc. and the Jet Propulsion Laboratory, California Institute of Technology, under contract to the National Aeronautics and Space Administration through the In Space Propulsion Technology Program, Prometheus Nuclear Space and Technology Program, and the Dawn Discovery mission.

References


Figure 6. Example frames from the 3D Movie animation movie produced from Cassini-like trajectory around Saturn.
Figure 7. Example frame from the 3D Movie animation movie produced from Cassini-like trajectory around Saturn.

Figure 8. The results of an analysis toolkit interactive tool applied to an optimal Earth-Mars-Vesta rendezvous (Mars flyby portion.)