Numerical Computation of a Continuous-Thrust State Transition Matrix Incorporating Accurate Hardware and Ephemeris Models

Donald Ellison
Bruce Conway
Jacob Englander

NAVIGATION & MISSION DESIGN BRANCH
NASA GSFC

www.nasa.gov
Authors

- Donald Ellison – Ph. D. Student, Department of Aerospace Engineering University of Illinois at Urbana-Champaign

- Bruce Conway – Professor, Department of Aerospace Engineering University of Illinois at Urbana-Champaign

- Jacob Englander – Aerospace Engineer, Navigation & Mission Design Branch, Code 595, NASA Goddard Space Flight Center
Outline

- Evolutionary Mission Trajectory Generator (EMTG)
- Finite-burn low-thrust (FBLT) transcription
- FBLT vs. multiple gravity-assist with low-thrust (MGALT) transcription
- FBLT match point constraint gradient calculation
- Continuous-thrust state transition matrix calculation
- Adaptive-step RK7(8)13M integrator
- FBLT match point time of flight gradient calculation
- Numerical Example
- Summary
The Evolutionary Mission Trajectory Generator (EMTG)

- Automated interplanetary low-thrust mission design tool
- Capabilities:
  - Global optimization of both the flyby sequence and the trajectory
  - No initial guess required, global search capability provided by monotonic basin hopping (MBH) algorithm
  - Works in multiple reference frames (interplanetary, moon tours, Earth orbit, etc)
  - Integration with SPICE ephemerides
  - Up-to-date models of launch vehicles, thrusters, power systems
  - Easy to use graphical user interface
- The purpose of EMTG is to automate almost all of the work so that an analyst can investigate a wide range of mission options in very little human time
- Computer time is CHEAP. Analyst time is EXPENSIVE
Finite-Burn Low-Thrust Transcription

\[
\ddot{r}_{cb-s/c} = -\frac{G}{r_{cb-s/c}^3} \left( m_{cb} + \sum_i G m_i \right) - \sum_i G m_i \left[ \frac{r_{cb-s/c} - r_{cb-3b}}{\|r_{cb-s/c} - r_{cb-3b}\|^3} \right] + \frac{\mathbf{u} \cdot D T}{m_s/c}
\]

\[
\dot{m} = -\|\mathbf{u}\| D \dot{m}_{max}
\]

**FBLT vs. MGALT**

<table>
<thead>
<tr>
<th>p</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_0$</td>
<td>Launch epoch</td>
<td></td>
</tr>
<tr>
<td>$v_\infty$</td>
<td>Magnitude of the outgoing velocity asymptote</td>
<td>first phase only</td>
</tr>
<tr>
<td>RA</td>
<td>Right ascension of launch asymptote</td>
<td>first phase only</td>
</tr>
<tr>
<td>DEC</td>
<td>Declination of launch asymptote</td>
<td>first phase only</td>
</tr>
<tr>
<td>TOF</td>
<td>Phase time of flight</td>
<td>$N_p$</td>
</tr>
<tr>
<td>$m_f$</td>
<td>Phase final mass</td>
<td>$N_p$</td>
</tr>
<tr>
<td>$u$</td>
<td>Segment control vector</td>
<td>one per FBLT segment ($3N \times N_p$)</td>
</tr>
<tr>
<td>$I_{sp}$</td>
<td>Propulsion system specific impulse</td>
<td>only for VSI-capable engines ($N \times N_p$)</td>
</tr>
<tr>
<td>$v_{\infty_0}$</td>
<td>Phase initial excess velocity vector</td>
<td>all but the first phase ($3N_p - 1$)</td>
</tr>
<tr>
<td>$v_{\infty_f}$</td>
<td>Phase final excess velocity vector</td>
<td>all phases, except the final phase of a rendezvous ($N_p$)</td>
</tr>
</tbody>
</table>

**Table 1:** Typical decision vector for a low-thrust mission FBLT mission.

---

**Navigation & Mission Design Branch, Code 595**

NASA GSFC
MGALT and FBLT share identical decision and constraint vectors
Control bounds, flyby model and TOF constraint gradients may be provided fully analytically

\[ \|u\| = \sqrt{u_{x_k}^2 + u_{y_k}^2 + u_{z_k}^2} \leq 1 \]

\[ c_{v_\infty} = v_\infty^+ - v_\infty^- = 0 \]

\[ c_{\text{flyby-altitude}} = r_{\text{periapse}} - (r_{\text{planet}} + h_{\text{safe}}) \geq 0 \]
\[ = \frac{\mu_{\text{planet}}}{v_\infty^2} \left[ \frac{1}{\sin(\delta/2)} - 1 \right] - (r_{\text{planet}} + h_{\text{safe}}) \geq 0 \]

\[ c_{\text{TOF}} = t_{\text{min}} \leq t_{\text{flight}} = \sum_{i=1}^{N_p} t_{\text{phase}_i} \leq t_{\text{max}} \]

\[ \delta = \arccos \left[ \frac{v_\infty^- \cdot v_\infty^+}{(v_\infty^-)^2 (v_\infty^+)^2} \right] \]
FBLT as a Nonlinear Program: Constraints

- FBLT transcription results in a large, sparse NLP
- We want to calculate the gradient of each NLP constraint w.r.t. each NLP parameter (KKT first order necessary conditions) → form the problem Jacobian
- SNOPT will do this with finite differencing…this is EXPENSIVE
- FBLT is already much slower than MGALT due to numerical integration vs. analytic Kepler, so it is even more important to supply user-defined derivatives
- The most challenging are the match point continuity constraints: state transition matrices

\[
c_{mp} = X_{mp_b} - X_{mp_f}
\]

\[
\frac{\partial c_{mp}}{\partial p} = \frac{\partial X_{mp_b}}{\partial p} - \frac{\partial X_{mp_f}}{\partial p}
\]
Traditional six-by-six STM is augmented to include mass and the current time step’s control parameters $u_x, u_y$ and $u_z$

Match point constraint vector sensitivities w.r.t. controls make up the majority of the dense entries in the Jacobian

An STM chain is constructed beginning with the segment of interest and proceeding using “stripped” STMs to the match point

\[
\Phi(t, t_0) = \begin{bmatrix}
\frac{\partial r}{\partial r_0} & \frac{\partial r}{\partial m_0} & \frac{\partial r}{\partial u_0} \\
\frac{\partial \dot{r}}{\partial r_0} & \frac{\partial \dot{r}}{\partial m_0} & \frac{\partial \dot{r}}{\partial u_0} \\
\frac{\partial m}{\partial r_0} & \frac{\partial m}{\partial m_0} & \frac{\partial m}{\partial u_0} \\
\frac{\partial u}{\partial r_0} & \frac{\partial u}{\partial m_0} & \frac{\partial u}{\partial u_0}
\end{bmatrix}, \quad \Phi_s(t, t_0) = \begin{bmatrix}
\frac{\partial r}{\partial r_0} & \frac{\partial r}{\partial r_0} & \frac{\partial r}{\partial m_0} \\
\frac{\partial \dot{r}}{\partial r_0} & \frac{\partial \dot{r}}{\partial r_0} & \frac{\partial \dot{r}}{\partial m_0} \\
\frac{\partial m}{\partial r_0} & \frac{\partial m}{\partial r_0} & \frac{\partial m}{\partial m_0} \\
\frac{\partial u}{\partial r_0} & \frac{\partial u}{\partial r_0} & \frac{\partial u}{\partial m_0}
\end{bmatrix} \begin{bmatrix}
0_{3 \times 3} \\
0_{3 \times 3} \\
0_{1 \times 3} \\
0_{3 \times 3}
\end{bmatrix}
\]

\[
\Phi(t_{mp}, t_i) = \Phi_s(t_{mp}, t_{mp-1}) \cdot \ldots \cdot \Phi_s(t_{i+2}, t_{i+1}) \cdot \Phi(t_{i+1}, t_i)
\]
FBLT State Transition Matrix Calculation

- Unlike the two-body problem (see Battin for example), analytic expressions for the perturbation STM entries don’t exist for true continuous thrust
- We can solve for them numerically
- Differential equations for the STM entries are appended to the physics EOM’s and integrated alongside them
- We must calculate the state propagation matrix (A) at each integration time step
- A also referred to as the fundamental solution/set

\[
\dot{X} = f = \begin{bmatrix} \dot{r} & \ddot{r} & \dot{m} & \dot{u}_x & \dot{u}_y & \dot{u}_z & \dot{s}_{11} & \dot{s}_{12} & \ldots & \dot{s}_{1010} \end{bmatrix}^T
\]

\[
\dot{\Phi} = A \Phi
\]


FBLT State Transition Matrix Calculation

\[ \dot{\Phi} = A \Phi \]
\[ A = \frac{\partial \dot{X}}{\partial X} \]

\[
\Phi = \begin{bmatrix}
  s_{11} & s_{12} & s_{13} & s_{14} & s_{15} & s_{16} & s_{17} & s_{18} & s_{19} & s_{110} \\
  s_{21} & s_{22} & s_{23} & s_{24} & s_{25} & s_{26} & s_{27} & s_{28} & s_{29} & s_{210} \\
  s_{31} & s_{32} & s_{33} & s_{34} & s_{35} & s_{36} & s_{37} & s_{38} & s_{39} & s_{310} \\
  s_{41} & s_{42} & s_{43} & s_{44} & s_{45} & s_{46} & s_{47} & s_{48} & s_{49} & s_{410} \\
  s_{51} & s_{52} & s_{53} & s_{54} & s_{55} & s_{56} & s_{57} & s_{58} & s_{59} & s_{510} \\
  s_{61} & s_{62} & s_{63} & s_{64} & s_{65} & s_{66} & s_{67} & s_{68} & s_{69} & s_{610} \\
  s_{71} & s_{72} & s_{73} & s_{74} & s_{75} & s_{76} & s_{77} & s_{78} & s_{79} & s_{710} \\
  s_{81} & s_{82} & s_{83} & s_{84} & s_{85} & s_{86} & s_{87} & s_{88} & s_{89} & s_{810} \\
  s_{91} & s_{92} & s_{93} & s_{94} & s_{95} & s_{96} & s_{97} & s_{98} & s_{99} & s_{910} \\
  s_{101} & s_{102} & s_{103} & s_{104} & s_{105} & s_{106} & s_{107} & s_{108} & s_{109} & s_{1010}
\end{bmatrix}
\]
FBLT State Transition Matrix Calculation

\[ \dot{\Phi} = A \Phi \]

\[ A = \frac{\partial \dot{X}}{\partial X} \]
Calculation of the State Propagation Matrix

\[ A = \begin{bmatrix}
  0 & \mathbb{I}_{3 \times 3} & 0 & 0 \\
  A_{21} & 0 & A_{23} & A_{24} \\
  A_{31} & 0 & 0 & A_{34} \\
  0 & 0 & 0 & 0 
\end{bmatrix} \]

\[ A_{21} = \frac{\partial \mathbf{r}_{cb-s/c}}{\partial \mathbf{r}_{cb-s/c}} = \frac{3 \mu_{cb} \mathbf{r}_{cb-s/c} \mathbf{r}_{cb-s/c}^T}{r_{cb-s/c}^5} - \frac{\mu_{cb}}{r_{cb-s/c}^3} \mathbb{I}_{3 \times 3} + \frac{3 \mu_{3b} \mathbf{r}_{3b-s/c} \mathbf{r}_{3b-s/c}^T}{r_{3b-s/c}^5} - \frac{\mu_{3b}}{r_{3b-s/c}^3} \mathbb{I}_{3 \times 3} + \frac{\mathbf{u}}{m_{s/c}} \cdot \frac{\partial T}{\partial P} \cdot \frac{\partial P}{\partial r_{\odot-s/c}} \cdot \frac{\partial r_{\odot-s/c}}{\partial \mathbf{r}_{cb-s/c}} \]
Calculation of the State Propagation Matrix

\[ A = \begin{bmatrix} \mathbb{I}_{3\times3} & 0 & 0 & 0 \\ \hline \\ A_{21} & 0 & A_{23} & A_{24} \\ \hline \\ 0 & 0 & 0 & A_{34} \\ \hline \\ 0 & 0 & 0 & 0 \end{bmatrix} \]

\[ A_{21} = \frac{\partial \mathbf{r}_{cb-s/c}}{\partial \mathbf{r}_{cb-s/c}} \]
\[ = \frac{3\mu_{cb} \mathbf{r}_{cb-s/c}^{T} \mathbf{r}_{cb-s/c}}{r_{cb-s/c}^{5}} - \frac{\mu_{cb}}{r_{cb-s/c}^{3}} \mathbb{I}_{3\times3} + \frac{3\mu_{3b} \mathbf{r}_{3b-s/c}^{T} \mathbf{r}_{3b-s/c}}{r_{3b-s/c}^{5}} - \frac{\mu_{3b}}{r_{3b-s/c}^{3}} \mathbb{I}_{3\times3} \]
\[ + \frac{\mathbf{u} \ D \ \partial T}{m_{s/c}} \cdot \frac{\partial P}{\partial T} \cdot \frac{\partial r_{\odot-s/c}}{\partial r_{cb-s/c}} \cdot \frac{\partial r_{\odot-s/c}}{\partial r_{cb-s/c}} \]

NAVIGATION & MISSION DESIGN BRANCH, CODE 595
NASA GSFC
Launch Vehicle, Thruster, and Power Modeling

- Launch vehicles are modeled using a polynomial fit
  \[ m_{\text{delivered}} = (1 - \sigma_{LV}) \left( f_{LV}C_3^5 + e_{LV}C_3^4 + d_{LV}C_3^3 + c_{LV}C_3^2 + b_{LV}C_3 + a_{LV} \right) \]
  where \( \sigma_{LV} \) is launch vehicle margin and \( C_3 \) is hyperbolic excess velocity

- Thrusters are modeled using either a polynomial fit to published thrust and mass flow rate data
  \[ \dot{m}_{\text{max}} = e_F P^4 + d_F P^3 + c_F P^2 + b_F P + a_F \]
  \[ T_{\text{max}} = e_T P^4 + d_T P^3 + c_T P^2 + b_T P + a_T \]
  or, when detailed performance data is unavailable
  \[ T_{\text{max}} = \frac{2 \eta P}{I_{sp} g_0} \]
  \[ \dot{m}_{\text{max}} = \frac{T_{\text{max}}}{I_{sp} g_0} \]

- Power is modeled by a standard polynomial model
  \[ \frac{P_0}{r^2} \left( \frac{\gamma_0 + \frac{\gamma_1}{r} + \frac{\gamma_2}{r^2}}{1 + \gamma_3 r + \gamma_4 r^2} \right) (1 - \tau)^t \]

\( P_0 \) is the power at beginning of life at 1 AU, \( \tau \) is the solar array degradation constant and \( t \) is the time since launch in years
EMTG’s Integrator: Adaptive RK7(8)13M

- Two main components
  - Adaptive step sizing algorithm
  - 8th order embedded explicit Runge-Kutta step calculation method

- Local truncation error between 7th and 8th order solutions is used to adaptively size the integration step

- Each FBLT segment $h_N$ is broken into $\delta_i$ sub-steps

$$h_N = \sum_i \delta_i$$
Adaptive Step Size Computation

- After each sub-step, the local truncation error between the 7th and 8th order solutions is calculated ($\Delta e$)
- If $\Delta e > \Delta T$ then $S = 0.98$, the time step will be reduced and the sub-step is reintegrated

- else $S = 1.01$, and the time step length is increased, potentially saving computation time

- $q$ is the order of the relative error of the method, $q = 8$ for RK7(8)13M

- The integration proceeds until integration across the full FBLT time step has been completed to within the user-specified error tolerance

$$\delta_{n+1} = \delta_n \cdot S \cdot \left(\frac{\Delta T}{\Delta e}\right)^\frac{1}{q}$$

EMTG’s Integrator: Adaptive RK8(7)13M

1. Calculate the gradient and step at the left hand side of the RK sub-step

\[ k_1 = \delta_n \cdot f \left( t_n, \hat{X}_n \right) \]

2. At each of the 13 stages, calculate the state sample point and the gradient/step at that point…store the gradient information

\[ \hat{X}_{n(i)} = \hat{X}_n + \sum_{j=1}^{i-1} a_{ij} k_j \quad i > j \quad i, j = 1, 2, 3, \ldots, s \]

\[ k_i = \delta_n \cdot f \left( t_n + c_i \delta_n, \hat{X}_{n(i)} \right) \]

3. At the right hand side of the sub-step, compute the 7th and 8th order solutions

\[ \hat{X}_{n+1} = \hat{X}_n + \sum_{i=1}^{s} \hat{b}_i k_i \quad X_{n+1} = \hat{X}_n + \sum_{i=1}^{s} b_i k_i \quad i = 2, 3, \ldots, s \]

4. Update time and adjust the step size

\[ t_{n+1} = t_n + \delta_n \]
<table>
<thead>
<tr>
<th>$c_i$</th>
<th>$a_{ij}$</th>
<th>$b_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1/18</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1/12</td>
<td>1/4</td>
<td>0</td>
</tr>
<tr>
<td>1/8</td>
<td>0 1/32</td>
<td>0</td>
</tr>
<tr>
<td>5/16</td>
<td>0 5/16</td>
<td>0</td>
</tr>
<tr>
<td>3/8</td>
<td>3/8</td>
<td>0</td>
</tr>
<tr>
<td>5/8</td>
<td>7/16</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3/4</td>
<td>0</td>
</tr>
</tbody>
</table>

Dormand-Prince RK7(8)13M Butcher Tableau

<table>
<thead>
<tr>
<th>$c_i$</th>
<th>$a_{ij}$</th>
<th>$b_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1.4005451</td>
</tr>
<tr>
<td>1/18</td>
<td>1/18</td>
<td>1.3451932</td>
</tr>
<tr>
<td>1/12</td>
<td>1/12</td>
<td>335180664</td>
</tr>
<tr>
<td>1/8</td>
<td>1/8</td>
<td>355176623</td>
</tr>
<tr>
<td>5/16</td>
<td>5/16</td>
<td>0</td>
</tr>
<tr>
<td>3/8</td>
<td>3/8</td>
<td>0</td>
</tr>
<tr>
<td>5/4</td>
<td>5/4</td>
<td>0</td>
</tr>
<tr>
<td>3/2</td>
<td>3/2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

8th order

7th order

Match Point Time of Flight Derivatives

- If we had an analytic equations for propagating the trajectory, TOF derivatives would be fairly simple (to first order)

\[ \delta \mathbf{r}_m = \mathbf{v}_m \cdot \delta TOF \]

- Unfortunately, we have a numerical approximation, so computing TOF derivatives involves taking derivatives of the components of the integrator itself.

- Derivatives of both the adaptive step routine as well as the RK step calculation must be accumulated as the integration is performed.

\[
\begin{align*}
\frac{\partial \hat{X}_{n+1}}{\partial TOF} &= \frac{\partial \hat{X}_n}{\partial TOF} + \sum_{i=1}^{s} \hat{b}_i \frac{\partial k_i}{\partial TOF} \\
\frac{\partial k_i}{\partial TOF} &= \frac{\partial \delta_n}{\partial TOF} f + \delta_n \frac{\partial f}{\partial TOF} \\
\frac{\partial \tilde{r}_{cb-s/c}}{\partial TOF} &= \frac{\partial \tilde{r}_{cb-s/c}}{\partial \mathbf{r}_{cb-s/c}} \cdot \frac{\partial \mathbf{r}_{cb-s/c}}{\partial TOF} + \frac{\partial \tilde{r}_{cb-s/c}}{\partial \mathbf{r}_{cb-3b}} \cdot \frac{\partial \mathbf{r}_{cb-3b}}{\partial TOF} + \frac{\partial \tilde{r}_{cb-s/c}}{\partial m_{s/c}} \cdot \frac{\partial m_{s/c}}{\partial TOF} + \frac{\partial \tilde{r}_{cb-s/c}}{\partial \dot{T}} \cdot \frac{\partial \dot{T}}{\partial TOF}
\end{align*}
\]

\[ \dot{X} = f = \begin{bmatrix} \dot{\mathbf{r}} \\ \ddot{\mathbf{r}} \\ \dot{m} \end{bmatrix} \]
TOF Derivative Challenges

- State gradient TOF derivatives for the spacecraft in turn require sensitivity of 3rd body position to changes in TOF
- EMTG uses the SPICE kernels to calculate solar system body states
- Analytical derivatives are only possible if you can access SPICE internally
- SPICE uses Chebyshev polynomial interpolation for planetary positions
- Fortunately, for planets only, the velocity polynomial is the time derivative of the position polynomial (we have direct access to the position curve derivative)
- This is not true for planetary satellites, velocity curve is fit separately
  - Another ephemeris reader (FIRE, Arora and Russell, 2008) would solve this
- EMTG’s power solar electric hardware models are TOF dependent

\[
\frac{\partial \vec{r}_{cb-s/c}}{\partial \text{TOF}} = \frac{\partial \vec{r}_{cb-s/c}}{\partial \text{TOF}} \cdot \frac{\partial \vec{r}_{cb-s/c}}{\partial \text{TOF}} + \frac{\partial \vec{\dot{r}}_{cb-3b}}{\partial \text{TOF}} \cdot \frac{\partial \vec{r}_{cb-3b}}{\partial \text{TOF}} + \frac{\partial \vec{\dot{r}}_{cb-s/c}}{\partial m_{s/c}} \cdot \frac{\partial m_{s/c}}{\partial \text{TOF}} + \frac{\partial \vec{\dot{r}}_{cb-s/c}}{\partial \tau} \cdot \frac{\partial \tau}{\partial \text{TOF}}
\]
TOF Derivative Challenges

- State gradient TOF derivatives for the spacecraft in turn require sensitivity of 3\textsuperscript{rd} body position to changes in TOF
- EMTG uses the SPICE kernels to calculate solar system body states
- Analytical derivatives are only possible if you can access SPICE internally
- SPICE uses Chebyshev polynomial interpolation for planetary positions
- Fortunately, for planets only, the velocity polynomial is the time derivative of the position polynomial (we have direct access to the position curve derivative)
- This is not true for planetary satellites, velocity curve is fit separately
  - Another ephemeris reader (FIRE, Arora and Russell, 2008) would solve this
- EMTG’s power solar electric hardware models are TOF dependent

\[
\frac{\partial \mathbf{v}_{cb-s/c}}{\partial \text{TOF}} = \frac{\partial \mathbf{v}_{cb-s/c}}{\partial \mathbf{r}_{cb-s/c}} \cdot \frac{\partial \mathbf{r}_{cb-s/c}}{\partial \text{TOF}} + \frac{\partial \mathbf{v}_{cb-3b}}{\partial \mathbf{r}_{cb-3b}} \cdot \frac{\partial \mathbf{r}_{cb-3b}}{\partial \text{TOF}} + \frac{\partial \mathbf{v}_{cb-s/c}}{\partial \mathbf{m}_{s/c}} \cdot \frac{\partial \mathbf{m}_{s/c}}{\partial \text{TOF}} + \frac{\partial \mathbf{v}_{cb-s/c}}{\partial T} \cdot \frac{\partial T}{\partial \text{TOF}}
\]

\[
\frac{\partial \mathbf{v}_{cb-s/c}}{\partial \mathbf{r}_{cb-3b}} = -\mu_{3b} \left[ \frac{3\mathbf{r}_{3b-s/c} \mathbf{r}_{3b-s/c}^T}{r_{3b-s/c}^5} - \frac{I_{3x3}}{r_{3b-s/c}^3} \right]
\]
TOF Derivative Challenges

- State gradient TOF derivatives for the spacecraft in turn require sensitivity of 3rd body position to changes in TOF
- EMTG uses the SPICE kernels to calculate solar system body states
- Analytical derivatives are only possible if you can access SPICE internally
- SPICE uses Chebyshev polynomial interpolation for planetary positions
- Fortunately, for planets only, the velocity polynomial is the time derivative of the position polynomial (we have direct access to the position curve derivative)
- This is not true for planetary satellites, velocity curve is fit separately
  - Another ephemeris reader (FIRE, Arora and Russell, 2008) would solve this
- EMTG’s power solar electric hardware models are TOF dependent

\[
\frac{\partial \mathbf{r}_{cb-s/c}}{\partial \text{TOF}} = \frac{\partial \mathbf{r}_{cb-s/c}}{\partial \text{TOF}} \cdot \frac{\partial \mathbf{r}_{cb-s/c}}{\partial \text{TOF}} + \frac{\partial \mathbf{r}_{cb-3b}}{\partial \text{TOF}} + \frac{\partial \mathbf{r}_{cb-s/c}}{\partial m_{s/c}} \cdot \frac{\partial m_{s/c}}{\partial \text{TOF}} + \frac{\partial \mathbf{r}_{cb-s/c}}{\partial T} \cdot \frac{\partial T}{\partial \text{TOF}}
\]
TOF Derivative Challenges

- State gradient TOF derivatives for the spacecraft in turn require sensitivity of 3rd body position to changes in TOF.
- EMTG uses the SPICE kernels to calculate solar system body states.
- Analytical derivatives are only possible if you can access SPICE internally.
- SPICE uses Chebyshev polynomial interpolation for planetary positions.
- Fortunately, for planets only, the velocity polynomial is the time derivative of the position polynomial (we have direct access to the position curve derivative).
- This is not true for planetary satellites, velocity curve is fit separately.
  - Another ephemeris reader (FIRE, Arora and Russell, 2008) would solve this.
- EMTG’s power solar electric hardware models are TOF dependent.

\[
\frac{\partial \mathbf{r}_{\text{cb}-s/c}}{\partial \text{TOF}} = \frac{\partial \mathbf{r}_{\text{cb}-s/c}}{\partial \text{TOF}} \cdot \frac{\partial \mathbf{r}_{\text{cb}-s/c}}{\partial \text{TOF}} + \frac{\partial \mathbf{r}_{\text{cb}-s/c}}{\partial \text{TOF}} \cdot \frac{\partial \mathbf{r}_{\text{cb}-3b}}{\partial \text{TOF}} + \frac{\partial \mathbf{r}_{\text{cb}-s/c}}{\partial m_{s/c}} \cdot \frac{\partial m_{s/c}}{\partial \text{TOF}} + \frac{\partial \mathbf{r}_{\text{cb}-s/c}}{\partial T} \cdot \frac{\partial T}{\partial \text{TOF}}
\]
Example: Solar Electric Earth to Jupiter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial mass</td>
<td>3000 kg</td>
</tr>
<tr>
<td>Earliest allowed launch date</td>
<td>January 1st, 2015</td>
</tr>
<tr>
<td>Latest allowed launch date</td>
<td>unbounded</td>
</tr>
<tr>
<td>Initial $v_{\infty}$</td>
<td>up to 6.97 km/s</td>
</tr>
<tr>
<td>Maximum flight time</td>
<td>unbounded</td>
</tr>
<tr>
<td>Arrival type</td>
<td>low-thrust rendezvous</td>
</tr>
<tr>
<td>Thruster</td>
<td>NEXT - high thrust variant</td>
</tr>
<tr>
<td>Thrust coefficients</td>
<td>$e_T = 0.09591 d_T = -1.98537 e_T = 11.47980 b_T = 15.06977 a_T = 14.51552$</td>
</tr>
<tr>
<td>Mass flow rate coefficients</td>
<td>$e_F = 0.01492 d_F = -0.27539 c_F = 1.60966 b_F = -2.53056 a_F = 3.22089$</td>
</tr>
<tr>
<td>Solar power coefficients</td>
<td>$\gamma_0 = 1.32077 \gamma_1 = -0.10848 \gamma_2 = -0.11665 \gamma_3 = 0.10843 \gamma_4 = -0.01279$</td>
</tr>
<tr>
<td>Solver parameters</td>
<td></td>
</tr>
<tr>
<td>Flyby sequence</td>
<td>Earth-Jupiter direct (E-J)</td>
</tr>
<tr>
<td>Number of time-steps</td>
<td>40</td>
</tr>
<tr>
<td>Ephemeris</td>
<td>SPICE</td>
</tr>
<tr>
<td>SNOPT feasibility tolerance</td>
<td>1.0e-5</td>
</tr>
<tr>
<td>Objective function</td>
<td>maximize final mass</td>
</tr>
<tr>
<td>Number of NLP parameters</td>
<td>126</td>
</tr>
<tr>
<td>Number of constraints (including objective function)</td>
<td>48</td>
</tr>
<tr>
<td>Dense Jacobian entries</td>
<td>1002</td>
</tr>
</tbody>
</table>

**Table 2:** Earth to Jupiter low-thrust direct transfer problem assumptions
Example: Solar Electric Earth to Jupiter

launch Earth 10/27/2016
C3 = 19.859 \, \text{km}^2 /\text{s}^2
DLA = 26.8\,^\circ
m = 3000 \, \text{kg}

LT rendezvous Jupiter 7/16/2029
m = 2118 \, \text{kg}
Example: Solar Electric Earth to Jupiter
Example: Solar Electric Earth to Jupiter
Example: Solar Electric Earth to Jupiter

- SNOPT was allowed to execute for 100 major iterations
- STM-based Jacobian run converged after 80 majors
- Finite differenced Jacobian run hit the 100 major iteration limit
- Both are equally effective at solving the problem from an MGALT initial guess
- Numerically computed STMs afford 13-15 times execution speed increase

<table>
<thead>
<tr>
<th>Metric</th>
<th>STM computed Jacobian</th>
<th>Finite Differenced Jacobian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final mass delivered to Jupiter</td>
<td>2117.68 kg</td>
<td>2117.53 kg</td>
</tr>
<tr>
<td>SNOPT execution time</td>
<td>44.04 s</td>
<td>672.91 s</td>
</tr>
</tbody>
</table>

Table 3: STM vs. finite differenced Jacobian SNOPT performance metrics
Next Steps

- Low-thrust preliminary design cadence has been drastically increased
- EMTG-General Mission Analysis Tool (GMAT) work flow has been improved
- Although GMAT can converge from an MGALT or FBLT initial guess, for complex problems sometimes even an FBLT cannot converge from MGALT
- For cases when MGALT is insufficient, it is crucial that initial guesses for GMAT be generated as fast as possible using FBLT
- Launch vehicle, power and thruster models are currently being integrated with GMAT
- It is recommended that the STM described in this presentation be incorporated into GMAT’s solvers as it can handle arbitrarily complex dynamics (i.e. SRP, gravitational harmonics could be added relatively easily)
- EMTG and GMAT native integrators are also quite similar
Conclusions

- The state transition matrix increases the efficiency of the finite-burn low-thrust transcription and allows for the tempo at which low-thrust preliminary design is performed to be increased, saving analysts’ time.
- It importantly includes accurate hardware models that can have a major influence on mission feasibility even at the preliminary design stage.
- This work enables FBLT to be used in the framework of a global optimizer for many problems.
- With EMTG’s performance increases serving as an example, the logical next step would be to repeat the same process inside the GMAT high-fidelity tool.
- This work has extensions in other areas of interest:
  - Spacecraft guidance (method of adjoints and the guidance matrix)
Thank you!

This work is supported by Rich Burns and was completed under contract with NASA GSFC NNX14AD27G

EMTG is available open-source at:
https://sourceforge.net/projects/emtg/