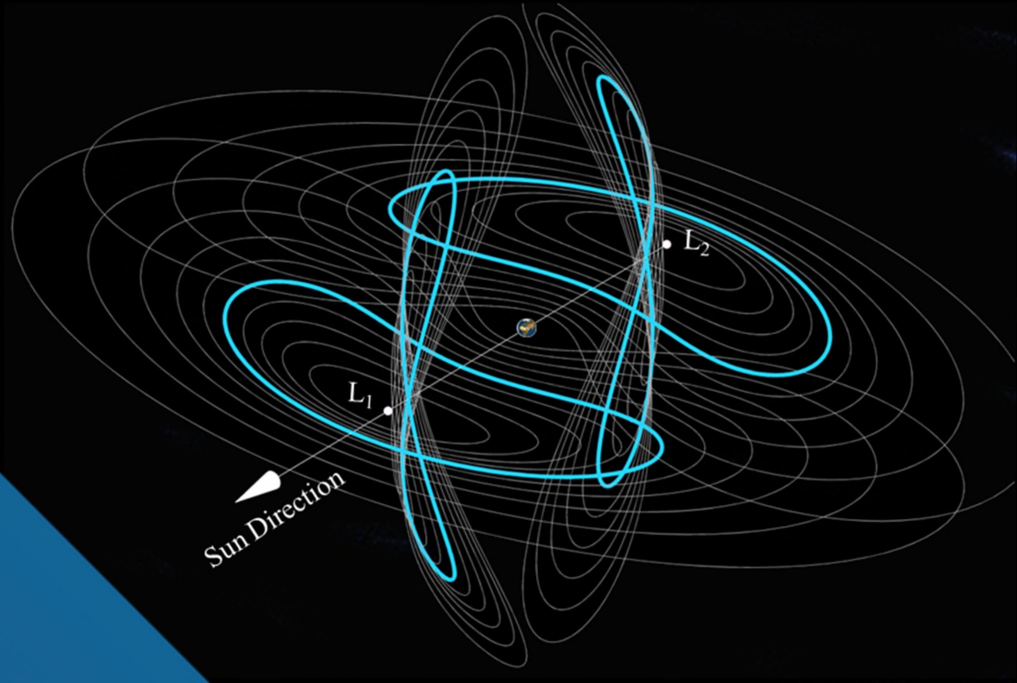
Mathematics and Programming for Astrodynamics and Trajectory Design

MSc in Astronautics and Space Engineering

Joan Pau Sánchez

Professor at ISAE-SUPAERO (Toulouse, France)

Visiting Lecturer at Cranfield University



Course Objectives

- This module is intended to provide you with an overview of state-of-the-art in applied mathematics and techniques for astrodynamics and trajectory design. The objective is to provide a fundamental understanding of the trending topics in mission analysis. The focus of the subject is not so much in deep understanding, but in providing breadth of knowledge and experiences.

Intended Learning Outcomes

On completion of this module you should :

- Be able to write reliable code to solve realistic mission analysis scenarios.
- Be able to apply a range of applied mathematical techniques to solve trajectory design problems and be able to independently expand on appropriate tools and know-how when necessary.
- Identify most common non-Keplerian orbit types and explain their applications.
- Reflect on the current challenges in trajectory design for both Earth observation

Bibliography

Some good books ordered by increasing depth and breath :

- *Orbital Mechanics,* V.A Chobotov, AIAA Education Series
- *Analytical Mechanics of Space Systems*, H.Shcaub, J.L.Junkins, AIAA Education Series
- *An Introduction to the Mathematics and Methods of Astrodynamics*, R.H.Battin, AIAA Educationan Series
- *Fundamentals of Astrodynamics and Applications*, D.A.Vallado, Space Technology Library

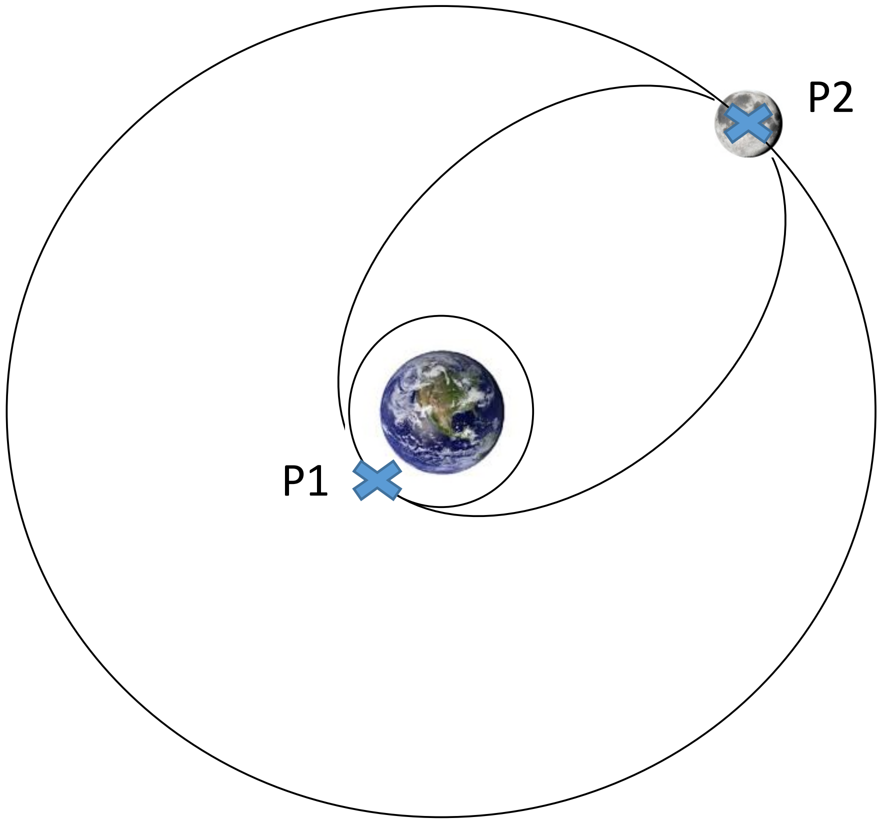
Session 1 - Content

A Lambert Arc DIY

- Review Hohmann
- Introduction of the Two-body Orbital Boundary-value Problem.
  (a.k.a. Lambert Arc)
- The Minimum Energy Orbit
- Lagrange Coefficients Solution to Orbital Motion
- A general algorithm to solve the Lambert Arc

Two-body Orbital Boundary-value Problem

- Two-body Orbital = Keplerian Motion



Hohmann transfer

Vis-viva Equation

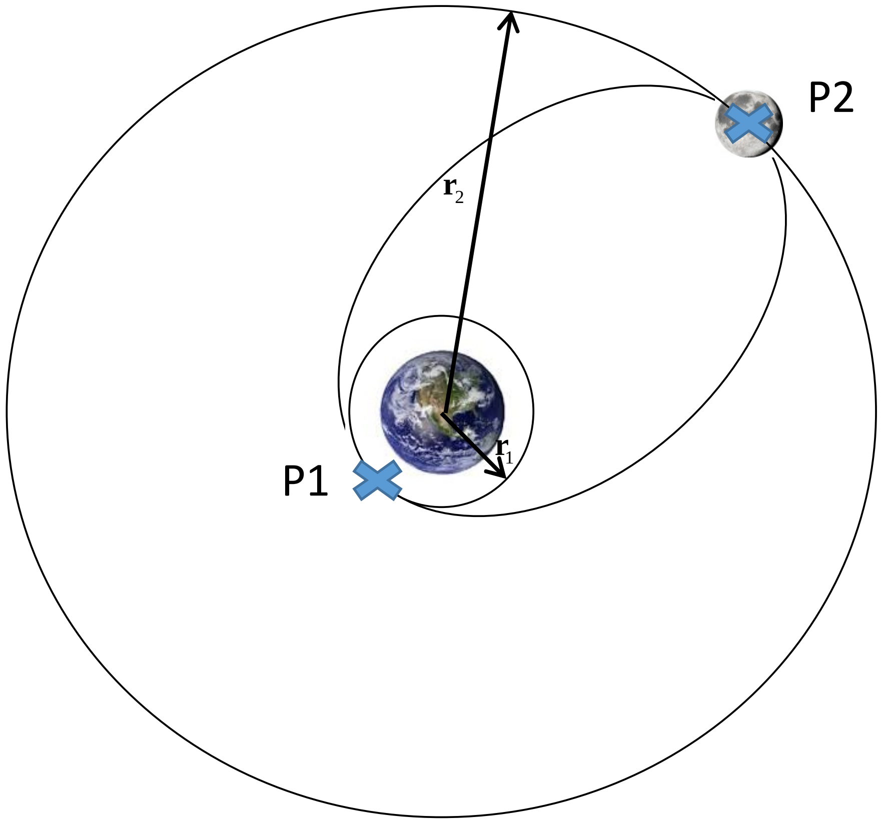
$$\frac{v^2}{2} - \frac{\mu}{r} = -\frac{\mu}{2a}$$

Vis-viva solved for *v*

$$v = \sqrt{\mu\left(\frac{2}{r} - \frac{1}{a}\right)}$$

Two-body Orbital Boundary-value Problem

- Two-body Orbital = Keplerian Motion



Hohmann transfer

$$v_1 = \sqrt{\frac{\mu_E}{r_1}}$$

$$v_2 = \sqrt{\mu_E\left(\frac{2}{r_1} - \frac{2}{r_1 + r_2}\right)}$$

$$\Delta v_s = v_2 - v_1$$

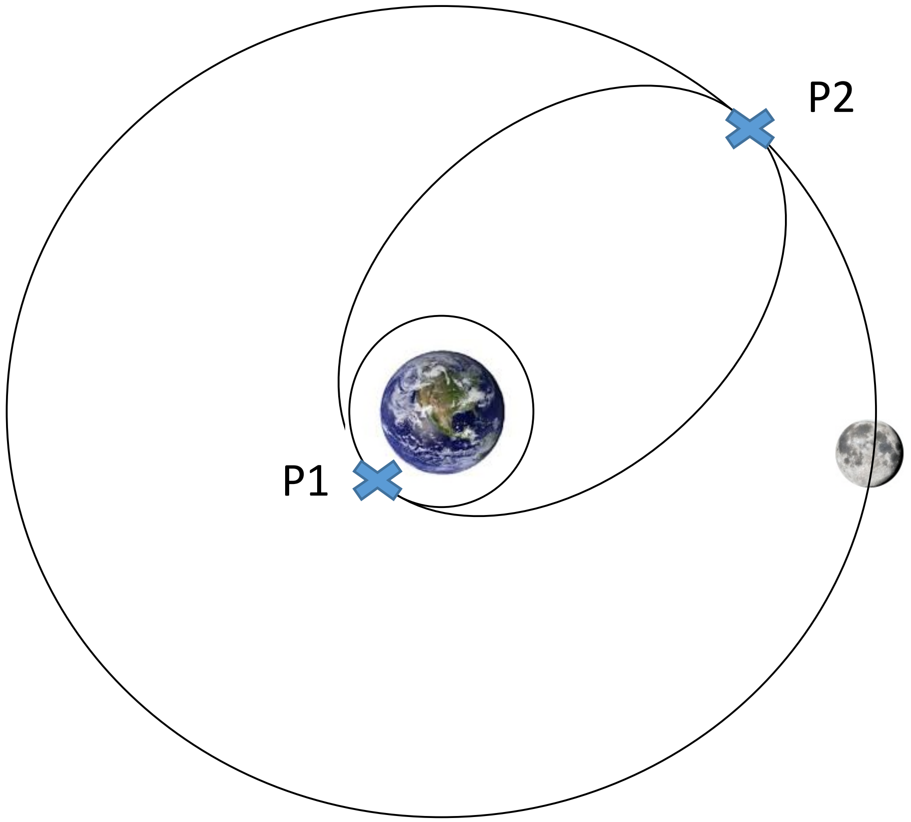
$$v_3 = \sqrt{\mu_E\left(\frac{2}{r_2} - \frac{2}{r_1 + r_2}\right)}$$

$$v_4 = \sqrt{\frac{\mu_E}{r_2}}$$

$$\Delta v_f = v_4 - v_3$$

Two-body Orbital Boundary-value Problem

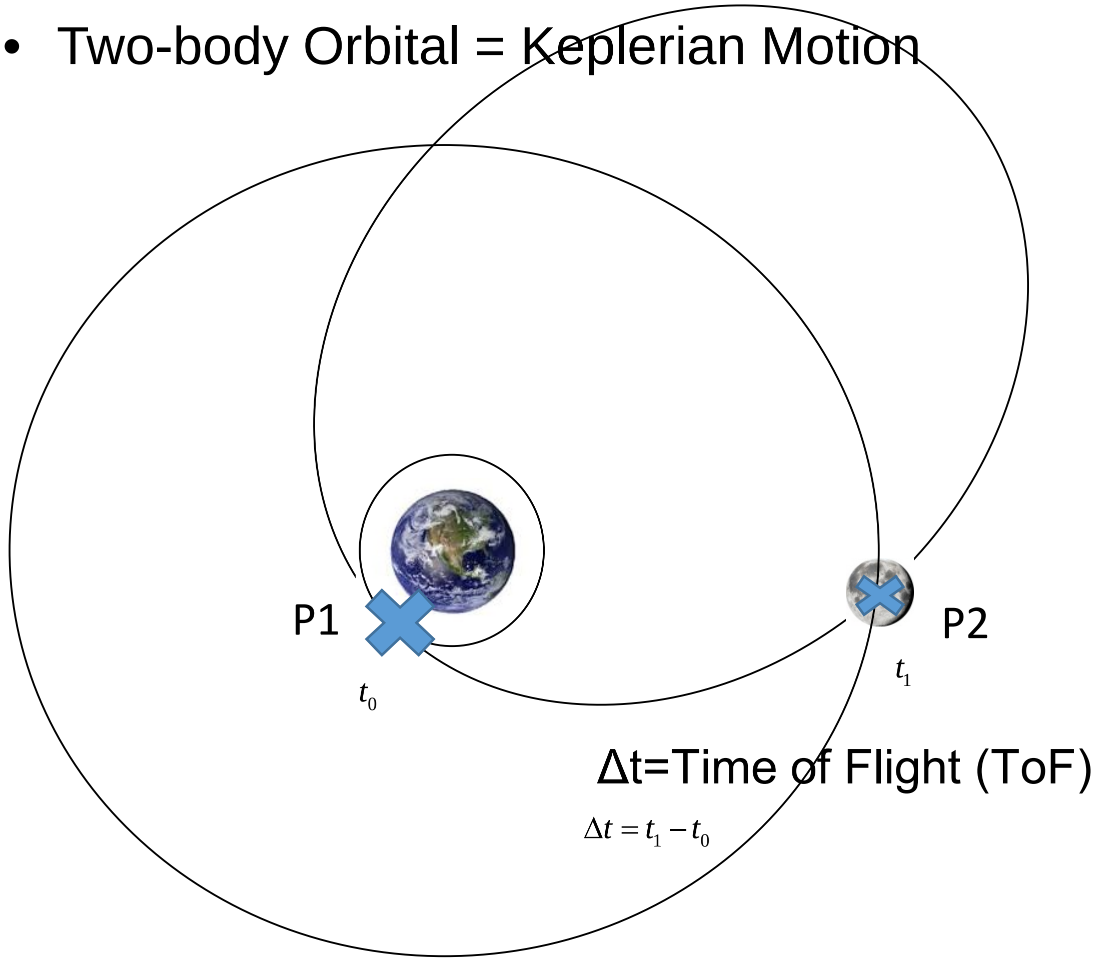
- Two-body Orbital = Keplerian Motion



What if the Moon is not there?

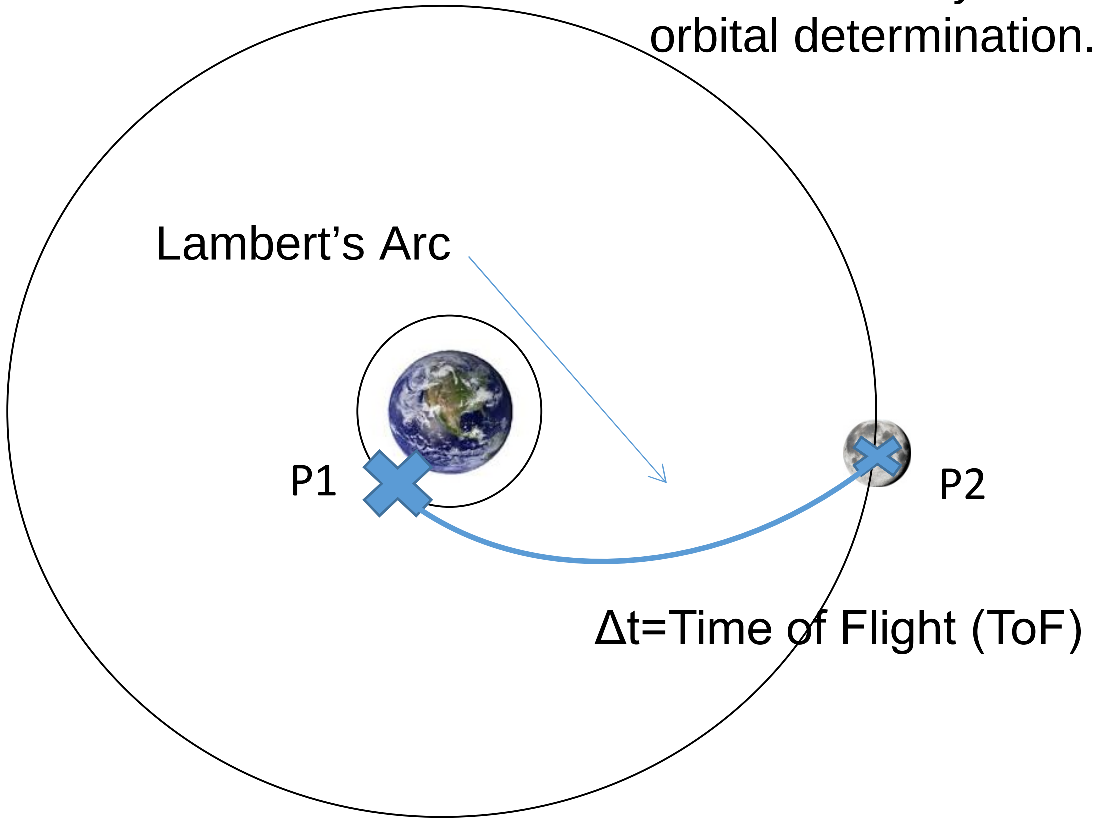
Two-body Orbital Boundary-value Problem

- Two-body Orbital = Keplerian Motion



Two-body Orbital Boundary-value Problem

- Two position vectors & time = Lambert's Problem
  - ✓ Solved not only for trajectory design but also for orbital determination.



Lambert’s Problem: Historical Perspective

- 1743: Leonhard Euler defines the problem and finds an infinite series solution for a parabolic case.
- 1761-1771: Johann H. Lambert geometric solutions.
- 1857: Gauss’s solution provides geometrical insight.
- Universal variables:
  - ✓Lancaster & Blanch (1969), Gooding (1988,1990), Izzo (2015). Bate (1971), Vallado (1997), Luo (2011), Thomson (1929), Arora (2013). Battin-Vaughan (1984), Loechler (1988), Shen (2004), MacLellan (2005).
- Semi-major axis:
  - ✓ Thorne (1995,2014), Prussing (2000), Chen (2013), Wailliez (2014).
- Semi-latus rectum (*p*-iteration):
  - ✓ Herrick-Liu (1959), Boltz (1984), Bate (1971).
- Eccentricity vector:
  - ✓ Avanzini (2008), He (2010), Zhang (2010, 2011), Wen (2014).

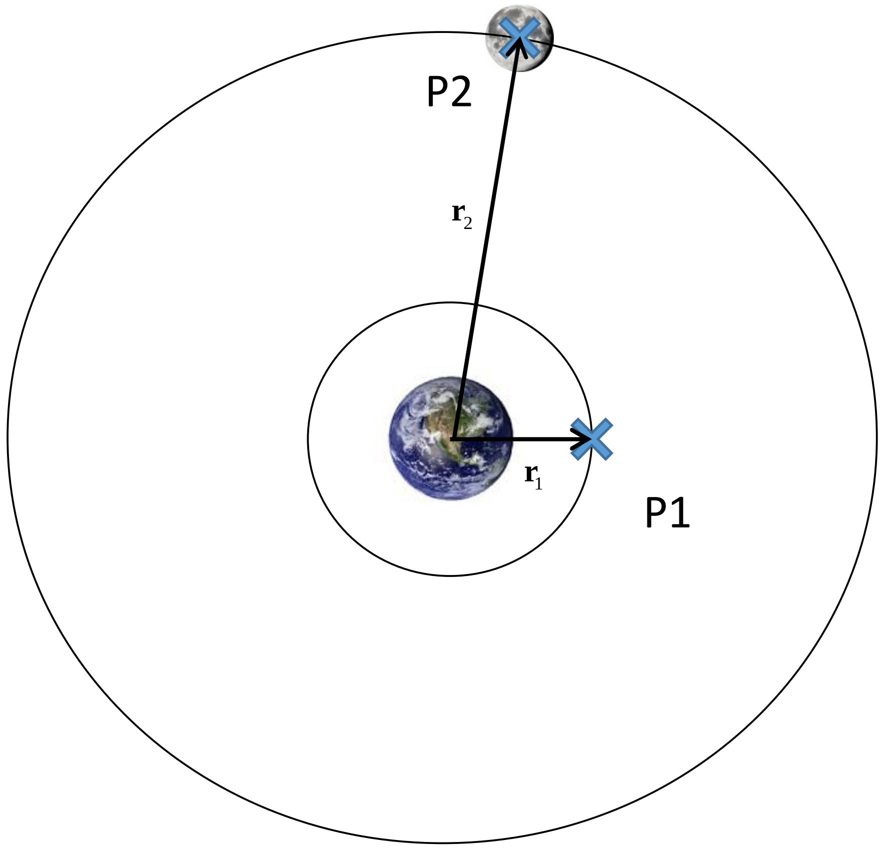
Lambert’s Problem

- We will attempt to solve it using a general approach that may be used to solve other targeting problems in higher fidelity dynamics (i.e. *n*-body problem or with other perturbations).
  - ✓ *Pros*: The techniques used are applicable to many other problems and you may come across them in future occasions.
  - ✓ *Cons*: The final algorithm is not very robust and its convergence tends to fail in many occasions.
- Algorithm as described in: *Analytical Mechanics of Space Systems*, H.Shcaub, J.L.Junkins, AIAA Education Series.

Lambert's Problem – Be patient!

- 1. Minimum Energy Transfer
  - ✓ MinETransfer $\left(\mathbf{r}_1, \mathbf{r}_2 \Rightarrow a_{\min}, e_{\min}, \Delta t_{\min}\right)$
  - ✓ MinETransfer $\left(\mathbf{r}_1, \mathbf{r}_2, t_m \Rightarrow \dot{\mathbf{r}}_1\right)$
- 2. Simple propagation solution - F & G Solutions for elliptical orbits:
  - ✓*FGKepler* $\left(\mathbf{r}_0, \dot{\mathbf{r}}_0, \Delta\theta \Rightarrow \mathbf{r}_f\right)$
  - ✓*FGKepler_dt* $\left(\mathbf{r}_0, \dot{\mathbf{r}}_0, \Delta t \Rightarrow \mathbf{r}_f\right)$
- *3. State Transition Matrix for Two-body-problem*
  - ✓*STM_Lambert*: $\left(\mathbf{r}_0, \dot{\mathbf{r}}_0, \Delta t \Rightarrow \frac{\partial \mathbf{r}_f}{\partial \dot{\mathbf{r}}_0}\right)$
- *4. Implement a Differential Corrector*
- *5. Implement a Continuation Method*

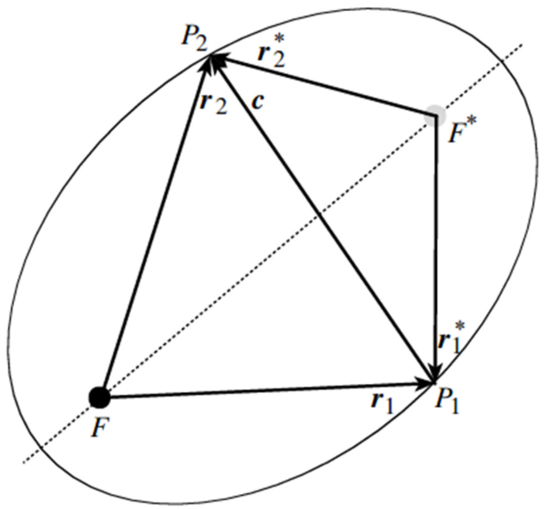
Minimum Energy Transfer



- Connect P1 and P2.

Minimum Energy Transfer

(Schaub & Junkins, 2002)



1. Download and install Socrative Student App or connect into website.
2. Connect into Room: GPQK5UNSK
3. Wait for question to appear, put your name in and answer.

Which of the following is correct?

A)
$$r_1 + r_1^* = a$$
$$r_2 + r_2^* = a$$

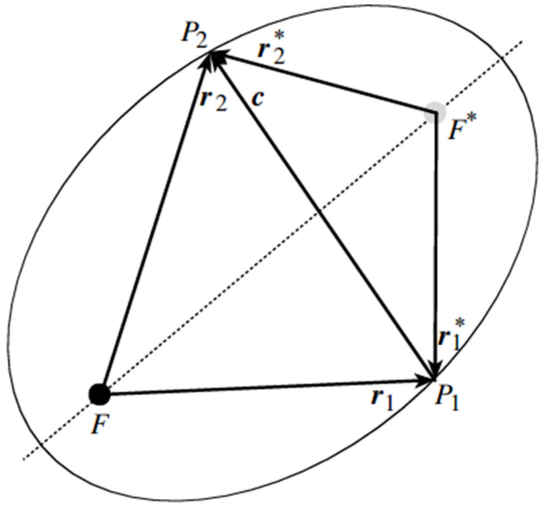
B)
$$r_1 + r_1^* = c$$
$$r_2 + r_2^* = c$$

C)
$$r_1 + r_1^* = 2a$$
$$r_2 + r_2^* = 2a$$

D) $r_1 + r_1^* = r_2 + r_2^*$

Minimum Energy Transfer

(Schaub & Junkins, 2002)



- Connect P1 and P2.

$$r_1 + r_1^* = 2a$$

$$r_2 + r_2^* = 2a$$

$$\underbrace{r_1 + r_2}_{\textbf{Fixed}} + r_1^* + r_2^* = 4a$$

- 3 unknown parameters: r_1^*, r_2^*, a

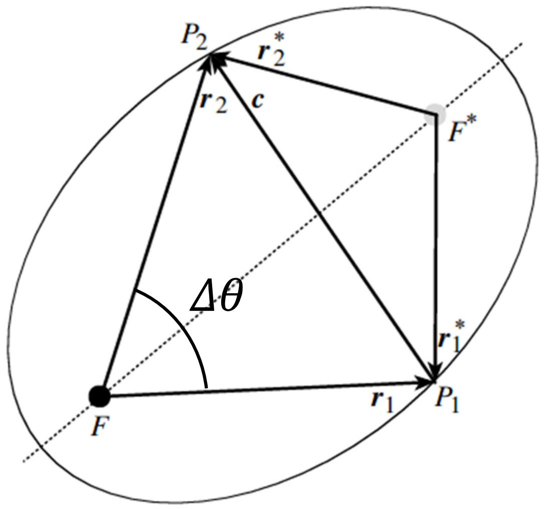
- Note the following inequality constraint: $c \le r_1^* + r_2^*$

$$\frac{v^2}{2} - \frac{\mu}{r} = \varepsilon = -\frac{\mu}{2a}$$

- Hence, the smaller *a* the smaller the energy ε.

Minimum Energy Transfer

(Schaub & Junkins, 2002)



- Hence, minimizing ε:

$$\frac{v^2}{2} - \frac{\mu}{r} = \varepsilon = -\frac{\mu}{2a}$$

$$\underbrace{r_1 + r_2}_{\textbf{Fixed}} + r_1^* + r_2^* = 4a$$

$$r_1^* + r_2^* = c$$

$$a_{\min} = \frac{1}{4}\left(r_1 + r_2 + c\right)$$

$$c = \left|\mathbf{r}_2 - \mathbf{r}_1\right| = \sqrt{r_2^2 + r_2^1 - 2r_1r_2\cos\left(\Delta\theta\right)}$$

Minimum Energy Transfer: Algorithm MinETransfer

Algorithm *MinETransfer* : $\left(\mathbf{r}_1, \mathbf{r}_2 \Rightarrow a_{\min}, e_{\min}, \Delta t_{\min}\right)$

Reference Books for all the algorithms I will present:

- *Orbital Mechanics,* V.A Chobotov, AIAA Education Series
- *Analytical Mechanics of Space Systems*, H.Shcaub, J.L.Junkins, AIAA Education Series
- *An Introduction to the Mathematics and Methods of Astrodynamics*, R.H.Battin, AIAA Educationan Series
- *Fundamentals of Astrodynamics and Applications*, D.A.Vallado, Space Technology Library

Minimum Energy Transfer: Algorithm MinETransfer

Algorithm *MinETransfer* : $\left(\mathbf{r}_1, \mathbf{r}_2, t_m \Rightarrow a_{\min}, e_{\min}, \Delta t_{\min}\right)$

$$c = \left|\mathbf{r}_2 - \mathbf{r}_1\right|$$

$$r_1 = \left|\mathbf{r}_1\right| \qquad r_2 = \left|\mathbf{r}_2\right|$$

$$a_{\min} = \frac{1}{4}\left(r_1 + r_2 + c\right)$$

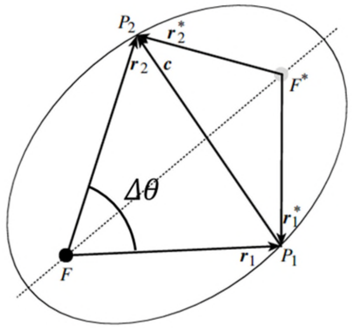
$$\cos\Delta\theta = \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2}$$

$$p_{\min} = \frac{r_1 r_2}{c}\left(1 - \cos\Delta\theta\right)$$

$$e_{\min} = \sqrt{1 - \frac{p_{\min}}{a_{\min}}}$$

$$\beta_e = 2\sin^{-1}\left(\sqrt{\frac{2a_{\min} - c}{2a_{\min}}}\right)$$

$$\Delta t_{\min} = \sqrt{\frac{a_{\min}^3}{\mu}}\left[\pi - t_m \cdot \left(\beta_e - \sin\beta_e\right)\right]$$



Where $t_m(+1)$ refers at Δt for the short path transfers and $t_m(-1)$ for the long path transfer.

Minimum Energy Transfer

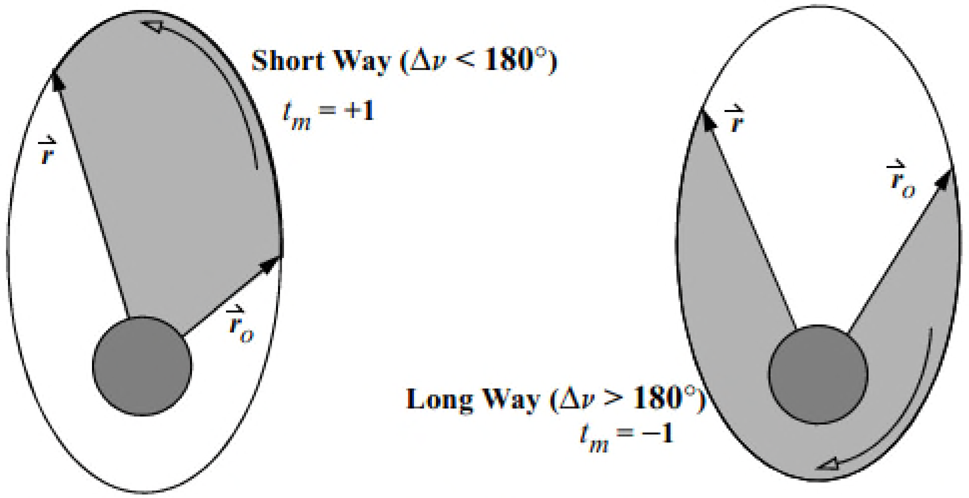


Figure 7-8. **Transfer Methods, t_m, for the Lambert Problem.** Traveling between the two specified points can take the long way or the short way. For the long way, the change in true anomaly exceeds 180°.

Courtesy (Vallado, 2013)

Exercise 1

ExoMars Trace Gas Orbiter departed Earth on 14/03/2016, and arrived at Mars on 15/10/2016. What is the minimum energy orbit that would link the position of Earth and Mars those two days? What is the time of flight of the minimum energy orbit? Was this the trajectory followed by ExoMars TGO?

Hints:

Use functions *EphSS_car* to compute the position of Earth and Mars at those dates (index Earth 3, index Mars 4).

When you are done:

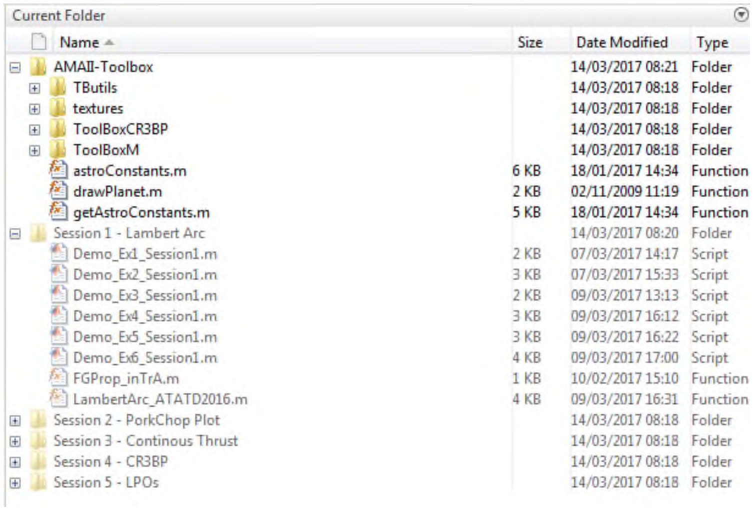
1. Connect into Room: GPQK5UNSK
2. Wait for question to appear, put your name in and answer.

Result Guide: $\Delta t_{\min}^{ShortPath} = 231.8 \text{ days}$

Session 1 MATLAB Guide

The following bullet points provide some basic hints on how completing the tasks of this course:

- Complete all the exercises by creating one script for each task. This way you will be able to revise easily your calculations, as well as comparing with the exercise solutions.
- Create a sensible working directory.
- Note that AMAII-Toolbox is in the path of the working environment.



Current Folder

Name	Size	Date Modified	Type
AMAII-Toolbox		14/03/2017 08:21	Folder
TButils		14/03/2017 08:18	Folder
textures		14/03/2017 08:18	Folder
ToolBoxCR3BP		14/03/2017 08:18	Folder
ToolBoxM		14/03/2017 08:18	Folder
astroConstants.m	6 KB	18/01/2017 14:34	Function
drawPlanet.m	2 KB	02/11/2009 11:19	Function
getAstroConstants.m	5 KB	18/01/2017 14:34	Function
Session 1 - Lambert Arc		14/03/2017 08:20	Folder
Demo_Ex1_Session1.m	2 KB	07/03/2017 14:17	Script
Demo_Ex2_Session1.m	3 KB	07/03/2017 15:33	Script
Demo_Ex3_Session1.m	2 KB	09/03/2017 13:13	Script
Demo_Ex4_Session1.m	3 KB	09/03/2017 16:12	Script
Demo_Ex5_Session1.m	3 KB	09/03/2017 16:22	Script
Demo_Ex6_Session1.m	4 KB	09/03/2017 17:00	Script
FGProp_inTrA.m	1 KB	10/02/2017 15:10	Function
LambertArc_ATATD2016.m	4 KB	09/03/2017 16:31	Function
Session 2 - PorkChop Plot		14/03/2017 08:18	Folder
Session 3 - Continous Thrust		14/03/2017 08:18	Folder
Session 4 - CR3BP		14/03/2017 08:18	Folder
Session 5 - LPOs		14/03/2017 08:18	Folder

FOLLOW VERY CLOSELY THE WORK GUIDE IN CANVAS

Download Session 1 MATLAB Guide (PART A) from BB

Exercise 1

ExoMars Trace Gas Orbiter departed Earth on 14/03/2016, and arrived at Mars on 15/10/2016. What is the minimum energy orbit that would link the position of Earth and Mars those two days? What is the time of flight of the minimum energy orbit? Was this the trajectory followed by ExoMars TGO?

Hints:

Use functions *EphSS_car* to compute the position of Earth and Mars at those dates (index Earth 3, index Mars 4).

When completed, connect to Socrative Room GPQK5UNSK and indicate it so in the relevant question

Result Guide: $\Delta t_{\min}^{ShortPath} = 231.8 \text{ days}$

Initial Value Problem and Position and Velocity as a Function of Time

Initial Value Problem:

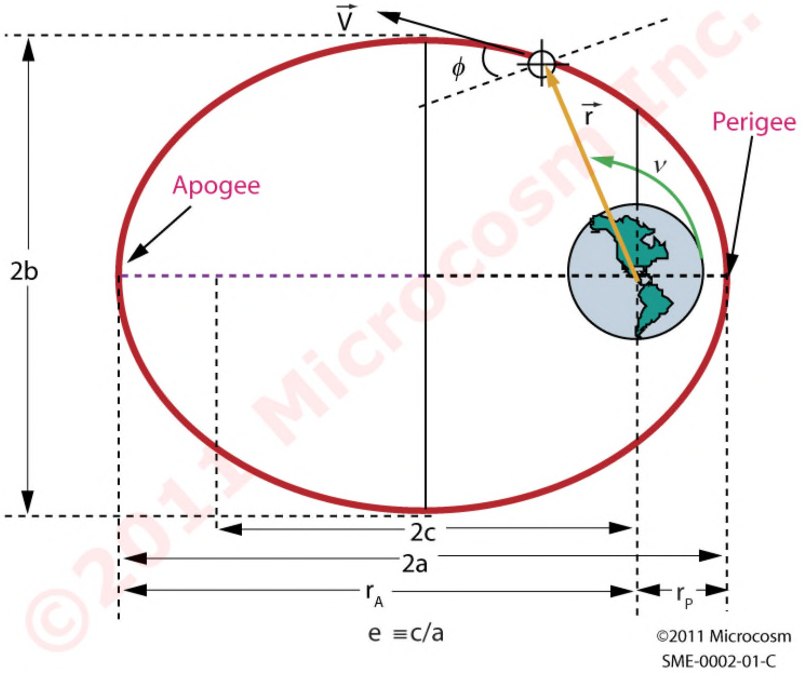
We may typically know the position and velocity vectors of a spacecraft at a given time (e.g. $\boldsymbol{r}_1$ and $\boldsymbol{v}_1$) and need to know the spacecraft position and velocity some time later (e.g. $\boldsymbol{r}_2$ and $\boldsymbol{v}_2$).

Position and Velocity Problem:

$$\boxed{\left(\mathbf{r}(t), \mathbf{v}(t)\right)}$$

Kepler's Equation

$$E - e\sin E = nt \qquad nt = M$$



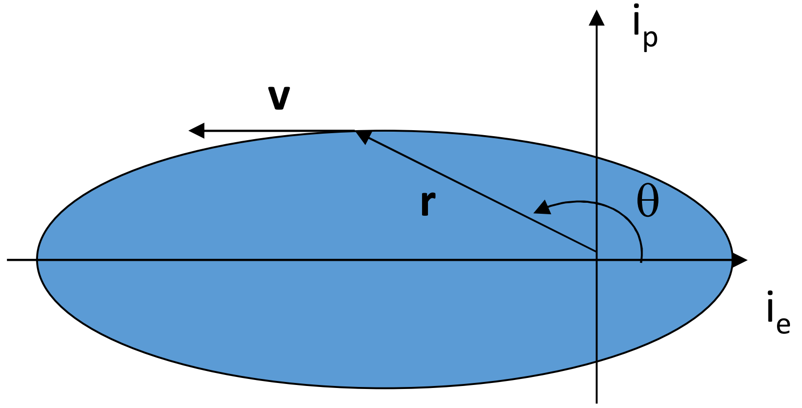
Initial Value Problem:
Lagrange Coefficients Solution to Orbital Motion

The general problem is to find the position and velocity vectors $\boldsymbol{r_2}$ and $\boldsymbol{v_2}$ at a given time $t_1+\Delta t$ once the position and velocity vectors $\boldsymbol{r_1}$ and $\boldsymbol{v_1}$ at t_1 are known.

Note that any position vector $\boldsymbol{r}$ and the velocity vector $\boldsymbol{v}$ at a given true anomaly θ can be expressed in terms of orbital plane coordinates as follows:

$$\mathbf{r} = r\cos\theta\,\mathbf{i}_e + r\sin\theta\,\mathbf{i}_p$$

$$\mathbf{v} = -\frac{\mu}{h}\sin\theta\,\mathbf{i}_e + \frac{\mu}{h}(e+\cos\theta)\,\mathbf{i}_p$$

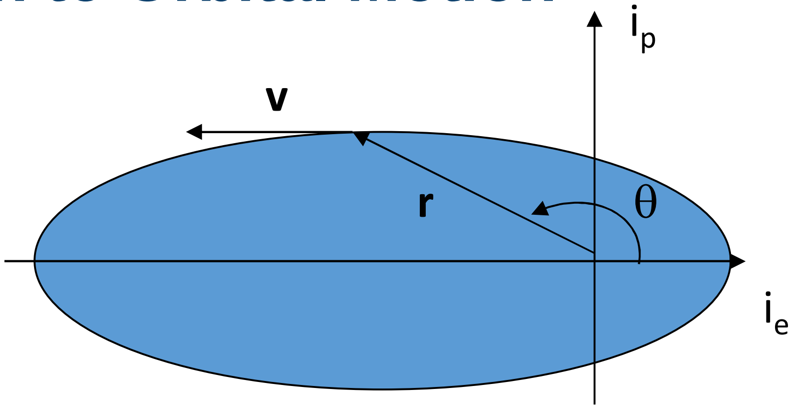


Where h is the angular momentum

Lagrange Coefficients Solution to Orbital Motion

$$\mathbf{r} = r\cos\theta\,\mathbf{i}_e + r\sin\theta\,\mathbf{i}_p$$

$$\mathbf{v} = -\frac{\mu}{h}\sin\theta\,\mathbf{i}_e + \frac{\mu}{h}(e+\cos\theta)\mathbf{i}_p$$



$$r = \frac{h^2}{\mu\left(1+e\cos\theta\right)}$$

$$\frac{d\mathbf{r}}{dt} = \frac{d}{dt}\left(r\cos\theta\,\mathbf{i}_e + r\sin\theta\,\mathbf{i}_p\right)$$

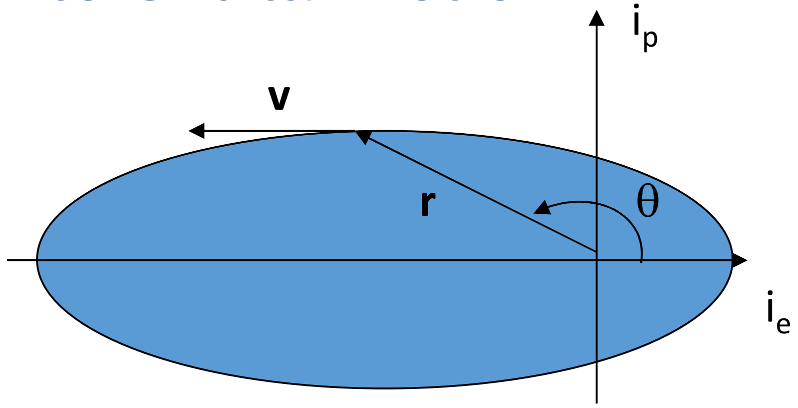
$$h = r^2\dot{\theta}$$

$$h^2 = \mu p$$

Lagrange Coefficients Solution to Orbital Motion

$$\mathbf{r} = r\cos\theta\mathbf{i}_e + r\sin\theta\mathbf{i}_p$$

$$\mathbf{v} = -\frac{\mu}{h}\sin\theta\mathbf{i}_e + \frac{\mu}{h}(e+\cos\theta)\mathbf{i}_p$$



$$\mathbf{r}_1 = r_1\cos\theta_1\mathbf{i}_e + r_1\sin\theta_1\mathbf{i}_p$$

$$\mathbf{v}_1 = -\frac{\mu}{h}\sin\theta_1\mathbf{i}_e + \frac{\mu}{h}(e+\cos\theta_1)\mathbf{i}_p$$

Lagrange Coefficients Solution to Orbital Motion

These equations are valid at the initial point $\boldsymbol{r_1}$ and $\boldsymbol{v_1}$. We can now invert the relationship and obtain the coordinate unit vectors as a function of the initial position and velocity vectors:

$$\mathbf{i}_e = \frac{\mu}{h^2}(e + \cos\theta_1)\mathbf{r}_1 - \frac{r_1}{h}\sin\theta_1 \mathbf{v}_1$$

$$\mathbf{i}_p = \frac{\mu}{h^2}\sin\theta_1 \mathbf{r}_1 + \frac{r_1}{h}\cos\theta_1 \mathbf{v}_1$$

Now substituting the above relations into the general form equation from slide 25, we can obtain:

$$\mathbf{r}_2 = F\mathbf{r}_1 + G\mathbf{v}_1$$

$$\mathbf{v}_2 = \dot{F}\mathbf{r}_1 + \dot{G}\mathbf{v}_1$$

Where F, G, Fdot and Gdot are known as Lagrange coefficients.

Lagrange Coefficients Solution to Orbital Motion

$$\mathbf{i}_e = \frac{\mu}{h^2}(e + \cos\theta_1)\mathbf{r}_1 - \frac{r_1}{h}\sin\theta_1\mathbf{v}_1$$

$$\mathbf{i}_p = \frac{\mu}{h^2}\sin\theta_1\mathbf{r}_1 + \frac{r_1}{h}\cos\theta_1\mathbf{v}_1$$

$$\mathbf{r}_2 = r_2\cos\theta_2\mathbf{i}_e + r_2\sin\theta_2\mathbf{i}_p$$

$$\mathbf{v}_2 = -\frac{\mu}{h}\sin\theta_2\mathbf{i}_e + \frac{\mu}{h}(e + \cos\theta_2)\mathbf{i}_p$$

$$\mathbf{r}_2 = r_2\cos\theta_2\left(\frac{\mu}{h^2}(e + \cos\theta_1)\mathbf{r}_1 - \frac{r_1}{h}\sin\theta_1\mathbf{v}_1\right) + r_2\sin\theta_2\left(\frac{\mu}{h^2}\sin\theta_1\mathbf{r}_1 + \frac{r_1}{h}\cos\theta_1\mathbf{v}_1\right)$$

$$\mathbf{v}_2 = -\frac{\mu}{h}\sin\theta_2\left(\frac{\mu}{h^2}(e + \cos\theta_1)\mathbf{r}_1 - \frac{r_1}{h}\sin\theta_1\mathbf{v}_1\right) + \frac{\mu}{h}(e + \cos\theta_2)\left(\frac{\mu}{h^2}\sin\theta_1\mathbf{r}_1 + \frac{r_1}{h}\cos\theta_1\mathbf{v}_1\right)$$

Finally, arrange $\mathbf{r}_1$ and $\mathbf{v}_1$ terms together to obtain the Lagrange coefficients.

Lagrange Coefficients Solution to Orbital Motion

Let us consider the true anomaly difference $\Delta\theta=\theta_2-\theta_1$, and define;

$$e\cos\theta_1=\frac{p}{r_1}-1 \qquad e\sin\theta_1=\frac{\sqrt{p}}{r_1}\sigma_1 \text{ where } \sigma_1=\frac{\mathbf{r}_1\cdot\mathbf{v}_1}{\sqrt{\mu}}$$

$$r(\Delta\theta)=\frac{pr_1}{r_1+(p-r_1)\cos\Delta\theta-\sqrt{p}\sigma_1\sin\Delta\theta}$$

Then the Lagrange coefficients can be written in the following way:

$$F=1-\frac{r}{p}(1-\cos\Delta\theta); \quad G=\frac{rr_1}{\sqrt{\mu p}}\sin\Delta\theta$$

$$\dot{F}=\frac{\sqrt{\mu}}{r_1 p}\left[\sigma_1(1-\cos\Delta\theta)-\sqrt{p}\sin\Delta\theta\right]; \quad \dot{G}=1-\frac{r_1}{p}(1-\cos\Delta\theta)$$

Minimum Energy Transfer

Statement: Find the $\boldsymbol{v}_1$ for minimum energy orbit.

Recall, $\mathbf{r}_2 = F\mathbf{r}_1 + G\mathbf{v}_1$

$$\mathbf{v}_1 = \frac{1}{G}\left(\mathbf{r}_2 - F\mathbf{r}_1\right)$$

$$F = 1 - \frac{r_2}{p}(1 - \cos\Delta\theta); \quad G = \frac{r_2 r_1}{\sqrt{\mu p}}\sin\Delta\theta$$

MinETransfer $\left(\mathbf{r}_1, \mathbf{r}_2 \Rightarrow a_{\min}, e_{\min}\right)$

Minimum Energy Transfer: Algorithm MinETransfer

Algorithm *MinETransfer* : $\left(\mathbf{r}_1, \mathbf{r}_2, t_m \Rightarrow \dot{\mathbf{r}}_1\right)$

$$c = \left|\mathbf{r}_2 - \mathbf{r}_1\right|$$

$$\cos\Delta\theta = \frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{r_1 r_2}$$

$$\sin\Delta\theta = t_m\sqrt{1-\cos^2\Delta\theta}$$

$$p_{\min} = \frac{r_1 r_2}{c}\left(1-\cos\Delta\theta\right)$$

$$F = 1 - \frac{r_2}{p_{\min}}(1-\cos\Delta\theta)$$

$$G = \frac{r_2 r_1}{\sqrt{\mu\, p_{\min}}}\sin\Delta\theta$$

$$\dot{\mathbf{r}}_1 = \frac{1}{G}\left(\mathbf{r}_2 - F\mathbf{r}_1\right)$$

F & G Solutions for elliptical orbits.

Algorithm *FGKepler_trA*: $\left(\mathbf{r}_0, \dot{\mathbf{r}}_0, \Delta\theta \Rightarrow \mathbf{r}_f\right)$

$$\mathbf{h} = \mathbf{r}_0 \times \dot{\mathbf{r}}_0$$

$$p = \frac{h^2}{\mu}$$

$$\sigma_0 = \frac{\mathbf{r}_0 \cdot \dot{\mathbf{r}}_0}{\sqrt{\mu}}$$

$$r_f = \frac{p r_0}{r_0 + (p - r_0)\cos\Delta\theta - \sqrt{p}\,\sigma_0 \sin\Delta\theta}$$

$$F = 1 - \frac{r_f}{p}(1 - \cos\Delta\theta)$$

$$G = \frac{r_f r_0}{\sqrt{\mu p}} \sin\Delta\theta$$

$$\mathbf{r}_f = F\mathbf{r}_0 + G\dot{\mathbf{r}}_0$$

Download Session 1 MATLAB Guide (PART A) from BB

Exercise 2

ExoMars Trace Gas Orbiter departed Earth on 14/03/2016, and arrived at Mars on 15/10/2016.

Program the F and G solutions to the two body problem. Verify the answer by comparing it to a numerical integration of the differential equations of motion:

$$\boxed{\ddot{\mathbf{r}} + \frac{\mu_S}{r^3}\mathbf{r} = 0}$$

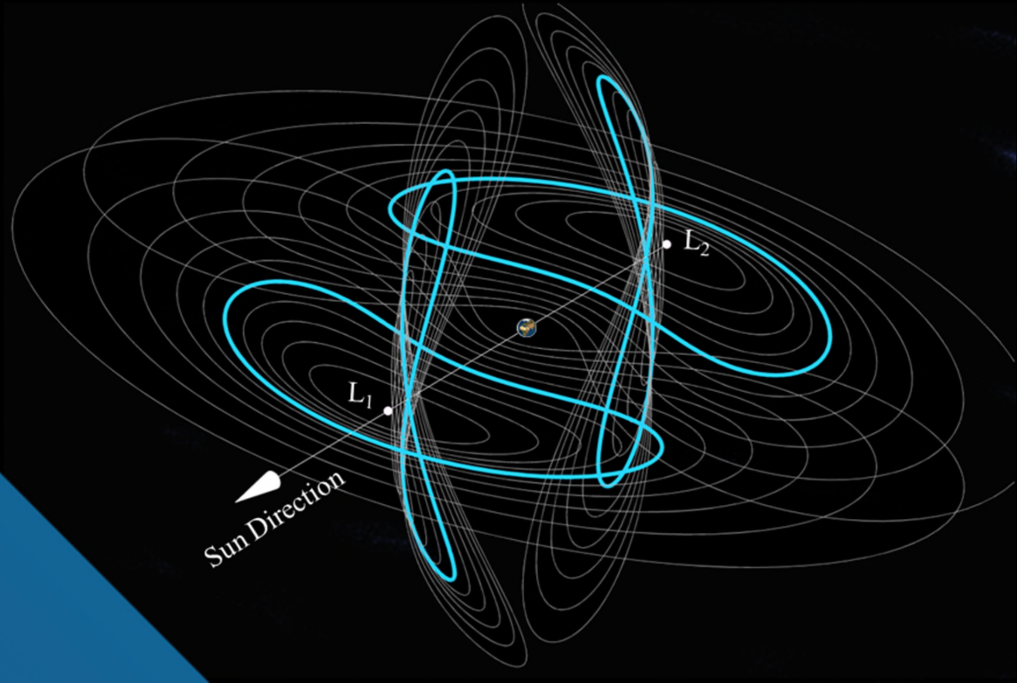
Using the F and G techniques, plot the orbit of Mars, Earth and the minimum energy transfer for ExoMars TGO.

When completed, connect to Socrative Room GPQK5UNSK and indicate it so in the relevant question

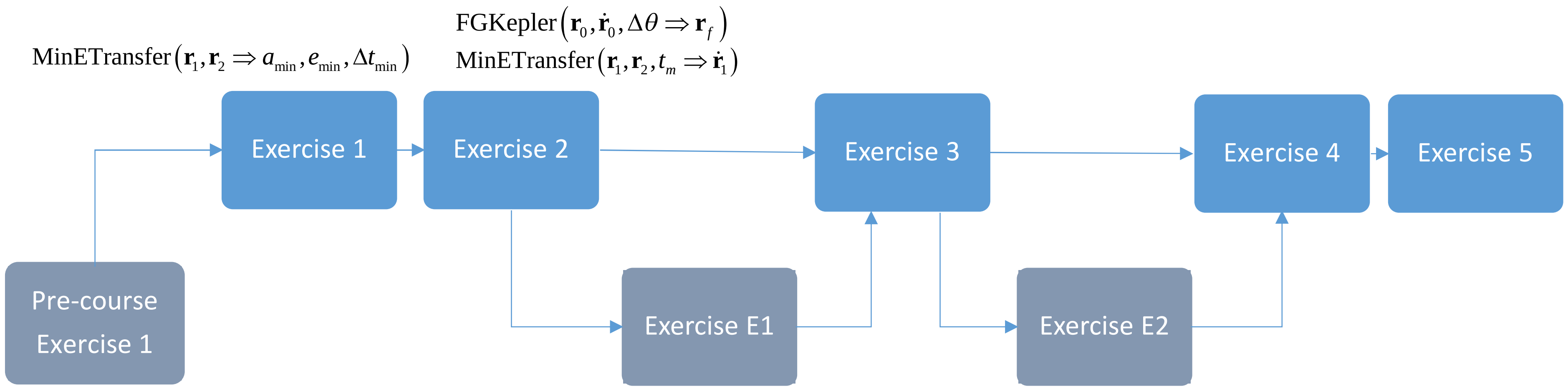
Result Guide: $\dot{r}_1 = \begin{bmatrix} -7.72 & -31.04 & -2.26 \end{bmatrix}$ km/s

Mathematics and Programming for Astrodynamics & Trajectory Design

End of day 1



Exercise Path



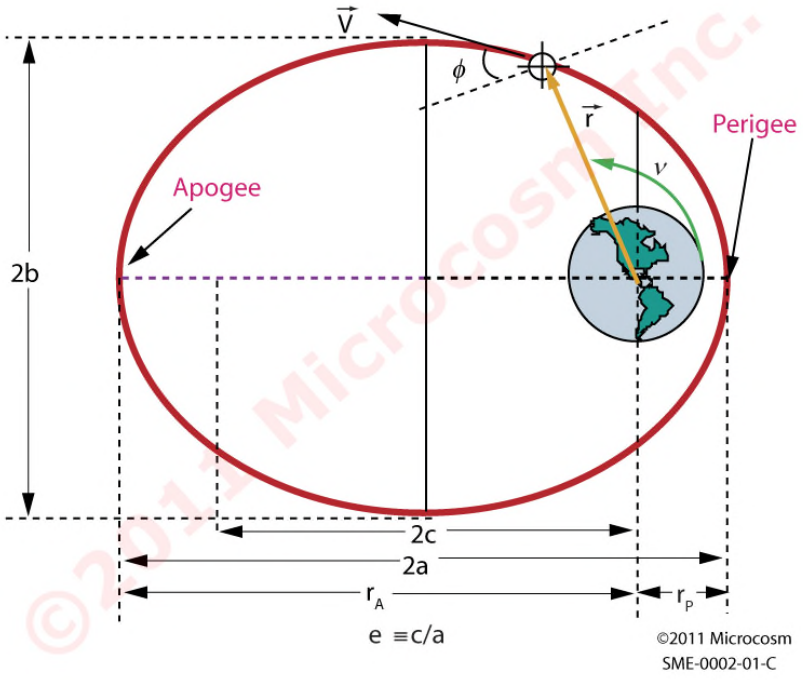
$\mathrm{MinETransfer}\left(\mathbf{r}_1,\mathbf{r}_2 \Rightarrow a_{\min}, e_{\min}, \Delta t_{\min}\right)$
$\mathrm{FGKepler}\left(\mathbf{r}_0,\dot{\mathbf{r}}_0,\Delta\theta \Rightarrow \mathbf{r}_f\right)$
$\mathrm{MinETransfer}\left(\mathbf{r}_1,\mathbf{r}_2,t_m \Rightarrow \dot{\mathbf{r}}_1\right)$
Pre-course Exercise 1
Exercise 1
Exercise 2
Exercise 3
Exercise 4
Exercise 5
Exercise E1
Exercise E2

Position and Velocity Problem as a Function of Time

Kepler's Equation

$$E - e\sin E = nt \qquad nt = M$$

$$\boxed{\left(\mathbf{r}(t), \mathbf{v}(t)\right)}$$



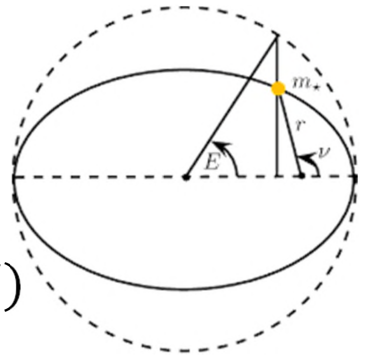
- Two different challenges arise with Kepler's equation:

(1) To compute the time to travel between two known points on an orbit.

(2) To find the location of a spacecraft in orbit after a certain amount of time.

$$\left(\mathbf{r}_0(t_0=0), \dot{\mathbf{r}}_0(t_0=0), \Delta t \Rightarrow M_f \overset{?}{\Rightarrow} E_f \Rightarrow \theta_f \Rightarrow \mathbf{r}_f\right)$$

Newton-Raphson Method



Kepler's equation relates time and position along an orbit.

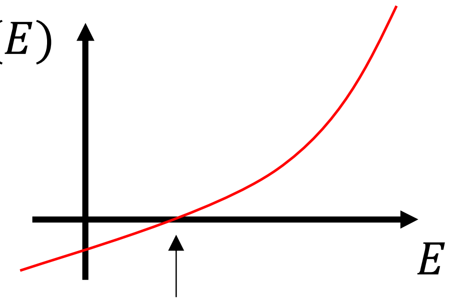
For an ellipse: $M(t) = n\left(t - t_{periapsis}\right) = E - e \cdot sinE = M(E)$

where E is the eccentric anomaly, and

$$n = \sqrt{\frac{\mu}{a^3}} \qquad a = \frac{\mu}{\frac{2\mu}{r} - v^2} \qquad r_{periapsis} = a(1-e)$$

If we want to know the position at a certain time t (i.e., we know $M(t) = M_t$), how can we obtain E? → Newton-Raphson Method.

$f(E) = M_t - M(E)$



value of E so that $f(E) = 0$?

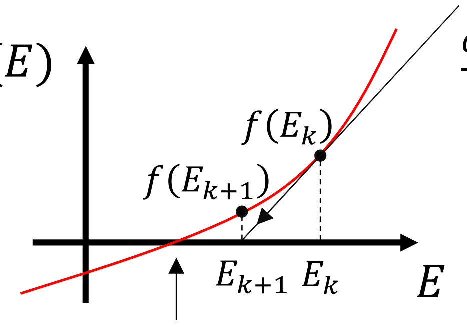
Newton-Raphson Method.

MATLAB already has some functions implemented to solve this kind of problems (e.g., fzero function), but Newton's method will come in useful as concept can be extended to other kinds of problems.

Newton-Raphson method is an **iterative**, **gradient-based** approach to find the root of a function.

Based on a current guess for the solution, E_k, a new guess is obtained through the derivate of function $f(E)$. Process is repeated until "sufficiently" accurate solution is obtained: $|f(E_{k+1})| < \varepsilon \equiv tolerance$

$$f(E) = M_t - M(E)$$



value of E so that $f(E) = 0$?

$$\frac{df(E_k)}{dE} = f'(E_k)$$

$$\longrightarrow \quad E_{k+1} = E_k - \frac{f(E_k)}{f'(E_k)}$$

Newton-Raphson Method.

A Newton-Raphson algorithm has a simple structure:

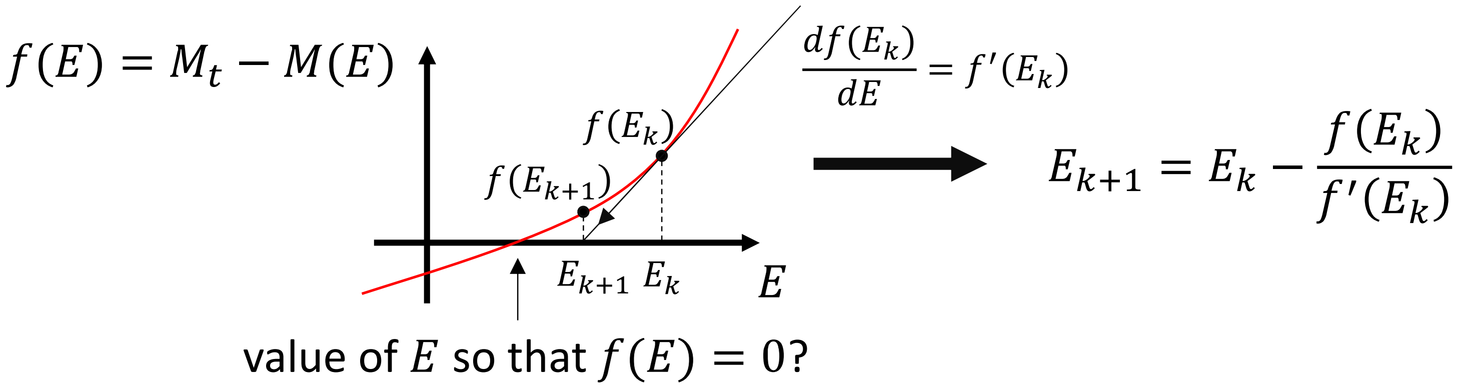
1) Initialize problem: compute $M_t = M(t)$, provide initial guess E_0.
2) Loop: has convergence been achieved?:

$$|f(E_k)| = |M_t - M(E_k)| < \text{tolerance?}$$

If not:

2a) compute derivative at current guess: $f'(E_k) = \frac{df(E_k)}{dE}$.

2b) compute new guess: $E_{k+1} = E_k - \frac{f(E_k)}{f'(E_k)}$


$f(E) = M_t - M(E)$
$\frac{df(E_k)}{dE} = f'(E_k)$
$f(E_k)$
$f(E_{k+1})$
E_{k+1}
E_k
E
$E_{k+1} = E_k - \frac{f(E_k)}{f'(E_k)}$
value of E so that $f(E) = 0$?

Exercise E1. Newton-Raphson Method

Implement a Newton-Raphson algorithm to compute the eccentric anomaly E for a given time Δt after periapsis.

Assume the central body is the Earth, an orbit with a 400-km-altitude perigee, a 6000-km-altitude apogee, and $\Delta t = 0.65$ hours.

Compare the result with the one obtained through MATLAB’s fzero function.

- How many iterations did your algorithm require?
- How did you select your initial guess?
- Can you graphically verify your solution?

Result Guide: 1.86 rad=106.5 deg

Exercise Newton-Raphson Method - Guide

1) Initialize problem: compute $M_t = M(t)$, provide initial guess E_0.

$$a = \frac{1}{2}(r_p + r_a)$$
$$r_p = a(1-e)$$
$$n = \sqrt{\frac{\mu}{a^3}}$$

2) Loop: has convergence been achieved?:
$|f(E_k)| = |M_t - M(E_k)| <$ tolerance?

$$M(t) = M_t = n\left(t - t_p\right)$$
$$M(E_k) = E_k - e \cdot sin(E_k)$$

If not:
2a) compute derivative at current guess.
2b) compute new guess.

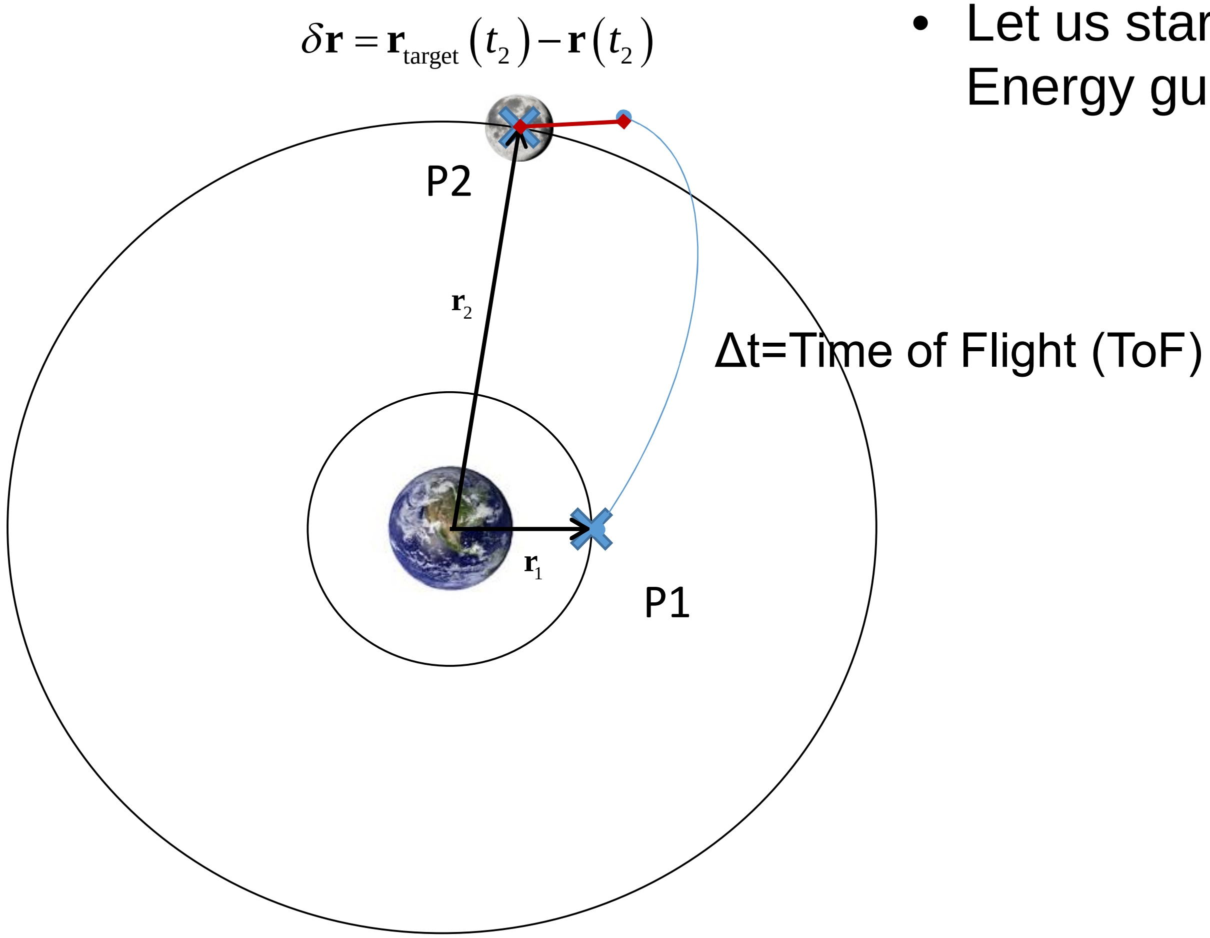
$$f(E_k) = M_t - M(E_k)$$

$$f'(E_k) = \frac{df(E_k)}{dE} = -1 + e \cdot cos(E_k)$$
$$E_{k+1} = E_k - \frac{f(E_k)}{f'(E_k)}$$

Lambert’s Problem

- We will attempt to solve it using a general approach that may be used to solve other targeting problems in higher fidelity dynamics (i.e. *n*-body problem or with other perturbations).
  - ✓ *Pros*: The techniques used are applicable to many other problems and you may come across them in future occasions.
  - ✓ *Cons*: The final algorithm is not very robust and its convergence tends to fail in many occasions.
- Algorithm as described in: *Analytical Mechanics of Space Systems*, H.Shcaub, J.L.Junkins, AIAA Education Series.

Lambert's Problem

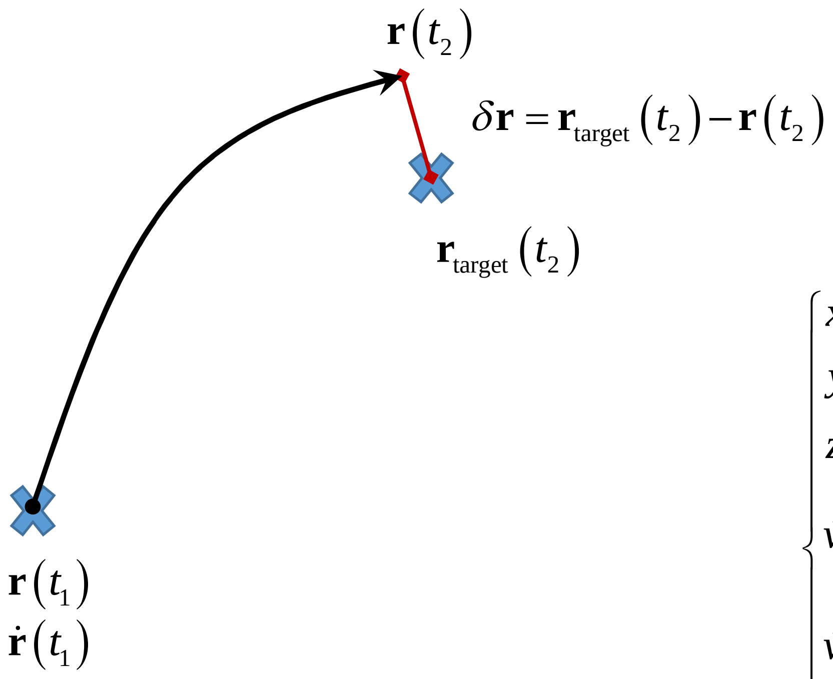


$\delta\mathbf{r} = \mathbf{r}_{\text{target}}(t_2) - \mathbf{r}(t_2)$
P2
$\mathbf{r}_2$
Δt=Time of Flight (ToF)
$\mathbf{r}_1$
P1

- Let us start with our Minimum Energy guess.

Lambert's Problem

Differential Correction or Shooting Method



$\mathbf{r}(t_2)$
$\delta\mathbf{r} = \mathbf{r}_{\text{target}}(t_2) - \mathbf{r}(t_2)$
$\mathbf{r}_{\text{target}}(t_2)$
$\mathbf{r}(t_1)$
$\dot{\mathbf{r}}(t_1)$

$$\begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{z} = v_z \\ \dot{v}_x = -\dfrac{\mu_E}{r^3}x \\ \dot{v}_y = -\dfrac{\mu_E}{r^3}y \\ \dot{v}_z = -\dfrac{\mu_E}{r^3}z \end{cases}$$

'Shoot' $\left(\mathbf{r}(t_1) \quad \dot{\mathbf{r}}(t_1)\right)$

Exercise 3

“Shoot” the minimum energy orbit for a Δt equivalent to the real ExoMars TGO transfer (Δt= 215 days) and compute the $d\mathbf{x}$ error of the final position state.

We still miss the algorithm *FGKepler_dt* in order to complete this task!!!

F & G Solutions for elliptical orbits.

Algorithm *FGKepler_dt*: $\left(\mathbf{r}_0, \dot{\mathbf{r}}_0, \Delta t \Rightarrow \mathbf{r}_f\right)$

$$a = \frac{\mu}{\left(\frac{2\mu}{r_0} - v_0^2\right)} \qquad n = \sqrt{\frac{\mu}{a^3}}$$

$$\Delta M = n \Delta T$$

$$\sigma_0 = \frac{\mathbf{r}_0 \cdot \dot{\mathbf{r}}_0}{\sqrt{\mu}}$$

Newton-Raphson Solver:

$$\Delta M = \Delta E - \left(1 - \frac{r_0}{a}\right) \sin \Delta E - \frac{\sigma_0}{\sqrt{a}} \left(\cos \Delta E - 1\right) \rightarrow \Delta E$$

* Implement using MATLAB's Fzero and initialize the search with ΔM value as first guess.

$$F = 1 - \frac{a}{r_0}(1 - \cos \Delta E)$$

$$G = \Delta T + \sqrt{\frac{a^3}{\mu}}\left(\sin \Delta E - \Delta E\right)$$

$$\mathbf{r}_f = F\mathbf{r}_0 + G\dot{\mathbf{r}}_0$$

Exercise 3

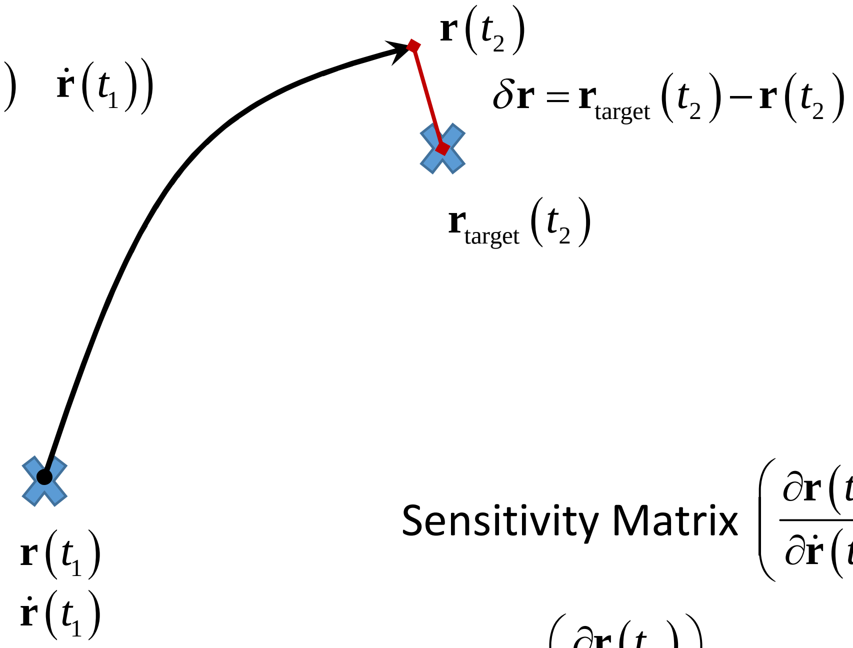
"Shoot" the minimum energy orbit for a Δt equivalent to the real ExoMars TGO transfer (Δt= 215 days) and compute the $d\mathbf{r}$ error of the final position state.

Result Guide: $\delta r = 3.3 \times 10^{7} \text{ km}$

When completed, connect to Socrative Room GPQK5UNSK and indicate it so in the relevant question

Lambert's Problem: *Shooting Method*

'Shoot' $\left(\mathbf{r}(t_1) \quad \dot{\mathbf{r}}(t_1)\right)$


$\mathbf{r}(t_2)$
$\delta\mathbf{r} = \mathbf{r}_{\text{target}}(t_2) - \mathbf{r}(t_2)$
$\mathbf{r}_{\text{target}}(t_2)$
$\mathbf{r}(t_1)$
$\dot{\mathbf{r}}(t_1)$

Sensitivity Matrix $\left(\dfrac{\partial\mathbf{r}(t_2)}{\partial\dot{\mathbf{r}}(t_1)}\right)$

$$\delta\mathbf{r}(t_2) = \left(\frac{\partial\mathbf{r}(t_2)}{\partial\dot{\mathbf{r}}(t_1)}\right)\delta\dot{\mathbf{r}}_{corr}(t_1)$$

$$\delta\dot{\mathbf{r}}_{corr}(t_1) = \left(\frac{\partial\mathbf{r}(t_2)}{\partial\dot{\mathbf{r}}(t_1)}\right)^{-1}\delta\mathbf{r}(t_2)$$

Differential Corrector or *Shooting Method*

- The objective of a differential corrector is to determine which modification of (**r**(t_0),**v**(t_0)) in order to reach a desired state (**r**(t_f),**v**(t_f))

$$\mathbf{x}(t) = g\left(t, \mathbf{x}(t_0)\right)$$

$$\delta\mathbf{x}_{correction} \;:\; \mathbf{x}_{\text{target}}(t) = g\left(t, \mathbf{x}(t_0) + \delta\mathbf{x}_{correction}\right)$$

Taylor series expansion:

$$\mathbf{x}_{\text{target}}(t) = g\left(t, \mathbf{x}(t_0)\right) + \frac{\partial g(\cdot)}{\partial \mathbf{x}}\delta\mathbf{x}_{cor} + \ldots + R_n$$

$$\frac{\partial g(\cdot)}{\partial \mathbf{x}} = \Phi(t, t_0) \equiv \text{STM}$$

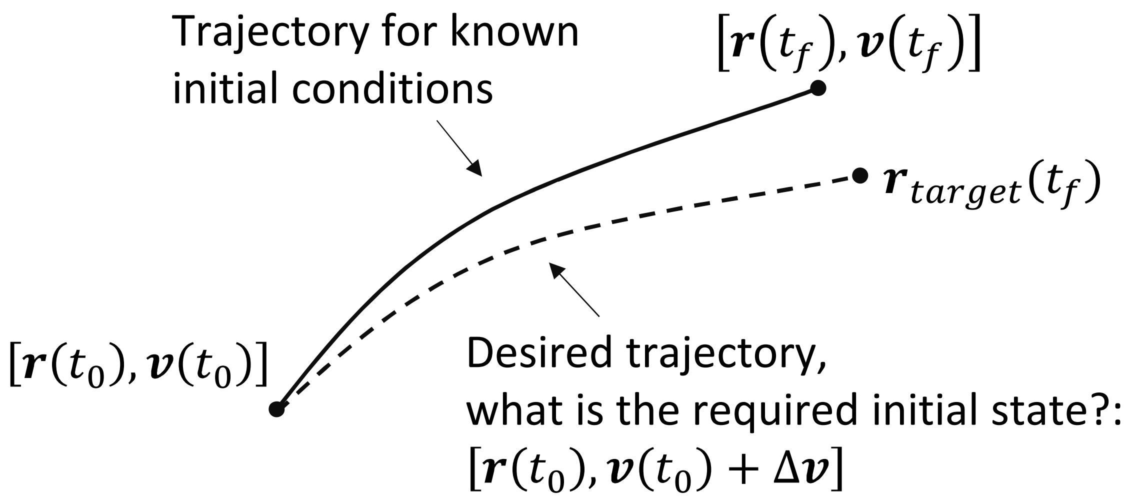
$$\delta\mathbf{x}_{cor} = \left(\frac{\partial g(\cdot)}{\partial \mathbf{x}}\right)^{-1}\left(\mathbf{x}_{\text{target}}(t) - \mathbf{x}(t)\right)$$

Recall the non-linearity:

$$\mathbf{x}_{\text{target}}(t) - g\left(t, \mathbf{x}(t_0) + \delta\mathbf{x}_{cor}\right) \overset{?}{\neq} 0$$

State-transition matrix (STM)

Provided a certain position and velocity at time t_0, what is the required change in velocity to achieve a desired change in position at time t_1?


Trajectory for known initial conditions
$[\boldsymbol{r}(t_f), \boldsymbol{v}(t_f)]$
$\boldsymbol{r}_{target}(t_f)$
$[\boldsymbol{r}(t_0), \boldsymbol{v}(t_0)]$
Desired trajectory, what is the required initial state?: $[\boldsymbol{r}(t_0), \boldsymbol{v}(t_0) + \Delta \boldsymbol{v}]$

$$\begin{bmatrix} \mathbf{r}_{\text{target}}(t_f) \\ \mathbf{v}_{\text{target}}(t_f) \end{bmatrix} \approx g\left(t, \mathbf{r}(t_0), \mathbf{v}(t_0)\right) + \Phi(t_f, t_0)\delta\mathbf{x}_{cor} \qquad \delta\mathbf{x}_{cor} = \begin{bmatrix} \mathbf{0}_{3\times1} \\ \partial\mathbf{v}_{cor}(t_0) \end{bmatrix}$$

State-transition matrix (STM)

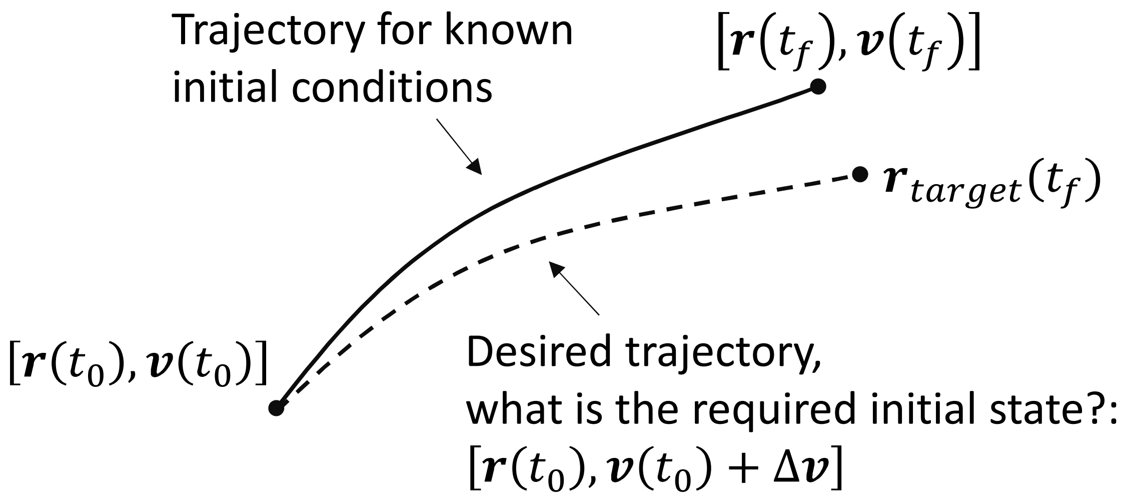
Ultimately, the STM relates variations in initial state to variations in final state:

$$\boldsymbol{\phi}(t_f,t_0) = \left[\begin{array}{ccc|ccc} \phi_{11} & \phi_{12} & \phi_{13} & \phi_{14} & \phi_{15} & \phi_{16} \\ \phi_{21} & \phi_{22} & \phi_{23} & \phi_{24} & \phi_{25} & \phi_{26} \\ \phi_{31} & \phi_{32} & \phi_{33} & \phi_{34} & \phi_{35} & \phi_{36} \\ \hline \phi_{41} & \phi_{42} & \phi_{43} & \phi_{44} & \phi_{45} & \phi_{46} \\ \phi_{51} & \phi_{52} & \phi_{53} & \phi_{54} & \phi_{55} & \phi_{56} \\ \phi_{61} & \phi_{62} & \phi_{63} & \phi_{64} & \phi_{65} & \phi_{66} \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{\phi}_{rr} & \boldsymbol{\phi}_{rv} \\ \hline \boldsymbol{\phi}_{vr} & \boldsymbol{\phi}_{vv} \end{array}\right]$$

The different blocks within the STM relate different elements of the state vectors:

$$\delta\boldsymbol{x}(t_f) = \boldsymbol{\phi}(t_f,t_0)\cdot\delta\boldsymbol{x}(t_0) \longrightarrow \begin{array}{c} \delta\boldsymbol{r}(t_f) \\ \\ \delta\boldsymbol{v}(t_f) \end{array} \begin{cases} \\ \\ \end{cases} \begin{bmatrix} \delta x(t_f) \\ \delta y(t_f) \\ \delta z(t_f) \\ \delta v_x(t_f) \\ \delta v_y(t_f) \\ \delta v_z(t_f) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}_{rr} & \boldsymbol{\phi}_{rv} \\ \boldsymbol{\phi}_{vr} & \boldsymbol{\phi}_{vv} \end{bmatrix} \begin{bmatrix} \delta x(t_0) \\ \delta y(t_0) \\ \delta z(t_0) \\ \delta v_x(t_0) \\ \delta v_y(t_0) \\ \delta v_z(t_0) \end{bmatrix} \begin{array}{c} \left.\right\} \delta\boldsymbol{r}(t_0) \\ \\ \left.\right\} \delta\boldsymbol{v}(t_0) \end{array}$$

State-transition matrix (STM)



Trajectory for known initial conditions
$[\boldsymbol{r}(t_f), \boldsymbol{v}(t_f)]$
$\boldsymbol{r}_{target}(t_f)$
$[\boldsymbol{r}(t_0), \boldsymbol{v}(t_0)]$
Desired trajectory, what is the required initial state?: $[\boldsymbol{r}(t_0), \boldsymbol{v}(t_0) + \Delta\boldsymbol{v}]$

$$\begin{bmatrix} \mathbf{r}_{\text{target}}(t_f) \\ \mathbf{v}_{\text{target}}(t_f) \end{bmatrix} \approx \begin{bmatrix} \mathbf{r}(t_f) \\ \mathbf{v}(t_f) \end{bmatrix} + \begin{bmatrix} \Phi_{rr} & \Phi_{rv} \\ \Phi_{vr} & \Phi_{vv} \end{bmatrix} \begin{bmatrix} \mathbf{0}_{3\times1} \\ \partial\mathbf{v}_{cor}(t_0) \end{bmatrix}$$

$$\mathbf{r}_{\text{target}}(t_f) \approx \mathbf{r}(t_f) + \Phi_{rv}\partial\mathbf{v}_{cor}(t_0)$$

$$\mathbf{r}_{\text{target}}(t_f) - \mathbf{r}(t_f) = \partial\mathbf{r}_{cor}(t_f) \approx \Phi_{rv}\partial\mathbf{v}_{cor}(t_0)$$

$$\partial\mathbf{v}_{cor}(t_0) = \Phi_{rv}^{-1}\partial\mathbf{r}_{cor}(t_f)$$

State-transition matrix Φ(STM) for 2BP

The particular implementation of the STM in 2BP have a closed-form solutions which can be described using F and G coefficient.

In particular the block $\boldsymbol{\phi}_{rv}$, which is the one you require to know the initial velocity $\boldsymbol{v}(t_0)$ to achieve a desired final position $\boldsymbol{r}(t_1)$, can be computed as:

$$\frac{\partial \mathbf{r}(t_f)}{\partial \mathbf{v}(t_0)} = \mathbf{\Phi}_{rv} = \frac{\partial \mathbf{r}_f}{\partial \dot{\mathbf{r}}_0} = \frac{r_0}{\mu}(1-F)\left(\Delta\mathbf{r}\cdot\dot{\mathbf{r}}_0^T - \Delta\mathbf{v}\cdot\mathbf{r}_0^T\right) + \frac{C}{\mu}\dot{\mathbf{r}}_f\cdot\dot{\mathbf{r}}_0^T + G\cdot I_{3\times3}$$

$$\Delta\mathbf{r} = \mathbf{r}_f - \mathbf{r}_0 \qquad \Delta\mathbf{v} = \dot{\mathbf{r}}_f - \dot{\mathbf{r}}_0$$

$$\dot{G} = 1 - \frac{a}{r_f}(1-\cos\Delta E) \qquad \dot{F} = -\frac{\sqrt{\mu a}}{r_f r_0}\sin(\Delta E) \qquad \dot{\boldsymbol{r}}_f = \dot{F}\boldsymbol{r}_0 + \dot{G}\dot{r}_0$$

$$C = a\sqrt{\frac{a^3}{\mu}}\left(3\sin\Delta E - (2+\cos\Delta E)\Delta E\right) - a\Delta t(1-\cos\Delta E)$$

F & G Solutions for elliptical orbits.

Algorithm *STM_Lambert*: $\left(\mathbf{r}_0, \dot{\mathbf{r}}_0, \Delta t \Rightarrow \frac{\partial \mathbf{r}_f}{\partial \dot{\mathbf{r}}_0}\right)$

$$\dot{F} = -\frac{\sqrt{\mu a}}{r_f r_0}\sin(\Delta E) \qquad \dot{G} = 1 - \frac{a}{r_f}\left(1 - \cos\Delta E\right)$$

$$\dot{\boldsymbol{r}}_f = \dot{F}\boldsymbol{r}_0 + \dot{G}\dot{r}_0$$

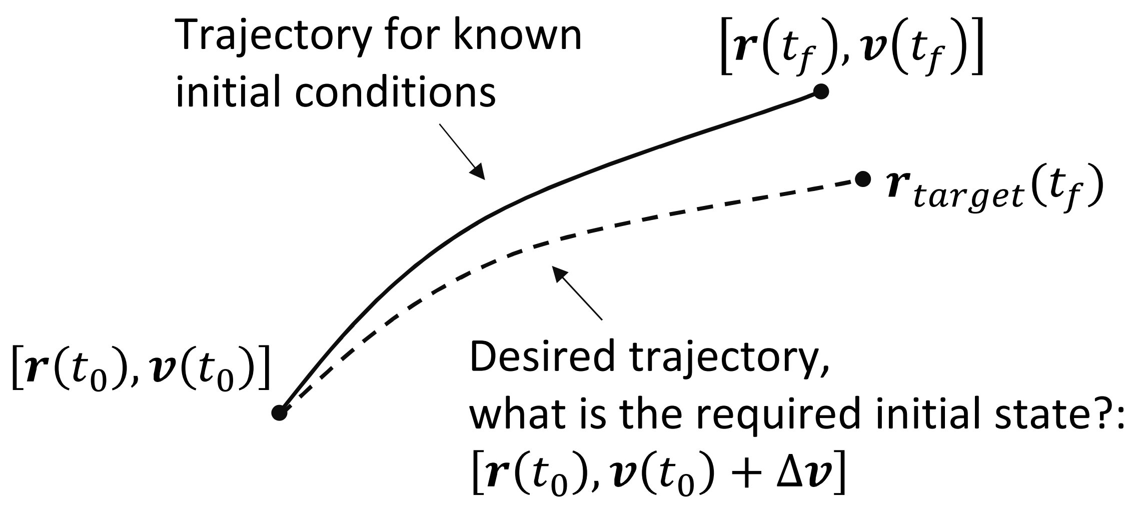
$$C = a\sqrt{\frac{a^3}{\mu}}\left(3\sin\Delta E - \left(2 + \cos\Delta E\right)\Delta E\right) - a\Delta t\left(1 - \cos\Delta E\right)$$

$$\Delta\mathbf{r} = \mathbf{r}_f - \mathbf{r}_0 \qquad \Delta\mathbf{v} = \dot{\mathbf{r}}_f - \dot{\mathbf{r}}_0$$

$$\frac{\partial \mathbf{r}_f}{\partial \dot{\mathbf{r}}_0} = \frac{r_0}{\mu}\left(1 - F\right)\left(\Delta\mathbf{r}\cdot\dot{\mathbf{r}}_0^T - \Delta\mathbf{v}\cdot\mathbf{r}_0^T\right) + \frac{C}{\mu}\dot{\mathbf{r}}_f\cdot\dot{\mathbf{r}}_0^T + G\cdot I_{3\times3}$$

Differential Correction or Shooting Method

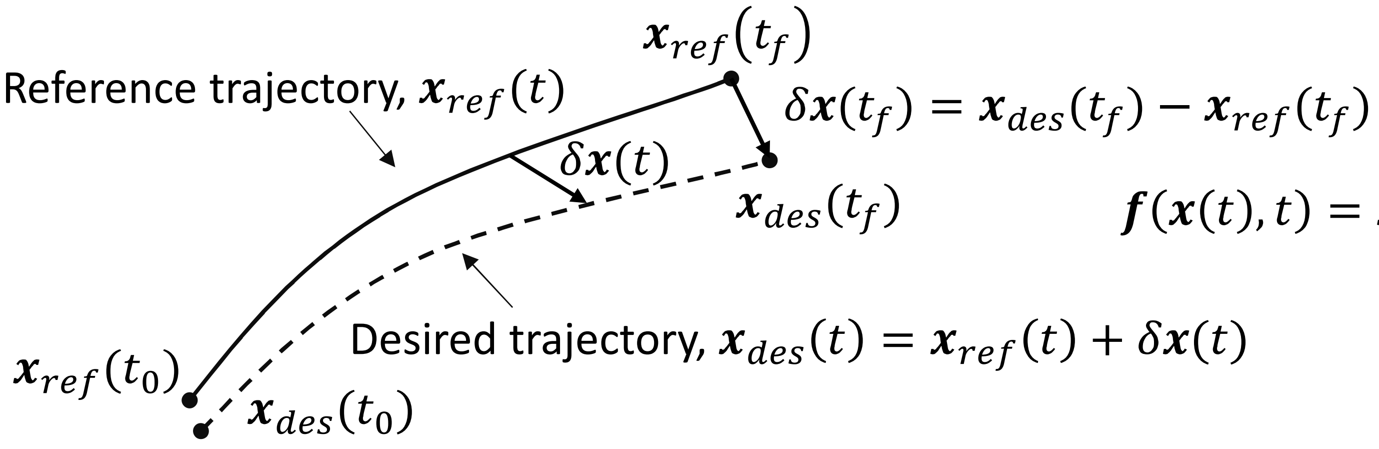
Provided a certain position and velocity at time t_0, what is the required change in velocity to achieve a desired change in position at time t_f?


Trajectory for known initial conditions
$[\boldsymbol{r}(t_f), \boldsymbol{v}(t_f)]$
$\boldsymbol{r}_{target}(t_f)$
$[\boldsymbol{r}(t_0), \boldsymbol{v}(t_0)]$
Desired trajectory, what is the required initial state?: $[\boldsymbol{r}(t_0), \boldsymbol{v}(t_0) + \Delta\boldsymbol{v}]$

Two-body problem EOM:

$$\begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{z} = v_z \\ \dot{v}_x = -\dfrac{\mu_E}{r^3} x \\ \dot{v}_y = -\dfrac{\mu_E}{r^3} y \\ \dot{v}_z = -\dfrac{\mu_E}{r^3} z \end{cases}$$

Differential Correction or Shooting Method



$\boldsymbol{x}_{ref}(t_f)$
Reference trajectory, $\boldsymbol{x}_{ref}(t)$
$\delta\boldsymbol{x}(t_f) = \boldsymbol{x}_{des}(t_f) - \boldsymbol{x}_{ref}(t_f)$
$\delta\boldsymbol{x}(t)$
$\boldsymbol{x}_{des}(t_f)$
Desired trajectory, $\boldsymbol{x}_{des}(t) = \boldsymbol{x}_{ref}(t) + \delta\boldsymbol{x}(t)$
$\boldsymbol{x}_{ref}(t_0)$
$\boldsymbol{x}_{des}(t_0)$

Two-body problem EOM:

$$\boldsymbol{f}(\boldsymbol{x}(t), t) = \dot{\boldsymbol{x}}(t) = \begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{z} = v_z \\ \dot{v}_x = -\dfrac{\mu_E}{r^3} x \\ \dot{v}_y = -\dfrac{\mu_E}{r^3} y \\ \dot{v}_z = -\dfrac{\mu_E}{r^3} z \end{cases}$$

The EOM along the **reference** trajectory are: $\boldsymbol{f}\left(\boldsymbol{x}_{ref}(t), t\right)$

The EOM along the **desired** trajectory are: $\boldsymbol{f}_{des} = \boldsymbol{f}\left(\boldsymbol{x}_{ref}(t) + \delta\boldsymbol{x}(t), t\right)$

$\boldsymbol{f}_{des}$ can be linearly approximated through a first-order Taylor expansion as:

$$\boldsymbol{f}_{des} = \dot{\boldsymbol{x}}_{des}(t) = \dot{\boldsymbol{x}}_{ref}(t) + \delta\dot{\boldsymbol{x}}(t) \approx \boldsymbol{f}\left(\boldsymbol{x}_{ref}(t), t\right) + \underbrace{\frac{\delta\boldsymbol{f}\left(\boldsymbol{x}_{ref}(t), t\right)}{\delta\boldsymbol{x}(t)}} \delta\boldsymbol{x}(t)$$

$\boldsymbol{A}(\boldsymbol{x}_{ref}(t), t) \equiv$ Jacobian matrix: partial derivatives of EOM with respect to state vector, along reference trajectory

Differential Correction or Shooting Method

We have an approximation for the EOM along the desired trajectory in terms of the reference trajectory and the deviation from it:

$$\boldsymbol{f}_{des} = \dot{\boldsymbol{x}}_{des}(t) = \dot{\boldsymbol{x}}_{ref}(t) + \delta\dot{\boldsymbol{x}}(t) \approx \boldsymbol{f}\big(\boldsymbol{x}_{ref}(t), t\big) + \underbrace{\frac{\delta \boldsymbol{f}\big(\boldsymbol{x}_{ref}(t), t\big)}{\delta \boldsymbol{x}(t)}}_{\boldsymbol{A}(\boldsymbol{x}_{ref}(t), t)} \delta\boldsymbol{x}(t)$$

The EOM for the **deviation vector** $\delta\boldsymbol{x}(t)$ can therefore be approximated as:

$$\delta\dot{\boldsymbol{x}}(t) \approx \boldsymbol{A}(\boldsymbol{x}_{ref}(t), t) \cdot \delta\boldsymbol{x}(t) \longrightarrow \text{Linear system}$$

What is the behavior of $\delta\boldsymbol{x}(t)$ for a given change in initial conditions $\delta\boldsymbol{x}(t_0)$?

General solution to a linear system of ordinary differential equations (ODE) can be constructed from a linear combination of independent solutions:

$$\boxed{\delta\boldsymbol{x}(t) = \underbrace{\boldsymbol{\phi}(t, t_0)}_{\text{Square matrix}} \cdot \delta\boldsymbol{x}(t_0)}$$

At t_0: $\boldsymbol{\phi}(t_0, t_0) =$ identity matrix

$$\dot{\boldsymbol{\phi}}(t, t_0) = \boldsymbol{A}\Big(x_{ref}(t), (t)\Big) \cdot \boldsymbol{\phi}(t, t_0)$$

Differential Correction or Shooting Method

We therefore have an expression for the deviation from the reference trajectory along time $\delta\boldsymbol{x}(t)$ as a function of the initial deviation $\delta\boldsymbol{x}(t_0)$.

The matrix relating these two quantities is referred to as **state-transition matrix**:

$$\delta\boldsymbol{x}(t) = \underbrace{\boldsymbol{\phi}(t, t_0)}_{\textbf{State-transition matrix (STM)}} \cdot \delta\boldsymbol{x}(t_0)$$

At t_0: $\boldsymbol{\phi}(t_0, t_0) =$ identity matrix

$$\dot{\boldsymbol{\phi}}(t, t_0) = \boldsymbol{A}\left(x_{ref}(t), (t)\right) \cdot \boldsymbol{\phi}(t, t_0)$$

Therefore, if we want to know the required change in initial conditions to achieve a desired state at time t_1, we can simply invert the above equation:

$$\delta\boldsymbol{x}(t_0) = \boldsymbol{\phi}\left(t_f, t_0\right)^{-1} \cdot \delta\boldsymbol{x}(t_1)$$

For which we need to know matrix $\boldsymbol{\phi}(t_f, t_0)$, which can be numerically integrated from time t_0 to t_f along with the reference trajectory.

Differential Correction or Shooting Method

At t_0: $\boldsymbol{\phi}(t_0, t_0) =$ identity matrix

$\dot{\boldsymbol{\phi}}(t, t_0) = \boldsymbol{A}\left(x_{ref}(t), (t)\right) \cdot \boldsymbol{\phi}(t, t_0)$ $\longrightarrow$ Integrate STM from t_0 to t_f to obtain $\boldsymbol{\phi}(t_f, t_0)$

For the specific case of the two-body problem, our EOM are as below.

These equations can be numerically integrated with the **reference initial conditions**:

$$\boldsymbol{f}(\boldsymbol{x}(t), t) = \begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{z} = v_z \\ \dot{v}_x = -\dfrac{\mu_E}{r^3} x \\ \dot{v}_y = -\dfrac{\mu_E}{r^3} y \\ \dot{v}_z = -\dfrac{\mu_E}{r^3} z \end{cases}$$

In order to integrate the STM along with the EOM, we need the Jacobian matrix $\boldsymbol{A}\left(x_{ref}(t), (t)\right)$:

$$\boldsymbol{A}(\boldsymbol{x}(t), (t)) = \frac{\delta \boldsymbol{f}(\boldsymbol{x}(t), t)}{\delta \boldsymbol{x}(t)} =$$

$$\begin{bmatrix} \delta\dot{x}/\delta x & \delta\dot{x}/\delta y & \delta\dot{x}/\delta z & \delta\dot{x}/\delta v_x & \delta\dot{x}/\delta v_y & \delta\dot{x}/\delta v_z \\ \delta\dot{y}/\delta x & \delta\dot{y}/\delta y & \delta\dot{y}/\delta z & \delta\dot{y}/\delta v_x & \delta\dot{y}/\delta v_y & \delta\dot{y}/\delta v_z \\ \delta\dot{z}/\delta x & \delta\dot{z}/\delta y & \delta\dot{z}/\delta z & \delta\dot{z}/\delta v_x & \delta\dot{z}/\delta v_y & \delta\dot{z}/\delta v_z \\ \delta\dot{v}_x/\delta x & \delta\dot{v}_x/\delta y & \delta\dot{v}_x/\delta z & \delta\dot{v}_x/\delta v_x & \delta\dot{v}_x/\delta v_y & \delta\dot{v}_x/\delta v_z \\ \delta\dot{v}_y/\delta x & \delta\dot{v}_y/\delta y & \delta\dot{v}_y/\delta z & \delta\dot{v}_y/\delta v_x & \delta\dot{v}_y/\delta v_y & \delta\dot{v}_y/\delta v_z \\ \delta\dot{v}_z/\delta x & \delta\dot{v}_z/\delta y & \delta\dot{v}_z/\delta z & \delta\dot{v}_z/\delta v_x & \delta\dot{v}_z/\delta v_y & \delta\dot{v}_z/\delta v_z \end{bmatrix}$$

Differential Correction or Shooting Method

If we integrate the EOM and the STM from time t_0 to time t_f, we can compute the required change in initial conditions $\delta\boldsymbol{x}(t_0)$ for a desired change in final state $\delta\boldsymbol{x}(t_f)$:

$$\delta\boldsymbol{x}(t_0) = \boldsymbol{\phi}\left(t_f, t_0\right)^{-\mathbf{1}} \cdot \delta\boldsymbol{x}(t_f)$$

The value of $\delta\boldsymbol{x}(t_0)$ is, however, obtained through a **linear approximation of the EOM**, whereas our system of EOM is **nonlinear**.

It may therefore be necessary to iteratively compute $\delta\boldsymbol{x}(t_0)$ until a final deviation “sufficiently” close to the desired $\delta\boldsymbol{x}(t_f)$ is obtained (i.e., convergence):

$\rightarrow$ **Differential corrector or shooting method**.

Differential Correction or Shooting Method

A shooting algorithm has the following structure:

1) Initialize problem: provide reference initial conditions $\boldsymbol{x}_{ref}(t_0)$, define desired final state $\boldsymbol{x}_{des}(t_f)$, define time interval $[t_0, t_f]$.

2) Iteration 0:

 2a) propagate reference EOM and STM from t_0 to t_f with reference initial conditions.

 2b) calculate difference between resulting final state and desired final state: $\delta\boldsymbol{x}(t_f) = \boldsymbol{x}_{des}(t_f) - \boldsymbol{x}_{ref}(t_f)$.

3) Loop: (if $|\delta\boldsymbol{x}(t_f)| > \varepsilon \equiv$ tolerance)

 3a) compute change in initial conditions: $\delta\boldsymbol{x}(t_0) = \boldsymbol{\phi}(t_f, t_0)^{\mathbf{-1}} \cdot \delta\boldsymbol{x}(t_f)$.

 3b) update reference initial conditions: $\boldsymbol{x}_{ref}^{k+1}(t_0) = \boldsymbol{x}_{ref}^{k}(t_0) + \delta\boldsymbol{x}(t_0)$.

 3c) propagate EOM and STM from t_0 to t_1 with updated reference initial conditions.

 3d) calculate difference between final state of updated reference trajectory and desired final state: $\delta\boldsymbol{x}(t_f) = \boldsymbol{x}_{des}(t_f) - \boldsymbol{x}_{ref}(t_f)$.

- Repeat loop until convergence is achieved or maximum number of iterations is reached (algorithm is not guaranteed to convergence, initial guess matters).

State-transition matrix (STM)

At t_0: $\boldsymbol{\phi}(t_0, t_0) =$ identity matrix

$$\dot{\boldsymbol{\phi}}(t, t_0) = \boldsymbol{A}\left(x_{ref}(t), (t)\right) \cdot \boldsymbol{\phi}(t, t_0) \longrightarrow$$ Integrate STM from t_0 to t_f to obtain $\boldsymbol{\phi}(t_f, t_0)$

In the two-body problem:

$$\boldsymbol{A}\big(\boldsymbol{x}(t), (t)\big) = \frac{\delta \boldsymbol{f}(\boldsymbol{x}(t), t)}{\delta \boldsymbol{x}(t)} =$$

$$\begin{bmatrix} \delta\dot{x}/\delta x & \delta\dot{x}/\delta y & \delta\dot{x}/\delta z & \delta\dot{x}/\delta v_x & \delta\dot{x}/\delta v_y & \delta\dot{x}/\delta v_z \\ \delta\dot{y}/\delta x & \delta\dot{y}/\delta y & \delta\dot{y}/\delta z & \delta\dot{y}/\delta v_x & \delta\dot{y}/\delta v_y & \delta\dot{y}/\delta v_z \\ \delta\dot{z}/\delta x & \delta\dot{z}/\delta y & \delta\dot{z}/\delta z & \delta\dot{z}/\delta v_x & \delta\dot{z}/\delta v_y & \delta\dot{z}/\delta v_z \\ \delta\dot{v}_x/\delta x & \delta\dot{v}_x/\delta y & \delta\dot{v}_x/\delta z & \delta\dot{v}_x/\delta v_x & \delta\dot{v}_x/\delta v_y & \delta\dot{v}_x/\delta v_z \\ \delta\dot{v}_y/\delta x & \delta\dot{v}_y/\delta y & \delta\dot{v}_y/\delta z & \delta\dot{v}_y/\delta v_x & \delta\dot{v}_y/\delta v_y & \delta\dot{v}_y/\delta v_z \\ \delta\dot{v}_z/\delta x & \delta\dot{v}_z/\delta y & \delta\dot{v}_z/\delta z & \delta\dot{v}_z/\delta v_x & \delta\dot{v}_z/\delta v_y & \delta\dot{v}_z/\delta v_z \end{bmatrix}$$

$$\boldsymbol{f}(\boldsymbol{x}(t), t) = \begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{z} = v_z \\ \dot{v}_x = -\dfrac{\mu_E}{r^3} x \\ \dot{v}_y = -\dfrac{\mu_E}{r^3} y \\ \dot{v}_z = -\dfrac{\mu_E}{r^3} z \end{cases}$$

$$\boldsymbol{A}_{21} = \begin{bmatrix} -\dfrac{\mu_E}{r^3} + \dfrac{3\mu_E x^2}{r^5} & \dfrac{3\mu_E xy}{r^5} & \dfrac{3\mu_E xz}{r^5} \\ \dfrac{3\mu_E xy}{r^5} & -\dfrac{\mu_E}{r^3} + \dfrac{3\mu_E y^2}{r^5} & \dfrac{3\mu_E yz}{r^5} \\ \dfrac{3\mu_E xz}{r^5} & \dfrac{3\mu_E yz}{r^5} & -\dfrac{\mu_E}{r^3} + \dfrac{3\mu_E z^2}{r^5} \end{bmatrix}$$

$$\boldsymbol{A}(\boldsymbol{x}(t), t) = \begin{bmatrix} \boldsymbol{0}_{3x3} & \boldsymbol{I}_{3x3} \\ \boldsymbol{A}_{21} & \boldsymbol{0}_{3x3} \end{bmatrix}$$

State-transition matrix (STM)

Ultimately, the STM relates variations in initial state to variations in final state:

$$\boldsymbol{\phi}(t_f,t_0)=\left[\begin{array}{ccc|ccc}\phi_{11}&\phi_{12}&\phi_{13}&\phi_{14}&\phi_{15}&\phi_{16}\\ \phi_{21}&\phi_{22}&\phi_{23}&\phi_{24}&\phi_{25}&\phi_{26}\\ \phi_{31}&\phi_{32}&\phi_{33}&\phi_{34}&\phi_{35}&\phi_{36}\\ \hline \phi_{41}&\phi_{42}&\phi_{43}&\phi_{44}&\phi_{45}&\phi_{46}\\ \phi_{51}&\phi_{52}&\phi_{53}&\phi_{54}&\phi_{55}&\phi_{56}\\ \phi_{61}&\phi_{62}&\phi_{63}&\phi_{64}&\phi_{65}&\phi_{66}\end{array}\right]=\left[\begin{array}{c|c}\boldsymbol{\phi}_{rr}&\boldsymbol{\phi}_{rv}\\ \hline \boldsymbol{\phi}_{vr}&\boldsymbol{\phi}_{vv}\end{array}\right]$$

The different blocks within the STM relate different elements of the state vectors:

$$\delta\boldsymbol{x}(t_f)=\boldsymbol{\phi}(t_f,t_0)\cdot\delta\boldsymbol{x}(t_0)\longrightarrow \begin{array}{l}\delta\boldsymbol{r}(t_f)\left\{\begin{array}{l}\\ \\ \\ \end{array}\right.\\ \delta\boldsymbol{v}(t_f)\left\{\begin{array}{l}\\ \\ \\ \end{array}\right.\end{array}\begin{bmatrix}\delta x(t_f)\\ \delta y(t_f)\\ \delta z(t_f)\\ \delta v_x(t_f)\\ \delta v_y(t_f)\\ \delta v_z(t_f)\end{bmatrix}=\begin{bmatrix}\boldsymbol{\phi}_{rr}&\boldsymbol{\phi}_{rv}\\ \boldsymbol{\phi}_{vr}&\boldsymbol{\phi}_{vv}\end{bmatrix}\begin{bmatrix}\delta x(t_0)\\ \delta y(t_0)\\ \delta z(t_0)\\ \delta v_x(t_0)\\ \delta v_y(t_0)\\ \delta v_z(t_0)\end{bmatrix}\begin{array}{l}\left.\begin{array}{l}\\ \\ \\ \end{array}\right\}\delta\boldsymbol{r}(t_0)\\ \left.\begin{array}{l}\\ \\ \\ \end{array}\right\}\delta\boldsymbol{v}(t_0)\end{array}$$

State-transition matrix (STM)

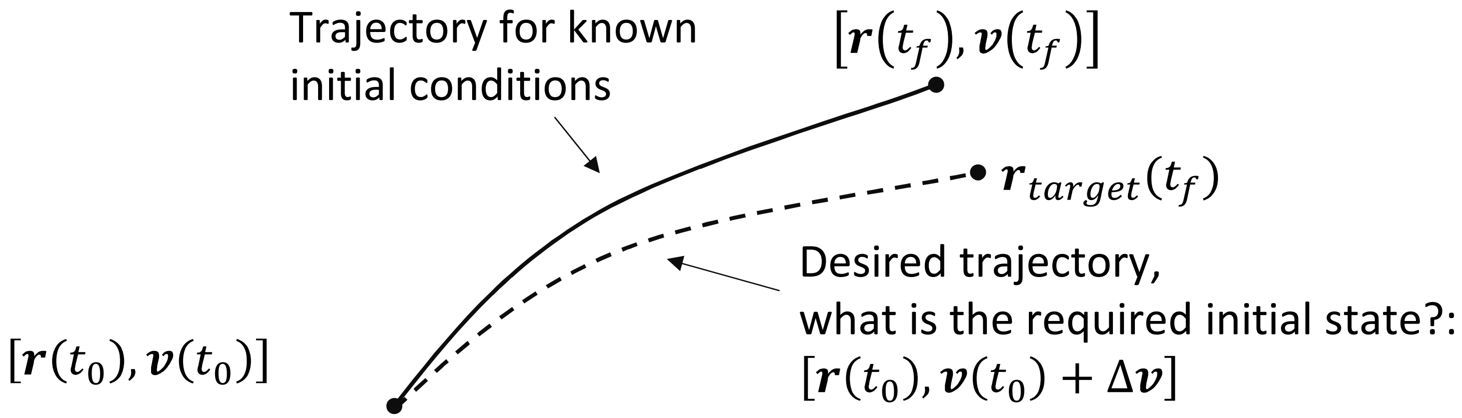
Ultimately, the STM relates variations in initial state to variations in final state:

$$\boldsymbol{\phi}(t_f, t_0) = \begin{bmatrix} \boldsymbol{\phi}_{rr} & \boldsymbol{\phi}_{rv} \\ \boldsymbol{\phi}_{vr} & \boldsymbol{\phi}_{vv} \end{bmatrix} = \begin{bmatrix} \frac{\delta x(t_f)}{\delta x(t_0)} & \frac{\delta x(t_f)}{\delta y(t_0)} & \frac{\delta x(t_f)}{\delta z(t_0)} & \frac{\delta x(t_f)}{\delta v_x(t_0)} & \frac{\delta x(t_f)}{\delta v_y(t_0)} & \frac{\delta x(t_f)}{\delta v_z(t_0)} \\ \frac{\delta y(t_f)}{\delta x(t_0)} & \frac{\delta y(t_f)}{\delta y(t_0)} & \frac{\delta y(t_f)}{\delta z(t_0)} & \frac{\delta y(t_f)}{\delta v_x(t_0)} & \frac{\delta y(t_f)}{\delta v_y(t_0)} & \frac{\delta y(t_f)}{\delta z(t_0)} \\ \frac{\delta z(t_f)}{\delta x(t_0)} & \frac{\delta z(t_f)}{\delta y(t_0)} & \frac{\delta z(t_f)}{\delta z(t_0)} & \frac{\delta z(t_f)}{\delta v_x(t_0)} & \frac{\delta z(t_f)}{\delta v_y(t_0)} & \frac{\delta z(t_f)}{\delta v_z(t_0)} \\ \frac{\delta v_x(t_f)}{\delta x(t_0)} & \frac{\delta v_x(t_f)}{\delta y(t_0)} & \frac{\delta v_x(t_f)}{\delta z(t_0)} & \frac{\delta v_x(t_f)}{\delta v_x(t_0)} & \frac{\delta v_x(t_f)}{\delta v_y(t_0)} & \frac{\delta v_x(t_f)}{\delta v_z(t_0)} \\ \frac{\delta v_y(t_f)}{\delta x(t_0)} & \frac{\delta v_y(t_f)}{\delta y(t_0)} & \frac{\delta v_y(t_f)}{\delta z(t_0)} & \frac{\delta v_y(t_f)}{\delta v_x(t_0)} & \frac{\delta v_y(t_f)}{\delta v_y(t_0)} & \frac{\delta v_y(t_f)}{\delta z(t_0)} \\ \frac{\delta v_z(t_f)}{\delta x(t_0)} & \frac{\delta v_z(t_f)}{\delta y(t_0)} & \frac{\delta v_z(t_f)}{\delta z(t_0)} & \frac{\delta v_z(t_f)}{\delta v_x(t_0)} & \frac{\delta v_z(t_f)}{\delta v_y(t_0)} & \frac{\delta v_z(t_f)}{\delta v_z(t_0)} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\delta \boldsymbol{r}(t_f)}{\delta \boldsymbol{r}(t_0)} & \frac{\delta \boldsymbol{r}(t_f)}{\delta \boldsymbol{v}(t_0)} \\ \frac{\delta \boldsymbol{v}(t_f)}{\delta \boldsymbol{r}(t_0)} & \frac{\delta \boldsymbol{v}(t_f)}{\delta \boldsymbol{v}(t_0)} \end{bmatrix} =$$

State-transition matrix (STM)

If we want to know the required initial velocity $\boldsymbol{v}(t_0)$ to achieve a desired final position $\boldsymbol{r}(t_f)$, then we are only interested in block $\boldsymbol{\phi}_{rv}$:


Trajectory for known initial conditions
$[\boldsymbol{r}(t_f), \boldsymbol{v}(t_f)]$
$\boldsymbol{r}_{target}(t_f)$
Desired trajectory, what is the required initial state?: $[\boldsymbol{r}(t_0), \boldsymbol{v}(t_0) + \Delta\boldsymbol{v}]$
$[\boldsymbol{r}(t_0), \boldsymbol{v}(t_0)]$

$$\begin{array}{cc} \delta\boldsymbol{r}(t_f) & \left\{ \begin{matrix} \\ \\ \\ \end{matrix} \right. \\ \delta\boldsymbol{v}(t_f) & \left\{ \begin{matrix} \\ \\ \\ \end{matrix} \right. \end{array} \begin{bmatrix} \delta x(t_f) \\ \delta y(t_f) \\ \delta z(t_f) \\ \delta v_x(t_f) \\ \delta v_y(t_f) \\ \delta v_z(t_f) \end{bmatrix} = \begin{bmatrix} \dfrac{\delta\boldsymbol{r}(t_f)}{\delta\boldsymbol{r}(t_0)} & \dfrac{\delta\boldsymbol{r}(t_f)}{\delta\boldsymbol{v}(t_0)} \\ \dfrac{\delta\boldsymbol{v}(t_f)}{\delta\boldsymbol{r}(t_0)} & \dfrac{\delta\boldsymbol{v}(t_f)}{\delta\boldsymbol{v}(t_0)} \end{bmatrix} \begin{bmatrix} \delta x(t_0) \\ \delta y(t_0) \\ \delta z(t_0) \\ \delta v_x(t_0) \\ \delta v_y(t_0) \\ \delta v_z(t_0) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}_{rr} & \boldsymbol{\phi}_{rv} \\ \boldsymbol{\phi}_{vr} & \boldsymbol{\phi}_{vv} \end{bmatrix} \begin{bmatrix} \delta x(t_0) \\ \delta y(t_0) \\ \delta z(t_0) \\ \delta v_x(t_0) \\ \delta v_y(t_0) \\ \delta v_z(t_0) \end{bmatrix} \begin{array}{cc} \left. \begin{matrix} \\ \\ \\ \end{matrix} \right\} & \delta\boldsymbol{r}(t_0) \\ \left. \begin{matrix} \\ \\ \\ \end{matrix} \right\} & \delta\boldsymbol{v}(t_0) \end{array}$$

State-transition matrix (STM)

A few interesting properties of the STM:

$$\boldsymbol{\phi}(t_2, t_0) = \boldsymbol{\phi}(t_2, t_{\mathbf{1}}) \cdot \boldsymbol{\phi}(t_1, t_0) \equiv \text{combination of time intervals}$$
$$\boldsymbol{\phi}(t_0, t_{\mathbf{1}}) = \boldsymbol{\phi}(t_1, t_0)^{-1} \equiv \text{backwards propagation of trajectory}$$

Here 0, 1,2 meaning 3 different consecutive general times

Inverting blocks of the STM:

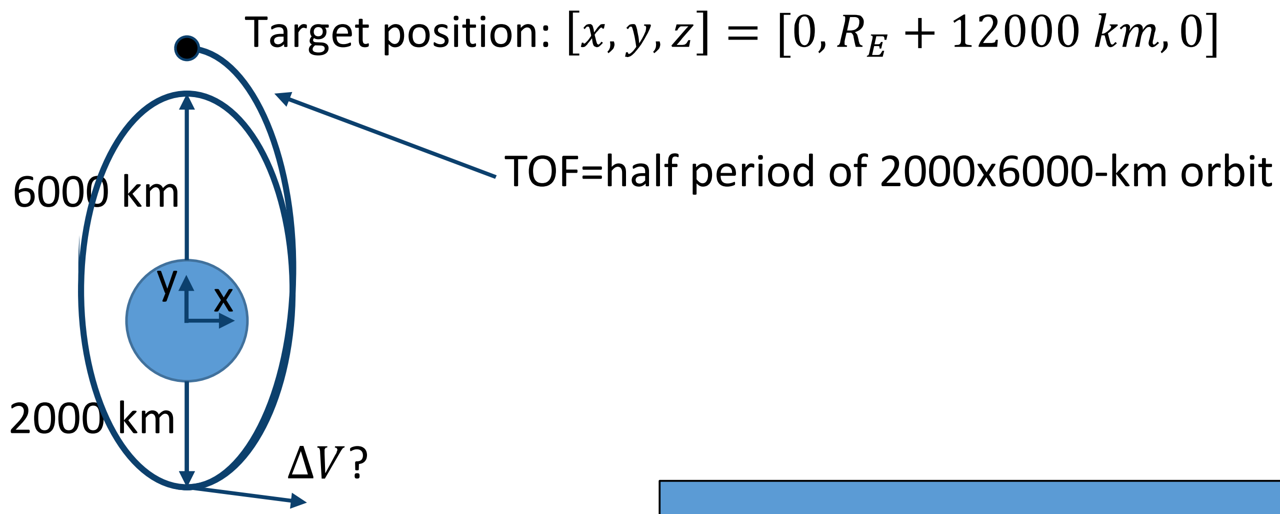
$$\delta \boldsymbol{r}(t_1) = \boldsymbol{\phi}_{rv}(t_1, t_0) \cdot \delta \boldsymbol{v}(t_0) = \boldsymbol{K} \cdot \delta \boldsymbol{v}(t_0)$$

$$\longrightarrow \quad \delta \boldsymbol{v}(t_0) = \boldsymbol{K}^T (\boldsymbol{K}\boldsymbol{K}^T)^{-1} \delta \boldsymbol{r}(t_1) \equiv \text{minimum norm solution}$$
(closest to the reference)

Exercise E2: Differential Correction/Shooting Method

Assume a spacecraft is in an equatorial orbit with perigee altitude of 2000 km and apogee altitude of 6000 km.

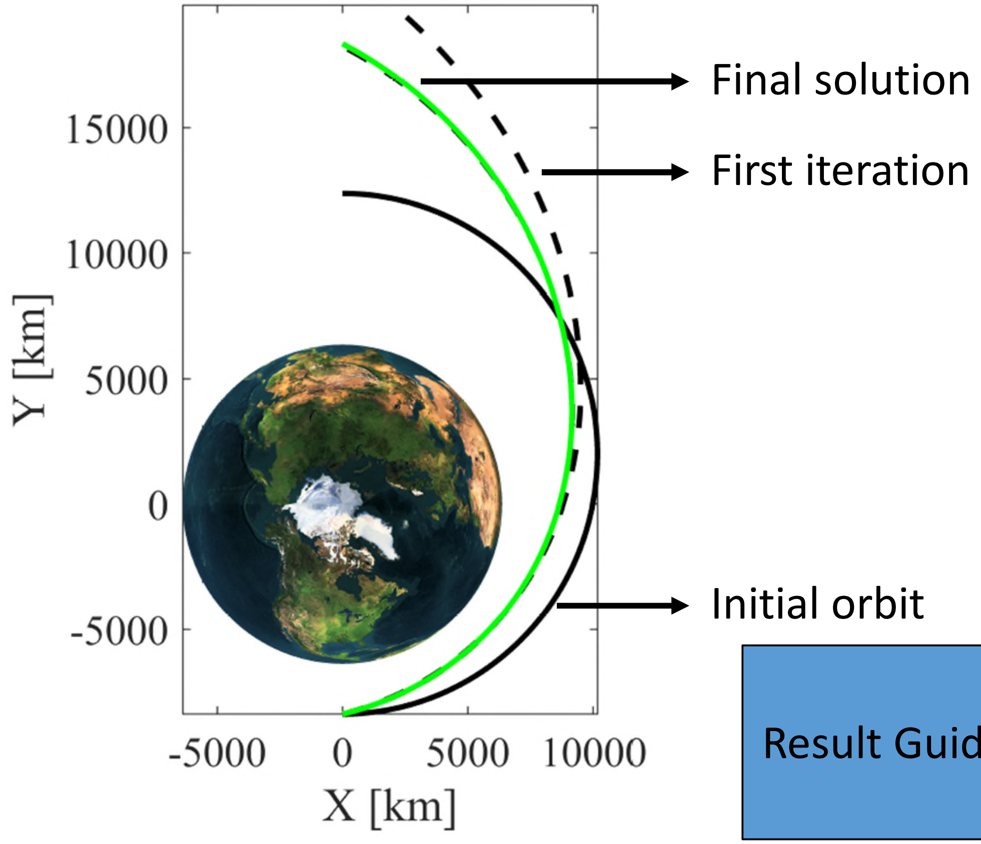
Implement a differential corrector to compute the ΔV required to raise the apogee up to 12000-km altitude in half a period of the 2000x6000-km orbit. Assume the maneuver is performed at the perigee.


Target position: $[x, y, z] = [0, R_E + 12000\ km, 0]$
TOF=half period of 2000x6000-km orbit
6000 km
y
x
2000 km
ΔV?

Result Guide: ΔV=2.011 km/s

Exercise E2: Differential Correction/Shooting Method

Implement a differential corrector to compute the ΔV required to raise the apogee up to 12000-km altitude in half a period of the 2000x6000-km orbit. Assume the maneuver is performed at the perigee.



Result Guide: ΔV=2.011 km/s

Exercise Differential Correction/Shooting Method - Guide (1/2)

1) Initialize problem: provide reference initial conditions $\boldsymbol{x}_{ref}(t_0)$, define desired final state $\boldsymbol{x}_{des}(t_f)$, define time interval $[t_0, t_f]$.

2) Iteration 0:

2a) propagate reference EOM and STM from t_0 to t_f with reference initial conditions.

2b) calculate difference between resulting final state and desired final state: $\delta\boldsymbol{x}(t_f) = \boldsymbol{x}_{des}(t_f) - \boldsymbol{x}_{ref}(t_f)$.

3) Loop: (if $|\delta\boldsymbol{x}(t_f)| > \varepsilon \equiv$ tolerance)

3a) compute change in initial conditions: $\delta\boldsymbol{x}(t_0) = \boldsymbol{\phi}(t_f, t_0)^{-\mathbf{1}} \cdot \delta\boldsymbol{x}(t_f)$.

3b) update reference initial conditions: $\boldsymbol{x}_{ref}^{k+1}(t_0) = \boldsymbol{x}_{ref}^{k}(t_0) + \delta\boldsymbol{x}(t_0)$.

3c) propagate EOM and STM from t_0 to t_f with updated reference initial conditions.

3d) calculate difference between final state of updated reference trajectory and desired final state: $\delta\boldsymbol{x}(t_f) = \boldsymbol{x}_{des}(t_f) - \boldsymbol{x}_{ref}(t_f)$.

- Repeat loop until convergence is achieved or maximum number of iterations is reached (algorithm is not guaranteed to convergence, initial guess matters).

Exercise Differential Correction/Shooting Method - Guide (2/2)

$$a = \frac{1}{2}(r_p + r_a)$$
$$r_p = a(1-e)$$
$$n = \sqrt{\frac{\mu}{a^3}}$$
$$T = \frac{2\pi}{n}$$
$$v(r) = \sqrt{\frac{2\mu}{r} - \frac{\mu}{a}}$$

Two-body problem:

$$\dot{\boldsymbol{r}} = \boldsymbol{v}$$
$$\dot{\boldsymbol{v}} = -\frac{\mu}{|\boldsymbol{r}|^3}\boldsymbol{r}$$

At t_0: $\boldsymbol{\phi}(t_0, t_0)$ = identity matrix

$$\dot{\boldsymbol{\phi}}(t,t_0) = \boldsymbol{A}\left(x_{ref}(t),(t)\right) \cdot \boldsymbol{\phi}(t,t_0)$$

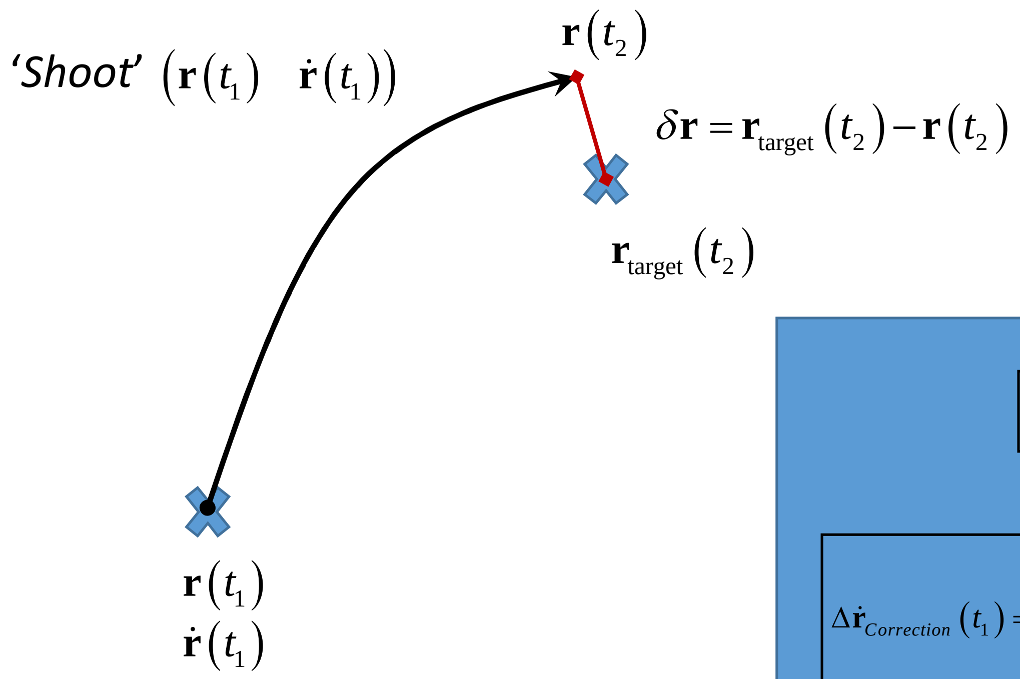
$$\boldsymbol{A}(\boldsymbol{x}(t),t) = \begin{bmatrix} \boldsymbol{0}_{3x3} & \boldsymbol{I}_{3x3} \\ \boldsymbol{A}_{21} & \boldsymbol{0}_{3x3} \end{bmatrix}$$

$$\boldsymbol{A}_{21} = \begin{bmatrix} -\frac{\mu_E}{r^3} + \frac{3\mu_E x^2}{r^5} & \frac{3\mu_E xy}{r^5} & \frac{3\mu_E xz}{r^5} \\ \frac{3\mu_E xy}{r^5} & -\frac{\mu_E}{r^3} + \frac{3\mu_E y^2}{r^5} & \frac{3\mu_E yz}{r^5} \\ \frac{3\mu_E xz}{r^5} & \frac{3\mu_E yz}{r^5} & -\frac{\mu_E}{r^3} + \frac{3\mu_E z^2}{r^5} \end{bmatrix}$$

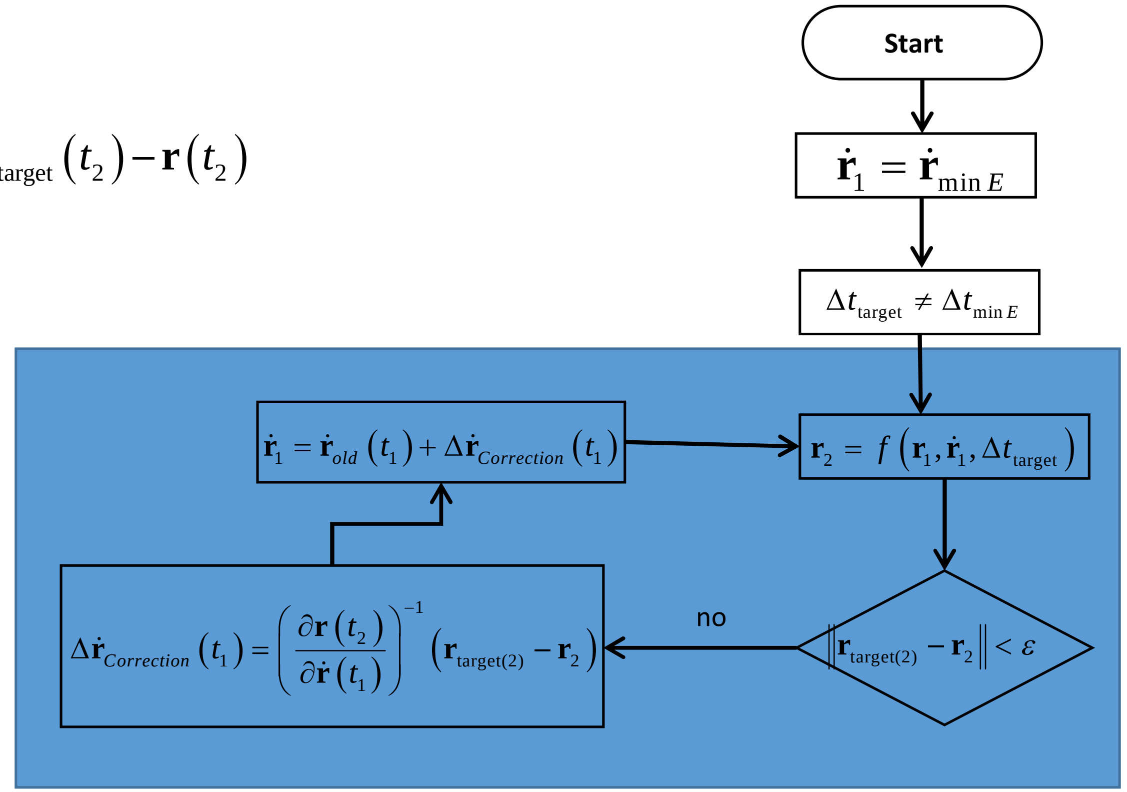
$$\delta\boldsymbol{r}(t_f) = \boldsymbol{\phi}_{rv}(t_f,t_0) \cdot \delta\boldsymbol{v}(t_0) = \boldsymbol{K} \cdot \delta\boldsymbol{v}(t_0)$$

$$\delta\boldsymbol{v}(t_0) = \boldsymbol{K}^T(\boldsymbol{K}\boldsymbol{K}^T)^{-1}\delta\boldsymbol{r}(t_1)$$

Lambert's Problem



'Shoot' $\left(\mathbf{r}(t_1) \quad \dot{\mathbf{r}}(t_1)\right)$
$\mathbf{r}(t_2)$
$\delta\mathbf{r} = \mathbf{r}_{\text{target}}(t_2) - \mathbf{r}(t_2)$
$\mathbf{r}_{\text{target}}(t_2)$
$\mathbf{r}(t_1)$
$\dot{\mathbf{r}}(t_1)$

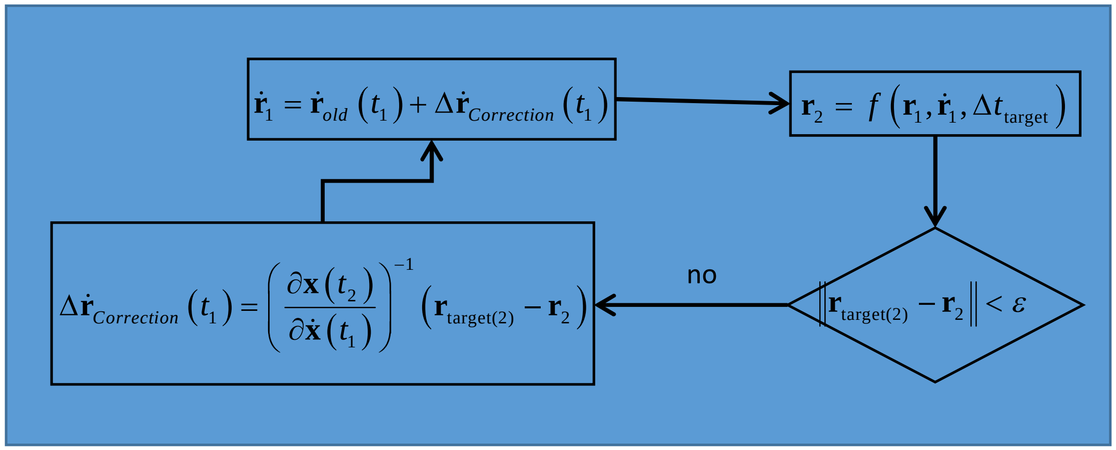


Start
$\dot{\mathbf{r}}_1 = \dot{\mathbf{r}}_{\min E}$
$\Delta t_{\text{target}} \neq \Delta t_{\min E}$
$\mathbf{r}_2 = f\left(\mathbf{r}_1, \dot{\mathbf{r}}_1, \Delta t_{\text{target}}\right)$
$\left\|\mathbf{r}_{\text{target}(2)} - \mathbf{r}_2\right\| < \varepsilon$
no
$\Delta\dot{\mathbf{r}}_{Correction}(t_1) = \left(\frac{\partial \mathbf{r}(t_2)}{\partial \dot{\mathbf{r}}(t_1)}\right)^{-1}\left(\mathbf{r}_{\text{target}(2)} - \mathbf{r}_2\right)$
$\dot{\mathbf{r}}_1 = \dot{\mathbf{r}}_{old}(t_1) + \Delta\dot{\mathbf{r}}_{Correction}(t_1)$

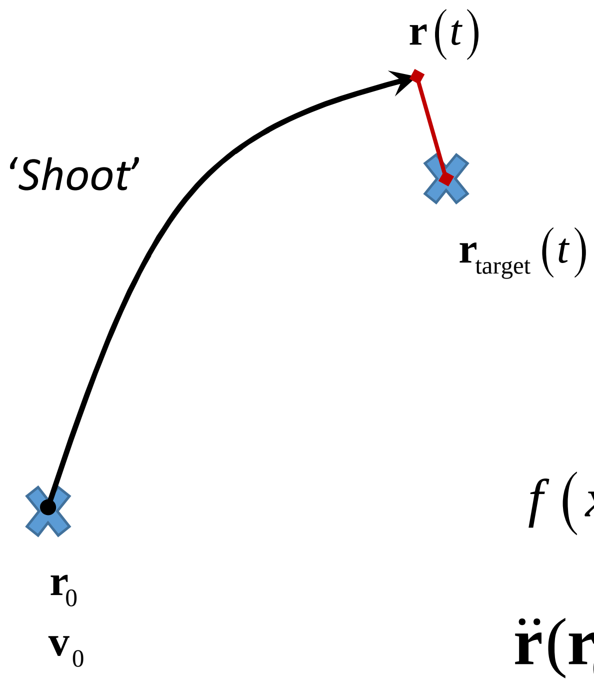
Sensitivity Matrix $\left(\dfrac{\partial \mathbf{r}(t_2)}{\partial \dot{\mathbf{r}}(t_1)}\right)$

$$\delta\dot{\mathbf{r}}_{corr}(t_1) = \left(\frac{\partial \mathbf{r}(t_2)}{\partial \dot{\mathbf{r}}(t_1)}\right)^{-1} \delta\mathbf{r}(t_2)$$

Differential Correction



$\dot{\mathbf{r}}_1 = \dot{\mathbf{r}}_{old}(t_1) + \Delta\dot{\mathbf{r}}_{Correction}(t_1)$
$\mathbf{r}_2 = f(\mathbf{r}_1, \dot{\mathbf{r}}_1, \Delta t_{\text{target}})$
$\left\|\mathbf{r}_{\text{target}(2)} - \mathbf{r}_2\right\| < \varepsilon$
no
$\Delta\dot{\mathbf{r}}_{Correction}(t_1) = \left(\frac{\partial \mathbf{x}(t_2)}{\partial \dot{\mathbf{x}}(t_1)}\right)^{-1}\left(\mathbf{r}_{\text{target}(2)} - \mathbf{r}_2\right)$



$$\ddot{\mathbf{r}}(\mathbf{r}_0, \mathbf{v}_0) = f(\mathbf{r}_0, \mathbf{v}_0) \qquad \ddot{\mathbf{r}} = -\frac{\mu_E}{r^3}\mathbf{r}$$

$$f(x) = f(a) + (x-a) f'(a) + \ldots$$

$$\ddot{\mathbf{r}}(\mathbf{r}_0 + \delta\mathbf{r}, \mathbf{v}_0 + \delta\mathbf{v}) = \ddot{\mathbf{r}}(\mathbf{r}_0, \mathbf{v}_0) + \mathbf{M}\begin{pmatrix}\delta\mathbf{r}\\ \delta\mathbf{v}\end{pmatrix}$$

$$\delta\mathbf{r}(t) = \frac{\partial \mathbf{r}(t)}{\partial \mathbf{v}_0}\delta\mathbf{v}_0 \qquad \left(\frac{\partial \mathbf{r}(t)}{\partial \mathbf{v}_0}\right)' = \mathbf{M}\left(\frac{\partial \mathbf{r}(t)}{\partial \mathbf{v}_0}\right); \left(\frac{\partial \mathbf{r}_0}{\partial \mathbf{v}_0}\right) = \mathbf{0}$$

Exercise 4

Use the STM to correct the previous shooting and come up with a better approximation of the velocity at departure.

When completed, connect to Socrative Room GPQK5UNSK and indicate it so in the relevant question

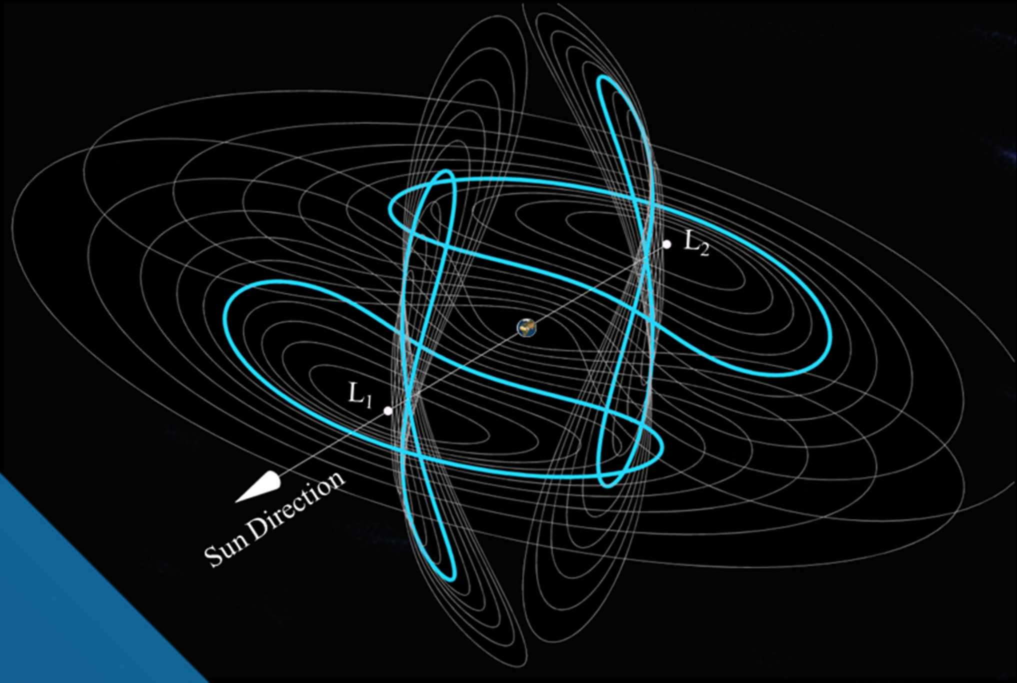
Exercise 5

Iterate the differential correction method until convergence to an error of less than 1m.

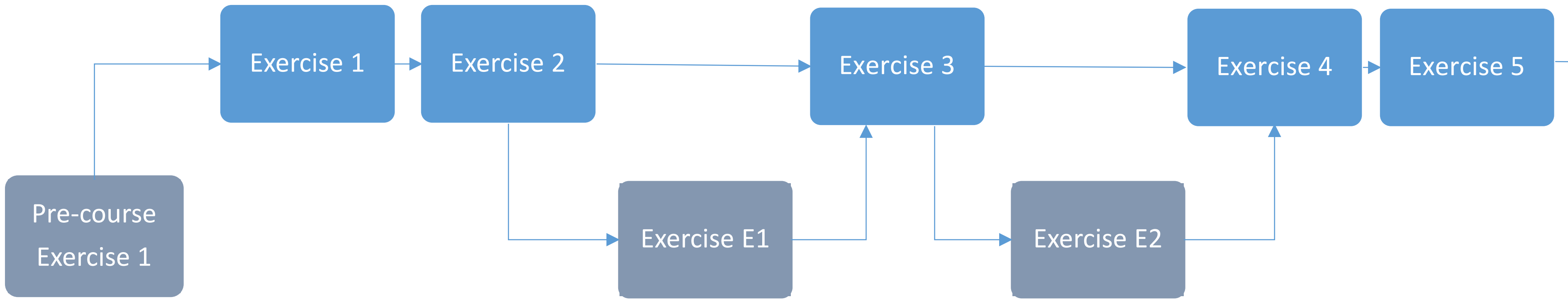
When completed, connect to Socrative Room GPQK5UNSK and indicate it so in the relevant question

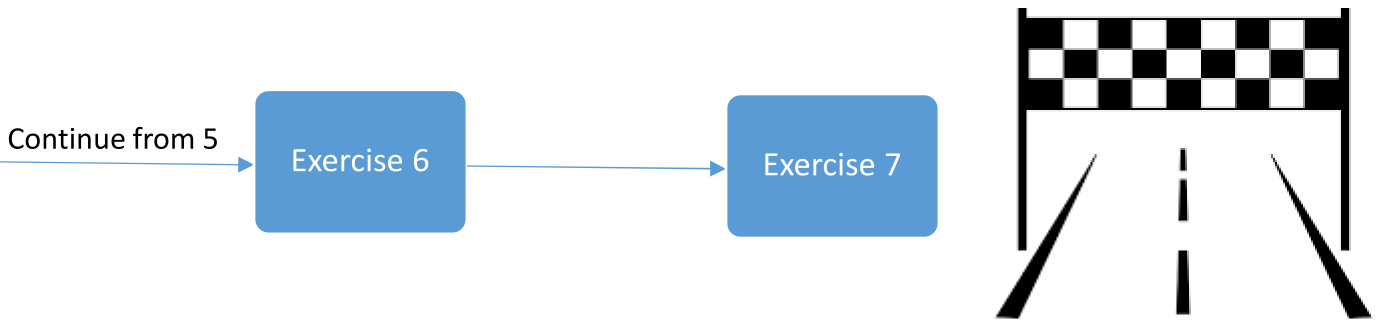
Advanced Topics in Astrodynamics and Trajectory Design

End of day 2



Exercise Path





F & G Solutions for elliptical orbits.

Algorithm *FGKepler_dt*: $\left(\mathbf{r}_0, \dot{\mathbf{r}}_0, \Delta t \Rightarrow \mathbf{r}_f, \dot{\mathbf{r}}_f\right)$

$$a = \frac{\mu}{\left(\frac{2\mu}{r_0} - v_0^2\right)} \qquad n = \sqrt{\frac{\mu}{a^3}}$$

$$\Delta M = n\Delta T$$

$$\sigma_0 = \frac{\mathbf{r}_0 \cdot \dot{\mathbf{r}}_0}{\sqrt{\mu}}$$

Newton-Raphson Solver:

$$\Delta M = \Delta E - \left(1 - \frac{r_0}{a}\right)\sin\Delta E - \frac{\sigma_0}{\sqrt{a}}\left(\cos\Delta E - 1\right) \rightarrow \Delta E$$

* Implement using MATLAB’s Fzero and initialize the search with ΔM value as first guess.

$$F = 1 - \frac{a}{r_0}(1 - \cos\Delta E)$$

$$G = \Delta T + \sqrt{\frac{a^3}{\mu}}\left(\sin\Delta E - \Delta E\right)$$

$$\dot{G} = 1 - \frac{a}{r_f}\left(1 - \cos\Delta E\right)$$

$$\mathbf{r}_f = F\mathbf{r}_0 + G\dot{\mathbf{r}}_0$$

$$\dot{\mathbf{r}}_f = \frac{1}{G}\left(\dot{G}\mathbf{r}_f - \mathbf{r}_0\right)$$

Exercise 6

What DV is required to launch ExoMars Trace Gas Orbiter on 14/03/2016, and arrive at Mars on 15/10/2016.

Result Guide: $\Delta v_{Total} = 7.5 \text{ km/s}$

When completed, connect to Socrative Room GPQK5UNSK and indicate it so in the relevant question

Exercise 6 - extended

Can you compute the DV required for a transfer with departure date on 9/01/2016 and arrival on 19/11/2016?

Continuation method

A continuation method is used when the initial guess is far away from the solution of the problem.

A continuation method is an approach to obtain **intermediate solutions**, **increasingly closer** to the desired solution.

Continuation of parameter P:

$$P_k = \lambda_k \cdot P_{desired} + (1 - \lambda_k) \cdot P_{initial} \quad \begin{cases} \lambda_k = \dfrac{k}{N} \\ k \equiv \text{current iteration} \\ N \equiv \text{total number of iterations} \end{cases}$$

Examples:

- We have a transfer trajectory of a certain TOF, $TOF_{initial}$, and we want a similar trajectory with different TOF, $TOF_{desired}$→we can look for solutions with intermediate values of the TOF, TOF_k.
- We have a transfer trajectory to an orbit of altitude, $h_{initial}$, but we want a transfer to an orbit of altitude $h_{desired}$→we can look for solutions for intermediate values of the orbit altitude, h_k.

Continuation method

The structure of a continuation algorithm is the following:

1) Initialize problem: provide/find initial guess (i.e., solution for an initial value of parameter P, $P_{initial}$), define desired value of parameter P, $P_{desired}$.

2) Loop (until $P_{desired}$ is reached):

    - For $k = 1{:}N$

    2a) compute new value of P: $P_k = \lambda_k \cdot P_{desired} + (1 - \lambda_k) \cdot P_{initial}$; $\lambda_k = \frac{k}{N}$

    2b) solve problem using as initial guess the solution obtained for P_{k-1}.

- A continuation algorithm is not guaranteed to converge. The more iterations ($N \uparrow$), the more likely a new solution can be obtained, but also the higher the computational demand.

Continuation Method

What if your first guess “shoots” very far from the target final state?

- We can apply a so-called continuation method to improve robustness of the algorithm.

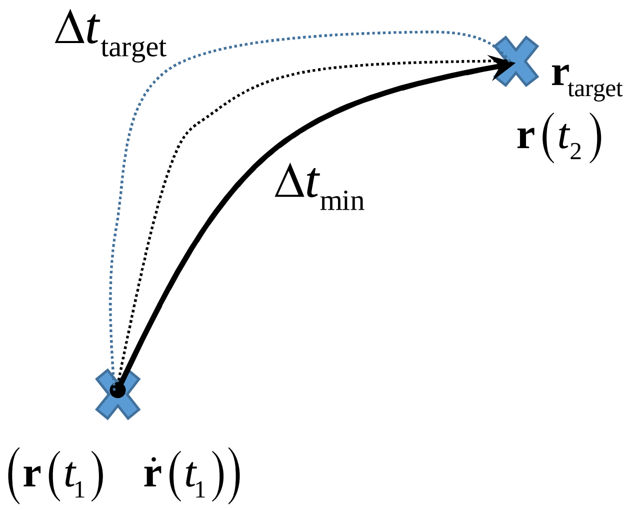
A differential corrector for:

$$\mathbf{x}_{\text{target}}(t_2) = f\left(t_2 - t_1, \mathbf{x}(t_1) + \delta\mathbf{x}_{correction}\right) \text{ may fail if } \left\|\mathbf{x}_{\text{target}}(t_2) - f\left(t_2 - t_1, \mathbf{x}(t_1)\right)\right\| \gg 0$$

However, remember that:

$$\mathbf{x}_{\text{target}}(t_*) = f\left(t_* - t_1, \mathbf{x}(t_1)\right)$$

$$t_* - t_1 = \Delta t_{\text{min}}$$



Δt_{target}
$\mathbf{r}_{\text{target}}$
$\mathbf{r}(t_2)$
Δt_{min}
$(\mathbf{r}(t_1) \quad \dot{\mathbf{r}}(t_1))$

Continuation Method

What if your first guess "shoots" very far from the target final state?

- We can apply a so-called continuation method to improve robustness of the algorithm.

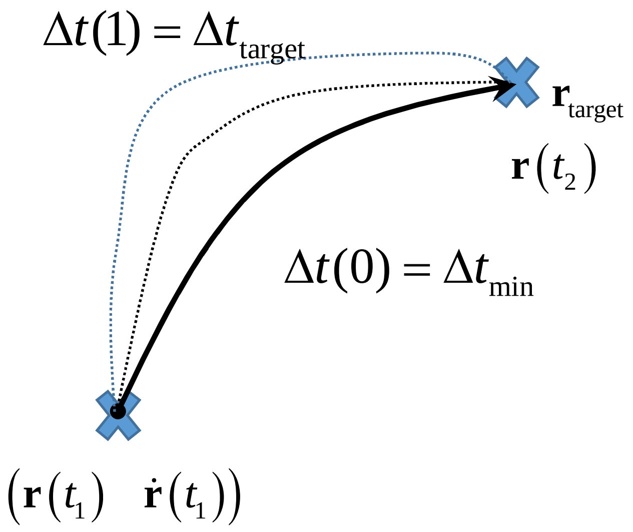
A differential corrector for:

$$\mathbf{x}_{\text{target}}(t_2) = f\left(t_2 - t_1, \mathbf{x}(t_1) + \delta\mathbf{x}_{correction}\right) \text{ may fail if } \left\|\mathbf{x}_{\text{target}}(t_2) - f\left(t_2 - t_1, \mathbf{x}(t_1)\right)\right\| \gg 0$$

$$\delta\mathbf{x}^*_{correction} := \mathbf{x}_{\text{target}}(t_1 + \Delta t) = f\left(\Delta t, \mathbf{x}(t_1) + \delta\mathbf{x}^*_{correction}\right)$$

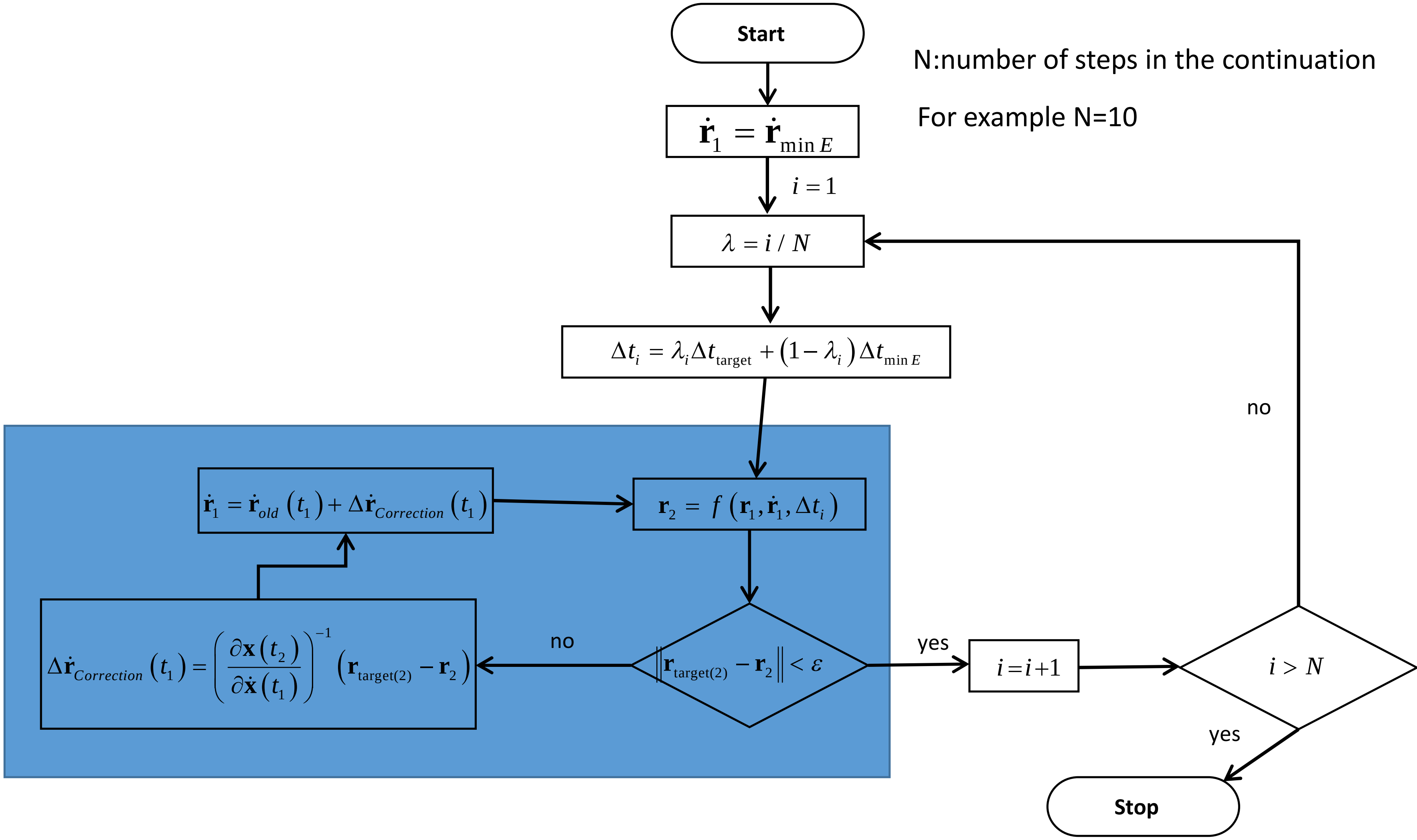
$$\Delta t := \left\|\mathbf{x}_{\text{target}}(t_1 + \Delta t) - f\left(\Delta t, \mathbf{x}(t_1)\right)\right\| \sim 0$$

$$\Delta t(\lambda) = \lambda \cdot \Delta t_{\text{target}} + (1-\lambda) \cdot \Delta t_{\min}$$


$\Delta t(1) = \Delta t_{\text{target}}$
$\mathbf{r}_{\text{target}}$
$\mathbf{r}(t_2)$
$\Delta t(0) = \Delta t_{\min}$
$(\mathbf{r}(t_1) \quad \dot{\mathbf{r}}(t_1))$

- We can then perform a continuation λ=[0,1]

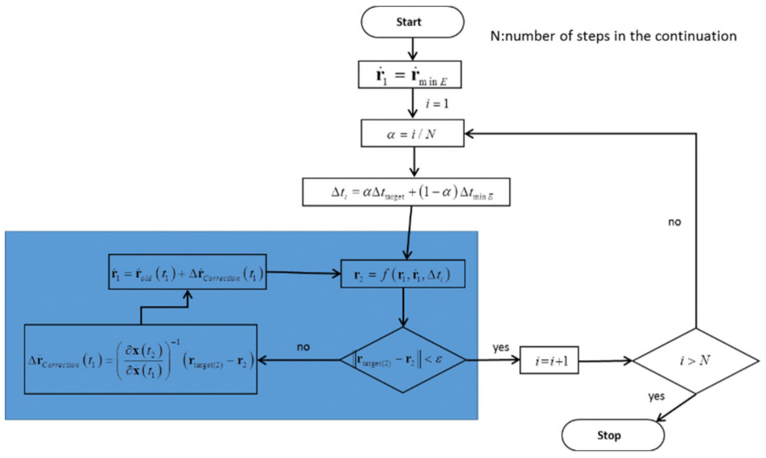
Lambert’s Problem – Continuation method


Start
$\dot{\mathbf{r}}_1 = \dot{\mathbf{r}}_{\min E}$
$i = 1$
$\lambda = i / N$
$\Delta t_i = \lambda_i \Delta t_{\text{target}} + (1-\lambda_i)\Delta t_{\min E}$
$\mathbf{r}_2 = f(\mathbf{r}_1, \dot{\mathbf{r}}_1, \Delta t_i)$
$\left\| \mathbf{r}_{\text{target}(2)} - \mathbf{r}_2 \right\| < \varepsilon$
no
$\Delta\dot{\mathbf{r}}_{Correction}(t_1) = \left(\frac{\partial \mathbf{x}(t_2)}{\partial \dot{\mathbf{x}}(t_1)}\right)^{-1} (\mathbf{r}_{\text{target}(2)} - \mathbf{r}_2)$
$\dot{\mathbf{r}}_1 = \dot{\mathbf{r}}_{old}(t_1) + \Delta\dot{\mathbf{r}}_{Correction}(t_1)$
yes
$i = i + 1$
$i > N$
no
yes
Stop

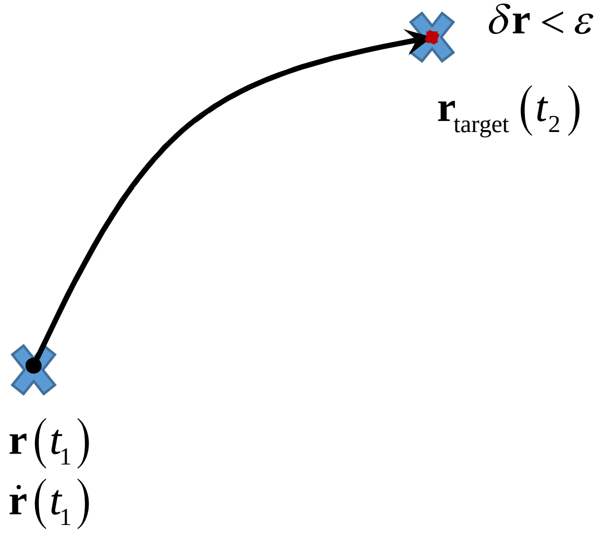
N:number of steps in the continuation

For example N=10

Lambert's Problem - reCap


Start
N:number of steps in the continuation
$\dot{\mathbf{r}}_1 = \dot{\mathbf{r}}_{\min E}$
$i = 1$
$\alpha = i / N$
$\Delta t_i = \alpha \Delta t_{\text{target}} + (1-\alpha)\Delta t_{\min E}$
$\dot{\mathbf{r}}_1 = \dot{\mathbf{r}}_{old}(t_1) + \Delta\dot{\mathbf{r}}_{Correction}(t_1)$
$\mathbf{r}_2 = f(\mathbf{r}_1, \dot{\mathbf{r}}_1, \Delta t_i)$
$\Delta\dot{\mathbf{r}}_{Correction}(t_1) = \left(\frac{\partial \mathbf{x}(t_2)}{\partial \mathbf{x}(t_1)}\right)^{-1} \left(\mathbf{r}_{\text{target}(2)} - \mathbf{r}_2\right)$
no
$\left\|\mathbf{r}_{\text{target}(2)} - \mathbf{r}_2\right\| < \varepsilon$
yes
$i = i+1$
$i > N$
no
yes
Stop

'Shoot' $\left(\mathbf{r}(t_1) \quad \dot{\mathbf{r}}(t_1)\right)$


$\delta\mathbf{r} < \varepsilon$
$\mathbf{r}_{\text{target}}(t_2)$
$\mathbf{r}(t_1)$
$\dot{\mathbf{r}}(t_1)$

1

$$\mathbf{r}(t_1) = \mathbf{r}_{Earth}(t_1)$$

$$\Delta v_1 = \left|\dot{\mathbf{r}}(t_1) - \dot{\mathbf{r}}_{Earth}(t_1)\right|$$

2

$$\mathbf{r}(t_2) = \mathbf{r}_{Mars}(t_2)$$

$$\dot{\mathbf{r}}(t_2)?$$

$$\Delta v_2 = \left|\dot{\mathbf{r}}(t_2) - \dot{\mathbf{r}}_{Mars}(t_1)\right|$$

$$\dot{G} = 1 - \frac{a}{r_f}\left(1 - \cos \Delta E\right)$$

$$\dot{\mathbf{r}}_f = \frac{1}{G}\left(\dot{G}\mathbf{r}_f - \mathbf{r}_0\right)$$

Algorithm *LambertArc_ATD2021* $\left(\mathbf{r}_1, \mathbf{r}_2, \Delta t_{\text{target}}, t_m \Rightarrow \dot{\mathbf{r}}_1, \dot{\mathbf{r}}_2\right)$

```
*MinETransfer* $\left(\mathbf{r}_1, \mathbf{r}_2, t_m \Rightarrow a_{\min}, \Delta t_{\min}, \dot{\mathbf{r}}_1\right)$

For *iterations 1 to N*
        $\lambda = \frac{iteration}{N}$
        $\Delta t(\lambda) = \lambda \cdot \Delta t_{\text{target}} + (1-\lambda) \cdot \Delta t_{\min}$
        While $\|\boldsymbol{\varepsilon}\| > 10^{-3}$ km

            *FGKepler_dt* $\left(\mathbf{r}_1, \dot{\mathbf{r}}_1, \Delta t(\lambda) \Rightarrow \mathbf{r}_2^*\right)$

            *STM_Lambert* $\left(\mathbf{r}_1, \dot{\mathbf{r}}_1, \Delta t(\lambda) \Rightarrow \Phi(t_0 + \Delta t, t_0)\right)$
            $\boldsymbol{\varepsilon} = \left(\mathbf{r}_2 - \mathbf{r}_2^*\right)$

            $\delta \mathbf{r}^*_{cor} = \left(\Phi(t_0 + \Delta t, t_0)\right)^{-1} \boldsymbol{\varepsilon}$

            $\dot{\mathbf{r}}_1 = \dot{\mathbf{r}}_1 + \delta \mathbf{r}^*_{cor}$

        End while

End for

*FGKepler_dt* $\left(\mathbf{r}_1, \dot{\mathbf{r}}_1, \Delta t_{\text{target}} \Rightarrow \mathbf{r}_2, \dot{\mathbf{r}}_2\right)$
```

Exercise 7

Complete Exercise 6, but this time using *LambertArc_ATATD2019* function

Cranfield
Aerospace

www.cranfield.ac.uk

T: +44 (0)1234 750111

@cranfielduni

@cranfielduni

/cranfielduni