AE4878 - Mission Geometry and Orbit Design
Homework Assignment 9 - Exponential Sinusoids

Time Spent: 15 hr

Ali Nawaz

March 27, 2018

1 Introduction

The objective of this report is to construct exponential sinusoidals that facilitate in designing low thrust, gravity assisted trajectories(esp. interplanetary). First a brief overview of the exponential sinusoidals is provided followed by simplifications of the orbital transfer problem in a Lambert's targeting problem.

Exponential sinusoidal

Example of an exponential sinusoidal is presented with the aid of Figure1[4]. Mathematically an exponential sinusoid is defined with the aid of Equation 1 [3][Slide 21/77]. If k_1 is set to zero a logarithmic spiral is obtained. While setting q to 0 gives a pure exponential sinusoid.

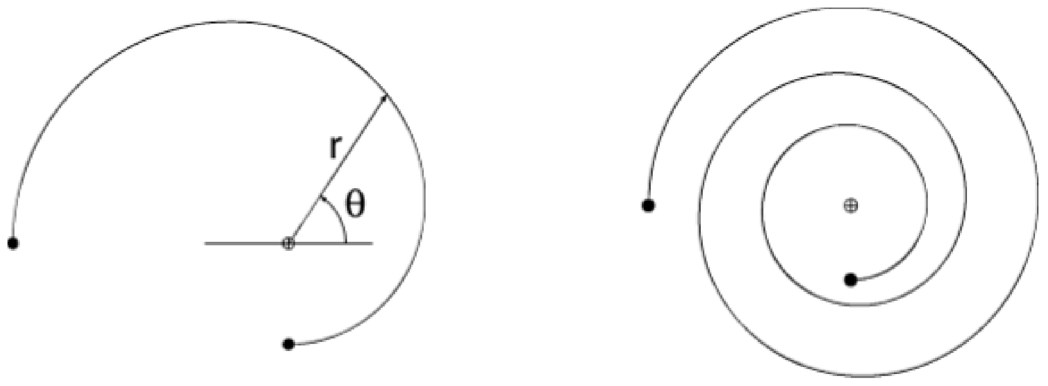

Figure 1: Example of exponential sinusoidal.

$$
\begin{aligned}
& r = k_0 \exp(q\theta + k_1 \sin(k_2\theta + \phi)) \\
& \text{where, r = radial distance [m]} \\
& k_0 = \text{scaling factor [m]} \\
& k_1 = \text{dynamic range parameter [-]} \\
& k_2 = \text{winding parameter [-]} \\
& q = \text{constant } [rad^{-1}] \\
& \theta = \text{true anomaly [rad]} \\
& \phi = \text{phase angle [rad]}
\end{aligned} \tag{1}
$$

Equations of motion can of the problem under consideration is outlined in Figure2[3][Slide28/77]. With the aid of Equation 1 this can be simplified as shown in the set of equations presented in Figure 3[3][Slide 29/77].

$$\begin{aligned}\ddot{r} - r\dot{\theta}^2 + \mu / r^2 &= a_T \sin\alpha \\ \frac{1}{r}\frac{d}{dt}\left(r^2\,\dot{\theta}\right) &= a_T \cos\alpha \\ (or: \quad \ddot{\theta} r + 2\dot{\theta}\dot{r} &= a_T \cos\alpha\,)\end{aligned}$$

Figure 2: Equations of motion for a given orbital trajectory.

$$\begin{aligned} r &= k_0 \exp(q\theta + k_1 \sin(k_2\theta + \varphi)) \\ \text{define:} & \\ c &= \cos(k_2\,\theta + \varphi) \\ s &= \sin(k_2\,\theta + \varphi) \\ \text{then} & \\ r &= k_0 \exp(q\theta + k_1 s) \\ \dot{r} &= r\,\dot{\theta}\,(q + k_1 k_2\, c) \\ \ddot{r} &= r[\,\ddot{\theta}\,(q + k_1 k_2 c) + \dot{\theta}^2\,(q + k_1 k_2 c)^2 - \dot{\theta}^2 k_1 k_2^{\,2} s\,]\end{aligned}$$

Figure 3: Equations of motion after inclusion of exponential sinusoidal expression.

In the above expression, the objective is to fit the dynamics of the orbital trajectory to kinematics description. This is inverse of the classical approach taken in celestial mechanics. The object here lies in finding $\dot{\theta}$, $\ddot{\theta}$, α and a_T. a_T represents the acceleration and α it's direction. Now that a general overview of the exponential sinusoidal is outlined and the corresponding equations of motions are described, Lambert targeting problem to facilitate orbital propagation is described in the following section.

Problem definition

Before explaining the Lambert targeting problem a few assumptions are made to simplify the above expression. Parameter q is set to 0, this turns the problem into a pure sinusoidal problem. Tangential thrusting is assumed, i.e. $\alpha = n \cdot \pi$, n = 0 for acceleration and 1 for deceleration[3][Slide 35/77]. Making these assumptions help decouple the expressions for $\dot{\theta}$ and α_T. The underlying expressions are outlined in Figure 4[3][Slide 37/77].

uncoupled

$$\dot{\theta}^2 = \left(\frac{\mu}{r^3}\right)\frac{1}{\tan^2\gamma + k_1 k_2^2 s + 1}$$

$$\begin{aligned} a_N &= \frac{(-1)^n}{2\cos\gamma}\left[\frac{\tan\gamma}{\tan^2\gamma + k_1 k_2^2 s + 1} - \frac{k_1 k_2^3 c - 2k_1 k_2^2 s \tan\gamma}{\left(\tan^2\gamma + k_1 k_2^2 s + 1\right)^2}\right] \\ &= \frac{(-1)^n \tan\gamma}{2\cos\gamma}\left[\frac{1}{\tan^2\gamma + k_1 k_2^2 s + 1} - \frac{k_2^2(1 - 2k_1 s)}{\left(\tan^2\gamma + k_1 k_2^2 s + 1\right)^2}\right]\end{aligned}$$

$$\begin{aligned} TOF &= \int_{t_0}^{t_f} dt = \int_{\theta_0}^{\theta_f} \frac{dt}{d\theta}\,d\theta = \int_{\theta_0}^{\theta_f} \sqrt{r^3(\tan^2\gamma + k_1 k_2^2 s + 1)/\mu}\; d\theta \\ &\approx \sum_{i=1}^{i_{\max}} \sqrt{r_i^3(\tan^2\gamma_i + k_1 k_2^2 s_i + 1)/\mu}\;\Delta\theta_i\end{aligned}$$

Figure 4: Characteristic exposinusoidal expressions with tangential thrusting.

For the problem under consideration the position of the departure planet r_1 and the target planet r_2 from the heliocentric co-ordinates is known. Maximum limitation on the required time of flight along with the number of *revolutions to target destination N is provided. Furthermore, the angle between r_1 and r_2 is known.*

An approach similar to Lambert targeting problem, outlined by Dario Izzo [2] can be applied. For the approach defined by Izzo, k_2 and N is assumed. Parameters r_1, r_2, Ψ, n and max. time of flight (TOF) is known. Using this information the first objective is to limit the flight path angle search space, i.e. $\gamma_{1,min}$ and $\gamma_{1,max}$. This is done with the aid of constraints outlined in Equation 2[3][Slide 39/77].

1. Constraint equation for tangential thrusting$(c^2 + s^2 = 1)$

$$k_1^2 k_2^4 - k_2^2(\tan(\gamma)^2) - k_{12s}^2 = 0$$

where, $k_{12s} = k_1 k_2^2 s$ (2)

2. Requirement for physically feasible solutions for tangential thrusting only (pericenter: $s = -1, c = 0$ and$\gamma = 0)$

$$|k_1 k_2^2| < 1$$

To realise the above expression in limiting the search space the definition outlined in Equation 3 is used[3][Slide 58/77].

$$\begin{aligned}
\bar{\theta} &= \Psi + 2\pi \cdot N \\
\Delta &= \frac{2(1 - \cos(k_2 \cdot \bar{\theta}))}{k_2^4} - ln^2\left(\frac{r_1}{r_2}\right) \\
\tan(\gamma_{1,min}) &= \frac{k_2}{2}\left[-ln\left(\frac{r_1}{r_2}\right)\cot\left(\frac{k_2\bar{\theta}}{2}\right) - \sqrt{\Delta}\right] \\
\tan(\gamma_{1,max}) &= \frac{k_2}{2}\left[-ln\left(\frac{r_1}{r_2}\right)\cot\left(\frac{k_2\bar{\theta}}{2}\right) + \sqrt{\Delta}\right]
\end{aligned} \tag{3}$$

Once the search space is reduced, initial γ_1 parameters is selected from the search space. For selected γ_1, the following parameters are derived: k_1, ϕ, k_0 and the TOF. Parameters k_1, k_0 and ϕ are derived as shown in Figures 5 and 6 [3][Slide 56,57].

magnitude of k_1 :

$$k_1^2 = \left(\frac{\ln\left(\frac{r_1}{r_2}\right) + \frac{\tan(\gamma_1)}{k_2}\sin(k_2\bar{\theta})}{1 - \cos(k_2\bar{\theta})}\right)^2 + \frac{\tan^2(\gamma_1)}{k_2^2}$$

sign of k_1 :

$$sign(k_1) = sign\left[\ln\left(\frac{r_1}{r_2}\right) + \frac{\tan(\gamma_1)}{k_2}\sin(k_2\bar{\theta})\right]$$

Figure 5: Expressions outlining the definition of k_1.

also

$$\varphi = \text{acos}\left(\frac{\tan(\gamma_1)}{k_1 k_2}\right)$$

and

$$k_0 = \frac{r_1}{\exp(k_1 \sin(\varphi))}$$

Figure 6: Expressions outlining the definition of k_0 and ϕ.

The expression for TOF is presented with the aid of Figure 7[3][Slide 65/77].

$$\begin{aligned}
TOF &= \int_{t_0}^{t_f} dt = \int_{\theta_0}^{\theta_f} \frac{dt}{d\theta} d\theta = \int_{\theta_0}^{\theta_f} \sqrt{r^3(\tan^2\gamma + k_1 k_2^2 s + 1)/\mu}\, d\theta \\
&\approx \sum_{i=1}^{i_{max}} \sqrt{r_i^3(\tan^2\gamma_i + k_1 k_2^2 s_i + 1)/\mu}\, \Delta\theta_i
\end{aligned}$$

Figure 7: Expression defining the time of flight for chosen parameters.

A quick overview of the approach undertaken in putting together the above expressions to obtain the required TOF for varying γ_1 parameters is outlined in the next section.

2 Solution overview

Figure 8 outlines a quick overview of the algorithm constructed to find TOF for given γ_1. The underlying scripts are presented in Section 5. Main file only calls the function *exposin_lambert( )*. The function *exposin_lambert( )* is used obtain TOF solution for defined parameters.The function *exposin_lambert( )* calls the function *TOF( )* or *TOFsimpsons( )* to obtain the integrated TOF.

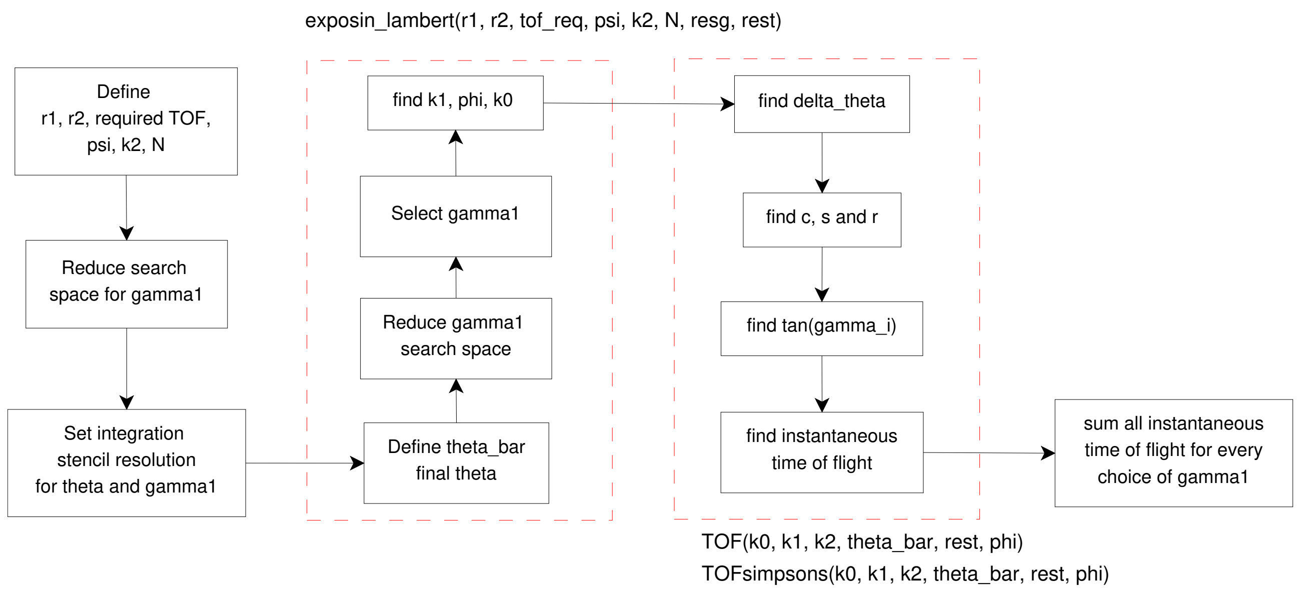
exposin_lambert(r1, r2, tof_req, psi, k2, N, resg, rest)
Define r1, r2, required TOF, psi, k2, N
Reduce search space for gamma1
Set integration stencil resolution for theta and gamma1
find k1, phi, k0
Select gamma1
Reduce gamma1 search space
Define theta_bar final theta
find delta_theta
find c, s and r
find tan(gamma_i)
find instantaneous time of flight
sum all instantaneous time of flight for every choice of gamma1
TOF(k0, k1, k2, theta_bar, rest, phi)
TOFsimpsons(k0, k1, k2, theta_bar, rest, phi)

Figure 8: Overview of the algorithm constructed to solve exposinusoidal with Lambert targeting approach.

Even though recursive adaptive algorithm are efficient in terms of solution accuracy and computational time both simple integration scheme and Simpson's composite integration scheme are used and later compared. To evaluate the simple integration scheme, the discretized integration scheme shown in Figure7 is used. However for Simpson's rule, the equation outlined in Figure9[1]. Algorithm outlining TOF with regular integration and composite Simpson's integration is outlined in Section 5.

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(x_0) + 2\sum_{j=1}^{n/2-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(x_n)\right],$$

Figure 9: Expression underlying the composite Simpson's rule.

3 Benchmark results

Now that the algorithms are outlined, first a benchmark test is carried out to evaluate the performance of the algorithms. The benchmark is first conducted for $\gamma_{1,min}$ and $\gamma_{1,max}$. Later the benchmark is extended to TOF followed by visual comparison. Integration is not still used while estimating $\gamma_{1,min}$ and $\gamma_{1,max}$. The results of $\gamma_{1,min}$ and $\gamma_{1,max}$ are verified against the results presented in [2]. Table 1 outlines the verification result. The results are verified and the algorithm is proven to be in line with the results in [2].

Table 1: Verification of $\gamma_{1,min}$ and $\gamma_{1,max}$.

N	$\gamma_{1,min}$ [2] [rad]	$\gamma_{1,max}$ [2] [rad]	$\gamma_{1,min}$ using script [rad]	$\gamma_{1,max}$ using script [rad]
0	-0.4849	0.8061	-0.4849	0.80616
1	-1.3139	1.3202	-1.3140	1.3202
2	-1.4213	1.4225	-1.42135	1.4225
3	-1.4602	1.4606	-1.4602	1.4606

In the second phase of benchmarking is conducted for the TOF. It is known that TOF is very sensitive to the underlying segmentation and the integration scheme. Thus visual verification is very crucial to evaluate the effects. All the algorithm presented is run for the following settings: r_1=1AU, r_2=1.5 AU, $k_2=\frac{1}{12}$, Ψ = 90 degrees, N = 2 and γ_1 =-80 degrees. The results are verified against the results outlined in [3][Slides- 66/77, 68/77 and 70/77]. From the Table 2 it can be observed that the results are quite sensitive the integration schemes and the resolution of discretization. [3] indicates that the results obtained via the Simpson's rule are accurate for higher discretization steps. It observed that for increasing steps the results converge closer and closer to similar values. To provide a visual verification Figures10-11 provide TOF estimated with the aid of composite Simpson's rule against γ_1 for N= 0 to 3, and no. of steps= 23000 and 100. While Figures 12 and 14 outline the same results for simple integration scheme. To provide

Table 2: Verification of different integration schemes against the schemes outlined in literature.

No. of Steps	**TOF [yrs] simple integration begin of each subinterval**	**TOF [yrs] simple integration center of subinterval**	**TOF [yrs] composite Simpson's rule**	***Scripted TOF[yrs] Simple integration begin of each subinterval***	***Scripted TOF [yrs] composite Simpson's rule***
10	1.28179	0.01031	1.21919	2.38380	1.223302
100	0.28415	0.28450	0.30564	0.39489	0.221678
1000	0.29729	0.30220	0.30238	0.30928	0.281609
2000	0.29974	0.30234	0.30238	0.30626	0.291946
10000	0.30184	0.30238	0.30238	0.30560	0.301030
23000	-	-	-	0.30366	0.302375

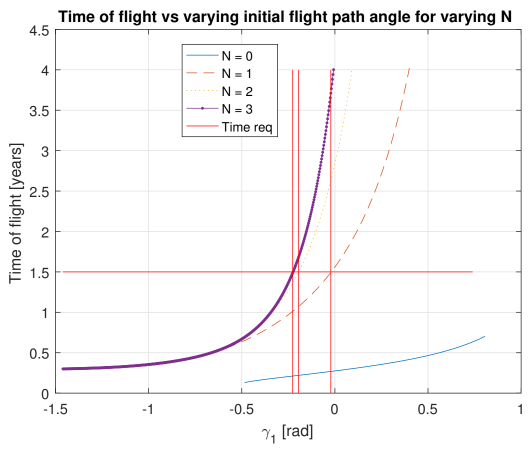

Figure 10: TOF with composite Simpsons for all N and no. of steps = 23000.

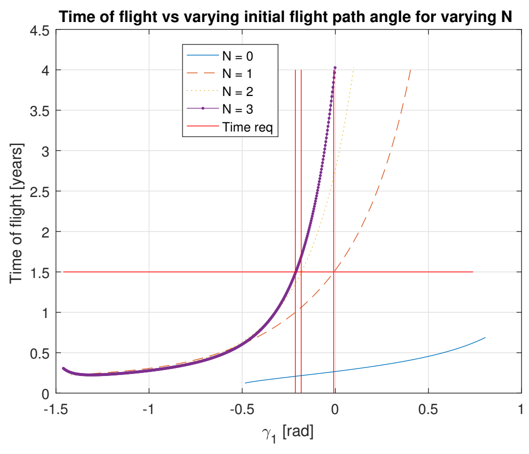

Figure 11: TOF with composite Simpsons for all N and no. of steps = 100.

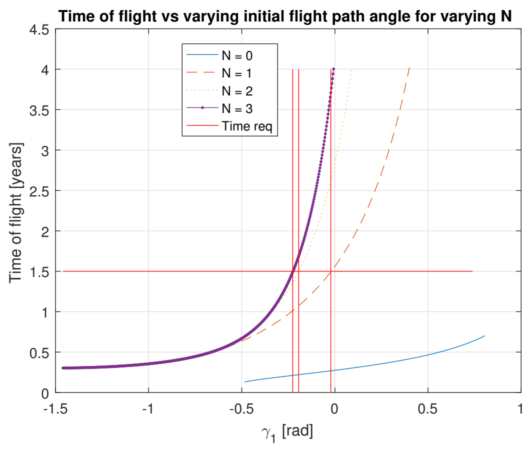

Figure 12: TOF with simple integration for all N and no. of steps = 23000.

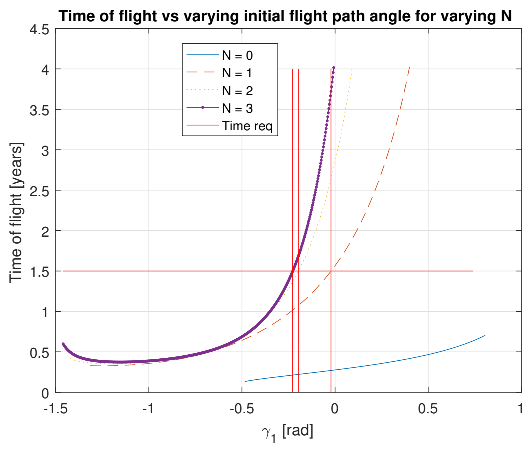

Figure 13: TOF with simple integration for all N and no. of steps = 100.

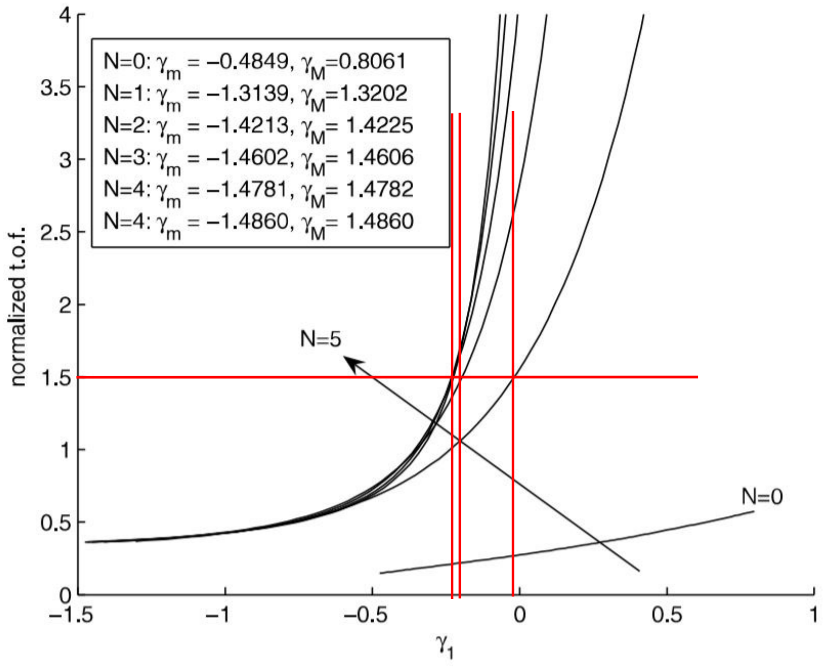

Fig. 2 Normalized time of flight versus the initial flight-path angle curves for the class $\mathcal{S}_{1/12}[1, 1.5, \pi/2, N]$.

Figure 14: TOF against gamma as presented in [2]

Visually it can be observed that both simple and composite Simpson's integration perform fairly well for the given problem. However, composite Simpson's integration is chosen for later estimations. Since it performed better for lower number of steps. The constructed algorithm is verified to perform as expected and can now be extended to solve the problem at hand.

4 Result evaluation and discussion

The algorithm constructed thus far is now extended to reproduce the time of flight curve for $S_{\frac{1}{12}}$ [1, 1.5, $\frac{\pi}{2}$, N = 0,1,2,3]. Where, $k_2 = \frac{1}{12}$, $\Psi = \frac{\pi}{2}$. TOF curve for N=0,1,2,3 is presented with the aid of Figure 15. Now the required flight time is updated to 2 years. Note that a year is estimated with the aid of sidereal days. 1 sidereal day = 86164.1004 seconds [5], while one year is taken as 365 days.

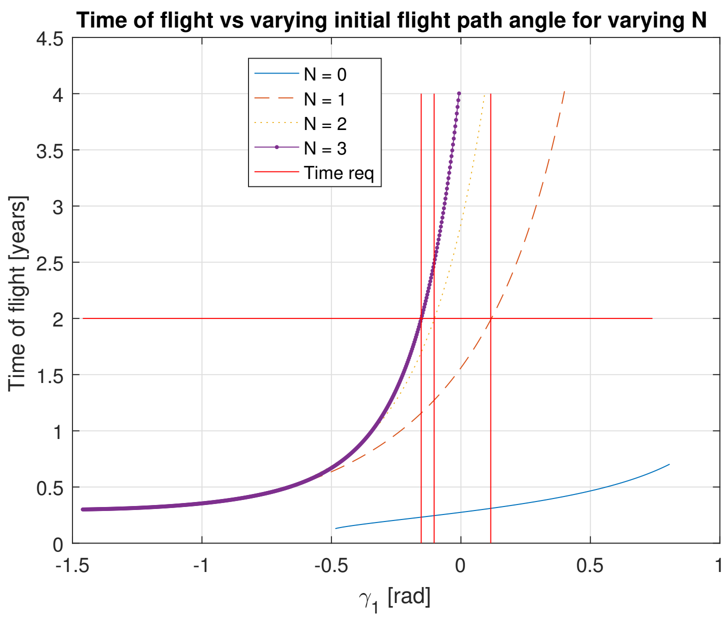

Figure 15: TOF against varying γ_1.

To conclude this section, the possible flight path choices (transfer options) of γ_1 for the given requirement of less than or equal to 2 years of flight time is outlined in Table 3. With this report and the algorithms presented, it is shown that Lambert solutions to exponential sinusoids can be a quick and lucrative of estimating low thrust gravity assisted interplanetary transfers.

Table 3: Ranges of γ_1 values feasible for the time of flight requirement of less than or equal to 2.0 years.

N	$\gamma_{1,min}$ [rad]	$\gamma_{1,max}$ [rad]
0	-0.4849	0.8062
1	-1.3140	0.1152
2	-1.4214	-0.1033
3	-1.4602	-0.1533

5 Matlab Scipt

All files can be found at the following Git source:
`https://github.com/Alixir/Mission-geometry-and-orbit-design`
Following is the main run file:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% AE4878 Mission Geometry and Orbit Design %%%%%%%%%%%
%%% Assignment 9 - Week 3.6 - EXPOSINS - 4, VERSION 3 %%%%
%%%%% Author Info: Ali Nawaz; Student ID - 4276477 %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all; clear all; clc;
%% Part 1 : Reproduction of time-of-flight curve S_1/12 [1,1.5,pi/2,0]

% Known parameter definitions
r1 = 1; % initial position [AU]
r2 = 1.5; % final position [AU]
tof_reqy = 4.0; % Max required time of flight [years]
psi = pi/2; % Angle between initial position vector r1 and final position vector r2 [rad]
k2 = 1/12; % k2 is a tuning parameter to tune the number of revolutions
resg = 1000; % split up the gamma range in resg intervals for integration
% rest = 23000; % split up the theta range in rest intervals for integration
% resg = 1; % split up the gamma range in resg intervals for integration
% rest = 100; % split up the theta range in rest intervals for integration
rest = 23000;
% N = 0
```

```
N = 0; % N is the number of revolutions
[gamma1_list0,tof0] = exposin_lambert(r1,r2,tof_reqy,psi,k2,N,resg,rest);
%% Part 2 Reproduction of time-of-flight curve S_1/12 [1,1.5,pi/2,N = 1,2,3]
% N = 1
N = 1; % N is the number of revolutions
[gamma1_list1,tof1] = exposin_lambert(r1,r2,tof_reqy,psi,k2,N,resg,rest);
% N = 2
N = 2; % N is the number of revolutions
[gamma1_list2,tof2] = exposin_lambert(r1,r2,tof_reqy,psi,k2,N,resg,rest);
% N = 3
N = 3; % N is the number of revolutions
[gamma1_list3,tof3] = exposin_lambert(r1,r2,tof_reqy,psi,k2,N,resg,rest);

%% Plot all results for N=0,1,2,3 and the required time of flight
gamma_treq = [min([gamma1_list0',gamma1_list1',gamma1_list2',gamma1_list3'])...
            :0.1:max([gamma1_list0',gamma1_list1',gamma1_list2',gamma1_list3'])];
treq_val = 2.*ones(size(gamma_treq)); % Time of flight requirement [years]

idx3 = find(tof3≤2,1,'last');
% y3 = [0:0.1:max([tof1',tof2',tof3'])];
y3 = linspace(0,tof_reqy,1000);
x3 = gamma1_list3(idx3)*ones(size(y3));

idx1 = find(tof1≤2,1,'last');
% y1 = [0:0.1:max([tof1',tof2',tof3'])];
y1 = linspace(0,tof_reqy,1000);
x1 = gamma1_list1(idx1)*ones(size(y1));

idx2 = find(tof2≤2,1,'last');
% y2 = [0:0.1:max([tof1',tof2',tof3'])];
y2 = linspace(0,tof_reqy,1000);
x2 = gamma1_list2(idx2)*ones(size(y2));

% figure(1)
plot(gamma1_list0,tof0,'-',gamma1_list1,tof1,'--',gamma1_list2,tof2,':',gamma1_list3,tof3,...
            '.-',gamma_treq,treq_val,'r-',x1,y1,'r-',x2,y2,'r-',x3,y3,'r-');
grid on
legend('N = 0','N = 1', 'N = 2', 'N = 3','Time req','location','best');
xlabel('\gamma_{1} [rad]');
ylabel('Time of flight [years]');
title(' Time of flight vs varying initial flight path angle for varying N')
```

Following is the function defined for obtaining Lambert solution to the exponential sinusoidal problem.

```
function [res1,res2] = exposin_lambert(r1,r2,tof_reqy,psi,k2,N,resg,rest)
    % Lambert solution to exponential sinusoids. Verified against
    % [Izzo,2006] - Lambert's problem for exponential sinusoids
    % https://pdfs.semanticscholar.org/d835/4da1d9deae15b70c29c09981d4e12559e8b7.pdf
    % INPUT DEFINTION:
    % r1, r2 -> initial and final position in AU
    % tof_req -> required time of flight in years
    % psi -> Angle between the initial and final positions r1,r2 in radians
    % k2 -> tuning parameter to tune the number of revolutions [-]
    % N -> no. of revolutions
    % resg -> split up the gamma range in resg intervals for integration
    % rest -> split up the theta range in rest intervals for integration
    % psi -> Angle between initial position vector r1 and final position vector r2 [rad]
    % OUTPUT DEFINTION:
    % res = [initial flight path angle gamma1 in radians, time of flight in years]
    % Step 1: Take in known inputs and define parameters
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script Author: Ali Nawaz, Delft University of Technology.
    % Aerospace Engineering Faculty [LR]
    % Mechanical, Maritime and Materials Engineering Faculty [3ME]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    AU = 149597870.66; %[km]
    sidereal_day = 86164.1004; %[s]
    tof_reqs = tof_reqy*365*sidereal_day; % Max required time of flight [seconds]

    % Step 2: Use the user defined value for k2
    % Step 3: Use the user defined value for N
    % Step 4: Compute gamma1_min and gamma1_max
    % gamma1 range defined by the constraint |k1*k2^2| < 1, to limit
    % options for gamma1. Limit search space, resulting in efficient
    % optimization.
    theta_bar = psi + 2*pi*N; % [rad]
    Δ = (2 * (1 - cos(k2*theta_bar))/(k2^4)) - (log(r1/r2))^2;
    tangamma1_min = (k2/2)*( - log(r1/r2)*cot((k2*theta_bar)/2) - sqrt(Δ));
    tangamma1_max = (k2/2)*( - log(r1/r2)*cot((k2*theta_bar)/2) + sqrt(Δ));
    gamma1_min = atan(tangamma1_min); % minimum value of gamma [rad]
    gamma1_max = atan(tangamma1_max); % maximum value of gamma [rad]

    % Step 5: Assume value for gamma1
    gamma_list = linspace(gamma1_min,gamma1_max,resg);
%     gamma_list = deg2rad(-80); % use it to verify script
    tof = [];
    for g = 1:length(gamma_list)
        gamma1 = gamma_list(g);
    % Step 6: Compute k0,k1, phi and time of flight
        sign_k1 = sign( log(r1/r2) + (tan(gamma1)/ k2)*sin(k2*theta_bar)); % sign k1
        mag_k1 = sqrt(  ((log(r1/r2) + (tan(gamma1)/ k2)*sin(k2*theta_bar))/ ( 1 - ...
            cos(k2*theta_bar)))^2 + (tan(gamma1)^2)/(k2^2)); % magnitude k1
        k1 = sign_k1*mag_k1; % final value of k1
        phi = acos(tan(gamma1)/(k1*k2)); % [rad]
        k0 = r1*AU/( exp(k1*sin(phi))); % final value of k0;
        % Simple integration scheme
%         tof(g) = TOF(k0,k1,k2,theta_bar,rest,phi); % Time of flight [s]
        % Composite Simpson's integration scheme
        tof(g) = TOFsimpsons(k0,k1,k2,theta_bar,rest,phi); % Time of flight [s]
   % Step 7: iterate until TOF matches TOFreq
        if tof(g) > tof_reqs
            break
        end
    end
    tof = tof./(365*sidereal_day); % Time of flight in years
    gamma_list = gamma_list(1:length(tof));
    res1 = gamma_list'; % Final results gamma1 [rad]
    res2 = tof'; % Final results time of flight [years]
end
```

Following is the function defined to estimate the time of flight with simple integration at a given instance.

```
function res = TOF(k0,k1,k2,theta_bar,rest,phi)
    % Time of flight estimation
    % INPUT DEFINITION:
    % k0 -> dimensionless scaling factor
    % k1 -> dynamic range parameter [-]
    % k2 -> winding parameter [-]
    % phi -> phase angle [rad]
    % theta_bar = maximum true anomaly [rad]
    % OUTPUT DEFINITION:
    % res = total time of flight
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script Author: Ali Nawaz, Delft University of Technology.
    % Aerospace Engineering Faculty [LR]
    % Mechanical, Maritime and Materials Engineering Faculty [3ME]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    AU = 149597870.66; % [km]
    % Assumption,q = 0 -> pure exponential sinusoid.
    q = 0;
    mu_sun = 1.327178*10^(11); % [km^3/s^2] Standard gravitational parameter Sun
    theta_list = linspace(0,theta_bar,rest+1); % discretize theta range for integration

    tof_dt=[];
    for dt = 2:length(theta_list)
        dtheta = theta_list(dt) - theta_list(dt-1); % Delta theta [rad]
        c = cos(k2*theta_list(dt) + phi);
        s = sin(k2*theta_list(dt) + phi);
        r = k0*exp(q*theta_list(dt)+k1*s); % radius from heliocentric co-ordinates at any ...
            given instance
        tang = k1*k2*c; % tan(gamma)

        tof_dt(dt) = sqrt( (r^3 * (tang^2 + k1*k2^2*s + 1))/mu_sun)*dtheta; % instantaneous ...
            time of flight [second]
    end
    res = sum(tof_dt); % total time of flight [seconds]
end
```

Following is the function constructed to obtain composite Simpson’s integration.

```matlab
function res = simpsons(f,lb,ub,rest)
    % Composite Simpson's rule for integration
    % INPUT DEFINITION:
    % f = function to be integration (e.g. @(x) sin(x))
    % lb = lower bound of integration
    % ub = Upper bounds of integration
    % rest = resolution of terms i.e. nummber of segments between lb and
    % ub.
    % OUTPUT:
    % Integrated value of the function
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script Author: Ali Nawaz, Delft University of Technology.
    % Aerospace Engineering Faculty [LR]
    % Mechanical, Maritime and Materials Engineering Faculty [3ME]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    h = (ub-lb)/rest;
    theta_list = linspace(lb,ub,rest);
    res =[f(theta_list(1))]; % Append the first term
    for j = 1:(rest/2 -2)
        res = [res, 2*f(theta_list(2*j + 1))]; % Append all the even parts
    end
    for j = 1:(rest/2 -1)
        res = [res, 4*f(theta_list(2*j))]; % Append all the odd parts
    end
    res = [res,f(theta_list(end))]; % Append the final term
    res = (h/3)*sum(res); % Final time of flight [seconds]
end
```

Following is the function defined to estimate the time of flight with composite Simpson integration scheme at a given instance.

```matlab
function res = TOFsimpsons(k0,k1,k2,theta_bar,rest,phi)
    % Time of flight estimation
    % INPUT DEFINITION:
    % k0 -> dimensionless scaling factor
    % k1 -> dynamic range parameter [-]
    % k2 -> winding parameter [-]
    % phi -> phase angle [rad]
    % theta_bar = maximum true anomaly [rad]
    % OUTPUT DEFINITION:
    % res = total time of flight
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script Author: Ali Nawaz, Delft University of Technology.
    % Aerospace Engineering Faculty [LR]
    % Mechanical, Maritime and Materials Engineering Faculty [3ME]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    AU = 149597870.66; % [km]
    % Assumption,q = 0 -> pure exponential sinusoid.
    q = 0;
    mu_sun = 1.327178*10^(11); % [km^3/s^2] Standard gravitational parameter Sun
    c = @(x) cos(k2*x + phi);
    s = @(x) sin(k2*x + phi);
    r = @(x) (k0*exp(q*x + k1*s(x))); % radius from heliocentric co-ordinates at any given ...
        instance
    tang = @(x) (k1*k2*c(x));
    f = @(x) sqrt( (r(x)^3)*(tang(x)^2 + k1*k2^2*s(x) + 1)/mu_sun);
    lb = 0;
    ub = theta_bar;
    res = simpsons(f,lb,ub,rest); % total time of flight [seconds]
end
```

References

[1] Composite Simpson's rule . https://en.wikipedia.org/wiki/Simpson%27s_rule. 17 March 2018.

[2] Dario Izzo. Lambert's problem for exponential sinusoids. *Journal of guidance, control and dynamics*, 29:1242–1245, 2006.

[3] R. Noomen. *AE4878 Mission Geometry and Orbit Design, exposins v4-20*. TU Delft, 2017.

[4] Anastassios E. Petropoulos and James M. Longuski. Shape-based algorithm for automated design of low-thrust, gravity-assist trajectories. *Journal of spacecrafts and rockets*, 41:787–796, 2004.

[5] James R. Wertz. *Orbit Constellation Design Management*. Springer, 2009.