

Predicting Mutual Visibility and Close Approach Conditions

This document describes several MATLAB scripts that can be used to predict important orbital events. Several of these scripts use a combination of one-dimensional minimization and root-finding to calculate event conditions. Scripts are provided that use analytic and numerical integration of the orbital motion during the solution process. Several of these scripts also provide graphic displays of event characteristics.

These MATLAB scripts can be used to solve the following orbital event problems:

- Mutual visibility between two satellites
- Closest approach between two satellites
- Closest approach between a satellite and ground site
- Closest approach between a satellite's sub point and a ground site

Mutual Visibility Between Two Satellites

This section describes two MATLAB scripts that can be used to determine mutual visibility between two satellites in circular or elliptical Earth orbits. These scripts use a combination of one-dimensional minimization and root-finding to calculate mutual visibility conditions. The objective function for these scripts is

$$f_{obj}(t) = -\mathbf{r}_1 \bullet \mathbf{r}_2 + r_s^2 - \sqrt{(r_1^2 - r_s^2)(r_2^2 - r_s^2)}$$

where

\mathbf{r}_1 = ECI position vector of the first satellite

\mathbf{r}_2 = ECI position vector of the second satellite

r_1 = geocentric radius of the first satellite

r_2 = geocentric radius of the second satellite

r_s = local radius of the Earth = $1.02r_{eq}$

r_{eq} = Earth equatorial radius

Whenever this function is negative, mutual visibility exists. Notice that the local radius r_s is increased by 2% to account for the thickness of the Earth's atmosphere in the calculations.

sat2sat1.m – mutual visibility between two Earth satellites – Kozai orbit propagation

In this script the orbits of both satellites are propagated using Kozai's method. Please note that the orbital elements can be either input interactively by the user or read from a simple data file.

Orbital Mechanics with MATLAB

The following are the contents of a typical two-satellite data file (twosats1.dat) included with this software suite.

```
satellite #1

semimajor axis (kilometers)
(semimajor axis > 0)
8000

orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
0

orbital inclination (degrees)
(0 <= inclination <= 180)
28.5

argument of perigee (degrees)
(0 <= argument of perigee <= 360)
0

right ascension of the ascending node (degrees)
(0 <= RAAN <= 360)
100

true anomaly (degrees)
(0 <= true anomaly <= 360)
45

satellite #2

semimajor axis (kilometers)
(semimajor axis > 0)
12000

orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
0.015

orbital inclination (degrees)
(0 <= inclination <= 180)
60

argument of perigee (degrees)
(0 <= argument of perigee <= 360)
270

right ascension of the ascending node (degrees)
(0 <= RAAN <= 360)
35

true anomaly (degrees)
(0 <= true anomaly <= 360)
0
```

Orbital Mechanics with MATLAB

The following is a typical user interaction with this script.

```
please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,1998

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0

orbital elements menu

    <1> user input

    <2> data file

? 1

initial orbital elements - first satellite

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? 0

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 28.5

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 45

initial orbital elements - second satellite

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 12000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? .015

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 60
```

Orbital Mechanics with MATLAB

```
please input the argument of perigee (degrees)
(0 <= argument of perigee <= 360)
? 270
```

```
please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 35
```

```
please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 0
```

```
please input the simulation duration in days
? 100
```

The script will ask if you would like to create a screen display of the mutual visibility conditions during the simulation. This prompt appears as follows:

```
would you like screen output (y = yes, n = no)
?
```

The following is the screen display for the first mutual visibility contact for this example.

```
begin mutual visibility contact

calendar date      01-Jan-1998
universal time     00:52:07

Julian date        2450814.5362

approach distance  14727.6235 kilometers

end mutual visibility contact

calendar date      01-Jan-1998
universal time     03:03:10

Julian date        2450814.6272

approach distance  14626.4657 kilometers

event duration     131.0526 minutes
```

After the simulation is complete the software will ask if you would like to create a data file of the mutual visibility conditions. This prompt appears as follows:

```
would you like to create an ascii data file (y = yes, n = no)
?
```

If you respond with *y* for yes, the software will ask you to input the name of the data file with

```
please input the mutual visibility data file name
(be sure to include a file name extension)
?
```

Orbital Mechanics with MATLAB

Please note that you may not want to create screen displays when requesting a data file.

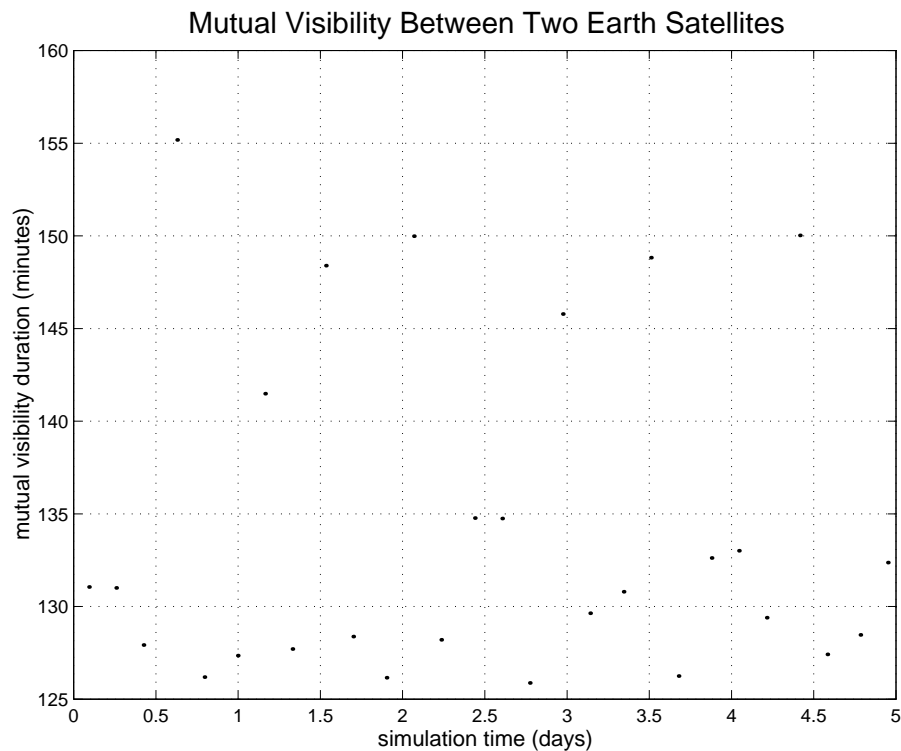
The following is part of the data file for this example. The first column is the simulation time at which mutual visibility begins, the second column is the simulation time at the mutual visibility “minimum”, and the third column is the simulation time at which mutual visibility ends. The fourth column is the event duration in minutes.

begin time (days)	min time (days)	end time (days)	duration (minutes)
0.0362	0.0933	0.1272	131.0526
0.2270	0.2588	0.3180	131.0096
0.4013	0.4247	0.4901	127.9245
0.5741	0.6292	0.6819	155.1782
0.7676	0.7951	0.8553	126.1890
0.9398	0.9982	1.0282	127.3524
1.1227	1.1640	1.2209	141.4849
1.3048	1.3300	1.3935	127.7107
1.4770	1.5342	1.5801	148.3959

The script will also ask if you would like to create a graphics display of the event duration versus simulation time with this final prompt.

```
would you like to display graphics (y = yes, n = no)
?
```

If you respond with *y* for yes, the software will create a graphics display like the following:



sat2sat2.m – mutual visibility between two Earth satellites – numerical integration

This script uses numerical integration to propagate the orbits of two satellites while searching for mutual visibility. The algorithm includes the effect of Earth oblateness due to J_2 but can easily be modified to include higher-order gravity effects and other orbital perturbations.

The following is a typical user interaction with this script. Note that the orbital elements can be either input interactively by the user or read from a simple data file.

```
please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,1998

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0

orbital elements menu

    <1> user input
    <2> data file
? 1

initial orbital elements - first satellite

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? 0

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 28.5

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 45

initial orbital elements - second satellite

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 12000

please input the orbital eccentricity (non-dimensional)
```

Orbital Mechanics with MATLAB

```
(0 <= eccentricity < 1)
? .015

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 60

please input the argument of perigee (degrees)
(0 <= argument of perigee <= 360)
? 270

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 35

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 0

please input the simulation duration in days
? 100

would you like screen output (y = yes, n = no)
? y
```

The following is the screen display of the first mutual visibility event for the example.

```
begin mutual visibility contact

calendar date      01-Jan-1998
universal time     00:52:12

Julian date        2450814.5362

approach distance  14717.2151 kilometers

end mutual visibility contact

calendar date      01-Jan-1998
universal time     03:02:45

Julian date        2450814.6269

approach distance  14614.1000 kilometers

event duration     130.5587 minutes
```

This script also includes the data file and graphics display options described in the previous `sat2sat1.m` script.

Closest Approach Between Two Earth Satellites

This section describes two MATLAB scripts that can be used to determine closest approach conditions between two satellites in circular or elliptical Earth orbits. The user can specify a minimum range constraint to use during the simulation. These scripts use a combination of one-dimensional minimization and root-finding to calculate close approach conditions. This section also includes a third MATLAB script that can be used to graphically display the separation distance between two Earth satellites.

The objective function for each application is given by the following expression:

$$f_{obj}(t) = \sqrt{|\mathbf{r}_1 - \mathbf{r}_2|^2} = \sqrt{(r_{1x} - r_{2x})^2 + (r_{1y} - r_{2y})^2 + (r_{1z} - r_{2z})^2}$$

where \mathbf{r}_1 is the ECI position vector of the first satellite and \mathbf{r}_2 is the ECI position vector of the second satellite.

During the root-finding computations, the objective function is formulated as

$$f_{obj}(t) = \sqrt{|\mathbf{r}_1 - \mathbf{r}_2|^2} - d_{\min}$$

where d_{\min} is the minimum separation constraint defined by the user.

ca2sats1.m – closest approach between two Earth satellites – Kozai orbit propagation

In this script, the orbits of both satellites are propagated using Kozai's J_2 method. Please note that the orbital elements can be either input interactively by the user or read from a simple ASCII data file.

The following are the contents of a typical two-satellite data file (twosats1.dat).

```
satellite #1

semimajor axis (kilometers)
(semimajor axis > 0)
8000

orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
0

orbital inclination (degrees)
(0 <= inclination <= 180)
28.5

argument of perigee (degrees)
(0 <= argument of perigee <= 360)
0
```

Orbital Mechanics with MATLAB

```
right ascension of the ascending node (degrees)
(0 <= RAAN <= 360)
100

true anomaly (degrees)
(0 <= true anomaly <= 360)
45

satellite #2

semimajor axis (kilometers)
(semimajor axis > 0)
12000

orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
0.015

orbital inclination (degrees)
(0 <= inclination <= 180)
60

argument of perigee (degrees)
(0 <= argument of perigee <= 360)
270

right ascension of the ascending node (degrees)
(0 <= RAAN <= 360)
35

true anomaly (degrees)
(0 <= true anomaly <= 360)
0
```

The following is a typical user interaction with this script.

```
please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,1998

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0

orbital elements menu

    <1> user input
    <2> data file

? 1

initial orbital elements - first satellite

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000
```

Orbital Mechanics with MATLAB

```
please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? 0

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 28.5

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 45

initial orbital elements - second satellite

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 12000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? .015

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 60

please input the argument of perigee (degrees)
(0 <= argument of perigee <= 360)
? 270

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 35

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 0

would you like to enforce a minimum distance constraint (y = yes, n = no)
? y

please input the minimum distance constraint (kilometers)
? 5000

please input the simulation duration in days
? 10
```

The script will ask if you would like to create a screen display of the mutual visibility conditions during the simulation. This prompt appears as follows:

```
would you like screen output (y = yes, n = no)
?
```

Orbital Mechanics with MATLAB

The following is the screen display for the first close approach for this example.

```
time and conditions at constraint entry

calendar date      01-Jan-1998
universal time     02:11:26

Julian date        2450814.5913

approach distance  5000.0007 kilometers

time and conditions at closest approach

calendar date      01-Jan-1998
universal time     02:14:45

Julian date        2450814.5936

approach distance  4820.5508 kilometers

time and conditions at constraint exit

calendar date      01-Jan-1998
universal time     02:18:01

Julian date        2450814.5958

approach distance  5000.0006 kilometers

event duration     6.5814 minutes
```

After the simulation is complete the software will ask if you would like to create a data file of the close approach conditions. This prompt appears as follows:

```
would you like to create an ascii data file (y = yes, n = no)
?
```

If you respond with *y* for yes, the software will ask you to input the name of the data file with

```
please input the close approach data file name
(be sure to include a file name extension)
?
```

Since you must manually cycle each screen display, you may not want to create screen displays when requesting a data file.

The following is part of the data file for this example. The first column is the simulation time at which close approach begins, the second column is the simulation time at the closest approach, and the third column is the simulation time at which close approach ends. The fourth column is the event duration in minutes.

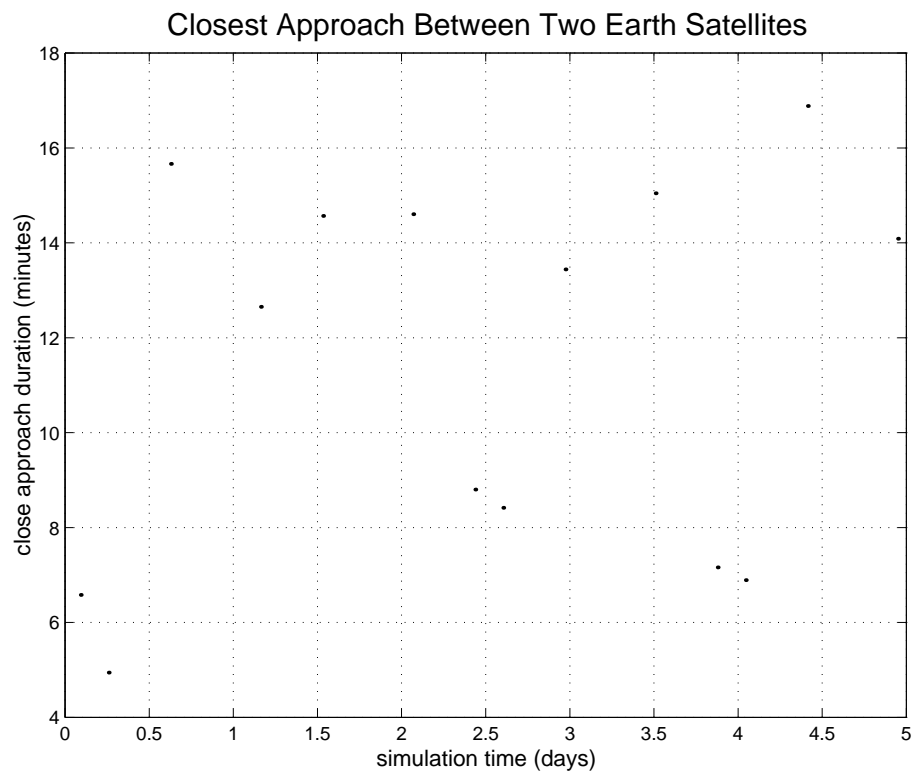
Orbital Mechanics with MATLAB

begin time (days)	ca time (days)	end time (days)	duration (minutes)
0.0913	0.0936	0.0958	6.5814
0.2575	0.2592	0.2609	4.9456
0.6234	0.6289	0.6343	15.6651
1.1600	1.1643	1.1688	12.6529
1.5288	1.5339	1.5389	14.5659
2.0643	2.0694	2.0745	14.6035
2.4357	2.4388	2.4418	8.8008
2.6020	2.6049	2.6079	8.4162
2.9696	2.9743	2.9789	13.4401

Finally, the script will also ask if you would like to create a graphics display of the event duration versus simulation time with this final prompt.

```
would you like to display graphics (y = yes, n = no)
?
```

If you respond with *y* for yes, the software will create a graphics display similar to the following:



ca2sats2.m – closest approach between two Earth satellites – numerical integration

This MATLAB script uses numerical integration to propagate the orbits of two Earth satellites while searching for close approach. The algorithm includes the effect of Earth oblateness J_2 but can easily be modified to include higher-order gravity effects, aerodynamic drag and other

Orbital Mechanics with MATLAB

orbital perturbations. The user can specify a minimum range constraint to use during the simulation.

The following is a typical user interaction with this script. Note that the orbital elements can be input interactively by the user or read from a simple data file.

```
please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,1998

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0

orbital elements menu

  <1> user input
  <2> data file

? 2

please input the orbital elements data file name
(be sure to include the file name extension)
? twosats1.dat

would you like to enforce a minimum distance constraint (y = yes, n = no)
? y

please input the minimum distance constraint (kilometers)
? 5000

please input the simulation duration in days
? 10

would you like screen output (y = yes, n = no)
? y
```

The following is the screen display of the first close approach event for the example.

```
time and conditions at constraint entry

calendar date      01-Jan-1998
universal time     02:11:37

Julian date        2450814.5914
approach distance   5000.0000 kilometers

time and conditions at closest approach

calendar date      01-Jan-1998
universal time     02:14:46
```

Orbital Mechanics with MATLAB

```
Julian date          2450814.5936
approach distance    4840.3891 kilometers
time and conditions at constraint exit
calendar date        01-Jan-1998
universal time        02:17:51
Julian date          2450814.5957
approach distance    5000.0000 kilometers
event duration        6.2193 minutes
```

This script also includes the data file and graphics display options described in the previous `ca2sats1.m` script.

`gca2sats.m` – graphics display of separation distance between two Earth satellites

This MATLAB script can be used to graphically display the separation distance between two Earth satellites. The user can specify a minimum range constraint to use during the creation of the graphics data. In this script the orbits of both satellites are analytically propagated using Kozai's J_2 method.

The following is a typical user interaction with this script.

```
please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,2001

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0

orbital elements menu

  <1> user input
  <2> data file

? 2

please input the orbital elements data file name
(be sure to include the file name extension)
? twosats1.dat

would you like to enforce a minimum distance constraint (y = yes, n = no)
? y

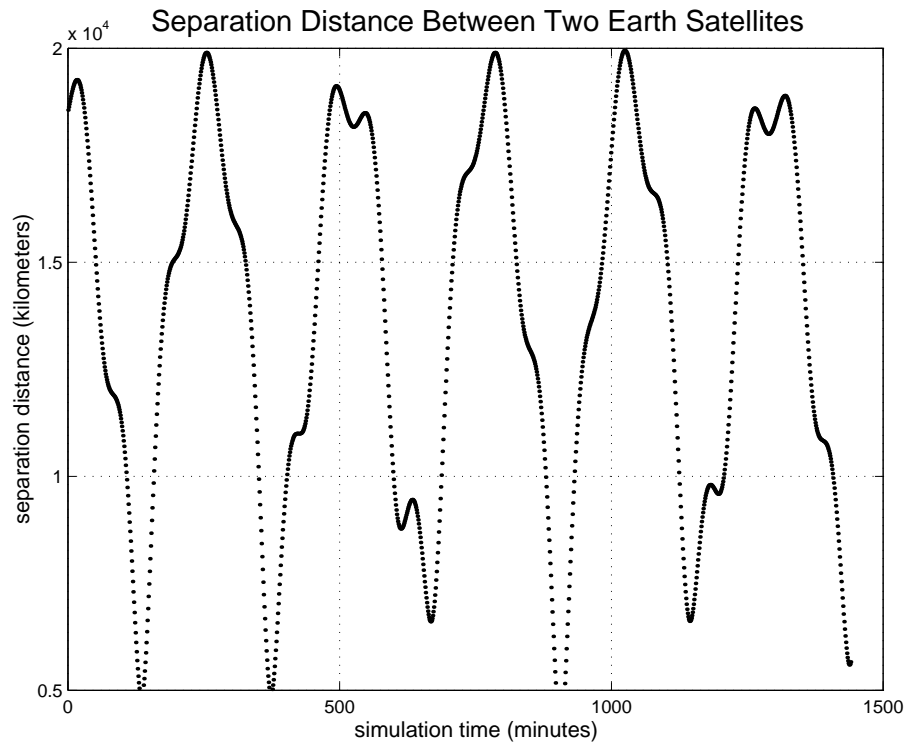
please input the minimum distance constraint (kilometers)
? 5000
```

Orbital Mechanics with MATLAB

```
please input the simulation duration in days  
? 1
```

```
please input the step size in minutes  
? 1
```

The following is the graphics display for this example. The data is displayed at the time step input by the user.



Closest Approach Between a Satellite and Ground Site

This section describes two MATLAB scripts that can be used to determine closest approach conditions between a satellite in a circular or elliptical Earth orbit and a ground site on an oblate Earth. This script uses a combination of one-dimensional minimization and root-finding to calculate close approach conditions. The user can specify a minimum slant range constraint to use during the simulation. This section also includes a third MATLAB script that can be used to graphically display the separation distance between an Earth satellite and ground site.

The objective function for these two applications is

$$f_{obj}(t) = \sqrt{|\mathbf{r}_{sat} - \mathbf{r}_{obs}|^2} = \sqrt{(r_{satx} - r_{obsx})^2 + (r_{saty} - r_{obsy})^2 + (r_{satz} - r_{obsz})^2}$$

where \mathbf{r}_{sat} is the ECI position vector of the satellite and \mathbf{r}_{obs} is the ECI position vector of the ground site.

Orbital Mechanics with MATLAB

During the root-finding calculations, the objective function is given by

$$f_{obj}(t) = \sqrt{|\mathbf{r}_{sat} - \mathbf{r}_{obs}|^2} - d_{min}$$

where d_{min} is an *optional* minimum separation constraint defined by the user.

rts2sat1.m – closest approach between a satellite and ground site – Kozai orbit propagation

This script file uses Kozai's J_2 orbit propagation method while looking for close approach between a satellite and a ground site on an oblate Earth. The user can specify a minimum slant range constraint to use during the simulation.

The following is a typical user interaction with this MATLAB script.

```
please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,1998

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0

initial orbital elements

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? 0

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 40

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 45

ground site coordinates

please input the geographic latitude of the ground site
(-90 <= degrees <= +90, 0 <= minutes <= 60, 0 <= seconds <= 60)
(north latitude is positive, south latitude is negative)
? 40,0,0
```

Orbital Mechanics with MATLAB

```
please input the geographic longitude of the ground site
(0 <= degrees <= 360, 0 <= minutes <= 60, 0 <= seconds <= 60)
(east longitude is positive, west longitude is negative)
? -105,0,0
```

```
please input the altitude of the ground site (meters)
(positive above sea level, negative below sea level)
? 1000
```

```
would you like to enforce a minimum distance constraint (y = yes, n = no)
? y
```

```
please input the minimum distance constraint (kilometers)
? 2000
```

```
please input the simulation duration in days
? 10
```

The script will ask if you would like to create a screen display of the close approach conditions during the simulation. This prompt appears as follows:

```
would you like screen output (y = yes, n = no)
?
```

The following is the first close approach calculated by this script.

```
time and conditions at constraint entry

calendar date      01-Jan-1998
universal time     09:55:00

Julian date        2450814.9132

slant range        1999.9992 kilometers
```

```
time and conditions at closest approach

calendar date      01-Jan-1998
universal time     09:56:53

Julian date        2450814.9145

slant range        1884.7159 kilometers
```

```
time and conditions at constraint exit

calendar date      01-Jan-1998
universal time     09:58:46

Julian date        2450814.9158

slant range        1999.9992 kilometers
event duration     3.7689 minutes
```

Orbital Mechanics with MATLAB

After the simulation is complete the software will ask if you would like to create a data file of the close approach conditions. This prompt appears as follows:

```
would you like to create an ascii data file (y = yes, n = no)
?
```

If you respond with *y* for yes, the software will ask you to input the name of the data file with

```
please input the close approach data file name
(be sure to include a file name extension)
?
```

Since you must manually cycle each screen display, you may not want to create screen displays when requesting a data file.

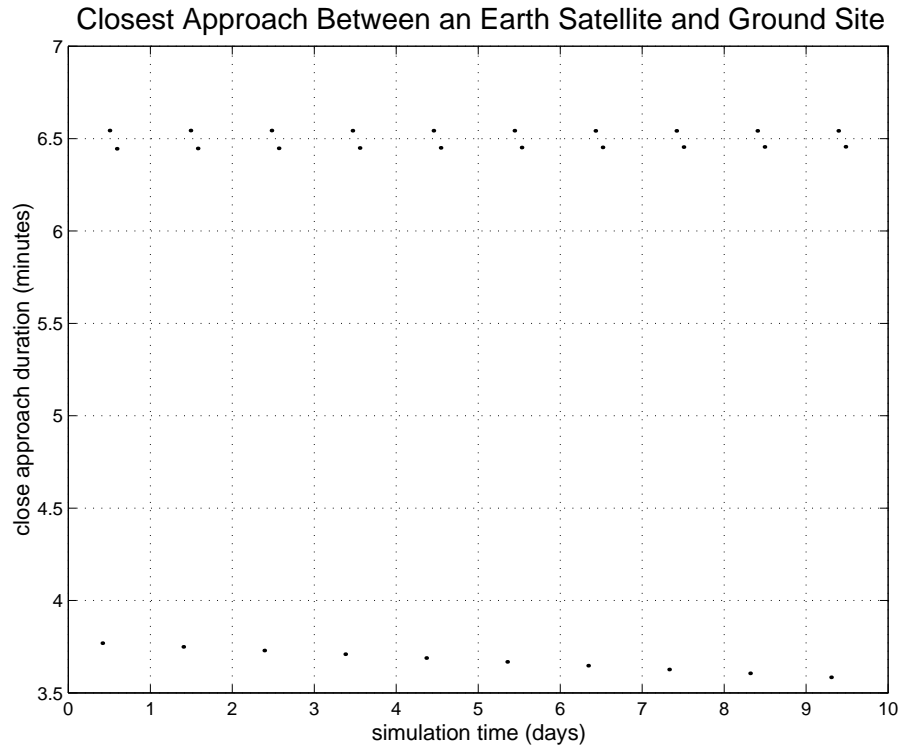
The following is part of the data file for this example. The first column is the simulation time at which close approach begins, the second column is the simulation time at the closest approach, and the third column is the simulation time at which close approach ends. The fourth column is the event duration in minutes.

begin time (days)	ca time (days)	end time (days)	duration (minutes)
0.4132	0.4145	0.4158	3.7689
0.4998	0.5021	0.5044	6.5436
0.5877	0.5900	0.5922	6.4451
1.4009	1.4022	1.4035	3.7492
1.4875	1.4897	1.4920	6.5434
1.5754	1.5776	1.5799	6.4463
2.3885	2.3898	2.3911	3.7293
2.4751	2.4774	2.4797	6.5432

The script will also ask if you would like to create a graphics display of the event duration versus simulation time with this final prompt.

```
would you like to display graphics (y = yes, n = no)
?
```

If you respond with *y* for yes, the software will create a graphics display similar to the following:



rts2sat2.m – closest approach between a satellite and ground site – numerical integration

This MATLAB script propagates a satellite’s orbit using numerical integration while searching for close approaches between a satellite and a ground site on an oblate Earth. It demonstrates one-dimensional minimization while numerically integrating the orbital equations of motion. The user can specify a minimum slant range constraint to use during the simulation. The algorithm includes the effect of Earth oblateness due to J_2 but can easily be modified to include higher-order gravity effects, aerodynamic drag and other orbital perturbations.

The following is a typical user interaction with this program.

```

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,1998

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0

initial orbital elements

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? 0
    
```

Orbital Mechanics with MATLAB

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 40

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 45

ground site coordinates

please input the geographic latitude of the ground site
(-90 <= degrees <= +90, 0 <= minutes <= 60, 0 <= seconds <= 60)
(north latitude is positive, south latitude is negative)
? 40,0,0

please input the geographic longitude of the ground site
(0 <= degrees <= 360, 0 <= minutes <= 60, 0 <= seconds <= 60)
(east longitude is positive, west longitude is negative)
? -105,0,0

please input the altitude of the ground site (meters)
(positive above sea level, negative below sea level)
? 1000

would you like to enforce a minimum distance constraint (y = yes, n = no)
? y

please input the minimum distance constraint (kilometers)
? 2000

please input the simulation duration in days
? 10

would you like screen output (y = yes, n = no)
? y

The following is the first close approach found by this application.

```
time and conditions at constraint entry

calendar date      01-Jan-1998
universal time     09:55:00

Julian date        2450814.9132

slant range        2000.0000 kilometers

time and conditions at closest approach
```

Orbital Mechanics with MATLAB

```
calendar date      01-Jan-1998
universal time     09:56:53

Julian date       2450814.9145

slant range       1884.8861 kilometers

time and conditions at constraint exit

calendar date     01-Jan-1998
universal time    09:58:46

Julian date       2450814.9158

slant range       2000.0000 kilometers

event duration    3.7699 minutes
```

This script also includes the data file and graphics display options described in the previous `rts2sat1.m` script.

grts2sat.m – graphics display of the separation distance between an Earth satellite and ground site

This MATLAB script can be used to graphically display the separation distance between a satellite and a ground site on an oblate Earth. The user can specify a minimum slant range constraint to use during the creation of the graphics data. In this script, the orbit of the satellite is propagated using Kozai's J_2 method.

The following is a typical user interaction with this script.

```
please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,2001

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0

initial orbital elements

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? 0

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 28.5
```

Orbital Mechanics with MATLAB

```
please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 0

ground site coordinates

please input the geographic latitude of the ground site
(-90 <= degrees <= +90, 0 <= minutes <= 60, 0 <= seconds <= 60)
(north latitude is positive, south latitude is negative)
? 40,0,0

please input the geographic longitude of the ground site
(0 <= degrees <= 360, 0 <= minutes <= 60, 0 <= seconds <= 60)
(east longitude is positive, west longitude is negative)
? -105,0,0

please input the altitude of the ground site (meters)
(positive above sea level, negative below sea level)
? 0

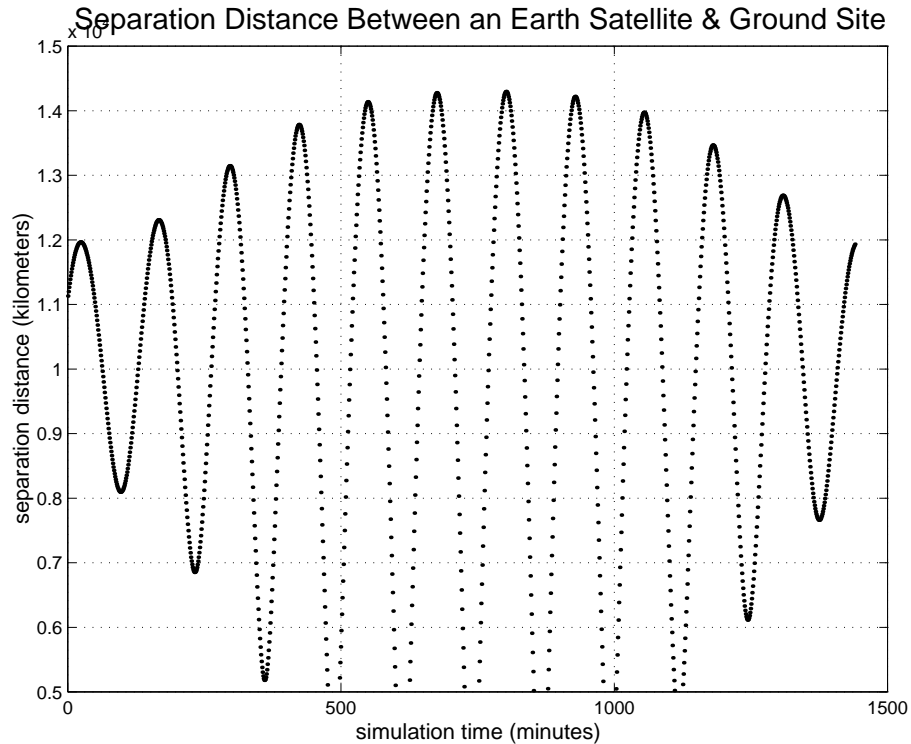
would you like to enforce a minimum distance constraint (y = yes, n = no)
? y

please input the minimum distance constraint (kilometers)
? 5000

please input the simulation duration in days
? 1

please input the step size in minutes
? 1
```

The following is the graphics display for this example. The data is displayed at the time step input by the user.



Closest Approach Between a Ground Site and a Satellite's Ground Track

This section describes three MATLAB scripts that can be used to determine closest approach conditions between the ground track of a satellite in a circular or elliptical Earth orbit and a ground site on an oblate Earth. The first two scripts use a combination of one-dimensional minimization and root-finding to calculate close approach conditions. The third script graphically displays the separation distance between a satellite's subpoint and a ground site for a user-defined simulation period.

The objective function for the first two applications is a function of the geodesic distance between the ground site and satellite subpoint which is in turn a function of the latitude and longitude of the ground site and satellite subpoint. The geodesic distance is computed using a MATLAB function called `geodesic.m`.

During the root-finding solution, the objective function is given by

$$f_{obj}(t) = d - d_{\min}$$

where d is the separation distance at any simulation time t and d_{\min} is an *optional* minimum separation distance constraint defined by the user. These scripts use a flattening factor f equal to $1/298.257$ and an equatorial radius r_{eq} equal to 6378.14 kilometers. The corresponding polar radius is determined from

$$r_p = r_{eq}(1 - f)$$

rts2trk1.m – closest approach between a satellite’s ground track and a ground site – Kozai orbit propagation

This script file uses Kozai’s J_2 orbit propagation method while looking for minimum separation distance between a satellite’s ground track and a ground site on an oblate Earth. The user can specify a minimum separation distance constraint to use during the simulation.

The following is a typical user interaction with this MATLAB script.

```
program rts2trak1

< closest approach between a ground site and a satellite groundtrack >

    < Kozai orbit propagation >

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,2003

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0

initial orbital elements

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? .015

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 38

please input the argument of perigee (degrees)
(0 <= argument of perigee <= 360)
? 200

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 45

ground site coordinates

please input the geographic latitude of the ground site
(-90 <= degrees <= +90, 0 <= minutes <= 60, 0 <= seconds <= 60)
```

Orbital Mechanics with MATLAB

(north latitude is positive, south latitude is negative)
? 40,0,0

please input the geographic longitude of the ground site
(0 <= degrees <= 360, 0 <= minutes <= 60, 0 <= seconds <= 60)
(east longitude is positive, west longitude is negative)
? -105,0,0

please input the altitude of the ground site (meters)
(positive above sea level, negative below sea level)
? 0

would you like to enforce a minimum distance constraint (y = yes, n = no)
? y

please input the minimum distance constraint (kilometers)
? 800

please input the simulation duration in days
? 1

would you like screen output (y = yes, n = no)
? y

The following is the script output for the first close approach.

time and conditions at constraint entry

calendar date	01-Jan-2003
universal time	10:51:48
Julian date	2452640.9526
distance	800.0000 kilometers
subpoint latitude	33.3275 degrees
subpoint longitude	251.6072 degrees

time and conditions at closest approach

calendar date	01-Jan-2003
universal time	10:53:32
Julian date	2452640.9538
distance	592.2875 kilometers
subpoint latitude	34.9296 degrees
subpoint longitude	257.0898 degrees

Orbital Mechanics with MATLAB

time and conditions at constraint exit

```
calendar date      01-Jan-2003
universal time     10:55:16

Julian date       2452640.9550

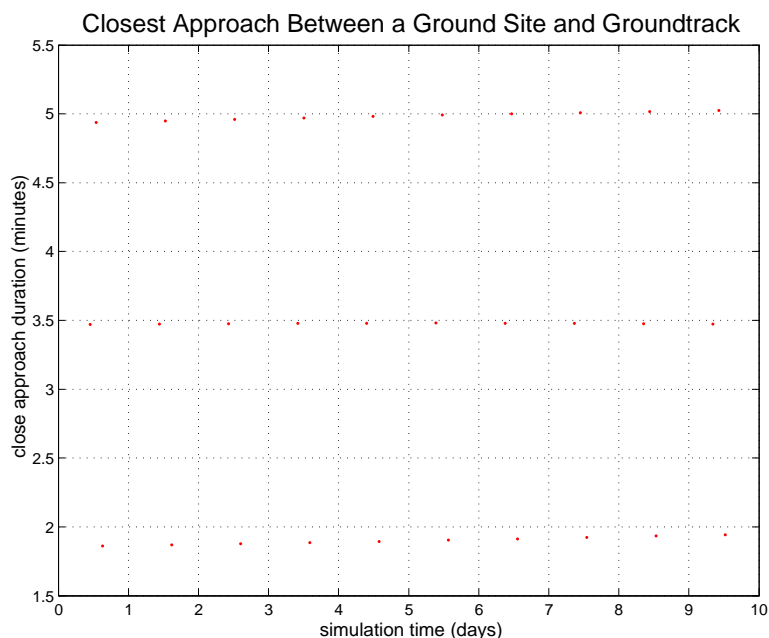
distance          800.0000 kilometers

subpoint latitude  36.2333 degrees

subpoint longitude 262.7839 degrees

event duration    3.4698 minutes
```

This MATLAB script also includes the data file and graphics display options described in previous scripts. The following is a graphics summary for this example.



Here is the first part of the data file for this example.

begin time (days)	ca time (days)	end time (days)	duration (minutes)
0.7222	0.7232	0.7243	2.9554
0.8092	0.8109	0.8125	4.7395
0.8976	0.8985	0.8994	2.5194
1.7098	1.7108	1.7118	2.9608
1.7968	1.7984	1.8001	4.7428
1.8852	1.8860	1.8869	2.5168
2.6973	2.6983	2.6994	2.9666
2.7843	2.7860	2.7876	4.7470
2.8727	2.8736	2.8745	2.5146
3.6848	3.6859	3.6869	2.9727

rts2trk2.m – closest approach between a satellite’s ground track and a ground site – numerical integration

This MATLAB script propagates a satellite’s orbit using numerical integration while searching for close approaches between a satellite’s subpoint and a ground site on an oblate Earth. It demonstrates one-dimensional root-finding and minimization while numerically integrating the orbital equations of motion. The user can specify a minimum separation distance constraint to use during the simulation. The algorithm includes the effect of Earth oblateness due to J_2 but can easily be modified to include higher-order gravity effects, aerodynamic drag and other orbital perturbations.

This script also includes the data file and graphics display options described in the previous scripts. The following is the first close approach output for the example given in the previous `rts2trak1.m` script.

```
time and conditions at constraint entry

calendar date      01-Jan-2003
universal time     10:52:04

Julian date        2452640.9528

distance           800.0000 kilometers
subpoint latitude  33.3379 degrees
subpoint longitude 251.5755 degrees

time and conditions at closest approach

calendar date      01-Jan-2003
universal time     10:53:48

Julian date        2452640.9540

distance           590.6390 kilometers
subpoint latitude  34.9422 degrees
subpoint longitude 257.0786 degrees

time and conditions at constraint exit

calendar date      01-Jan-2003
universal time     10:55:33

Julian date        2452640.9552

distance           800.0000 kilometers
subpoint latitude  36.2459 degrees
```

Orbital Mechanics with MATLAB

```
subpoint longitude      262.7944  degrees
event duration          3.4813  minutes
```

grts2trk.m – graphics display of the separation distance between a satellite’s ground track and a ground site

This MATLAB script can be used to graphically display the separation distance between a satellite’s subpoint and a ground site on an oblate Earth. The user can specify a minimum separation distance constraint to use during the creation of the graphics data. In this script the orbit of the satellite is propagated using Kozai’s J_2 method.

The following is a typical user interaction with this script.

```
program grts2trk
< separation distance between a ground site and Earth satellite ground track >
    < Kozai orbit propagation >

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,2003

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0

initial orbital elements

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? .015

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 38

please input the argument of perigee (degrees)
(0 <= argument of perigee <= 360)
? 200

please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 100

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 45
```

Orbital Mechanics with MATLAB

ground site coordinates

please input the geographic latitude of the ground site
(-90 <= degrees <= +90, 0 <= minutes <= 60, 0 <= seconds <= 60)
(north latitude is positive, south latitude is negative)
? 40,0,0

please input the geographic longitude of the ground site
(0 <= degrees <= 360, 0 <= minutes <= 60, 0 <= seconds <= 60)
(east longitude is positive, west longitude is negative)
? -105,0,0

please input the altitude of the ground site (meters)
(positive above sea level, negative below sea level)
? 0

would you like to enforce a minimum distance constraint (y = yes, n = no)
? n

please input the simulation duration in days
? 1

please input the step size in minutes
? 1

The following is the graphics display for this example.

