

Predicting Orbital Events: Part 1

This *Numerit* program (demoevt1) demonstrates how to calculate *extrema* type orbital events. These are events which involve the minimum or maximum of a *scalar objective function*. Typical extrema events include apogee and perigee of a satellite, closest approach between a satellite and a ground site, minimum angular separation between two satellites, and closest approach between two satellites.

This demonstration program calculates the calendar date, universal time and slant range for closest approach conditions between an Earth satellite and a ground site. It uses a one-dimensional optimization algorithm due to Richard Brent to solve this problem. Additional information about this numerical method can be found in *Algorithms for Minimization Without Derivatives*, R.P. Brent, Prentice-Hall, 1972. As the title of this book indicates, this algorithm does not require derivatives of the objective function. This feature is important because the analytic first derivative of many objective functions is difficult to derive and code.

The following is the main part of the source code of this demonstration program with several explanatory comments.

The program begins by interactively requesting the initial calendar date as follows:

```
` request initial calendar date
cdate = "1,1,1998"
getdate(cdate, month, day, year)
```

It will then request the initial universal time with this prompt:

```
utstr = "0,0,0"
gettime(utstr, uthr, utmin, utsec)
```

The request for initial orbital elements uses the following code

```
ioev[6] = 0
ioev = 1,1,1,1,1,1
oev1[1] = 8000
oev1[2] = 0
oev1[3] = 40
oev1[4] = 0
oev1[5] = 100
oev1[6] = 45
getoe(ioev, oev1)
```

Notice how *Numerit* lets us set initial values for inputs.

Orbital Mechanics with Numerit

The program will also ask for the ground site geodetic coordinates with this next prompts.

```
obslatstr = "40,0,0"
obslongstr = "-105,0,0"
obsalt = 1000
getobs(obslatstr, obslongstr, obslat, obslong, obsalt)
```

Finally, the software requests the simulation duration in *days* with this next prompt:

```
loop
  ndays = 1
  input " simulation duration (days) ?" ndays
  if (ndays > 0) break
```

The next section of the source code defines the *search* and *function* step sizes. The search step size `dt` tells the algorithm how far to move when looking for a function extrema and the function step size `dtsml` tells the algorithm how far to move when looking for an increase (or decrease) in the objective function value.

```
` define search parameters (days)
dt = period / 4
dtsml = 10 / 86400
```

The next section of the code defines the initial and final times of the search interval.

```
` find events
ti = 0
tf = ndays
```

The actual search is performed by the following statements:

```
` redirect objective and events routines
objfunc -> objfunc1
events -> events1
oevent1 (ti, tf, dt, dtsml)
```

The search function `oevent1` requires both a function which evaluates the user-defined scalar objective value and a function which displays the circumstances of the orbital events. Notice how we have "rerouted" the names of both the objective (`objfunc1`) and event (`events1`) functions prior to calling the main routine which searches for the events. Rerouting is a powerful feature unique to *Numerit Pro*.

Choosing the Search Parameters

The proper selection of the search parameters dt and dt_{sml} depends on the type of orbital event you are trying to predict. For example, the value of dt depends on how often the event occurs. For events which happen once per orbital period (aphelion, perihelion, etc.), a good value for dt is one fourth of the satellite's orbital period. When calculating events involving two satellites, one should use one fourth of the smaller of the two orbital periods. Be careful not to make dt too large or the algorithm may "step over" one or more solutions.

The value of dt_{sml} must be selected such that it produces a "big" enough change in the objective function to tell the algorithm in which direction to move. This is similar to choosing the value of x when numerically estimating the derivative of a function of the form $y = f(x)$. For most events a value of $dt_{sml} = 10 / 86400$ days should be adequate. An examination of a plot of the objective function over a period of time can also provide insight into the proper selection of the dt_{sml} parameter.

The *Numerit* function which solves this problem has the following syntax and arguments:

```
function oevent1 (ti, tf, dt, dt_sml)
` predict "extrema" type orbital events
` input
`  ti    = initial simulation time
`  tf    = final simulation time
`  dt    = step size used for bounding minima
`  dt_sml = small step size used to determine whether
`          the function is increasing or decreasing
```

During the search for events, this function calls a function named *minima* which calculates the *extremum* of a one-dimensional user-defined objective function. The syntax of this optimization function is as follows:

```
function minima (a, b, tolm, xmin, fmin)
` one-dimensional minimization function
` Brent's method
` input
`  a    = initial x search value
`  b    = final x search value
`  tolm = convergence criterion
` output
`  xmin = minimum x value
`  fmin = minimum function value
```

Orbital Mechanics with Numerit

This function requires an objective function coded as `name(x, fx)` where `x` is the current function argument, `fx` is the scalar value of the objective function evaluated at `x` and `name` is the name of the user-defined function.

The source code of the objective function for this example is as follows:

```
function objfunc1(x, fx)
` objective function
` slant range between an Earth
` satellite and ground site
` input
` x = function argument
` output
` fx = function value at x
` Orbital Mechanics with Numerit
.....
` observer local sidereal time
obslst = modulo(%gsti + %omega * x + %obslong)
` compute ground site position vector
sobslst = sin(obslst)
cobslst = cos(obslst)
rsite[1] = %gx * %cobslat * cobslst
rsite[2] = %gx * %cobslat * sobslst
rsite[3] = %gy * %sobslat
` compute satellite position vector
kozail(0, x, %oev1, oev2, rsat, vsat)
` eci vector from observer to satellite
for i = 1 to 3
    obs2sat[i] = rsat[i] - rsite[i]
` observer-to-satellite slant range
srange = sqrt(vdot(obs2sat, obs2sat))
%drsaved = srange
fx = srange
```

Orbital Mechanics with Numerit

This code simply calculates the scalar slant range of the satellite, in kilometers, f_x as a function of the time argument x . For this example the time argument is the elapsed time, in days, since the simulation began.

The objective function is given by the expression

$$f(t) = \sqrt{\left(r_{sat_x} - r_{site_x}\right)^2 + \left(r_{sat_y} - r_{site_y}\right)^2 + \left(r_{sat_z} - r_{site_z}\right)^2} \quad (1)$$

where r_{sat_x} , r_{sat_y} and r_{sat_z} are the rectangular, geocentric position components of the satellite and r_{site_x} , r_{site_y} and r_{site_z} are the rectangular, geocentric position components of the ground site at any search time t .

Finally, the function `oevent1` requires another function named `events1` which displays the circumstances of each orbital event. The function source code for this example is as follows:

```
function events1 (ti, tf, topt)
` display circumstances of orbital events
` input
` ti = initial simulation time
` tf = final simulation time
` topt = extrema time
` Orbital Mechanics with Numerit
.....
` compute and display conditions at extrema
objfunc(topt, froot)
jdate = %jdatei + topt
gdate(jdate, month, day, year)
thours = 24 * (day - trunc(day))
println
prtdate(month, trunc(day), year)
println
prtime(thours)
println
outprec 12
println "slant range = ", %drsaved, " kilometers"
```

Orbital Mechanics with Numerit

This function must be coded as `name(ti, tf, topt)` where `name` is the name of the user-defined function. The function parameters should be exactly as they're shown here.

The following is a typical draft output created with this program. It illustrates the first few close approaches. The initial calendar date was January 1, 1998, the initial universal time was 0,0,0 and the search interval was 10 days.

```
demoevt1
```

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

```
January 1, 1998
```

```
1 h 27 m 43.956 s
```

```
slant range = 9196.25906938 kilometers
```

```
January 1, 1998
```

```
3 h 43 m 3.34314 s
```

```
slant range = 7243.38698759 kilometers
```

```
January 1, 1998
```

```
5 h 47 m 35.5884 s
```

```
slant range = 5061.41431492 kilometers
```