Justin Wang
604219879

User Guide

A concise description of all the routines and applications: what they do, calling syntax, output format, and return values or messages.

Routines:

The bis function is a modification of the algorithm for the bisection method. It approximates the solution to a root finding problem by checking smaller intervals to apply the intermediate value theorem.

>bis(a,b,tolb,func,pars)
a and b are the endpoints of the initial interval
tolb is the error tolerance for the bisection method
func is the root solving problem expressed as a function set equal to 0
%func=@(x)(x-e*sin(x)-Mt); %Kepler's equation to find root E(t)
pars stores all constants in the Kepler's equation, Mt and e.

output format: [root a b]

return values:
the bis function returns root, which is the midpoint of the final computed interval [a,b]

The nwt function is a modification of the algorithm for Newton's method. This method provides faster convergence to the actual root when f'(p) does not equal 0.

>nwt(seed,toln,nmax,func,dfunc,pars)
seed is the initial approximation p0
toln is the error tolerance set for Newton's method
nmax is the maximum number of iterations
func is the root solving problem f(x)=0
dfunc is the first derivative of func
%func=@(x)(x-e*sin(x)-Mt); %Kepler's equation to find root E(t)
%dfunc=@(x)1-e*cos(x); %derivative of f
pars stores all constants in the Kepler's equation, Mt and e

output format: [root found iter]

return values:
the nwt function returns root, the approximation for the zeros of the equation
found is a boolean logical indicating whether the root is found
iter is the number of iterations actually used

The bisnwt function uses algorithm 1 of the project description to find a root of a function. It combines the bisection method and Newton's method.

>bisnwt(a,b,tolb,toln,nmax,func,dfunc,pars)
a and b are the endpoints of the initial interval
tolb is the error tolerance for the bisection method
toln is the error tolerance set for Newton's method
nmax is the maximum number of iterations
func is the root solving problem f(x)=0
dfunc is the first derivative of func
%func=@(x)(x-e*sin(x)-Mt); %Kepler's equation to find root E(t)
%dfunc=@(x)1-e*cos(x); %derivative of f
pars stores all constants in the Kepler's equation, Mt and e

output format: [root iter itern]

return values:
iter is the number of iterations used
itern is the number of iterations used in Newton's method

The application, tanom, solves the Kepler equation and gives the position of the satellite in its orbital plane.

>tanom(T,,e,n,varargin)
T : the period of the orbit in hours
e: the eccentricity of the orbit
n: the number of points where we want to localize the satellite on its orbit
the following arguments can optionally be passed through varargin
tolb : the error for the bisection method
toln ; the error for Newton's method
nmax: the maximum iterations that we allow to Newton's method

output format: [orbit]
orbit is a matrix of the format where the columns in order are:
ti, ei, vi, ri, xi, yi

Implementation:

The argument toln in the routine tanom.m represents the error that we allow in the approximation of the Kepler equation. If we want the error in the coordinates of the orbit to be smaller than 10^-6, we use toln=1e-16.

We could also change the computations to work with a vector x in Rn to find all the points of the orbit at once. It would be a faster way to compute a function for many values, because it avoids setting up each function evaluation.

Test:
First I verified that there were no issues with bis function, by testing a simple function, and setting pars=0. Then I tested nwt function similarly and found no erros.I debugged for an infinite loop that occurred in bisnwt.
In the console window type:

```
>> tanom(4,.25,100)

ans =

  1.0e+04 *

         0         0         0    0.9595    0.9595         0
    0.0000    0.0000    0.0000    0.9606    0.9550    0.1036
    0.0000    0.0000    0.0000    0.9639    0.9416    0.2063
    0.0000    0.0000    0.0000    0.9694    0.9196    0.3070
    0.0000    0.0000    0.0000    0.9770    0.8892    0.4050
…
```

Solutions:

Plot the orbits that you have obtained with your selected initial data and parameters.

```
>>orbit=tanom(4,.25,100)
>> plot(orbit(:,5),orbit(:,6))
```

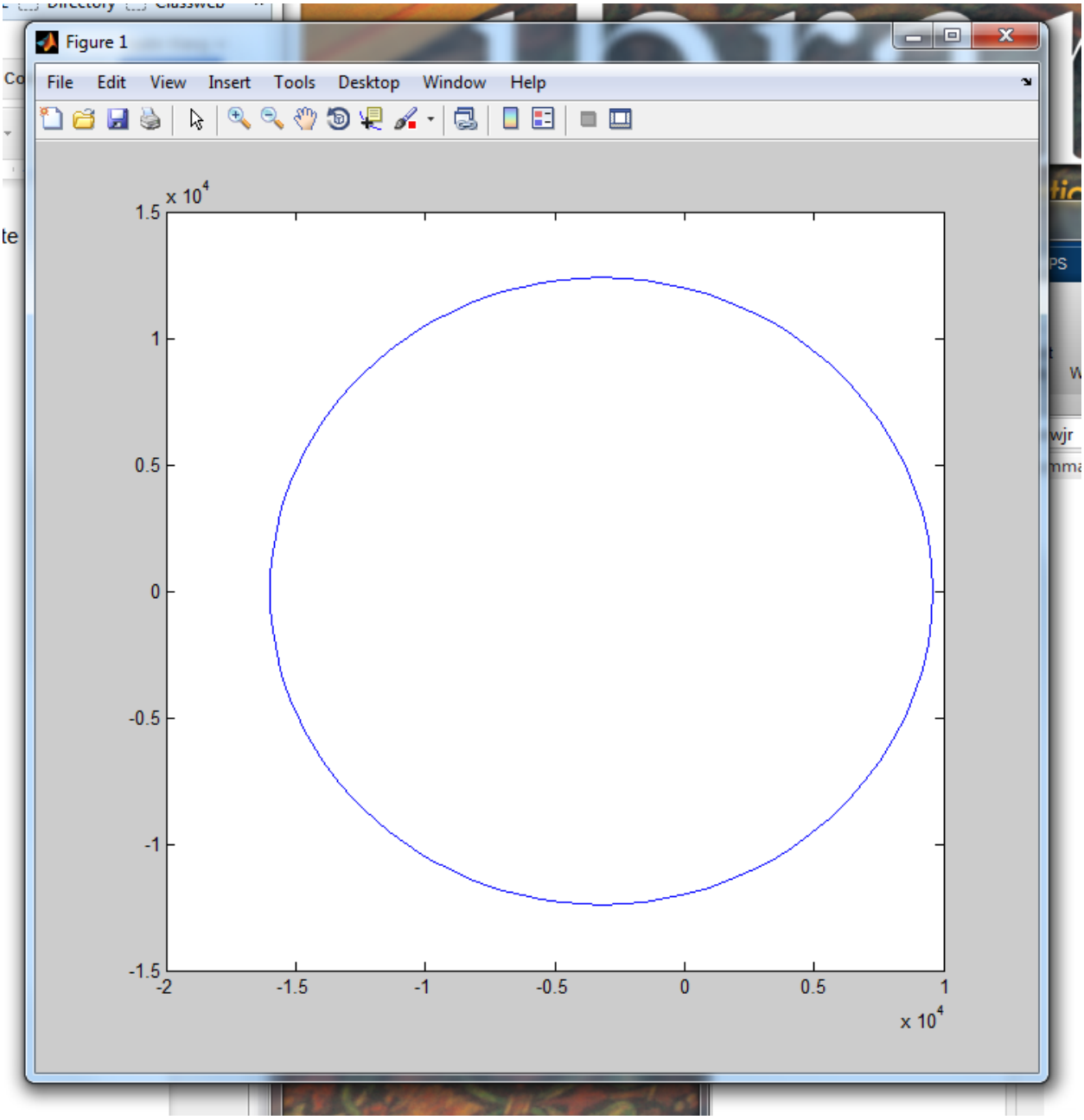
Figure 1
File Edit View Insert Tools Desktop Window Help
x 10^4
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1.5
-1
-0.5
0
0.5
1
x 10^4