

MAT 5610 – NUMERICAL ANALYSIS TERM PROJECT

Jeremy Morris, Issac Ben, Haimanot Kassa

General Introduction

This was a fascinating project for many of us who came across a real world application of nonlinear systems for the first time. Nonlinear systems are relatively more difficult than linear systems. Thus we must say that there were a number of difficulties that we faced. Among them, the one that took longer to debug was that while adjusting for time t in the satellite program, we forgot to save the old value by using “*temp*.” We want to thank our member, Jeremy Morris, for finding out this subtle bug. Also, while writing the receiver program, understanding Newton’s method for nonlinear systems by least squares approach took a considerable amount of time. Nonetheless, through hard work, this team was able to tackle most of such problems.

We also want to comment on that we found it very difficult to match this project with other literatures. There was a scarcity of books that are related to the project. The books in the library were all checked out and we were not able to get a reference for the project.

Satellite Specification

For the *satellite* program we have three objects that will do all the work. They are called *Houston* and *Satellite* and *CustomMath*. The *Houston* Object will keep track of all *Satellite* objects and be in charge of input and output. The *Satellite* program processes data in the following fashion:

1. It computes the position \mathbf{x}_v of the vehicle in Cartesian coordinate system at time t_v
2. For each of the 24 satellites, it returns the following data i_s, t_s, \mathbf{x}_s
3. It also writes a copy of the standard input and standard output to the file called *satellite.log*

Here is how our objects interact to carry this out:

1. *Houston* converts the latitude/longitude coordinates into Cartesian coordinates.
2. *Houston* sends \mathbf{x}_v and t_v to each satellite in turn
 1. each *satellite* computes \mathbf{x}_s and t_s to *Houston*
 2. they return \mathbf{x}_s and t_s to *Houston*
3. *Houston* decides if coordinates are above horizon

4. **Houston** writes to *stdout* the following: $i_s t_s x_s$; for the satellites it decides which are above the horizon at t_s
5. **Houston** then writes all of this to the log file.

The specifications for our two objects:

- **Houston**

- Methods

PrintLogFile writes the appropriate stuff to log file

Longlatt2cart takes in vector containing longitude/latitude data and sets XV with Cartesian coordinates. Also adjusts result for time.

AboveHorizon given the Cartesian coordinates of the satellite, it returns true if the satellite is above the horizon and for the values stored in XV

Constructor takes longitude/latitude data, stores it somehow, calls **longlatt2cart** to set XV , initializes satellites by getting data from *data.dat*, calling **new**, using **push_back** onto S .

- Data Members

XV array of doubles, stores Cartesian coordinates of vehicle

S vector of satellite objects

- **Satellite**

- Methods

dot takes two vectors, returns a dot product

getXs takes a value t and returns current position of satellite represents x_s in f and fp

getXsAtTs performs the following to calculate ts

setXsTs uses **newton**, f , fp to do Newton's method until converges and sets ts/xs

constructor takes in a vector containing values from **data.dot**, initializes the local data members accordingly, calls **setXsTs** to figure out **x**,

- Data Members

u taken from **data.dat** i.e. passed in from Houston

v taken from **data.dat** i.e. passed in from Houston

theta taken from **data.dat** i.e. passed in from Houston

p taken from **data.dat** i.e. passed in from Houston

h taken from **data.dat** i.e. passed in from Houston

Xs stores final time and position after doing Newton's method.

- CustomMath

- Methods

Deg2rad takes in degree, minute, second information and outputs the appropriate radian

dot takes two vectors, returns a dot product

Receiver Specification

The main task of this portion of our project is solving systems of nonlinear algebraic equations using Newton's method. Generally, we are trying to find an **X** that satisfies **F(X) = 0** where both **F** and **X** are vectors. In our case, **F(X)** is **GRAD(X)**.

We used least squares approach as in our first homework questions #13 and #14. Actually, the code we wrote is more or less the same as the answers given for these questions. (pp. 15-18).