Math 151A Project 2 Report

Justin J. Wang

3/10/2014

1. User Guide:

This is a routine to approximate the path of migrating birds, given points in time, which are the interpolation points, and x(t) and y(t), the parametric curves corresponding to the path followed by the member of the flock. We use two polynomial interpolation approaches, Lagrange, and Cubic Spline. The functions implemented are as follows:

divdif.m

```
function [P, Q]=divdif(x,y)
```

P stores all coefficients of Lagrange polynomial, which contains all the diagonal entries of F
Q is a matrix with entries Fi,j=[xi-j,...,xj]
implements algorithm 3.2 in the textbook
Newton's Divided-Difference Formula

evalP.m

function pt=evalP(P,t_int,t_test)

evaluates the polynomial with coefficients P associated with interpolation points t_int at the test points t_test and returns that value in pt.
task is to evaluate the polynomial P(x)=(x-x0)*Q(x)+b0 at x0.
work recursively backwards, starting at Fnn(x-x_n-1)

spline3.m

function S=spline3(t,ft,varargin)

computes the cubic splines corresponding
to the interpolation points {(ti,f(ti))} and stores all coefficients in S

To use CLAMPED, use format: spline3(t,ft, 'clamped', fa,fb)
Note: make sure to specify the derivative values at the endpoints, fa, and fb, or the code will not work properly.

construct NATURAL cubic spline interpolant S for the function f, defined at numbers xo,...,xn, satisfying S''(x0)=S''(xn)=0
By default when varargin is empty, we use the natural boundary condition without any declarations. So unless you specify the string, 'clamped' for the third argument input, the algorithm will run the natural cubic spline method.
S stores the vectors, ft, b, c, d, as columns of the matrix.

evalS.m

function st=evalS(S,t_int,t_test)

Uses a method to evaluate the spline with coefficients stored in S at the point(s) t_test and returns that value in st.
t_int is the vector of the interpolation points.

findpath.m

function [path, coeff]=findpath(ip,tp,method,varargin)

finds the path matrix of the form (1) and the coefficient matrix of the form (2) or (4). Here ip is a matrix of the form (1) storing all interpolation points, tp is a column vector storing t0,...tm-1, and method specifies the interpolation method, eg. strings "lagrange" and "spline"

Implementation:

Derive the linear system (3) based on the definition of the cubic spline. Compared to the linear systems in Page 149 and 155 of the textbook, which version is more stable?

Answer: The clamped cubic spline on page 155 of the textbook is more stable because, at the endpoints, the interpolating polynomial will not oscillate at the endpoints, as it would for the natural spline. This is why it is advantageous to use the clamped cubic spline. However, the clamped method requires inputs, FPO=f'(x0), and FPN=f'(xn), so you would have to evaluate derivatives at the endpoints.

Approximate the function f(x)= 1/(1+25x^2) on the interval [-1,1] using 10, 20, and 50 equidistant nodes, and Chebyshev nodes cos (2j-1)*pi/2n, j=1,...n, respectively. Compute absolute error associated to each approach and comment on the behavior of the error as we increase the number of nodes.

As we increase the number of nodes, the error should decrease because there is more data available to approximate.

Solutions:

The Final Trajectory is heart shaped.