**Appendix ~: Code**

function res = letsdoathing(nWaypoints, startPoint, endPoint)

% Sean Carter and Rebecca Jordan's Linearity II Optimization Project

% takes in a number of waypoints to place, and [x, y] start & end points.

% plots the fastest sailing path across a randomly generated windfield

% returns the time taken to reach endpoint

% set number of waypoints, start point, and end point

nWaypoints = nWaypoints;

startPoint = startPoint;

endPoint = endPoint;

% wangle is the direction the wind is COMING FROM

% wind speed is in m/s

wangle = 3\*pi/4 ;

wspeed = 2;

% x and y matrices are generated based on the wind speed and direction

% n is the number of values in the grid

% at any [x, y] point the x and y components of the wind vectors are

% the values at those indices in the x and y matrices respectively.

n = 30;

windX = makeWindFun(n,n);

windY = makeWindFun(n,n);

% generates a random squiggle sort of between the points as first guess

X0 = [];

for i = 1:nWaypoints

X0(i,1) = startPoint(1) + (rand\*(endPoint(1) - startPoint(1))/(nWaypoints+1)) \* i;

X0(i,2) = startPoint(2) + (rand\*(endPoint(2) - startPoint(2))/(nWaypoints+1)) \* i;

end

% run the optimization

[x, fval] = fmincon(@(Points)getTimeFromPoints(startPoint, endPoint, Points, windX, windY), X0, [], [], [], [], ones(nWaypoints, 2), Inf)

%append the start and end points to the list of waypoints

fullpath = [startPoint;x;endPoint];

%graphing code

clf

hold on

% initial guess

%plot(X0(:,1), X0(:,2),'g')

% wind vector

quiver(-1.\*windX, -1.\*windY)

% path, with dots at waypoints

plot(fullpath(:,1),fullpath(:,2),'r')

plot(fullpath(:,1),fullpath(:,2),'r.')

axis([10,20,10,20])

legend('Wind', 'Optimal Path')

axis equal

res = fval;

end

function res = getTimeFromPoints(startPoint, endPoint, Points, windX, windY)

% Objective function for letsdoathing.m

% takes in [x, y] start and end points, a list of waypoints,

% and matrices of the x and y components of wind at each point.

% append the start and end points to the matrix of waypoints

X = [startPoint;Points;endPoint];

% initialize time taken

totalTime = 0;

for step = 1:length(X)-1

% interpolate between each two consecutive points

interpolated\_path = makeMorePoints(X(step,:),X(step+1,:), 15);

% add time taken to sail this segment

totalTime = totalTime + speedOfPath(interpolated\_path);

% add penalty for tacking on all but the last step

if step < (length(X) - 1)

tackwindx = interp2(windX, X(step+1,1), X(step+1,2));

tackwindy = interp2(windY, X(step+1,1), X(step+1,2));

totalTime = totalTime + tacktime(X(step,:), X(step+1,:), X(step+2,:), tackwindx, tackwindy);

end

end

% return time taken

res = totalTime;

function res = makeMorePoints(p1, p2, n)

% returns a list of n points between [x,y] points p1 and p2

res =[];

for i = 1:n+2

res(i,1) = p1(1) + ((p2(1) - p1(1))/(n+1)) \* (i-1);

res(i,2) = p1(2) + ((p2(2) - p1(2))/(n+1)) \* (i-1);

end

end

function res = speedOfPath(path)

% takes in a list of waypoints

% returns the time it takes to sail that path

% not counting tacking delays

% use MATLAB's built in interpolation to find wind along the path

X\_wind = interp2(windX, path(:,1), path(:,2), 'linear');

Y\_wind = interp2(windY, path(:,1), path(:,2), 'linear');

% initialize time taken

time = 0;

for i = 1:length(path)-1

% determine relative wind angle

angle = abs(windangle(path(i,:), path(i+1,:), X\_wind(i), Y\_wind(i)));

% Finding the boat speed based on relative wind angle

speed = norm([X\_wind(i), Y\_wind(i)])\*(0.45\*cos(angle) + 0.55);

% add time for this leg to total time

time = time + distancebetweenpoints(path(i,:),path(i+1,:))/speed;

end

% return time taken

res = time;

end

function res = distancebetweenpoints(p1,p2)

% simple distance between two points calculation

dx = p2(1) - p1(1);

dy = p2(2) - p1(2);

res = norm([dx,dy]);

end

function res = windangle(p1,p2, wX, wY)

% given two [x,y] points and the wind X and Y values

% returns the angle between the wind and the boat heading

% from -pi to pi. starboard tack is positive.

pathx = p2(1) - p1(1);

pathy = p2(2) - p1(2);

wnd = atan2(wY,wX);

pth = atan2(pathy, pathx);

ang = pi - (wnd - pth);

%disp([wnd, pth, ang])

if ang > pi

ang = ang - 2\*pi;

end

res = ang;

end

function res = tacktime(p1, p2, p3, wx, wy)

% given three waypoints and the x and y wind components,

% determines whether the boat changes tack at p2

angle1 = windangle(p1, p2, wx, wy);

angle2 = windangle(p2, p3, wx, wy);

% starboard tack is a positive angle, port tack is negative.

% sign of angle changes if tack changes

if sign(angle1) == sign(angle2)

delay = 0;

else

delay = 15;

end

res = delay;

end

end

function Iwind = makeWindFun(SZX,SZY)

% This is just a helper function to make a random scalar function. This is

% called twice to generate a random wind field.

%

% Copyright (c) 2012, MathWorks, Inc.

%

windFineness = 0.1;

if ~exist('SZX','var')

SZX = 50;

SZY = 50;

end

N = 50; % Various parameters used in generating a random "smooth" matrix

NL = 40;

NP = 500;

rx = randn(NL,N);

rx = interpft(rx,NP);

ry = randn(NL,N);

ry = interpft(ry,NP);

I = (rx\*ry');

[xgi,ygi] = meshgrid(linspace(1,2 + 498\*windFineness,SZX+1),linspace(1,2 + 498\*windFineness,SZY+1));

Iwind = 10\*interp2(1:500,1:500,I,xgi,ygi);