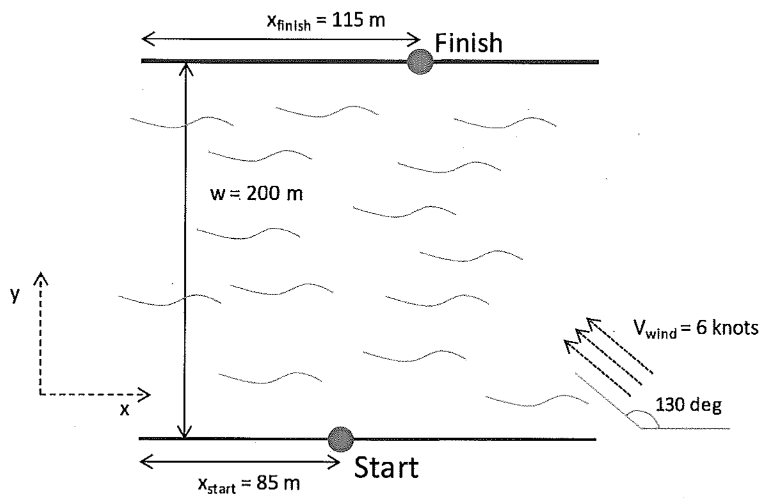
Optimal Control Qual Exam for Tim Coon
Take-Home (48 hours)

Numerical Optimization of a River Crossing:

Your task is to cross a river in minimum time using a sailboat (you may assume a point mass model). A diagram of the river crossing is shown in the below figure.



The river is flowing *from left to right* with $V_{river} = 4*V_{max}*y*(w-y)/w^2$, where V_{max} is 32 knots. Your solution to this problem must include a diagram of the crossing profile (e.g. path as a function of x and y) and a plot of the sail angle vs. time for the optimal path. You may not use GPOPS for this problem, but you may use any method within Matlab (e.g. *fmincon*, *fsolve*, etc.) that you deem appropriate. In addition to the above mentioned plots, please include:

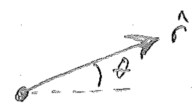
a) A formal statement of the discretized static optimization problem that you used to solve for your optimal path. Include your objective function and ALL constraints.
b) The augmented Lagrangian function. Write out each of the terms expanded as much as possible, but leave in terms of *i*. That is, expand the state terms but not time other than the first and last time steps as is traditional.
c) The similarly expanded Hamiltonian, if you did not expand as part of b).
d) Perform an analysis (experimenting is sufficient) to determine the value of V_{max} that results in an optimal path with the minimum crossing time (an optimum of optimums, so to speak). Why is the river velocity that results in this not zero?
e) What is the minimum crossing time if you are not required to finish at a specific location on the far bank (use V_{max} = 32 knots)? Which far bank location results in this minimum crossing time?
f) *Optional:* If you solve this problem by supplying the gradients to the optimizer in Matlab, you will automatically get full credit. You must also state why your code does not require the explicit calculation of the lagrange multiplier, v.

– b/c $(x-x_f)+(y-y_f)=0 \equiv (x-x_f)=0$ & $(y-y_f)=0$

INCLUDE ANY CODE YOU WRITE (comments are helpful)

b/c $(x-x_f) \le 0$ & $(y-y_f) \le 0$

Only Dr. Cobb or Maj. Dillsaver may answer questions. Open book, (your) open notes. You may use any code from Mech 622 that you consider helpful. Do not use the internet.



SAIL

WIND

a) FORMAL STATEMENT OF OPTIMIZATION PROBLEM

MINIMIZE: $J = t_f$

SUBJECT TO: $x_{i+1} = x_i + V_{x_i}\Delta$

$y_{i+1} = y_i + V_{y_i}\Delta$

BOUNDARY COND.: $x_0 = 85\,m \qquad y_0 = 0$

$x_f = 115\,m \qquad y_f = 200\,m$

$0 \le y_i \le 200\,m \quad \forall\, t_i$

$\longrightarrow \; V_{x_i} = c_1 V_{w/r_i} \sin(\theta_i) + V_{river}(y_i)$

$\Rightarrow V_{w/r} = V_{w_x} - V_{river}(y_i)$

$\Rightarrow V_{w_x} = V_w \cos(A_w)$

$\longrightarrow \; V_{y_i} = C_2 V_{w_y} \cos(\theta_i)$

$\Rightarrow V_{w_y} = V_w \sin(A_w)$

SEE BRYSON 4.3

SOLVE USING AN INDIRECT NUMERICAL METHOD.

COMPARE TO BOLZA FORM

COST: $J = \phi(\bar{x}(N), \Delta) + \sum_{i=0}^{N-1} L(\bar{x}(i), \bar{u}(i), \Delta)$

STATE EQNS: $\bar{x}(i+1) = \bar{f}(\bar{x}(i), \bar{u}(i), \Delta) \qquad \bar{x}(0) = \bar{x}_0$

TERM CONST: $\bar{\psi}(\bar{x}(N), \Delta) = 0$

THUS, $\phi = N\Delta \qquad L = 0 \qquad \bar{x} = \begin{bmatrix} x \\ y \end{bmatrix} \qquad f = \begin{bmatrix} x + V_x\Delta \\ y + V_y\Delta \end{bmatrix} \qquad \bar{\psi} = \begin{bmatrix} x(N) - x_f \\ y(N) - y_f \end{bmatrix}$

EXPAND STATE TERMS, BUT NOT TIME OTHER THAN FIRST AND LAST STEPS

$\bar{J}$ ≡ AUG. LAGRANGIAN
$\bar{\Phi}$ ≡ TERMINAL COST (AUG.)
$\bar{x}$ ≡ STATE VARIABLE VECTOR
Δ ≡ TIME STEP
$\bar{\lambda}$ ≡ LAGRANGE VECTOR
ϕ ≡ TERMINAL COST
$\bar{\nu}$ ≡ TERMINAL LAGRANGE VECTOR
$\bar{\psi}$ ≡ TERMINAL CONSTRAINTS
H ≡ HAMILTONIAN
L ≡ RUNNING COST FUNCTION
$\bar{f}$ ≡ STATE EQNS

$\lambda^x_{(i+1)}(x_{(i)} + V_x(i)\Delta)$

b) THE AUGMENTED LAGRANGIAN FUNCTION.

GENERAL FORM

$$\bar{J} = \Phi(\bar{x}(N), \Delta) - \bar{\lambda}^T(N)\bar{x}(N) + \bar{\lambda}^T(0)\bar{x}_0 + \sum_{i=0}^{N-1}\left[H(\bar{x}_{(i)}, \bar{u}_{(i)}, \Delta) - \bar{\lambda}^T_{(i)}\bar{x}_{(i)}\right] \quad (2.69)$$

$$\Rightarrow \Phi(\bar{x}_{(N)}, \Delta) = \phi(\bar{x}(N), \Delta) + \bar{\nu}^T\bar{\psi}(\bar{x}_{(N)}, \Delta)$$

$$\Rightarrow H(\bar{x}_{(i)}, \bar{u}_{(i)}, \Delta) = \cancel{L}^{0}(\bar{x}(i), \bar{u}(i), \Delta) + \lambda^T(i+1)\bar{f}(x(i), u(i), \Delta)$$

$$\bar{J} = N\Delta + \nu^x(x(N) - x_f) + \nu^y(y(N) - y_f) - \lambda^x(N)x(N) - \lambda^y(N)y(N) + \lambda^x(0)x_0 + \lambda^y(0)y_0 + \sum_{i=0}^{N-1}\left[\lambda^x_{(i+1)}x(i+1) + \lambda^y_{(i+1)}y(i+1) - \lambda^x(i)x(i) - \lambda^y(i)y(i)\right]$$

$$\bar{J} = N\Delta + \nu^x(x(N) - x_f) + \nu^y(y(N) - y_f) - \lambda^x(N)x(N) - \lambda^y(N)y(N) + \lambda^x(0)x_0 + \lambda^y(0)y_0 + \sum_{i=0}^{N-1}\left[\lambda^x_{(i+1)}(x_{(i)} + V_x(i)\Delta) + \lambda^y_{(i+1)}(y_{(i)} + V_y(i)\Delta) - \lambda^x(i)x(i) - \lambda^y(i)y(i)\right]$$

c) EXPANDED HAMILTONIAN

$$H(\bar{x}_{(i)}, \bar{u}_{(i)}, \Delta) = \lambda^x_{(i+1)}(x_{(i)} + V^x(i)\Delta) + \lambda^y_{(i+1)}(y_{(i)} + V^y(i)\Delta)$$

COSTATE: $H_{\bar{x}}(i) - \lambda^T(i) = 0$

$$\hookrightarrow \lambda^T(i) = \lambda^T(i+1)f_{\bar{x}}(i) = \lambda^T_{(i+1)}\begin{bmatrix} f^x_x & f^x_y \\ f^y_x & f^y_y \end{bmatrix}(i)$$

$$\hookrightarrow f^x_x = 1 \quad f^x_y = [c_1\sin(\theta) + 1]\,\partial(V_{wind})/\partial y$$
$$f^y_x = 0 \quad f^y_y = 1$$

CONTROL: $H_{u(i)} = 0$

$$\hookrightarrow \lambda^T_{(i+1)}f_\theta(i) = 0 = \begin{bmatrix} \lambda^x_{(i+1)} & \lambda^y_{(i+1)} \end{bmatrix}\begin{bmatrix} f^x_\theta \\ f^y_\theta \end{bmatrix}$$

$$0 = \lambda^x_{(i+1)}[c_1 V_{wx/r}(i)\cos(\theta_{(i)})\Delta] - \lambda^y_{(i+1)}[c_2 V_{wy}\sin(\theta_i)\Delta]$$

NUMERICAL SOLUTION WITH GRADIENT METHOD

1) GUESS $u = [\theta \,;\, dt]$ & t_f

2) FORWARD SEQUENCE THE STATE EQNS

3) CALCULATE FINAL COSTATE $\lambda^{\phi}(N) = \phi_x^T \quad \lambda^{\psi}(N) = \psi_x^T$

4) BACKWARD SEQUENCE COSTATES AND STORE PULSE RESPONSE SEQUENCES, $H_u^{\phi}(i)$ & $H_u^{\psi}(i)$ $i = 0, \ldots, N-1$, AND GRADIENTS, $\bar{\phi}_\Delta$, $\bar{\psi}_\Delta$

$$\bar{\phi}_\Delta = \phi_\Delta + \sum_{i=0}^{N-1} H_\Delta^{\phi}(i) \qquad \bar{\psi}_\Delta = \psi_\Delta + \sum_{i=0}^{N-1} H_\Delta^{\psi}$$

$$[H_u^{\phi}(i)]^T = f_u^T(i)\, \lambda^{\phi}(i+1) \qquad [H_u^{\psi}]^T = f_u^T(i)\, \lambda^{\psi}(i+1)$$

5) fmincon() DETERMINES A SEARCH DIRECTION FOR THE CONTROL VECTOR USING A STEEPEST DESCENT METHOD, THEN REPEATS THE ABOVE STEPS GIVEN THE NEW CONTROL VECTOR. ONCE ALL FIRST-ORDER OPTIMALITY CRITERIA AND CONSTRAINTS ARE SATISFIED TO WITHIN THE SPECIFIED TOLERANCES, A FEASIBLE OPTIMAL SOLUTION IS OBTAINED.

d) USING $V_{max} = [0 : 0.1 : 2]$, THE MIN CROSSING TIME IS FOUND USING THE NUMERICAL SOLUTION WITH GRADIENT METHOD AT EACH VALUE OF V_{max} AND THE REULT IS PLOTTED. THERE IS CLEARLY ONLY ONE MINIMUM. IT IS NOT AT $V_{max} = 0$ BECAUSE THE RIVER MOTION CAN BE UTILIZED TO MOVE THE REQUIRED 30m DOWNRIVER.

$$t_f = 120.9\,s \qquad V_{max} = 0.7 \text{ Knots}$$

e) THE MIN CROSSING TIME IS FOUND, AGAIN USING NUMERICAL METHOD, WHEN $X(N)$ IS NOT CONSTRAINED. THE RESULTING PATH AND CONTROL IS PLOTTED.

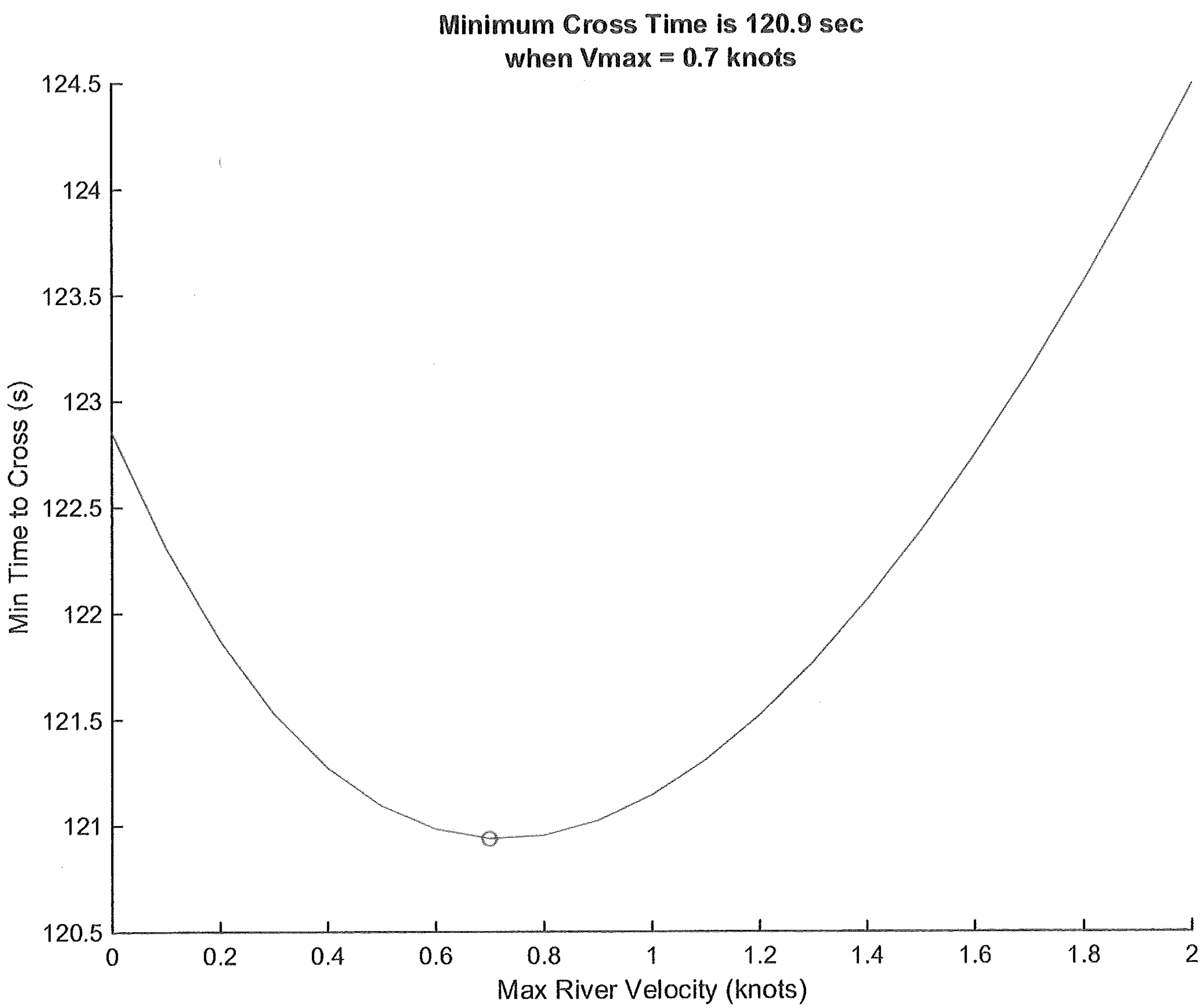
$$t_f = 169.3\,s \qquad X_f = 1941\,m$$

f) THE PROBLEM IS SOLVED WITH ORIGINAL V_{max} AND X_f WHILE SUPPLYING THE GRADIENTS OF THE COST TO fmincon(). ATTEMPTS TO SUPPLY GRADIENTS OF THE CONSTRAINTS WERE UNSUCCESSFUL BECAUSE I DID NOT KNOW HOW TO CALCULATE THE GRADIENTS OF THE INEQUALITY CONSTRAINTS. THIS RESULT IS PLOTTED. (IN FACT, COST GRADIENT WAS SUPPLIED IN ALL FOREGOING SIMULATIONS).

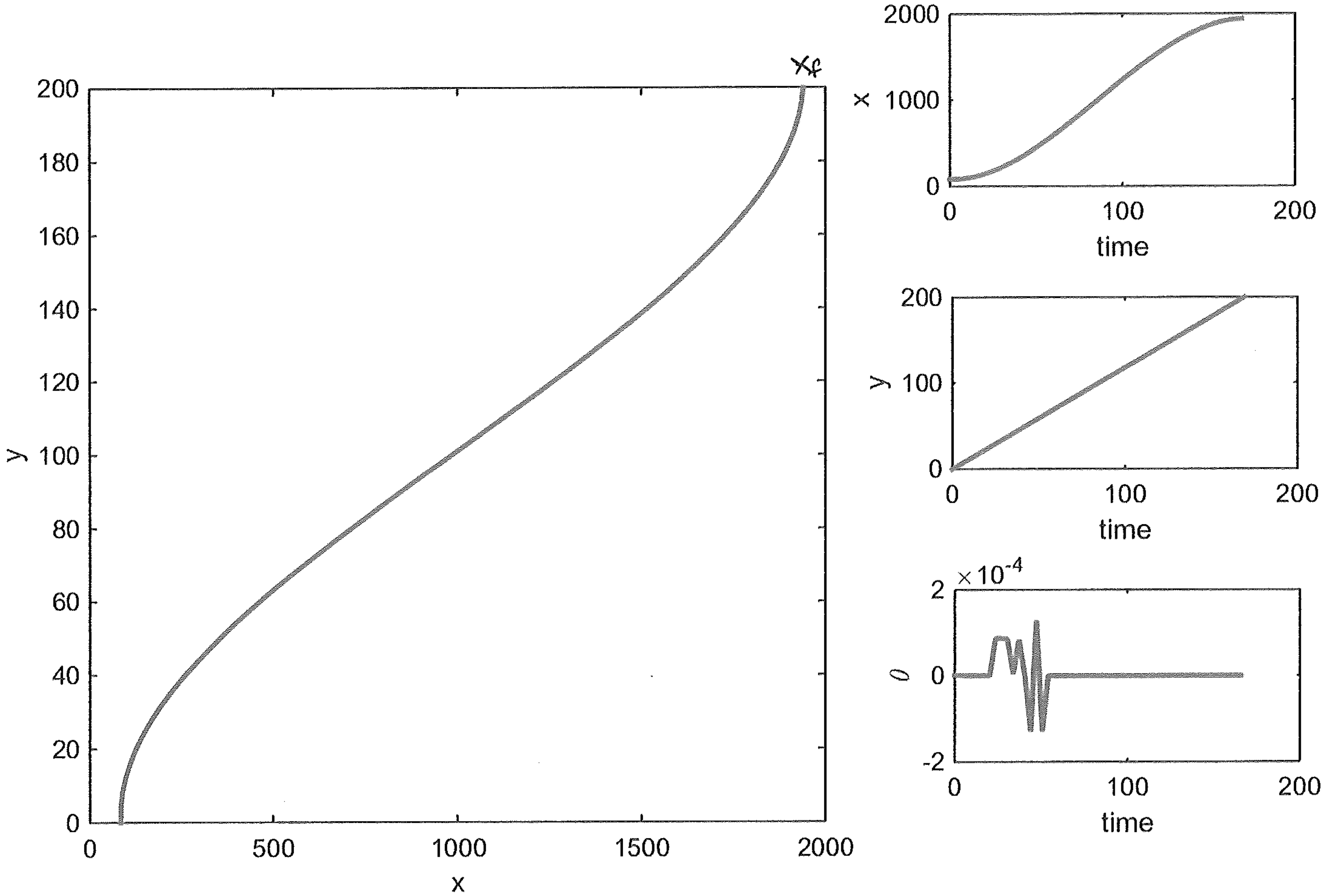
*COULD USE SLACK VARIABLES

$$t_f = 752.5\,s$$

* THIS CODE SET IS ATTACHED


Minimum Cross Time is 120.9 sec
when Vmax = 0.7 knots
Min Time to Cross (s)
Max River Velocity (knots)
124.5
124
123.5
123
122.5
122
121.5
121
120.5
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

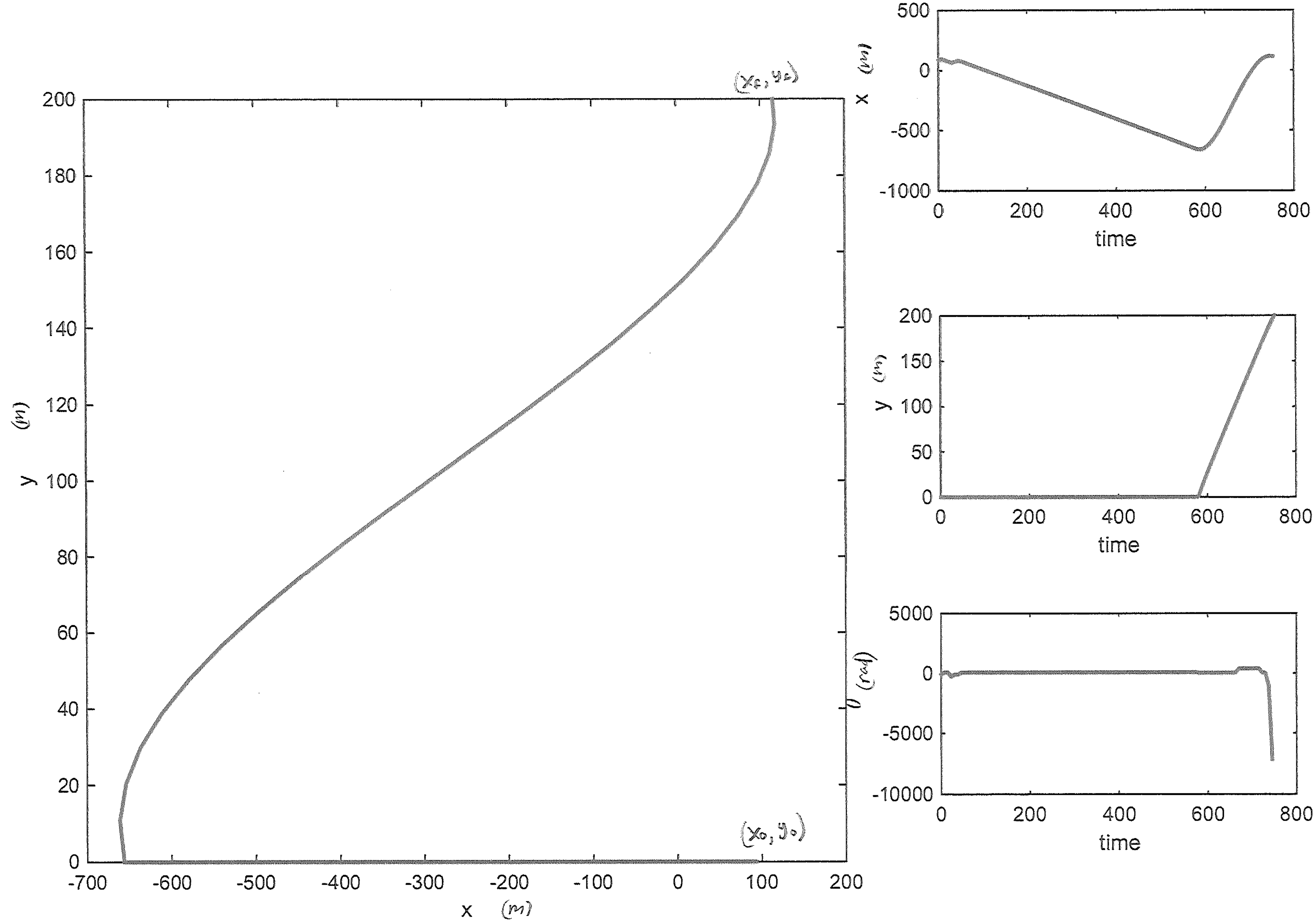
Min Cross Time with Free xf



t_f =169.3135 s

$X_f = 1941$ m

Sailboat Min Time Solution (GradObj)



$t_f = 752.4918$ s

```
% Tim Coon
% Qualifying Exam Question #2, MECH 622
% 01/26/2014
% Adapted from Bryson P4_3_5
% Solve the sailboat problem by an indirect numerical method wherein the
% control vector is guessed, then the states are propagated forward, the
% terminal constraints are used to find final costates, then the costates
% are propagated backwards to find the derivative of the cost function wrt
% the control at each time step. These derivatives are used by the gradient
% search optimization algorithm of fmincon() to determine the search
% direction that reduces the cost function. (Ref Bryson Sec 3.2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Simplified Model - Formal Statement of the Optimization Problem
% Minimize:      J = tf
% State Eq:      x(i+1) = x(i) + Vx(i)*dt
%                y(i+1) = y(i) + Vy(i)*dt
%                Vx(i) = c1*Vwxr(i)*sin(th(i)) + Vriver(y(i))
%                    Vwxr(i) = Vwx - Vriver(y(i))
%                    Vwx = Vw*cos(Aw)
%                    Vwy = c2*Vwy*cos(th(i))
% Constraints:   x(0) = x0  y(0) = y0
%                x(N) = xf  y(N) = yf
%                0 <= y(i) <= w   i = 1,2,3,...,N
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear; close all; clc;

%------------ given ------------------------
global c1 c2 xf yf Vmax w Vwx Vwy
Vwind = 6*0.514;            % (m/s)
Awind = deg2rad(130);       % (rad)
Vwx = Vwind*cos(Awind);     % (m/s)
Vwy = Vwind*sin(Awind);     % (m/s)
Vmax = 32*0.514;            % (m/s) max river velocity
% Vmax = Vr_ms(i);
w = 200;     % (m)
c1 = 0.7;    % coefficient for Vx
c2 = 0.7;    % coefficient for Vy
%---final states (posiion)
xf=115;      % (m)
yf=w;        % (m)
%---Set initial states [u x v y]
x0 = 85;     % (m)
y0 = 0;      % (m)
ns=2;  % number of state variables

%------------ guesses ------------------------
%---Set initial guess for final time
tf0=100;
%---Set number of time steps
N=100;
```

```
%---Set initial guess for control input (just try all zeros)
nc=1;                  % number of control inputs
th0=(pi/2)*ones(N,1);      % theta is the sail angle wrt global +x
%---Set initial guess for states
s0=[linspace(x0,xf,N); linspace(y0,yf,N)];

%------------ nlp setup -----------------------
%---Set number of equality and inequality constraints
ncons=2;         % total number of constraints (neqcons+nineqcons)
neqcons=2;       % number of equality constraints
%---Define design vector, independent design variables in fmincon
% [theta deltaT]
u0=[th0; tf0/N];
% u0_data = load('u0_guess_N=100.mat');
% u0 = u0_data.u;

%---Define a vector holding all of the dimensions
dims=[N ns nc ncons];
name='dvdp';
% bounds on the input
lb = Awind-(3*pi/4); ub = Awind-(pi/2);
% Set options for fmincon
options = optimset('GradObj','on','Display','Iter','GradConstr','off','MaxIter',
1e6,'MaxFunEvals',1e7);
u = fmincon(@(u0)Obj_MECH622_Qual(u0,s0,name,dims),u0,[],[],[],[],[],[],...
                         @(u0)Constr_MECH622_Qual(u0,s0,name,dims),options);

%% Retrieve state vector at optimum, organize for plotting
[f,g,s]=Obj_MECH622_Qual(u,s0,name,dims);
x=s(1,:);
y=s(2,:);
th=[u(1:N)]*180/pi;
tf=N*u(N+1);
t=tf*[0:1/N:1];

%% Plot
figure(1)
suptitle('Sailboat Min Time Solution (GradObj)')
subplot(4,6,[1 2 3 4, 7 8 9 10, 13 14 15 16, 19 20 21 22])
plot(x,y,'linewidth',2)
xlabel('x'); ylabel('y')
axis square
subplot(4,6,5:6)
plot(t,x,'linewidth',2)
xlabel('time'); ylabel('x');
subplot(4,6,11:12)
plot(t,y,'linewidth',2)
xlabel('time'); ylabel('y');
subplot(4,6,17:18)
plot(t(1:end-1),th,'linewidth',2)
xlabel('time'); ylabel('\theta');
```

```
ax = subplot(4,6,23:24);
set(ax,'visible','off')
t_f = strcat('t_f = ', num2str(tf,'%6.4f'));
ht = text(0.2,0.3,t_f);
ht.FontSize = 16;
```

```
function [c,ceq,GC,GCeq]=Constr_MECH622_Qual(ut,s0,name,dims)
% contraints for MECH 622 Qual
% s0 = initial state vector
% name = name of function to propagate and other calcs
% dims = vector of pertinent dimensions

N=dims(1);
ns=dims(2);
nc=dims(3);
ncons=dims(4);
nt1=ncons+1;
N1=N+1;
s=zeros(ns,N1);
la=zeros(ns,nt1);
Hu=zeros(nt1,nc,N);
s(:,1)=s0(:,1); n2=[2:nt1];
dt=ut(end);
tf=dt*N;
% Put u in matrix form, nc rows, N columns
for i=1:N
    u(1:nc,i)=ut((i-1)*nc+1:i*nc);
end
% Forward sequencing and store state histories, s:
for i=1:N
    s(:,i+1)=feval(name,u(:,i),s(:,i),dt,(i-1)*dt,1);
end
% Performance index, terminal constraints & gradients:
[Phi,Phis,Phid]=feval(name,zeros(nc,1),s(:,N1),dt,tf,2);
phi=Phi(1);     % cost function
psi=Phi(n2);    % terminal constraints
la=Phis';       % lecture 14, page 3
% Backward sequencing and store Hu(:,:,i):
for i=N:-1:1
    [fs,fu,fd]=feval(name,u(:,i),s(:,i),dt,(i-1)*dt,3);
    Hu(:,:,i)=la'*fu;
    Phid=Phid+la'*fd;
    la=fs'*la;
end
%---separate into phi and psi derivative terms
% phid=Phid(1);
% psid=Phid(n2);
%---pull out initial Hu at first time point
Humatrix=Hu(:,:,1);
%---organize into columns (each column is a time step)
for i=2:N,
    Humatrix=[Humatrix Hu(:,:,i)];
end
Humatrix=[Humatrix Phid];
Huphid=Humatrix(1,:)';  %---this is the derivative of cost function
% (total time) w.r.t. theta (at each time step) and delta_t
% note that derivative w.r.t. delta_t is equal to N
```

```
Hupsid=[Humatrix(n2,:)]'; %---this is the derivative of terminal position

% constraints w.r.t. theta (at each time step) and delta_t
y = s(2,:);
c = [-y'; y'-200];   % inequality constraint vector(c <= 0)
ceq = psi;           % equality constraint vector (ceq == 0)
GC = [];
GCeq = Hupsid;
```

```
function [phi,Huphid,s]=Obj_MECH622_Qual(ut,s0,name,dims)
% ut = vector of controls
% s0 = initial state vector
% name = name of function to propagate and other calcs
% dims = vector of pertinent dimensions

N=dims(1);
ns=dims(2);
nc=dims(3);
ncons=dims(4);
nt1=ncons+1;
N1=N+1;
s=zeros(ns,N1);
la=zeros(ns,nt1);
Hu=zeros(nt1,nc,N);
s(:,1)=s0(:,1); n2=[2:nt1];
dt=ut(end);
tf=dt*N;
% Put u in matrix form, nc rows, N columns
for i=1:N
    u(1:nc,i)=ut((i-1)*nc+1:i*nc);
end
% Forward sequencing and store state histories, s:
for i=1:N
    s(:,i+1)=feval(name,u(:,i),s(:,i),dt,(i-1)*dt,1);
end
% Performance index, terminal constraints & gradients:
[Phi,Phis,Phid]=feval(name,zeros(nc,1),s(:,N1),dt,tf,2);
phi=Phi(1);     % cost function
psi=Phi(n2);    % terminal constraints
la=Phis';       % lecture 14, page 3
% Backward sequencing and store Hu(:,:,i):
for i=N:-1:1
    [fs,fu,fd]=feval(name,u(:,i),s(:,i),dt,(i-1)*dt,3);
    Hu(:,:,i)=la'*fu;
    Phid=Phid+la'*fd;
    la=fs'*la;
end
%---separate into phi and psi derivative terms
phid=Phid(1);
psid=Phid(n2);
%---pull out initial Hu at first time point
Humatrix=Hu(:,:,1);
%---organize into columns (each column is a time step)
for i=2:N,
    Humatrix=[Humatrix Hu(:,:,i)];
end
Humatrix=[Humatrix Phid];
Huphid=Humatrix(1,:)';  %---these are the derivatives of cost function

% (total time) w.r.t. theta (at each time step) and delta_t
```

```
% note that derivative w.r.t. delta_t is equal to N

% Hupsid=[Humatrix(n2,:)]'; %---this is the derivative of terminal position
% constraints w.r.t. theta (at each time step) and delta_t
```

```
function [f1,f2,f3]=dvdp(u,s,dt,tf,flg)
% Discete Velocity Direction Programming for min tf to specified xf,yf
% s=[x y]'; [th]=control;
%
% Modified by Coon 11/2014
%
global c1 c2 xf yf Vmax w Vwx Vwy
x=s(1);
y=s(2);
th = u;
Vriver = calcVriver(y);
dVr_dy = 4*Vmax*(w-2*y)/w^2;
Vwxr = Vwx - Vriver;            % Vel of wind wrt river

if flg==1,      % f1=f(x,th,tf)
    % calculate the next state vector
    f1 = s + [c1*Vwxr*sin(th)+Vriver; c2*Vwy*cos(th)]*dt;
elseif flg==2, % f1=[phi; psi], f2=[phis; psis]', f3=[phid; psid];
    % f1 = [perf index; term constr]; f2 = state grads; f3 = dt grads
    f1=[tf; x-xf; y-yf];
    f2=[0 0; 1 0; 0 1];       % [phi_x phi_y; psix_x psix_y; psiy_x psiy_y]
    f3=[tf/dt 0 0]';          % [Phi_d]
elseif flg==3, % f1=f_s, f2=f_u, f3=f_dt;
    f1=[1 (-c1*sin(th)+1)*dVr_dy*dt; 0 1];
    f2=dt*[c1*Vwxr*cos(th); -c2*Vwy*sin(th)];
    f3=[c1*Vwxr*sin(th)+Vriver; c2*Vwy*cos(th)];
end
```