6.3 Path Planning for a Simple Car

In [ ]:

```
!pip install -q pyomo
!wget -N -q "https://ampl.com/dl/open/ipopt/ipopt-linux64.zip"
!unzip -o -q ipopt-linux64

ipopt_executable = '/content/ipopt'
```

Pyomo Model

In [2]:

```
from pyomo.environ import *
from pyomo.dae import *

L = 2
tf = 50

# create a model object
m = ConcreteModel()

# define the independent variable
m.time = ContinuousSet(bounds=(0, tf))

# define control inputs
m.a = Var(m.time)
m.v = Var(m.time, domain=Reals, bounds=(-0.1,0.1))

# define the dependent variables
m.x = Var(m.time)
m.y = Var(m.time)
m.t = Var(m.time)
m.u = Var(m.time)
m.p = Var(m.time, domain=Reals, bounds=(-0.5,0.5))

m.dxdt = DerivativeVar(m.x)
m.dydt = DerivativeVar(m.y)
m.dtdt = DerivativeVar(m.t)
m.dudt = DerivativeVar(m.u)
m.dpdt = DerivativeVar(m.p)

# define the differential equation as a constraint
m.ode_x = Constraint(m.time, rule=lambda m, time: m.dxdt[time] ==
m.u[time]*cos(m.t[time]))
m.ode_y = Constraint(m.time, rule=lambda m, time: m.dydt[time] ==
m.u[time]*sin(m.t[time]))
m.ode_t = Constraint(m.time, rule=lambda m, time: m.dtdt[time] ==
m.u[time]*tan(m.p[time])/L)
m.ode_u = Constraint(m.time, rule=lambda m, time: m.dudt[time] == m.a[time])
m.ode_p = Constraint(m.time, rule=lambda m, time: m.dpdt[time] == m.v[time])

# path constraints
m.path_x1 = Constraint(m.time, rule=lambda m, time: m.x[time] >= 0)
m.path_y1 = Constraint(m.time, rule=lambda m, time: m.y[time] >= 0)

# initial conditions
m.ic = ConstraintList()
m.ic.add(m.x[0]==0)
m.ic.add(m.y[0]==0)
m.ic.add(m.t[0]==0)
```

```
m.ic.add(m.u[0]==0)
m.ic.add(m.p[0]==0)

# final conditions
m.fc = ConstraintList()
m.fc.add(m.x[tf]==0)
m.fc.add(m.y[tf]==20)
m.fc.add(m.t[tf]==0)
m.fc.add(m.u[tf]==0)
m.fc.add(m.p[tf]==0)

# define the optimization objective
m.integral = Integral(m.time, wrt=m.time,
                      rule=lambda m, time: 0.2*m.p[time]**2 + m.a[time]**2 + m.v[time]**
2)
m.obj = Objective(expr=m.integral)

# transform and solve
TransformationFactory('dae.collocation').apply_to(m, wrt=m.time, nfe=3, ncp=12, method=
'BACKWARD')
solver = SolverFactory('ipopt', executable=ipopt_executable)
solver.solve(m).write()
```

```
# ==========================================================
# = Solver Results                                         =
# ==========================================================
# ----------------------------------------------------------
#   Problem Information
# ----------------------------------------------------------
Problem:
- Lower bound: -inf
  Upper bound: inf
  Number of objectives: 1
  Number of constraints: 449
  Number of variables: 444
  Sense: unknown
# ----------------------------------------------------------
#   Solver Information
# ----------------------------------------------------------
Solver:
- Status: ok
  Message: Ipopt 3.12.8\x3a Optimal Solution Found
  Termination condition: optimal
  Id: 0
  Error rc: 0
  Time: 1.157731294631958
# ----------------------------------------------------------
#   Solution Information
# ----------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
```

Accessing Solution Data

In [ ]:

```
# access the results
time = [time for time in m.time]

a = [m.a[time]() for time in m.time]
v = [m.v[time]() for time in m.time]

x = [m.x[time]() for time in m.time]
y = [m.y[time]() for time in m.time]
t = [m.t[time]() for time in m.time]
u = [m.u[time]() for time in m.time]
p = [m.p[time]() for time in m.time]
```

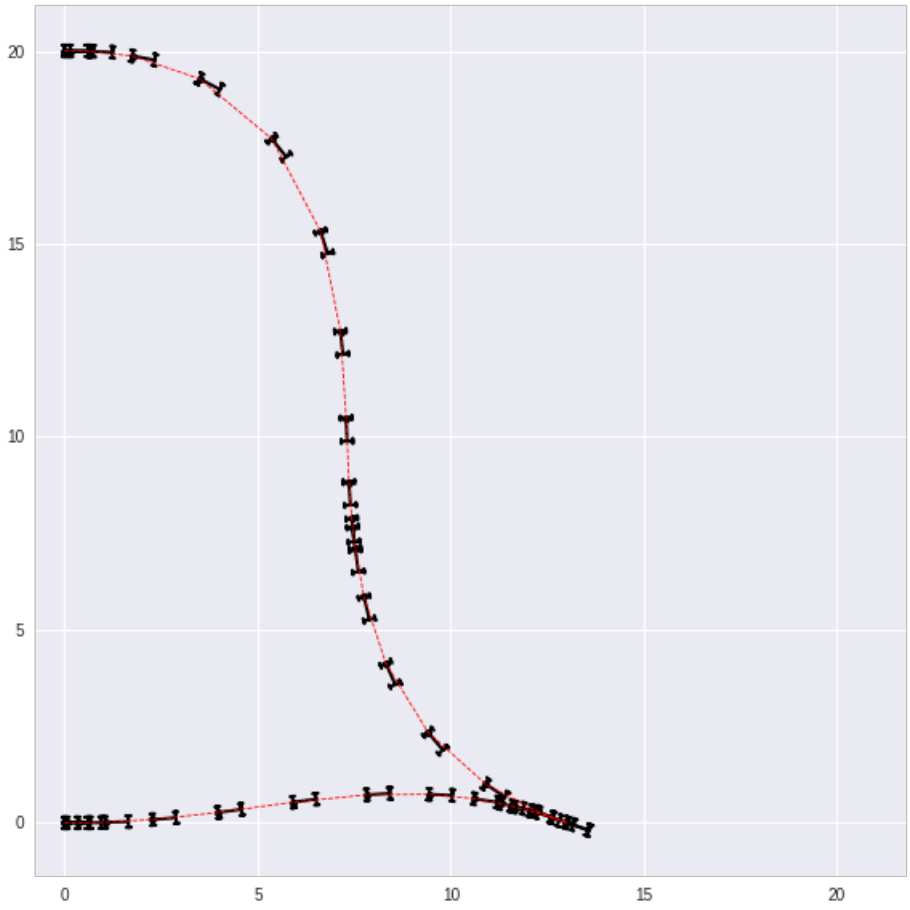
Visualizing the Car Path

In [4]:

```
% matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

scl=0.3

def draw_car(x=0, y=0, theta=0, phi=0):
    R = np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])
    car = np.array([[0.2, 0.5], [-0.2, 0.5], [0, 0.5], [0, -0.5],
                    [0.2, -0.5], [-0.2, -0.5], [0, -0.5], [0, 0], [L, 0], [L, 0.5],
                    [L + 0.2*np.cos(phi), 0.5 + 0.2*np.sin(phi)],
                    [L - 0.2*np.cos(phi), 0.5 - 0.2*np.sin(phi)], [L, 0.5],[L, -0.5],
                    [L + 0.2*np.cos(phi), -0.5 + 0.2*np.sin(phi)],
                    [L - 0.2*np.cos(phi), -0.5 - 0.2*np.sin(phi)]])
    carz= scl*R.dot(car.T)
    plt.plot(x + carz[0], y + carz[1], 'k', lw=2)
    plt.plot(x, y, 'k.', ms=10)

plt.figure(figsize=(10,10))
for xs,ys,ts,ps in zip(x,y,t,p):
    draw_car(xs, ys, ts, scl*ps)
plt.plot(x, y, 'r--', lw=0.8)
plt.axis('square')
```

Out[4]:

```
(-0.7475794562067691,
 21.812257339073334,
 -1.3839156648530575,
 21.175921130427046)
```

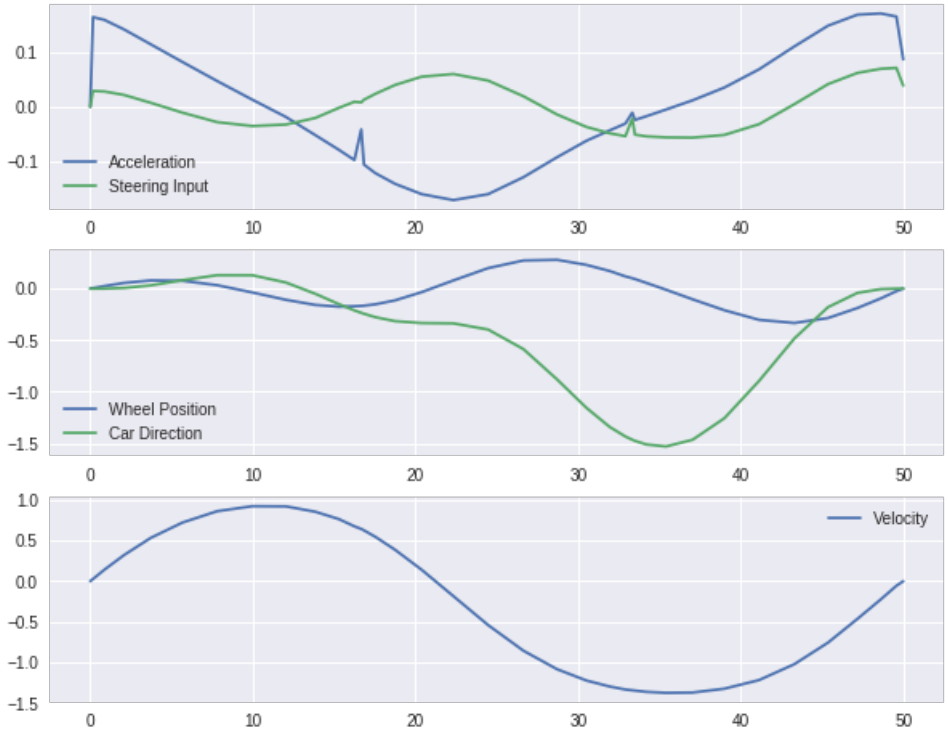


In [5]:

```
plt.figure(figsize=(10,8))
plt.subplot(311)
plt.plot(time, a, time, v)
plt.legend(['Acceleration','Steering Input'])

plt.subplot(312)
plt.plot(time, p, time, t)
plt.legend(['Wheel Position','Car Direction'])

plt.subplot(313)
plt.plot(time, u)
plt.legend(['Velocity'])
```

Out[5]:

```
<matplotlib.legend.Legend at 0x7ff2c45977b8>
```



In [ ]: