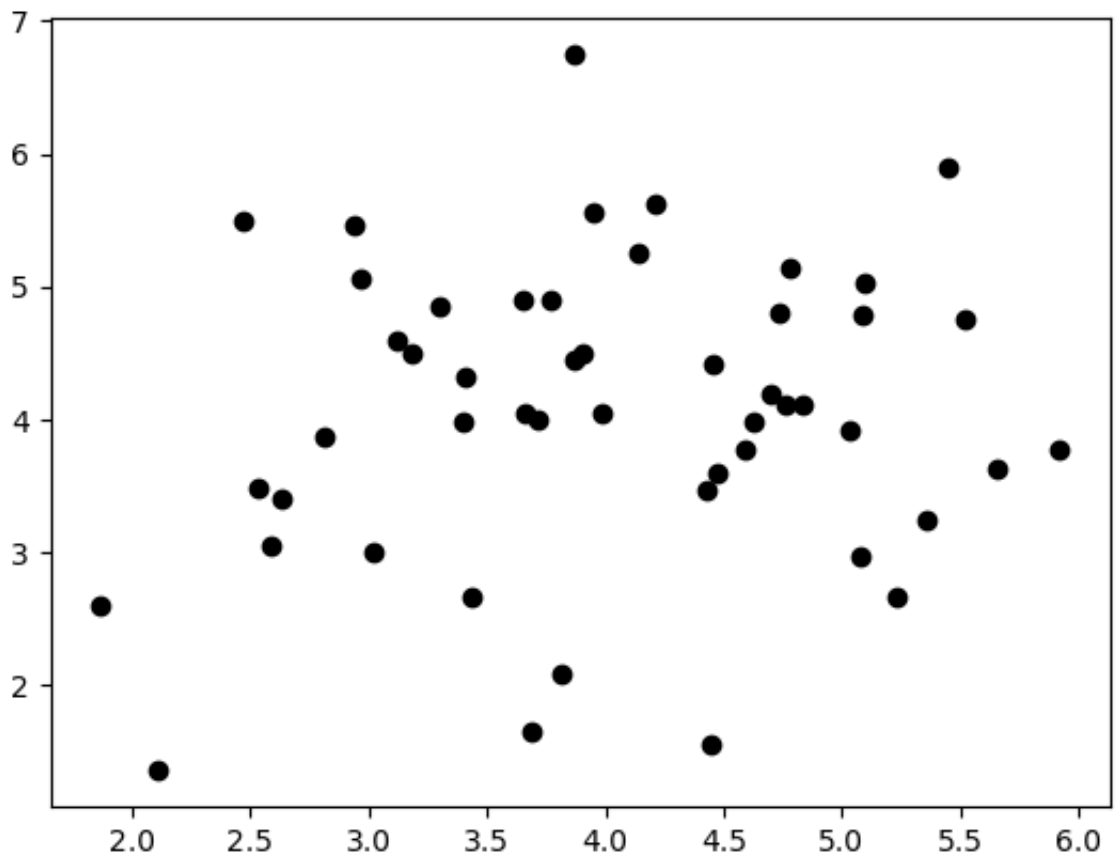
Homework 5 Question 1: Enclosing circle

Given a set of points in the plane $x_i \in R^2$, we would like to find the circle with smallest possible area that contains all of the points. Explain how to model this as an optimization problem. To test your model, generate a set of 50 random points using the code $X = 4 + randn(2, 50)$ (this generates a 2×50 matrix X whose columns are the x_i). Produce a plot of the randomly generated points along with the enclosing circle of smallest area.

Problem Data

In [12]:
```
using PyPlot
X = 4 + randn(2,50) # generate 50 random points
scatter( X[1,:], X[2,:], color="black") # plot the 50 points
;
```



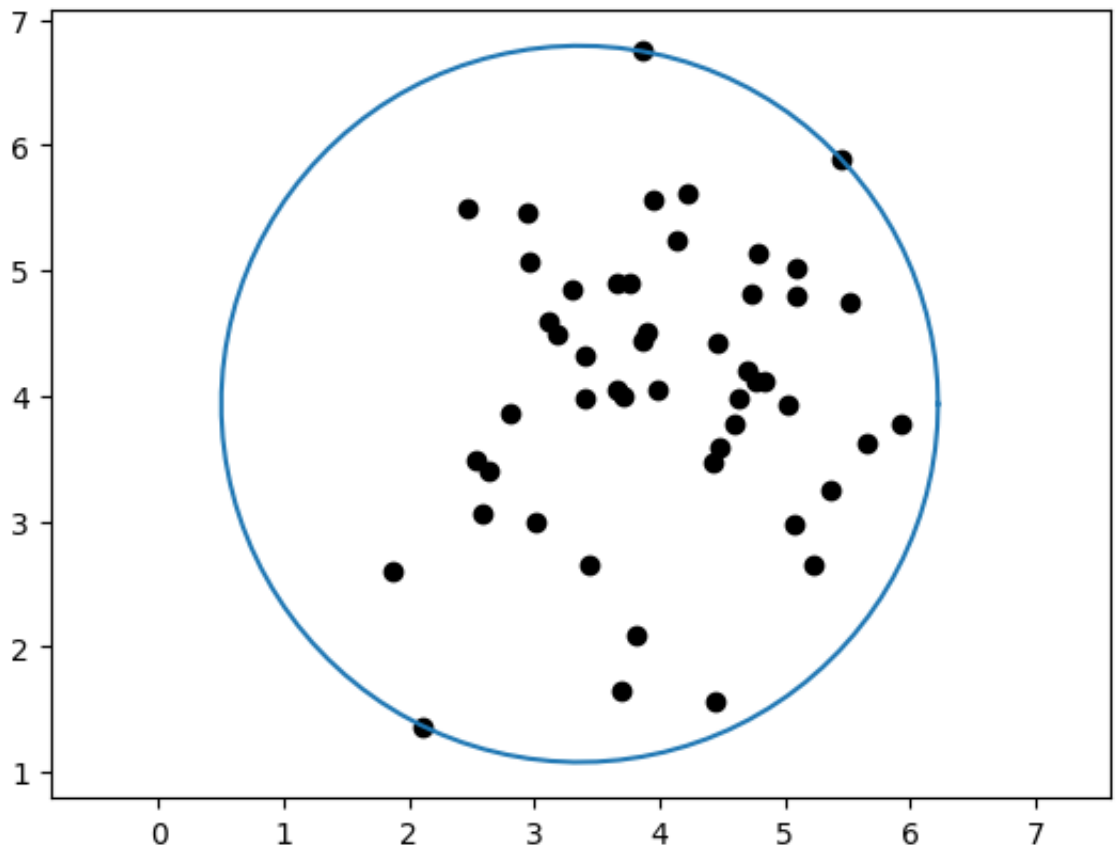
Problem Model

In [14]:
```
using JuMP, Gurobi

m = Model(solver=GurobiSolver(OutputFlag=0))

# The number of dimensions each point has
dimensions = length(X[:,1])

# Number of points
points = length(X[1,:])

# Variable to define circle, Centre and Radius
@variable(m, centre[1:dimensions])
@variable(m, rSquare >= 0)

# Constraint to enclose all the points within the circle
@constraint(m, enclosing[i in 1:points], sum((X[j,i] - centre[j])^2 for j in 1:dimensions) <= rSquare

@objective(m, Min, rSquare);
```

In [15]:
```
status = solve(m)
centre = getvalue(centre)
radius = sqrt(getobjectivevalue(m))
println("Status: ", status)
println("Min radius: ",radius)
println("Circle centered at: ",centre)
```

```
Status: Optimal
Min radius: 2.85871319235452
Circle centered at: [3.36254,3.93511]
```

In [16]:
```
t = linspace(0,2pi,100) # parameter that traverses the circle
plot( centre[1] + radius*cos(t), centre[2] + radius*sin(t)) # plot circle radius r with center (x1,x2
scatter( X[1,:], X[2,:], color="black") # plot the 50 points
axis("equal") # make x and y scales equal
;
```



Homework 5 Question 2: Quadratic form positivity.

You're presented with the constraint (1):

$$2x^2 + 2y^2 + 9z^2 + 8xy - 6xz - 6yz \leq 1$$

a) It turns out the above constraint is not convex. In other words, the set of (x, y, z) satisfying the constraint (1) is not an ellipsoid. Explain why this is the case.

In [1]:
```
Q = [2  4 -3;
     4  2 -3;
    -3 -3  9;]

λ,V = eig(Q)
println("Eigen values: ",λ)
println("Eigen vectors")
[println(V[r,:]) for r in 1:3];
```

```
Eigen values: [-2.0,3.0,12.0]
Eigen vectors
[0.707107,-0.57735,-0.408248]
[-0.707107,-0.57735,-0.408248]
[0.0,-0.57735,0.816497]
```

The above eigen value decomposition shows that the matrix Q is indefinite. For the constraint to be an ellipsoid it should be positive definite.

b) Show that the following QCQP is unbounded:

$$\begin{aligned} \text{maximize} \quad & x^2 + y^2 + z^2 \\ \text{subject to:} \quad & (1) \end{aligned}$$

The idea behind the code is explained as folows. The given constraint can be written in transformed dimensions as

$$-2p^2 + 3q^2 + 12r^2 - 1 <= 0$$

In order to make the equation negative, the p term should overpower the other terms in the equation. The equations for p, q and r in terms of x,y and z can be calculated by

$$V' * [x; y; z] = [p; q; r]$$

So we chose values for x,y,z in for the above equations to make the p-term larger than q and r. This results in solving the set of linear equation

$$A * b = c$$

in which Matrix A is the transpose of eigen vector matrix and c is vector representing points in transformed coordinate system(p,q,r). To make the p-dimension larger we simply choose [1;0;0] as c. Now b is the point in the older coordinate system. Multiplying c with a positive value would give us more and more negative values of the constraint above. The value of the objective is also shown

In [6]:
```
# Matrix P for the quadratic objective function x^2 + y^2 + z^2
P = [1 0 0;
     0 1 0;
     0 0 1;]

for k = 1:10
    c = [k; 0; 0]
    point = V'\c
    println("K: ",k)
    println("Quadratic Constraint Value: ", point'*Q*point - 1)
    println("Quadratic Objective Value: ",point'P*point)
end
```

```
K: 1
Quadratic Constraint Value: [-3.0]
Quadratic Objective Value: [1.0]
K: 2
Quadratic Constraint Value: [-9.0]
Quadratic Objective Value: [4.0]
K: 3
Quadratic Constraint Value: [-19.0]
Quadratic Objective Value: [9.0]
K: 4
Quadratic Constraint Value: [-33.0]
Quadratic Objective Value: [16.0]
K: 5
Quadratic Constraint Value: [-51.0]
Quadratic Objective Value: [25.0]
K: 6
Quadratic Constraint Value: [-73.0]
Quadratic Objective Value: [36.0]
K: 7
Quadratic Constraint Value: [-99.0]
Quadratic Objective Value: [49.0]
K: 8
Quadratic Constraint Value: [-129.0]
Quadratic Objective Value: [64.0]
K: 9
Quadratic Constraint Value: [-163.0]
Quadratic Objective Value: [81.0]
K: 10
Quadratic Constraint Value: [-201.0]
Quadratic Objective Value: [100.0]
```

This shows that the problem is unbounded as we can get points which satisfy the constraint and make the objective larger and larger.

Homework 5 Question 3: Hovercraft rendezvous.

Alice and Bob are cruising on Lake Mendota in their hovercrafts. Each hovercraft has the following dynamics: Dynamics of each hovercraft:

$$x_{t+1} = x_t + v_t/3600$$
$$v_{t+1} = v_t + u_t$$

At time t (in seconds), $x_t \in R^2$ is the position (in miles), $v_t \in R^2$ is the velocity (in miles per hour), and $u_t \in R^2$ is the thrust in normalized units. At t = 1, Alice has a speed of 20 mph going North, and Bob is located half a mile East of Alice, moving due East at 30 mph. Alice and Bob would like to rendezvous at t = 60 seconds. The location at which they meet is not important, but the time is!

a) Find the sequence of thruster inputs for Alice (u^A) and Bob (u^B) that achieves a rendezvous at t = 60 while minimizing the total energy used by both hovercraft:

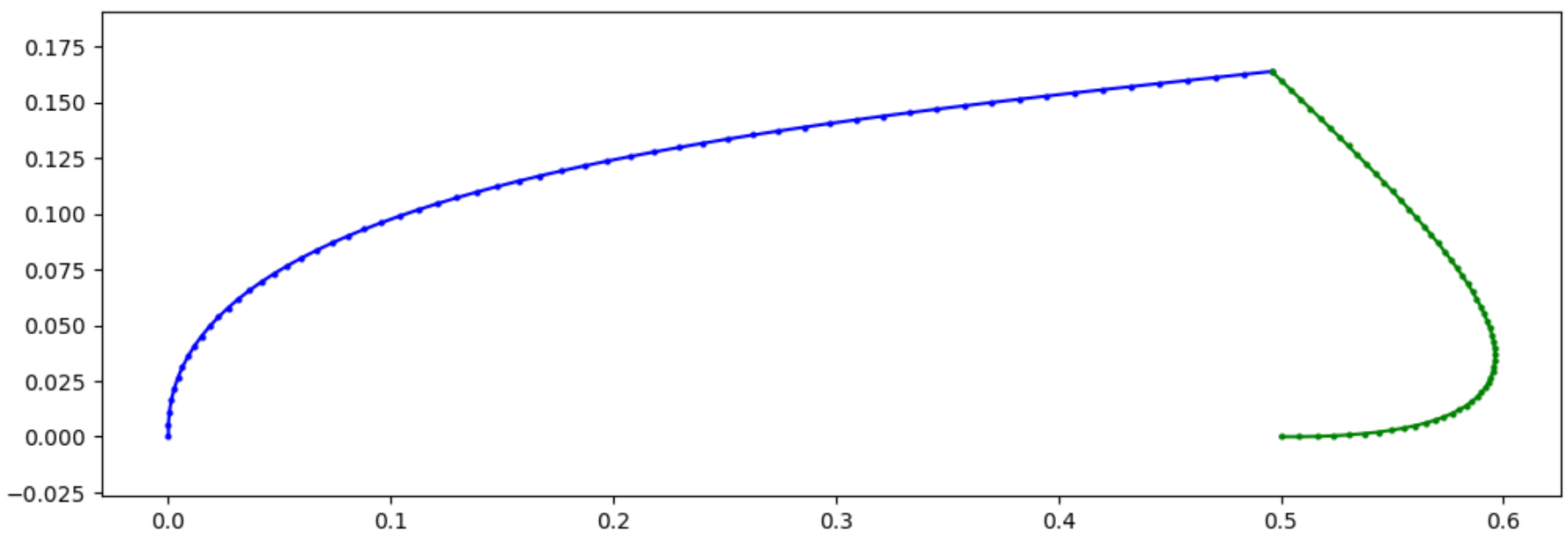
$$total\ energy = \sum_{t=1}^{60} ||u_t^A||^2 + \sum_{t=1}^{60} ||u_t^B||^2$$

Plot the trajectories of each hovercraft to verify that they do indeed rendezvous.

In [9]:
```
using JuMP, Mosek

person = [:Alice, :Bob]
k = 2              # number of waypoints
T = zeros(Int,k)   # vector of timepoints

T[1] = 1
T[2] = 60

m = Model(solver = MosekSolver(LOG=0))

@variable(m, x[person, 1:2, 1:T[k]])  # resulting position
@variable(m, v[person, 1:2, 1:T[k]])  # resulting velocity
@variable(m, u[person, 1:2, 1:T[k]])  # thruster input

# satisfy the dynamics (with provided initial velocity)
# (x,y) := (East, North)
@constraint(m, v[:Alice,:,1] .== [0;20])
@constraint(m, v[:Bob,:,1] .== [30;0])

# constraint for initial distance between Alice and Bob
@constraint(m, x[:Alice,:,1] .== [0;0])
@constraint(m, x[:Bob,:,1] .== [0.5;0])

# constraints for dynamics of the model
for p in person
    for t in 1:T[k]-1
        @constraint(m, x[p,:,t+1] .== x[p,:,t] + v[p,:,t]/3600.)
        @constraint(m, v[p,:,t+1] .== v[p,:,t] + u[p,:,t])
    end
end

# hit all the waypoints
@constraint(m, x[:Alice,:,T[2]] .== x[:Bob,:,T[2]])

# Minimize the thrusts
@objective(m, Min, sum(sum(u[p,:,:].^2 for p in person)))
solve(m)
```

Out[9]: :Optimal

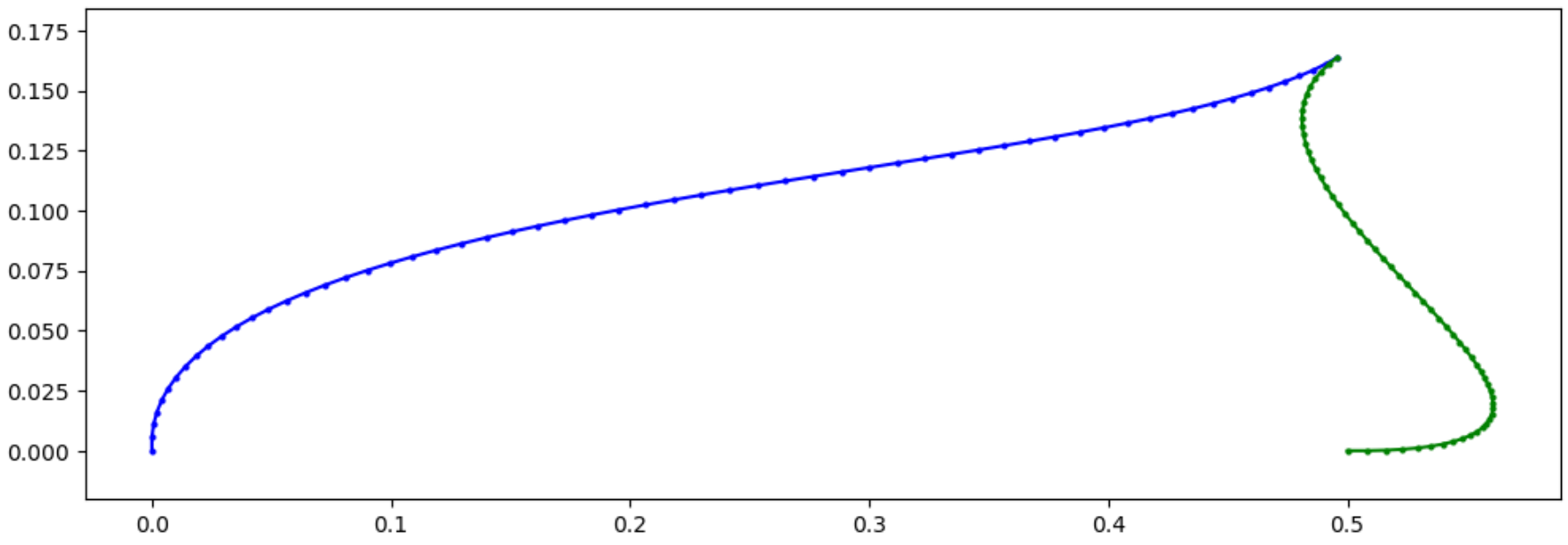
In [10]:
```
X = getvalue(x)
println("Optimal rendezvous location: ",X[:Alice,:,60])
using PyPlot
figure(figsize=(12,4))
plot( X[:Alice,1,:][:], X[:Alice,2,:][:], "b.-", markersize=4 )
plot( X[:Bob,1,:][:], X[:Bob,2,:][:], "g.-", markersize=4 )
axis("equal");
```




```
Optimal rendezvous location: [0.495833,0.163889]
```

b) In addition to arriving at the same place at the same time, Alice and Bob should also make sure their velocity vectors match when they rendezvous (otherwise, they might crash!) Solve the rendezvous problem again with the additional velocity matching constraint and plot the resulting trajectories. Is the optimal rendezvous location different from the one found in the first part?

In [11]:
```
@constraint(m, v[:Alice,:,T[2]] .== v[:Bob,:,T[2]])
solve(m)
X = getvalue(x)
println("Optimal rendezvous location: ",X[:Alice,:,60])
using PyPlot
figure(figsize=(12,4))
plot( X[:Alice,1,:][:], X[:Alice,2,:][:], "b.-", markersize=4 )
plot( X[:Bob,1,:][:], X[:Bob,2,:][:], "g.-", markersize=4 )
axis("equal");
```




```
Optimal rendezvous location: [0.495833,0.163889]
```

c) Alice and Bob forgot about one important detail. The hovercrafts each have a top speed of 35 mph. The solutions found in the previous parts are unacceptable because they require Alice to exceed the maximum allowable speed. First, verify that this is indeed the case. Second, explain how to alter your model to account for the speed limit. Finally, solve the rendezvous problem one last time with all the constraints in place and verify that your solution respects the speed limit.

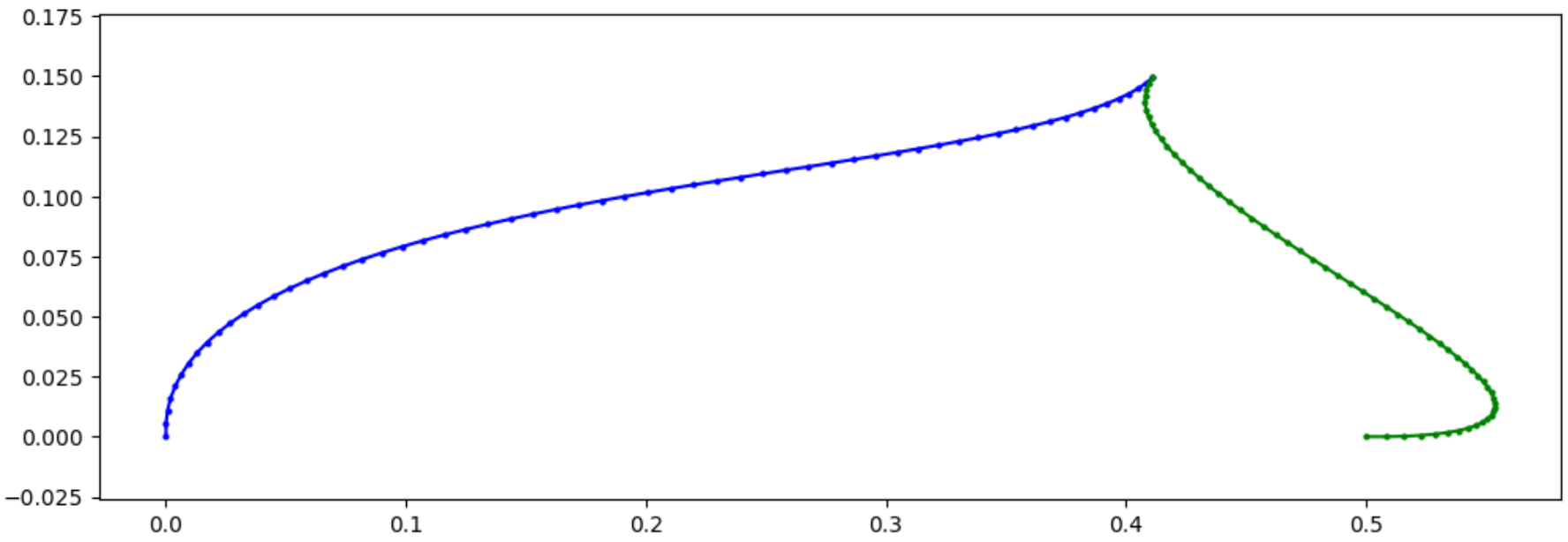
In [12]:
```
V = getvalue(v)
[println("Max velocity for ",p ," : ",maximum(norm(V[p,:,t]) for t in 1:T[k])," mph") for p in person
```

```
Max velocity for Alice : 42.79755747529796 mph
Max velocity for Bob : 30.0 mph
```

In [13]:
```
@constraint(m, maxspeed[t in 1:T[k], p in person], sum(v[p,:,t].^2) <= 35^2)
solve(m)
X = getvalue(x)
println("Optimal rendezvous location: ",X[:Alice,:,60])
```

```
Optimal rendezvous location: [0.41129,0.149452]
```

In [14]:
```
using PyPlot
figure(figsize=(12,4))
plot( X[:Alice,1,:][:], X[:Alice,2,:][:], "b.-", markersize=4 )
plot( X[:Bob,1,:][:], X[:Bob,2,:][:], "g.-", markersize=4 )
axis("equal");
```



In [15]:
```
V = getvalue(v)
[println("Max velocity for ",p ," : ",maximum(norm(V[p,:,t]) for t in 1:T[k])," mph") for p in person
```

```
Max velocity for Alice : 34.99999964530002 mph
Max velocity for Bob : 30.0 mph
```