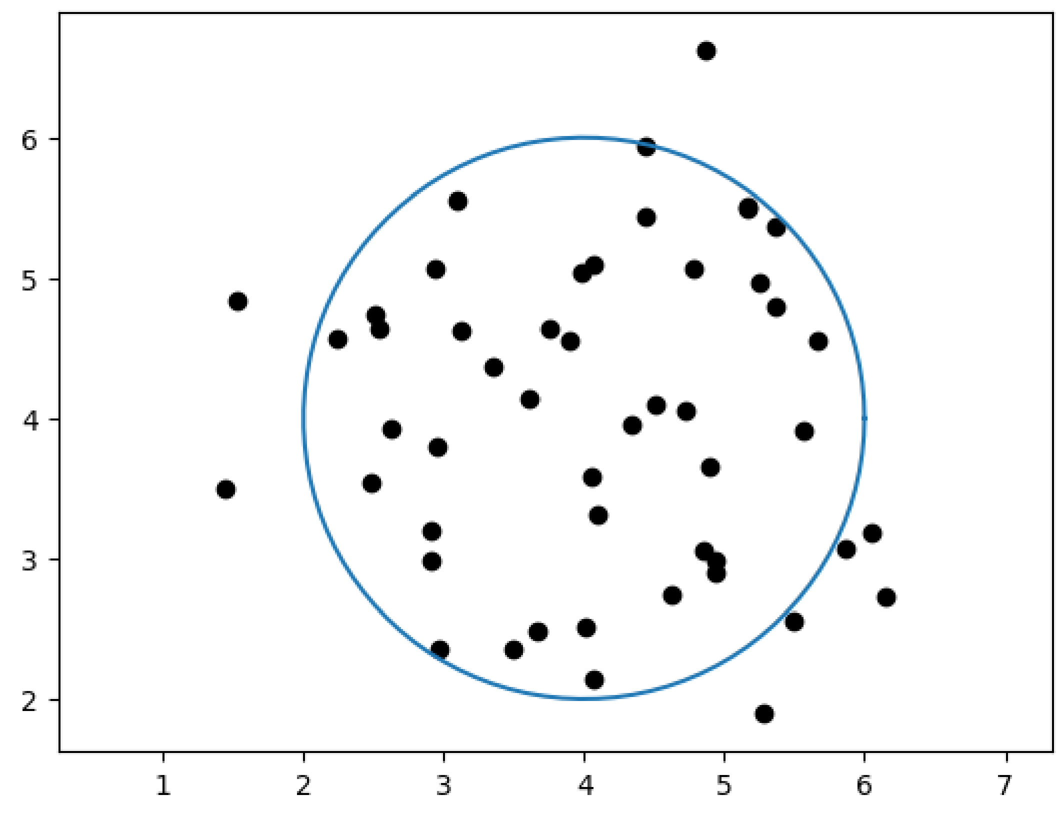
Enclosing Circle

Here we find the circle with the smallest area (radius) that which encompasses all the generated points. We do this by finding the point, say p, in the plane which minimizes the sum of the distances (squared) from p to all the randomly generated points. Then we find the randomly generated point which is farthest from p and use that distance as our radius.

Below we exercise the code given in the assignment to see what the points look like, along with a reference circle of radius 2.

In [249]:
```
using PyPlot
X = 4 + randn(2,50) # generate 50 random points
t = linspace(0,2pi,100) # parameter that traverses the circle
r = 2; x1 = 4; x2 = 4 # radius and coordinates of the center
plot( x1 + r*cos(t), x2 + r*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
```



Out[249]:
```
(1.1704366014855305,6.408442058316499,1.6248441496476056,6.900309249525197)
```

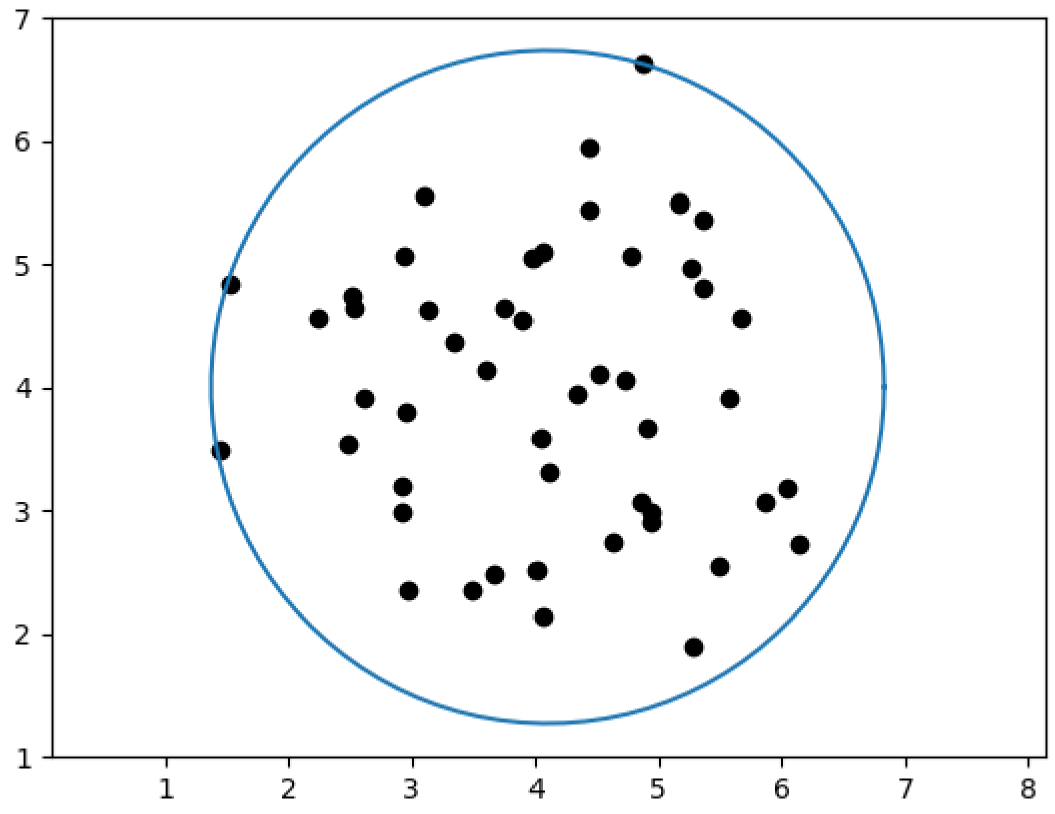
In [250]:
```
using JuMP, Gurobi

m = Model(solver = GurobiSolver())

@variable(m, center_x)
@variable(m, center_y)
@objective(m, Min, sum((center_x - X[1,i])^2 + (center_y - X[2,i])^2 for i = 1:50))
solve(m)
```

```
Optimize a model with 0 rows, 2 columns and 0 nonzeros
Model has 2 quadratic objective terms
Coefficient statistics:
  Matrix range     [0e+00, 0e+00]
  Objective range  [4e+02, 4e+02]
  QObjective range [1e+02, 1e+02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [0e+00, 0e+00]
Presolve removed 0 rows and 2 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Barrier solved model in 0 iterations and 0.00 seconds
Optimal objective -1.64201938e+03
```

Out[250]:
```
:Optimal
```

In [251]:
```
x = getvalue(center_x)
y = getvalue(center_y)

max = 0
d = 0
for i = 1:50
    d = sum(((x - X[1,i])^2 + (y - X[2,i])^2)^(1/2))
    if d > max
        max = d
    end
end
```

In [252]:
```
t = linspace(0,2pi,100) # parameter that traverses the circle
r = max
plot( x + r*cos(t), y + r*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
```



Out[252]:
```
(1.0964852116539272,7.106235993505178,0.9978539108963548,7.00836124928748)
```

Quadratic Form Positivity

Here we examine a quadratic constraint and explain why it is not convex. Then we look at why maximizing a cost function expressible as a positive-definite quadratic subject to the prior constraint is infeasible.

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

is equivalent to

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}^T \begin{bmatrix} 2 & 4 & -3 \\ 4 & 2 & -3 \\ -3 & -3 & 9 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

After finding the eigenpairs of the symmetric matrix above, we find that

$$\begin{bmatrix} 2 & 4 & -3 \\ 4 & 2 & -3 \\ -3 & -3 & 9 \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 \\ -1 & 1 & 1 \\ 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 12 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} -1 & 1 & -1 \\ -1 & 1 & 1 \\ 2 & 1 & 0 \end{bmatrix}^T$$

We use the following change of variables

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 \\ -1 & 1 & 1 \\ 2 & 1 & 0 \end{bmatrix}^T \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Thus

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}^T \begin{bmatrix} 2 & 4 & -3 \\ 4 & 2 & -3 \\ -3 & -3 & 9 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}^T \begin{bmatrix} 12 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Therefore our original quadratic constraint can be represented by

$$12p^2 + 3q^2 - 2r^2 \leq 1$$

and then the coefficient matrix is not positive-definite, implying the constraint is not convex.

By solving the above change of variables in terms of x, y, and z, and also noting that we can make r arbitrarily large while still satisfying the constraint, we see that we can let p = q = 0 and let r = k be some arbitrarily large number. Then

$$x^2 + y^2 + z^2 = (-k/2)^2 + (k/2)^2$$

Now we can clearly see that the above cost function maximizes to infinity while still satisfying its constraint.

Hovercraft Rendezvous

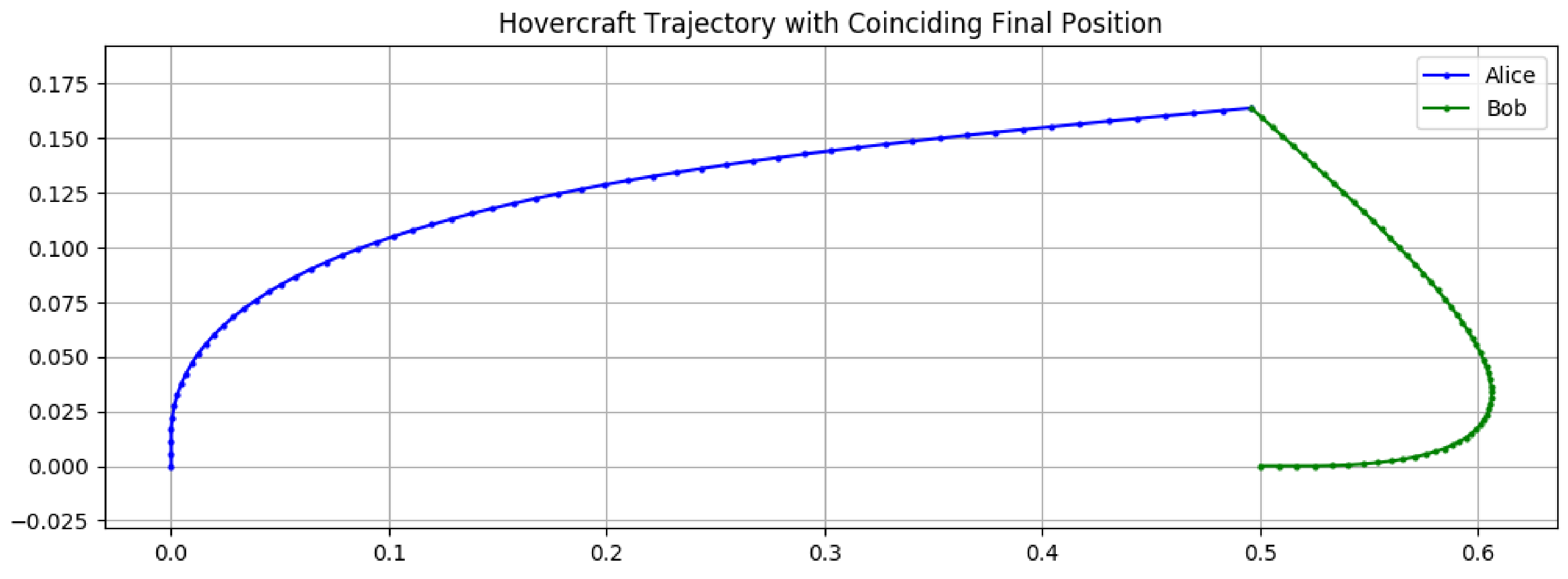
Model the different hovercraft situations that we always seem to find ourselves in.

In [253]:
```
m = Model(solver=GurobiSolver())
@variable(m, alice[1:2,1:60])   #position
@variable(m, bob[1:2,1:60])
@variable(m, alice_v[1:2,1:60]) #velocity
@variable(m, bob_v[1:2,1:60])
@variable(m, alice_u[1:2,1:60]) #thrust
@variable(m, bob_u[1:2,1:60])
for i = 1:2
    for j = 1:2
        @constraint(m, alice_u[i,j] == 0) #initial thrusts of zero
        @constraint(m, bob_u[i,j] == 0)
    end
end
@constraint(m, alice_v[1,1] == 0)   #initial velocities
@constraint(m, alice_v[2,1] == 20)
@constraint(m, bob_v[1,1] == 30)
@constraint(m, bob_v[2,1] == 0)

@constraint(m, alice[1,1] == 0)
@constraint(m, alice[2,1] == 0)
@constraint(m, bob[1,1] == 1/2)
@constraint(m, bob[2,1] == 0)


for i = 2:60   #hovercraft dynamics
    @constraint(m, alice[:,i] .== alice[:,i-1] + (1/3600)*alice_v[:,i-1])
    @constraint(m, bob[:,i] .== bob[:,i-1] + (1/3600)*bob_v[:,i-1])

    @constraint(m, alice_v[:,i] .== alice_v[:,i-1] + alice_u[:,i-1])
    @constraint(m, bob_v[:,i] .== bob_v[:,i-1] + bob_u[:,i-1])
end
@constraint(m, bob[:,60] .== alice[:,60])  #coinciding final positions
@objective(m, Min, sum(dot(alice_u[:,i],alice_u[:,i]) + dot(bob_u[:,i],bob_u[:,i]) for i = 1:60))
solve(m)
```

```
Optimize a model with 490 rows, 720 columns and 1436 nonzeros
Model has 240 quadratic objective terms
Coefficient statistics:
  Matrix range     [3e-04, 1e+00]
  Objective range  [0e+00, 0e+00]
  QObjective range [2e+00, 2e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [5e-01, 3e+01]
Presolve removed 488 rows and 496 columns
Presolve time: 0.00s
Presolved: 2 rows, 224 columns, 224 nonzeros
Presolved model has 224 quadratic objective terms
Ordering time: 0.00s

Barrier statistics:
 Free vars  : 224
 AA' NZ     : 0.000e+00
 Factor NZ  : 2.000e+00
 Factor Ops : 2.000e+00 (less than 1 second per iteration)
 Threads    : 1

                  Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   1.17583505e+02 -1.17583505e+02  7.11e-14 1.66e+00  0.00e+00     0s
   1   1.17583505e+02  1.17583270e+02  8.53e-14 1.66e-06  0.00e+00     0s
   2   1.17583505e+02  1.17583505e+02  2.84e-14 1.66e-12  0.00e+00     0s

Barrier solved model in 2 iterations and 0.00 seconds
Optimal objective 1.17583505e+02
```

Out[253]:
```
:Optimal
```

In [254]:
```
a1 = getvalue(alice)
b1 = getvalue(bob)

alice_f = zeros(2,2)
bob_f = zeros(2,2)

alice_f[:,1] = a1[:,60]
bob_f[:,1]   = b1[:,60]

figure(figsize=(12,4))
plot(a1[1,:][:], a1[2,:][:], "b.-", markersize=4)
plot(b1[1,:][:], b1[2,:][:], "g.-", markersize=4)
legend(["Alice","Bob"])
title("Hovercraft Trajectory with Coinciding Final Position")
axis("equal")
grid()
```

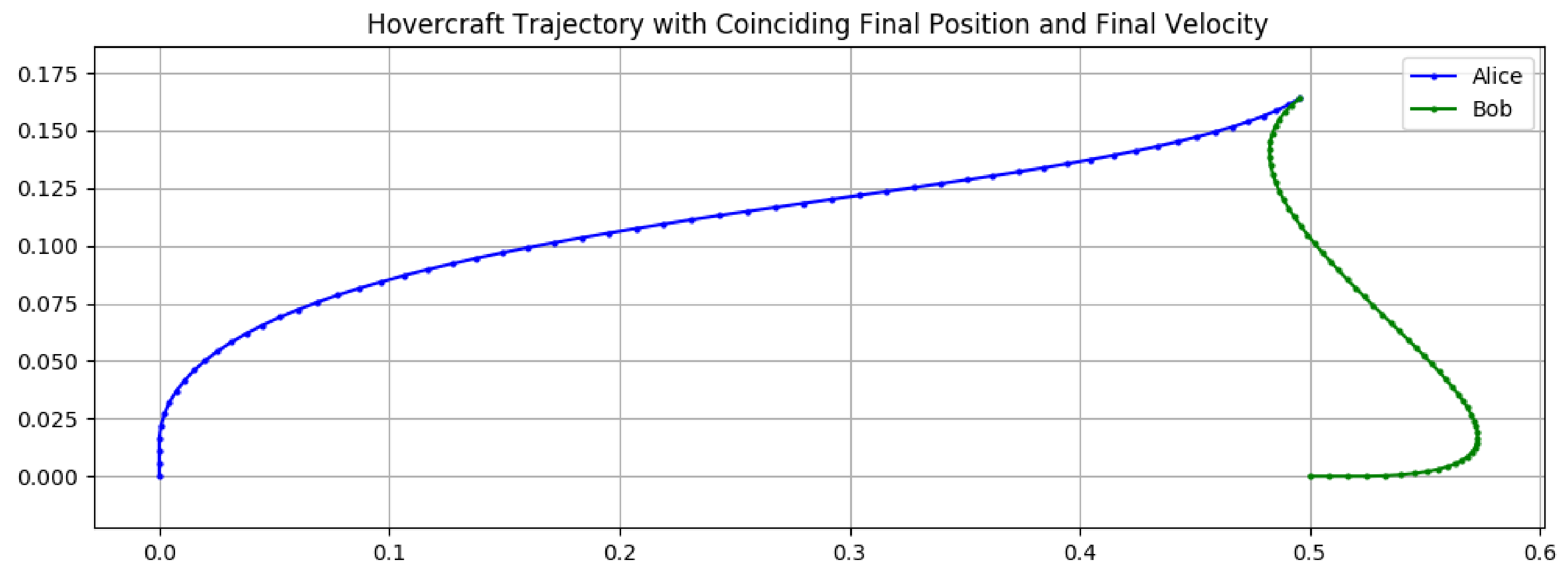


In [255]:
```
@constraint(m, alice_v[:,60] .== bob_v[:,60])
solve(m)
```

```
Optimize a model with 492 rows, 720 columns and 1440 nonzeros
Model has 240 quadratic objective terms
Coefficient statistics:
  Matrix range     [3e-04, 1e+00]
  Objective range  [0e+00, 0e+00]
  QObjective range [2e+00, 2e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [5e-01, 3e+01]
Presolve removed 488 rows and 492 columns
Presolve time: 0.00s
Presolved: 4 rows, 228 columns, 452 nonzeros
Presolved model has 228 quadratic objective terms
Ordering time: 0.00s

Barrier statistics:
 Free vars  : 228
 AA' NZ     : 2.000e+00
 Factor NZ  : 6.000e+00
 Factor Ops : 1.000e+01 (less than 1 second per iteration)
 Threads    : 1

                  Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   2.65399490e+02 -2.65399490e+02  1.09e-12 2.74e+00  0.00e+00     0s
   1   2.65399490e+02  2.65398959e+02  6.39e-14 2.74e-06  0.00e+00     0s
   2   2.65399490e+02  2.65399490e+02  2.84e-14 2.74e-12  0.00e+00     0s

Barrier solved model in 2 iterations and 0.00 seconds
Optimal objective 2.65399490e+02
```

Out[255]:
```
:Optimal
```

In [256]:
```
a2 = getvalue(alice)
b2 = getvalue(bob)

alice_f[:,2] = a2[:,60]
bob_f[:,2]   = b2[:,60]

figure(figsize=(12,4))
plot(a2[1,:][:], a2[2,:][:], "b.-", markersize=4)
plot(b2[1,:][:], b2[2,:][:], "g.-", markersize=4)
legend(["Alice","Bob"])
title("Hovercraft Trajectory with Coinciding Final Position and Final Velocity")
axis("equal")
grid()
```



Here we test to see if the final positions for the subproblems, so far, have similar final positions.

In [257]:
```
if alice_f == bob_f
    println("The final positions for parts a and b are the same.")
else
    println("The final positions for parts a and b are the different.")
end
```
```
The final positions for parts a and b are the same.
```

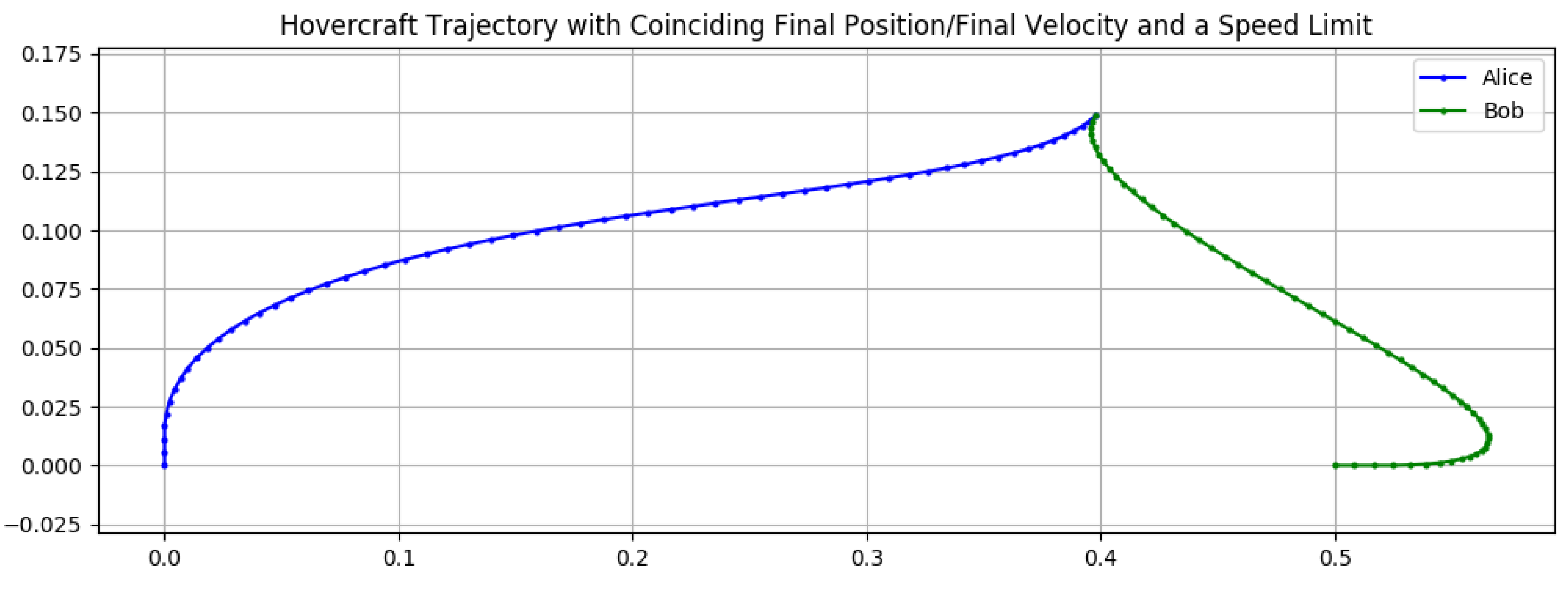
In [258]:
```
max_speed = 0
v = getvalue(alice_v)

for i = 1:60
    s = norm(v[:,i])
    if s > max_speed
        max_speed = s
    end
end

println("The maximum speed attained by Alice: ", max_speed, " mph")
```
```
The maximum speed attained by Alice: 44.287754250450654 mph
```

In [259]:
```
for i = 1:60
   @constraint(m, dot(alice_v[:,i], alice_v[:,i]) <= 1225)
   @constraint(m, dot(bob_v[:,i], bob_v[:,i]) <= 1225)
end
solve(m)
```

```
Optimize a model with 492 rows, 720 columns and 1440 nonzeros
Model has 240 quadratic objective terms
Model has 120 quadratic constraints
Coefficient statistics:
  Matrix range     [3e-04, 1e+00]
  QMatrix range    [1e+00, 1e+00]
  Objective range  [0e+00, 0e+00]
  QObjective range [2e+00, 2e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [5e-01, 3e+01]
Presolve removed 46 rows and 50 columns
Presolve time: 0.00s
Presolved: 565 rows, 790 columns, 1460 nonzeros
Presolved model has 115 second-order cone constraints
Ordering time: 0.00s

Barrier statistics:
 Dense cols : 1
 Free vars  : 218
 AA' NZ     : 2.684e+03
 Factor NZ  : 7.280e+03 (roughly 1 MByte of memory)
 Factor Ops : 1.049e+05 (less than 1 second per iteration)
 Threads    : 1

                  Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   1.39986312e+04  1.39986312e+04  8.45e+02 4.23e+03  1.54e+04     0s
   1  -4.14162379e+03 -1.92534112e+05  5.06e+01 4.61e+02  1.18e+03     0s
   2  -4.13945323e+02 -7.52808007e+04  7.78e+00 5.07e-04  1.80e+02     0s
   3   8.59751907e+02 -5.47059164e+04  5.64e-01 2.97e-04  8.49e+01     0s
   4   7.40481798e+02 -3.26387928e+04  2.85e-01 1.71e-04  4.99e+01     0s
   5   9.11602025e+02 -1.06472439e+04  2.50e-02 5.38e-05  1.69e+01     0s
   6   6.32390406e+02 -6.31613619e+03  1.38e-02 3.25e-05  1.02e+01     0s
   7   4.56986926e+02 -1.75929960e+03  9.69e-04 1.05e-05  3.23e+00     0s
   8   3.82929551e+02 -1.01055184e+03  3.91e-04 6.70e-06  2.03e+00     0s
   9   3.28000201e+02 -4.14719245e+02  1.85e-04 3.57e-06  1.08e+00     0s
  10   2.98724386e+02 -1.28595578e+02  5.81e-05 2.09e-06  6.23e-01     0s
  11   2.88673873e+02  8.75203753e+01  1.20e-05 9.91e-07  2.93e-01     0s
  12   2.79998716e+02  1.73293081e+02  4.03e-06 5.35e-07  1.56e-01     0s
  13   2.73520671e+02  2.56493793e+02  1.05e-06 6.98e-08  2.48e-02     0s
  14   2.71082491e+02  2.67615819e+02  1.91e-07 3.44e-08  5.05e-03     0s
  15   2.70726863e+02  2.70512292e+02  6.76e-08 6.78e-09  3.13e-04     0s
  16   2.70641671e+02  2.70620219e+02  1.09e-08 8.31e-09  3.13e-05     0s
  17   2.70635892e+02  2.70633120e+02  7.20e-10 2.80e-09  4.04e-06     0s
  18   2.70635142e+02  2.70634945e+02  9.91e-11 1.14e-10  2.88e-07     0s

Barrier solved model in 18 iterations and 0.02 seconds
Optimal objective 2.70635142e+02
```

Out[259]: :Optimal

In [260]:
```
a3 = getvalue(alice)
b3 = getvalue(bob)

alice_f[:,3] = a3[:,60]
bob_f[:,3]   = b3[:,60]

figure(figsize=(12,4))
plot(a3[1,:][:], a3[2,:][:], "b.-", markersize=4)
plot(b3[1,:][:], b3[2,:][:], "g.-", markersize=4)
legend(["Alice","Bob"])
title("Hovercraft Trajectory with Coinciding Final Position/Final Velocity and a Speed Limit")
axis("equal")
grid()
```



In [261]:
```
max_speed = 0
v1 = getvalue(alice_v)

for i = 1:60
    s = norm(v1[:,i])
    if s > max_speed
        max_speed = s
    end
end

println("The maximum speed attained by Alice: ", max_speed, " mph")
```

```
The maximum speed attained by Alice: 34.999993946446835 mph
```