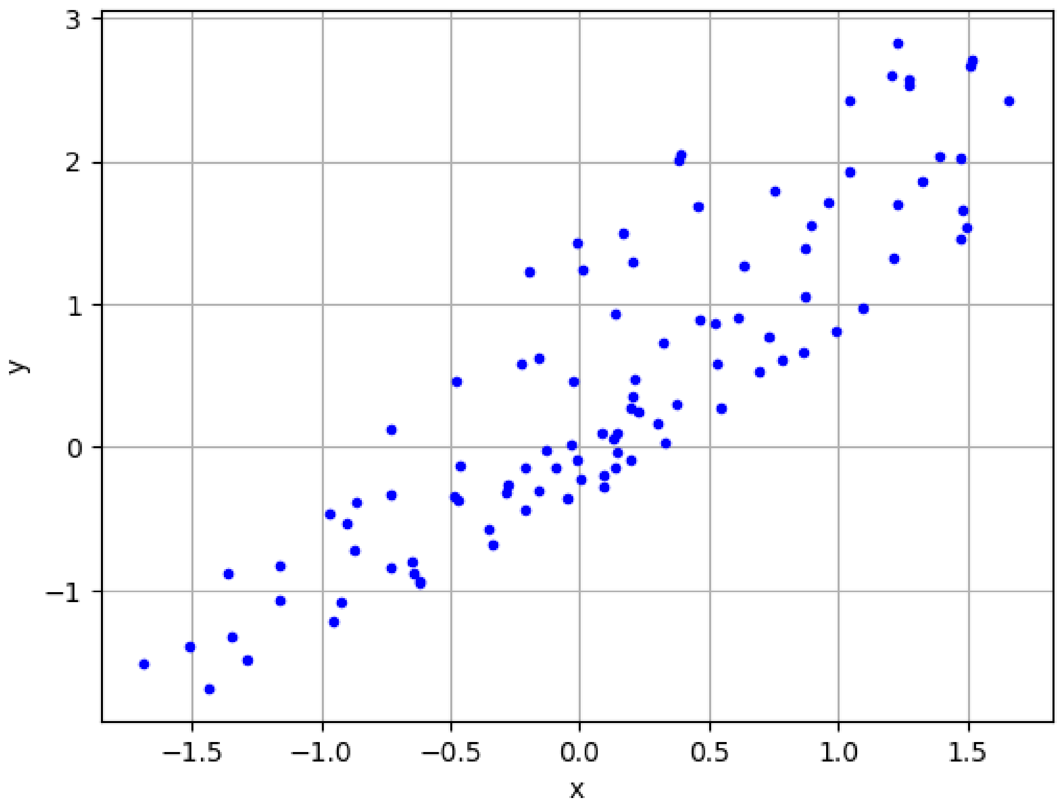
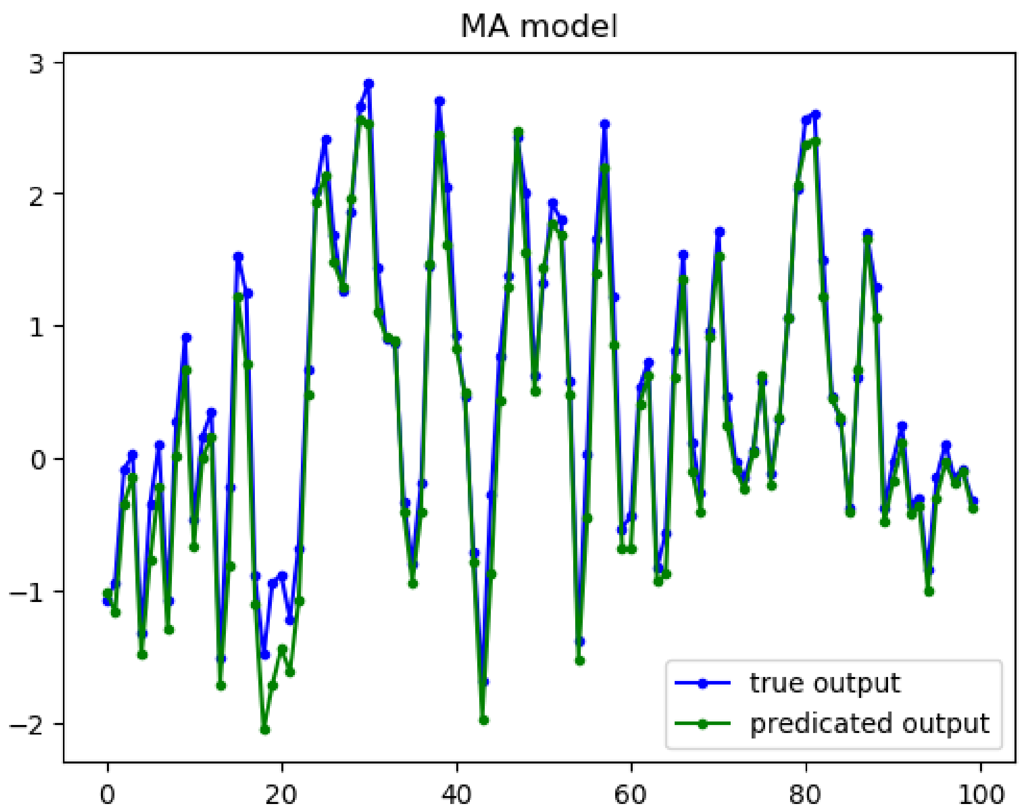
In [15]:

```
#Problem 1a
using PyPlot

input = readcsv("uy_data.csv")
x = input[:, 1]
y = input[:, 2]

plot(x,y,"b.")
xlabel("x")
ylabel("y")
grid("on")
```



In [16]:

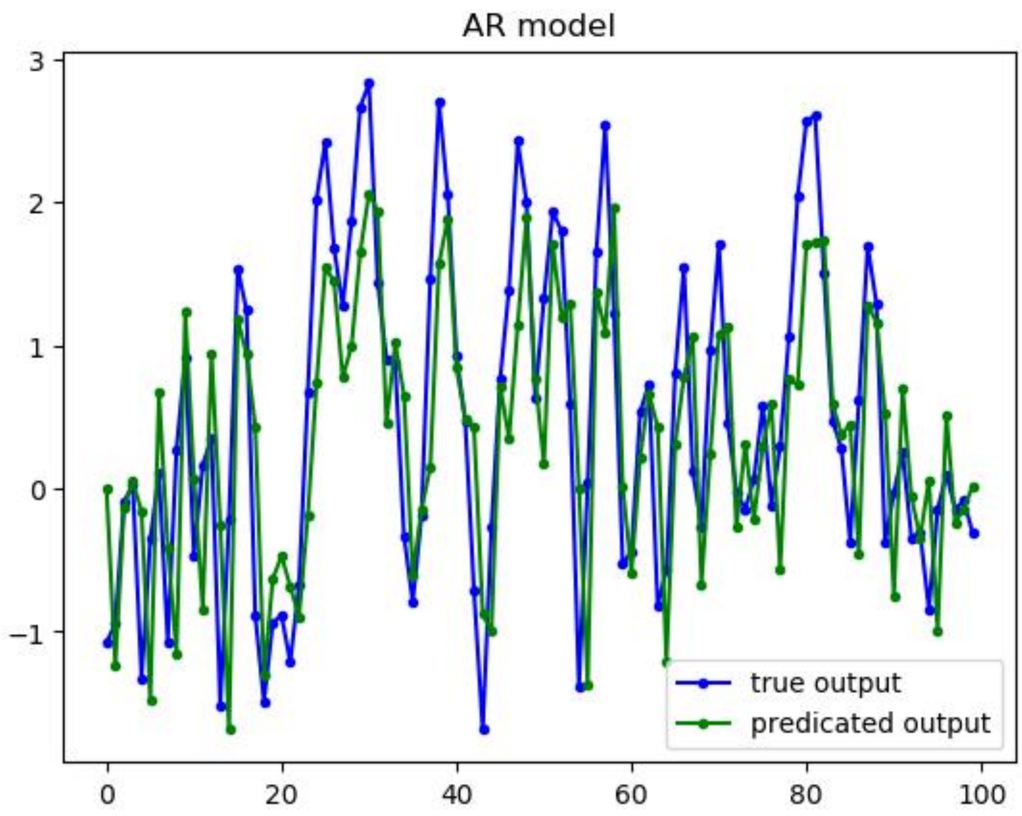
```
#using MA model
width = 5
A_MA = zeros(length(x), width)
for i = 1:width
    A_MA[i:end,i] = x[1:end-i+1]
end

wopt_MA = A_MA\y
yest_MA = A_MA*wopt_MA

plot(y, "b.-", yest_MA, "g.-")
legend(["true output", "predicated output"], loc = "lower r
ight")
title("MA model")
println()
println(norm(yest_MA-y))
```




```
2.460854388269911
```

In [18]:

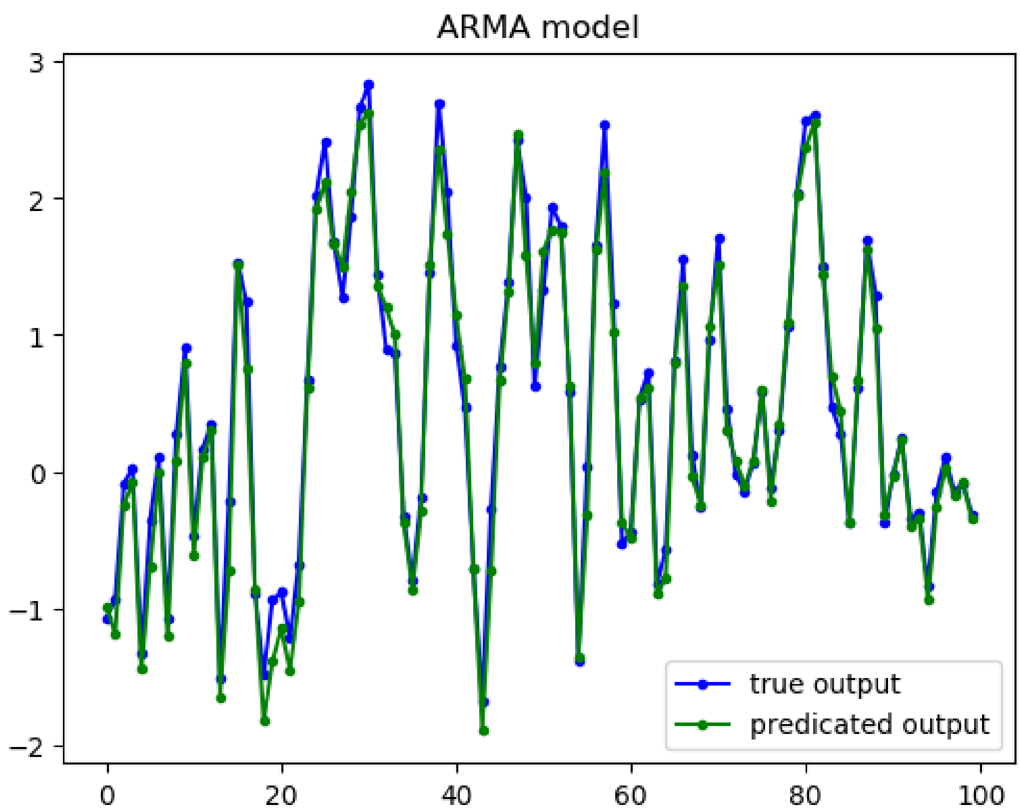
```
#using AR model
width = 5
A_AR = zeros(length(y), width)
for i = 1:width
    A_AR[i+1:end,i] = y[1:end-i]
end

wopt_AR = A_AR\y
yest_AR = A_AR*wopt_AR

plot(y,"b.-",yest_AR,"g.-")
legend(["true output", "predicated output"], loc="lower rig
ht")
title("AR model")
println()
println(norm(yest_AR-y))
```




```
7.436691765656793
```

In [20]:

```
#Problem 1b
#using ARMA model
k = 1
l = 1
A_ARMA = zeros(length(y), k+1)
for i = 1:k
    A_ARMA[i+1:end,i] = y[1:end-i]
End

for i = 1:l
    A_ARMA[i:end,k+i] = x[1:end-i+l]
end
wopt_ARMA = A_ARMA\y
yest_ARMA = A_ARMA*wopt_ARMA
plot(y,"b.-", yest_ARMA, "g.-")
legend(["true output","predicated output”)
title("ARMA model")
println()
println(norm(yest_ARMA-y))
```



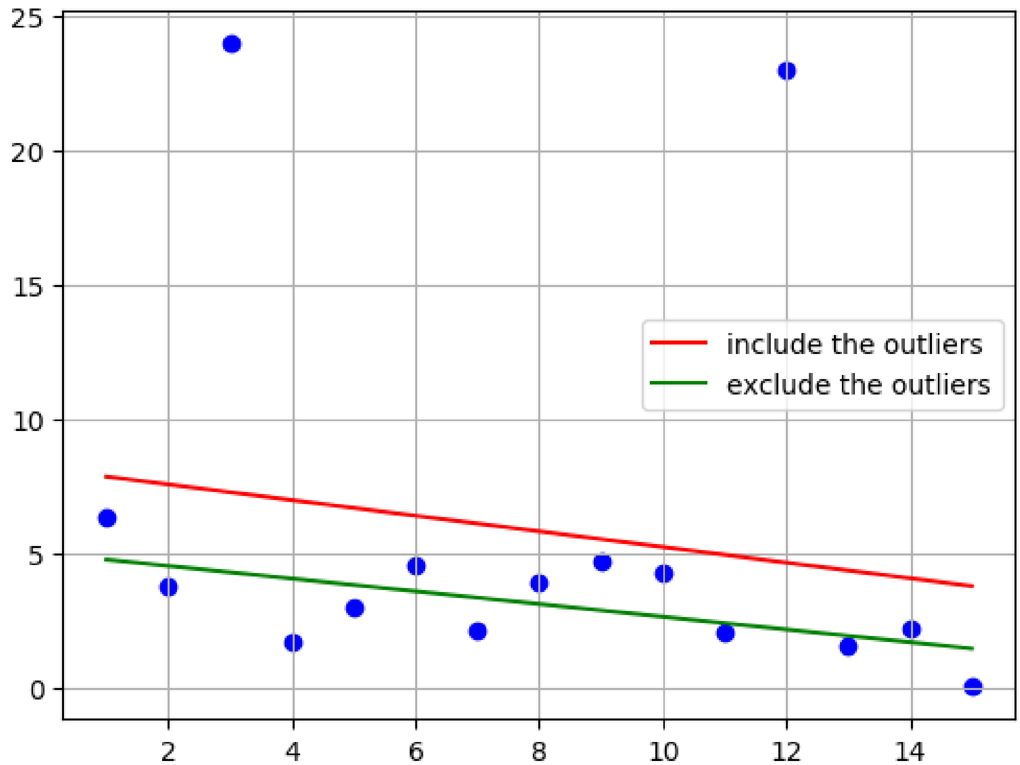

```
1.8565828148734604
```

In [1]:

```
#Problem 2a
x = [1:15;]
y = [6.31 3.78 24 1.71 2.99 4.53 2.11 3.88 4.67 4.25 2.06 2
3 1.58 2.17 0.02]
x_drop_outliers = [1 2 4 5 6 7 8 9 10 11 13 14 15]
y_drop_outliers = [6.31 3.78 1.71 2.99 4.53 2.11 3.88 4.67
4.25 2.06 1.58 2.17 0.02]

using JuMP, Gurobi, PyPlot

# include the outliers
m1 = Model(solver=GurobiSolver(OutputFlag=0))

@variable(m1, a1)
@variable(m1, b1)

@objective(m1, Min, sum{(y[i]-a1*x[i]-b1)^2, i=1:15})

solve(m1)

a1 = getvalue(a1)
b1 = getvalue(b1)

println("best fit of least square(include the outliers):")
println("a1:", a1)
println("b1:", b1)

# exclude the outliers
m2 = Model(solver=GurobiSolver(OutputFlag=0))

@variable(m2, a2)
@variable(m2, b2)

@objective(m2, Min, sum{(y_drop_outliers[i]-a2*x_drop_outli
ers[i]-b2)^2, i=1:13})

solve(m2)

a2 = getvalue(a2)
b2 = getvalue(b2)

println("best fit of least square excluding the outliers:")
```

```
println("a2:", a2)
println("b2:", b2)

scatter(x, y, color="blue")
plot(x, a1*x+b1, color="red", label="include the outliers")
plot(x, a2*x+b2, color="green", label="exclude the outliers
")
legend()
grid("on")
```

```
Academic license - for non-commercial use only
best fit of least square(include the outliers):
a1:
```

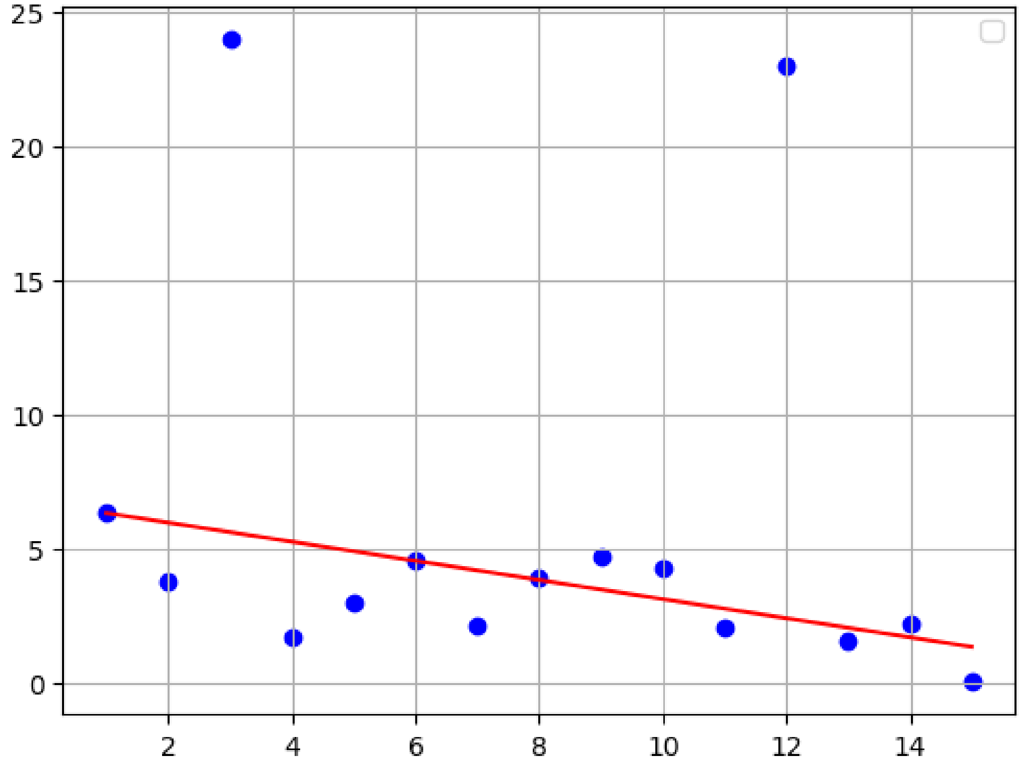



```
-0.29078571428551947
b1:8.130285714279665
Academic license - for non-commercial use only
best fit of least square excluding the outliers:
a2:-0.23648422408233874
b2:4.9916033483557305
```

Here we can see that the red that includes the outliers is significantly higher than the blue that does not.

In [2]:

```
#Problem 2b
x = [1:15;]
y = [6.31 3.78 24 1.71 2.99 4.53 2.11 3.88 4.67 4.25 2.06 2
3 1.58 2.17 0.02]

using JuMP, Gurobi, PyPlot

# include the outliers
m = Model(solver=GurobiSolver(OutputFlag=0))

@variable(m, a)
@variable(m, b)

# l1-norm
@variable(m, t[1:15])
for i in 1:15
    @constraint(m, y[i]-a*x[i]-b <= t[i])
    @constraint(m, -t[i] <= y[i]-a*x[i]-b)
end

@objective(m, Min, sum(t))

solve(m)

a = getvalue(a)
b = getvalue(b)

println("best fit of l1 cost function(include the outliers)
:")
println("a:", a)
println("b:", b)

scatter(x, y, color="blue")
plot(x, a*x+b, color="red")
legend()
grid("on")
```



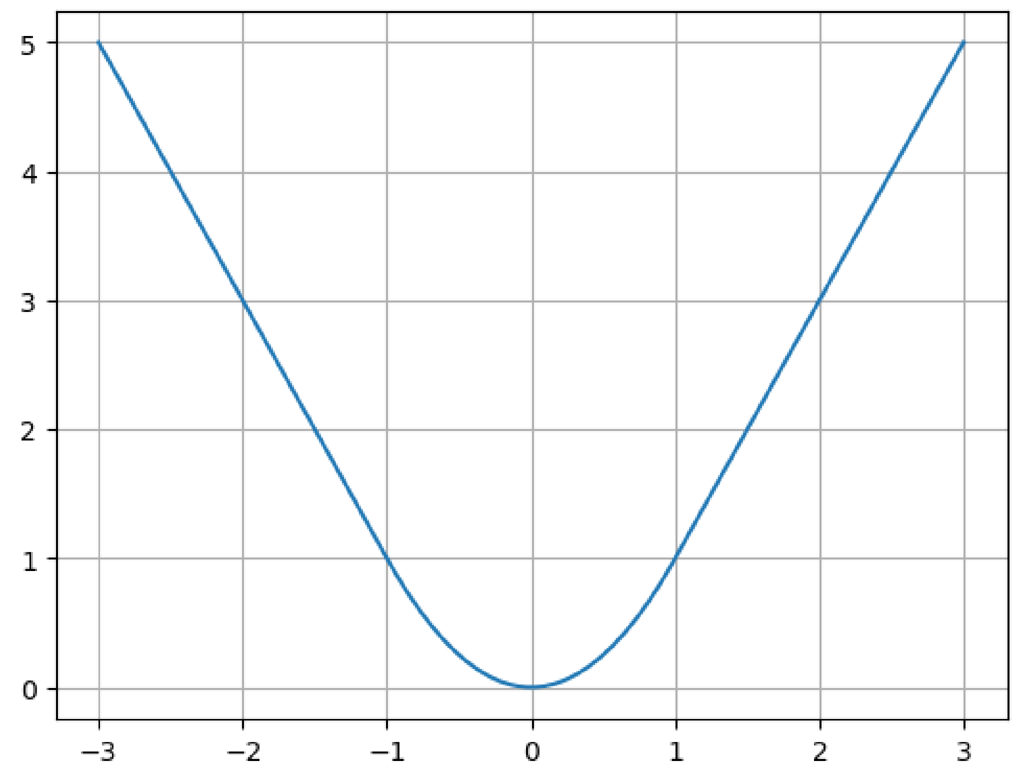

```
Academic license - for non-commercial use only
best fit of l1 cost function(include the outliers):
a:-0.35599999999999987
b:6.6659999999999995
```

Here we see that the l1 cost handles outliers better than least squares does. We can justify this by calculating the errors of l2 and l1 cost functions.

```
error_l2 = 0
error_l1 = 0
for i in 1:13
    error_l2 = error_l2 + (y_drop_outliers[i]-a1*x_drop_outliers[i]-b1)^2
    error_l1 = error_l1 + (y_drop_outliers[i]-a*x_drop_outliers[i]-b)^2
end
println("error_l2:", error_l2)
println("error_l1:", error_l1)
using JuMP, Gurobi, PyPlot
```

```
error_l2:115.98970497417054
error_l1:22.92230631250642
```

```
#Problem 2c
M = 1

X = linspace(-3, 3, 100)
y = []

m = Model(solver=GurobiSolver(OutputFlag=0))
function HuberLoss(x)
    @variable(m, v >= 0)
    @variable(m, w <= M)
    @constraint(m, x <= w+v)
    @constraint(m, -w-v <= x)
    @objective(m, Min, w^2 + 2*M*v)

    solve(m)

    return getobjectivevalue(m)
end

for x in X
    push!(y, HuberLoss(x))
end

plot(X, y)
grid("on")
```

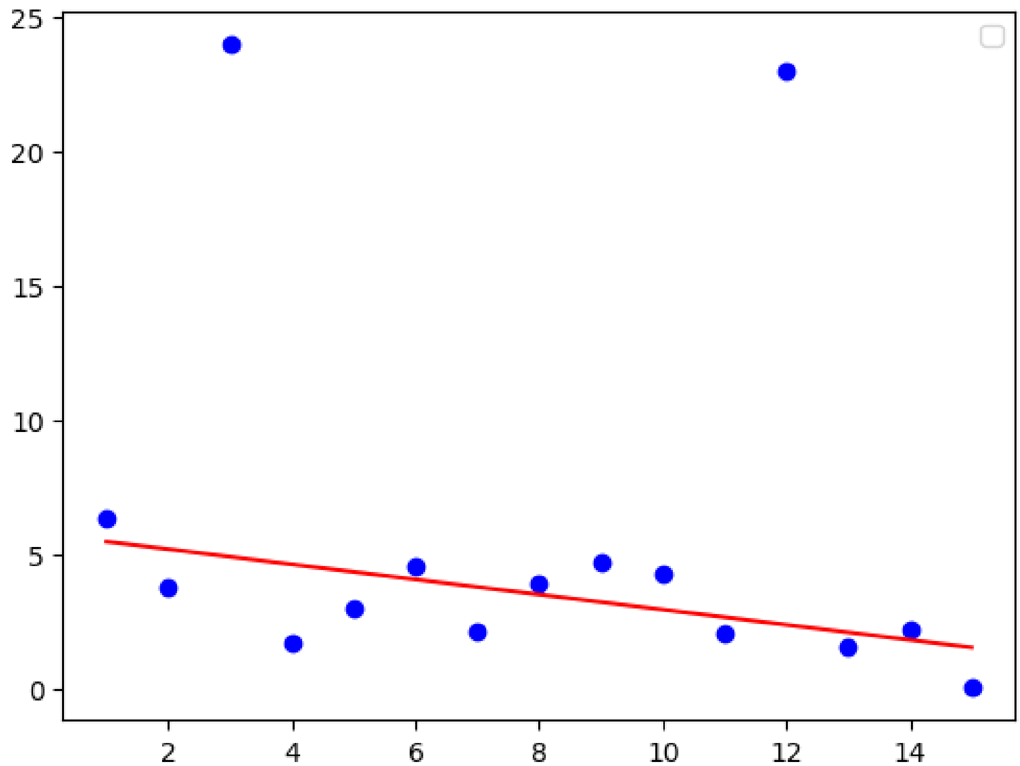



Then we can find the best fir to the data using Huber loss with M=1. This problem can be written as:

In [25]:

```
x = [1:15;]
y = [6.31 3.78 24 1.71 2.99 4.53 2.11 3.88 4.67 4.25 2.06 2
3 1.58 2.17 0.02]

using JuMP, Gurobi, PyPlot

M = 1
mh = Model(solver=GurobiSolver(OutputFlag=0))

@variable(mh, a)
@variable(mh, b)
@variable(mh, v[1:15] >= 0)
@variable(mh, w[1:15] <= M)

for i in 1:15
    @constraint(mh, y[i]-a*x[i]-b <= w[i]+v[i])
    @constraint(mh, -w[i]-v[i] <= y[i]-a*x[i]-b)
end
```

```
@objective(mh, Min, sum{w[i]^2+2*M*v[i], i=1:15})

solve(mh)

a = getvalue(a)
b = getvalue(b)

println("best fit of Huber loss(include the outliers):")
println("a:", a)
println("b:", b)

scatter(x, y, color="blue")
plot(x, a*x+b, color="red")
legend()
```




```
Academic license - for non-commercial use only
best fit of Huber loss(include the outliers):
a:-0.2811079944855355
b:5.738120618284127
No handles with labels found to put in legend.
```

Out[25]:

(3a) The flow can be $a_4 T r^{2} * \pi r^2 = a_4 \pi T r^4$

The geometric program is:

$$\max_{T,r,w} \; a_4 \pi T r^4$$

$$\text{S.T.} \quad T_{min} \le T \le T_{max}$$

$$r_{min} \le r \le r_{max}$$

$$w_{min} \le w \le w_{max}$$

$$w \le .1r$$

$$a_1 \frac{Tr}{w} + a_2 r + a_3 r w \le C_{max}$$

rewriting and applying log to both sides:

$$-a_4 \min_{T,r,w} \; -(\log T + 4\log r)$$

$$\text{S.T.} \quad \log T_{min} - \log T \le 0 \qquad \log T - \log T_{max} \le 0$$

$$\log r_{min} - \log r \le 0 \qquad \log r - \log r_{max} \le 0$$

$$\log w_{min} - \log w \le 0 \qquad \log w - \log w_{max} \le 0$$

$$\log 10 + \log w - \log r \le 0$$

$$\log\left(e^{\log \frac{a_1}{C_{max}} + \log T + \log r - \log w} + e^{\log \frac{a_2}{C_{max}} + \log r} + e^{\log \frac{a_3}{C_{max}} + \log r + \log w}\right) \le 0$$

which is a convex optimization problem

(3b)

Assume each variable has a lower bound of 0 and no upper bound.

Let $C_{max} = 500$ and $a_1 = a_2 = a_3 = a_4 = 1$:

$$-\pi \min_{x,y,z} \; -(x + 4y)$$

$$\text{S.T.} \quad \log 10 + z - y \le 0$$

$$\log\left(e^{\log \frac{1}{500} + x + y - z} + e^{\log \frac{1}{500} + y} + e^{\log \frac{1}{500} + y + z}\right) \le 0$$

In [26]:

```julia
using JuMP, Mosek

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

@variable(m, x)
@variable(m, y)
@variable(m, z)

@constraint(m, log(10) + z - y <= 0)
@NLconstraint(m, exp(-log(500) + x + y -z) + exp(-log(500)
+ y) + exp(-log(500) + y + z) <= 1)

@objective(m, Min, -(x + 4y))

solve(m)

x = getvalue(x)
y = getvalue(y)
z = getvalue(z)

println("x:", x)
println("y:", y)
println("z:", z)

println()

T = exp(x)
r = exp(y)
w = exp(z)

println("T:", T)
println("r:", r)
println("w:", w)

println("heat:", -pi*getobjectivevalue(m))

T*r/w + r + r*w
```

```
x:-1.3862943748502012
y:5.521460911013415
z:-0.6931472068641269

T:0.24999999656742236
```

```
r:249.99999828779218
w:0.49999998684790936
heat:65.02955191674955
```

Out[26]:

```
499.9999948592956
```