HW6 Starter Code

For Q1

In [30]:
```
using Pkg
Pkg.add("JuMP")
Pkg.add("HiGHS")
Pkg.add("Ipopt")
Pkg.add("LinearAlgebra")
Pkg.add("Gurobi")
```

```
    Updating registry at `C:\Users\X\.julia\registries\General.toml`
   Resolving package versions...
  No Changes to `C:\Users\X\.julia\environments\v1.8\Project.toml`
  No Changes to `C:\Users\X\.julia\environments\v1.8\Manifest.toml`
   Resolving package versions...
  No Changes to `C:\Users\X\.julia\environments\v1.8\Project.toml`
  No Changes to `C:\Users\X\.julia\environments\v1.8\Manifest.toml`
   Resolving package versions...
  No Changes to `C:\Users\X\.julia\environments\v1.8\Project.toml`
  No Changes to `C:\Users\X\.julia\environments\v1.8\Manifest.toml`
   Resolving package versions...
  No Changes to `C:\Users\X\.julia\environments\v1.8\Project.toml`
  No Changes to `C:\Users\X\.julia\environments\v1.8\Manifest.toml`
   Resolving package versions...
  No Changes to `C:\Users\X\.julia\environments\v1.8\Project.toml`
  No Changes to `C:\Users\X\.julia\environments\v1.8\Manifest.toml`
```

In [31]:
```
using JuMP, HiGHS, Ipopt
using LinearAlgebra

M = 20
N = 100

# generate the random polyhedron

b = rand(M)*100
A = rand(M,N)*4 .- 2

# model for 1-norm projection
norm1 = Model(HiGHS.Optimizer)

# here define variables for 1-norm projection
@variable(norm1, x1[1:N])

# model for 2-norm projection
norm2 = Model(Ipopt.Optimizer)

# here define variables for 2-norm projection
@variable(norm2, x2[1:N])

# model for infinity-norm projection
norm_inf = Model(HiGHS.Optimizer)

# here define variables for infinity-norm projection
```

```
@variable(norm_inf, x3[1:N])

# use these commands to suppress diagnostic output
set_silent(norm1)
set_silent(norm2)
set_silent(norm_inf)

# do "iteration" tests of random points to see what fraction are in the polyhedr

count_in=0
iteration = 1000

# Here insert "iteration" random trials for points and check to see whether they
# Increment count_in if they are.
# save a point x_hat that is NOT in P
# Calculate prob_in_P.

x_hat = zeros(N)
x_h = rand(iteration, N)*2 .- 1

i = 1
for i in 1:iteration
    if (all(A * x_h[i, :] .<= b))
        count_in = count_in + 1
    else
        x_hat = x_h[i, :]
    end
end
prob_in_P = count_in / iteration

print("probability of point in polyhedron = ",prob_in_P)

# here define constraints for the three projection models

# define z_norm2 to be the SQUARE of the 2-norm distance to P
# define z_norm1 to be the 1-norm distance to P
# define z_norm_inf to be the inf-norm distance to P

@variable(norm1, t[1:N])
@expression(norm1, z_norm1, sum(t[i] for i in 1:N))
@constraint(norm1, -t .<= x1 - x_hat)
@constraint(norm1, x1 - x_hat .<= t)

@expression(norm2, z_norm2, sum((x2[i] - x_hat[i])^2 for i in 1:N))

@variable(norm_inf, t)
@expression(norm1, z_norm_inf, t)
@constraint(norm_inf, -t .<= x3 - x_hat)
@constraint(norm_inf, x3 - x_hat .<= t)

# solve the three models and print results.

print("\n************** norm2 **************\n")
@objective(norm2, Min, z_norm2)
optimize!(norm2);
print("2-norm distance to P is ",sqrt(value(z_norm2)))

print("\n************** norm1 **************\n")
@objective(norm1, Min, z_norm1)
optimize!(norm1);
```

```
print("1-norm distance to P is ", value(z_norm1))

print("\n************** norm inf **************\n")
@objective(norm_inf, Min, z_norm_inf)
optimize!(norm_inf);
print("inf-norm distance to P is ",value(z_norm_inf))

print("\n*********************************\n")
```

```
probability of point in polyhedron = 0.485
************** norm2 **************

******************************************************************************
This program contains Ipopt, a library for large-scale nonlinear optimization.
 Ipopt is released as open source code under the Eclipse Public License (EPL).
         For more information visit https://github.com/coin-or/Ipopt
******************************************************************************

2-norm distance to P is 8.640746403294207e-8
************** norm1 **************
1-norm distance to P is 0.0
************** norm inf **************
inf-norm distance to P is -0.0
*********************************
```

For Q2

Eigenvectors and eigenvalues

a)

$$v = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, Q = \begin{bmatrix} 2 & 2 & -3 \\ 2 & 6 & -3 \\ -3 & -3 & 2 \end{bmatrix}.$$

b)

The eigenvalues of Q is about $\begin{bmatrix} -1.099 \\ 1.999 \\ 9.099 \end{bmatrix}$, which are not all large equal to 0, so Q is not positive semidefinite, so it is not an ellipsoid.

In [12]:
```
using LinearAlgebra

A = [2 2 -3; 2 6 -3; -3 -3 2]

# use eigvals() to get the eigenvalues of a matrix
eigvals(A)
```

Out[12]:
```
3-element Vector{Float64}:
 -1.0990195135927852
  1.9999999999999998
  9.099019513592784
```

In [13]:
```
# we can also use eigen() to obtain its eigenvalues and eigenvectors
# here L is the array of eigen values, U is the matrix of eigenvectors
(L, U) = eigen(A)
```

Out[13]:
```
Eigen{Float64, Float64, Matrix{Float64}, Vector{Float64}}
values:
3-element Vector{Float64}:
 -1.0990195135927703
  2.0000000000000036
  9.099019513592784
vectors:
3×3 Matrix{Float64}:
 -0.641359  -0.639602  -0.423754
 -0.138236   0.639602  -0.756175
 -0.754685   0.426401   0.49863
```

c)

$$v^TQv = v^TU\begin{bmatrix}-1.099 & 0 & 0\\ 0 & 1.999 & 0\\ 0 & 0 & 9.099\end{bmatrix}U^Tv = v^TU$$
$$\begin{bmatrix}0 & 0 & 0\\ 0 & 1.999 & 0\\ 0 & 0 & 9.099\end{bmatrix}U^Tv - v^TU$$
$$\begin{bmatrix}1.099 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}U^Tv = v^TQ_1v - v^TQ_2v = ||Q_1^{1/2}v||^2 - ||Q_2^{1/2}v||^2,$$

$$U = \begin{bmatrix}-0.641359 & -0.639602 & -0.423754\\ -0.138236 & 0.639602 & -0.756175\\ -0.754685 & 0.426401 & 0.49863\end{bmatrix}.$$

So,

$A = U\lambda_1^{1/2}U^T, B = U\lambda_2^{1/2}U^T$, culculated as below:

In [24]:
```
(L, U) = eigen(A)
lambda1 = diagm([0, L[2], L[3]])
lambda2 = diagm([L[1]*(-1), 0, 0])
a = U * sqrt(lambda1) * transpose(U)
```

Out[24]:
```
3×3 Matrix{Float64}:
  1.1202     0.388029  -1.02306
  0.388029   2.30335   -0.751666
 -1.02306   -0.751666   1.00712
```

In [25]:
```
b = U * sqrt(lambda2) * transpose(U)
```

Out[25]:
```
3×3 Matrix{Float64}:
 0.431226   0.0929445  0.507422
 0.0929445  0.0200328  0.109368
 0.507422   0.109368   0.597082
```

d)

Because the first eigenvalue is negative, so x, y, z can be in that direction to make the magnitude large and satisfy the constraint, which is $\begin{bmatrix} -0.641359 \\ -0.138236 \\ -0.754685 \end{bmatrix}$. x, y, z can be multiple of it.

For Q3

1. Plotting

In [34]:
```
using PyPlot, CSV, DataFrames

data = CSV.read("lasso_data.csv", DataFrame)
x = data[:,1]
y = data[:,2]

function plotpoints()
    plot(x, y, ".");
    xlabel("x"); ylabel("y");
    n=size(data, 1);
end
```

Out[34]:
```
plotpoints (generic function with 1 method)
```

In [64]:
```
# plot a line on top of it

f(x) = -6 * x + 1.8;

# points of the line
xs = range(0.1, 0.5, length=100);
ys = f.(xs);

function plot2(xs, ys, d)
    # plot the scatters again
    # plot(x, y, ".");
    plotpoints()
    # plot the line
    plot(xs,ys,"-");
    title(string("polynomial fit (deg=", d, ") ")); # using ordinary least squar
    # xlabel("x"); ylabel("y");
end
```

Out[64]:
```
plot2 (generic function with 1 method)
```

a)

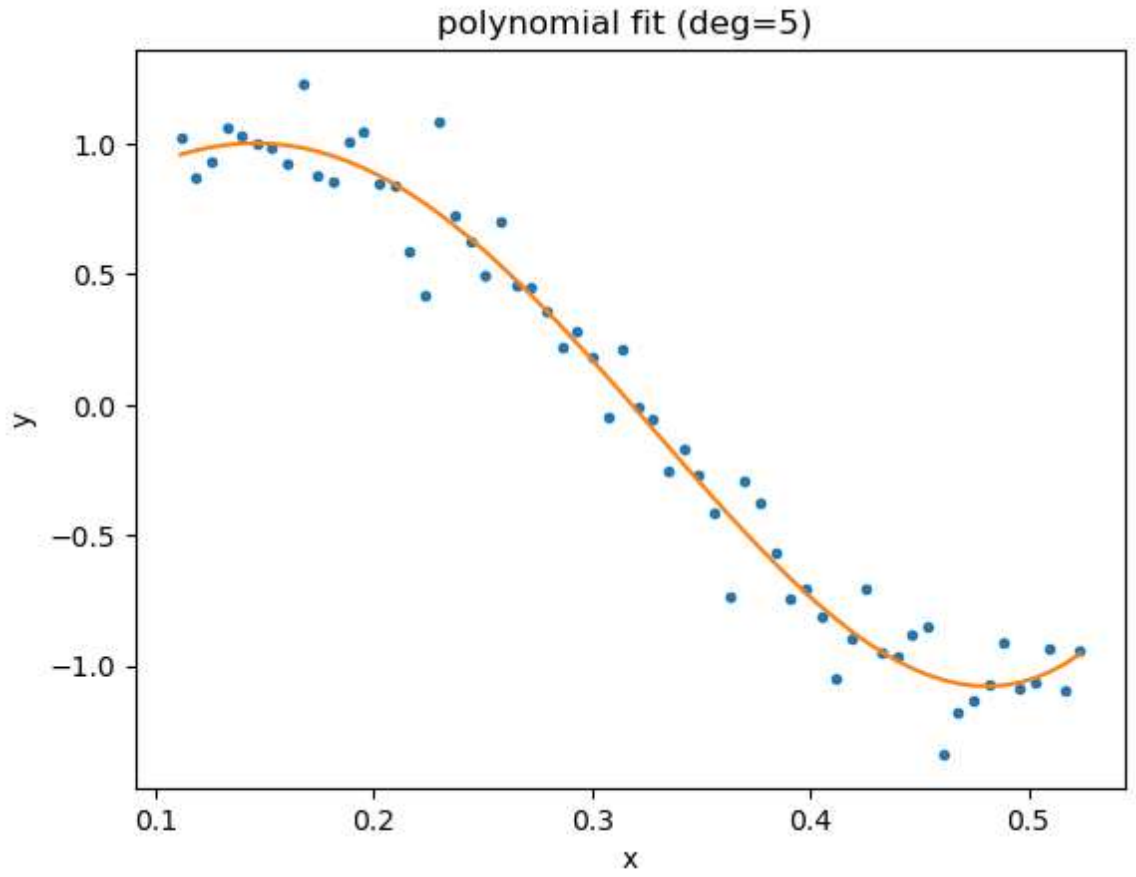
The magnitudes of the coefficients in the resulting polynomial fits are large.

In [65]:
```
using Gurobi

# Order of polynomial used for fitting
k = 5

# Create the A matrix (where each row corresponds to one observation x_i and eac
```

```
n = length(x)
A = zeros(n,k+1)
for i = 1:n
    for j = 1:k+1
        A[i,j] = x[i]^(k+1-j)
    end
end

m = Model(Gurobi.Optimizer)

@variable(m, u[1:k+1])
@objective(m, Min, sum( (y - A*u).^2 ) )

optimize!(m)

println(value.(u))
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2024-02-24
Gurobi Optimizer version 10.0.1 build v10.0.1rc0 (win64)

CPU model: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, instruction set [SSE2|AVX|
AVX2]
Thread count: 6 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 0 rows, 6 columns and 0 nonzeros
Model fingerprint: 0x5170d1f1
Model has 21 quadratic objective terms
Coefficient statistics:
  Matrix range     [0e+00, 0e+00]
  Objective range  [9e-01, 1e+01]
  QObjective range [2e-02, 1e+02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [0e+00, 0e+00]
Presolve time: 0.00s
Presolved: 0 rows, 6 columns, 0 nonzeros
Presolved model has 21 quadratic objective terms
Ordering time: 0.00s

Barrier statistics:
 Free vars  : 11
 AA' NZ     : 1.000e+01
 Factor NZ  : 1.500e+01
 Factor Ops : 5.500e+01 (less than 1 second per iteration)
 Threads    : 1

                 Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   3.89193022e+01  3.89193022e+01  0.00e+00 2.88e+04  0.00e+00     0s
   1   3.85214444e+01  3.89182495e+01  2.10e-08 2.87e+04  0.00e+00     0s
   2   3.81335071e+01  3.89151743e+01  4.19e-08 2.85e+04  0.00e+00     0s
   3   3.76319355e+01  3.89081298e+01  1.04e-07 2.83e+04  0.00e+00     0s
   4   3.59854377e+01  3.88599488e+01  2.31e-07 2.77e+04  0.00e+00     0s
   5   3.43346343e+01  3.87706178e+01  5.08e-07 2.70e+04  0.00e+00     0s
   6   2.99729589e+01  3.83137905e+01  1.03e-06 2.52e+04  0.00e+00     0s
   7   1.70915607e+01  3.42967433e+01  1.35e-06 1.88e+04  0.00e+00     0s
   8   6.27027332e+00  2.39136805e+01  2.36e-06 1.07e+04  0.00e+00     0s
   9   1.07118873e+00  1.01402113e+00  1.92e-06 1.07e-02  0.00e+00     0s
  10   1.01399311e+00  1.01399301e+00  3.57e-12 1.07e-08  0.00e+00     0s

Barrier solved model in 10 iterations and 0.00 seconds (0.00 work units)
Optimal objective 1.01399311e+00


User-callback calls 57, time in user-callback 0.00 sec
[-247.38754866715516, 495.9334908255358, -254.1415654489827, 16.75730132518639
7, 5.737209156076103, 0.3898945489261928]
```

In [66]:
```
plot2(x, value.(A*u), 5)
```



Out[66]: PyObject Text(0.5, 1.0, 'polynomial fit (deg=5) ')

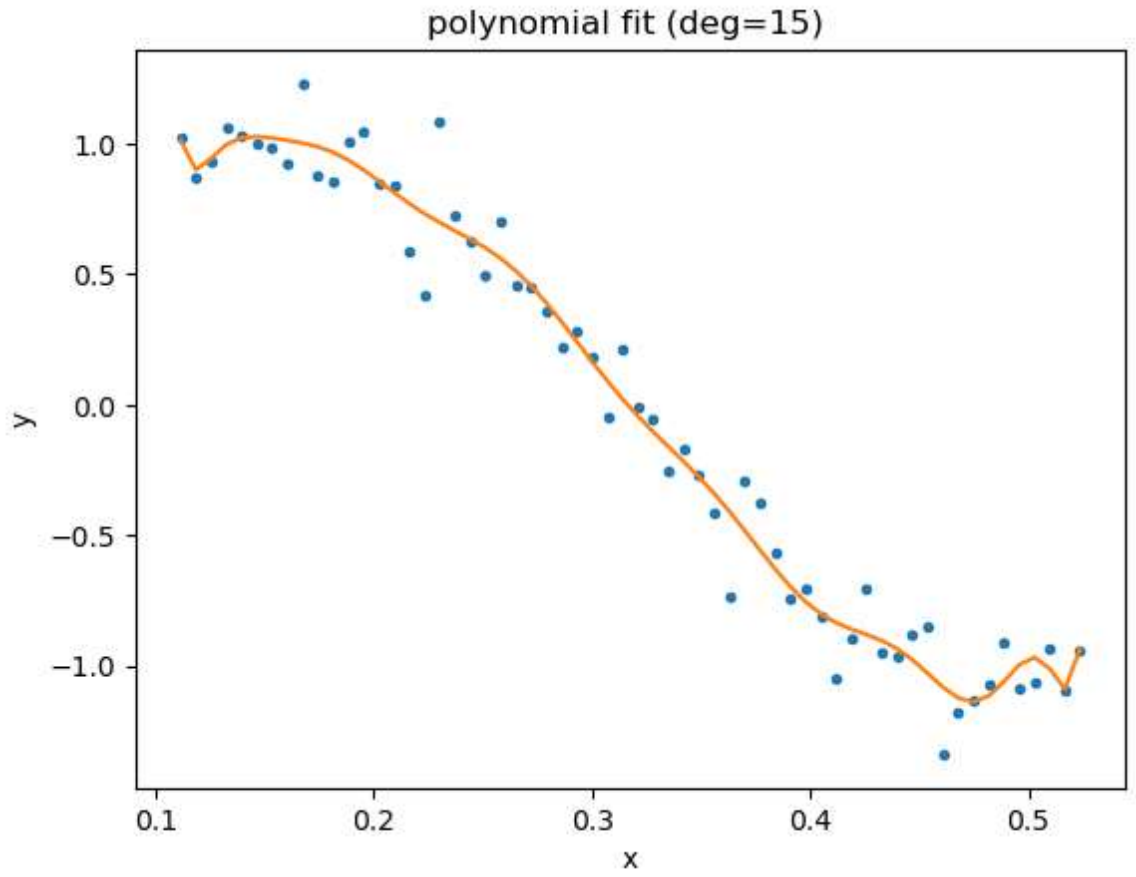
In [67]:
```
using Gurobi

# Order of polynomial used for fitting
k = 15

# Create the A matrix (where each row corresponds to one observation x_i and eac
n = length(x)
A = zeros(n,k+1)
for i = 1:n
    for j = 1:k+1
        A[i,j] = x[i]^(k+1-j)
    end
end

m = Model(Gurobi.Optimizer)

@variable(m, u[1:k+1])
@objective(m, Min, sum( (y - A*u).^2 ) )

optimize!(m)

u = A\y
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2024-02-24
Gurobi Optimizer version 10.0.1 build v10.0.1rc0 (win64)

CPU model: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, instruction set [SSE2|AVX|
AVX2]
Thread count: 6 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 0 rows, 16 columns and 0 nonzeros
Model fingerprint: 0xb0d3a865
Model has 136 quadratic objective terms
Coefficient statistics:
  Matrix range     [0e+00, 0e+00]
  Objective range  [6e-04, 1e+01]
  QObjective range [2e-08, 1e+02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [0e+00, 0e+00]
Presolve time: 0.00s
Presolved: 0 rows, 16 columns, 0 nonzeros
Presolved model has 136 quadratic objective terms
Ordering time: 0.00s

Barrier statistics:
 Free vars  : 24
 AA' NZ     : 2.800e+01
 Factor NZ  : 3.600e+01
 Factor Ops : 2.040e+02 (less than 1 second per iteration)
 Threads    : 1

                 Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   3.89193022e+01  3.89193022e+01  0.00e+00 7.45e+00  0.00e+00     0s
   1   1.01407204e+00  1.02152533e+00  4.92e-09 6.98e-05  0.00e+00     0s
   2   1.00888453e+00  1.01293923e+00  1.22e-08 4.07e-06  0.00e+00     0s
   3  -1.78610828e-01  1.01305424e+00  1.46e-08 3.59e-06  0.00e+00     0s
   4  -3.68020836e+00  1.01314276e+00  6.87e-09 7.65e-07  0.00e+00     0s
   5  -7.18176452e+00  1.01314276e+00  8.84e-11 7.65e-07  0.00e+00     0s
   6  -1.06833207e+01  1.01314276e+00  3.01e-10 7.65e-07  0.00e+00     0s
   7  -1.41848769e+01  1.01314276e+00  7.54e-10 7.65e-07  0.00e+00     0s
   8  -1.76864331e+01  1.01314276e+00  3.61e-10 7.65e-07  0.00e+00     0s
   9  -2.11879893e+01  1.01314276e+00  4.25e-10 7.65e-07  0.00e+00     0s
  10  -2.46895454e+01  1.01314276e+00  7.08e-10 7.65e-07  0.00e+00     0s
  11  -2.81911016e+01  1.01314276e+00  3.92e-10 7.65e-07  0.00e+00     0s
  12  -3.16926578e+01  1.01314276e+00  1.28e-09 7.65e-07  0.00e+00     0s
  13  -3.51942140e+01  1.01314276e+00  2.56e-09 7.65e-07  0.00e+00     0s
  14  -3.86957702e+01  1.01314276e+00  1.02e-09 7.65e-07  0.00e+00     0s
  15  -4.21973264e+01  1.01314276e+00  7.69e-10 7.65e-07  0.00e+00     0s
  16  -4.56988826e+01  1.01314276e+00  5.14e-10 7.65e-07  0.00e+00     0s
  17  -4.92004387e+01  1.01314276e+00  1.71e-09 7.65e-07  0.00e+00     0s
  18  -5.27019949e+01  1.01314276e+00  9.34e-10 7.65e-07  0.00e+00     0s
  19  -5.62035511e+01  1.01314276e+00  8.88e-10 7.65e-07  0.00e+00     0s
  20  -5.97051073e+01  1.01314276e+00  3.26e-09 7.65e-07  0.00e+00     0s
  21  -6.32066635e+01  1.01314276e+00  1.31e-09 7.65e-07  0.00e+00     0s
  22  -6.67082197e+01  1.01314276e+00  1.27e-09 7.65e-07  0.00e+00     0s
  23  -7.02097759e+01  1.01314276e+00  2.11e-09 7.65e-07  0.00e+00     0s
  24  -7.37113321e+01  1.01314276e+00  4.71e-09 7.65e-07  0.00e+00     0s
  25  -7.72128882e+01  1.01314276e+00  4.77e-09 7.65e-07  0.00e+00     0s
  26  -8.07144444e+01  1.01314276e+00  3.09e-09 7.65e-07  0.00e+00     0s
  27  -8.42160006e+01  1.01314276e+00  1.81e-09 7.65e-07  0.00e+00     0s
  28  -8.77175568e+01  1.01314276e+00  2.15e-09 7.65e-07  0.00e+00     0s
```

```
  29  -9.12191130e+01  1.01314276e+00  4.42e-09 7.65e-07  0.00e+00     0s
  30  -9.47206691e+01  1.01314276e+00  1.00e-09 7.65e-07  0.00e+00     0s
  31  -9.82222253e+01  1.01314276e+00  8.83e-10 7.65e-07  0.00e+00     0s
  32  -1.01723782e+02  1.01314276e+00  3.34e-09 7.65e-07  0.00e+00     0s
  33  -1.05225338e+02  1.01314276e+00  3.93e-09 7.65e-07  0.00e+00     0s
  34  -1.08726894e+02  1.01314276e+00  3.58e-09 7.65e-07  0.00e+00     0s
  35  -1.12228450e+02  1.01314276e+00  2.48e-09 7.65e-07  0.00e+00     0s
  36  -1.15730006e+02  1.01314276e+00  1.33e-08 7.65e-07  0.00e+00     0s
  37  -1.19231562e+02  1.01314276e+00  6.78e-09 7.65e-07  0.00e+00     0s
  38  -1.22733119e+02  1.01314276e+00  2.14e-09 7.65e-07  0.00e+00     0s
  39  -1.26234675e+02  1.01314276e+00  3.31e-09 7.65e-07  0.00e+00     0s
  40  -1.29736231e+02  1.01314276e+00  3.21e-09 7.65e-07  0.00e+00     0s
  41  -1.33237787e+02  1.01314276e+00  3.65e-09 7.65e-07  0.00e+00     0s
  42  -1.36739343e+02  1.01314276e+00  9.92e-09 7.65e-07  0.00e+00     0s
  43  -1.40240900e+02  1.01314276e+00  1.38e-08 7.65e-07  0.00e+00     0s
  44  -1.43742456e+02  1.01314276e+00  1.32e-08 7.65e-07  0.00e+00     0s
  45  -1.47244012e+02  1.01314276e+00  9.29e-09 7.65e-07  0.00e+00     0s
  46  -1.50745568e+02  1.01314276e+00  1.17e-08 7.65e-07  0.00e+00     0s
  47  -1.54247124e+02  1.01314276e+00  7.48e-09 7.65e-07  0.00e+00     0s
  48  -1.57748680e+02  1.01314276e+00  1.45e-09 7.65e-07  0.00e+00     0s
  49  -1.61250237e+02  1.01314276e+00  2.75e-09 7.65e-07  0.00e+00     0s
  50  -1.64751793e+02  1.01314276e+00  6.20e-09 7.65e-07  0.00e+00     0s
  51  -1.68253349e+02  1.01314276e+00  9.40e-09 7.65e-07  0.00e+00     0s

Barrier performed 51 iterations in 0.00 seconds (0.00 work units)
Numerical trouble encountered

Model may be infeasible or unbounded.  Consider using the
homogeneous algorithm (through parameter 'BarHomogeneous')


User-callback calls 142, time in user-callback 0.00 sec
```

Out[67]:
```
16-element Vector{Float64}:
       8.10196734771287e11
      -2.711203586668869e12
       3.414033253041093e12
      -1.1828012881790938e12
      -1.9910418508955015e12
       3.3098175101053374e12
      -2.588031315622572e12
       1.313321578703679e12
      -4.700347018226031e11
       1.2226835359193762e11
      -2.3274602520483597e10
       3.2112202895422864e9
      -3.1250932858722436e8
       2.0323937244193077e7
 -791829.5776914086
   13959.473392449543
```

In [68]:
```
plot2(x, value.(A*u), 15)
```



Out[68]: PyObject Text(0.5, 1.0, 'polynomial fit (deg=15) ')

b)

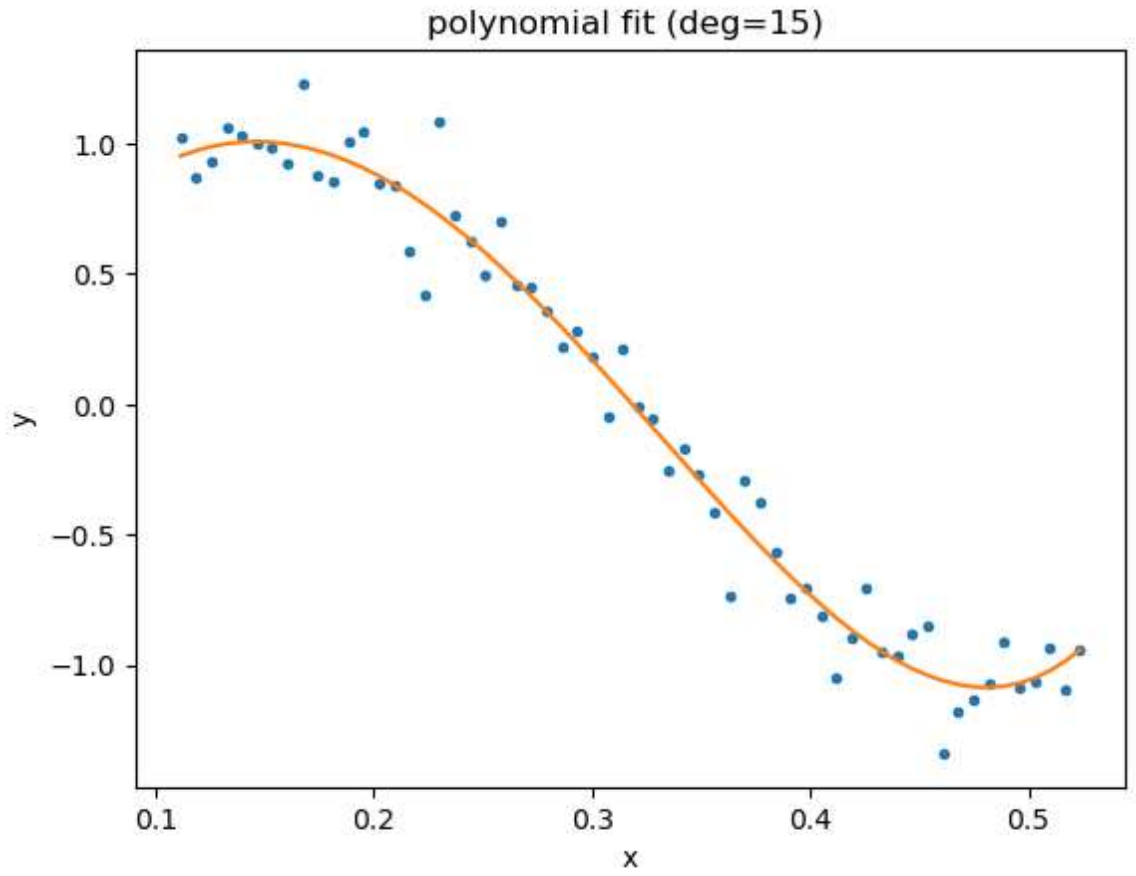
The fit is more smooth. The magnitudes of the coefficients become smaller.

In [69]:
```
lambda = 1e-6

# Order of polynomial used for fitting
k = 15

# Create the A matrix (where each row corresponds to one observation x_i and eac
n = length(x)
A = zeros(n,k+1)
for i = 1:n
    for j = 1:k+1
        A[i,j] = x[i]^(k+1-j)
    end
end

m = Model(Gurobi.Optimizer)

@variable(m, u[1:k+1])
@objective(m, Min, sum( (y - A*u).^2 )+lambda * sum(u.^2) )

optimize!(m)

u = value.(u)
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2024-02-24
Gurobi Optimizer version 10.0.1 build v10.0.1rc0 (win64)

CPU model: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, instruction set [SSE2|AVX|
AVX2]
Thread count: 6 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 0 rows, 16 columns and 0 nonzeros
Model fingerprint: 0xbc5f72a2
Model has 136 quadratic objective terms
Coefficient statistics:
  Matrix range     [0e+00, 0e+00]
  Objective range  [6e-04, 1e+01]
  QObjective range [9e-08, 1e+02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [0e+00, 0e+00]
Presolve time: 0.00s
Presolved: 0 rows, 16 columns, 0 nonzeros
Presolved model has 136 quadratic objective terms
Ordering time: 0.00s

Barrier statistics:
 Free vars  : 31
 AA' NZ     : 1.050e+02
 Factor NZ  : 1.200e+02
 Factor Ops : 1.240e+03 (less than 1 second per iteration)
 Threads    : 1

                  Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   3.89193022e+01  3.89193022e+01  0.00e+00 7.61e+00  0.00e+00     0s
   1   1.02255860e+00  1.02901621e+00  5.72e-09 6.10e-05  0.00e+00     0s
   2   1.02232900e+00  1.02232928e+00  1.75e-08 4.94e-07  0.00e+00     0s
   3   1.02232895e+00  1.02232895e+00  2.20e-11 2.12e-11  0.00e+00     0s

Barrier solved model in 3 iterations and 0.00 seconds (0.00 work units)
Optimal objective 1.02232895e+00


User-callback calls 43, time in user-callback 0.00 sec
```

Out[69]:
```
16-element Vector{Float64}:
  -0.5678886387696518
  -0.934125691797301
  -1.4894063560214876
  -2.2752867235667322
  -3.266879495767818
  -4.253387313194705
  -4.621193052728553
  -3.055094758523602
   2.6232774943328967
  14.708534820390213
  32.18768266640952
  42.66479301924787
  15.60090251702806
 -58.101276059176634
  15.382605620637523
  -0.06959120627288001
```

In [71]:
```
plot2(x, value.(A*u), 15)
```



Out[71]:
```
PyObject Text(0.5, 1.0, 'polynomial fit (deg=15) ')
```

c)

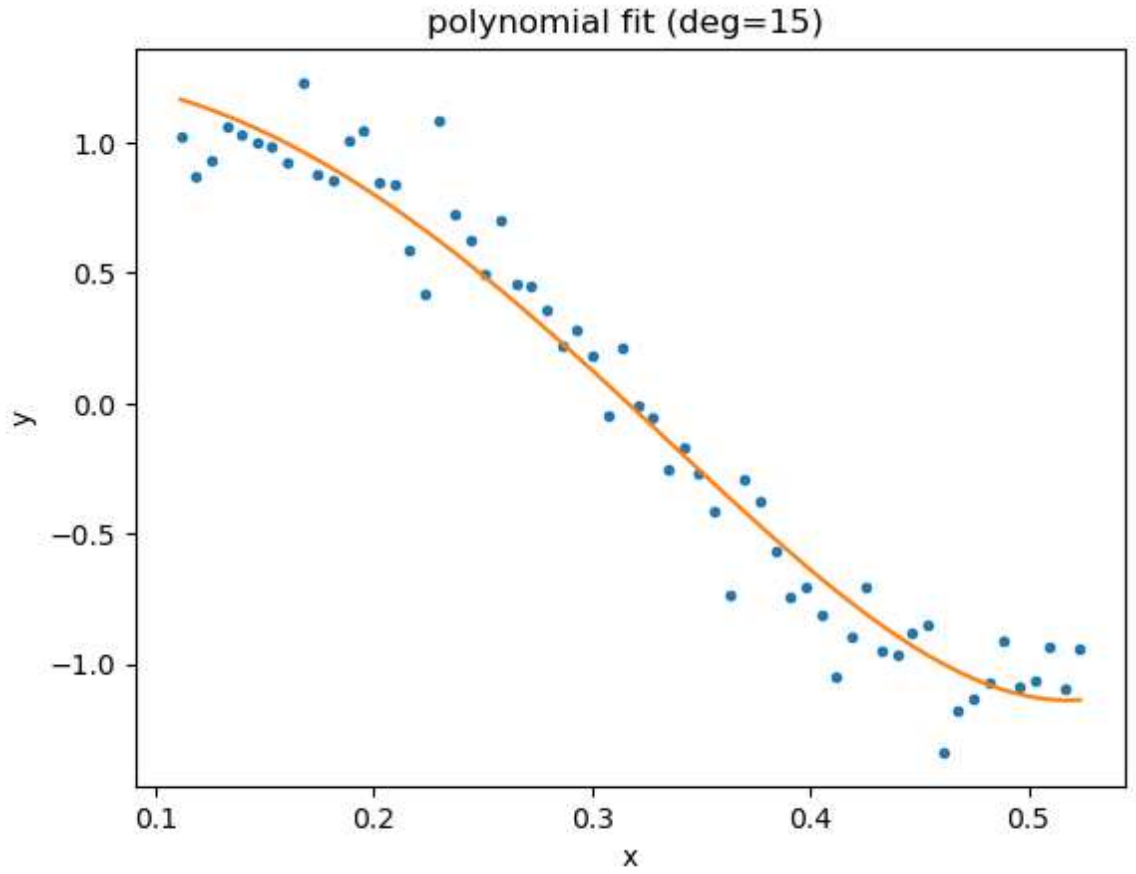
In [96]:
```
lambda = 1e-2

# Order of polynomial used for fitting
k = 15

# Create the A matrix (where each row corresponds to one observation x_i and eac
n = length(x)
A = zeros(n,k+1)
for i = 1:n
    for j = 1:k+1
        A[i,j] = x[i]^(k+1-j)
    end
end

m = Model(Gurobi.Optimizer)

@variable(m, u[1:k+1])
@variable(m, t[1:k+1])
@objective(m, Min, sum( (y - A*u).^2 )+lambda * sum(t) )
@constraint(m, u .<= t)
@constraint(m, -t .<= u)

optimize!(m)

u = value.(u)
```

```
# for i in 1:k+1
#     if u[i] < 1e-5
#         u[i] = 0
#     end
# end
# print(u)
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2024-02-24
Gurobi Optimizer version 10.0.1 build v10.0.1rc0 (win64)

CPU model: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, instruction set [SSE2|AVX|
AVX2]
Thread count: 6 physical cores, 12 logical processors, using up to 12 threads

Optimize a model with 32 rows, 32 columns and 64 nonzeros
Model fingerprint: 0xa5fd5fdc
Model has 136 quadratic objective terms
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [6e-04, 1e+01]
  QObjective range [2e-08, 1e+02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [0e+00, 0e+00]
Presolve removed 16 rows and 0 columns
Presolve time: 0.00s
Presolved: 16 rows, 32 columns, 32 nonzeros
Presolved model has 136 quadratic objective terms
Ordering time: 0.00s

Barrier statistics:
 Free vars  : 24
 AA' NZ     : 1.270e+02
 Factor NZ  : 3.000e+02
 Factor Ops : 4.900e+03 (less than 1 second per iteration)
 Threads    : 1

                 Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   2.38919302e+02  3.89193022e+01  0.00e+00 3.73e+00  1.00e+06     0s
   1   2.04247933e+02  1.59007009e+01  2.95e-09 1.02e+00  2.02e+03     0s
   2   2.00524311e+02  2.47628955e+00  1.32e-08 5.92e-03  1.34e+01     0s
   3   6.05829795e+01  1.81167786e+00  1.24e-08 7.33e-06  1.84e+00     0s
   4   2.65938311e+00  1.89221752e+00  3.14e-08 8.41e-07  2.40e-02     0s
   5   2.25400988e+00  2.16258829e+00  5.85e-09 4.74e-09  2.86e-03     0s
   6   2.17928819e+00  2.17057743e+00  5.90e-09 4.41e-10  2.72e-04     0s
   7   2.17436476e+00  2.17416249e+00  8.62e-14 5.83e-12  6.32e-06     0s
   8   2.17423797e+00  2.17423777e+00  1.27e-12 2.22e-10  6.35e-09     0s

Barrier solved model in 8 iterations and 0.00 seconds (0.00 work units)
Optimal objective 2.17423797e+00


User-callback calls 102, time in user-callback 0.00 sec
```

Out[96]:
```
16-element Vector{Float64}:
   4.307785693238739e-9
   8.123577257654983e-9
   1.5219471477613384e-8
   2.827746372203576e-8
   5.2011655633996665e-8
   9.465168061501895e-8
   1.7123323210537725e-7
   3.149018611525007e-7
   6.364434005583881e-7
   1.8836762456719228e-6
  49.27184239931585
   1.7604396218617868e-5
   1.657472429674768e-7
 -18.58190159019269
   1.5062833546659014
   1.2285896146848978
```

In [97]:
```
plot2(x, value.(A*u), 15)
```



Out[97]:
```
PyObject Text(0.5, 1.0, 'polynomial fit (deg=15) ')
```

In [ ]: