Problem 1 - Least Squares in general

The small problem (a)

Linear Programming Instance for Part (a)

Objective:

Minimize $r_1^+ + r_1^- + r_2^+ + r_2^- + r_3^+ + r_3^- + r_4^+ + r_4^- + r_5^+ + r_5^-$

Subject to:

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 - 3 &= r_1^+ - r_1^- \\ 2x_1 - x_4 - 1 &= r_2^+ - r_2^- \\ -3x_2 + x_3 &= r_3^+ - r_3^- \\ x_1 - x_2 - x_4 + 1 &= r_4^+ - r_4^- \\ 2x_3 + 3x_4 - 6 &= r_5^+ - r_5^- \end{aligned}$$

$$r_1^+, r_1^-, r_2^+, r_2^-, r_3^+, r_3^-, r_4^+, r_4^-, r_5^+, r_5^- \geq 0$$

In [2]:
```
using JuMP, Clp, GLPK

model = Model(GLPK.Optimizer)

@variable(model, x[1:4])
@variable(model, r[1:5] >= 0)

@constraint(model, r[1] >=  x[1] + x[2] + x[3] + x[4] - 3)
@constraint(model, r[1] >= -x[1] - x[2] - x[3] - x[4] + 3)

@constraint(model, r[2] >=  2x[1] - x[4] - 1)
@constraint(model, r[2] >= -2x[1] + x[4] + 1)

@constraint(model, r[3] >= -3x[2] + x[3])
@constraint(model, r[3] >=  3x[2] - x[3])

@constraint(model, r[4] >=  x[1] - x[2] - x[4] + 1)
@constraint(model, r[4] >= -x[1] + x[2] + x[4] - 1)

@constraint(model, r[5] >=  2x[3] + 3x[4] - 6)
@constraint(model, r[5] >= -2x[3] - 3x[4] + 6)

@objective(model, Min, sum(r))

optimize!(model)
```

```
x_opt = value.(x)
total_residual = objective_value(model)

println("Optimal x values: ", x_opt)
println("Minimum total residual: ", total_residual)
```

```
Optimal x values: [2.0, -0.5, -1.5, 3.0]
Minimum total residual: 0.5000000000000001
```

The small problem (b)

Linear Programming Instance for Part (b)

Objective:

Minimize t

Subject to:

$$\begin{aligned}
x_1 + x_2 + x_3 + x_4 - 3 &= r_1^+ - r_1^- \\
2x_1 - x_4 - 1 &= r_2^+ - r_2^- \\
-3x_2 + x_3 &= r_3^+ - r_3^- \\
x_1 - x_2 - x_4 + 1 &= r_4^+ - r_4^- \\
2x_3 + 3x_4 - 6 &= r_5^+ - r_5^-
\end{aligned}$$

$$\begin{aligned}
&r_1^+ + r_1^- \leq t \\
&r_2^+ + r_2^- \leq t \\
&r_3^+ + r_3^- \leq t \\
&r_4^+ + r_4^- \leq t \\
&r_5^+ + r_5^- \leq t
\end{aligned}$$

$$r_1^+, r_1^-, r_2^+, r_2^-, r_3^+, r_3^-, r_4^+, r_4^-, r_5^+, r_5^-, t \geq 0$$

In [7]:
```
model = Model(Clp.Optimizer)

@variable(model, x[1:4])
@variable(model, r[1:5] >= 0)
@variable(model, z >= 0)

@constraint(model, r[1] >=  x[1] + x[2] + x[3] + x[4] - 3)
@constraint(model, r[1] >= -x[1] - x[2] - x[3] - x[4] + 3)

@constraint(model, r[2] >=  2x[1] - x[4] - 1)
@constraint(model, r[2] >= -2x[1] + x[4] + 1)

@constraint(model, r[3] >= -3x[2] + x[3])
@constraint(model, r[3] >=  3x[2] - x[3])
```

```
@constraint(model, r[4] >=  x[1] - x[2] - x[4] + 1)
@constraint(model, r[4] >= -x[1] + x[2] + x[4] - 1)

@constraint(model, r[5] >=  2x[3] + 3x[4] - 6)
@constraint(model, r[5] >= -2x[3] - 3x[4] + 6)

@constraint(model, r[1] <= z)
@constraint(model, r[2] <= z)
@constraint(model, r[3] <= z)
@constraint(model, r[4] <= z)
@constraint(model, r[5] <= z)

@objective(model, Min, z)

optimize!(model)

x_opt = value.(x)
max_residual = value(z)

println("Optimal x values: ", x_opt)
println("Minimum max-residual: ", max_residual)
```

```
Optimal x values: [1.75, -0.25, -1.0, 2.75]
Minimum max-residual: 0.25
Coin0506I Presolve 10 (-5) rows, 5 (-5) columns and 36 (-10) elements
Clp0006I 0  Obj 0 Primal inf 6.499996 (4)
Clp0006I 5  Obj 0.3
Clp0006I 5  Obj 2.5e+09 Primal inf 2.5e+09 (1) Dual inf 1.101e+13 (3) w.o. f
ree dual inf (2)
Clp0006I 7  Obj 0.25
Clp0000I Optimal - objective value 0.25
Coin0511I After Postsolve, objective 0.25, infeasibilities - dual 0 (0), pri
mal 0 (0)
Clp0032I Optimal objective 0.25 - 7 iterations time 0.002, Presolve 0.00
```

What is the minimum max-residual that can be achieved?

The minimum max-residual that can be achieved is 0.25

The big problem (c)

Given:

- $J = 1, 2, \ldots, |J|$: Set representing the indices of decision variables.
- $I = 1, 2, \ldots, |I|$: Set representing the indices of equations, where $|I| > |J|$.

The problem is defined by the following system of equations: $\sum_{j \in J} a_{ij} x_j = b_i \quad \forall i \in I$

Define the absolute residual for each equation (i \in I) as:
$r_i(x^*) = \left| \sum_{j \in J} (a_{ij} x_j) - b_i \right| \quad \forall i \in I$

To minimize the total residual, we can write the linear programming problem as follows:

Objective:

Minimize the total residual: $\min \sum_{i \in I} r_i$

Constraints:

1. For each equation (i in I): $r_i \geq \sum_{j \in J} a_{ij} x_j - b_i$ $r_i \geq b_i - \sum_{j \in J} a_{ij} x_j$

2. Non-negativity constraints for the residuals: $r_i \geq 0 \quad \forall i \in I$

Decision Variables:

- x_j: The decision variables for $j \in J$.
- r_i: The residuals for $i \in I$.

In [5]:
```
nr = 400  # number of rows of the coefficient matrix (number of equations)
nc = 200  # number of columns of the coefficient matrix (number of decision

A = zeros(nr, nc)  # initialize coefficient matrix
b = zeros(nr)  # initialize right-hand side of equations

for i in 1:nr
    b[i] = rand(-75:75)   # generate random right-hand side
    for j in 1:nc
        A[i, j] = rand(-7:7)  # generate random coefficient
    end
end
```

In [6]:
```
# minimize the total residual

model_total = Model(GLPK.Optimizer)

# Define variables
@variable(model_total, x[1:nc])
@variable(model_total, r[1:nr] >= 0)

# Define constraints for residuals
for i in 1:nr
    @constraint(model_total, r[i] >= sum(A[i, j] * x[j] for j in 1:nc) - b[i
    @constraint(model_total, r[i] >= -sum(A[i, j] * x[j] for j in 1:nc) + b[
end

# Define objective to minimize the total residual
@objective(model_total, Min, sum(r))

# Optimize the model
optimize!(model_total)

# Get results
x_opt_total = value.(x)
total_residual = objective_value(model_total)
```

```
# Display the results
println("Minimum total residual: ", total_residual)
```

```
Minimum total residual: 8824.582838577906
```

The minimum total residue is 8816.605015799909

The big problem (d)

Given:

- $J = 1, 2, \ldots, |J|$: Set representing the indices of decision variables.
- $I = 1, 2, \ldots, |I|$: Set representing the indices of equations, where $|I| > |J|$.

The problem is defined by the following system of equations: $\sum_{j \in J} a_{ij} x_j = b_i \quad \forall i \in I$

Define the absolute residual for each equation $i \in I$ as:
$r_i(x^*) = \left| \sum_{j \in J} (a_{ij} x_j) - b_i \right| \quad \forall i \in I$

To minimize the maximum residual, we can write the linear programming problem as follows:

Objective:

Minimize the maximum residual: $\min \max_{i \in I} r_i$

Constraints:

1. For each equation (i \in I): $r_i \geq \sum_{j \in J} a_{ij} x_j - b_i \; r_i \geq b_i - \sum_{j \in J} a_{ij} x_j$
2. Non-negativity constraints for the residuals: $r_i \geq 0 \quad \forall i \in I$
3. Introduce a new variable (R) to represent the maximum residual: $R \geq r_i \quad \forall i \in I$

Decision Variables:

- x_j: The decision variables for $j \in J$.
- r_i: The residuals for $i \in I$.
- R: The maximum residual.

In [8]:
```
# minimizing the max residual

model_max = Model(Clp.Optimizer)

# Define variables
@variable(model_max, x[1:nc])
@variable(model_max, r[1:nr] >= 0)
@variable(model_max, z >= 0)

# Define constraints for residuals
```

```
for i in 1:nr
    @constraint(model_max, r[i] >= sum(A[i, j] * x[j] for j in 1:nc) - b[i])
    @constraint(model_max, r[i] >= -sum(A[i, j] * x[j] for j in 1:nc) + b[i]
end

# Define constraints for maximum residual
for i in 1:nr
    @constraint(model_max, r[i] <= z)
end

# Define objective to minimize the maximum residual
@objective(model_max, Min, z)

# Optimize the model
optimize!(model_max)

# Get results
x_opt_max = value.(x)
max_residual = value(z)

# Display the results
println("Optimal x values for minimizing maximum residual: ", x_opt_max)
println("Minimum maximum residual: ", max_residual)
```

```
Optimal x values for minimizing maximum residual: [0.6112953060595889, 0.532
0113604874339, -0.5617560393528208, -0.4670057380135108, 0.1238636652481254
2, 0.1389050995606967, 0.7194104722462624, -1.4502989765651733, 0.3817327707
3835116, -0.17506804363317136, 1.5666205544187461, 0.2510303258046613, 0.041
301305059265846, -0.9918901553185867, -1.2318489820839513, 0.771154752283456
9, 1.3286780508252023, -0.5652960220021633, 0.6352230873909273, -1.599829778
847888, -0.046275270965097974, -0.49409665220190835, 0.1256629963280998, -0.
2592687554933389, 0.11950969207896302, -0.7392541476331865, -0.4633275889777
278, 0.41707013645517094, 0.7416598097725876, -0.04395496934409093, -0.26075
52312044752, 0.2456576539787131, -0.49627980688355916, -0.15552414884819582,
0.8335536131755518, -0.17326147308619783, -0.8909657647783709, -0.3724209837
736477, 1.0824360272200206, 0.5142593118297761, -0.4588761761686262, 0.11186
209093517545, -0.061373388990573895, 0.6660409724208172, -1.072281296571823
1, -0.7976330921856429, -1.329507833987535, -0.8747669007375386, 0.691716266
3422675, -0.3660296723418811, -0.12454761152892212, -0.20097321009716984, -
0.2738757047900683, 0.8752514813737587, -0.27162238465627514, -0.06815920925
280018, 0.25724420337823695, 0.022067000280529193, -0.7371486606464824, 0.29
07301895958046, -0.2884402369973269, 0.9320046573873506, 0.5008160134971658,
-0.3936535837935098, 0.23962501811252762, 0.7789668145891938, -0.79448684083
84802, -0.21026464492489905, 1.3208726875708252, 0.3920898956463524, 0.13947
07207794611, 0.7880057316597716, -0.5343916847078055, 0.09538062478627628, -
1.1028535799219463, -0.7145547552101971, 0.6412767615961248, -1.083976713748
5374, 0.4503616776508077, -0.8397765815115921, -0.9995866898214244, 0.159571
17325770448, -0.720390974798152, -0.04155272740463248, 0.028099446638951395,
0.19772933928068084, 1.0819119519340143, 1.3297797749870117, -0.957614221238
7055, -0.47312842747721573, 0.46414061365632076, -0.5583188859450398, -0.174
57533452664145, 0.4339402928918114, 1.2769052316846146, 0.748767888393257,
0.4732218907686894, -0.809049618661395, 0.9781239508608195, -0.5945345069081
774, 0.3738249636053481, 0.6599539639476797, 0.8709371049081714, 0.271972603
4162431, 1.4983875604035366, 1.2698369989913705, 0.6062270398157369, 0.53245
30261749103, 0.05360480672230623, -0.27479736137886474, 0.5096010409078617,
0.5519992059696778, 0.6747339383450778, -0.20187206359298915, 0.804204134827
5923, 1.3709031147272466, 0.22850477936178926, -0.2192102681808768, -0.40845
132344999246, -0.15194847943675577, -0.21889617962610328, 0.0417431415204060
7, -1.4126762026086552, -0.43271602975619416, 0.79994918208802, -0.284914253
7759399, 1.1559339095568597, 1.0014549069817704, -0.39147438162833065, 0.012
528136139716407, -0.23285363144540167, 0.8739631364596296, -0.09531014798574
328, -0.5715898818950809, 0.618931592488838, 0.387437349275832, -0.497489944
8891933, -0.2413847691844286, -0.3005720233048981, 0.6203100228523064, 0.234
50511330678228, -0.5332879211150715, -0.44395050855737567, -0.87421562943053
3, 0.698861003777729, 0.4183160125288558, -0.33463178780907576, 0.8727526989
352414, 0.8654717741243085, -0.7449441049129654, 0.6322285266619574, -0.1951
44951034703, -1.0833569374131427, 0.31670009508906016, -0.17541242021729236,
0.30579213249351495, -0.058360273996409716, 0.07201476219339341, -0.04547111
4989398445, -0.6088541371124865, -0.12974446193903086, -0.23173535100236461,
0.016697726416904375, -0.023984541665692225, 0.5973300899893765, 0.195182108
46203882, 0.4523599775845635, 0.09725986969840898, 0.18723690464441367, 0.42
397859637654683, 0.006234187575615939, -1.2679203036623372, -0.0995099044092
9151, -0.38453504763190843, 0.22543835651568264, 0.7315258792981705, -0.3959
8420332828704, 0.1259274522843831, 0.7219399412295542, 0.7818129539982908,
0.7410795764577199, -1.5173502643738386, -0.3341747854942647, -1.36976876641
30869, -0.08388441106369642, -0.7155038719951128, 0.2878754179465982, -0.591
94507924476, 0.5462864948775704, -0.6635241437729442, -0.9367264608249174, -
0.96192966372153, -0.4650795547956427, -0.21847021139602066, 0.0103773507794
22858, 0.1993154956249774, -0.3934193915283871, -0.42404947812441074, -1.211
1180583388903, -0.046727569603705076]
```

```
Minimum maximum residual: 46.464663320375024
Coin0506I Presolve 800 (-400) rows, 201 (-400) columns and 150178 (-800) ele
ments
Clp0006I 0  Obj 0 Primal inf 2093.4282 (396)
Clp0006I 43  Obj 2.7139492e-15 Primal inf 21532.228 (357)
Clp0006I 134  Obj 4.9783011e-11 Primal inf 3447491.8 (283)
Clp0006I 225  Obj 52.96895 Primal inf 2438.1526 (179)
Clp0006I 316  Obj 55.057976 Primal inf 547.78218 (101)
Clp0006I 367  Obj 55.250535
Clp0006I 367  Obj 1.2870428e+10 Primal inf 2.9944031e+14 (251) Dual inf 4.99
3002e+16 (133) w.o. free dual inf (81)
Clp0006I 458  Obj 55.234483 Primal inf 177.29406 (48) Dual inf 3.3329992e+15
(99)
Clp0006I 533  Obj 58.512765 Dual inf 37.648109 (93)
Clp0006I 624  Obj 48.835745 Dual inf 13.717196 (80)
Clp0006I 715  Obj 46.739923 Dual inf 2.4666495 (60)
Clp0006I 761  Obj 46.464663
Clp0000I Optimal - objective value 46.464663
Coin0511I After Postsolve, objective 46.464663, infeasibilities - dual 0
(0), primal 0 (0)
Clp0032I Optimal objective 46.46466332 - 761 iterations time 0.452, Presolve
0.09
```

The minimum max-residual that can be achieved is 46.464663320375024

Problem 2 - Curve fitting

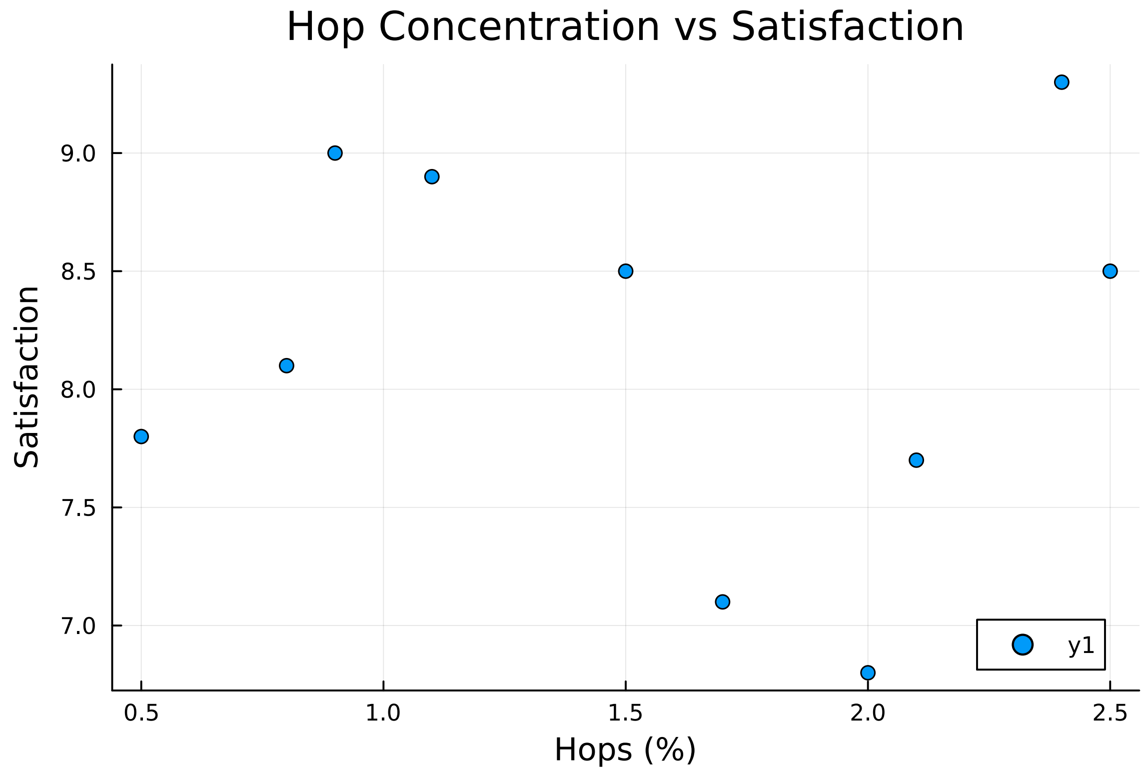
In [9]:
```
using Plots
```

(a)

In [10]:
```
hops = [0.5, 0.8, 0.9, 1.1, 1.5, 1.7, 2.0, 2.1, 2.4, 2.5]
satisfaction = [7.8, 8.1, 9.0, 8.9, 8.5, 7.1, 6.8, 7.7, 9.3, 8.5]

plot(hops, satisfaction, seriestype = :scatter, xlabel = "Hops (%)", ylabel
```

Out[10]:



(b)

In [15]:

```
using Ipopt, Plots, LinearAlgebra

hops = [0.5, 0.8, 0.9, 1.1, 1.5, 1.7, 2.0, 2.1, 2.4, 2.5]
satisfaction = [7.8, 8.1, 9.0, 8.9, 8.5, 7.1, 6.8, 7.7, 9.3, 8.5]

# Define the function to fit
function model_function(x, a, b, c, d)
    return a * x + b * exp(x) + c * sin(x) + d
end

model = Model(Ipopt.Optimizer)

@variable(model, a)
@variable(model, b)
@variable(model, c)
@variable(model, d)

# Objective: minimize the sum of squared errors
@objective(model, Min, sum((satisfaction[i] - model_function(hops[i], a, b,

# Solve the model
optimize!(model)

# Extract the optimized parameters
a_opt = value(a)
b_opt = value(b)
c_opt = value(c)
d_opt = value(d)
```

```
println("Optimized parameters: a = $a_opt, b = $b_opt, c = $c_opt, d = $d_op

# Calculate the 2-norm error
predicted_satisfaction = [model_function(x, a_opt, b_opt, c_opt, d_opt) for
two_norm_error = norm(satisfaction - predicted_satisfaction)
println("2-norm error: " ,two_norm_error)
```

```
This is Ipopt version 3.14.14, running with linear solver MUMPS 5.6.2.

Number of nonzeros in equality constraint Jacobian...:        0
Number of nonzeros in inequality constraint Jacobian.:        0
Number of nonzeros in Lagrangian Hessian.............:       10

Total number of variables............................:        4
                     variables with only lower bounds:        0
                variables with lower and upper bounds:        0
                     variables with only upper bounds:        0
Total number of equality constraints.................:        0
Total number of inequality constraints...............:        0
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr
ls
   0  6.7359000e+02 0.00e+00 1.00e+02  -1.0 0.00e+00    -  0.00e+00 0.00e+00
0
   1  2.6644489e+00 0.00e+00 8.36e-14  -1.0 1.99e+01    -  1.00e+00 1.00e+00
f  1

Number of Iterations....: 1

                                   (scaled)                 (unscaled)
Objective...............:   2.7993106278686641e-01    2.6644489419869615e+00
Dual infeasibility......:   8.3608785909743307e-14    7.9580786405131221e-13
Constraint violation....:   0.0000000000000000e+00    0.0000000000000000e+00
Variable bound violation:   0.0000000000000000e+00    0.0000000000000000e+00
Complementarity.........:   0.0000000000000000e+00    0.0000000000000000e+00
Overall NLP error.......:   8.3608785909743307e-14    7.9580786405131221e-13


Number of objective function evaluations             = 2
Number of objective gradient evaluations             = 2
Number of equality constraint evaluations            = 0
Number of inequality constraint evaluations          = 0
Number of equality constraint Jacobian evaluations   = 0
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations             = 1
Total seconds in IPOPT                               = 0.001

EXIT: Optimal Solution Found.
Optimized parameters: a = -19.92405666822937, b = 3.6982140271403594, c = 1
8.8488012491193, d = 2.678875250509952
2-norm error: 1.6323139838862373
```

(c)

In [28]:
```
# Define the polynomial function to fit
function polynomial_model(params, x)
    a1, a2, a3, a4 = params
    return a1 * x.^3 .+ a2 * x.^2 .+ a3 * x .+ a4
end

# Define the objective function (2-norm error)
function poly_objective(params)
    return sum((satisfaction .- polynomial_model(params, hops)).^2)
end

# Initial guess for parameters
poly_initial_params = [1.0, 1.0, 1.0, 1.0]

# Perform optimization
poly_result = optimize(poly_objective, poly_initial_params, NelderMead())

# Extract the optimal parameters
poly_optimal_params = Optim.minimizer(poly_result)
println("Optimal parameters of a for the polynomial model: ", poly_optimal_p

# Compute the 2-norm error
poly_error = poly_objective(poly_optimal_params)
println("2-norm error for the polynomial model: ", poly_error)
```

```
Optimal parameters of a for the polynomial model: [2.8593624570873386, -12.4
5785813110275, 15.932574891318868, 2.4449346161573584]
2-norm error for the polynomial model: 2.4000170488399393
```

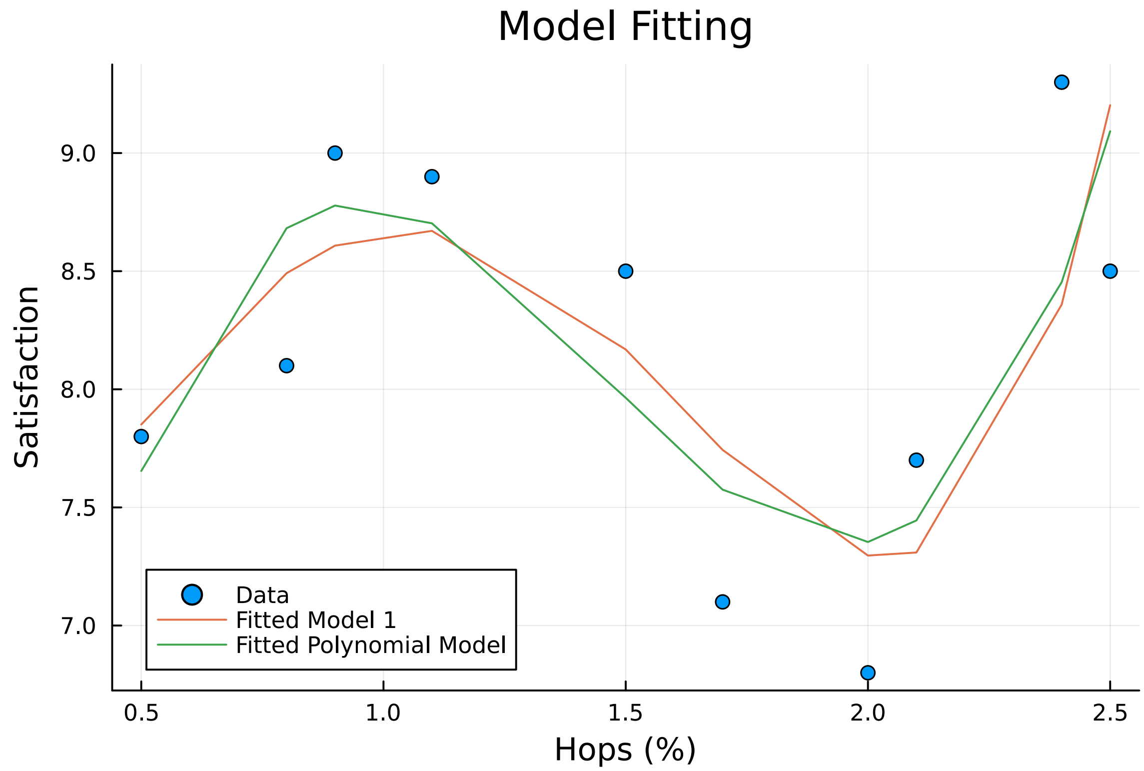
(d)

In [29]:
```
using Plots

# Generate data for the fitted models
fitted_custom_model = custom_model(optimal_params, hops)
fitted_polynomial_model = polynomial_model(poly_optimal_params, hops)

# Plot the data and the fitted models
plot(hops, satisfaction, seriestype = :scatter, label = "Data", xlabel = "Ho
plot!(hops, fitted_custom_model, seriestype = :line, label = "Fitted Model 1
plot!(hops, fitted_polynomial_model, seriestype = :line, label = "Fitted Pol
```

Out[29]:



What value would you expect to get with an input of $x = 0.01$ in each case?

In [30]:
```
poly_optimal_params = poly_optimal_params

# Define the polynomial model function
function polynomial_model(params, x)
    a1, a2, a3, a4 = params
    return a1 * x.^3 .+ a2 * x.^2 .+ a3 * x .+ a4
end

# Evaluate the models at x = 0.01
x_val = 0.01
custom_model_value = custom_model(optimal_params, x_val)
polynomial_model_value = polynomial_model(poly_optimal_params, x_val)

println("Expected value at x = 0.01 for custom model: ", custom_model_value)
println("Expected value at x = 0.01 for polynomial model: ", polynomial_mode
```

```
Expected value at x = 0.01 for custom model: 6.403493492990263
Expected value at x = 0.01 for polynomial model: 2.603017438619894
```

Which model do you believe describes the data better, and why?

The first model seems to capture the overall trend of the data points quite well. It appears to smooth out the variations while still following the general shape of the data. The polynomial model also follows the data points closely but might overfit in some regions, especially where there are more fluctuations in the data points. If the goal is to fit the data as closely as possible, even at the risk of overfitting, the polynomial model might be better.

Problem 3 - Lots of Norms

(a)

Linear Programming Formulation

Objective: $\min e_1 + e_2 + e_3$

Constraints:

1. $e_1 \geq 2I + 2P - 5$
2. $e_1 \geq -(2I + 2P - 5)$
3. $e_2 \geq I + 3P - 6$
4. $e_2 \geq -(I + 3P - 6)$
5. $e_3 \geq 3I - P - 4$
6. $e_3 \geq -(3I - P - 4)$
7. $e_1, e_2, e_3 \geq 0$

Decision Variables:

- I
- P
- e_1, e_2, e_3

In [31]:
```
using JuMP
using GLPK

# Create a new model with the GLPK optimizer
model = Model(GLPK.Optimizer)

# Define variables
@variable(model, I)
@variable(model, P)
@variable(model, r[1:3] >= 0)

# Define constraints
@constraint(model, r[1] >= 2*I + 2*P - 5)
@constraint(model, r[1] >= -2*I - 2*P + 5)
@constraint(model, r[2] >= I + 3*P - 6)
@constraint(model, r[2] >= -I - 3*P + 6)
@constraint(model, r[3] >= 3*I - P - 4)
@constraint(model, r[3] >= -3*I + P + 4)

# Define objective to minimize the total absolute deviation
@objective(model, Min, sum(r))

# Optimize the model
optimize!(model)
```

```
# Get results
I_opt = value(I)
P_opt = value(P)
total_absolute_deviation = objective_value(model)

# Display the results
println("Optimal I: ", I_opt)
println("Optimal P: ", P_opt)
println("Total absolute deviation (1-norm): ", total_absolute_deviation)
```

```
Optimal I: 1.8
Optimal P: 1.4
Total absolute deviation (1-norm): 1.3999999999999995
```

(b)

Nonlinear Optimization Model

Objective: $\min \sqrt{(2I + 2P - 5)^2 + (I + 3P - 6)^2 + (3I - P - 4)^2}$

Subject to: $I + P = 3$

Decision Variables:

- I
- P

In [32]:
```
using JuMP
using Ipopt

# Create a new model with the Ipopt optimizer
model2 = Model(Ipopt.Optimizer)

# Define variables
@variable(model2, I)
@variable(model2, P)

# Define objective to minimize the 2-norm of the residuals
@objective(model2, Min, (2*I + 2*P - 5)^2 + (I + 3*P - 6)^2 + (3*I - P - 4)^

# Add final constraint
@constraint(model2, I + P == 3)

# Optimize the model
optimize!(model2)

# Get results
I_opt2 = value(I)
P_opt2 = value(P)
total_2norm = objective_value(model2)

# Display the results
```

```
println("Optimal I: ", I_opt2)
println("Optimal P: ", P_opt2)
println("Total 2-norm of the residuals: ", sqrt(total_2norm))
```

```
This is Ipopt version 3.14.14, running with linear solver MUMPS 5.6.2.

Number of nonzeros in equality constraint Jacobian...:        2
Number of nonzeros in inequality constraint Jacobian.:        0
Number of nonzeros in Lagrangian Hessian.............:        3

Total number of variables............................:        2
                     variables with only lower bounds:        0
                variables with lower and upper bounds:        0
                     variables with only upper bounds:        0
Total number of equality constraints.................:        1
Total number of inequality constraints...............:        0
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr
ls
   0  7.7000000e+01 3.00e+00 4.00e+00  -1.0 0.00e+00    -  0.00e+00 0.00e+00
0
   1  1.2000000e+00 0.00e+00 3.55e-15  -1.0 1.70e+00    -  1.00e+00 1.00e+00
f  1

Number of Iterations....: 1

                                   (scaled)                 (unscaled)
Objective...............:   1.1999999999999993e+00    1.1999999999999993e+00
Dual infeasibility......:   3.5527136788005009e-15    3.5527136788005009e-15
Constraint violation....:   0.0000000000000000e+00    0.0000000000000000e+00
Variable bound violation:   0.0000000000000000e+00    0.0000000000000000e+00
Complementarity.........:   0.0000000000000000e+00    0.0000000000000000e+00
Overall NLP error.......:   3.5527136788005009e-15    3.5527136788005009e-15


Number of objective function evaluations             = 2
Number of objective gradient evaluations             = 2
Number of equality constraint evaluations            = 2
Number of inequality constraint evaluations          = 0
Number of equality constraint Jacobian evaluations   = 1
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations             = 1
Total seconds in IPOPT                               = 0.002

EXIT: Optimal Solution Found.
Optimal I: 1.6999999999999997
Optimal P: 1.3000000000000003
Total 2-norm of the residuals: 1.095445115010332
```

(c)

System of Linear Equations

Given the original system of equations:

1. $2I + 2P = 5$
2. $I + 3P = 6$
3. $3I - P = 4$

And the additional constraint: 4. $I + P = 3$

We can write this system as:

$$\begin{cases} 2I + 2P = 5 \\ I + 3P = 6 \\ 3I - P = 4 \\ I + P = 3 \end{cases}$$

In [34]:
```
using JuMP
using Ipopt

# Create a new model with the Ipopt optimizer
model3 = Model(Ipopt.Optimizer)

# Define variables
@variable(model3, I)
@variable(model3, P)
@variable(model3, r1)
@variable(model3, r2)
@variable(model3, r3)

# Define constraints for residuals
@constraint(model3, r1 == 2*I + 2*P - 5)
@constraint(model3, r2 == I + 3*P - 6)
@constraint(model3, r3 == 3*I - P - 4)

# Add the final constraint
@constraint(model3, I + P == 3)

# Define objective to minimize the 2-norm of the residuals
@objective(model3, Min, r1^2 + r2^2 + r3^2)

# Optimize the model
optimize!(model3)

# Get results
I_opt3 = value(I)
P_opt3 = value(P)
total_2norm_linear = sqrt(value(r1^2 + r2^2 + r3^2))

# Display the results
println("Optimal I: ", I_opt3)
```

```
println("Optimal P: ", P_opt3)
println("Total 2-norm of the residuals: ", total_2norm_linear)
```

```
This is Ipopt version 3.14.14, running with linear solver MUMPS 5.6.2.

Number of nonzeros in equality constraint Jacobian...:       11
Number of nonzeros in inequality constraint Jacobian.:        0
Number of nonzeros in Lagrangian Hessian.............:        3

Total number of variables............................:        5
                     variables with only lower bounds:        0
                variables with lower and upper bounds:        0
                     variables with only upper bounds:        0
Total number of equality constraints.................:        4
Total number of inequality constraints...............:        0
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:        0
        inequality constraints with only upper bounds:        0

iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr
ls
   0  0.0000000e+00 6.00e+00 0.00e+00  -1.0 0.00e+00    -  0.00e+00 0.00e+00
0
   1  1.2000000e+00 0.00e+00 0.00e+00  -1.0 1.70e+00    -  1.00e+00 1.00e+00
h  1

Number of Iterations....: 1

                                   (scaled)                 (unscaled)
Objective...............:   1.1999999999999995e+00    1.1999999999999995e+00
Dual infeasibility......:   0.0000000000000000e+00    0.0000000000000000e+00
Constraint violation....:   0.0000000000000000e+00    0.0000000000000000e+00
Variable bound violation:   0.0000000000000000e+00    0.0000000000000000e+00
Complementarity.........:   0.0000000000000000e+00    0.0000000000000000e+00
Overall NLP error.......:   0.0000000000000000e+00    0.0000000000000000e+00


Number of objective function evaluations             = 2
Number of objective gradient evaluations             = 2
Number of equality constraint evaluations            = 2
Number of inequality constraint evaluations          = 0
Number of equality constraint Jacobian evaluations   = 1
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations             = 1
Total seconds in IPOPT                               = 0.005

EXIT: Optimal Solution Found.
Optimal I: 1.7
Optimal P: 1.2999999999999998
Total 2-norm of the residuals: 1.095445115010332
```

In [ ]: