Project 3: Engineering Design Optimization

Robert Moss
Stanford University, Stanford, CA 94305
MOSSR@CS.STANFORD.EDU
AA222/CS361

1 *Question 1, Problem 1*

What is the cheapest combination of ingredients and filler to produce one kg of product?

Task 1. We can formulate the optimization problem as follows. Let $\mathbf{c} = [60, 40]$ to represent the cost per kilogram of each ingredient (omitting the filler which we'll addressed later). Let $\mathbf{x} = [x_1, x_2]$ be the design variables where x_1 is the portion of ingredient 1 and x_2 is the portion of ingredient 2, both in kilograms. Let the matrix

$$\mathbf{A} = \begin{bmatrix} 200 & 100 \\ 130 & 70 \\ 15 & 35 \\ 0 & 7 \\ 655 & 788 \end{bmatrix}$$

which captures the amount of vitamins and minerals in each ingredient in grams per kilogram, and accounts for the ''other'' content in the ingredient (i.e. content aside from the given vitamins and minerals to sum to 1 kg). Let $\mathbf{b} = [90, 50, 20, 2, 0]$ to capture the minimum vitamin and mineral requirements, in grams. Finally, let s be a slack variable that represents the amount of the product in grams taken up by the filler. Formally, we get the following optimization problem:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{b} \\ & \mathbf{A}\mathbf{x} \cdot \mathbf{1} + s = 1000 \\ & \mathbf{x} \geq \mathbf{0} \\ & s \geq 0 \end{aligned}$$

The constraint $\mathbf{A}\mathbf{x} \geq \mathbf{b}$ ensures we meet the minimum vitamin and mineral requirements encoded in $\mathbf{b}$. The constraint $\mathbf{A}\mathbf{x} \cdot \mathbf{1} + s = 1000$ ensures that the sum of the grams of ingredients and the grams of filler equals 1000 grams (i.e. 1 kg, where $\mathbf{1}$ is the ones-vector used for convenient summations). Finally, the constraints $\mathbf{x} \geq \mathbf{0}$ and $s \geq 0$ ensure that there is a non-negative amount of the ingredients and filler, respectively.

Task 2. To solve this optimization problem, we used `JuMP.jl` and wrote the following code:

```
using JuMP
using GLPK
using LinearAlgebra

model = Model(GLPK.Optimizer)

c = [60, 40] # cost/kg

A = [200  100;
     130   70;
      15   35;
       0    7;
     655  788] # grams/kg

b = [90, 50, 20, 2, 0] # grams

N, M = length(c), length(b)
@variable(model, x[1:N]) # ingredients
@variable(model, s) # filler/slack

@objective(model, Min, c'x)

@constraints(model, begin
    A*x .≥ b
    A*x·ones(M) + s == 1000
    x .≥ 0
    s ≥ 0
end)

optimize!(model)
```

We get the optimal solution of $\mathbf{x} \approx [0.2091, 0.4818]$ kg with a filler/slack of $s \approx 309.09$ grams and the minimized value of the cost function $\mathbf{c}^\top \mathbf{x} \approx 31.818$.

Task 3. From the results in task (2), we get that the ideal recipe is about 0.2091 kg of ingredient 1, about 0.4818 kg of ingredient 2, and the rest is filler (about 0.3091 kg). This has an associated manufacturing cost of about 31.818 with a resulting nutritional composition of about 90 grams of vitamin A, 60.91 grams of vitamin B, 20 grams of vitamin C, 3.373 grams of minerals, 516.63 grams of the ''other'' contents, and 309.09 grams of filler (for a total of 1 kg of product).

2 *Question 1, Problem 2*

What is the cheapest combination of ingredients and filler to produce one kg of product with the given requirements?

Task 4. Similar to task (1), we can formulate the optimization problem as follows. Let $\mathbf{c} = [60, 40, 25]$ to represent the cost per kilogram of each ingredient and the one-time-additive cost of using ingredient 2. Let $\mathbf{x} = [x_1, x_2, x_b]$ be the mixed design variables where x_1 is the portion of ingredient 1 in kg, x_2 is the portion of ingredient 2 in kg, and x_b is a binary variable used to control the additive cost of 25. Let the matrix

$$\mathbf{A} = \begin{bmatrix} 200 & 100 & 0 \\ 130 & 70 & 0 \\ 15 & 35 & 0 \\ 0 & 7 & 0 \\ 655 & 788 & 0 \end{bmatrix}$$

which captures the amount of vitamins and minerals in each ingredient in grams per kilogram, accounting for the additive cost (i.e. the all-zeros column) and the ''other'' content in the ingredient. Let $\mathbf{b} = [90, 50, 20, 2, 0]$ to capture the minimum vitamin and minerals requirements in grams. Let s be a slack variable that represents the amount of the product in grams taken up by the filler. Let the binary variable $\mathbf{y} \in \mathbb{B}^M$ where $M = |\mathbf{b}| = 4$ which is used to control the ''at least 2'' satisfactory relaxation. Let $\mathbf{y}' = \mathbf{y}_{1:M-1}$ be the subset of the indicator variables that exclude the ''other'' content of the ingredients. Let $u = 1$ which is the upper bound on the value of x_2 (for ingredient 2). Formally, we get the optimization problem:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{b} \odot \mathbf{y} \\ & \mathbf{y}' \cdot \mathbf{1} \geq 2 \\ & \mathbf{A}\mathbf{x} \cdot \mathbf{1} + s = 1000 \\ & x_b - \frac{x_2}{u} \geq 0,\ \mathbf{x} \geq \mathbf{0},\ s \geq 0 \end{aligned}$$

Where $\odot$ is used as the component-wise vector multiplication. The constraint $\mathbf{A}\mathbf{x} \geq \mathbf{b} \odot \mathbf{y}$ ensures that *when* a particular ingredient/mineral requirement is being met (as indicated by $\mathbf{y}$) then we want to make sure that the requirements are actually met as encoded by $\mathbf{b}$. The constraint $\mathbf{y}' \cdot \mathbf{1} \geq 2$ ensures that *at least* 2 of the requirements are met (where $\mathbf{1}$ is the ones-vector, used for convenient summations). The constraint $\mathbf{A}\mathbf{x} \cdot \mathbf{1} + s = 1000$ ensures that the sum of the grams of ingredients and the grams of filler equals 1000 grams (i.e. 1 kg). The constraint $x_b - \frac{x_2}{u} \geq 0$, as taken from the *hint*, ensures that when the value of $x_2 > 0$, then $x_b = 1$, and otherwise x_b can be either 1 or 0 (either way, it won't matter). Finally, the constraints $\mathbf{x} \geq \mathbf{0}$ and $s \geq 0$ ensure that there is a non-negative amount of the ingredients and filler, respectively.

Task 5. To solve this optimization problem, we used `JuMP.jl` and wrote the following code:

```
model = Model(GLPK.Optimizer)
c = [60, 40, 25]       # cost/kg
A = [200  100   0;
     130   70   0;
      15   35   0;
       0    7   0;
     655  788   0]     # grams/kg
b = [90, 50, 20, 2, 0] # grams

N, M = length(c), length(b)
@variable(model, x[i=1:N], binary=(i == N))
@variable(model, y[1:M], Bin) # relaxation indicator
@variable(model, s)           # filler/slack
y′ = y[1:end-1] # subset, excludes "other" content
u = 1           # upper bound on x₂ (ingredient 2)

@objective(model, Min, c'x)
@constraints(model, begin
    A*x .≥ b .* y
    y′·ones(M-1) ≥ 2
    A*x·ones(M) + s == 1000
    x[N] - x[2]/u .≥ 0
    x .≥ 0
    s ≥ 0
end)
optimize!(model)
```

We get the optimal solution of $\mathbf{x} = [0.45, 0.0, 0.0]$ kg with a filler/slack of $s = 550.0$ grams, the relaxation indicator of $\mathbf{y} = [1, 1, 0, 0]$, and the minimized value of the cost function $\mathbf{c}^\top \mathbf{x} = 27.0$.

Task 6. The ideal recipe has 0.45 kg of ingredient 1, 0 kg of ingredient 2, and 0.55 kg of filler. This has an associated manufacturing cost of 27 with a resulting nutritional composition of 90 grams of vitamin A, 58.5 grams of vitamin B, 6.75 grams of vitamin C, 0 grams of minerals, 294.75 grams of the ''other'' contents, and 550.0 grams of filler (for a total of 1 kg of product). Only the first two nutritional requirements were satisfied (where $\mathbf{y} = [1, 1, 0, 0]$), resulting in a relaxation of the vitamin C and minerals requirements.

The interpretation of the results is that given the relaxation of only needing to satisfy *at least* 2 requirements, the optimal solution omits the use of ingredient 2. This is in part due to the additional cost of 25 (on top of the second ingredient's original cost). Another factor is since only ingredient 2 has non-zero minerals our original formulation *always* required some amount of ingredient 2 to satisfy the constraints. But now with the relaxation, the minerals constraint is no longer required to be met resulting in ingredient 2 not being in the optimal product.

3 *Question 2, Problem 1*

Use expression optimization to solve for the two expressions $x(t)$ and $y(t)$ that describe the motion of stars in a trinary star system.

Task 1. We use an objective function that is the sum of the squared residual between the true function $x(t)$ and the approximate function $\hat{x}(t)$. A regularization term and multiplicative penalty term are included to provide stability in the final expression. For regularization, we want to limit the number of operations to encourage simpler expressions. The multiplicative penalty term ρ uses the approximate derivative to penalize expressions with opposite slope. Let x_t represent a shorthand for $x(t)$. Formally, we get the objective function:

$$\left[\sum_{t=t_0}^{t_{\max}} (x_t - \hat{x}_t)^2 \, \rho^{\mathbb{1}[\text{sign}(\nabla x_t) \neq \text{sign}(\nabla \hat{x}_t)]}\right] + \frac{\lambda}{2}|\hat{x}|^2$$

We use a regularization constant of $\lambda = 0.05$ and we let $|\hat{x}|$ be the number of operations in an expression $\hat{x}$. We use $\mathbb{1}$ as the indicator function, which applies the penalty $\rho = 10$ when the derivatives are opposite signs for ∇x_t and $\nabla \hat{x}_t$ at time t.

Task 2. We use a grammar that incorporates mathematical operations of $+, -, \times, /$ with trigonometric functions sin and cos, the exponentiation function exp, the square root, the time input variable t, the constant π, and digits $1 - 9$. Formally, we get the grammar:

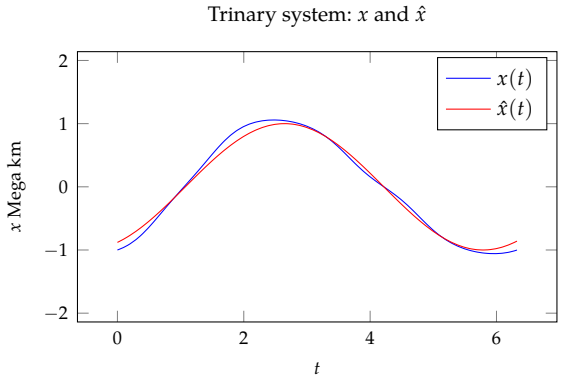
$$\begin{aligned}
&\mathbb{R} \to \mathbb{R} + \mathbb{R} \mid \mathbb{R} - \mathbb{R} \mid \mathbb{R} \times \mathbb{R} \mid \mathbb{R} \,/\, \mathbb{R} \\
&\mathbb{R} \to \sin(\mathbb{R}) \mid \cos(\mathbb{R}) \mid \exp(\mathbb{R}) \mid \text{sqrt}(\mathbb{R}) \\
&\mathbb{R} \to t \mid \pi \\
&\mathbb{R} \to 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}$$

Task 3. We could use population methods due to the nature of expression optimization. Particularly, we used *genetic programming* to evolute a population of expressions. Genetic programming is well suited for expression optimization because we sample from a discrete distribution of expressions generated from the grammar and treat each expression as an individual in the population. We set identical hyperparameters when optimizing for $\hat{x}$ and $\hat{y}$. The hyperparameters are shown in the code below.

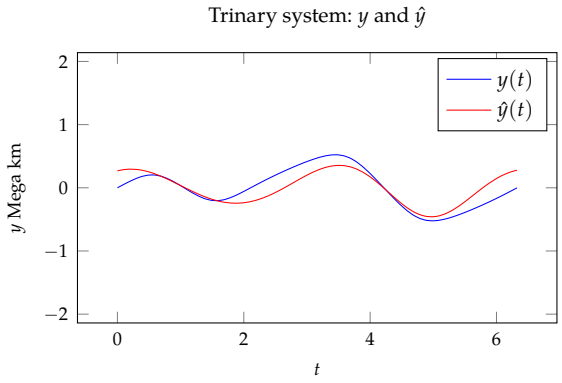
Task 4. We used the `ExprOptimization.jl` package and wrote the following code:

```
using ExprOptimization, LinearAlgebra, Random, DelimitedFiles
Random.seed!(0)

import Base.length
length(sym::Symbol) = 1 # counting operations
length(ex::Expr) = sum(length(a) for a in ex.args)
R(expr, λ=0.05) = λ/2*length(expr)^2 # regularization

𝕀(b) = b ? 1 : 0 # indicator function
penalty(∇v, ∇v′, ρ=10) = ρ^𝕀(sign(∇v) ≠ sign(∇v′))
v′(t, S, ex) = Core.eval(merge(S, Dict(:t=>t)), ex)
∇(V) = vcat(0, diff(V)) # derivative

const traj = readdlm("traj.txt", ',')
const T, X, Y = traj[:,1], traj[:,2], traj[:,3]

const grammar = @grammar begin
    ℝ = (ℝ + ℝ) | (ℝ - ℝ) | (ℝ * ℝ) | (ℝ / ℝ)
    ℝ = sin(ℝ) | cos(ℝ) | exp(ℝ) | sqrt(ℝ)
    ℝ = t | π
    ℝ = |(1:9)
end
const S = SymbolTable(grammar)

function objective(V, ∇V, tree, grammar)
    ex = get_executable(tree, grammar)
    V′ = try [v′(t, S, ex) for t in T] catch; return Inf end
    ∇V′ = ∇(V′)
    return sum((vₜ - vₜ′)^2 * penalty(∇vₜ, ∇vₜ′) for (vₜ, vₜ′, ∇vₜ, ∇vₜ′) in zip(V, V′, ∇V, ∇V′)) + R(ex)
end
optimize(GeneticProgram(3000, 100, 6, 0.3, 0.3, 0.4), grammar, :ℝ, (t,g)->objective(X,∇(X),t,g))
optimize(GeneticProgram(3000, 100, 6, 0.3, 0.3, 0.4), grammar, :ℝ, (t,g)->objective(Y,∇(Y),t,g))
```

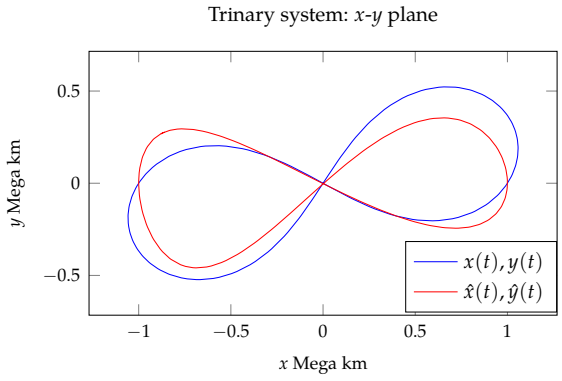
Task 5 (a). Below, we plot the ground truth data $x(t)$ compared to the approximated expression $\hat{x}(t)$.



Next, we plot the ground truth data $y(t)$ compared to the approximated expression $\hat{y}(t)$.



Finally, we plot the ground truth data $(x(t), y(t))$ compared to the approximated expression $(\hat{x}(t), \hat{y}(t))$.



Task 5 (b). To show that our $\hat{x}(t)$ is quantitatively better than simply predicting the mean of the $x(t)$ trajectory, notice that we can use our objective function to calculate the loss if we replace $\hat{x}(t)$ with the mean $\bar{x}(t) \approx -0.0267$ instead. If we use $\bar{x}(t)$ we get a loss of about 751.41 compared to a loss of about 1.69 if we use our approximated $\hat{x}(t)$. If we use $\bar{y}(t) \approx 8 \times 10^{-6}$ we get a loss of about 115.59 compared to a loss of about 8.04 if we use our approximated $\hat{y}(t)$.

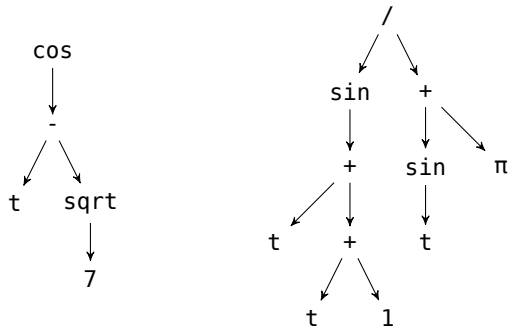
Task 5 (c). Running the code finds the best fit expressions:

```
x(t) = cos(t - sqrt(7))
y(t) = sin(t + (t + 1)) / (sin(t) + π)
```

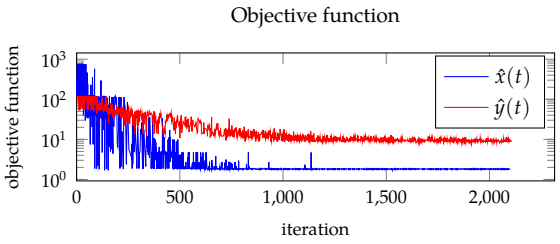
These correspond to the approximate closed form equations for $x(t)$ and $y(t)$:

$$\hat{x}(t) = \cos(t - \sqrt{7})$$
$$\hat{y}(t) = \frac{\sin(t + (t+1))}{\sin(t) + \pi} = \frac{\sin(2t+1)}{\sin(t) + \pi} \qquad \text{(simplified)}$$

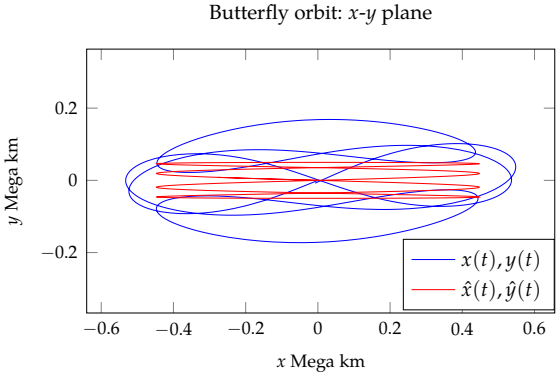
The grammar trees for $\hat{x}$ and $\hat{y}$ are, respectively:



Task 5 (d). Below, we plot the objective function in log-scale as a function of the number of iterations.



Task 6. The expressions $\hat{x}(t) = \sin((7 - (8+3) * t)/\sqrt{5}$ and $\hat{y}(t) = \sin(t)/\exp(3)$ were generated for the following butterfly orbit:



4 *Rhyme*

You may use my name. What type of poem is optimal?

Optimization Haiku.

minimize, choose x.
subject to constraints, complex.
optimality, next.