CVXPY x NASA Course 2024

Philipp Schiele   Steven Diamond   Parth Nobel   Akshay Agrawal

June 24, 2024

Homework review

Solutions:

1. Portfolio optimization: `https://marimo.app/l/z7n5vh`
2. Diet problem: `https://marimo.app/l/8gi4dr`
3. Minimum fuel problem: `https://marimo.app/l/c5zt9u`

Controlling Self-landing Rockets

Outline

Background

Formulating the problem

Model predictive control

Background on this lecture

- SpaceX has used CVXGEN, a code generator for convex problems, as part of their control system for landing Falcon 9 rockets
- this lecture is based on work by Thomas Lipp, Lars Blackmore, and Yoshi Kuwata
- a simplified version of the problem was added as an exercise to Convex Optimization
- we are not involved in landing actual rockets
- we are not affiliated with SpaceX

Outline

Background

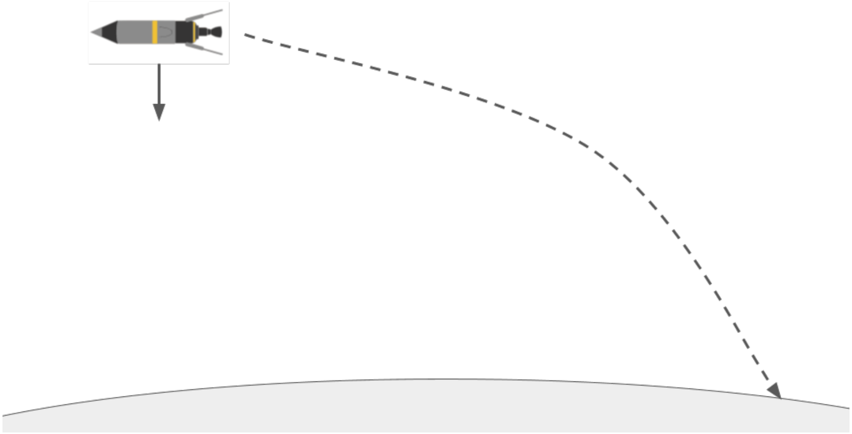
Formulating the problem

Model predictive control

Moving through space

- an object in motion remains in motion at constant speed and in a straight line unless acted on by an unbalanced force

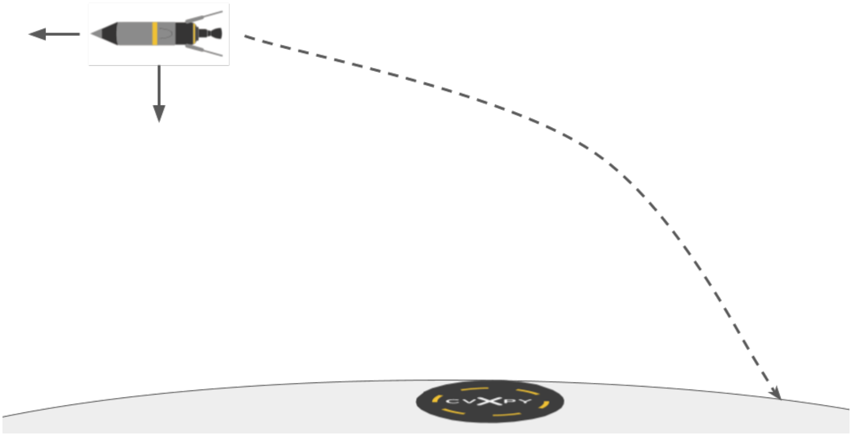
Gravity



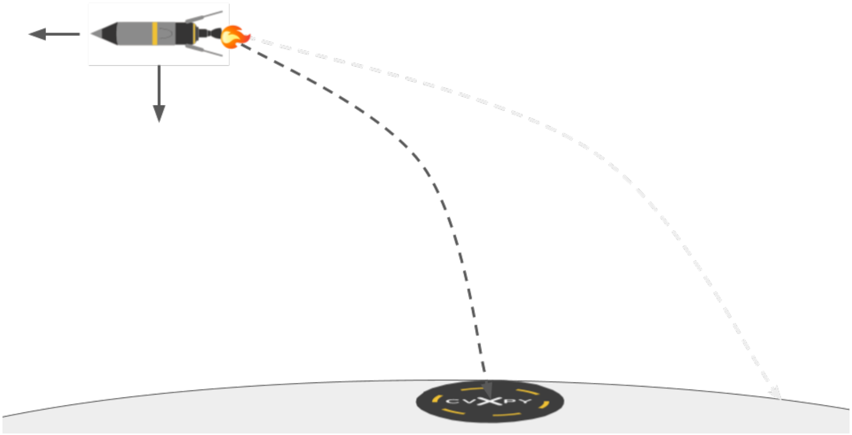
Gravity

- on earth, gravity accelerates objects towards its center at about $9.8\,\mathrm{m\,s^{-2}}$
- even at the height of the ISS, $400\,\mathrm{km}$ above the surface, gravity is still about $89\,\%$ as strong as on the ground
- for the landing problem, we assume that gravity is constant
- absent other forces, the rocket falls down in a parabola (depending on initial velocity)

Targeting the landing pad



Targeting the landing pad



Gravity

- we apply a force by firing the rocket’s engines
- can choose direction and magnitude of the force
- want to find a sequence of forces that brings the rocket to the landing pad

Formalizing the problem

- spacecraft dynamics:

$$m\ddot{p} = f - mge_3$$

  with $p(t) \in \mathbf{R}^3$ position, $f(t) \in \mathbf{R}^3$ thrust,
  m mass, g gravity
- we require $p(T) = 0$ and $\dot{p}(T) = 0$
- the initial position $p(0)$ and velocity $\dot{p}(0)$ are given
- upper bound on the thrust: $\|f(t)\|_2 \leq f_{\max}$

Discretization

- approximate the continuous-time dynamics by a discrete-time system
- we discretize time into N intervals of length h
- use p_k and f_k to denote $p(kh)$ and $f(kh)$
- apply constant force f_k during interval k
- velocity changes according to the force applied

$$v_{k+1} = v_k + \frac{h}{m} f_k - hge_3,$$

- position changes according to the average velocity

$$p_{k+1} = p_k + \frac{h}{2}(v_{k+1} + v_k)$$

Objective function

- want to minimize the total fuel used
- fuel used is proportional to the magnitude of the thrust, *i.e.*,

$$\sum_{k=1}^{N} \gamma \|f_k\|_2$$

  where γ is the factor of proportionality
- other objectives like minimum time descent are also possible

Specifying the problem in CVXPY

Problem specification

`https://marimo.app/l/emz5ss`

Specifying the problem in CVXPY

Solution: Problem specification

`https://marimo.app/l/os00bu`

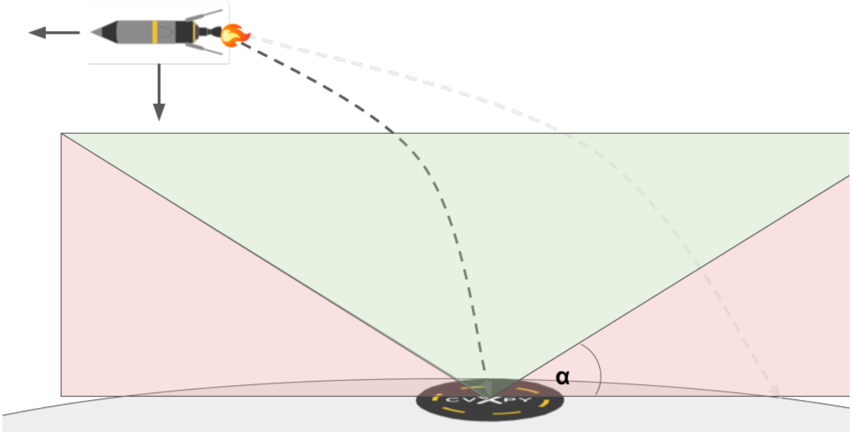
Specifying the problem in CVXPY

```
V = cp.Variable((K + 1, 3))  # velocity
P = cp.Variable((K + 1, 3))  # position
F = cp.Variable((K, 3))  # thrust

constraints = [
    ...
]

fuel_consumption = gamma * cp.sum(cp.norm(F, 2, axis=1))
objective = cp.Minimize(fuel_consumption)

problem = cp.Problem(objective, constraints)
problem.solve()
```

Specifying the constraints in CVXPY

```
constraints = [
    P[0] == p0,
    V[0] == v0,

    V[1:, :2] == V[:-1, :2] + (h / m) * F[:, :2],
    V[1:, 2] == V[:-1, 2] + (h / m) * F[:, 2] - (h * g),  # gravity

    P[1:] == P[:-1] + (h / 2) * (V[:-1] + V[1:]),

    cp.linalg.norm(F, 2, axis=1) <= Fmax,

    P[K] == p_target,
    V[K] == [0, 0, 0]
]
```

Glide-slope constraint



Glide-slope constraint

- the rocket should not leave the glide-slope cone
- the glide-slope cone is parametrized by the angle α
- requires that
  $$p_k^T e_3 \geq \tan\alpha \|p_k^T e_1, p_k^T e_2\|_2$$
  for all k
- add the constraint to the optimization problem

Outline

Background

Formulating the problem

Model predictive control

The mission

- simulate landing in the Kerbal Space Program
- use CVXPY to control rocket autonomously
- land the rocket back on the launch pad
- . . . ideally in one piece

- Others have done this before
- *e.g.*, `https://github.com/jonnyhyman/G-FOLD-Python`

Model predictive control (MPC)

- Core idea: repeatedly solve the optimization problem
- At each time step, policy is the first step of the solution
- Even simplified models can lead to good results

Algorithm 1 MPC Loop

Require: $T^{\max} > 0$

while true **do**

  $p_t, v_t \leftarrow$ update state

  If $p_t = p^{\text{target}}$; **break**

  solve optimization problem $P_{p_t, v_t, T^{\max}}$

  perform first step of optimal policy

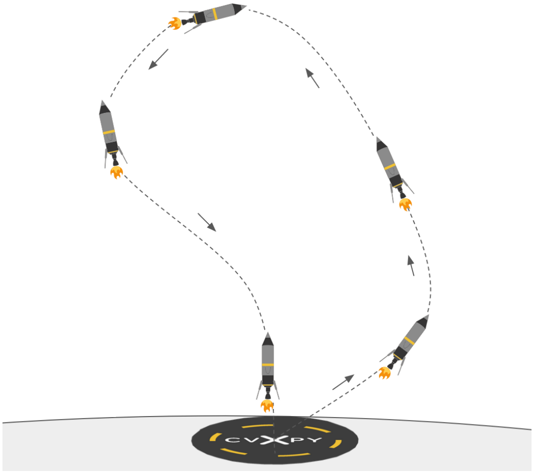
end while

---

kRPC

- mod for KSP
- allows to control the game programmatically
- provides telemetry data
- . . . and a lot more
- `https://krpc.github.io/krpc/`

Test flight!



Making the optimization problem robust

- We do not know the exact landing time T
- Current model does not account for early landing
- **Solution**: add a penalty to the height

- Model is agnostic to approach angle
- **Solution**: add penalty to x and y deviations

- Hard constraints are not robust
- Want to prevent infeasibility
- **Solution**: add soft constraints