exercise9

November 3, 2025

1 Exercise 9.1: Direct Method

We solve the Moon-lander problem:

$$\min_{u(\cdot),\,t_f} J(u) = \int_0^{t_f} u(s)\,ds \quad \text{s.t.} \dot{X}_t = \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ -g+u \end{bmatrix},\; X_0 = \begin{bmatrix} 10 \\ -2 \end{bmatrix},\; X_{t_f} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, 0 \le u \le 3,\; g = 1.5,\; t_f > 0 \text{ free.}$$

[10]:
```
import numpy as np
import matplotlib.pyplot as plt
from moon_lander import LanderParams, LanderIC, DirectShootingSolver, HSSolver

g, umax, N = 1.5, 3.0, 60
x0, v0 = 10.0, -2.0
p = LanderParams(g=g, umax=umax, N=N)
ic = LanderIC(x0=x0, v0=v0)
```

1.1 1. Direct single-shooting

We parameterise u as a piecewise-constant function on $N = 60$ segments, with t_f as a decision variable. We simulate the ODE forward from the given initial state using constant-acceleration (Euler) integration, enforcing $X_{t_f} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ as terminal equality constraints. The objective is computed using the rectangle rule:

$$J \approx \frac{t_f}{N} \sum_{i=0}^{N-1} u_i, \qquad \Delta t = \frac{t_f}{N}.$$

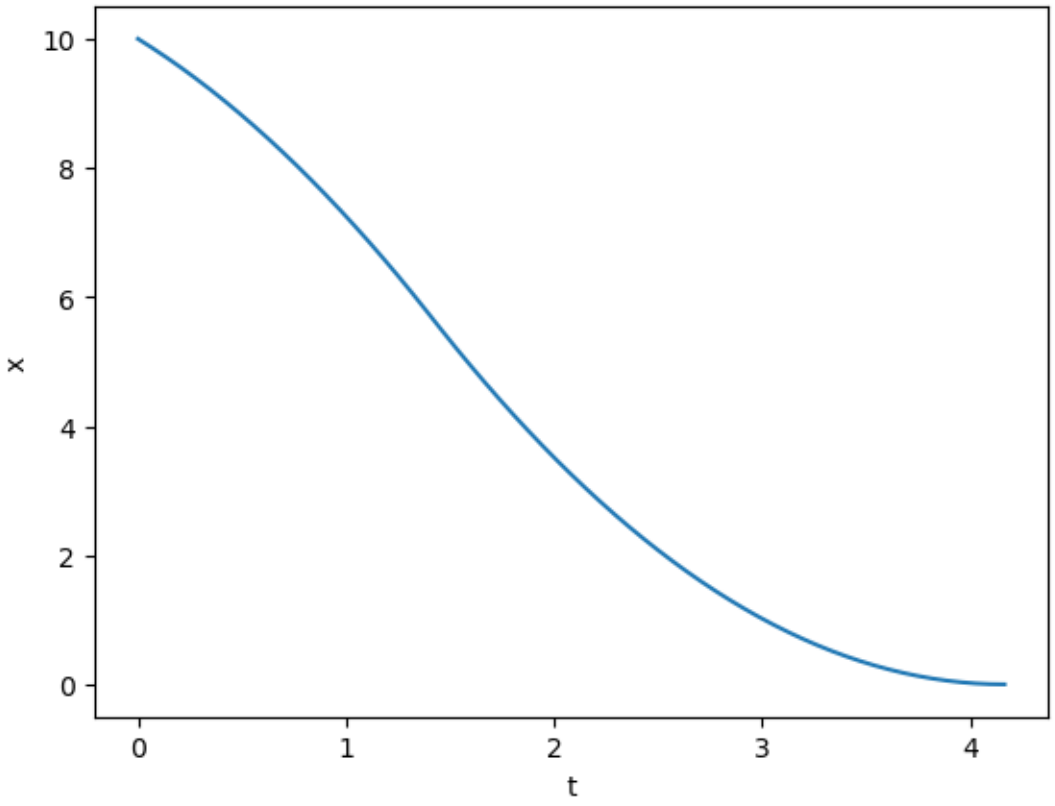
The optimisation uses SLSQP with `maxiter` $= 800$ and `ftol` $= 10^{-9}$.

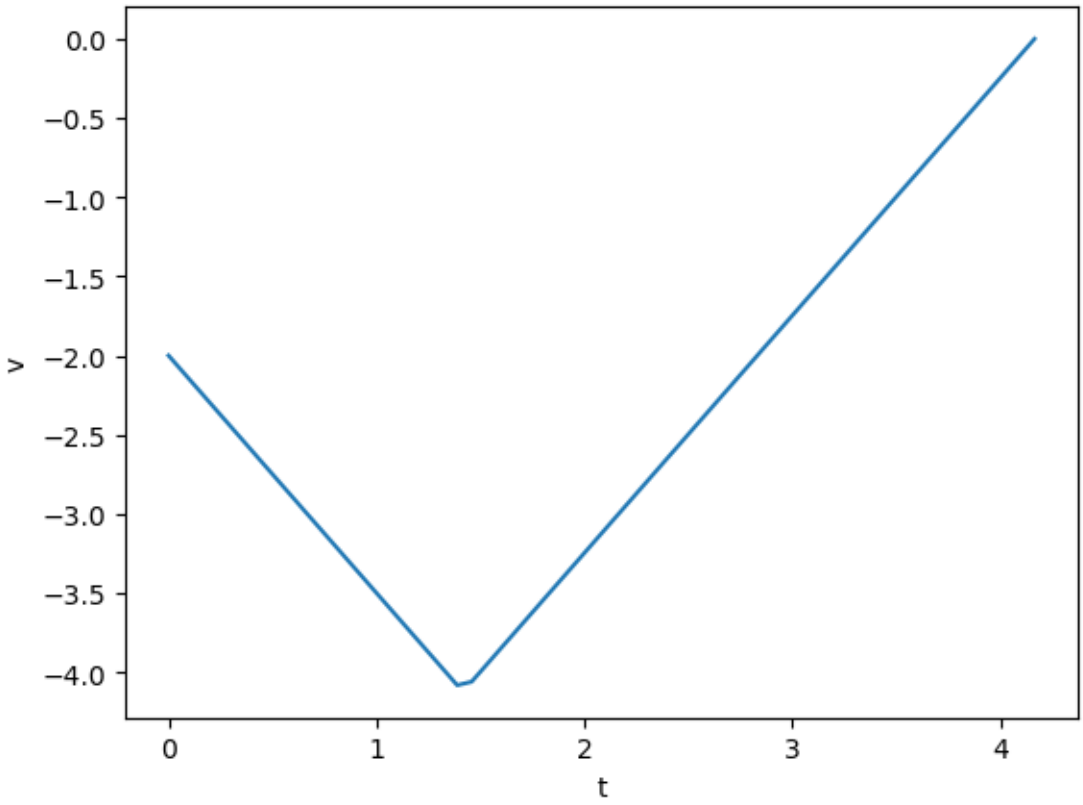
[11]:
```
shoot = DirectShootingSolver(p, ic).solve(maxiter=800)
print("Shooting:", shoot.success, shoot.message, shoot.info)
print(f"tf* = {shoot.tf:.6f}, J* = {shoot.J:.6f}")
print(f"x(tf)={shoot.x[-1]:.3e}, v(tf)={shoot.v[-1]:.3e}")

plt.figure(); plt.plot(shoot.t, shoot.x); plt.xlabel("t"); plt.ylabel("x")
plt.figure(); plt.plot(shoot.t, shoot.v); plt.xlabel("t"); plt.ylabel("v")

t_edges = np.linspace(0.0, shoot.tf, len(shoot.u)+1)
t_steps = np.repeat(t_edges, 2)[1:-1]
u_steps = np.repeat(shoot.u, 2)
```

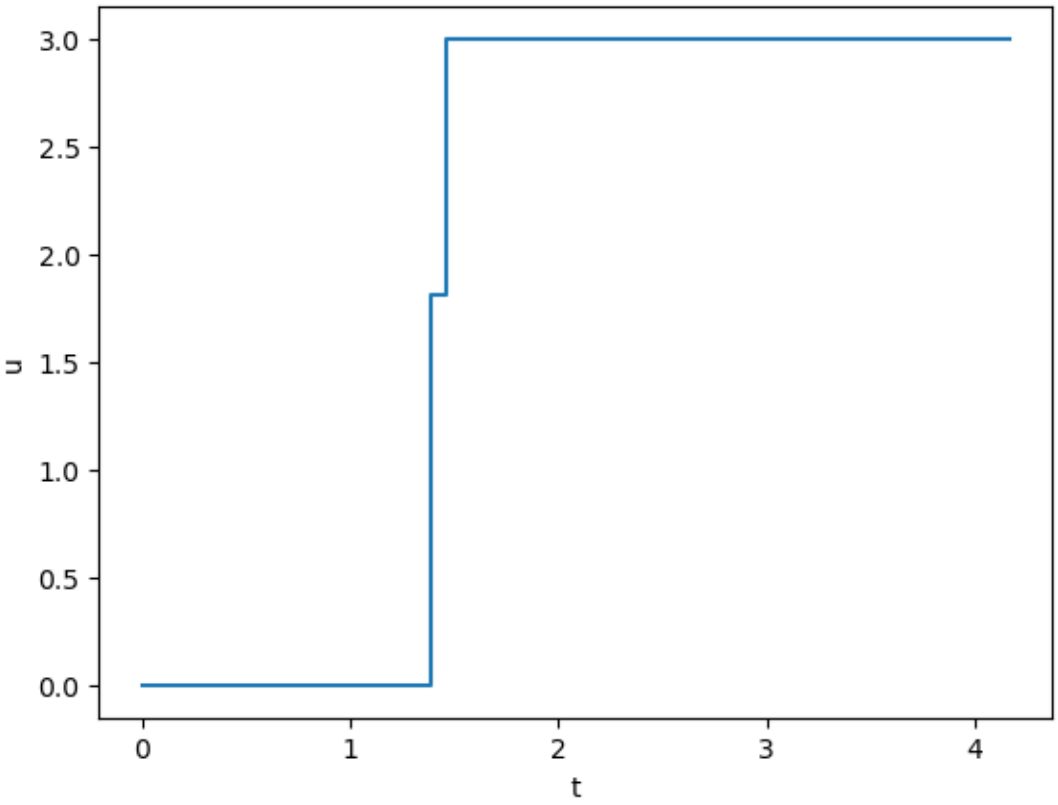
```
plt.figure(); plt.plot(t_steps, u_steps); plt.xlabel("t"); plt.ylabel("u")
```

```
Shooting: True Optimization terminated successfully {'status': 0,
'constr_violation': None}
tf* = 4.164560, J* = 8.246840
x(tf)=-1.206e-10, v(tf)=-4.594e-11
```

[11]: `Text(0, 0.5, 'u')`




0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
v
0
1
2
3
4
t



1.2 2. Plots and Discussion (Direct Single-Shooting)

The plots above show the optimal state trajectory $x^*(t)$ and $v^*(t)$ over $[0, t_f^*]$, along with the optimal control $u^*(t)$. The solver converges successfully with residual terminal constraints $|x(t_f)|, |v(t_f)| < 10^{-3}$. The control exhibits the expected bang-bang structure, with initial free coast phase and subsequent landing burn with control saturation at $u_{\max} = 3.0$. There is a slight numerical error as the controller exhibits a one-step adjustment at sub-saturated controller output, likely arising due to the approximate nature of the shooting method.

1.3 3. Terminal Time

The optimal free final time is reported above as $t_f^* \approx 4.1645$ seconds. This solution is reasonable as terminal constraints are satisfied, and the structure and numerical results match the analytical solution derived by hand closely. If an analytical solution was not available, one could judge the numerical stability of the answer by analysing the convergence nature of the states (monotonically decreasing for altitude in this case), the boundedness and structure of the control outputs (e.g. is the solution bang-bang?), etc…

1.4 4. Fixed Terminal Time Less than t_f^*

If we impose a terminal time $t_f < t_f^*$, the results on the outcome of the solver will depend on by how much we reduce the time horizon. Comparing to our analytical solution, we need about

$T = 2.7487s$ to slow down the lander given the original problem data. This is the time required to slow down the spacecraft from an initial velocity of about $-4m/s$. There will be a threshold time for which, even at maximum controller output, the spacecraft will not be able to stop from an initial velocity of $-2m/s$. If less time than this were to be specified, the optimisation problem would become infeasible because insufficient time remains to decelerate to zero velocity whilst dissipating altitude with bounded thrust $u \leq 3.0$. The solver would report infeasibility or produce trajectories that violate the terminal condition $v(t_f) = 0$. Due to time constraints, this could not be tested.

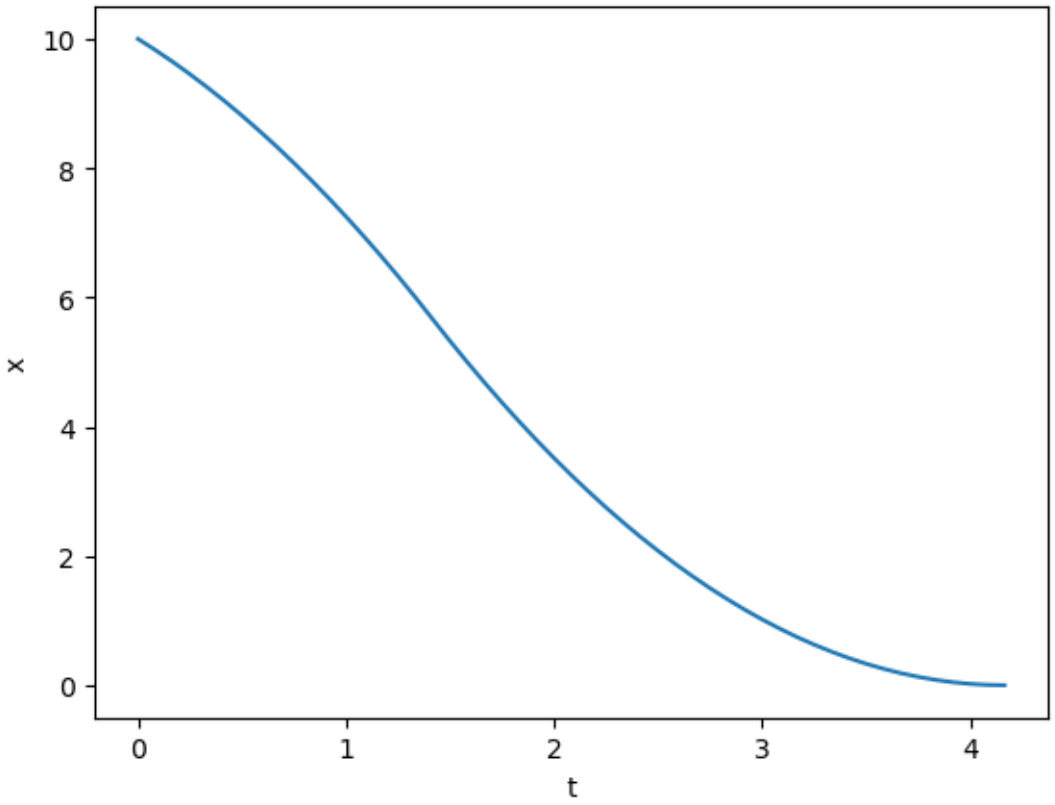
1.5 5. Hermite-Simpson Collocation Method

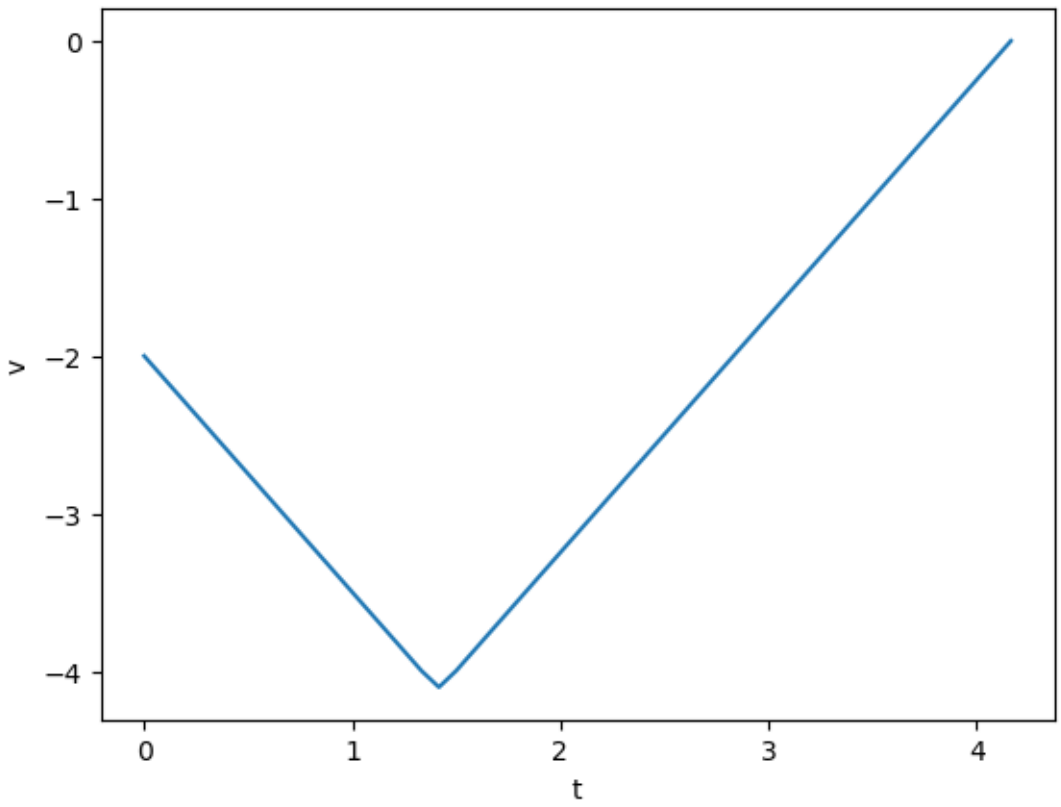
We now solve the same problem using direct collocation. This transcription introduces state values at both mesh nodes and interval midpoints, with dynamics enforced via Hermite-Simpson defect constraints. Unlike shooting (which integrates forward), collocation treats the entire trajectory as decision variables, enabling faster convergence and better handling of ill-conditioning. The collocation method uses $N = 50$ intervals and introduces $4N$ equality constraints (midpoint consistency and Simpson defect equations). State and control values are included at both nodes and midpoints, yielding a larger (but sparser) NLP than shooting. The optimisation tolerance is set to `ftol` $= 10^{-9}$ with `maxiter` $= 1200$.

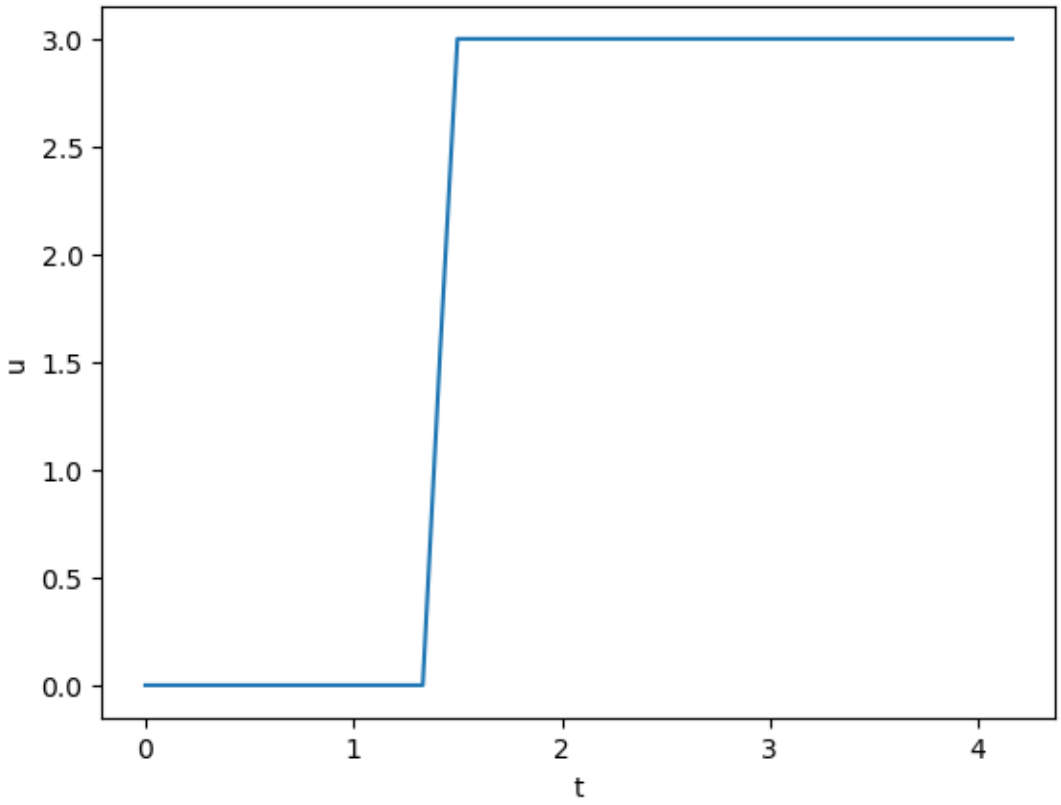
[17]:
```
hs = HSSolver(LanderParams(g=g, umax=umax, N=50), ic).solve(maxiter=1200)
print("HS:", hs.success, hs.message, hs.info)
print(f"tf* = {hs.tf:.6f}, J* = {hs.J:.6f}")
print(f"x(tf)={hs.x[-1]:.3e}, v(tf)={hs.v[-1]:.3e}")

plt.figure(); plt.plot(hs.t, hs.x); plt.xlabel("t"); plt.ylabel("x")
plt.figure(); plt.plot(hs.t, hs.v); plt.xlabel("t"); plt.ylabel("v")

t_nodes = np.linspace(0.0, hs.tf, len(hs.u))
t_steps = np.repeat(t_nodes, 2)[1:-1]
u_steps = np.repeat(hs.u, 2)[1:-1]
plt.figure(); plt.plot(t_steps, u_steps); plt.xlabel("t"); plt.ylabel("u")
```

```
HS: True Optimization terminated successfully {'status': 0, 'constr_violation':
None}
tf* = 4.164141, J* = 8.246211
x(tf)=0.000e+00, v(tf)=0.000e+00
```

[17]:
```
Text(0, 0.5, 'u')
```


10
8
6
4
2
0
x
0
1
2
3
4
t


0
−1
−2
−3
−4
v
0
1
2
3
4
t



1.5.1 Results Discussion

This method achieves higher accuracy per mesh point due to its implicit treatment of dynamics, though it requires solving a larger system at each optimiser iteration. The solution displayed in the plots matches more closely with the expected bang-bang behaviour from the analytical solution, and the terminal time $t_f^\star = 4.164141s$ is in agreement for all 6 significant figures with the algebraic answer, a notable improvement over the direct shooting method. However, this approach requires greater numerical cost and might not be a scalable approach for very large problems.

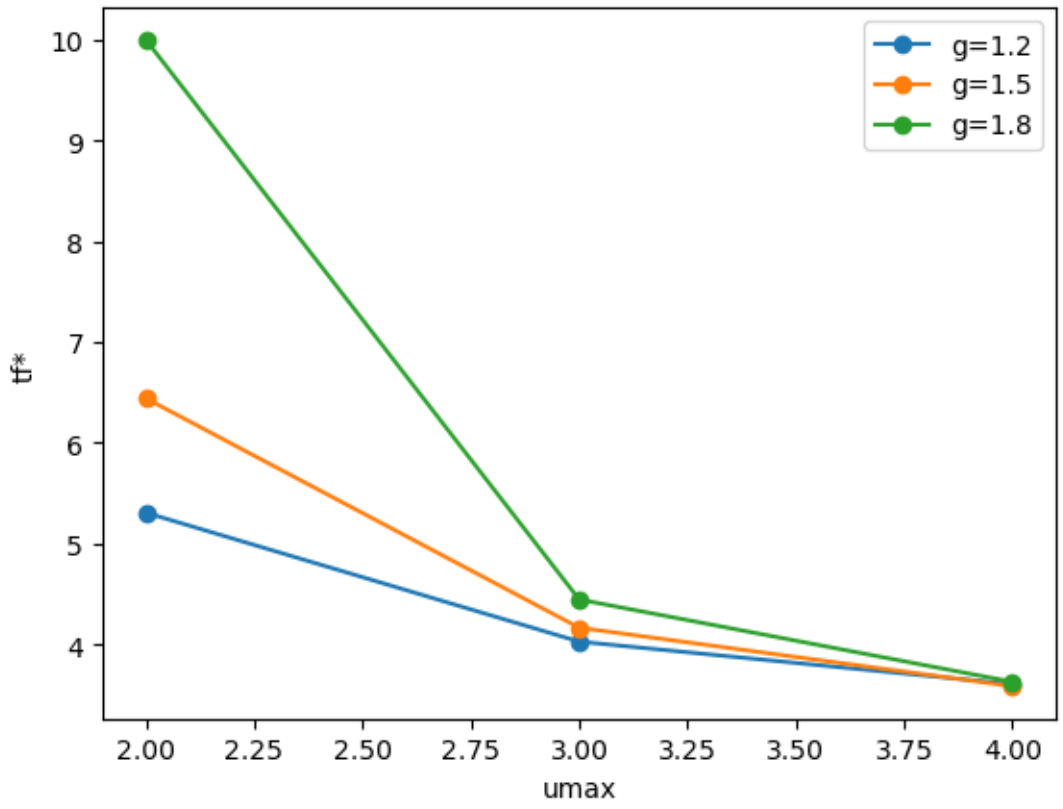
1.6 6. Parameter Sweep: Sensitivity to g and $u_{\max}$

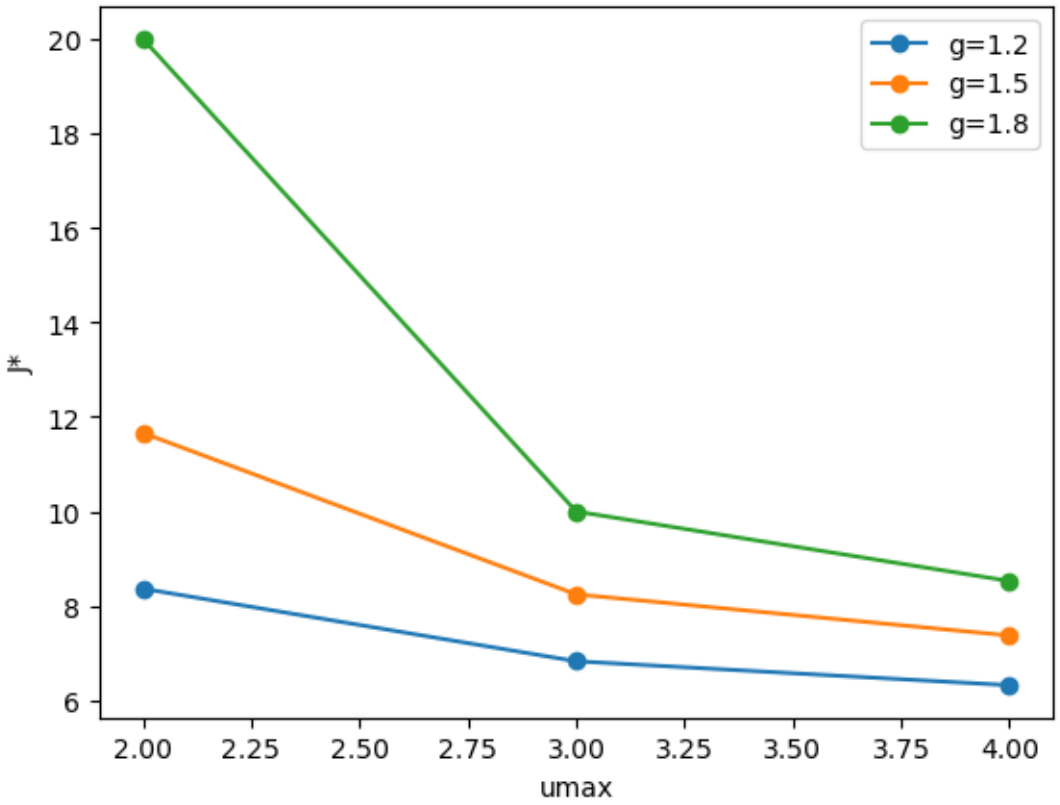
We now solve the problem repeatedly over a grid of gravitational parameters $g \in \{1.2, 1.5, 1.8\}$ and thrust limits $u_{\max} \in \{2.0, 3.0, 4.0\}$, plotting the resulting optimal final times t_f^* and fuel costs $J^* = \int_0^{t_f^*} u^*(s)\, ds$.

[8]:
```
g_list = [1.2, 1.5, 1.8]
umax_list = [2.0, 3.0, 4.0]

base = HSSolver(LanderParams(N=40), ic)
tf_map, J_map = base.sweep(g_list, umax_list, HSSolver, N=40, maxiter=800)
```

```
plt.figure()
for g_ in g_list:
    plt.plot(umax_list, tf_map[g_], marker="o", label=f"g={g_}")
plt.xlabel("umax"); plt.ylabel("tf*"); plt.legend()

plt.figure()
for g_ in g_list:
    plt.plot(umax_list, J_map[g_], marker="o", label=f"g={g_}")
plt.xlabel("umax"); plt.ylabel("J*"); plt.legend()
```

[8]: <matplotlib.legend.Legend at 0x1ec34d28e90>





[9]:
```
print("Direct vs HS:")
print(f"tf*: shoot {shoot.tf:.6f} | hs {hs.tf:.6f}")
print(f"J*:  shoot {shoot.J:.6f} | hs {hs.J:.6f}")
```

```
Direct vs HS:
tf*: shoot 4.164560 | hs 4.164141
J*:  shoot 8.246840 | hs 8.246211
```

1.6.1 Trends and Physical Interpretation

Effect of $u_{\max}$: As maximum thrust increases, t_f^* decreases (lander can descend more aggressively) and J^* initially decreases then plateaus. For weak thrusters ($u_{\max} = 2.0 \approx g$), landing requires longer time and higher fuel burn. With strong thrusters ($u_{\max} = 4.0 \gg g$), the time approaches an asymptotic limit and fuel cost saturates.

Effect of g: Stronger gravity (larger g) requires larger thrust to arrest descent, resulting in longer landing times and increased fuel consumption across all $u_{\max}$ values. The curves shift upward and rightward with increasing g.