Exercise: Direct collocation

In this exercise we consider a nonlinear system with $x \in \mathbb{R}^8$: $\dot{x} = A\sqrt{x} + Bu$, with A a sparse matrix, and a steady state $\sqrt{x^s} = -A^{-1}Bu_s$, and $u_s = \begin{bmatrix}1 & 1\end{bmatrix}^T$.
Locate `sparse_system`, which contains a reference implementation of the following transcribed optimal control problem:

$$
\begin{array}{ll}
\underset{x_1,x_2,\dots x_{N+1},u_1,u_2,\dots,u_N,\bar{x}}{\text{minimize}} & 1\times 10^{-6}\cdot \sum_{k=1}^{N} \|u_k\|_2^2 + \|x_{N+1}-\bar{x}\|_2^2 \\
\text{subject to} & F(x_k,u_k) = x_{k+1}, \quad k = 1\dots N \\
 & 0.01 \le x_k \le 0.1, \quad k = 1\dots N+1 \\
 & x_1 = x^s.
\end{array} \tag{1}
$$

Here, F is implemented using a CasADi Runge-Kutta Integrator Function.
Verify that the reference file runs successfully.

1 Direct collocation

Tasks:

1. Note down the timing information of IPOPT: `Total CPU secs in IPOPT (w/o function evaluations)` and `Total CPU secs in NLP function evaluations`
2. Change the Runge-Kutta `rk` integrator to a `collocation` integrator. This runs a lot slower. Can you explain?

   Is the numerical result identical to the Runge-Kutta integrator?
3. Introduce extra decision variables to accommodate for the helper states at the collocation points X^c. Do this inside the gap-closing for loop.
4. Open up the reference solution for 6 of the collocation integrator exercise. We had a CasADi Function Π for the Legendre polynomial exactly interpolating through initial state X_0 and collocation helper states X^c. Use `is_linear` to verify that Π and therefore $\dot{\Pi}$ are in fact linear in the coefficients.
5. The linear mapping can be obtained from `collocation_coeff`. Using its help, add the collocation constraints inside the multiple shooting for loop, and update the gap-closing constraint. Use a 'radau' scheme with degree 3, which is the default for CasADi's collocation Integrator.

   Verify that the solver converges to the exact same solution as in 1.2.

   Compare the runtimes.
6. Inspect the sparsity of the constraint Jacobian.

   Where do coefficients come from?

2 Lifting Runge-Kutta

Tasks:

1. Start again from the original implementation `sparse_system`. Replace the call to discretized system `F` with an explicit Runge-Kutta formula inside the gap-closing loop:

   ```
   for k=1:N
       x = X(:,k);
       u = U(:,k);
       k1 = f(x, u);
       k2 = f(x + dt/2 * k1, u);
       k3 = f(x + dt/2 * k2, u);
       k4 = f(x + dt * k3, u);
       xf = x+dt/6*(k1 +2*k2 +2*k3 +k4);
       opti.subject_to(X(:,k+1)==xf);
   end
   ```

   Verify that this change does not alter the optimal solution.

   Inspect the constraint Jacobian sparsity of this transcription, and compare it to the reference implementation. Is the part related to the discretized dynamics sparse or dense?

2. Lift the intermediate expressions `k1`, `k2`, `k3`, `k4`: create decision variables for them, and replace assignments with constraints.

   Does this give the same numerical result? Is the part related to the discretized dynamics sparse or dense?