

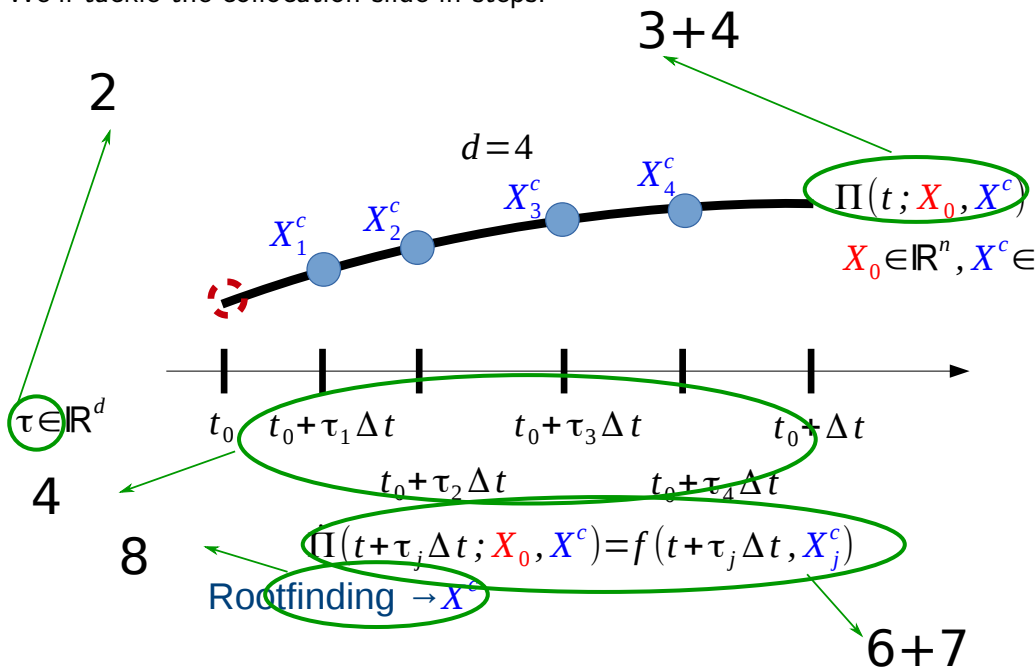
## Exercise: Collocation integrator

In this exercise, we'll be integrating the following time-dependent ODE:

$$\frac{d \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}{dt} = \begin{bmatrix} (1 - x_2^2)x_1 - x_2 + t \\ x_1 \end{bmatrix},$$

from  $t_0 = 2$  to  $t_f = 2.1$ , with an initial value of  $x_0 = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$ .

We'll tackle the collocation slide in steps:



Tasks:

1. Implement a CasADi Function  $f$  that maps from time and state to the ODE's right-hand side. Create suitable MX symbols and use the following boilerplate:

```
f = Function('f', cell_of_input_symbols, cell_of_output_expressions, ...
            cell_of_input_labels, cell_of_output_labels);
```

Choose the cells such that  $f$  is represented as  $f: (t, x[2]) \rightarrow (rhs[2])$  MXFunction

Verify that you obtain  $[2.25; 1]$  for the given initial value.

2. We will use a degree  $d = 4$  legendre scheme for our collocation integrator. Make use of CasADi's `collocation_points` helper function to obtain values for the dimensionless collocation points  $\tau$ . Use `help collocation_points` for help.

3. Locate and download the `LagrangePolynomialEval` function in the course material.

Its mathematical prescription is given by:

$$p([X_1, \dots, X_N], [Y_1, \dots, Y_N], x) = \sum_{i=1}^N \left( \prod_{\substack{1 \leq j \leq N \\ j \neq i}} \frac{x - X_j}{X_i - X_j} \right) Y_i \quad (1)$$

Plot the interpolating values for  $X=[0 \ 0.5 \ 1]$ ,  $Y=[7 \ 1 \ 3]$  and a range of  $x$  values between 0 and 1. Verify that the function performs an exact interpolation through each point  $(X_i, Y_i)$ .

4. Scale the dimensionless collocation points  $\tau_i$  to obtain a collocation grid of the integration interval  $[t_0, t_f]$ .
5. Next step is to construct  $\Pi$  as a CasADi Function with the following signature:

```
Pi:(t,X0[2],Xc[2x4])-(Pi[2]) MXFunction
```

We desire that this Function interpolates exactly through  $(t_0, X_0)$  and also through the points constructed by the collocation grid and the helper states  $X_c$  defined on that grid.

You'll need to introduce some MX symbols, and call `LagrangePolynomialEval` with three symbolic expressions as arguments. Verify that  $\Pi$  evaluated at  $t = t_0 + 0.05$ ,  $X_0 = x_0$ , and  $X_c = [x_0+1 \ x_0+2 \ x_0+4 \ x_0+5]$ , yields  $[3.6501; 3.1501]$ .

6. Construct a CasADi function of the time-derivative of the  $\Pi$ :

```
dot_Pi:(t,X0[2],Xc[2x4])-(dPi[2]) MXFunction
```

Verify that it evaluates to  $[61.1122; 61.1122]$  for the same evaluation point.

7. Create a 8-by-1 expression  $g$  that represent the collocation residual (polynomial derivative should match right-hand side) for our system on the integration interval. Verify that, for  $X_0 = x_0$ , and  $X_c = [x_0+1 \ x_0+2 \ x_0+4 \ x_0+5]$ , this yields  $[107.130; 103.137; 35.726; 16.509; \dots]$ .

Hint: To create a matrix expression out of individual parts formed in a for-loop, use the following idiom:

```
g = {};
for j=1:d
    g{end+1} = ...;
end
g = vertcat(g{:});
```

8. Finally, use a rootfinder to find the values of helper states  $X^c$ . Use  $X_0$  as a parameter ( $p$ ) of the rootfinder. Verify that you have an equal amount of unknowns and equations first. Note that the rootfinder expects a column vector of unknowns. You may need to flatten a matrix with `mat(:)` or equivalent `vec(mat)`.

Verify that your collocation integrator ends up at  $x_f = [1.226454824197, 0.6113162319035]$ .

9. Verify that this result is identical to the result from CasADi's built-in integrators:

```
ode = struct('x',x,'t',t,'ode',rhs);
options = struct;
options.t0 = t0;
options.tf = tf;
options.number_of_finite_elements = 1;
options.interpolation_order = 3;
intg = integrator('intg','collocation',ode,options);
res=intg('x0',x0);
res.xf
```