

## Problem Formulation

$$\begin{aligned}
& \text{/* Cost Function */} \\
\min_{x(\cdot), u(\cdot), z(\cdot), s(\cdot), s^e} \quad & \int_0^T l(x(\tau), u(\tau), z(\tau), p) + \frac{1}{2} \begin{bmatrix} s_l(\tau) \\ s_u(\tau) \\ 1 \end{bmatrix}^\top \begin{bmatrix} Z_l & 0 & z_l \\ 0 & Z_u & z_u \\ z_l^\top & z_u^\top & 0 \end{bmatrix} \begin{bmatrix} s_l(\tau) \\ s_u(\tau) \\ 1 \end{bmatrix} d\tau + \\
& m(x(T), z(T), p) + \frac{1}{2} \begin{bmatrix} s_l^e \\ s_u^e \\ 1 \end{bmatrix}^\top \begin{bmatrix} Z_l^e & 0 & z_l^e \\ 0 & Z_u^e & z_u^e \\ z_l^{e\top} & z_u^{e\top} & 0 \end{bmatrix} \begin{bmatrix} s_l^e \\ s_u^e \\ 1 \end{bmatrix} \\
& \text{/* Initial value */} \\
\text{s.t.} \quad & x(0) - \bar{x}_0 = 0, \\
& \text{/* Dynamics */} \\
& f_{\text{impl}}(x(t), \dot{x}(t), u(t), z(t), p) = 0, \quad t \in [0, T], \\
& \text{/* Path Constraints with lower bound */} \\
& \underline{h} \leq h(x(t), u(t), p) + J_{\text{sh}} s_{l,h}(t), \quad t \in [0, T], \\
& \underline{x} \leq J_{\text{bx}} x(t) + J_{\text{sbx}} s_{l,bx}(t), \quad t \in [0, T], \\
& \underline{u} \leq J_{\text{bu}} u(t) + J_{\text{sbu}} s_{l,bu}(t), \quad t \in [0, T], \\
& \underline{g} \leq Cx(t) + Du(t) + J_{\text{sg}} s_{l,g}(t), \quad t \in [0, T], \\
& s_{l,h}(t), s_{l,bx}(t), s_{l,bu}(t), s_{l,g}(t) \geq 0, \quad t \in [0, T], \\
& \text{/* Path Constraints with upper bound */} \\
& h(x(t), u(t), p) - J_{\text{sh}} s_{u,h}(t) \leq \bar{h}, \quad t \in [0, T], \\
& J_{\text{bx}} x(t) - J_{\text{sbx}} s_{u,bx}(t) \leq \bar{x}, \quad t \in [0, T], \\
& J_{\text{bu}} u(t) - J_{\text{sbu}} s_{u,bu}(t) \leq \bar{u}, \quad t \in [0, T], \\
& Cx(t) + Du(t) - J_{\text{sg}} s_{u,g}(t) \leq \bar{g}, \quad t \in [0, T], \\
& s_{u,h}(t), s_{u,bx}(t), s_{u,bu}(t), s_{u,g}(t) \geq 0, \quad t \in [0, T], \\
& \text{/* Terminal Constraints with lower bound */} \\
& \underline{h}^e \leq h^e(x(T), p) + J_{\text{sh}}^e s_{l,h}^e, \\
& \underline{x}^e \leq J_{\text{bx}}^e x(T) + J_{\text{sbx}}^e s_{l,bx}^e, \\
& \underline{g}^e \leq C^e x(T) + J_{\text{sg}}^e s_{l,g}^e \leq \bar{g}^e, \\
& s_{l,h}^e, s_{l,bx}^e, s_{l,bu}^e, s_{l,g}^e \geq 0, \\
& \text{/* Terminal Constraints with upper bound */} \\
& h^e(x(T), p) - J_{\text{sh}}^e s_h^e \leq \bar{h}^e, \\
& J_{\text{bx}}^e x(T) - J_{\text{sbx}}^e s_{bx}^e \leq \bar{x}^e, \\
& C^e x(T) - J_{\text{sg}}^e s_g^e \leq \bar{g}^e \\
& s_{u,h}^e, s_{u,bx}^e, s_{u,bu}^e, s_{u,g}^e \geq 0,
\end{aligned}$$

with

- state vector  $x : \mathbb{R} \rightarrow \mathbb{R}^{n_x}$
- control vector  $u : \mathbb{R} \rightarrow \mathbb{R}^{n_u}$
- algebraic state vector  $z : \mathbb{R} \rightarrow \mathbb{R}^{n_z}$
- model parameters  $p \in \mathbb{R}^{n_p}$
- slacks for path constraints  $s_l(t) = (s_{l,bu}, s_{l,bx}, s_{l,g}, s_{l,h}) \in \mathbb{R}^{n_s}$  and  $s_u(t) = (s_{u,bu}, s_{u,bx}, s_{u,g}, s_{u,h}) \in \mathbb{R}^{n_s}$
- slacks for terminal constraints  $s_l^e(t) = (s_{l,bx}^e, s_{l,g}^e, s_{l,h}^e) \in \mathbb{R}^{n_s^e}$  and  $s_u^e(t) = (s_{u,bx}^e, s_{u,g}^e, s_{u,h}^e) \in \mathbb{R}^{n_s^e}$

# 1 Dynamics

The function  $f_{\text{impl}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_x+n_z}$  describes the dynamics as a fully implicit DAE.

We offer to discretize  $F$  with a classic implicit Runge-Kutta (**irk**) or a structure exploiting implicit Runge-Kutta method (**irk\_gnsf**).

Additionally, we offer an explicit Runge-Kutta integrator (**erk**), which can be used with explicit ODE models, i.e. models of the form

$$f_{\text{expl}}(x, u, p) = \dot{x}$$

Mathematical Expression	string identifier	data type	required
$f_{\text{impl}}$ respectively $f_{\text{expl}}$	<b>dyn_expr_f</b>	CasADi expression	yes
-	<b>dyn_type</b>	string ( <b>explicit</b> or <b>implicit</b> )	yes

# 2 Cost

There are different **acados** modules to model the cost functions.

- $l : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_z} \rightarrow \mathbb{R}$  is the Lagrange objective term.
- $m : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \rightarrow \mathbb{R}$  is the Mayer objective term.

to decide which one is used set **cost\_type** for  $l$ , **cost\_type\_e** for  $m$ .

Setting the slack penalties is done in the same way for all cost modules, namely:

Mathematical Expression	string identifier	data type	required
$Z_l$	<b>cost_Zl</b>	double	no
$Z_u$	<b>cost_Zu</b>	double	no
$z_l$	<b>cost_zl</b>	double	no
$z_u$	<b>cost_zu</b>	double	no
$Z_l^e$	<b>cost_Zl_e</b>	double	no
$Z_u^e$	<b>cost_Zu_e</b>	double	no
$z_l^e$	<b>cost_zl_e</b>	double	no
$z_u^e$	<b>cost_zu_e</b>	double	no

Moreover, you can specify **cost\_Z**, to set  $Z_l, Z_u$  to the same values, i.e. use a symmetric L2 slack penalty. Similarly, **cost\_z**, **cost\_Z\_e**, **cost\_z\_e** can be used to set symmetric slack L1 penalties, respectively penalties for the terminal slack variables.

## Cost module: auto

Set **cost\_type** to **auto** (default). In this case we detect if the cost function specified is a linear least squares term and transcribe it in the corresponding form. Otherwise, it is formulated using the external cost module. Note: slack penalties are optional and we plan to detect them from the expressions in future versions.

Mathematical Expression	string identifier	data type	required
$l$	<b>cost_expr_ext_cost</b>	CasADi expression	yes
$m$	<b>cost_expr_ext_cost_e</b>	CasADi expression	yes

## Cost module: external

Set `cost_type` to `ext_cost`.

Mathematical Expression	string identifier	data type	required
$l$	<code>cost_expr_ext_cost</code>	CasADi expression	yes
$m$	<code>cost_expr_ext_cost_e</code>	CasADi expression	yes

## Cost module: linear least squares

Set `cost_type` to `linear_ls`.

The Lagrange cost term has the form

$$l(x, u, z) = \left\| \underbrace{V_x x + V_u u + V_z z}_{y} - y_{\text{ref}} \right\|_W$$

with matrices  $V_x, V_u, V_z, W$  of appropriate dimensions.

Similarly, the Mayer cost term has the form

$$m(x, u, z) = \left\| \underbrace{V_x^e x}_{y^e} - y_{\text{ref}}^e \right\|_{W^e}$$

with matrices  $V_x^e, W^e$  of appropriate dimensions.

Mathematical Expression	string identifier	data type	required
$V_x$	<code>cost_V_x</code>	double	yes
$V_u$	<code>cost_V_u</code>	double	yes
$V_z$	<code>cost_V_z</code>	double	yes
$W$	<code>cost_W</code>	double	yes
$y_{\text{ref}}$	<code>cost_y_ref</code>	double	yes
$V_x^e$	<code>cost_V_x_e</code>	double	yes
$W^e$	<code>cost_W_e</code>	double	yes
$y_{\text{ref}}^e$	<code>cost_y_ref_e</code>	double	yes

## Cost module: nonlinear least squares

Set `cost_type` to `nonlinear_ls`.

The cost function has the same form as in the linear least squares module.

The only difference is that  $y$ , respectively  $y^e$  are defined as `CasADi` expressions, instead of the matrices  $V_x, V_u, V_z$ , respectively  $V_x^e$

Mathematical Expression	string identifier	data type	required
$y$	<code>cost_expr_y</code>	<code>CasADi</code> expression	yes
$W$	<code>cost_W</code>	double	yes
$y_{\text{ref}}$	<code>cost_y_ref</code>	double	yes
$y^e$	<code>cost_expr_y_e</code>	<code>CasADi</code> expression	yes
$y_{\text{ref}}^e$	<code>cost_y_ref_e</code>	double	yes

## 3 Path Constraints

Mathematical Expression	string identifier	data type	required
$\bar{x}_0$	<code>constr_x0</code>	double	yes
$J_{\text{bx}}$	<code>constr_Jbx</code>	double	no
$\underline{x}$	<code>constr_lbx</code>	double	no
$\bar{x}$	<code>constr_ubx</code>	double	no
$J_{\text{bu}}$	<code>constr_Jbu</code>	double	no
$\underline{u}$	<code>constr_lbu</code>	double	no
$\bar{u}$	<code>constr_ubu</code>	double	no
$C$	<code>constr_C</code>	double	no
$D$	<code>constr_D</code>	double	no
$\underline{g}$	<code>constr_lg</code>	double	no
$\bar{g}$	<code>constr_uh</code>	double	no
$h$	<code>constr_expr_h</code>	<code>CasADi</code> expression	no
$\underline{h}$	<code>constr_lh</code>	double	no
$\bar{h}$	<code>constr_uh</code>	double	no
$J_{\text{sbx}}$	<code>constr_Jsbx</code>	double	no
$J_{\text{sbu}}$	<code>constr_Jsbu</code>	double	no
$J_{\text{sg}}$	<code>constr_Jsg</code>	double	no
$J_{\text{sbx}}$	<code>constr_Jsh</code>	double	no

## 4 Terminal Constraints

Mathematical Expression	string identifier	data type	required
$J_{\text{bx}}$	<code>constr_Jbx_e</code>	double	no
$\underline{x}^e$	<code>constr_lbx_e</code>	double	no
$\bar{x}^e$	<code>constr_ubx_e</code>	double	no
$C^e$	<code>constr_C_e</code>	double	no
$\underline{g}^e$	<code>constr_lg</code>	double	no
$\bar{g}^e$	<code>constr_ug</code>	double	no
$h^e$	<code>constr_expr_h_e</code>	CasADi expression	no
$\underline{h}^e$	<code>constr_lh_e</code>	double	no
$\bar{h}^e$	<code>constr_uh_e</code>	double	no
$J_{\text{sbx}}$	<code>constr_Jsbx</code>	double	no
$J_{\text{sg}}^e$	<code>constr_Jsg_e</code>	double	no
$J_{\text{sbx}}^e$	<code>constr_Jsh_e</code>	double	no