Problem Formulation

$$
\begin{aligned}
\min_{x(\cdot),\,u(\cdot),\,z(\cdot),\,s(\cdot)} \quad & \int_0^T l(x(\tau),u(\tau),z(\tau),p) + \frac{1}{2}s(\tau)^\top Z s(\tau) + z_s^\top \,|s(\tau)|\,\mathrm{d}\tau + \\
& \quad m(x(T),z(T),s(T),p) + \frac{1}{2}s(T)^\top Z^{\mathrm{e}} s(T) + z_s^{\mathrm{e}\top}\,|s(T)| \\
\text{s.t.} \quad & x(0) - \bar{x}_0 = 0, \\
& f_{\mathrm{impl}}(x(t),\dot{x}(t),u(t),z(t),p) = 0, && t\in[0,\,T), \\
& \underline{h} \le h(x(t),u(t),p) + J_{\mathrm{sh}} s_{\mathrm{h}} \le \bar{h}, && t\in[0,\,T), \\
& \underline{x} \le J_{\mathrm{bx}} x(t) + J_{\mathrm{sbx}} s_{\mathrm{bx}}(t) \le \bar{x}, && t\in[0,\,T), \\
& \underline{u} \le J_{\mathrm{bu}} u(t) + J_{\mathrm{sbu}} s_{\mathrm{bu}}(t) \le \bar{u}, && t\in[0,\,T), \\
& \underline{g} \le Cx(t) + Du(t) + J_{\mathrm{sg}} s_{\mathrm{g}} \le \bar{g}, && t\in[0,\,T), \\
& \underline{h}^{\mathrm{e}} \le h^{\mathrm{e}}(x(T),p) + J_{\mathrm{sh}}^{\mathrm{e}} s_{\mathrm{h}}^{\mathrm{e}} \le \bar{h}^{\mathrm{e}}, \\
& \underline{x}^{\mathrm{e}} \le J_{\mathrm{bx}}^{\mathrm{e}} x(T) + J_{\mathrm{sbx}} s_{\mathrm{bx}}^{\mathrm{e}} \le \bar{x}^{e}, \\
& \underline{g}^{\mathrm{e}} \le C^{\mathrm{e}} x(T) \le \bar{g}^{\mathrm{e}} + J_{\mathrm{sg}}^{\mathrm{e}} s_{\mathrm{g}}^{\mathrm{e}}
\end{aligned}
$$

Hereby:

- $x \in \mathbb{R}^{n_x}$ state vector
- $u \in \mathbb{R}^{n_u}$ control vector
- $z \in \mathbb{R}^{n_z}$ algebraic state vector
- $s \in \mathbb{R}^{n_s}$ slack variables, which are concatenated as $s = (s_{\mathrm{bu}}, s_{\mathrm{bx}}, s_{\mathrm{g}}, s_{\mathrm{h}})$
- $s^{\mathrm{e}} \in \mathbb{R}^{n_s}$ terminal slack variables, which are concatenated as $s = (s_{\mathrm{bx}}^{\mathrm{e}}, s_{\mathrm{g}}^{\mathrm{e}}, s_{\mathrm{h}}^{\mathrm{e}})$
- $p \in \mathbb{R}^{n_p}$ parameters

1 Dynamics

The function $f_{\mathrm{impl}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x+n_z}$ 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

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

Mathematical Expression	string identifier	data type
f_{impl} respectively f_{expl}	`dyn_expr_f`	`CasADi` expression
-	`dyn_type`	string (`explicit` or `implicit`)

2 Cost

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

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

to decide which one is used set `cost_type` for l, `cost_type_e` for m.

Cost module: `auto`

Set `cost_type` to `auto` (default). Hereby 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 will be detected form the expressions in future versions.

Mathematical Expression	string identifier	data type
l	`cost_expr_ext_cost`	`CasADi` expression
m	`cost_expr_ext_cost_e`	`CasADi` expression
Z	`cost_Z`	double
z_s	`cost_z`	double
Z^{e}	`cost_Z_e`	double
z_s^{e}	`cost_z_e`	double

Cost module: `external`

Set `cost_type` to `ext_cost`.

Mathematical Expression	string identifier	data type
l	`cost_expr_ext_cost`	`CasADi` expression
m	`cost_expr_ext_cost_e`	`CasADi` expression
Z	`cost_Z`	double
z_s	`cost_z`	double
Z^{e}	`cost_Z_e`	double
z_s^{e}	`cost_z_e`	double

Cost module: `linear least squares`

Set `cost_type` to `linear_ls`.
The Lagrange cost term has the form

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

with matrices V_x, V_u, V_z, W of appropriate dimensions.
Similarly, the Mayer cost term has the form

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

with matrices $V_x^{\mathrm{e}}, W^{\mathrm{e}}$ of appropriate dimensions.

Mathematical Expression	string identifier	data type
V_x	`cost_V_x`	double
V_u	`cost_V_u`	double
V_z	`cost_V_z`	double
W	`cost_W`	double
y_{ref}	`cost_y_ref`	double
V_x^{e}	`cost_V_x_e`	double
W^{e}	`cost_W_e`	double
$y_{\text{ref}}^{\text{e}}$	`cost_y_ref_e`	double
Z	`cost_Z`	double
z_s	`cost_z`	double
Z^{e}	`cost_Z_e`	double
z_s^{e}	`cost_z_e`	double

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
y	`cost_expr_y`	`CasADi` expression
W	`cost_W`	double
y_{ref}	`cost_y_ref`	double
y^{e}	`cost_expr_y_e`	`CasADi` expression
$y_{\text{ref}}^{\text{e}}$	`cost_y_ref_e`	double
Z	`cost_Z`	double
z_s	`cost_z`	double
Z^{e}	`cost_Z_e`	double
z_s^{e}	`cost_z_e`	double

3 Path Constraints

Mathematical Expression	string identifier	data type
$\bar{x}_0$	`constr_x0`	double
J_{bx}	`constr_Jbx`	double
$\underline{x}$	`constr_lbx`	double
$\bar{x}$	`constr_ubx`	double
J_{bu}	`constr_Jbu`	double
$\underline{u}$	`constr_lbu`	double
$\bar{u}$	`constr_ubu`	double
C	`constr_C`	double
D	`constr_D`	double
$\underline{g}$	`constr_lg`	double
$\bar{g}$	`constr_ug`	double
h	`constr_expr_h`	`CasADi` expression
$\underline{h}$	`constr_lh`	double
$\bar{h}$	`constr_uh`	double
J_{sbx}	`constr_Jsbx`	double
J_{sbu}	`constr_Jsbu`	double
J_{sg}	`constr_Jsg`	double
J_{sbx}	`constr_Jsh`	double

4 Terminal Constraints

Mathematical Expression	string identifier	data type
J_{bx}	`constr_Jbx_e`	double
$\underline{x}^{\text{e}}$	`constr_lbx_e`	double
$\bar{x}^{\text{e}}$	`constr_ubx_e`	double
C^{e}	`constr_C_e`	double
$\underline{g}^{\text{e}}$	`constr_lg`	double
$\bar{g}^{\text{e}}$	`constr_ug`	double
h^{e}	`constr_expr_h_e`	`CasADi` expression
$\underline{h}^{\text{e}}$	`constr_lh_e`	double
$\bar{h}^{\text{e}}$	`constr_uh_e`	double
J_{sbx}	`constr_Jsbx`	double
J_{sg}^{e}	`constr_Jsg_e`	double
$J_{\text{sbx}}^{\text{e}}$	`constr_Jsh_e`	double