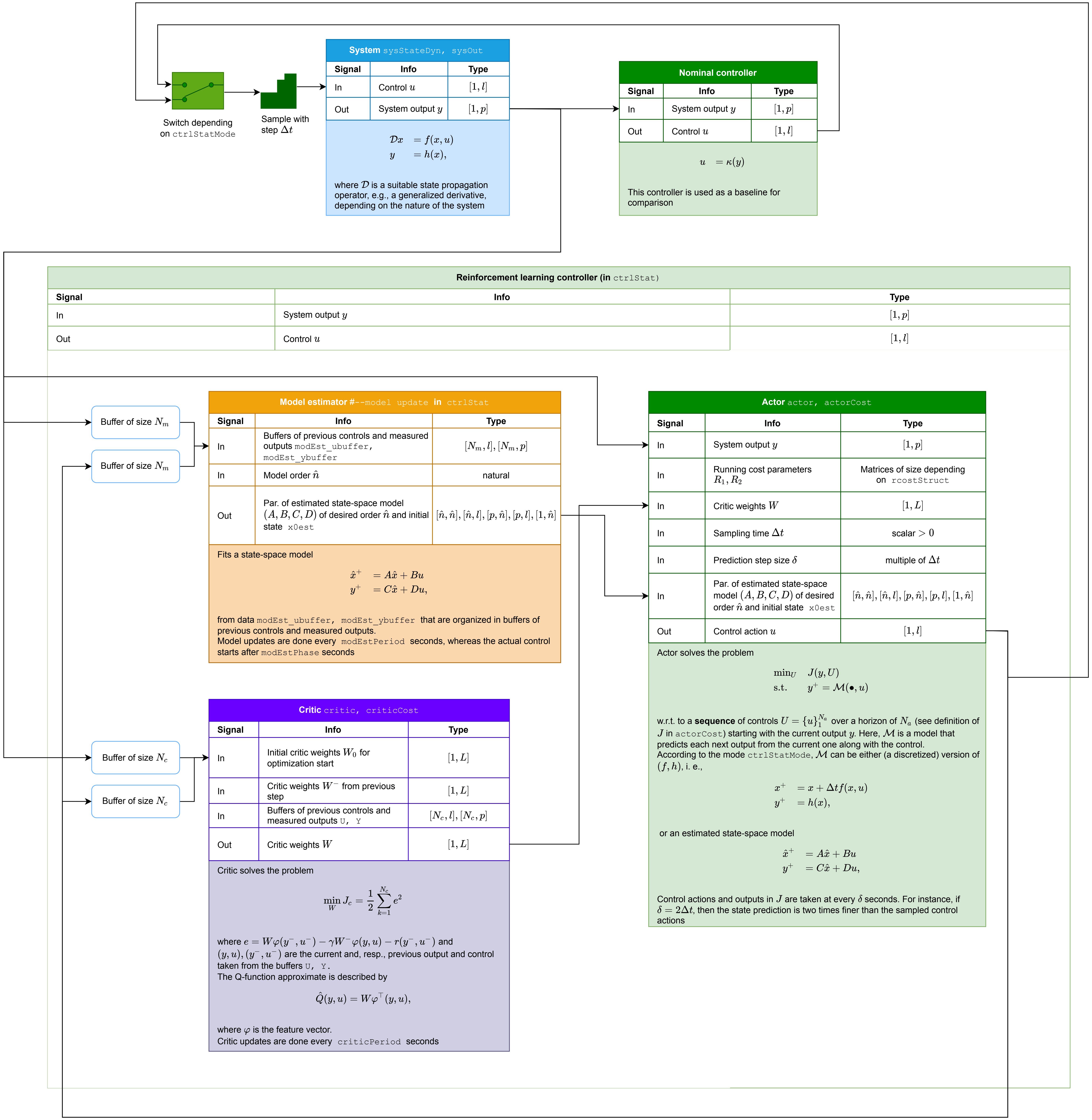
Legend		
Math	**Description**	**In code**
$[\bullet \times \bullet]$	Matrix	`[•,•]`
$[1 \times \bullet]$	vector or 1-dim. array	`[•,]`
r	Running cost	`rcost`
N_a	Actor control horizon	`Nactor`
N_c	Critic stack size	`Ncritic`
N_m	Model estimator stack size	`modEstBufferSize`
J	Controller cost function	`actorCost`
J_c	Critic cost function	`criticCost`
$Q, \hat{Q}$	Q-function and its approximate	
e	Temporal difference	`e`
W, W^-	Current and previous critic weights	`W, Wprev`
R_1, R_2	Parameters of r	`R1, R2`
γ	Discounting factor	`gamma`
L	Number of critic weights	`dimCrit`
Δt	Controller sampling time	`dt`
δ	Prediction step size	`predStepSize`
t_0, t_1	Start time and total time of one episode	`t0, t1`
	Number of episodes	`Nruns`
x, n	State and its dimension	`x, dimState`
u, l	Control input and its dimension	`u, dimInput`
y, p	Output and its dimension	`y, dimOutput`
$\hat{x}, \hat{n}$	Estimated model state and its dimension	`-, modelOrder`

Controller mode `ctrlStatMode`	
0 - Manual control	
10 - Nominal control	
1, 2 - Model-predictive control (MPC)	$J\left(y_1, \{u\}_1^{N_a}\right) = \sum_{k=1}^{N_a} r(y_k, u_k)$
3, 4 - RL/ADP via $N_a - 1$ roll-outs of r	$J\left(y_1, \{u\}_1^{N_a}\right) = \sum_{k=1}^{N_a-1} r(y_k, u_k) + \hat{Q}(y_{N_a}, u_{N_a})$
5, 6 - RL/ADP via normalized stacked Q-learning	$J\left(y_1, \{u\}_1^{N_a}\right) = \frac{1}{N_a} \sum_{k=1}^{N_a-1} r(y_k, u_k) + \hat{Q}(y_{N_a}, u_{N_a})$

Modes 1, 3, 5 use true model (f, h) for prediction
Modes 2, 4, 6 use a state-space model estimated online



System `sysStateDyn, sysOut`

Signal	Info	Type
In	Control u	$[1, l]$
Out	System output y	$[1, p]$

$$\begin{aligned} \mathcal{D}x &= f(x, u) \\ y &= h(x), \end{aligned}$$

where $\mathcal{D}$ is a suitable state propagation operator, e.g., a generalized derivative, depending on the nature of the system

Nominal controller

Signal	Info	Type
In	System output y	$[1, p]$
Out	Control u	$[1, l]$

$$u = \kappa(y)$$

This controller is used as a baseline for comparison

Reinforcement learning controller (in `ctrlStat`)

Signal	Info	Type
In	System output y	$[1, p]$
Out	Control u	$[1, l]$

Model estimator #`--model update` **in** `ctrlStat`

Signal	Info	Type
In	Buffers of previous controls and measured outputs `modEst_ubuffer, modEst_ybuffer`	$[N_m, l], [N_m, p]$
In	Model order $\hat{n}$	natural
Out	Par. of estimated state-space model (A, B, C, D) of desired order $\hat{n}$ and initial state `x0est`	$[\hat{n}, \hat{n}], [\hat{n}, l], [p, \hat{n}], [p, l], [1, \hat{n}]$

Fits a state-space model

$$\begin{aligned} \hat{x}^+ &= A\hat{x} + Bu \\ y^+ &= C\hat{x} + Du, \end{aligned}$$

from data `modEst_ubuffer, modEst_ybuffer` that are organized in buffers of previous controls and measured outputs.
Model updates are done every `modEstPeriod` seconds, whereas the actual control starts after `modEstPhase` seconds

Actor `actor, actorCost`

Signal	Info	Type
In	System output y	$[1, p]$
In	Running cost parameters R_1, R_2	Matrices of size depending on `rcostStruct`
In	Critic weights W	$[1, L]$
In	Sampling time Δt	scalar > 0
In	Prediction step size δ	multiple of Δt
In	Par. of estimated state-space model (A, B, C, D) of desired order $\hat{n}$ and initial state `x0est`	$[\hat{n}, \hat{n}], [\hat{n}, l], [p, \hat{n}], [p, l], [1, \hat{n}]$
Out	Control action u	$[1, l]$

Actor solves the problem

$$\begin{aligned} \min_U \quad & J(y, U) \\ \text{s.t.} \quad & y^+ = \mathcal{M}(\bullet, u) \end{aligned}$$

w.r.t. to a **sequence** of controls $U = \{u\}_1^{N_a}$ over a horizon of N_a (see definition of J in `actorCost`) starting with the current output y. Here, $\mathcal{M}$ is a model that predicts each next output from the current one along with the control.
According to the mode `ctrlStatMode`, $\mathcal{M}$ can be either (a discretized) version of (f, h), i. e.,

$$\begin{aligned} x^+ &= x + \Delta t f(x, u) \\ y^+ &= h(x), \end{aligned}$$

or an estimated state-space model

$$\begin{aligned} \hat{x}^+ &= A\hat{x} + Bu \\ y^+ &= C\hat{x} + Du, \end{aligned}$$

Control actions and outputs in J are taken at every δ seconds. For instance, if $\delta = 2\Delta t$, then the state prediction is two times finer than the sampled control actions

Critic `critic, criticCost`

Signal	Info	Type
In	Initial critic weights W_0 for optimization start	$[1, L]$
In	Critic weights W^- from previous step	$[1, L]$
In	Buffers of previous controls and measured outputs `U, Y`	$[N_c, l], [N_c, p]$
Out	Critic weights W	$[1, L]$

Critic solves the problem

$$\min_W J_c = \frac{1}{2} \sum_{k=1}^{N_c} e^2$$

where $e = W\varphi(y^-, u^-) - \gamma W^- \varphi(y, u) - r(y^-, u^-)$ and $(y, u), (y^-, u^-)$ are the current and, resp., previous output and control taken from the buffers `U, Y.`
The Q-function approximate is described by

$$\hat{Q}(y, u) = W\varphi^\top(y, u),$$

where φ is the feature vector.
Critic updates are done every `criticPeriod` seconds