Background.

The objective of this project is to simulate target detection in a noisy environment. Everything starts with the signal, $\boxed{S_n,\ n \geqslant 1}$, which we assume to be known and deterministic. To model the environment, we assume an auto-regressive noise $\boxed{V_n,\ n \geqslant 1}$, of order p. More specifically

$$V_n = \sum_{i=1}^{p} \varrho_i V_{n-i} + W_n, \qquad n \geqslant 1,$$

where $V_i = 0$ for $i \leqslant 0$; $\varrho_1, \cdots, \varrho_p$ are known with $\varrho_1^2 + \cdots + \varrho_p^2 < 1$; and $W_n \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$ with σ^2 known.

The observed process X_n, $n \geqslant 1$, is defined by

$$\boxed{X_n = \mu S_n + V_n},$$

where μ is unknown, and determines the signal "strength".

Let $p_n = \min\{p, n-1\}$, and for $n \geqslant 1$ define

$$Y_n = X_n - \sum_{i=1}^{p_n} \varrho_i X_{n-i}, \qquad R_n = S_n - \sum_{i=1}^{p_n} \varrho_i S_{n-i}.$$

It is not difficult to see that

$$Y_n = \mu R_n + W_n, \tag{1}$$

and $Y_n \overset{\text{indep.}}{\sim} \mathcal{N}(\mu R_n, \sigma^2)$. We will refer to it as the "adjusted" signal in this project. Note that all of the procedures we are interested in (defined below) will only depend on Y_n, not X_n, so technically one does not need to generate the underlying process (X_n) at all.

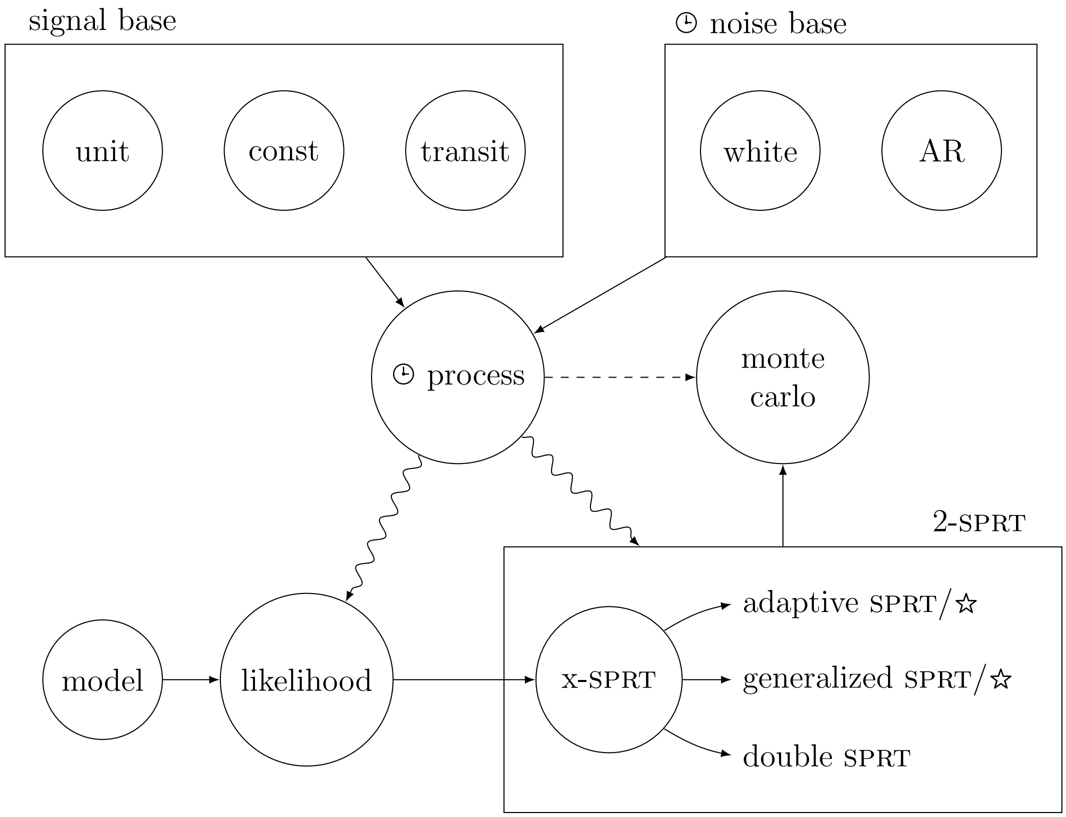
We are now in position to formulate the hypothesis testing problem. Let $\mu_0 = 0$, and $\mu_1 > \mu_0$ be given. We are interested in testing

$$\boxed{\mathcal{H}_0 : \mu = \mu_0 \qquad \text{vs.} \qquad \mathcal{H}_1 : \mu \geqslant \mu_1.} \tag{2}$$

In what follows we will also need to introduce auxiliary hypotheses $\{\mathcal{H}_\theta : \mu = \theta\}$.

Project `hypotheses`.

This project relies heavily on CRTP (curiously recurring template pattern) structure that serves as a compile-time interface/abstract class/virtual class. The singleton structure was taken from this post on `stackoverflow`.



Fundamental structures.

- Signals: `unit_signal`, `constant_signal`, `transitionary_signal`.
- Noises: `white_noise`, `auto_regressive_noise`.
- Processes: `process`.
- Hypotheses models: `model`.
- Decision rules: `adaptive_sprt`(`_star`), `generalized_sprt`(`_star`), `double_sprt`.

CRTP structures.

The following are the base abstract CRTP structures:

- `timed`: for timed random sequences, denoted with ◷ on the diagram.
- `signal_base`: for signals.
- `noise_base`: for noises.
- `observer`: for observers of `process`, denoted with ⇝ on the diagram.
- `two_sprt`: for SPRT-based decision rules.

2-sprt structure.

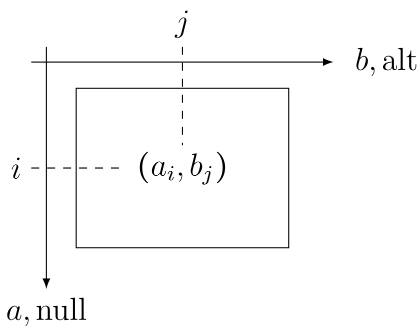
This abstract class represents decision rules (τ, d), where τ is a stopping time of the form $\tau = \min\{\tau_0, \tau_1\}$, and d is the decision ruling in favor of $\mathcal{H}_k$ if $\tau = \tau_k$, for $k = 0, 1$. The auxiliary stopping times are

$$\tau_0 = \{t \mid T_0 > a\}, \qquad \tau_1 = \{t \mid T_1 > b\},$$

where T_0 and T_1 are some scalar-valued SPRT-based statistics, and a, b are thresholds serving as design parameters.

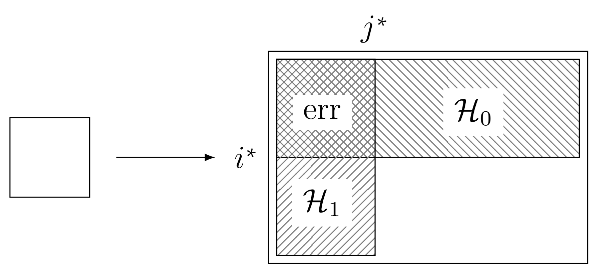
For efficiency reasons, rather than perform simulations for a given pair of thresholds (a, b), one would construct a grid of thresholds $(a_i, b_j)_{i,j}$ with

$$a_1 < a_2 < \cdots < a_m, \qquad b_1 < b_2 < \cdots < b_n.$$



The simulation then would proceed as long as at least one of the thresholds has not been crossed, or equivalently, as long as (a_m, b_n) has not been crossed. For each pair of thresholds, one would keep track of the observed stopping time, and the decision made.

Let i^* denote the index of the first uncrossed null threshold, a_{i^*}. Let j^* denote the index of the first uncrossed alt threshold, b_{j^*}.



When we start off with the first observation, the entire shaded region in the corresponding matrices will be filled; so each next step only require updating the remaining (rectangular) region of the matrix—for either of the two operating characteristics (OC): stopping time, or decision. So with each next step another Γ-shaped block of the OC matrices will be filled out, yielding an opportunity to optimize the whole process.

Project `simulator`.

Now to the actual program. The description of simulations to be run are stored in `config`. More specifically, it contains information necessary to create

- `signal`;
- `noise`;
- `monte_carlo`;
- list of `two_sprt`'s;
- list of `run`'s to be performed.

The description of simulation, `run`, contains information about

- `model`;
- list of `simulation_pair`'s;
- list of thresholds for each `two_sprt`.

A `simulation_pair` is the pair of signal strengths: the one simulated in the `process`, denoted here with ν; and the one analyzed by the `two_sprt`, which we will denote with λ. Regardless of the provided list of `simulation_pair`'s, there are four mandatory pairs to assess basic operating characteristics, summarized in this table:

	$\lambda = \mu_0$	$\lambda = \mu_1$
$\nu = \mu_0$	ESS_{μ_0}	PMS
$\nu = \mu_1$	PFA	ESS_{μ_1}