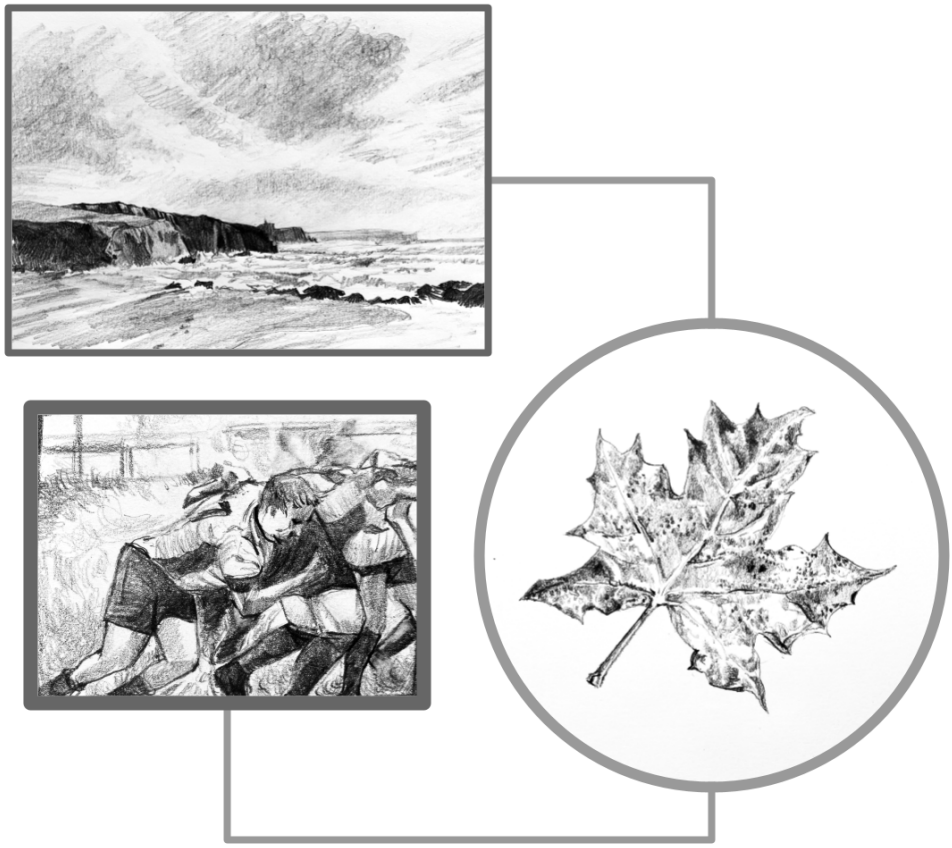

9

Interacting with systems in general

Concept. To design and build a generalised concept of interacting with stochastic processes of any kind. The mathematical formalism and software that we introduce here will serve as a common language and interface for any simulation studies into manipulating real world phenomena, and should enable the learning of control algorithms. We will call this software 'dexetera', since it originates as an extension to the stochadex. For the mathematically-inclined, this chapter will cover how dexetera is structured in theory by developing some useful extensions to the stochadex formalism and illustrating with some simple examples. For the programmers, the public Git repository for the code described in this chapter can be found here: https://github.com/umbralcalc/dexetera.

9.1 Formalising general interactions

Let's start by considering how we might adapt the mathematical formalism we have been using so far to be able to take actions which can manipulate the state at each timestep. Using the mathematical notation that we inherited from the stochadex, we may extend the formula for updating the state history matrix $X' \to X$ to include a vector-valued 'state measurement' function $M_{\mathtt{t}}$ and two layers of possible interactions which are facilitated by a new vector-valued 'parametric action' function $G_{\mathtt{t}}$ and a new vector-valued 'state action' function $H_{\mathtt{t}}$. During a timestep over which these actions are applied, the stochadex state update formula can be extended to look like this system of equations

$$\mathcal{S}^i_{\mathtt{t}} = M^i_{\mathtt{t}}(X', Z_{\mathtt{t}}, \mathtt{t}) \tag{9.1}$$

$$Z^i_{\mathtt{t}+1} = G^i_{\mathtt{t}+1}(Z_{\mathtt{t}}, \mathcal{S}_{\mathtt{t}}, \mathcal{A}_{\mathtt{t}}, \mathtt{t}) \tag{9.2}$$

$$X^i_{\mathtt{t}+1} = H^i_{\mathtt{t}+1}[F_{\mathtt{t}}(X', Z_{\mathtt{t}+1}, \mathtt{t}), S_{\mathtt{t}}, \mathcal{A}_{\mathtt{t}}, \mathtt{t}]\,, \tag{9.3}$$

where, other than the new functions, we have introduced; the concept of a 'measured state' $\mathcal{S}_{\mathtt{t}}$ of the system (some vector of the that doesn't have to share the same length as $X_{\mathtt{t}}$) and the actions performed $\mathcal{A}_{\mathtt{t}}$ on the system (some vector of parameters which define what actions are taken) at timestep $\mathtt{t}$.

In Eqs. (9.2) and (9.3), notice that we have replaced the constant vector of parameters z (as in the stochadex formalism) for a time-dependent vector $Z_\mathtt{t}$ of parameters that can be updated by $G_\mathtt{t}$ at any (but not necessarily every) timestep. From the perspective of the whole matrix X update step, the measurement of $M_\mathtt{t}$ proceeded by the actions of both $G_\mathtt{t}$ and $H_\mathtt{t}$ all combine to become technically equivalent to applying a formal composition of functions like this

$$X^i_{\mathtt{t}+1} = H^i_{\mathtt{t}+1}(F_\mathtt{t}(X', G^i_{\mathtt{t}+1}(Z_\mathtt{t}, M_\mathtt{t}(X', Z_\mathtt{t}, \mathtt{t}), \mathcal{A}_\mathtt{t}, \mathtt{t}), \mathtt{t}), M_\mathtt{t}(X', Z_\mathtt{t}, \mathtt{t}), \mathcal{A}_\mathtt{t}, \mathtt{t}) \tag{9.4}$$

$$= \mathcal{F}^i_{\mathtt{t}+1}(X', Z_\mathtt{t}, \mathcal{A}_\mathtt{t}, \mathtt{t})\,, \tag{9.5}$$

which looks like an absolute mess! However, it illustrates how $\mathcal{F}$, which refers to a modified version of the F function, contains our actions but no other new parameters need be specified; only function operations. Hence, while we have provided two distinct ways one might encode actions to manipulate a stochastic phenomenon, we shall often just refer to them together as 'taking an action' $\mathcal{A}_\mathtt{t}$ at timestep $\mathtt{t}$ — because the parameters which define either type of action at this timestep should all be stored within $\mathcal{A}_\mathtt{t}$ anyway. It is, however, important not to forget the full mathematical formulation when performing calculations.

In Go, the code for the new iteration formula given by Eq. (9.5), which includes taking actions in the same timestep, would look something like this.

```
type ActionsVector       []float64
type MeasuredStateVector []float64

// iterate the state history forward in time by one step
// with actions in the process
func IterationFormulaWithActions(
    stateHistory StateHistory,
    otherParams  OtherParams,
    actions ActionsVector,
    timeStepNumber int,
) StateVector {
    // measuring the state
    var measuredState MeasuredStateVector
    measuredState = MeasureTheState(stateHistory, otherParams, timeStepNumber)
    // taking a parametric action
    newestOtherParams := TakeParametricAction(
        otherParams,
        measuredState,
        actions,
        timeStepNumber,
    )
    newestStateVector := IterationFormula(
        stateHistory,
        otherParams,
        timeStepNumber,
    )
    // taking a state action
    newestStateVector = TakeStateAction(
        newestStateVector,
        measuredState,
        actions,
        timeStepNumber,
    )
    return newestStateVector
}
```

Bibliography