Dynamical systems: integral and differential formulations, transition from continuous to discrete time and Volterra kernels

**Continuous dynamical systems: from integral to differential forms**

Let  be defined by the following integral equation:

 (A9)

where  is an arbitrary function time. If  is an invariant kernel (that is, if ), then the right-hand term of equation A9 becomes a convolution operation and  is simply the impulse response function of (to inputs ). In this case, one can derive a differential form of Eq. A9. For example, let  be given by the truncated (causal) exponential kernel:

 (A10)

Applying the Fourier transform on the left and right hand sides of equation A9 then yields:

 (A11)

where the tilde notation refers to the Fourier-transformed functions of time and  is the Fourier dual variable (frequency). Applying the inverse Fourier transform on the left and right hand sides of equation A11 now gives the following differential form for Eqs. A9-10:

 (A12)

where the dot operator refers to the temporal derivative . Other forms of the convolution kernel basically yield different ordinary differential equations (cf. Table A1).

|  |  |
| --- | --- |
| impulse response function | ODE |
|  |  |
|  |  |
|  |  |

**Table A1**. Examples of ODEs induced by different impulse response functions.

When integrated over time (cf. next section), ODEs yield the temporal dynamics of , given inputs . Importantly, convolution and ODE formulations are exactly equivalent. In particular, this implies that the instantaneous relationship between the motion of  and the inputs  (implicit in the ODE formulation) captures the fact that  is actually dependent upon the history of past inputs (implicit in the convolution formulation). We have used this equivalence in the last section of the manuscript, when deriving the "augmented" Q-learning model (cf. first line of Table A1).

**From continuous to discrete dynamical systems**

Most continuous dynamical systems can be described using ODEs, with the following normal form:

 (A13)

where  are the system's states,  are some arbitrary exogenous inputs that drive the system's states, and  returns the system's motion  given its current state . The ODE in Equation A13 can also be written using the following integral formulation:

 (A14)

where  is the discretization step. Note that Equation A14 may not be a convolution operation. Equation A14 can be converted into a finite difference equation using local linearization, as follows:

 (A15)

where  now indexes discrete time steps,  is the identity matrix and  is the system's Jacobian (evaluated at the current state ). Critically, when the discretization step  is small compared to the inverse of the highest eigenvalue of , one has: . Inserting this limit into Eq. A15 yields the so-called Euler discretization scheme, which is accurate whenever  is orders of magnitude smaller than the quickest decay time of the system. We have used this in the section "Getting closer to continuous dynamical systems" of the main manuscript, when introducing the "microtime resolution" of dynamical systems in the VBA toolbox.

**Volterra decomposition**

Most dynamical systems can be described in terms of input-output relationships, where the output  is typically a function of the history of past inputs  to the system. Volterra series allow a decomposition of the input-output transformation, as follows:

 (A16)

where we have chosen a discrete-time formulation. Here,  is some arbitrary time lag and we have truncated the series (up to second order). Equation A16 essentially derives from a recursive Taylor expansion of Equation A14. First-order Volterra kernels  capture the linear weight of lagged inputs onto the output. First-order Volterra kernels would be equivalent to impulse response functions, would the system be linear (cf. Equation A9 in continuous time). Second-order Volterra kernels  capture additional hysteresis effects, whereby the systematic response to some past input depends upon the input it received at some other point in time. For example, systems exhibiting "refractory periods" can be captured using second-order Volterra kernels that cancel the first-order impact of inputs during recovery time.

When performing dynamical systems' inversion, the VBA toolbox computes the first-order Volterra kernels of the sampled system's observable outputs and estimated hidden states and observables, w.r.t. to all inputs. The estimation is performed by the function ‘VBA\_VolterraKernels.m’, which essentially uses the VBA inversion of Equation A16, given some maximum lag (set in *options.kernelSize*; default=16). Note that one can perform a Volterra decomposition of the system w.r.t. any arbitrary set of inputs by calling ‘VBA\_VolterraKernels.m’, having appropriately reset the inputs in *out.u*. Replacing the estimated kernels in *out.diagnostics.kernels* allows one to eyeball the new Volterra decomposition from the "kernels" tab (cf. ‘VBA\_ReDisplay.m’). Importantly, fit accuracy metrics (coefficient of determination in the continuous case, and balanced classification accuracy in the binomial case) allows one to evaluate whether the truncated Volterra series faithfully captures the input-output relationship.