VS 265: Neural Computation **Fall 2018**

Simulating differential equations

Redwood Center for Theoretical Neuroscience

1 Discrete-time systems

- In most real physical systems, such as neurons, time is continuous. Thus, we use mathematical constructs such as $\int () dt$ and $\frac{d}{dt}()$ to represent summation and differences over continuous time, which is infinitely divisible.

- If we wish to simulate such systems on a digital computer, then we have no choice but to discretize time. Note however that this is *not* true of all computation in general. Analog computers, which pre-dated the digital computers we have today, serve as very useful simulation tools for studying complex dynamical systems using op-amps and capacitors without the need to discretize time.

- In the digital computer, we represent the continuous-time signal, $x(t)$, by sampling at discrete points in time:

  $$X(n) = x(n\Delta t)$$

  where Δt is the sampling interval and n is the sample number (an integer). Thus, $X(n)$ represents a sample of $x(t)$ at time $t = n\Delta t$. Note that the sampling interval Δt must be picked sufficiently small so as to capture the significant time-varying structure in $x(t)$, otherwise *aliasing* will result (this is what makes wagon wheels and propellers look like they're sometimes moving backwards in movies).

2 Difference equations

- A discrete-time approximation to the derivative is computed by taking the difference between adjacent samples divided by the sampling interval:

  $$\frac{dx}{dt} \approx \frac{X(n+1) - X(n)}{\Delta t}$$

  In the limit as $\Delta t \to 0$, this relationship becomes an equality (by definition).

- Now lets say we wish to simulate the differential equation $\dot{x} + x = 0$. Re-expressing this as a discrete-time *difference equation*, we have

  $$\frac{X(n+1) - X(n)}{\Delta t} + X(n) = 0$$

  With a little algebraic manipulation of terms, we obtain

  $$X(n+1) = (1 - \Delta t)X(n)$$

Thus, to simulate this system on a computer, we simply run a loop for $n = 1 : n_{max}$ and set X to a fraction of $(1 - \Delta t)$ its previous value at each iteration. Note however that if we pick Δt too large, then the system will not decay to zero but rather explode to $-\infty$. In this case, we must have $\Delta t < 1$ since $\tau = 1$.

- In a similar fashion, we can simulate the leaky-integrator with time-constant τ

$$\tau \dot{x} + x = f(t)$$

via the difference equation

$$\frac{X(n+1) - X(n)}{\Delta t} + X(n) = F(n)$$

which results in the discete-time equation

$$X(n+1) = (1 - \alpha)X(n) + \alpha F(n)$$

where $\alpha = \frac{\Delta t}{\tau}$ and $F(n) = f(n\Delta t)$. Note again though that in order for this simulation to work we must have $0 < \alpha < 1$, and so Δt must be picked to be small relative to τ.

- The discrete-time version of the leaky integrator gives us another perspective on what it is computing. Here we see that at each time step, the next value of X is a weighted sum of the current value of X and the current value of the input F. The weights that are used to combine X and F add to one. Thus, if $\alpha = 0.1$ then the next value of X is 90% of its current value plus 10% of the current value of F. It is easy to show that this recursive computation is equivalent to taking an expontentially decaying weighted sum of the present and past values of F.

- This method of simulating a differential equation is known as *Euler's method*. It is by far the simplest method of simulating a differential equation. Its disadvantage though is that it only crudely approximates the derivative, and so Δt must be picked very small to obtain accurate simulations. A small Δt means that many iterations are required, which demands more time. More efficient methods for simulating differential equations, such as the *Runge-Kutta method*, achieve the same degree of accuracy with larger time steps and hence fewer iterations.