Lecture 4: Discrete-Time State-Space Models of Dynamical Systems

Many physical (e.g., mechanical) systems evolve in continuous-time, however state estimation and control algorithms are ultimately implemented on digital computers where sensors sample the data at discrete time intervals. For this reason *discrete system* models or hybrid *sampled-data system* models are convenient to use. Discrete system models are also useful for saving computational resources on low-power computer systems that would be overburdened by numerically integrating the continuous dynamics. In a discrete-time system model the state evolves in discrete jumps with intervals Δt in between. Often Δt is a constant and each instant is denoted by a time-step integer k. Beginning with the initial state $\boldsymbol{x}_0$ at time t_0 (time-step $k = 0$) the next discrete state is $\boldsymbol{x}_1$ at time $t_1 = t_0 + \Delta t$ (time-step $k = 1$) and so on to give a set of states:

$$\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \cdots \tag{1}$$

at corresponding times

$$t_0, t_1, t_2, t_3, \cdots \tag{2}$$

where $t_k = k\Delta t$. In this lecture, we will discuss discrete-time models for continuous-time LTV, LTI, and nonlinear systems.

Discretizing LTV Systems [Rugh, 1996, Ch. 20]

Recall that for a linear-time-varying (LTV) system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t) \tag{3}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}(t)\boldsymbol{x}(t) \tag{4}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{5}$$

the solution for a non-zero input is given by the variation of constants formula

$$\boldsymbol{x}(t) = \boldsymbol{\Phi}(t, t_0)\boldsymbol{x}_0 + \int_{t_0}^{t} \boldsymbol{\Phi}(t, \sigma)\boldsymbol{B}(\sigma)\boldsymbol{u}(\sigma)d\sigma \tag{6}$$

When discretizing the LTV system we must model the value of the control input between time-steps. The simplest model (and also an accurate reflection of many digital systems) is that the control input remains constant during each timestep: $\boldsymbol{u}(t) = \boldsymbol{u}_{k-1}$ for all $t_{k-1} \leq t \leq t_k$, known as a *zero-order hold* for the control input. Then the discretized LTV system is obtained by modifying (6)with the appropriate start time $t_0 = t_{k-1}$ and end time $t = t_k$ and holding the control as a constant during the integration interval:

$$\boldsymbol{x}(t_k) = \boldsymbol{\Phi}(t_k, t_{k-1})\boldsymbol{x}(t_{k-1}) + \int_{t_{k-1}}^{t_k} \boldsymbol{\Phi}(t_k, \sigma)\boldsymbol{B}(\sigma)\boldsymbol{u}(t_{k-1})d\sigma \tag{7}$$

$$\boldsymbol{x}_k = \underbrace{\boldsymbol{\Phi}(t_k, t_{k-1})}_{\boldsymbol{F}_{k-1}}\boldsymbol{x}_{k-1} + \underbrace{\int_{t_{k-1}}^{t_k} \boldsymbol{\Phi}(t_k, \sigma)\boldsymbol{B}(\sigma)d\sigma}_{\boldsymbol{G}_{k-1}}\,\boldsymbol{u}_{k-1} \tag{8}$$

$$\implies \boldsymbol{x}_k = \boldsymbol{F}_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{G}_{k-1}\boldsymbol{u}_{k-1} \tag{9}$$

and similarly for the measurement equation

$$\boldsymbol{y}(t_k) = \underbrace{\boldsymbol{C}(t_k)}_{\boldsymbol{H}_k} \boldsymbol{x}(t_k) \tag{10}$$

$$\implies \boldsymbol{y}_k = \boldsymbol{H}_k \boldsymbol{x}_k \tag{11}$$

Notice that (9) looks very similar to a standard linear system equation but it does not include an $\dot{x}$ term. This is not an ODE! It is a *finite difference equation* that algebraically relates the previous state and control, $\boldsymbol{x}_{k-1}$ and $\boldsymbol{u}_{k-1}$ at step t_{k-1} to the new state at time t_k. Given an initial condition $\boldsymbol{x}_0$ and a control input history $\boldsymbol{u}_0, \boldsymbol{u}_1, \boldsymbol{u}_2, \ldots$ we can recursively use (9) and (11) to give the corresponding state history $\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ and output history $\boldsymbol{y}_0, \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots$ Since the system is time-varying the matrices $\boldsymbol{F}_{k-1}$, $\boldsymbol{G}_{k-1}$, and $\boldsymbol{H}_k$ will also change with each discrete step k.

Discretizing LTI Systems [Simon, 2006, Sec. 1.4]

In the case that our system is linear time-invariant (LTI)

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \tag{12}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) \tag{13}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{14}$$

Recall that an LTI system has solution give by the variation of constants formula:

$$\boldsymbol{x}(t) = e^{\boldsymbol{A}(t-t_0)}\boldsymbol{x}_0 + \int_{t_0}^{t} e^{\boldsymbol{A}(t-\sigma)}\boldsymbol{B}\boldsymbol{u}(\sigma)d\sigma \tag{15}$$

As before, modifying (6)with the appropriate start time $t_0 = t_{k-1}$ and end time $t = t_k$ with a fixed sampling time $\Delta t_k - t_{k-1}$ and holding the control as a constant:

$$\boldsymbol{x}_k = e^{\boldsymbol{A}(t_k - t_{k-1})}\boldsymbol{x}_{k-1} + \int_{t_{k-1}}^{t_k} e^{\boldsymbol{A}(t_k-\sigma)}\boldsymbol{B}\boldsymbol{u}_{k-1}d\sigma \tag{16}$$

$$= e^{\boldsymbol{A}\Delta t}\boldsymbol{x}_{k-1} + \int_{t_{k-1}}^{t_k} e^{\boldsymbol{A}(t_k-\sigma)}d\sigma\boldsymbol{B}\boldsymbol{u}_{k-1} \tag{17}$$

To simplify the above expression, define $\alpha = \sigma - t_{k-1}$ so that $d\alpha = d\sigma$ and the integration bounds change as

$$\boldsymbol{x}_k = e^{\boldsymbol{A}\Delta t}\boldsymbol{x}_{k-1} + \int_{0}^{t_k - t_{k-1}} e^{\boldsymbol{A}(t_k-(\alpha+t_{k-1}))}d\alpha\boldsymbol{B}\boldsymbol{u}_{k-1} \tag{18}$$

$$= e^{\boldsymbol{A}\Delta t}\boldsymbol{x}_{k-1} + \int_{0}^{\Delta t} e^{\boldsymbol{A}\Delta t}e^{-\boldsymbol{A}\alpha}d\alpha\boldsymbol{B}\boldsymbol{u}_{k-1} \tag{19}$$

$$= e^{\boldsymbol{A}\Delta t}\boldsymbol{x}_{k-1} + e^{\boldsymbol{A}\Delta t}\left(\int_{0}^{\Delta t} e^{-\boldsymbol{A}\alpha}d\alpha\right)\boldsymbol{B}\boldsymbol{u}_{k-1} \tag{20}$$

Thus, the discretized system has the form

$$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{G}\boldsymbol{u}_{k-1} \tag{21}$$

$$\boldsymbol{y}_k = \boldsymbol{H}\boldsymbol{x}_k \tag{22}$$

where

$$\boldsymbol{F} = e^{\boldsymbol{A}\Delta t} \tag{23}$$

$$\boldsymbol{G} = e^{\boldsymbol{A}\Delta t} \left(\int_0^{\Delta t} e^{-\boldsymbol{A}\alpha} d\alpha \right) \boldsymbol{B} \tag{24}$$

Notice that for an LTI system the matrices $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{H}$ are all constants and do not vary with the time step. Since the measurement equation is just a transformation of the state vector it is equal to the continuous case, $\boldsymbol{H} = \boldsymbol{C}$. The situation can be even further simplified if the $\boldsymbol{A}$ matrix is invertible. The integral in (20) is equivalent to

$$\int_0^{\Delta t} e^{-\boldsymbol{A}\alpha} d\alpha = \int_0^{\Delta t} \sum_{k=1}^{\infty} \frac{(-\boldsymbol{A}\alpha)^k}{k!} d\alpha \tag{25}$$

$$= \int_0^{\Delta t} \left[\boldsymbol{I} - \boldsymbol{A}\alpha + \boldsymbol{A}^2 \frac{\alpha^2}{2!} - \cdots \right] d\alpha \tag{26}$$

$$= \left[\boldsymbol{I}\alpha - \boldsymbol{A}\frac{\alpha^2}{2} + \boldsymbol{A}^2 \frac{\alpha^3}{3!} - \cdots \right]_0^{\Delta t} \tag{27}$$

Now premultiply all the terms inside the square brackets by $\boldsymbol{A}$ but also postmultiply $\boldsymbol{A}^{-1}$ outside the square brackets to leave the quantity unchanged

$$\int_0^{\Delta t} e^{-\boldsymbol{A}\alpha} d\alpha = \left[\boldsymbol{A}\alpha - \boldsymbol{A}^2 \frac{\alpha^2}{2} + \boldsymbol{A}^3 \frac{\alpha^3}{3!} - \cdots \right]_0^{\Delta t} \boldsymbol{A}^{-1} \tag{28}$$

Next, add and subtract $\boldsymbol{I}$ inside the square brackets and recognize the expression for $e^{-\boldsymbol{A}\Delta t}$:

$$\int_0^{\Delta t} e^{-\boldsymbol{A}\alpha} d\alpha = \left[\boldsymbol{I} - \boldsymbol{I} + \boldsymbol{A}\alpha - \boldsymbol{A}^2 \frac{\alpha^2}{2} + \boldsymbol{A}^3 \frac{\alpha^3}{3!} - \cdots \right]_0^{\Delta t} \boldsymbol{A}^{-1} \tag{29}$$

$$\int_0^{\Delta t} e^{-\boldsymbol{A}\alpha} d\alpha = \left[\boldsymbol{I} - \left\{ \boldsymbol{I} - \boldsymbol{A}\alpha + \boldsymbol{A}^2 \frac{\alpha^2}{2} - \boldsymbol{A}^3 \frac{\alpha^3}{3!} - \cdots \right\}_0^{\Delta t} \right] \boldsymbol{A}^{-1} \tag{30}$$

$$\int_0^{\Delta t} e^{-\boldsymbol{A}\alpha} d\alpha = \left[\boldsymbol{I} - e^{-\boldsymbol{A}\Delta t} \right] \boldsymbol{A}^{-1} \tag{31}$$

so that we arrive at the simplified form:

$$\implies \boldsymbol{G} = \boldsymbol{F} \left[\boldsymbol{I} - e^{-\boldsymbol{A}\Delta t} \right] \boldsymbol{A}^{-1} \boldsymbol{B} \tag{32}$$

Example. Consider the following system

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{33}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \tag{34}$$

The matrix $\boldsymbol{A}$ is invertible, so we can use (32) but it is not easy to compute the matrix exponential required by hand. We can turn to MATLAB for this numerical example:

```
A = [0 1; -2 -3];
B = [0; 1];
dt = 0.1;
F = expm(A*dt)
G = F*(eye(2,2) - expm(-A*dt))*inv(A)*B
```

gives the discrete-time system

$$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{G}\boldsymbol{u}_{k-1} \tag{35}$$

$$\boldsymbol{x}_k = \begin{bmatrix} 0.9909 & 0.0861 \\ -0.1722 & 0.7326 \end{bmatrix} \boldsymbol{x}_{k-1} + \begin{bmatrix} 0.0045 \\ 0.0861 \end{bmatrix} u_{k-1} \tag{36}$$

Note that MATLAB has a built-in function `c2d` that converts from continuous to discrete time. The matrices $\boldsymbol{F}$ and $\boldsymbol{G}$ can also be determined from:

```
sysc = ss(A,B,[],[]);  % continuous system
sysd = c2d(sysc, dt); % discrete system

F = sysd.A % rename matrices
G = sysd.B
```

Approximate Discretization for LTI Systems. The above transformations are exact in the sense that they precisely give the state and measurement evolution equivalent to the continuous system but at discrete points. No error is introduced in the process. However, in some cases it may be convenient to approximate the discretization (e.g., if the discretization is over a sufficiently small time-step). An approximate discretization can be obtained by truncating the matrix exponential (e.g., leaving only the n terms). Keeping only the first two terms leads to the approximate system dynamics

$$\boldsymbol{F} = e^{\boldsymbol{A}\Delta t} \tag{37}$$

$$\implies \boldsymbol{F} \approx \boldsymbol{I} + \boldsymbol{A}\Delta t \tag{38}$$

$$\boldsymbol{G} = e^{\boldsymbol{A}\Delta t}\left(\int_0^{\Delta t} e^{-\boldsymbol{A}\alpha} d\alpha\right)\boldsymbol{B} \tag{39}$$

$$\approx (\boldsymbol{I} + \boldsymbol{A}\Delta t)\left(\int_0^{\Delta t} (\boldsymbol{I} - \boldsymbol{A}\alpha) d\alpha\right)\boldsymbol{B} \tag{40}$$

$$= (\boldsymbol{I} + \boldsymbol{A}\Delta t)(\boldsymbol{I}\Delta t - \boldsymbol{A}(\Delta t^2/2))\boldsymbol{B} \tag{41}$$

$$= \boldsymbol{B}\Delta t - \boldsymbol{A}\boldsymbol{B}(\Delta t^2/2) + \boldsymbol{A}\boldsymbol{B}\Delta t^2 - \boldsymbol{A}^2\boldsymbol{B}(\Delta t^3/2) \tag{42}$$

$$\implies \boldsymbol{G} \approx \boldsymbol{B}\Delta t \tag{43}$$

where higher order terms are ignored in the last step computing the matrix $\boldsymbol{G}$.

Example. Continuing the above example, we can approximate the discretization with

```
F = eye(2,2) + A*dt
G = B*dt
```

to give

$$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{G}\boldsymbol{u}_{k-1} \tag{44}$$

$$\boldsymbol{x}_k = \begin{bmatrix} 1.0000 & 0.1000 \\ -0.2000 & 0.7000 \end{bmatrix} \boldsymbol{x}_{k-1} + \begin{bmatrix} 0 \\ 0.1000 \end{bmatrix} u_{k-1} \tag{45}$$

Notice the similarity of this result with (36). As Δt grows small these two systems become increasingly similar.

Discretizing Nonlinear Systems

Discretizing a generic nonlinear systems

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{46}$$

$$\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t)) \tag{47}$$

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{48}$$

is difficult unless the particular system is "easy" enough to compute the following integral:

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \int_{t_{k-1}}^{t_k} \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \tag{49}$$

and the measurements require no discretization with $\boldsymbol{y}(t) = \boldsymbol{h}(\boldsymbol{x}(t))$ becoming:

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k) \tag{50}$$

The new discrete system above can then be summarized by a finite-difference equation as:

$$\boldsymbol{x}_k = \boldsymbol{f}^d_{k-1}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) \tag{51}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k) \tag{52}$$

where we've introduced

$$\boldsymbol{f}^d_{k-1}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) = \boldsymbol{x}_{k-1} + \int_{t_{k-1}}^{t_k} \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))dt \tag{53}$$

to represent the discrete-time dynamics. In the vein of (38) we could also approximate the discretization using any numerical integration scheme (e.g., Runge-Kutta or Euler's method). We'll discuss numerical integration in an upcoming lecture, but you may already be familiar with Euler's method which would approximate the integration as follows:

$$\boldsymbol{x}_k \approx \boldsymbol{x}_{k-1} + \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1})\Delta t = \boldsymbol{f}^d(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) \tag{54}$$

Example. Consider the following model of a mobile robot called the *Dubins car*:

$$\dot{x} = v\cos\theta \tag{55}$$
$$\dot{y} = v\sin\theta \tag{56}$$
$$\dot{\theta} = u \tag{57}$$

where (x, y) is the planar position, θ is the orientation, v is a speed control, and u is a turn-rate control. This system is continuous and nonlinear, but for a constant speed and turn-rate it can be discretized exactly. The state of the system at time t_{k-1} is x_{k-1}, y_{k-1}, and θ_{k-1}. Notice that the third equation (for heading control) is independent of the others and can be integrated seperately:

$$d\theta = udt \tag{58}$$
$$\int_{\theta_{k-1}}^{\theta(t)} d\theta = \int_{t_{k-1}}^{t} udt \tag{59}$$
$$\theta(t) = \theta_{k-1} + u(t - t_{k-1}) \tag{60}$$

Using this expression for $\theta(t)$ we can also make the RHS of equations (55) and (56) decoupled from θ and depend only on time. Multiply the $\dot{x}$ equation by $(d\theta/dt)^{-1} = 1/u$

$$\frac{dx}{dt}\left(\frac{d\theta}{dt}\right)^{-1} = v\cos\theta\frac{1}{u} \tag{61}$$
$$\frac{dx}{d\theta} = \frac{v}{u}\cos\theta \tag{62}$$

where now θ is the dependent variable. Integrating this expression and substituting (60):

$$\int_{x_{k-1}}^{x(t)} dx = \int_{\theta_{k-1}}^{\theta(t)} \frac{v}{u}\cos\theta d\theta \tag{63}$$
$$x(t) = x_{k-1} + \frac{v}{u}\Big[\sin\theta\Big]_{\theta_{k-1}}^{\theta_{k-1}+u(t-t_{k-1})} \tag{64}$$
$$x(t) = x_{k-1} + \frac{v}{u}\left(\sin(\theta_{k-1} + u(t - t_{k-1})) - \sin\theta_{k-1}\right) \tag{65}$$

and similarly for $y(t)$

$$\int_{y_{k-1}}^{y(t)} dx = \int_{\theta_{k-1}}^{\theta(t)} \frac{v}{u}\sin\theta d\theta \tag{66}$$
$$y(t) = y_{k-1} + \frac{v}{u}\Big[-\cos\theta\Big]_{\theta_{k-1}}^{\theta_{k-1}+u(t-t_{k-1})} \tag{67}$$
$$y(t) = y_{k-1} + \frac{v}{u}\left(-\cos(\theta_{k-1} + u(t - t_{k-1})) + \cos\theta_{k-1}\right) \tag{68}$$

Suppose the speed and turn-rate is constant over the interval t_{k-1} to t_k with $u = u_{k-1}$ and $v = v_{k-1}$. Then, evaluating (60), (65), (68) at time t_k and assuming that the time-step is constant (i.e., $\Delta t = t_k - t_{k-1}$) we obtain a nonlinear discrete-time system:

$$x_k = x_{k-1} + \frac{v_{k-1}}{u_{k-1}} \left(\sin(\theta_{k-1} + u_{k-1}\Delta t) - \sin\theta_{k-1}\right) \tag{69}$$

$$y_k = y_{k-1} + \frac{v_{k-1}}{u_{k-1}} \left(-\cos(\theta_{k-1} + u_{k-1}\Delta t) + \cos\theta_{k-1}\right) \tag{70}$$

$$\theta_k = \theta_{k-1} + u_{k-1}\Delta t \tag{71}$$

It is important to emphasize that, in this case, the discretized system above exactly matches the continuous-time solution (at the discrete points) under the same assumptions of the control inputs and initial conditions. On the other hand, an approximate discretization is, for example (using Euler's method):

$$x_k = x_{k-1} + v_{k-1}\cos\theta_{k-1}\Delta t \tag{72}$$

$$y_k = y_{k-1} + v_{k-1}\sin\theta_{k-1}\Delta t \tag{73}$$

$$\theta_k = \theta_{k-1} + u_{k-1}\Delta t \tag{74}$$

Notice that this approach entirely neglects the turning motion as it updates x_k and y_k (i.e., it approximates the update by straight line motion). This approach introduces errors that will accrue over time.

Inherently Discrete Systems

While much of our emphasis above has been on converting continuous systems into discrete ones, some systems are inherently discrete to begin with. For example, communication networks are often represented by nodes that exchange packets of information at discrete time intervals, or digital computer often implement systems that evolve in discrete jumps rather than continuously in time. Such systems are already in discrete form.

Example (Consensus Network, [Olfati-Saber et al., 2007]). The term *consensus* refers to the ability of a multi-agent system to reach an agreement, usually regarding the value of some mathematical quantity. In the simplest case we could consider a series of processor nodes (we'll call them *agents*) that want to agree on the value of single scalar α — perhaps this scalar represents some important parameter of their operation (in robotics, it might represent the distance along a path at which robots should meet or a time at which robots should rendezvous). In any case, we assume there are N agents and each of them begins with some proposed value for α that can be arranged into a single state vector, such as:

$$\boldsymbol{x}_i(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \\ \vdots \\ x_N(0) \end{bmatrix} \tag{75}$$

where $x_i(k)$ denotes the ith agents proposed state at time-step k. If all of the agents are connected to one another they could simply share their proposed values and agree on one using a rule (e.g., taking the average). However, networked systems may not be fully connected and, instead, we may have agent i being able to communicate with agent j but not with agent k and so on. To describe the *topology* of the network we can introduce a parameter a_{ij} that is equal to 1 if agent i and j can communicate and equal to 0 otherwise. (The matrix of a_{ij} values is called the *adjacency matrix* and the topology is described by a *graph*.) Then, at each discrete time increment, the agents exchange their proposed values with one another and adjust their own proposed value (for the next time step) according to a weighted difference of the values they received with their own proposal

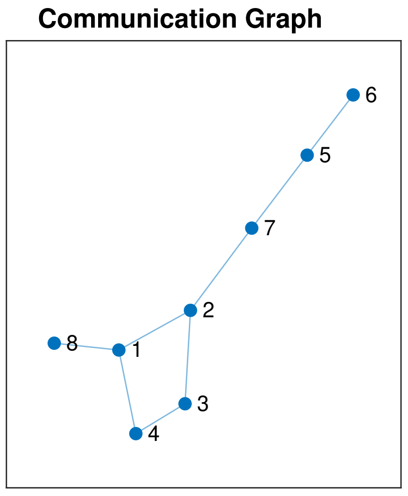
$$x_i(k) = x_i(k-1) + \epsilon \sum_{j \in \mathcal{N}_i} a_{ij}(x_j(k-1) - x_i(k-1)) \tag{76}$$

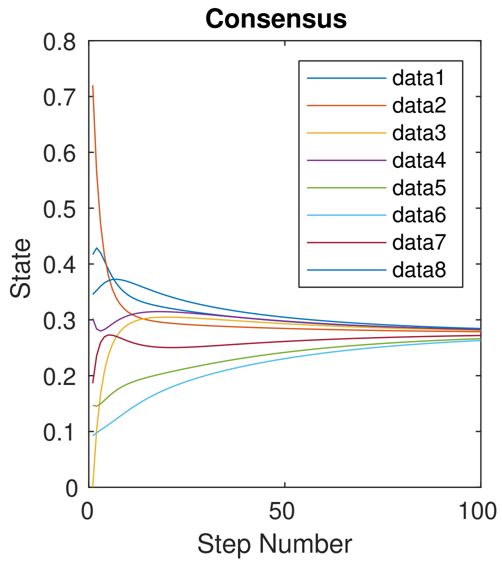
where ϵ is a weighting parameter. The above expression can be rearranged as a discrete-time equation

$$\boldsymbol{x}_k = \boldsymbol{P}\boldsymbol{x}_{k-1} \tag{77}$$

where $\boldsymbol{P} = \boldsymbol{I} - \epsilon\boldsymbol{L}$ where $\boldsymbol{L}$ is a $N \times N$ matrix called the *graph Laplacian* that describes the communication topology of the network (i.e., it is related to the a_{ij} values). For a graph that is *connected* (i.e., there exists a path for information to flow from each agent to every other agent) the system will reach consensus and converge to a single value that depends on the initial conditions.

This idea is illustrated below with 8 agents forming a connected graph with each proposing a random initial scalar value from zero to one. The simulation proceeds in 100 steps and at each step the agent only has access to information provided by agents with whom they are directly connected to (i.e., the neighbors of the agent).





```
% A graph where pairs (s,t) are start/tail ends of undirected edges
s = [1 1 1 2 2 3 5 5];
t = [2 4 8 3 7 4 7 6];
G = graph(s,t);
```

```
A = full(adjacency(G)) % adjacency matrix
D = diag(degree(G))% degree matrix
L = D - A % graph Laplacian
n = 8; % number of nodes
rng(1); % make deterministic
Nsteps = 100;
eps = 0.1;
x(:,1) = rand(8,1); % initial proposal
P = eye(n,n) - eps*L;
for k = 2:1:Nsteps
    x(:,k) = P*x(:,k-1);
end
```

References

[Olfati-Saber et al., 2007] Olfati-Saber, R., Fax, J. A., and Murray, R. M. (2007). Consensus and Cooperation in Networked Multi-Agent Systems. *Proceedings of the IEEE*, 95(1):1–7.

[Rugh, 1996] Rugh, W. J. (1996). *Linear System Theory*. Prentice-Hall, Inc.

[Simon, 2006] Simon, D. (2006). *Optimal State Estimation: Kalman, H infinity, and Nonlinear Approaches*. John Wiley & Sons.