Ordinary Differential Equations and Dynamical Systems

ME 416 - Prof. Tron

Thursday 21st March, 2019

1 Explicit first-order autonomous Ordinary Differental Equations (ODEs)

Definition Given a function $x(t)$ of a single *independent variable* t, an *explicit first-order autonomous Ordinary Differental Equations (ODE)* is of the form

$$\dot{x}(t) = f(x(t)) \tag{1}$$

A *solution* to an ODE is a parametrized curve $x(t)$ satisfying the equality in the equation. In order to select a single solution, it is necessary to speficy *initial conditions* $x(0) = x_0$.

Often, the dependence on independent variable is not stated outright, writing $\dot{x} = f(x)$

All the equations we will see in this class are, or can be reduced to, explicit first-order autonomous ODEs, so we will omit the qualifying adjectives.

In robotics, x typically represents the *state* of a robot, which, in general, can be a vector (position), a rotation, or a pose.

Example consider a reference frame $\mathcal{B}$ that rotates at a constant speed with respect to a world reference frame $\mathcal{W}$, i.e., ${}^{\mathcal{W}}R_{\mathcal{B}} = \begin{bmatrix} \cos(\omega t) & -\sin(\omega t) \\ \sin(\omega t) & \cos(\omega t) \end{bmatrix}$, with ${}^{\mathcal{W}}T_{\mathcal{B}} = 0$; consider a point ${}^{\mathcal{B}}x$ with constant coordinates. From the rigid body transformation equation we have ${}^{\mathcal{W}}x = {}^{\mathcal{W}}R_{\mathcal{B}}\,{}^{\mathcal{B}}x$, which correspond to circular trajectories. This motion can also be re-interpreted as the solution to an ODE. Taking the derivative of ${}^{\mathcal{W}}x$ we have:

$${}^{\mathcal{W}}\dot{x} = {}^{\mathcal{W}}\dot{R}_{\mathcal{B}}\,{}^{\mathcal{B}}x = \omega \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} {}^{\mathcal{W}}R_{\mathcal{B}}\,{}^{\mathcal{B}}x = \omega \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} {}^{\mathcal{W}}x, \tag{2}$$

TODO: example 2-D constant velocity linear motion

★★★TODO: discussion on the existence and uniqueness of the solutions

1.1 ★★ *Graphical interpretation of an ODE*

It is instructive to consider the case where $x \in \mathbb{R}^2$. In this case, f is a function that maps from $\mathbb{R}^2$ to $\mathbb{R}^2$, or, more accurately, from points to vectors. We can therefore represent f as a *vector field*, that is, by drawing an arrow for the vector $f(x)$ at the corresponding point x. You can imagine building a solution $x(t)$ by considering the field as a water current, leaving a small boat at the initial condition x_0, and then recording its path.

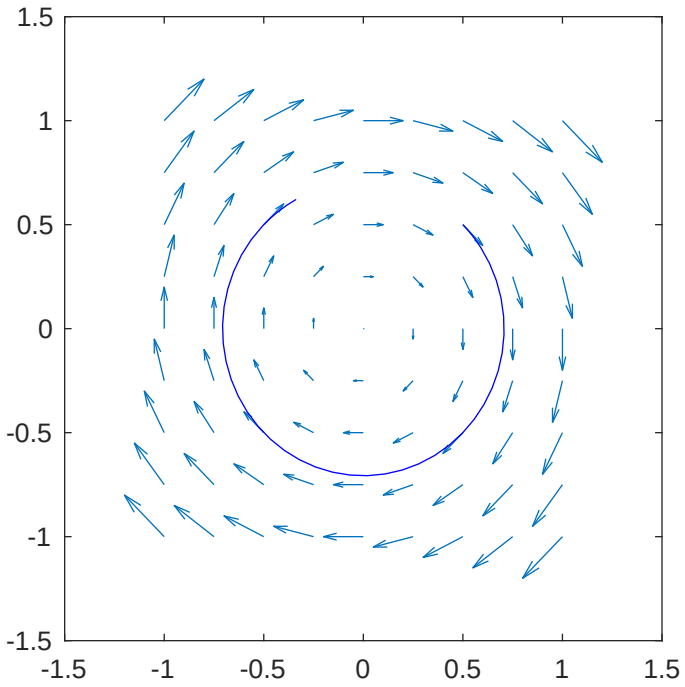

Figure 1: Field $f(x) = -\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} x$ and solution for $x(0) = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$, $t \in [0, 5]$.

Note: Recall that $\dot{x}$, when x is a vector, can be interpreted as a vector tangent to the parametric curve $x(t)$. Hence, geometrically, what the ODE $\dot{x} = f(x)$ is really stating is that tangent to the curve must be equal to $f(x)$ at every point.

Example (continued): The ODE obtained from the purely rotational motion in (2) can be written as the ODE ${}^{\mathcal{W}}\dot{x} = f({}^{\mathcal{W}}x)$, where the field $f(x)$ is given by $f(x) = \omega \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} {}^{\mathcal{W}}x$. This field is represented by the arrows in Figure 1 (for the case $\omega = 1$).

A solution ${}^{\mathcal{W}}x$ corresponding to ${}^{\mathcal{W}}x(0) = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$, $t \in [0, 5]$ is also shown in the same figure. Note how the arrows are always tangent to the curve.

1.2 ★★Explanation of the adjectives

- Explicit: $\dot{x}$ appears outside of $f()$. The alternative is an *implicit* equation, which has the form $f(\dot{x}, x) = 0$.
- First-order: the equation contains only derivatives of the first order (i.e., there is no $\ddot{x}$, $\dddot{x}$, etc.).
- Autonomous: $f(\cdot)$ does not depend explicitly on time. The alternative is a *non-autonoumous* equation, which as the form $\dot{x} = f(t, x)$.
- Ordinary: there is only a single *independent variable*. The alternative is a *partial differential equation*, where the state x depends on $t_1, t_2, \ldots$.

In addition, an ODE is *linear* if $f()$ is a linear function.

TODO: Level *: offer alternative characterization in terms of sum of solutions and superposition principle

1.3 ★★Reduction of second-order ODEs to first order

In general, given an explicit second-order ODE, e.g., of the form

$$\ddot{x} + \dot{x} = f(x), \tag{3}$$

it is possible to write it as a first-order ODE by "packing" the derivatives of the different orders in a new independent variable. For instance, we can rewrite (3) as a first order ODE by introducing $v = \dot{x}$, so that (3) becomes $\dot{v} = -\dot{x} + f(x) = -v + f(x)$.

Considering the entire state $z = \begin{bmatrix} x \\ v \end{bmatrix}$, the resulting ODE is then

$$\dot{z} = \begin{bmatrix} \dot{x} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ -v + f(x) \end{bmatrix}, \tag{4}$$

which is first-order, albeit with double the number of variables.

TODO: example: mass under gravity

2 Numerical integration of ODEs

For some classes of ODEs (linear ODEs are an important case), it is possible to find explicit, analytical solutions (see the Wikipedia page for a summary). When such solutions cannot be found, or are too computationally expensieve, one can find an *approximate solution* numerically. There are many different *ODE solvers* (methods) for this, offering different tradeoffs between computation requirements, accuracy of the solution, and types of problems they can be applied to.

At a high level, given the *continuous time* ODE $\dot{x}(t) = f(x(t))$ (this denomination is given because here we can evaluate the equation for any t), an ODE solver defines a discrete increasing sequence of times, a *grid*, $t_0 = 0, t_1, t_2, \ldots$, and returns a *discrete time* solution $x[k]$ that approximates the true continuous time solution $x(t)$. Here, k is an integer for the discretized time (notice the use of square brackets), and the approximation of the solution is intended in the sense that $x[k] \approx x(t_k)$.

There are a few things to keep in mind:

- A solution x[k] computed by a solver does not say anything for instants t that are not in the grid $t_0, t_1, t_2, \ldots$ (although it is always possible to interpolate).
- Since solvers compute solutions progressively (starting from t_0 and moving forward), as a rule of thumb, the accuracy of the solution reduces as t increases (i.e., the errors accumulate).
- As another rule of thumb, using a denser grid (i.e., selecting a larger number of instants t_k for a same period of time) increases the accuracy of the solution, at the expense of a higher computational load.

2.1 Euler method

Euler explicit method (a.k.a. forward Euler method) is the simplest ODE solver. The time grid is defined by picking a *step size* h and then letting $t_k = kh$ (i.e., $t_{k+1} = t_k + h$ and the time instants are regularly spaced).

Then, the solution is built incrementally following the iteration $x[k+1] = x[k] + hf(x[k])$.

Derivation from approximation of the derivative Assume that we have computed the solution up to t_k, i.e., we have $x[k] \approx x(kh)$. From the definition of the derivative, we have

$$\dot{x}(t) = \lim_{h \to \infty} \frac{x(t+h) - x(t)}{h} = f(x(t)), \tag{5}$$

where the last equality comes from the ODE.

We can approximate the derivative by approximating the limit with a small, but finite $h > 0$. Performing this approximation for $t = kh$ we have

$$\frac{x(kh+h) - x(kh)}{h} \approx f(x(kh)), \tag{6}$$

which, using the approximation $x[k] \approx x(kh)$, becomes

$$\frac{x((k+1)h) - x(kh)}{h} \approx \frac{x[k+1] - x[k]}{h} f(x[k])). \tag{7}$$

Rearranging, we obtain the Euler's method equation:

$$x[k+1] = x[k] + hf(x[k])). \tag{8}$$

Graphical interpretation Graphically, Euler's method can be intepreted as: starting from the solution $x[k]$ at the current step, compute the field $f()$ at that point, follow the arrow after scaling it by h, let this new point be the value for $x[k+1]$ at the successive time step, and then repeat. An example of the application of Euler's method for two different values of h is shown in Figure 2. Notice that both Euler's solution capture the main undulatory behavior of the solution, but the solution with $h = 0.5$ is much closer to the true solution.

2.2 Numerical stability

Euler's method can be numerically unstable, meaning that the numerical solution grows very large for equations where the exact solution does not. A typical way to counter this problem is to pick a smaller step size h.

TODO: show result on rotational field

3 Dynamical systems

Dynamical systems are defined by three elements:

State: A vector with all the quantities of interest of a system.

Inputs: What we can control to affect how the state evolves.

Model: Differential equation indicating how the state changes over time given the current state and inputs.

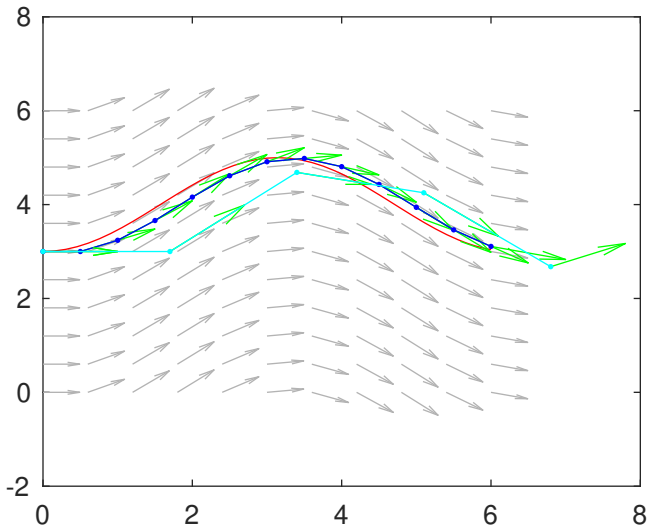

Figure 2: Example of application of Euler's method on the field $f(x) = \begin{bmatrix} 1 \\ \sin(x) \end{bmatrix}$. Gray arrow: field $f(x)$. Red: true solution. Blue and Cyan: Euler's approximate solution with $h = 0.5$ and $h = 1.7$. Green arrows: field $f(x)$ evaluated at the points in Euler's solutions.

Figure 3: Trivia: Euler's method is the method that saves the day in the movie "Hidden Figures". Credit: TM and (C) 2017 Twentieth Century Fox Film Corporation.

Example (differential drive robot) As we derived in class, the linear and angular velocity of a differential drive robot are governed by the equation:

$$\dot{z} = A(z)u, \tag{9}$$

where the three elements of the dynamical system are:

State: The vector $z = \begin{bmatrix} {}^{\mathcal{W}}T_{\mathcal{B}} \\ \theta \end{bmatrix}$.

Inputs: The vector $u = \begin{bmatrix} u_{LW} \\ u_{RW} \end{bmatrix}$.

Model: Equation (9), where $A(z)$ is the 3×2 matrix derived in another part of these notes.

★★TODO: talk about outputs.

3.1 Dynamical systems with open-loop control

Open loop control refers to the fact that the inputs as a function of time $u(t)$ are decided beforehand. By substituting $u(t)$ in the dynamical system equation, we obtain a (generally non-autonomous) ODE which we can integrate to see what is the resulting trajectory of the system $z(t)$. The integration of the ODE can be done either analytically (if possible) or approximately using Euler's method.

Since dynamical models are often idealization of real systems, open loop control can approximately maneuver the system on a desired trajectory, but it is not robust (i.e., the actual trajectory of the system will not be exactly the same as the desired one).

Example (differential drive robot) In one of the questions in Homework 2 you are asked to derive the trajectory $z(t)$ for inputs $u(t)$ which are constant. The odometry nodes in the other questions of Homework 2 compute $z(t)$ when $u(t)$ is piecewise constant.