Solving Sequence of ODE IVPs Using Explicit Euler Method

MATLAB function *ode45* is a variable step-size 4'th and 5'th order Runge Kutta ODE-IVP (Ordinary Differential Equation Initial Value Problem) solver. You may not be exposed to theoretical basis for this type of solver and may find some discomfort using it. However, it is likely that you are exposed to the explicit Euler method, which is the simplest method for solving ODE-IVP. This method is briefly reviewed here. You can choose to use this approach for solving the ODEs for your problem.

1 Simulation Problem Description

The simulation problem at hand is to solve the following sequence of ODE-IVPs

$$\begin{aligned} \frac{d\mathbf{X}}{dt} &= \mathbf{f}[\mathbf{X}(t), \mathbf{U}(k), D(k)] && (1)\\ kT &\leq t < (k+1)T && (2)\\ \mathbf{X}(kT) &= \mathbf{X}(k) : \text{Initial Condition} && (3)\\ k &= 0, 1, 2, ...N_s - 1 \text{and } T : \text{Sampling interval} && (4) \end{aligned}$$

Values of $\mathbf{U}(k)$ and $D(k)$ are held constant over each sampling interval $kT \leq t < (k+1)T$ where $k = 0, 1, 2, ...N_s - 1$. Given $\mathbf{U}(0)$ and $D(0)$ we start with initial condition $\mathbf{X}(0)$ and solve the ODE-IVP over interval $0 \leq t < T$ and arrive at $\mathbf{X}(T) = \mathbf{X}(1)$. Then, given $\mathbf{U}(1)$ and $D(1)$, we take $\mathbf{X}(T) = \mathbf{X}(1)$ as initial condition and solve the ODE-IVP over interval $T \leq t < 2T$ and arrive at $\mathbf{X}(\mathbf{2}T) = \mathbf{X}(2)$, and so on. Thus, we go hopping in time 0,T,2T,....where T represents the **sampling interval**. In the demo program uploaded on Moodle, this sequence of ODE-IVPs is solved using MATLAB function ode45.

2 Explicit Euler Method

To solve for

$$\begin{aligned} \frac{d\mathbf{X}}{dt} &= \mathbf{f}[\mathbf{X}(t), \mathbf{U}(k), D(k)] && (5)\\ kT &\leq t < (k+1)T && (6) \end{aligned}$$

we can assume that

$$\mathbf{f}[\mathbf{X}(t), \mathbf{U}(k), D(k)] \cong \mathbf{f}[\mathbf{X}(k), \mathbf{U}(k), D(k)] \text{ for } kT \leq t < (k+1)T \quad (7)$$

and also approximate LHS as

$$\frac{d\mathbf{X}}{dt} \cong \frac{\mathbf{X}((k+1)T) - \mathbf{X}(kT)}{T} = \frac{\mathbf{X}(k+1) - \mathbf{X}(k)}{T} \quad (8)$$

Combining these two approximations, we arrive at

$$\frac{\mathbf{X}(k+1) - \mathbf{X}(k)}{T} = \mathbf{f}[\mathbf{X}(k), \mathbf{U}(k), D(k)] \quad (9)$$

and

$$\begin{aligned} \mathbf{X}(k+1) &= \mathbf{X}(k) + T \times \mathbf{f}[\mathbf{X}(k), \mathbf{U}(k), D(k)] && (10)\\ k &= 0, 1, 2, ..., N_s - 1 && (11) \end{aligned}$$

Thus, to use this algorithm, simply replace code for ode45 solver, i.e.

[T,Xt]=ode45('QTank_ Dynamics',[0samp_ T],Xk(:,k));

Xk(:,k+1)=Xt(end,:)';

by the following

*Xk(:,k+1) = Xk(:,k) + samp_ T * QTank_ Dynamics(kT(k), Xk(:,k)) ;*

This simple approach works only when T is sufficiently small and assumption (7) holds. However, if derivative changes significantly in the in interval $kT \leq t < (k+1)T$, then we need to modify the computation scheme. Let is assume that the interval $kT \leq t < (k+1)T$ is further subdivided into smaller intervals as follows

$$\begin{aligned} kT &\leq t < kT + h \\ kT + h &\leq t < kT + 2h \\ &\ldots \\ kT + (n-1)h &\leq t < kT + nh = (k+1)T \end{aligned}$$

where $h = T/n$ (and $h << T$) represents **integration interval**. Now, we can assume that the derivative remains constant over

$$\mathbf{f}[\mathbf{X}(t), \mathbf{U}(k), D(k)] \cong \mathbf{f}[\mathbf{X}(kT + jh), \mathbf{U}(k), D(k)] \text{ for } kT + jh \leq t < (j+1)h \quad (12)$$

$$j = 0, 1, ..., n-1 \quad (13)$$

and solve ODE-IVP using explicit Euler method as follows

$$\frac{\mathbf{X}(kT + (j+1)h) - \mathbf{X}(kT + jh)}{h} = \mathbf{f}[\mathbf{X}(kT + jh), \mathbf{U}(k), D(k)] \quad (14)$$

or

$$\begin{aligned} \mathbf{X}(kT + (j+1)h) &= \mathbf{X}(kT + jh) + h \times \mathbf{f}[\mathbf{X}(kT + jh), \mathbf{U}(k), D(k)] \\ j &= 0, 1, ..., n-1 \\ \mathbf{X}(kT) &= \mathbf{X}(k) \quad \text{(Initial Condition)} \\ \mathbf{X}((k+1)T) &= \mathbf{X}(k+1) = \mathbf{X}(kT + (j+1)h) \end{aligned}$$

To implement this algorithm, modify

*Xk(:,k+1) = Xk(:,k) + samp_ T * QTank_ Dynamics(kT(k), Xk(:,k)) ;*

as follows

X0 = *Xk(:,k)*

h = samp_T/n ;

for j=0:n-1

X1 = X0 + h * *QTank_ Dynamics(kT(k)+jh, X0) ;*

X0 = X1 ;

end

Xk(:,k+1) = X1 ;

To make this code work for System 1 and get results similar to ode45, you have to select n = 10 or higher, i.e., integration interval h is 0.01 or smaller.

Even though n = 1 and h = T works for System 2, to get results closer to oder45, it is better to use $n \geq 10$ even for System 2.