Model Predictive Control

Lecture 03B - The Van der Pol Oscillator Problem

John Bagterp Jørgensen

Department of Applied Mathematics and Computer Science
Technical University of Denmark

02619 Model Predictive Control

Deterministic and Stochastic Differential Equations

- Ordinary differential equations (ODEs)

$$dx(t) = f(x(t))dt$$

- Stochastic differential equations (SDEs)
  with state independent diffusion

$$d\boldsymbol{x}(t) = f(\boldsymbol{x}(t))dt + \sigma d\boldsymbol{\omega}(t)$$

- Stochastic differential equations (SDEs)
  with state dependent diffusion

$$d\boldsymbol{x}(t) = f(\boldsymbol{x}(t))dt + g(\boldsymbol{x}(t))d\boldsymbol{\omega}(t)$$

Different ways of writing ODEs

- The dot-form

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

- The standard differential form

$$\frac{dx(t)}{dt} = f(x(t))$$

- Another differential form

$$dx(t) = f(x(t))dt$$

- Integral form

$$x(t) - x(t_0) = \int_{t_0}^{t} f(x(s))ds$$

Numerical methods for ODEs

- ODEs
$$dx(t) = f(x(t))dt$$
- Non-stiff system - explicit method (explicit Euler)
$$x_{k+1} - x_k = f(x_k)\Delta t_k$$
- Stiff system - implicit method (implicit Euler)
$$x_{k+1} - x_k = f(x_{k+1})\Delta t_k$$

Different ways of deriving the explicit Euler method

- ODEs

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

- Differential standard form

$$\frac{dx(t)}{dt} = f(x(t)) \qquad \frac{x_{k+1} - x_k}{t_{k+1} - t_k} = \frac{\Delta x_k}{\Delta t_k} = f(x_k)$$

- Another differential form

$$dx(t) = f(x(t)dt \qquad x_{k+1} - x_k = \Delta x_k = f(x_k)\Delta t_k$$

- Integral form

$$x(t_{k+1}) - x(t_k) = \int_{t_k}^{t_{k+1}} f(x(t))dt \qquad x_{k+1} - x_k = f(x_k)\Delta t_k$$

- Numerical procedure

$$x_{k+1} = x_k + f(x_k)\Delta t_k$$

Standard Wiener Process (multivariate)

```
function [W,Tw,dW] = StdWienerProcess(T,N,nW,Ns,seed)
% StdWienerProcess   Ns realizations of a standard Wiener process
%
% Syntax: [W,Tw,dW] = StdWienerProcess(T,N,Ns,seed)
%         W    : Standard Wiener process in [0,T]
%         Tw   : Time points
%         dW   : White noise used to generate the Wiener process
%
%         T    : Final time
%         N    : Number of intervals
%         nW   : Dimension of W(k)
%         Ns   : Number of realizations
%         seed : To set the random number generator (optional)

if nargin == 4
   rng(seed);
end
dt = T/N;
dW = sqrt(dt)*randn(nW,N,Ns);
W  = [zeros(nW,1,Ns) cumsum(dW,2)];
Tw = 0:dt:T;
```

Explicit-Explicit SDE Solver

```
function X=SDEsolverExplicitExplicit(ffun,gfun,T,x0,W,varargin)

N  = length(T);
nx = length(x0);
X  = zeros(nx,N);

X(:,1) = x0;
for k=1:N-1
   f = feval(ffun,T(k),X(:,k),varargin{:});
   g = feval(gfun,T(k),X(:,k),varargin{:});
   dt = T(k+1)-T(k);
   dW = W(:,k+1)-W(:,k);
   psi = X(:,k) + g*dW;
   X(:,k+1) = psi + f*dt;
end
```

Different ways of deriving the implicit Euler method

- ODEs

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

- Differential standard form

$$\frac{dx(t)}{dt} = f(x(t)) \qquad \frac{x_{k+1} - x_k}{t_{k+1} - t_k} = \frac{\Delta x_k}{\Delta t_k} = f(x_{k+1})$$

- Another differential form

$$dx(t) = f(x(t)dt \qquad x_{k+1} - x_k = \Delta x_k = f(x_{k+1})\Delta t_k$$

- Integral form

$$x(t_{k+1}) - x(t_k) = \int_{t_k}^{t_{k+1}} f(x(t))dt \qquad x_{k+1} - x_k = f(x_{k+1})\Delta t_k$$

- Numerical procedure (solve for x_{k+1} by some root-finding method, e.g. Newton's method)

$$R_k(x_{k+1}) = x_{k+1} - f(x_{k+1})\Delta t_k - x_k = 0$$

Numerical methods for SDEs

- SDEs
$$d\boldsymbol{x}(t) = f(\boldsymbol{x}(t))dt + g(\boldsymbol{x}(t))d\boldsymbol{\omega}(t)$$
- Non-stiff system - explicit-explicit method (Euler-Maruyama)
$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = f(\boldsymbol{x}_k)\Delta t_k + g(\boldsymbol{x}_k)\Delta \boldsymbol{w}_k, \quad \Delta \boldsymbol{w}_k \sim N_{iid}(0, I\Delta t_k)$$
$$x_{k+1} = f(x_k)\Delta t_k + \psi_k, \qquad \psi_k = x_k + g(x_k)\Delta w_k$$
- Stiff system - implicit-explicit method
$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = f(\boldsymbol{x}_{k+1})\Delta t_k + g(\boldsymbol{x}_k)\Delta \boldsymbol{w}_k, \quad \Delta \boldsymbol{w}_k \sim N_{iid}(0, I\Delta t_k)$$
$$R_k(x_{k+1}) = x_{k+1} - f(x_{k+1})\Delta t_k - \psi_k = 0, \quad \psi_k = x_k + g(x_k)\Delta w_k$$

Implicit-explicit solution method for SDEs

- Stiff system - implicit-explicit method

$$\boldsymbol{x}_{k+1}-\boldsymbol{x}_k = f(\boldsymbol{x}_{k+1})\Delta t_k+g(\boldsymbol{x}_k)\Delta \boldsymbol{w}_k, \quad \Delta \boldsymbol{w}_k \sim N_{iid}(0, I\Delta t_k)$$

- Residual equation

$$R_k(x_{k+1}) = x_{k+1}-f(x_{k+1})\Delta t_k-\psi_k = 0, \quad \psi_k = x_k+g(x_k)\Delta w_k$$

- ODEs: $\psi_k = x_k$. SDEs: $\psi_k = x_k + g(x_k)\Delta w_k$.
- This system of nonlinear equations can be solved by Newton's method

$$\begin{aligned}
&\|R_k(x_{k+1})\| > \epsilon : \\
&M_k = \frac{\partial R_k}{\partial x_{k+1}} = I - J(x_{k+1})\Delta t_k, \quad J(x_{k+1}) = \frac{\partial f}{\partial x}(x_{k+1}) \\
&M_k \Delta x_{k+1} = -R_k(x_{k+1}) \\
&x_{k+1} := x_{k+1} + \Delta x_{k+1}
\end{aligned}$$

Newton Method in Implicit-Explicit SDE Solver

```
function [x,f,J] = SDENewtonSolver(ffun,t,dt,psi,xinit,tol,maxit,varargin)

I = eye(length(xinit));
x = xinit;
[f,J] = feval(ffun,t,x,varargin{:});
R = x - f*dt - psi;
it = 1;
while ( (norm(R,'inf') > tol) & (it <= maxit) )
   dRdx = I - J*dt;
   mdx = dRdx\R;
   x   = x - mdx;
   [f,J] = feval(ffun,t,x,varargin{:});
   R = x - f*dt - psi;
   it = it+1;
end
```

Implicit-Explicit SDE Solver

```
function X=SDEsolverImplicitExplicit(ffun,gfun,T,x0,W,varargin)

tol   = 1.0e-8;
maxit = 100;

N  = length(T);
nx = length(x0);
X  = zeros(nx,N);

X(:,1) = x0;
k=1;
   [f,~]   = feval(ffun,T(k),X(:,k),varargin{:});
for k=1:N-1
   g   = feval(gfun,T(k),X(:,k),varargin{:});
   dt  = T(k+1)-T(k);
   dW  = W(:,k+1)-W(:,k);
   psi = X(:,k) + g*dW;
   xinit   = psi + f*dt;
   [X(:,k+1),f,~] = SDENewtonSolver(...
                                 ffun,...
                                T(:,k+1),dt,psi,xinit,...
                                tol,maxit,varargin{:});
end
```

The Van der Pol Oscillator Problem

- Second order differential equation

$$\ddot{y}(t) = \mu(1 - y(t)^2)\dot{y}(t) - y(t)$$

- Equivalent system of first order ordinary differential equations (ODEs) $[x_1(t) = y(t),\, x_2(t) = \dot{y}(t)]$

$$\begin{aligned}\dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= \mu(1 - x_1(t)^2)x_2(t) - x_1(t)\end{aligned}$$

- SDE version (state independent diffusion)

$$\begin{aligned}d\boldsymbol{x}_1(t) &= \boldsymbol{x}_2(t)dt \\ d\boldsymbol{x}_2(t) &= \left[\mu(1 - \boldsymbol{x}_1(t)^2)\boldsymbol{x}_2(t) - \boldsymbol{x}_1(t)\right] dt + \sigma d\boldsymbol{\omega}(t)\end{aligned}$$

- SDE version (state dependent diffusion)

$$\begin{aligned}d\boldsymbol{x}_1(t) &= \boldsymbol{x}_2(t)dt \\ d\boldsymbol{x}_2(t) &= \left[\mu(1 - \boldsymbol{x}_1(t)^2)\boldsymbol{x}_2(t) - \boldsymbol{x}_1(t)\right] dt + \sigma(1 + \boldsymbol{x}_1(t)^2)d\boldsymbol{\omega}(t)\end{aligned}$$

Van der Pol Oscillator Problem - Matlab

- Drift

```
function [f,J] = VanderpolDrift(t,x,p)

mu = p(1);
tmp = mu*(1.0-x(1)*x(1));
f = zeros(2,1);
f(1,1) = x(2);
f(2,1) = tmp*x(2)-x(1);

if nargout > 1
    J = [0 1; -2*mu*x(1)*x(2)-1.0 tmp];
end
```

- Diffusion 1

```
function g = VanderPolDiffusion1(t,x,p)

sigma = p(2);
g = [0.0; sigma];
```

- Diffusion 2

```
function g = VanderPolDiffusion2(t,x,p)

sigma = p(2);
g = [0.0; sigma*(1.0+x(1)*x(1))];
```

Examples of Numerical Solution

- ODEs

$$dx_1(t) = x_2(t)dt$$
$$dx_2(t) = \left[\mu(1 - x_1(t)^2)x_2(t) - x_1(t)\right] dt$$

- Parameter

$$\mu = 3$$

- Initial conditions

$$x_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

- Explicit method (explicit Euler)

$$x_{k+1} - x_k = f(x_k)\Delta t_k$$

- Implicit method (implicit Euler)

$$x_{k+1} - x_k = f(x_{k+1})\Delta t_k$$

Examples of Numerical Solution

- SDE with state independent diffusion

$$
\begin{aligned}
d\boldsymbol{x}_1(t) &= \boldsymbol{x}_2(t)dt \\
d\boldsymbol{x}_2(t) &= \left[\mu(1 - \boldsymbol{x}_1(t)^2)\boldsymbol{x}_2(t) - \boldsymbol{x}_1(t)\right] dt + \sigma d\boldsymbol{\omega}(t)
\end{aligned}
$$

- Parameters

$$\mu = 3 \qquad \sigma = 1$$

- Initial conditions

$$x_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

- Explicit-explicit (Euler-Maryuama) solution method

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = f(\boldsymbol{x}_k)\Delta t_k + g(\boldsymbol{x}_k)\Delta \boldsymbol{w}_k, \quad \Delta \boldsymbol{w}_k \sim N_{iid}(0, I\Delta t_k)$$

- Implicit-explicit solution method

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = f(\boldsymbol{x}_{k+1})\Delta t_k + g(\boldsymbol{x}_k)\Delta \boldsymbol{w}_k, \quad \Delta \boldsymbol{w}_k \sim N_{iid}(0, I\Delta t_k)$$

Source: Frogerais, Bellanger, Senhadji (2013)

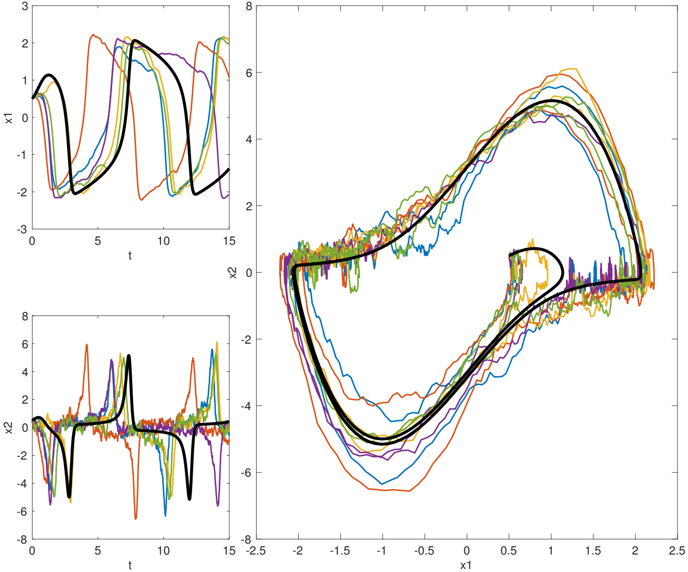
```
mu    = 3;
sigma = 1.0;
x0    = [0.5; 0.5];
p     = [mu; sigma];

tf = 5*mu;
nw = 1;
N  = 1000;
Ns = 5;
seed = 100;

[W,T,~]=StdWienerProcess(tf,N,nw,Ns,seed);
X = zeros(length(x0),N+1,Ns);
for i=1:Ns
   X(:,:,i) = SDEsolverExplicitExplicit(...
                    @VanderPolDrift,@VanderPolDiffusion1,...
                    T,x0,W(:,:,i),p);
end
Xd = SDEsolverExplicitExplicit(...
               @VanderPolDrift,@VanderPolDiffusion1,...
               T,x0,W(:,:,i),[mu; 0.0]);
```

Numerical Solution - Van der Pol - SDE version 1

$\mu = 3,\ \sigma = 1, x_0 = [0.5; 0.5]$



Examples of Numerical Solution

- SDE with state dependent diffusion

$$
\begin{aligned}
d\boldsymbol{x}_1(t) &= \boldsymbol{x}_2(t)dt \\
d\boldsymbol{x}_2(t) &= \left[\mu(1-\boldsymbol{x}_1(t)^2)\boldsymbol{x}_2(t) - \boldsymbol{x}_1(t)\right]dt + \sigma(1+\boldsymbol{x}_1(t)^2)d\boldsymbol{\omega}(t)
\end{aligned}
$$

- Parameters

$$\mu = 3 \qquad \sigma = 0.5$$

- Initial conditions

$$x_0 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

- Explicit-explicit (Euler-Maryuama) solution method

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = f(\boldsymbol{x}_k)\Delta t_k + g(\boldsymbol{x}_k)\Delta\boldsymbol{w}_k, \quad \Delta\boldsymbol{w}_k \sim N_{iid}(0, I\Delta t_k)$$

- Implicit-explicit solution method

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = f(\boldsymbol{x}_{k+1})\Delta t_k + g(\boldsymbol{x}_k)\Delta\boldsymbol{w}_k, \quad \Delta\boldsymbol{w}_k \sim N_{iid}(0, I\Delta t_k)$$

Source: Mazzoni (2008)

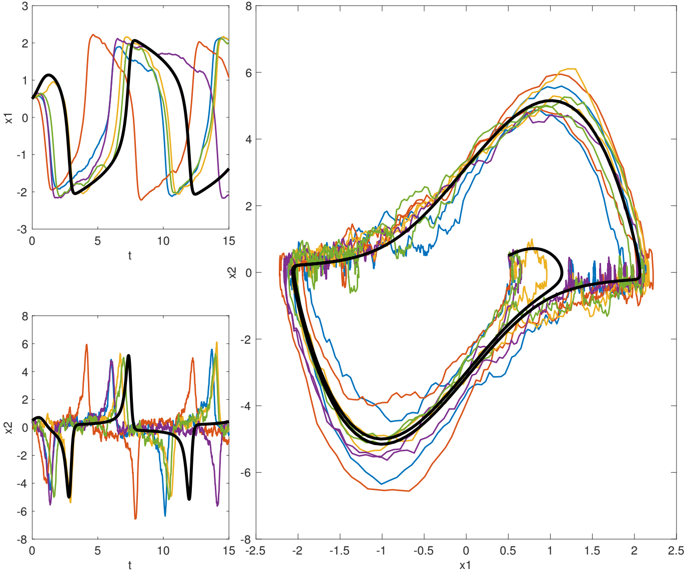
```
mu    = 3;
sigma = 0.5;
x0    = [0.5; 0.5];
p     = [mu; sigma];

tf = 5*mu;
nw = 1;
N  = 5000;
Ns = 5;
seed = 100;

[W,T,~]=StdWienerProcess(tf,N,nw,Ns,seed);
X = zeros(length(x0),N+1,Ns);
for i=1:Ns
   X(:,:,i) = SDEsolverExplicitExplicit(...
                   @VanderPolDrift,@VanderPolDiffusion2,...
                   T,x0,W(:,:,i),p);
end
Xd = SDEsolverExplicitExplicit(...
               @VanderPolDrift,@VanderPolDiffusion2,...
               T,x0,W(:,:,i),[mu; 0.0]);
```

Numerical Solution - Van der Pol - SDE version 1

$\mu = 3,\ \sigma = 1, x_0 = [0.5; 0.5]$



Numerical Solution - Van der Pol - SDE version 2

$\mu = 3$, $\sigma = 0.5$,$x_0 = [0.5; 0.5]$