The Robust Exact Differentiator Toolbox:

Improved Discrete-Time Realization

available at: www.reichhartinger.at

M. Reichhartinger[1], S. Koch[1,3], H. Niederwieser[1], S.K. Spurgeon[2]

[1]Institute of Automation and Control, Graz University of Technology
[2] Department of Electronic & Electrical Engineering, University College London
[3] Christian Doppler Laboratory for Model Based Control of Complex Test Bed Systems

Continuous-time differentiator revisited and main idea of the discrete-time design of the implemented differentiator

Continuous-time generation of $f(t)$

chain of $n+1$ integrators

$$\frac{\mathrm{d}x}{\mathrm{d}t} = Ax + e_{n+1} f^{(n+1)}$$
$$y = e_1^{\mathrm{T}} x$$

- $f(t)$ ... signal to be differentiated
- $A = \begin{pmatrix} 0_{n\times 1} & I_{n\times n} \\ 0 & 0_{1\times n} \end{pmatrix}$
- $x = \begin{bmatrix} x_0 & \dots & x_n \end{bmatrix}^{\mathrm{T}}$

Discrete-time generation of $f(k\tau)$

$$x_{k+1} = \Phi x_k + \tau h_k$$
$$y_k = e_1^{\mathrm{T}} x_k$$

- τ ... sampling time, $k = 0, 1, 2, \dots$
- $\Phi = e^{A\tau}$
- $h_k = \begin{bmatrix} \frac{\tau^n}{(n+1)!} f^{(n+1)} & \frac{\tau^{n-1}}{(n)!} f^{(n+1)} & \dots & f^{(n+1)} \end{bmatrix}^{\mathrm{T}}$

Continuous-time differentiator

$$\frac{\mathrm{d}\hat{x}}{\mathrm{d}t} = A\hat{x} + \psi(\sigma_0)\sigma_0$$

- $\psi^{\mathrm{T}} = \begin{bmatrix} \psi_0(\sigma_0) & \psi_1(\sigma_0) & \dots & \psi_n(\sigma_0) \end{bmatrix}$
- $\psi_i(\sigma_0) = k_i |\sigma_0|^{\frac{n-i}{n+1}} \quad i = 0, 1, \dots, n$

error dynamics with errors $\sigma = x - \hat{x}$

$$\frac{\mathrm{d}\sigma}{\mathrm{d}t} = [A - \psi(\sigma_0) e_1^{\mathrm{T}}]\sigma + e_{n+1} f^{(n+1)}$$

The matrix $[A - \psi(\sigma_0) e_1^{\mathrm{T}}]$ has eigenvalues

$$s_i = p_i |\sigma_0|^{-\frac{1}{n+1}}$$

where $p_i \in \mathbb{C}$ are the roots of

$$p^{n+1} + k_0 p^n + \dots + k_{n-1} p + k_n$$

Discrete-time differentiator

$$\hat{x}_{k+1} = \Phi \hat{x}_k + \lambda(\sigma_{0,k})\sigma_{0,k}$$

- $\lambda^{\mathrm{T}} = \begin{bmatrix} \lambda_0(\sigma_{0,k}) & \lambda_1(\sigma_{0,k}) & \dots & \lambda_n(\sigma_{0,k}) \end{bmatrix}$

error dynamics with errors $\sigma_{0,k} = x_{0,k} - \hat{x}_{0,k}$

$$\sigma_{k+1} = [\Phi - \lambda(\sigma_{0,k}) e_1^{\mathrm{T}}]\sigma_k + \tau h_k$$

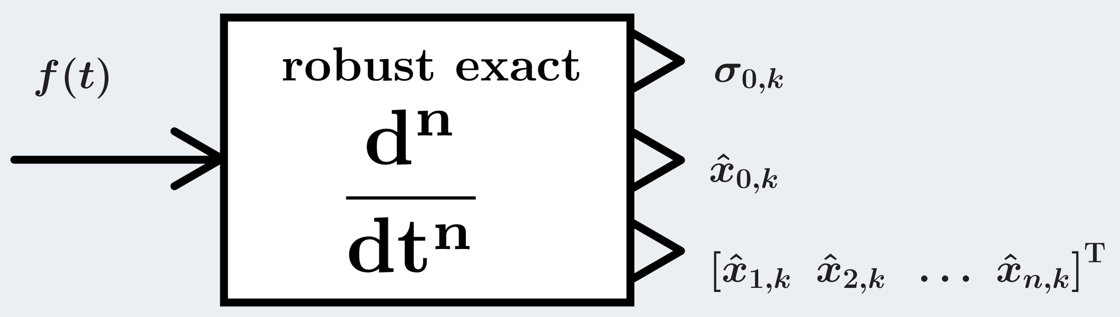
Main idea

Design the injection function $\lambda(\sigma_{0,k})$ such that the eigenvalues of the matrix $[\Phi - \lambda(\sigma_{0,k}) e_1^{\mathrm{T}}]$ are located at

$$z_i = e^{\tau s_{i,k}(\sigma_{0,k})}$$

where $s_{i,k}(\sigma_{0,k}) = p_i |\sigma_{0,k}|^{-\frac{1}{n+1}}$, i.e. the sampled continuous-time eigenvalue s_i.

Simulink® block



input signal:

- f ... signal to be differentiated

output signals:

- $\sigma_{0,k}$ ... estimation error $f_k - \hat{x}_{0,k}$
- $\hat{x}_{0,k}$ ... estimation of f
- $\begin{bmatrix} \hat{x}_{1,k} & \dots & \hat{x}_{n,k} \end{bmatrix}$ ... estimated derivatives

parameters:

- n ... order of the differentiator
- τ ... discretization time
- c ... root location, i.e. $p^{n+1} + k_0 p^n + \dots + k_{n-1} p + k_n = (p+c)^{n+1}$

Sketch of the implementation

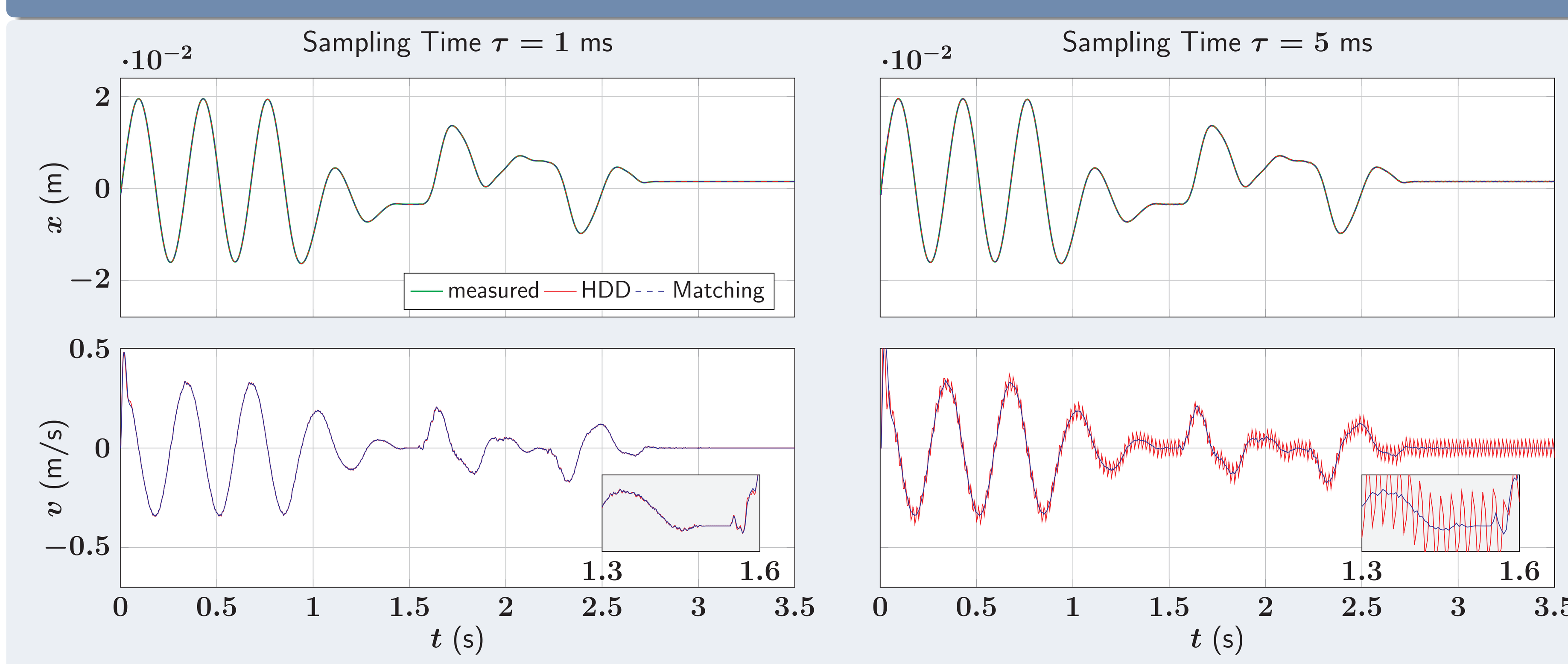
- Implementation using a so-called Matlab function block
- Automatic code generation is supported
- Implemented up to order $n = 10$

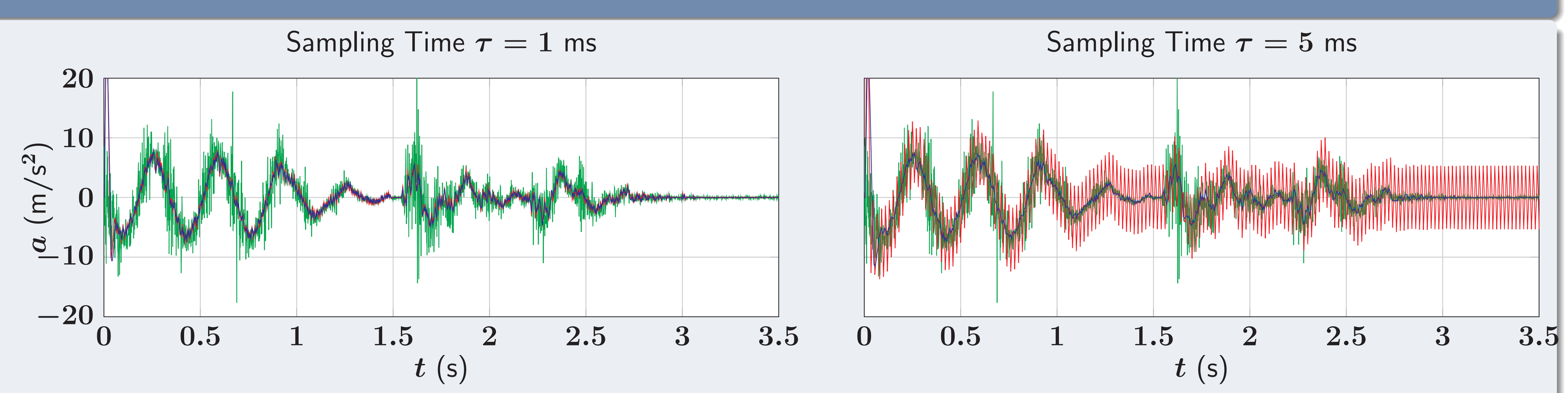
Code fragment of a differentiator of order $n = 2$:

```
n = 2;                                    % n ... order of the differentiator

% Compute Error
sigma_0=f-xp(1);                          % sigma_0 ... estimation error
if abs(sigma_0) == 0
    z = 0;
else
    s = -c*(abs(sigma_0))^(-1/(n+1));     % current eigenvalue
    z = exp(tau*s);                       % Matching discretization approach
    %z = 1/(1-(s*tau));                   % Implicit discretization approach
end
% Second Order Differentiator
lambda = ...
[(3 - 3*z)*abs(sigma_0)
(((z - 1)/tau*abs(sigma_0)^(1/2))^2*(z + 5))*tau/2
-((z - 1)/tau*abs(sigma_0)^(1/3))^3*tau];
% Compute Estimates
xp = Phi*xp + lambda*sign(sigma_0);
```

Experimental results of a vertically moving platform: velocity and acceleration estimation using different discretization times





Conclusion

- New discretization scheme allows to eliminate the discretization chattering
- Accuracy is improved for large discretization times

markus.reichhartinger@tugraz.at