Global Least Squares Solution for LTI System Response

Gerhard Rath
University of Leoben
Chair of Automation
Leoben, Austria
Email: gerhard.rath@unileoben.ac.at

Matthew Harker
University of Leoben
Chair of Automation
Leoben, Austria
Email: matthew.harker@unileoben.ac.at

***Abstract*—Linear time invariant (LTI) systems are the most important method to describe dynamic systems for the purpose of modeling, simulation and design of controllers. Dynamic systems are a process developing over time, hence the solution is an initial value problem (IVP). Solvers for ordinary differential equations (ODE) are commonly used for evaluation. A new idea is to solve such systems directly after discretization based on an least-squares problem with equality constraints (LSE). The approach is demonstrated on a simple system with four state variables and a rank-deficient system matrix. A comparison with standard ODE solution shows the correctness of the solution, advantages are discussed.**

I. Introduction

Many physical processes may be described with differential equations. More than one independent variable leads to partial differential equations (PDE) [1]. In many cases assumptions for simplifications can be made, reducing the equations to one independent variable and their derivatives, these are ordinary differential equations (ODE) [2]. They can be divided into initial value problems (IVP) and boundary value problems (BVP).

Dynamic systems are processes developing over time, usually IVP. Higher order differential equations are reduced to systems of first order ODE [3]. If the system is linear and time invariant (LTI), the state space formulation will be

$$\begin{aligned} \dot{\boldsymbol{x}} &= \mathbf{A}\boldsymbol{x} + \mathbf{B}\boldsymbol{u} \\ \boldsymbol{x}\,(0) &= \boldsymbol{x}_0 \end{aligned} \tag{1}$$

with $\mathbf{A}$ is the *dynamic matrix*, $\mathbf{B}$ is the *input coupling matrix* and $\boldsymbol{x}_0$ are the initial conditions of the state vector $\boldsymbol{x}$. For solving such IVP, several methods have been developed, e.g. Heun, Runge-Kutta, Dormand-Prince, or Bogacki-Shampine, which are available in every software for solving ODEs [4].

An alternative approach to find a global solution after discretization of a system was used to find a curve whose differentials are known by inclinometer measurements [5], [6]. In [7] such method was used for a process in time, solving the canonical system for an optimization problem. It had the advantage to solve the BVP without the need of the matrix exponential, an operation, which may cause problems [8]. Later it turned out that the proposed method applied to stiff systems has the advantage to find a solution, while classical solvers have problems [9].

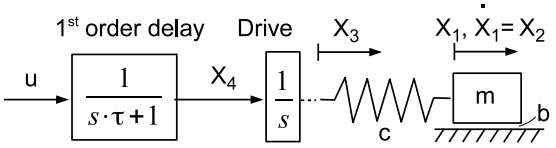

Fig. 1. Positioning system of fourth order

In the actual work the method is applied to find the response of a continuous-time LTI system in state-space representation to an arbitrary control input, subject to given initial conditions.

The approach taken in this paper is to discretize the LTI system directly over the entire time interval as a linear system of equations. The initial conditions (or boundary values as the case may be) take the form of linear constraints imposed on the solution of the systems of equations describing the LTI system.

II. Example System

A. Mechanical System

As a demonstration example for a LTI system the model of a drive with elastic coupling to a mass is used (Fig. 1). A mass m with friction b to the environment is coupled with an elastic element with a spring constant of c to a drive. The position of the mass is x_1, x_2 is its speed, x_3 is the position of the left spring end, and x_4 is the speed of the drive. The control input u is speed, and the dynamical behavior of the drive is modeled with a first-order delay with a time constant τ. This system can be used as a basic model for many drives transporting a payload. For example, applied to a hydraulic axis, the first-order part with the time constant τ is a good representation for the behavior of a servo-drive [10].

The system that was used for this work is an electrically driven linear drive in our laboratory. Usually electronics for servo motors have limited speeds and accelerations. To overcome this non-linear behavior, a pre-filter of first order with a proper time constant τ was used. The load was a ball hanging on a string with a length of approximately one meter, modeling a portal crane with a swinging load. Since the mass is only a scaling factor, $m = 1$ was assumed. The spring constant follows from the string length for a

ISBN 978-80-261-0642-5,

linearized pendulum and is $c = 13$. Friction was determined experimentally from decreasing amplitude of oscillations with $b = 0.2$. The time constant $\tau = 1$ is a pre-filter applied to the frequency inverter driving the servo motor.

B. Mathematical Model

The resulting system is of fourth order and can be modelled with

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{c}{m} & -\frac{b}{m} & \frac{c}{m} & 0 \\ 0 & 0 & 0 & \frac{1}{A} \\ 0 & 0 & 0 & -\frac{1}{\tau} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{\tau} \end{bmatrix} \cdot u. \tag{2}$$

The initial conditions

$$\boldsymbol{x}\,(0) = \begin{bmatrix} x_{10} & x_{20} & x_{30} & x_{40} \end{bmatrix}^T \tag{3}$$

are the initial load position x_{10} , the initial speed x_{20}, the initial position of the spring x_{30}, and the initial drive speed x_{40}. All are assumed to be zero.

III. Direct Global Solution Approach

A. Discretization

The system equation (1) rewritten is

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t). \tag{4}$$

Transposing it to

$$\dot{\boldsymbol{x}}^{\mathrm{T}} = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{u}^{\mathrm{T}}\boldsymbol{B}^{\mathrm{T}} \tag{5}$$

makes it more convenient for discretization. This set of equations can be evaluated at each discrete time step, to obtain,

$$\begin{aligned} \dot{\boldsymbol{x}}^{\mathrm{T}}(0) &= \boldsymbol{x}^{\mathrm{T}}(0)\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{u}^{\mathrm{T}}(0)\boldsymbol{B}^{\mathrm{T}} \\ \dot{\boldsymbol{x}}^{\mathrm{T}}(t_1) &= \boldsymbol{x}^{\mathrm{T}}(t_1)\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{u}^{\mathrm{T}}(t_1)\boldsymbol{B}^{\mathrm{T}} \\ &\vdots \\ \dot{\boldsymbol{x}}^{\mathrm{T}}(t_f) &= \boldsymbol{x}^{\mathrm{T}}(t_f)\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{u}^{\mathrm{T}}(t_f)\boldsymbol{B}^{\mathrm{T}}. \end{aligned} \tag{6}$$

This set of equations, however, can be stacked together to define the matrix equation [11],

$$\dot{\boldsymbol{X}} = \boldsymbol{X}\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{U}\boldsymbol{B}^{\mathrm{T}}. \tag{7}$$

Now all discrete time values of the state variables are concentrated in

$$\boldsymbol{X} = \begin{bmatrix} x_1\,(0) & x_2\,(0) & \cdots & x_p\,(0) \\ x_1\,(t_1) & x_2\,(t_1) & \cdots & x_p\,(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ x_1\,(t_f) & x_2\,(t_f) & \cdots & x_p\,(t_f) \end{bmatrix}, \tag{8}$$

and all input values are in

$$\boldsymbol{U} = \begin{bmatrix} u_1\,(0) & u_2\,(0) & \cdots & u_q\,(0) \\ u_1\,(t_1) & u_2\,(t_1) & \cdots & u_q\,(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ u_1\,(t_f) & u_2\,(t_f) & \cdots & u_q\,(t_f) \end{bmatrix}. \tag{9}$$

The number of state variables is represented by p, the number of inputs is q, and n is the number of samples in time from 0 to t_f.

With an appropriate numerical approximation to a differentiating matrix [5], [6], (7) can be written as,

$$\boldsymbol{D}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{U}\boldsymbol{B}^{\mathrm{T}}, \tag{10}$$

where the matrix $\boldsymbol{D}$ is effectively applied to each column of $\boldsymbol{X}$. Vectorizing this matrix equation [12], i.e., stacking the columns of the equation into a vector leads to

$$\mathrm{vec}(\boldsymbol{D}\boldsymbol{X}) = \mathrm{vec}(\boldsymbol{X}\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{U}\boldsymbol{B}^{\mathrm{T}}). \tag{11}$$

After introducing the Kronecker product follows

$$\begin{aligned} &(\boldsymbol{I}_p \otimes \boldsymbol{D})\,\mathrm{vec}(\boldsymbol{X}) = \\ &(\boldsymbol{A} \otimes \boldsymbol{I}_n)\,\mathrm{vec}(\boldsymbol{X}) + (\boldsymbol{B} \otimes \boldsymbol{I}_n)\,\mathrm{vec}(\boldsymbol{U}) \end{aligned} \tag{12}$$

and further

$$[(\boldsymbol{I}_p \otimes \boldsymbol{D}) - (\boldsymbol{A} \otimes \boldsymbol{I}_n)]\,\mathrm{vec}(\boldsymbol{X}) = (\boldsymbol{B} \otimes \boldsymbol{I}_n)\,\mathrm{vec}(\boldsymbol{U}). \tag{13}$$

With the abbreviations

$$\boldsymbol{b} = (\boldsymbol{B} \otimes \boldsymbol{I}_n) \tag{14}$$

and

$$\boldsymbol{A}_1 = (\boldsymbol{I}_p \otimes \boldsymbol{D}) - (\boldsymbol{A} \otimes \boldsymbol{I}_n) \tag{15}$$

Equation (13) becomes

$$\boldsymbol{A}_1\,\mathrm{vec}(\boldsymbol{X}) = \boldsymbol{b}\,\mathrm{vec}(\boldsymbol{U}) \tag{16}$$

or finally

$$\boldsymbol{A}_1\,\mathrm{vec}(\boldsymbol{X}) - \boldsymbol{b}\,\mathrm{vec}(\boldsymbol{U}) = \boldsymbol{0}. \tag{17}$$

B. Initial Conditions

Equation (17) will be solved as a least squares (LS) problem, but before this, the initial conditions (3) are to be processed. They can be formulated as a constraint leading to a least squares problem with equality constraints (LSE). In the formulation

$$\boldsymbol{C}^{\mathrm{T}}\,\mathrm{vec}(\boldsymbol{X}) = \boldsymbol{d}, \tag{18}$$

$\mathrm{vec}(\boldsymbol{X})$ is the vector with the discrete time instances of all state variables. The matrix $\boldsymbol{C}^{\mathrm{T}}$ has p rows representing the initial condition for each state variable, in our example there are four. Each row contains a one in a proper column to express the condition. The column vector $\boldsymbol{d}$ contains the values for the initial conditions.

C. Solution

Now the LSE problem to be solved is

$$\|\boldsymbol{A}_1\,\mathrm{vec}(\boldsymbol{X}) - \boldsymbol{b}\,\mathrm{vec}(\boldsymbol{U})\|_2^2 = \min \tag{19}$$

subject to the constraints

$$\boldsymbol{C}^{\mathrm{T}}\,\mathrm{vec}(\boldsymbol{X}) = \boldsymbol{d}. \tag{20}$$

A possible solution according to [12] requires the QR decomposition of the matrix $\boldsymbol{C}$ and its partitioning

$$\boldsymbol{C} = \boldsymbol{Q}\boldsymbol{R} = \begin{bmatrix} \hat{\boldsymbol{Q}} & \tilde{\boldsymbol{Q}} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{R}} \\ \boldsymbol{0} \end{bmatrix}. \tag{21}$$

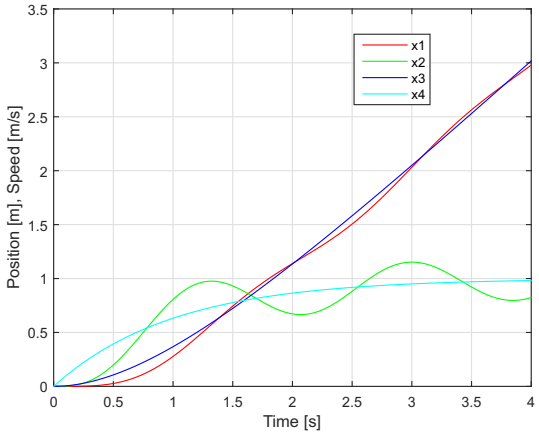

Fig. 2. Step response obtained with proposed method

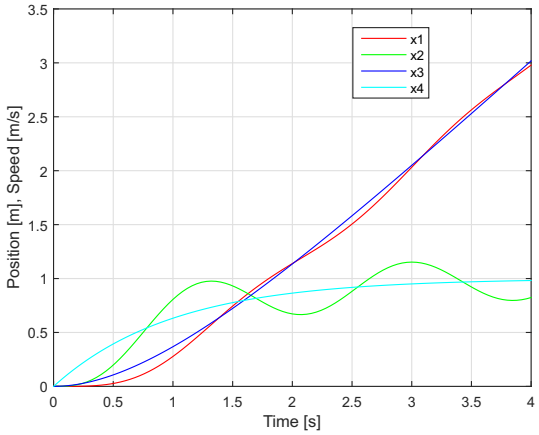

Fig. 3. Step response obtained with Matlab®

Now the desired state variables over time can be found as

$$\mathrm{vec}(\boldsymbol{X}) = \boldsymbol{M}\boldsymbol{b}\,\mathrm{vec}(\boldsymbol{U}) + (\boldsymbol{I} - \boldsymbol{M}\boldsymbol{A}_1)\,\hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}^{-\mathrm{T}}\boldsymbol{d} \quad (22)$$

with the abbreviation

$$\boldsymbol{M} = \tilde{\boldsymbol{Q}}\left(\boldsymbol{A}_1\tilde{\boldsymbol{Q}}\right)^{+}. \quad (23)$$

It is remarkable, that the right side of (22) is split into two terms, one describing the influence of the inputs (with $\mathrm{vec}(\boldsymbol{U})$), the other the effect of the initial conditions (with $\boldsymbol{d}$). This means, for a given system, the response to a changed input or different initial conditions can be evaluated simply with one matrix multiplication.

IV. Results

For verification, step responses are calculated from the system represented by (2) and compared. As a reference, the exact solution for the actual system was calculated. It was found using the transfer function

$$G(s) = \frac{x_1(s)}{u(s)} = \frac{1}{s\,(s\tau + 1)\left(s^2\frac{m}{c} + s\frac{b}{c} + 1\right)} \quad (24)$$

for the system (2). This is possible, if all initial conditions $\boldsymbol{d}$ are zero. With the input step $u(s) = \frac{1}{s}$, making a partial fraction expansion and inverse Laplace transform, the exact analytical solution $x_1(t)$ can be found.

With the proposed global LS solution the step response was calculated using (22) and (23). Of course this method is not restricted to step response, any arbitrary input $\boldsymbol{u}$ and initial conditions $\boldsymbol{d}$ are possible instead. The differentiating matrix $\boldsymbol{D}$ in (10) was built with Legendre polynomials of seventh order [5]. If the time span of 4 s is devided into $n = 100$ samples for discretization, the system matrix $\boldsymbol{A}_1$ in (19) has $np = 400$ rows and columns. The result of the computation is shown in Fig. 2.

For comparison, the same system was tested with Matlab®, creating a LTI object to represent the system (2). The code snippet for creating the system's state space representation and calculating the step response is:

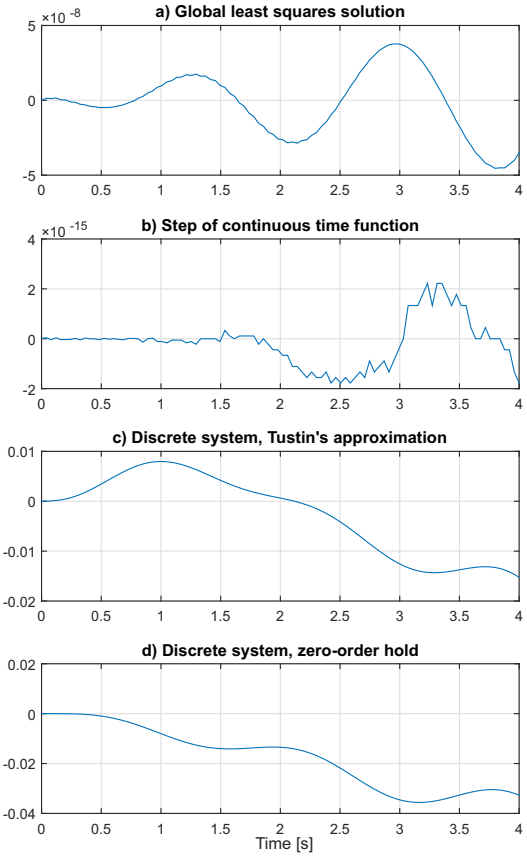

Fig. 4. Errors in step response for state variable x_1

```
sys = ss(A,B,eye(4),0);
[x1, t] = step(sys, 4);
```

As to be seen in Fig. 3, it obviously yields the same result.

Fig. 4a displays the deviation from the exact solution, while 4b shows the result for Matlab's *step* function.

The next experiments are to create discrete transfer functions with

Method	Elapsed time
Global LS solution	70 ms
Step function, continuous	3 ms
Discrete (Tustin)	2 ms
Discrete (ZOH)	2 ms
ODE solution (ode45)	4 ms

TABLE I: Computing time

```
sys2 = c2d(sys,T, method);
x1 = step(sys, 4);
```

where T is the sampling time. Fig. 4c shows the result for the discrete transfer function with *method* = 'tustin', and Fig. 4d for *method* = 'zoh' (zero-order hold). Transfer functions created with Matlab's *c2d* function are accepting streaming input data, the other methods operate on block data. Finally, in Fig. 4e the solution of (2) with the *ode45* function is given. It returns for this case 81 values with variable time steps, instead of 100 equidistant values like the other examples.

Computing times are given in Tab. I. They were measured on a PC with an i3 processor running Windows 7®. Due to the different procedures, the times cannot be compared directly, but it is obvious, that the proposed method is a serious candidate for implementation even on embedded systems.

Once the system is given, the response to a different input can be evaluated equivalent to convolution, but with only one additional matrix multiplication in (22). This would reduce the elapsed computing time essentially. The same is valid for changing the initial conditions.

V. Conclusion

This paper presented a direct matrix-based solution to the solution of an LTI system. The method was shown to be in very close agreement with the standard convolution-based solution. It was shown that the solution to the system of differential equations can be written as an explicit function of the control functions and the initial conditions. One advantage of the new method is that the order of approximation of the derivatives can be increased without altering the algorithm in any way. Similarly, alternative interpolation functions, such as trigonometric or exponential functions can be used. It is also a trivial matter to extend the proposed algorithm to time-varying systems, e.g., where the matrix $\boldsymbol{A} = \boldsymbol{A}(t)$ and/or $\boldsymbol{B} = \boldsymbol{B}(t)$ are time-dependent. Furthermore, it is straightforward to solve a system with intermediate value conditions in addition to initial and final values. Potential future work is to solve the matrix equation describing the system directly, which would further accelerate the computation, and provide an efficient means of treating systems with a large number of states and controls.

Acknowledgment

The authors would like to thank Prof. Paul O'Leary, the head of our institution, for initiating, enabling and inspiring this work.

References

[1] L. Evans, *Partial Differential Equations*, vol. 19 of *Graduate studies in mathematics*. American Mathematical Society, 2 ed., 2010.

[2] R. Burden and J. Faires, *Numerical Analysis*. Brooks/Cole, Cengage Learning, 9 ed., 2011.

[3] G. Franklin, J. Powell, and M. Workman, *Digital Control of Dynamic Systems*. Menlo Park, CA, USA: Addison-Wesley Longman, Inc., third ed., 1997.

[4] L. Shampine and M. Reichelt, "The matlab ode suite," *SIAM Journal on Scientific Computing*, vol. 18, pp. 1–22, 1997.

[5] P. O'Leary, C. Gugg, M. Harker, and G. Rath, *Mathematical Model and Software Architecture for the Solution of Inverse Problems Involving Sensor Arrays*, pp. 586–589. Inst. of Electrical and Electronics Engineers, 2014.

[6] P. O'Leary and M. Harker, "A framework for the evaluation of inclinometer data in the measurement of structures," *Instrumentation and Measurement, IEEE Transactions on*, vol. 61, pp. 1237–1251, May 2012.

[7] G. Rath and M. Harker, "Direct numerical solution of optimal control problems," in *2016 5th Mediterranean Conference on Embedded Computing (MECO)*, pp. 304–308, June 2016.

[8] C. Moler and C. V. Loan, "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later," *SIAM Review*, vol. 45, no. 1, pp. 3–49, 2003.

[9] G. Rath, M. Harker, and E. Zaev, "Direct numerical solution of stiff ode systems in optimal control." submitted to MECO, 2017.

[10] L. Larsson, "Modelling of the swash plate control actuator in an axial piston pump for a hardware-in- the-loop simulation test rig," in *Proceedings of the 9th FPNI Ph.D. Symposium on Fluid Power*, 2016.

[11] M. Harker, *Differential Equations, Inverse Problems, and Fractional Calculus in Mechatronics*. Habilitation thesis, Montanuniversität Leoben, 2015.

[12] G. Golub and C. Van Loan, *Matrix Computations*. Baltimore: The Johns Hopkins University Press, 4th ed., 2014.