Numerical algorithm for feedback linearizable systems

S. S. Willson, Philippe Müllhaupt and Dominique Bonvin

Laboratoire d'Automatique, Station 9, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland.

Abstract. A numerical algorithm that achieves asymptotic stability for feedback linearizable systems is presented. The nonlinear systems can be represented in various forms that include differential equations, simulated physical models or lookup tables. The proposed algorithm is based on a quotient method and proceeds iteratively. At each step, the dynamic system is desensitized with respect to the current input vector field. Control is obtained by tracking a desired value along the input vector field at each step. The numerical algorithm uses the direction on the tangent manifold at a given point and its variation around that point. This enables the algorithm to produce control values simply using a simulator of the nonlinear system.

Keywords: Numerical algorithms, feedback-linearization, quotient, control law design, nonlinear dynamical systems
PACS: 05.10.-a

Feedback linearization is an effective method to handle nonlinear systems. However, it requires exact knowledge of the system and, furthermore, there exist conditions for a nonlinear system to be feedback linearizable (FL). Moreover, for feedback linearization, an output function of relative degree n must be known [1, 2], where n is the dimension of the system. Computing such an output requires a systematic procedure, such as the one proposed in [3, 4], which proceeds by successively generating quotients that are desensitized with respect to the input vector field. This method has been extended to produce a control design technique applicable to FL input-affine single-input systems of the form [5]:

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})u. \tag{1}$$

For non-FL systems, the method requires approximations [6, 7]. However, for some non-FL systems, the system of equations turns out to be so complicated that, even after approximations, the method is not applicable due to the lack of closed-form solutions to some of the equations. This difficulty has motivated the use of numerical algorithms to compute the control input. This paper presents a numerical algorithm that computes the input for a given state of the system. The algorithm requires the time derivatives of the states at particular state and input values. The time derivatives can be obtained from the system's differential equations (through traditional modeling) or through advanced computer-generated physical modeling. Computer-generated physical modeling is an interactive technique that connects electro-mechanical components to simulate dynamical systems [8]. Simscape™ is such a simulation tool. With it, a designer can simulate complicated electro-mechanical systems without having full knowledge of the underlying differential equations. The inability of these tools to provide the governing differential equations make them unsuited for developing nonlinear control laws. Using the proposed numerical algorithm, it is possible to harness the simulation capabilities of such tools. Moreover, if measured data are available for a system, it is also possible to compute the derivatives via numerical differentiation and use that for the proposed numerical algorithm

NUMERICAL QUOTIENT ALGORITHM

The numerical version encompasses the forward decomposition stage and backward control design stage of the quotient method [5]. The numerical algorithm assumes that there exists a procedure to compute the time derivatives of the states for given state and input values. That is, it is assumed that there exists a black box that simulates

$$\dot{\mathbf{x}} = \psi(\mathbf{x}, u). \tag{2}$$

The algorithm is oblivious of the way the dervatives are calculated. However, the time derivatives should be smooth with respect to small variations of the states and input. A possible implementation of the algorithm is shown in Figure 1. The decomposition stage creates a realization of the diffeomorphism that locally converts (2) into feedback form [9]. In the control design stage, the control errors are evaluated sequentially in order to generate the control input.

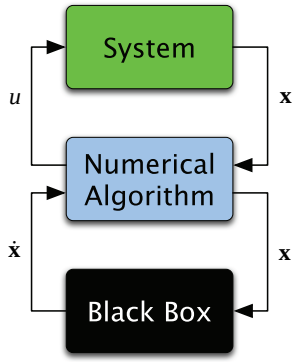

FIGURE 1. The various elements of the control scheme: the back box generates $\dot{\mathbf{x}}$, while the numerical algorithm computes u.

Forward decomposition stage

A transformation matrix is sought that locally transforms (2) into feedback form. The forward stage generates a system of the following form:

$$\begin{array}{rcl} \dot{y}_1 & = & f_{y,1} + g_{y,1} e_2 \\ \dot{y}_2 & = & f_{y,2} + g_{y,2} e_3 \\ \vdots & & \\ \dot{y}_{n-1} & = & f_{y,n-1} + g_{y,n-1} e_n \\ \dot{y}_n & = & f_{y,n} + g_{y,n} u, \end{array} \tag{3}$$

The complete stage requires $n-1$ steps. Every step involves computing a orthogonal matrix T such that $Tg(\mathbf{x}) = (0,\ldots,0,*)^T$. This Transformation matrix along with first order finite difference helps in determining all the terms in (3).

The algorithm. At the beginning of the algorithm, the current state $\mathbf{x}$ is known. The implementation of the algorithm proceeds as follows:

- Initialization:
  - Compute g_n by evaluating (2) at $f_+ = \psi(\mathbf{x}, \Delta u)$ and $f_- = \psi(\mathbf{x}, -\Delta u)$ and using the central-difference formula

$$g_n = \frac{f_+ - f_-}{2\Delta u},$$

    where Δu is a small number appropriate for the system under consideration. This step approximates the computation of

$$g(\mathbf{x}) = \frac{\partial \psi(\mathbf{x}, u)}{\partial u}.$$

  - Compute f_n by evaluating (2) at $f_n = \psi(\mathbf{x}, 0)$. This step is equivalent to computing $f(\mathbf{x}) = \psi(\mathbf{x}, 0)$.
- Iterative steps: At the beginning of the k^{th} iteration, the two $p = (n+1-k)-$dimensional vectors f_p and g_p are available. An orthogonal $(n \times n)$ transformation matrix T is also available (it is the identity matrix for $k = 1$).
  - Compute the orthogonal matrix T_p for g_p such that

$$T_p g_p = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ ||g_p|| \end{pmatrix}.$$

    T_p is a transformation that brings g_p to canonical form. This step is equivalent to computing the push-forward operator to obtain the normal form of the input vector field.

- Compute $f_z = T_p f_p.$, which brings f_p to the transformed coordinate where g_p is in canonical form.
- Compute a new T by composing T_p and T:

$$T_{\text{new}} = \begin{pmatrix} T_p & 0_{p\times(n-p)} \\ 0_{(n-p)\times p} & I_{(n-p)\times(n-p)} \end{pmatrix} T_{\text{old}}.$$

- The next few steps computes the input vector for the next iteration.
  - Compute the point corresponding to $\mathbf{x}$ in the transformed coordinate $\mathbf{z} = T\mathbf{x}$ and perturb this vector at the p^{th} element of $\mathbf{z}$ to obtain $\mathbf{z}_+ = \mathbf{z} + \Delta z$ and $\mathbf{z}_- = \mathbf{z} - \Delta z$.
  - Transform the points $\mathbf{z}_+$ and $\mathbf{z}_-$ back using $T^{-1} = T^T$ to obtain $\mathbf{x}_+ = T^T \mathbf{z}_+$ and $\mathbf{x}_- = T^T \mathbf{z}_-$.
  - Compute f for the points x_+ and x_- and bring them back to the transformed space:

$$f_{z+} = T\psi(\mathbf{x}_+, 0) \text{ and } f_{\mathbf{z}-} = T\psi(\mathbf{x}_-, 0).$$

  - Compute

$$g_z = \frac{f_{\mathbf{z}+} - f_{\mathbf{z}-}}{2\Delta z}.$$

- This iteration generates:
  - f_{p-1} : the first $p-1$ terms of the f_p vector. This is required for the next iteration.
  - g_{p-1} : the first $p-1$ terms of the g_z vector. This is required for the next iteration.
  - The transformation matrix T required for the next iteration.
  - In eq. (3), $g_{y,p} = ||g_p||$. This is required for the backward stage.
  - In eq. (3), $f_{y,p}$ is the p^{th} term of f_p. This is required for the backward stage.

• Termination: The forward process terminates when $p = 1$.

The forward decomposition transforms the nonlinear system (1), which is a local realization of (2), into the local feedback representation (3). Next, the backward stage will determine the control input based on this local feedback representation.

Backward control design stage

The forward decomposition stage provides the values of $f_{y,1}, \ldots, f_{y,n}$ and $g_{y,1}, \ldots, g_{y,n}$ in eq. (3). In addition, it also provides the transformation matrix T, which is a local realization of the push-forward operator required to obtain the feedback form of the system (1). Equipped with these values, the backward stage computes the input value.

• Initialization:
  - Compute $\mathbf{y} = A\mathbf{x}$ and $\dot{\mathbf{y}} = A\psi(\mathbf{x}, u_p)$, where u_p is the control input calculated at the previous time sample.
  - Set $e_1 = -L_1 y_1$, where y_1 is the first element of the vector $\mathbf{y}$ and L_1 is the proportional feedback gain.

• Iteration steps: At the beginning of the k^{th} iteration, e_k and $\dot{\mathbf{y}}$ are known.
  - Compute $\dot{e}_k$. There are two ways to compute this value:
    - Use the values of e_k from previous time samples to estimate $\dot{e}_k$. For example,

$$\dot{e}_k = \frac{e_k - e_{k-1}}{\Delta T}, \tag{4}$$

      where ΔT is the sampling period. This is computationally fast, but it is approximate and it creates difficulties at the initialization stage.
    - Use the central-difference formula to compute $\frac{\partial e_k}{\partial \mathbf{x}}\Big|_{\mathbf{x}}$ and use $\dot{e}_k = \frac{\partial e_k}{\partial \mathbf{x}}\dot{x} = \frac{\partial e_k}{\partial \mathbf{x}}\Big|_{\mathbf{x}} \psi(\mathbf{x}, u_p)$. This is a better estimate of $\dot{e}_k$, which however is computationally very demanding.
  - Implementing $\dot{e}_k = -L_k e_k$ and rearranging gives:

$$e_{k+1} = \frac{\dot{e}_k + \dot{y}_k + L_k e_k - f_{y,k}}{g_{y,k}}. \tag{5}$$

    Here, L_k is a positive gain and $\dot{y}_k$ is the k^{th} element of $\dot{\mathbf{y}}$.

- Termination: After n iterations, we obtain e_{n+1}, which is the required control input.

The accuracy of the algorithm suffers since central-difference is used to compute the partial derivatives. It restricts the algorithm to slowly varying systems. A fast varying system would require adaptive gradient estimation to determine the partial derivatives required during the forward and backward stages. Other approximations that effects the accuracy of the algorithm are

- The forward stage introduces approximations for computing the input vector field. The transformation matrix T is used to map the points on the base manifold. However, the diffeomorphism and the inverse diffeomorphism must be used for this mapping. Here, it is assumed that the diffeomorphism is $A\mathbf{x}$, which is only true when the system under consideration is linear.
- During the backward stage, computing $e_1 = -L_1 y_1$ is an approximation. The reason is the same as stated above: the value of $\mathbf{y}$ is obtained using a linear transformation of $\mathbf{x}$, whereas $\mathbf{y}$ should be obtained using a diffeomorphism at $\mathbf{x}$.

CONCLUSIONS

A numerical algorithm based on the quotient method has been presented for stabilizing feedback linearizable systems. The algorithm involves two stages. The forward stage transforms the system in order to (i) successively desensitize it with respect to the current input vector, and (ii) reduce its dimension. The result is a feedback form. The backward stage constructs a succession of proportional controllers to, at each step, track the error between a transformed state and its desired value. The numerical algorithm requires the time derivatives of the states for given state and input values. These time derivatives can be obtained through various means, such as traditional models based on differential equations or computer-generated physical modeling. Physical modeling provides the opportunity to deal with complex electro-mechanical systems, for which it is difficult to obtain the governing differential equations. The numerical algorithm requires various approximations, which restricts the domain of attraction of the resulting control scheme. If the underlying system is linear, this algorithm is globally attractive. If the system is feedback linearizable, the algorithm is capable of stabilizing a large domain of initial conditions, and in some cases, it can also be globally attractive.

This algorithm represents an initial attempt to develop numerical algorithms for the stabilization of a wider class of nonlinear systems. In fact, it can be regarded as an attempt to extend Miminis-Paige algorithm to nonlinear systems [10]. This work requires further research to fine tune the algorithm and investigate numerical issues. Further research is also required regarding the effect of replacing the proportional linear feedback with various types of nonlinear feedback.

ACKNOWLEDGMENTS

Support by the Swiss National Science Foundation under Grant **FN 200021-126916/1** is gratefully acknowledged.

REFERENCES

1. H.K. Khalil. *Nonlinear Systems*. Prentice-Hall, Englewood Cliffs, N.J., 3rd edition, 2002.
2. H. Nijmeijer and A. van der Schaft. *Nonlinear Dynamical Control Systems*. Springer Verlag, New York, 1990.
3. P. Müllhaupt. Quotient submanifolds for static feedback linearization. *Systems & Control Letters*, 55:549–557, 2006.
4. I.A. Tall. State and feedback linearization of single-input control systems. *Systems & Control Letters*, 59:429–441, 2010.
5. S.S. Willson, P. Müllhaupt, and D. Bonvin. Quotient method for designing nonlinear control law. *50th IEEE Conf. on Decision and Control*, 2011.
6. S.S. Willson, P. Müllhaupt, and D. Bonvin. Quotient method for controlling the acrobot. *48th IEEE Conf. on Decision and Control*, 2009.
7. D. Ingram, S.S. Willson, P. Mullhaupt, and D. Bonvin. Stabilization of the cart-pendulum system through approximate manifold decomposition. *18th IFAC World Congress*, 2011.
8. J.F. Cañete, C. Galindo, and I.G. Moral. *System engineering and automation*. Springer Verlag, Berlin, 2011.
9. M. Krstic, P.V. Kokotovic, and I. Kanellakopoulos. *Nonlinear and Adaptive Control Design*. John Wiley & Sons, Inc., New York, NY, USA, 1995.
10. G.S. Miminis and C.C. Paige. An algorithm for pole assignment of time invariant linear systems. *International Journal of Control*, 35(2):341–354, 1982.