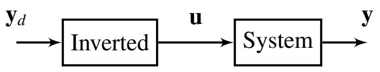
A Comparison of Non-Linear Dynamic Inversion and Inverse Simulation

Murray L. Ireland*, Thaleia Flessa† Douglas Thomson‡ and Euan McGookin.§
School of Engineering, University of Glasgow, Glasgow, UK, G20 8NZ

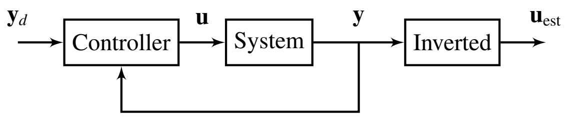
I. Introduction

The need to invert a system can arise in a number of areas of engineering. System inversion can be useful in control applications, permitting a linear mapping between output and controller and simplifying controller design [1, 2]. An inverted system can allow input trajectories to be determined for a desired output trajectory [3, 4]. This is useful when evaluating the feasibility of vehicle trajectories in the context of realistic input commands from both human and automatic controllers [5, 6, 7]. The ability to determine a system's input from its output also has use in fault detection and isolation methods [8] and model validation [9]. In the former case an inverted system model permits generation of input residuals in the same way that a conventional model can enable the generation of output residuals [10].

These applications may be categorised in two distinct structures. In the first, the inverted system is supplied with a desired trajectory $\mathbf{y}_d$ and subsequently determines the input $\mathbf{u}$ which will drive the system along this trajectory. This is illustrated in Figure 1a. This approach is most beneficial in the design of trajectories for autonomous and piloted vehicles. In the second structural form, the system receives its input $\mathbf{u}$ through conventional means, such as a feedback control system. The output $\mathbf{y}$ is then supplied to the inverted system with the goal of producing an estimate of the input $\mathbf{u}_{\text{est}}$. This is illustrated in Figure 1b. This approach is most beneficial in areas of fault detection and isolation, and validation.



(a) Inverted system as a method for calculating corresponding inputs for desired trajectories.



(b) Inverted system as a method for reconstructing input signals.

Figure 1. Structural variants for inverted system in relation to original system.

A system model may be inverted through different means. Systems which may be described in control affine form can be *analytically* inverted to provide an expression for input in terms of state and output. This approach is known as non-linear dynamic inversion (NDI) or feedback linearisation [1]. Alternatively, a system may be *numerically* inverted by employing the Newton-Raphson algorithm at discrete time increments. This method is known as Inverse Simulation (InvSim) [11]. Each method has advantages over the other in terms of accuracy, ease of use and feasibility. This paper compares the two approaches in application to an example non-linear system. Simulation results are presented which consider both of the structural forms presented in Figure 1.

II. Theoretical Background

Consider a generic, non-linear system, described in control affine form by

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \qquad \mathbf{y} = h(\mathbf{x}) \tag{1}$$

and represented in block diagram form in Figure 2. As stated previously, a capability for specifying the output $\mathbf{y}$ and obtaining the corresponding input $\mathbf{u}$ is useful in several applications. This relationship may be achieved by inverting the system, either analytically or numerically. It is possible to analytically invert any system which is considered control affine by first obtaining an explicit mapping between output and input. For complex systems, even those that are control affine, attainment of this mapping may be non-trivial. In contrast, numerical inversion is not constrained to control affine-only systems. It is, however, subject to considerations of numerical stability from which the analytical method may not suffer [12]. Additionally, as it uses an iterative Newton-Raphson algorithm, its processing time is typically greater than that of an analytical approach.

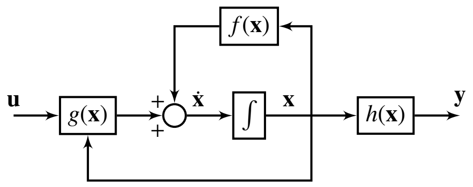


Figure 2. Structure of non-linear control affine system.

This section presents the theoretical background behind both NDI and InvSim, in the context of inverting a system of the form given in Equation (1).

A. Non-Linear Dynamic Inversion

Consider the system described by Equation (1). In order to invert this system and obtain an expression for the input, it is first necessary to obtain an expression for the output which explicitly includes the input. This is achieved by successively differentiating the output until this mapping is obtained. At the first derivative, the expression

$$\dot{\mathbf{y}} = L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} \tag{2}$$

is obtained [1]. The terms L_f and L_g are the *Lie-derivatives* in the directions of f and g, respectively, and are given by

$$L_f = f_1\frac{\partial}{\partial x_1} + \ldots + f_n\frac{\partial}{\partial x_n}, \qquad L_g = g_1\frac{\partial}{\partial x_1} + \ldots + g_n\frac{\partial}{\partial x_n}$$

for a state of length n. For the relationship given in Equation (2), the desired mapping is obtained only if $L_g h(\mathbf{x}) \neq 0$. If this condition does not hold, it is necessary to further differentiate Equation (2). The mapping ultimately occurs at the *relative degree* ν of the system, where the νth derivative of the output may be explicitly related to the input, as in the expression

$$\mathbf{y}^{(\nu)} = L_f^{\nu} h(\mathbf{x}) + L_g L_f^{\nu-1} h(\mathbf{x})\mathbf{u} \tag{3}$$

where the condition $L_g L_f^{\nu-1} h(\mathbf{x}) \neq 0$ must hold. The solution for input is then

$$\mathbf{u} = \left(L_g L_f^{\nu-1} h(\mathbf{x})\right)^{-1} \left(\mathbf{y}^{(\nu)} - L_f^{\nu} h(\mathbf{x})\right) \tag{4}$$

*Research Associate, School of Engineering, James Watt (South) Building, Murray.Ireland@glasgow.ac.uk
†PhD Student, School of Engineering, James Watt (South) Building.
‡Senior Lecturer, School of Engineering, James Watt (South) Building.
§Senior Lecturer, School of Engineering, James Watt (South) Building.

Substitution of Equation (4) into Equation (3) simply yields $\mathbf{y}^{(\nu)} = \mathbf{y}^{(\nu)}$. Similarly, substituting Equation (3) into Equation (4) yields $\mathbf{u} = \mathbf{u}$. In theory, the inverted system cancels out the non-linearities and dynamics of the system perfectly. In empirical testing, however, replication of the input is dependent on the smoothness of the supplied output signal. Additionally, systems with high relative degree can experience drift due to minute numerical errors. Figure 3 shows the structure of an analytically-inverted control affine system as used in simulation or experimental application.

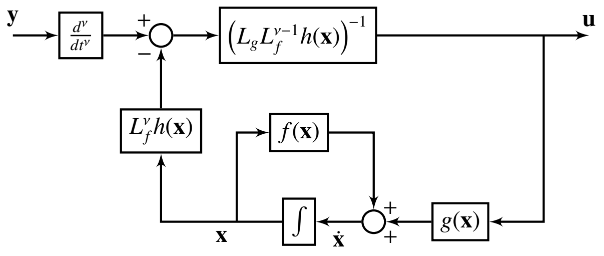


Figure 3. Structure of inverted non-linear, control affine system.

In practical implementation, the NDI algorithm receives an output $\mathbf{y}(t_k)$ at a discrete step k, with sample interval $h_{\text{samp}} = t_{k+1} - t_k$. The process model is then integrated between t_k and t_{k+1} at a step size $h_{\text{int}} = h_{\text{samp}}/N, N \in \mathbb{N}$. The accuracy and computational load of the NDI algorithm is thus dependent on these two step sizes, h_{samp} and h_{int}.

B. Inverse Simulation

Inverse Simulation provides a solution to Equation (1) for $\mathbf{u}$ using an iterative Newton-Raphson algorithm, which runs at discrete intervals during operation. It is similar to the NDI algorithm illustrated in Figure 3 in that it may act as an open-loop controller for the system. InvSim is initialised with approximate values for state and input, and then supplied with an output to track. As stated previously, this output may be either a desired trajectory or the true system output. At each time step t_k, the Newton-Raphson algorithm attempts to converge on a solution for input $\mathbf{u}(t_k)$, based on the supplied output $\mathbf{y}^{(a)}(t_k)$ and state and input from the previous interval. Unlike NDI, where the relative degree of the output is fixed by the equations of motion, the differential order a of the output supplied to InvSim may be chosen as desired. The value of a impacts the stability of the InvSim algorithm and must be chosen with care, as discussed in [4]. Upon the Newton-Raphson algorithm converging on a solution, the state and input are recorded and used to initialise the algorithm at the next interval. The most prominent InvSim solution is the *Genisa* algorithm [13], the behaviour of which is illustrated in Figure 4.

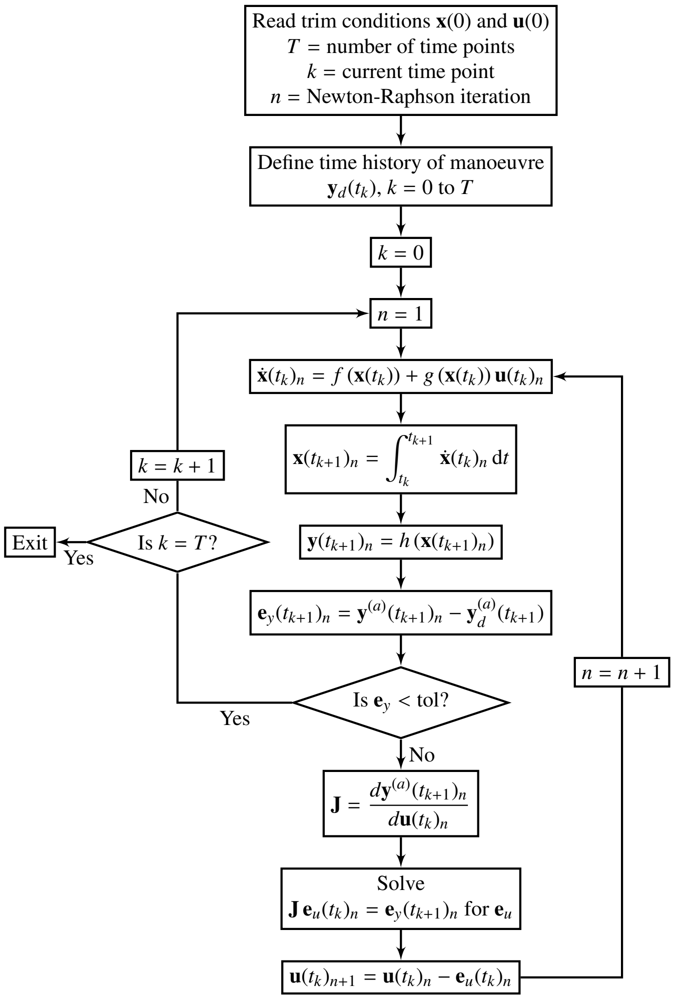
Like NDI, InvSim requires a model of the system in order to operate. It does not, however, require an analytical solution for $\mathbf{u}$ and is therefore not constrained to control affine systems. In practice, however, systems with a non-linear dependency can result in multiple solutions for input, for a given output.

Practical implementation of InvSim is similar to NDI in the discrete sampling of the output at each step k and the use of an integration loop to update the state. The sample interval is again denoted h_{samp} and the integration step size is defined $h_{\text{int}} = h_{\text{samp}}/N, N \in \mathbb{N}$. The accuracy and computational load of InvSim is partially dependent on these two parameters. Other parameters which affect the performance of the algorithm include the differential order a and the termination conditions of the Newton-Raphson algorithm.

III. A Candidate Non-Linear Model

The effectiveness of NDI and InvSim in inverting a control affine system is investigated. In this study, the following candidate model is used:

$$\dot{\mathbf{x}} = \begin{bmatrix} x_2 x_3 - x_1^2 \\ x_3 x_4 - x_2^2 \\ (x_1+1)u_1 - \frac{1}{10}x_1 x_3 \\ (x_2+1)u_2 - \frac{1}{10}x_2 x_4 \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} x_1 x_2 \\ x_2 \end{bmatrix} \tag{5}$$



Read trim conditions $\mathbf{x}(0)$ and $\mathbf{u}(0)$
T = number of time points
k = current time point
n = Newton-Raphson iteration
Define time history of manoeuvre
$\mathbf{y}_d(t_k)$, $k = 0$ to T
$k = 0$
$n = 1$
$\dot{\mathbf{x}}(t_k)_n = f(\mathbf{x}(t_k)) + g(\mathbf{x}(t_k))\,\mathbf{u}(t_k)_n$
$\mathbf{x}(t_{k+1})_n = \int_{t_k}^{t_{k+1}} \dot{\mathbf{x}}(t_k)_n \, dt$
$\mathbf{y}(t_{k+1})_n = h(\mathbf{x}(t_{k+1})_n)$
$\mathbf{e}_y(t_{k+1})_n = \mathbf{y}^{(a)}(t_{k+1})_n - \mathbf{y}_d^{(a)}(t_{k+1})$
Is $\mathbf{e}_y <$ tol?
Yes
No
$\mathbf{J} = \frac{d\mathbf{y}^{(a)}(t_{k+1})_n}{d\mathbf{u}(t_k)_n}$
Solve
$\mathbf{J}\,\mathbf{e}_u(t_k)_n = \mathbf{e}_y(t_{k+1})_n$ for $\mathbf{e}_u$
$\mathbf{u}(t_k)_{n+1} = \mathbf{u}(t_k)_n - \mathbf{e}_u(t_k)_n$
$n = n + 1$
$k = k + 1$
No
Is $k = T$?
Yes
Exit

Figure 4. Genisa Inverse Simulation algorithm.

This model is chosen such that it satisfies a number of criteria. First, it is control affine and conforms to Equation (1), where $f(\mathbf{x})$, $g(\mathbf{x})$ and $h(\mathbf{x})$ are non-linear. Second, a relative degree of $\nu = 2$ is specified. This value is sufficiently high that the complexity of the differentiated system and Lie derivatives are clear, but low enough that the concept may be followed easily. Additionally, the application of NDI and InvSim has predominantly been in control applications for mechanical systems, where the relative degree is low [4]. In terms of complexity, the system is chosen such that it has both linear and non-linear components in both the process and measurement models. The number of inputs and outputs are equal, a common constraint in applications of each method [7, 11, 14].

NDI requires an explicit mapping between output and input, which occurs at relative degree ν, corresponding to the νth derivative of the output. The first derivative of the output is found to be

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{x}_1 x_2 + x_1 \dot{x}_2 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \left(x_2 x_3 - x_1^2\right) x_2 + \left(x_3 x_4 - x_2^2\right) x_1 \\ x_3 x_4 - x_2^2 \end{bmatrix} \tag{6}$$

where the required dependency on input has not been obtained. As the relative degree of the candidate system is $\nu = 2$, the corresponding mapping is given by

$$\ddot{\mathbf{y}} = L_f^2 h(\mathbf{x}) + L_g L_f h(\mathbf{x})\mathbf{u} \tag{7}$$

where the Lie derivatives are given by

$$L_f^2h(\mathbf{x}) = \begin{bmatrix} 2x_1x_2\left(x_1^2 + x_1x_2 + x_2^2\right) + 3x_2x_3\left(x_3x_4 - x_2^2\right)\dots \\ \dots - \frac{21}{10}x_1x_3\left(x_1x_4 + x_2^2 + x_2x_4\right) \\ 2x_2^3 - \frac{21}{10}x_2x_3x_4 - \frac{1}{10}x_1x_3x_4 \end{bmatrix} \tag{8}$$
$$L_gL_fh(\mathbf{x}) = \begin{bmatrix} (x_1+1)\left(x_1x_4 + x_2^2\right) & (x_2+1)\,x_1x_3 \\ (x_1+1)\,x_4 & (x_1+1)\,x_3 \end{bmatrix}$$

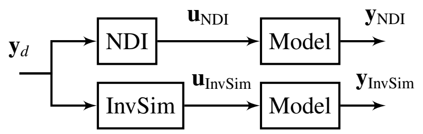
Construction of the inverted model is then simply achieved by substituting these Lie derivatives into the expression

$$\mathbf{u} = \left(L_gL_fh(\mathbf{x})\right)^{-1}\left(\ddot{\mathbf{y}} - L_f^2h(\mathbf{x})\right) \tag{9}$$

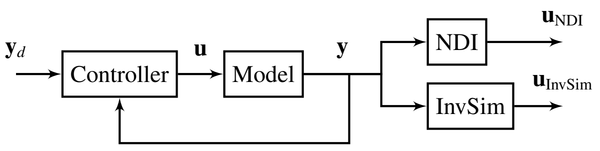
where $\ddot{\mathbf{y}}$ may be a desired trajectory or the actual trajectory of the system.

IV. Simulation Testing and Results

The accuracy and computational cost of NDI and InvSim may be compared in simulation. Two test cases are considered, corresponding to the different structures illustrated in Figure 1. These may be more explicitly defined in Figure 5.


$\mathbf{y}_d$
NDI
$\mathbf{u}_{\text{NDI}}$
Model
$\mathbf{y}_{\text{NDI}}$
InvSim
$\mathbf{u}_{\text{InvSim}}$
Model
$\mathbf{y}_{\text{InvSim}}$

(a) Test case 1, using inverted model as open-loop controller.



(b) Test case 2, using inverted model to estimate input.

Figure 5. Complete system structures for each test case.

In the first case, the system, here represented by the model given in Equation (5), is controlled in open-loop by either the NDI-derived inverse system or the InvSim numerically-inverted system. The goal in this instance is to ensure the model tracks the desired trajectory as closely as possible. The desired and actual trajectories are thus compared. In the second case, the model is controlled in closed-loop by a proportional-integral-derivative (PID) controller. The actual system output then provides the excitation for the inverted models. The true inputs are thus compared with the estimated inputs of each inversion method.

A. Setup

Inverse Simulation and NDI both require that the trajectory supplied to the inverted system is smooth [11], such that it represents a trajectory that the system could feasibly follow. For the case where the system structure resembles Figure 1b, the dynamics of the system itself help to ensure that the trajectory is smooth. For the case where the system is supplied with a desired trajectory, such as in Figure 1a, this trajectory must be carefully defined. A fifth-order polynomial function of time specifies the trajectory in this instance. This has the advantage of providing smooth derivatives up to the fifth order. The polynomials are distinct for each output and are defined as

$$\mathbf{y}_d = \begin{bmatrix} 5.94\times10^{-5} \\ 6.60\times10^{-5} \end{bmatrix} t^5 - \begin{bmatrix} 1.485\times10^{-3} \\ 1.650\times10^{-3} \end{bmatrix} t^4 + \begin{bmatrix} 9.9\times10^{-3} \\ 1.1\times10^{-2} \end{bmatrix} t^3 + \begin{bmatrix} 0.01 \\ 0.1 \end{bmatrix} \tag{10}$$

A feedback controller is also required for case 2. A PID law is used, defined by

$$\mathbf{u} = K_p\left(\mathbf{y}_d - \mathbf{y}\right) + K_i\int\left(\mathbf{y}_d - \mathbf{y}\right)\,\mathrm{d}t + K_d\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathbf{y}_d - \mathbf{y}\right) \tag{11}$$

where $K_p = 2$, $K_i = 0.2$ and $K_d = 2$ are the proportional, integral and derivative gains, respectively. The desired trajectory $\mathbf{y}_d$ is simply the vector of polynomials given in Equation (10).

The system model, NDI model and InvSim algorithm are initialised with the following conditions

$$\mathbf{x} = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} 0.01 \\ 0.1 \end{bmatrix} \tag{12}$$

In order to provide a consistent comparison between the two methods, the sample step h_{samp} and integration step h_{int} are the same for each algorithm. These, and other key parameters are defined in Table 1. A fourth-order Runge-Kutta algorithm is used as the integration method for both algorithms and the system model.

Table 1. NDI and InvSim settings.

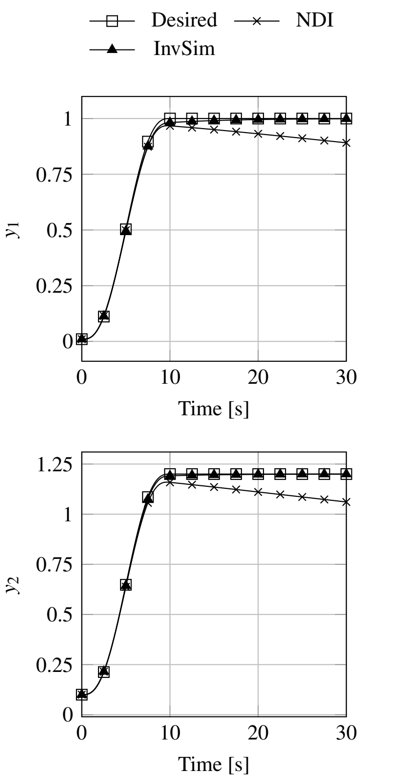
Parameter	Value
Sample interval, h_{samp}	0.01
Integration step, h_{int}	0.001
InvSim differential order, a	1
Newton-Raphson termination tolerance, tol	1×10^{-6}

B. Case 1: Open-loop control with inverted system

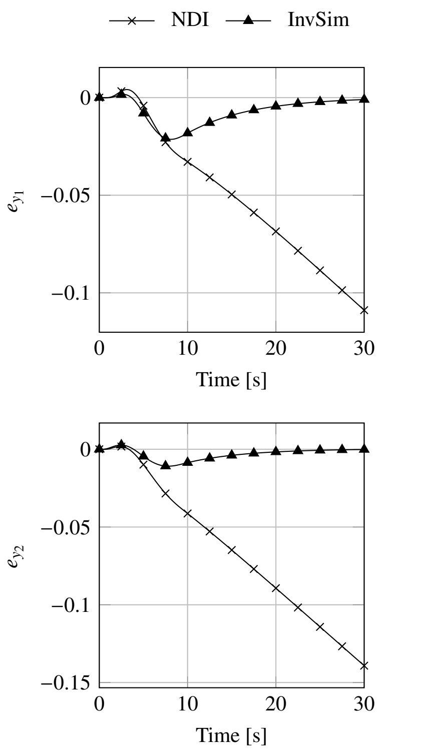
In the first case, the inverted system acts as an open-loop controller for the system, receiving a desired trajectory $\mathbf{y}_d$ and providing the inputs necessary to drive the system along this trajectory. Figure 6a shows the system outputs for each method, $\mathbf{y}_{\text{NDI}}$ and $\mathbf{y}_{\text{InvSim}}$, in comparison to the desired output. In the case of the InvSim algorithm, the output is supplied at differential order $a = 1$. The deviation of each output from the desired result may be seen more clearly in Figure 6b, where $\mathbf{e}_y = \mathbf{y} - \mathbf{y}_d$. The InvSim-derived input results in an output which deviates from the desired output during the transient phase before converging upon it again during the steady-state phase. Conversely, the NDI-derived input results in an error of linearly increasing magnitude during the steady-state phase.

The superior performance of the InvSim algorithm in tracking the desired trajectory may be attributed primarily to its exciting signal. The InvSim algorithm receives the desired trajectory in the form of its first derivative $\dot{\mathbf{y}}_d$, while the NDI solution must receive the output at the νth derivative, such that $L_gL_f^{\nu-1}h(\mathbf{x}) \neq 0$. Here, as $\nu = 2$, the increase in differential order of the NDI exciting signal results in the demonstrated drift. When the differential order of the InvSim exciting signal is chosen to be $a = 2$, this linear drift is also evident in the output of the InvSim-driven output. The errors for this case are shown in Figure 6c. Such drift errors may be minimised by employing as low a derivative of the output as possible.

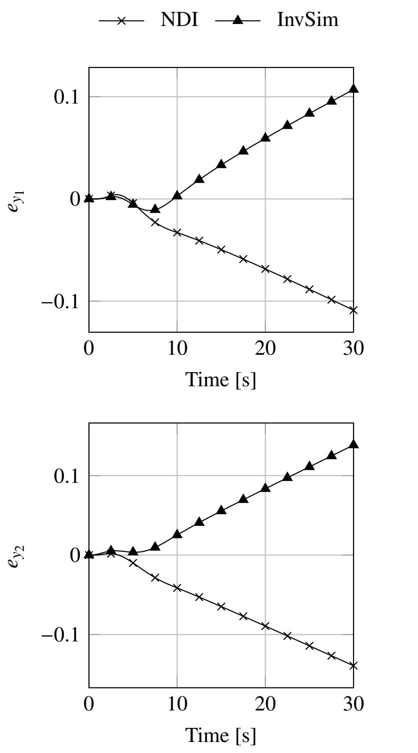
While InvSim is shown to provide superior tracking accuracy for the presented case, this comes at a cost. Where NDI obtains the input from symbolic inversion of the system model, InvSim achieves the same result through an iterative approach. NDI requires only a single state update between each sample step, in addition to solving the expression given in Equation (4). InvSim, however requires several calls of the system process and measurement models. With reference to the flowchart in Figure 4: for each iteration n, the process and measurement models are updated once to provide data for the error $\mathbf{e}_y$ and $2p$ times to populate the Jacobian $\mathbf{J}$ using central differencing, where p is the number of inputs. That is, twice per input: once for a positive perturbation and once for a negative perturbation. On the last iteration $n = n_{\max}$, the error tolerance is satisfied and the loop exits after calling both models a single time. Thus, the system model is called $(2p+1)(n_{\max}-1)+1$ times per time step in InvSim, in comparison to once for NDI. This result is generally applicable to any use of the Genisa algorithm as used here. The number of iterations per step has means $n_{\max} = 1.405$ for $a = 1$ and $n_{\max} = 2.015$ for $a = 2$. Empirical testing finds NDI to be approximately 3.1 times as fast as InvSim when $a = 1$ and 5.9 times as fast when $a = 2$.



(a) Outputs of model as driven by NDI- and InvSim-derived inputs, compared with desired trajectory. The InvSim exciting signal is supplied at differential order $a = 1$.



(b) Difference between model outputs for each method and desired trajectory. The InvSim exciting signal is supplied at differential order $a = 1$.



(c) Difference between model outputs for each method and desired trajectory. The InvSim exciting signal is supplied at differential order $a = 2$.

Figure 6. Results for Case 1.

C. Case 2: Input estimation from true output

In the second case, the inverted system is intended to estimate the true inputs of the system, receiving the actual system outputs $\mathbf{y}$. The system itself is driven in closed-loop by a PID controller. The resulting system output is shown in comparison to the reference trajectory in Figure 7.

Figure 8a shows the estimated inputs of each method in comparison to the true system inputs, supplied by the PID controller. Figure 8b shows the deviation of the estimated inputs from the true input, where $\mathbf{e}_u = \mathbf{u}_{\mathrm{est}} - \mathbf{u}$. Note that $a = 1$ for these results. Again, a linear drift is evident in the NDI result during the system's steady-state phase. In contrast, the InvSim-estimated input demonstrates an error an order of magnitude less during this phase.

Once again, the superior accuracy of InvSim may be attributed due to NDI's reliance on higher-order derivatives of the output as a driving signal. This is exemplified in Figure 8c, which shows the same errors when $a = 2$. The InvSim-estimated input now exhibits a drift error of similar gradient to that of the NDI-estimated input.

Analysis of the respective runtimes of each algorithm for this case yields results similar to those of the first case. The mean iterations for the settings $a = 1$ and $a = 2$ are found to be $n_{\max} = 1.540$ and $n_{\max} = 1.999$, respectively. The NDI algorithm is found to be comparatively faster than the InvSim algorithm at equivalent sample and integration step sizes. Here, it is found that NDI is approximately 3.9 and 6.2 times as fast as InvSim for the cases $a = 1$ and $a = 2$, respectively.

D. Analysis of results

In both cases, it is clear that NDI is the faster algorithm, lacking the iterative behaviour of InvSim. However, its reliance on a higher-order derivative as a forcing term results in the drifting error shown in Figures 6 and 8. Use of an equivalent-order forcing term for InvSim yields a

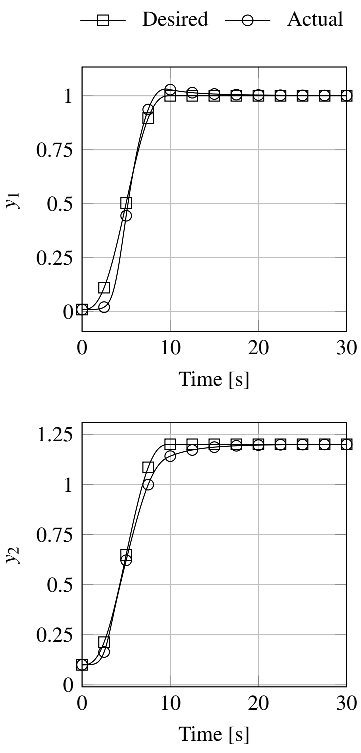
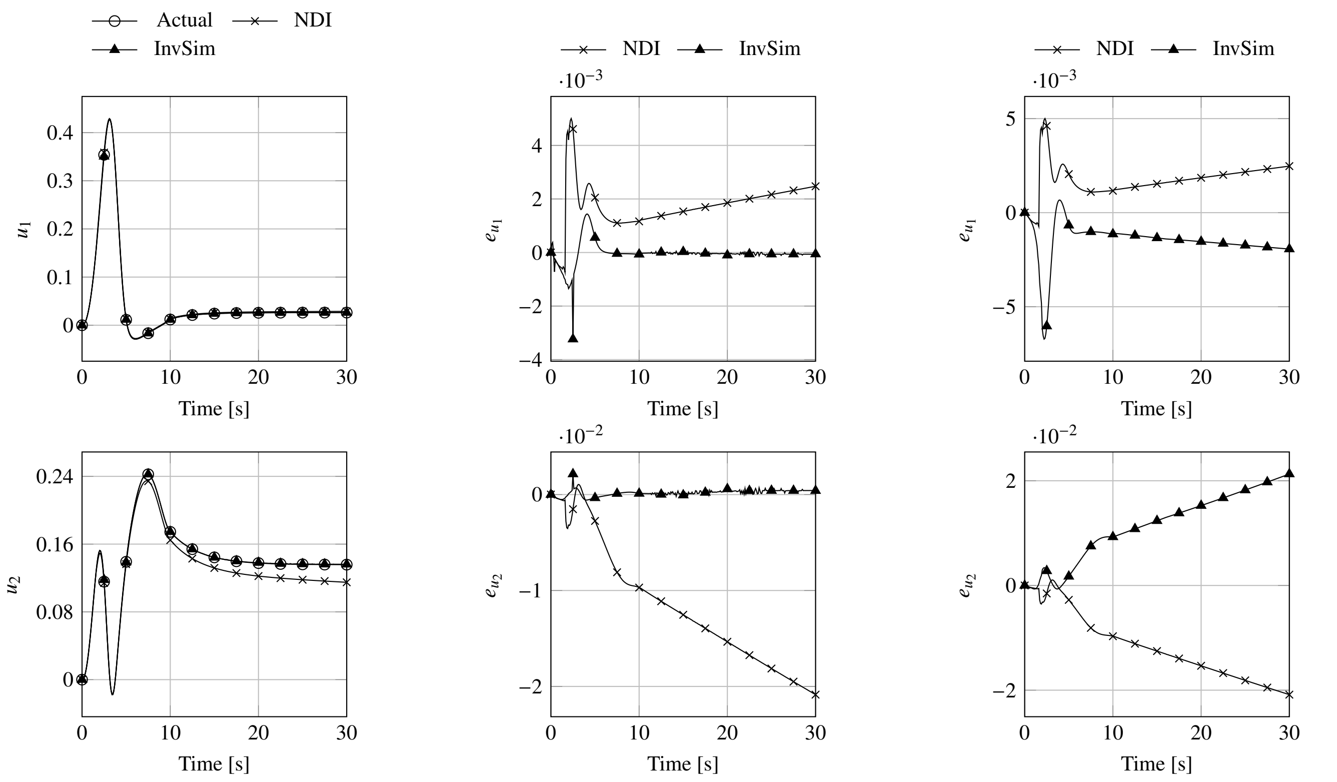


Figure 7. System output in comparison to desired trajectory, using closed-loop PID control.



(a) Estimated inputs from NDI and InvSim, compared to true system inputs. The InvSim exciting signal is supplied at differential order $a = 1$.

(b) Difference between estimated inputs for each method and actual system inputs. The InvSim exciting signal is supplied at differential order $a = 1$.

(c) Output of model, in comparison to desired trajectory supplied to PID controller. The InvSim exciting signal is supplied at differential order $a = 2$.

Figure 8. Case 2 results.

comparable drift in the error. Additionally, the InvSim algorithm is found to run slower for the higher value of a, due to the greater number of iterations required to converge on an acceptable solution.

V. Conclusions

Some conclusions may be drawn on the performance of the NDI and InvSim algorithms on the presented system. First considering the simulation results, it is clear that InvSim demonstrates greater accuracy in both cases, at the cost of greater computational expense. The drift error in the NDI results may be reduced by decreasing its integration step. This has the effect of increasing the runtime and reducing its advantage over InvSim. The ultimate advantage of one method over the other in offline use is strongly dependent on the system, its relative degree and its complexity.

For real-time applications, the opposing requirements of accuracy and runtime must be balanced. With NDI, the runtime is consistent each time the algorithm is executed. This makes implementation straightforward, as the sample and integration steps may be chosen such that the algorithm is guaranteed to run in real-time. As stated previously, however, these properties determine the accuracy of the NDI approach. NDI is thus ultimately limited in its accuracy by the computational capability of the available hardware. In contrast, the runtime of the InvSim algorithm varies with the number of iterations required for convergence and therefore may vary with each execution. This makes implementation of InvSim in real-time more problematic, as the maximum possible runtime must be considered when selecting sample and integration steps. Additionally, these properties ultimately impact the convergence of the algorithm, thus changing them may result in an increase in iterations and therefore runtime. A potential solution is to specify a small iteration limit, however this can reduce the accuracy of the algorithm. It is worth noting that number of iterations for the presented results rarely exceeded 2 and was never greater than 3 for any case.

This greater complexity in implementing InvSim over NDI is offset by InvSim's more flexible nature. A change in model is automatically handled by InvSim, with potential alterations in the algorithm settings. Conversely, any model change requires that NDI's inverse model be redefined and redeployed. This can ultimately be just as time consuming as the temporal adjustments required by InvSim. Furthermore, practical implementation is almost guaranteed to require such model changes.

The flexibility of InvSim may be extended to other systems. As a system grows in complexity, its inversion through analytical means becomes less trivial. InvSim differs from this in that, when using the Genisa algorithm, it is essentially decoupled from the system itself. The system model may simply be swapped for another. The only consequent changes required by InvSim are then those to settings such as time step and differential order. Additionally, while the differential order of the driving signal does impact the stability and accuracy of the algorithm, it is not tied to the relative degree of the system as it is when using NDI. Thus, the differential order in InvSim may be selected to achieve a balance between demands on minimising drift error and ensuring the stability of the algorithm. Conversely, the drift error in NDI may be minimised only be reducing the integration step. Finally, InvSim is also able to facilitate systems that are not control affine, while NDI cannot without some linearisation of the system. As the majority of systems are not control affine in reality, this ability is a major advantage of InvSim in practical implementation.

Acknowledgments

The work in this note was supported by a UK Space Agency CREST 3 award.

Data

The data presented in this note is available online [15].

References

[1] T. Glad and L. Ljung, *Control Theory: Multivariable and Nonlinear Methods*. London: CRC Press, 2000.

[2] M. Ireland, A. Vargas, and D. Anderson, “A Comparison of Closed-Loop Performance of Multirotor Configurations Using Non-Linear Dynamic Inversion Control,” *Aerospace*, vol. 2, no. 2, pp. 325–352, 2015. [Online]. Available: http://www.mdpi.com/2226-4310/2/2/325/

[3] R. Hess, C. Gao, and S. Wang, “A generalized technique for inverse simulation applied to aircraft manoeuvres,” *Journal of Guidance, Control and Dynamics*, vol. 14, no. 5, pp. 920–926, 1991.

[4] D. Thomson and R. Bradley, “Inverse simulation as a tool for flight dynamics research – Principles and applications,” *Progress in Aerospace Sciences*, vol. 42, no. 3, pp. 174–210, May 2006.

[5] K. Worrall, D. Thomson, E. McGookin, and T. Flessa, “Autonomous Planetary Rover Control using Inverse Simulation,” in *13th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2015)*. Noordwijk: ESA/ESTEC, May 2015.

[6] D. Murray-Smith and E. McGookin, “A case study involving continuous system methods of inverse simulation for an unmanned aerial vehicle application,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 229, no. 14, pp. 2700–2717, 2015. [Online]. Available: http://pig.sagepub.com/lookup/doi/10.1177/0954410015586842

[7] D. G. Thomson and R. Bradley, “The principles and practical application of helicopter inverse simulation,” *Simulation Practice and Theory*, vol. 6, no. 97, pp. 47–70, 1998.

[8] M. L. Ireland, K. J. Worrall, R. Mackenzie, T. Flessa, E. McGookin, and D. Thomson, “A Comparison of Inverse Simulation-Based Fault Detection in a Simple Robotic Rover with a Traditional Model-Based Method,” in *19th International Conference on Autonomous Robots and Agents (ICARA 2017)*. Madrid: ICARA, March 2017.

[9] R. Bradley, G. D. Padfield, D. J. Murray-Smith, and D. G. Thomson, “Validation of helicopter mathematical models,” *Transactions of the Institute of Measurement and Control*, vol. 12, no. 4, pp. 186–196, 1990.

[10] R. Isermann, *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*. Berlin: Springer-Verlag, 2006. [Online]. Available: http://link.springer.com/10.1007/3-540-30368-5

[11] D. J. Murray-Smith, “The inverse simulation approach: a focused review of methods and applications,” *Mathematics and Computers in Simulation*, vol. 53, no. 4-6, pp. 239–247, October 2000.

[12] T. Flessa, E. W. McGookin, and D. G. Thomson, “Numerical stability of inverse simulation algorithms applied to planetary rover navigation,” in *24th Mediterranean Conference on Control and Automation, MED 2016*, Athens, 2016.

[13] S. Rutherford and D. Thomson, “Improved methodology for inverse simulation,” *Aeronautical Journal*, vol. 100, no. 993, pp. 79–85, 1996.

[14] A. Das, K. Subbarao, and F. Lewis, “Dynamic inversion with zero-dynamics stabilisation for quadrotor control,” *Control Theory & Applications, IET*, vol. 3, no. 3, pp. 303–314, 2009.

[15] M. L. Ireland, T. Flessa, D. Thomson, and E. McGookin, “A comparison of non-linear dynamic inversion and inverse simulation,” 2017. [Online]. Available: http://dx.doi.org/10.5525/gla.researchdata.413