Journal of Computer Engineering
ISSN 10003428

Original Research Article
Vol 9, Issue 4, 2020 | ISSN: 1000-3428

Comparative Analysis of Simulation-Based and Model-Based Fault Detection in Robotic Systems

Elliot R. Jenkins, Ava S. Patel, Maya N. Khan, Julianne L. Fraser, Liam C. Mitchell, Ethan R. Douglas

Department of Mechanical Engineering, University of Bristol, Bristol, UK; Centre for Robotics and Autonomous Systems, University of Melbourne, Melbourne, Australia

Abstract—Robotic rovers which are designed to work in extra-terrestrial environments present a unique challenge in terms of the reliability and availability of systems throughout the mission. Should some fault occur, with the nearest human potentially millions of kilometres away, detection and identification of the fault must be performed solely by the robot and its subsystems. Faults in the system sensors are relatively straightforward to detect, through the residuals produced by comparison of the system output with that of a simple model. However, faults in the input, that is, the actuators of the system, are harder to detect. A step change in the input signal, caused potentially by the loss of an actuator, can propagate through the system, resulting in complex residuals in multiple outputs. These residuals can be difficult to isolate or distinguish from residuals caused by environmental disturbances. While a more complex fault detection method or additional sensors could be used to solve these issues, an alternative is presented here. Using inverse simulation (InvSim), the inputs and outputs of the mathematical model of the rover system are reversed. Thus, for a desired trajectory, the corresponding actuator inputs are obtained. A step fault near the input then manifests itself as a step change in the residual between the system inputs and the input trajectory obtained through inverse simulation. This approach avoids the need for additional hardware on a mass- and power-critical system such as the rover. The InvSim fault detection method is applied to a simple four-wheeled rover in simulation. Additive system faults and an external disturbance force and are applied to the vehicle in turn, such that the dynamic response and sensor output of the rover are impacted. Basic model-based fault detection is then employed to provide output residuals which may be analysed to provide information on the fault/disturbance. InvSim-based fault detection is then employed, similarly providing *input* residuals which provide further information on the fault/disturbance. The input residuals are shown to provide clearer information on the location and magnitude of an input fault than the output residuals. Additionally, they can allow faults to be more clearly discriminated from environmental disturbances.

I. Introduction

PLANETARY rovers are unique among the many robotic explorers humanity has launched into space in that they alone have lived and worked for months on other worlds, gathering crucial information and transmitting it to Earth. Although the first successful lunar rover mission was preceded by the Apollo 11 landing [1], successful rover missions to Mars have been occurring for almost two decades: well in advance of any manned mission to the red planet. In addition to the lack of human assistance, planetary exploration also offers two additional challenges [2]. First, the transmission latency between the planet and Earth can make remote control of the rover slow at best or completely infeasible at worst. This highlights the need for at least semi-autonomous capability in the rover. Second, many planetary environments present their own problems such as unshielded solar radiation, rough terrain and poor visibility.

The rover must be designed to deal with both these environmental factors and its mission requirements. This clearly results in a vehicle of great complexity, with consideration in the design for not only known variables, but also for predicted variables. Thus, while the rover must contend with the challenges of its mission and environment, it must also handle problems within its own hardware and software: particularly the occurrence and impact of faults.

The detection of and reaction to faults is considered in the field of fault detection, isolation and recovery (FDIR). The first stage in the FDIR process, fault detection, concerns identifying when a fault has occurred. The second, fault isolation or diagnosis, involves locating the fault within the system and providing information on it such that it can be compensated for during the recovery phase. The FDIR technique of *model-based fault detection and isolation* is a popular method for detecting and locating such faults. In its simplest form, it involves comparing the outputs of a fault-afflicted system with the outputs of a fault-free mathematical model of the system [3]. The error between the true outputs and the model outputs is known as the *residual*. Both systems are supplied with identical input signals, thus any discrepancies in the residual may be attributed to any combination of the following phenomena: Unmodelled behaviours; environmental or internal disturbances; and system faults. Output residuals are typically sufficient when detecting and isolating sensor faults at the output. Hardware-based faults earlier in the system, however, can manifest in multiple output residuals, making these faults more difficult to isolate. State observers, output observers and the Kalman filter may be used to determine other

How to cite: Elliot R. Jenkins, Ava S. Patel, Maya N. Khan, Julianne L. Fraser, Liam C. Mitchell, Ethan R. Douglas (2020). Comparative Analysis of Simulation-Based and Model-Based Fault Detection in Robotic Systems. Journal of Computer Engineering, 9(4), 47–59. DOI: https://doi.org/10.5281/zenodo.19341597

system variables from which to generate residuals [3]. These residuals may then provide clearer results on the fault location and severity. Alternatively, more complex methods may be used to detect and isolate faults, such as particle filters [4], [5].

A set of variables not typically used in residual generation are the system inputs. This is because they are usually required as information by any model-based fault detection method.

One method by which input residuals may be generated is *inverse simulation* (InvSim). This is because InvSim receives the output trajectory (or desired trajectory) of a system and provides the corresponding input signals. The true system inputs are thus not required. The inputs produced by InvSim are then compared to the true system inputs to provide input residuals. Faults earlier in the system should be easier to isolate using input rather than output residuals. This concept is investigated in this paper by applying, in sequence, a sensor fault, actuator fault and disturbance to a simulated planetary rover.

The structure of the paper is as follows. Section II describes the process of InvSim and its application to fault detection in greater detail. Section III describes the mathematical model of the rover. Section IV describes simulation testing of the fault detection and isolation algorithm. This testing utilises the model of Section III to both represent the real system and the software model used in model-based and InvSim-based FDIR. The only difference between the models is the presence of faults in the model representing the real system. Finally, Section V states the conclusions of the research.

II. Overview of Inverse Simulation and Application to Fault Detection

The intended goal of inverse simulation is intuitive. Where a conventional simulation employs a system model which receives inputs and provides a corresponding set of outputs, InvSim does the opposite. For a given set of outputs, InvSim attempts to determine the inputs which will result in the system producing these outputs. This technique is most often used in the context of identifying trajectories which a mechanical system can feasibly follow. Examples of this application include the helicopter [6]–[9], autonomous underwater vehicle [10], unmanned aerial vehicle [11] and robotic rover [12], [13]. It differs from analytical approaches to system inversion such as non-linear dynamic inversion in that it may consider systems which contain discontinuities or are not *control-affine*. InvSim has also been used in model validation [8], [9].

Inverse simulation essentially operates as follows. A temporal loop iterates in discrete time intervals in much the same way a conventional simulation does. Thus, each iteration k of the loop describes a discrete point in time t_k. Where a conventional simulation would determine the state derivatives at each time step and then integrate them to obtain the states and outputs of the next increment, InvSim instead uses a Newton-Raphson algorithm to determine the states and inputs from the outputs. This algorithm operates on a non-linear equation of the form $y(t_{k+1}) = F(x(t_k),u(t_k))$ and attempts to find a solution for $u(t_k)$ which results in a negligible error between the output $y(t_{k+1})$ and a desired output $y_d(t_{k+1})$. The Newton-Raphson algorithm iterates until this error is below a specified tolerance, at which point the InvSim algorithm progresses to the next time step.

A number of InvSim algorithms exist. The method employed in this paper is the *Integration* algorithm, so named because it employs numerical integration to obtain the outputs from the inputs on each iteration of the Newton-Raphson loop. More specifically, the methodology used is the *Genisa* algorithm [14]. This algorithm is best described visually, shown in Fig. 1.

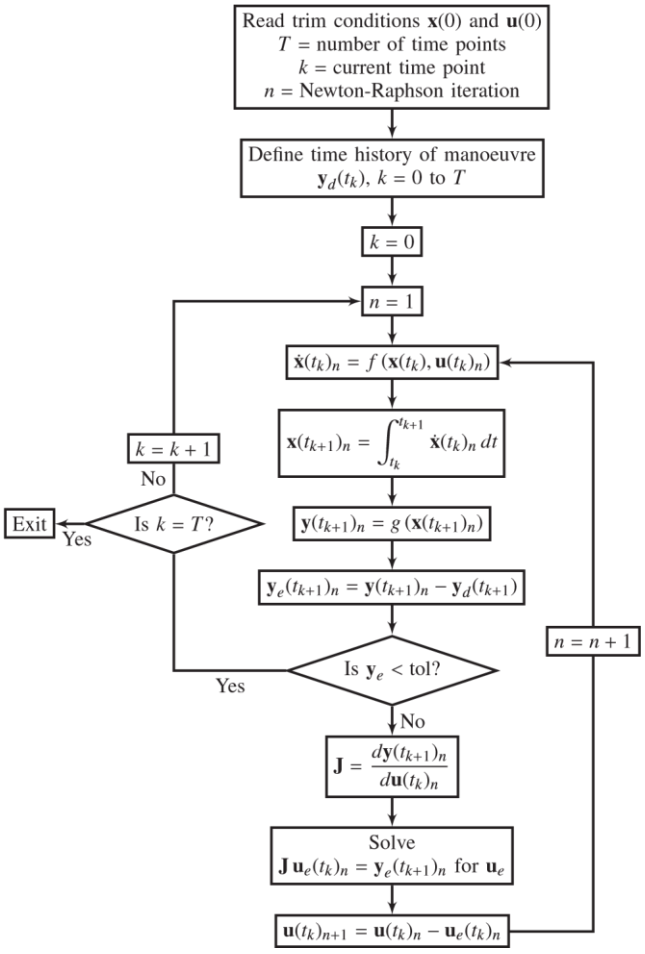


Fig. 1 Genisa inverse simulation algorithm

The application of InvSim described in this paper differs from the aforementioned studies in that the desired trajectory supplied to the InvSim algorithm is not a path of unknown feasibility. Just as the pairing of parity equations

with a conventional model uses the inputs of the actual system to obtain a residual between real output and modelled output, their use with InvSim provides a residual between real inputs and InvSim-derived inputs from supplying the outputs of the actual system. Just as faults near the output are more easily detected than those near the input when using parity equations on a conventional model, the reverse is true when using InvSim.

This approach may be proven mathematically for a linear system. Consider a system described by the transfer function

$$G(s) = \frac{y(s)}{u(s)} \quad (1)$$

The system is subject to a fault at the input f_u or a fault at the output f_y. The output y of the system is thus related to the input by u by the expression $y(s) = G(s)[u(s) + f_u(s)] + f_y(s)$ (2)

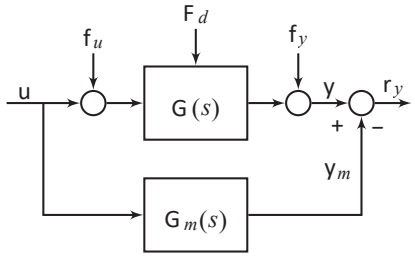


Fig. 2 Residual generation using system and model outputs

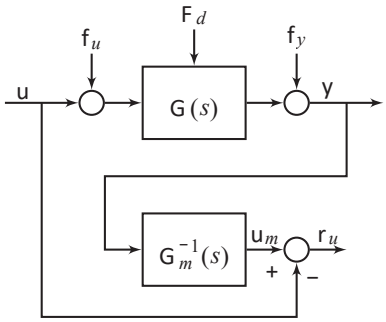


Fig. 3 Residual calculation using estimated input from model in InvSim

Model-based fault detection using output residuals, illustrated in Fig. 2, uses a model of the system G_m to provide an expected output y_m for the same input. The output residual may thus be calculated $r_y(s) = y(s) - y_m(s)$

$$= G(s)[u(s) + f_u(s)] + f_y(s) - G_m(s)u(s) \quad (3)$$

$= G_m(s)f_u(s) + f_y(s)$ for the case where the model exactly matches the system, that is $G_m(s) = G(s)$. Where only an output fault has occurred, the output residual is simply $r_y(s) = f_y(s)$, making detection and isolation of the fault a trivial problem. Where only an input fault has occurred, the residual is $r_y(s) = G(s)f_u(s)$. This is clearly less trivial to solve, as the residual describes the fault as propagated through the system dynamics.

Consider the approach used in InvSim-based fault detection, illustrated in Fig. 3. The numerical InvSim process may be represented analytically by $u_m(s) = G^{-}{}_m{}^1(s)y(s)$. The system is similarly solved for the input. Considering again an input fault only, the input residual may be defined as $r_u(s) = u_m(s) - u(s)$

$$= G^{-}{}_m{}^1(s)y(s) - G^{-1}(s)y(s) - f_u(s)$$

$$= G_{-m^1}(s) - G_{-1}(s)y(s) + f_u(s) \quad (4)$$

$$= f_u(s)$$

for the case where $G_m(s) = G(s)$. The input residual thus measures the input fault directly, making location and estimation of severity a trivial issue.

Both Figs. 2 and 3 also show the presence of a disturbance F_d. The influence of this disturbance on the system is described in the next section, while its effects on the fault detection and isolation process are highlighted in Section IV.

Fig. 4 Lynxmotion 4WD3 rover

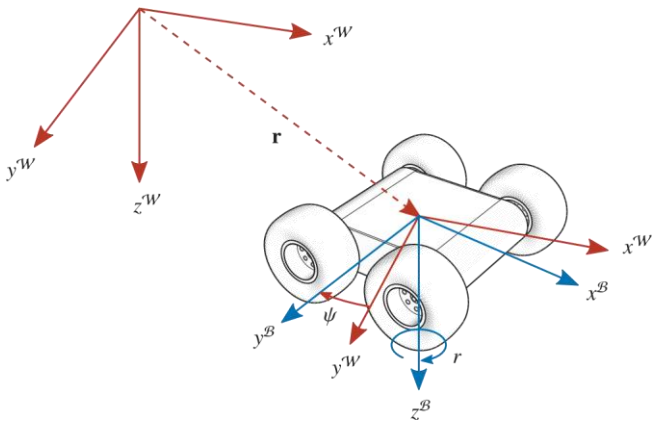


Fig. 5 World and rover body frames of reference. Note that roll and pitch variables are omitted as the rover operates on flat, level terrain

III. Rover Model

A simple mathematical model of a robotic rover is used in both the simulation which represents the real system and the fault detection solutions. The system model and fault detection model differ only in that the former can include faults while the latter neglects them. This model includes actuator dynamics, damping forces and wheel slippage. The modelled environment is very simple, considering a flat, level terrain. The model is validated against experimental data from a Lynxmotion 4WD3 rover (Fig. 4), as described in [15].

A. Rigid Body Model

The rover is represented as a rigid body with position $\mathrm{r} = [x,y,z]^T$ and orientation $\eta = [\varphi,\theta,\psi]^T$ in an inertially fixed reference frame W, shown in Fig. 5. The orientation of W is largely arbitrary, aside from the requirement that z^W points in the direction of the local gravity vector. A body-fixed frame B is defined such that its origin is at the centre of mass of the rover, the x^B-axis is positive in the nominal forward direction of the vehicle, y^B is positive in the starboard direction and z^B is positive downward and collinear with z^W when the rover is on level terrain.

The inertial position of the rover is related to its velocity $\mathrm{v} = [u,v,w]^T$ in B by

$$\dot{\mathrm{r}} = \mathrm{R}^W_B \mathrm{v} \tag{5}$$

The direction cosine matrix R^W_B describes the transformation from B to W and is given by

$$\mathrm{R}^B_W = \begin{bmatrix} c\varphi c\theta & -s\varphi c\psi + c\varphi s\theta s\psi & s\varphi s\psi + c\varphi s\theta c\psi \\ s\varphi c\theta & c\varphi c\psi + s\varphi s\theta s\psi & -c\varphi s\psi + s\varphi s\theta c\psi \\ -s\theta & c\theta s\psi & c\theta c\psi \end{bmatrix} \tag{6}$$

where s denotes sinφ, c denotes cosθ and so forth. The reverse transformation, from W to B is simply obtained from the transpose, that is $\mathrm{R}^B_W = \mathrm{R}^W_B$.

The orientation of the rover is similarly related to its angular velocity $\omega = [p,q,r]^T$, also described in B. This relationship is

$$\dot{\eta} = \mathrm{R}_\eta \omega$$

$$\mathrm{R}_\eta = \begin{bmatrix} 1 & \sin\varphi\tan\theta & \cos\varphi\tan\theta \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi\sec\theta & \cos\varphi\sec\theta \end{bmatrix} \tag{7}$$

The rigid body response of the rover is obtained from the Newton-Euler formalism, where the linear and angular velocities of a rotating frame of reference, here the body frame B, are related to a net force and moment, respectively. Rearranging the Newton-Euler rigid body equations such that they resemble a non-linear state-space model yields

$$\dot{\mathrm{v}} = \frac{1}{m}\mathrm{F} - \omega \times \mathrm{v} \tag{8}$$

$$\dot{\omega} = \mathrm{I}^{-1}(\mathrm{M} - \omega \times \mathrm{I}\omega)$$

where m is the rover mass and I is the inertia tensor. The net force F and net moment M are both described in B.

B. Force and Moment Contributions

The force and moment are comprised of propulsive (p), aerodynamic (a), frictional (f) and disturbance (d) components, that is

$$\begin{aligned} \mathrm{F} &= \mathrm{F}_p + \mathrm{F}_a + \mathrm{F}_f + \mathrm{F}_d \\ \mathrm{M} &= \mathrm{M}_p + \mathrm{M}_f + \mathrm{M}_d \end{aligned} \tag{9}$$

The composition of these elements are described as follows.

1) Propulsion: A propulsive component is produced by the driving force of the wheels, as shown in Fig. 6. This is also used to control the vehicle; surge velocity by the net force and heading by the differential between each side. Each motor j produces a torque $\tau_j, j \in \{1,2,3,4\}$, as described in Section III-C. The force $F_{w,j}$ produced by each wheel is simply obtained with consideration of the wheel radius R_w, giving τ_j

$$F_{w,j} = \frac{\tau_j}{R_w} \tag{10}$$

The net propulsive force is then

$$\mathrm{F}_p = \sum_{i=j}^{4} F_{w,j} \cdot \begin{bmatrix} \cos\beta \\ \sin\beta \\ 0 \end{bmatrix} \tag{11}$$

where β is the slip angle [16]. It is obtained through consideration of the elements of the velocity vector, that is $\mathrm{v} = [u,v,w]^T$.

$$\beta = \arcsin\frac{v}{\mathrm{v}} \tag{12}$$

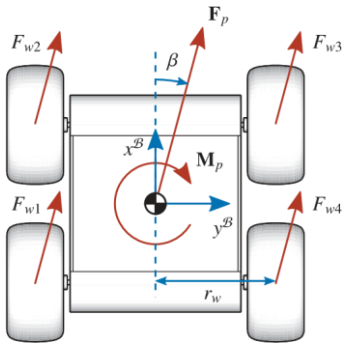


Fig. 6 Rover plan view with wheel forces and numbering

where v is the magnitude of the speed vector.

The differential forces in the wheels give rise to a yawing moment. As each wheel has identical moment arm r_w about the yaw axis, the net propulsive moment may be given by

$$\mathrm{M}_p = \sum_{i=j}^{4} r_w F_{w,j} d_j \cdot \hat{\mathrm{z}} \tag{13}$$

where d_j is an element of $\mathrm{d} = [1,1,-1,-1]^T$ and determines the direction of the yawing moment produced by each wheel and $\hat{\mathrm{z}}$ is the unit direction vector the local z-axis.

2) Aerodynamic Drag: The rover from which the model is derived is designed to operate in Earth's atmosphere, thus aerodynamic drag will occur. The drag force resists the motion of the rover and is thus dependent on the magnitude and direction of the speed vector, giving

$$\mathrm{F}_a = -\frac{1}{2}\rho C_d \mathrm{v}^2 \cdot A_{xy} \sin\beta \tag{14}$$

where ρ is the atmospheric density, C_d is the aerodynamic drag coefficient and A_x and A_y are the surface areas projected in the x- and y-axes, respectively. Torsional drag is assumed to be negligible.

3) Friction: As the rover uses wheels for locomotion, rolling friction must be considered [15]. The rolling friction is dependent on the weight of the rover, the velocity in each degree of freedom and a frictional coefficient σ. The resulting frictional forces and moments are then given by

$$\mathrm{F}_f = -mg \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \end{bmatrix} \mathrm{v}_\sigma$$

$$\mathrm{M}_f = -mgr_w \begin{bmatrix} \sigma_p \\ \sigma_q \\ \sigma_r \end{bmatrix} \omega_\sigma \tag{15}$$

4) Disturbance Force: A disturbance force is included in the system model but not the software model used by the fault detection algorithms. This force F_d acts at a position r_d with respect to the body-fixed frame B, as illustrated in Fig. 7. The consequent disturbance torque is thus simply $\mathrm{M}_d = \mathrm{r}_d \times \mathrm{F}_d$.

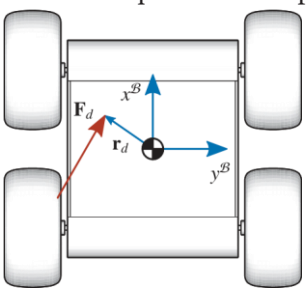


Fig. 7 Disturbance force acting at point on rover body

C. Motor Dynamics

The Lynxmotion 4WD3 employs four DC motors, wired in parallel such that motors on a single side receive the same voltage. There are consequently two control inputs, corresponding to each side of the rover.

The dynamics of the motor are composed of electrical and mechanical components. The electrical component describes the behaviour of the current i_j in the circuit of a motor $j = \{1,2,3,4\}$ in response to a voltage input. This is given by

$$\dot{i}_j = \frac{1}{L}\left(V_j - Ri_j - K_e\Omega_j\right) \tag{16}$$

where L is the inductance in the circuit, R is the resistance and K_e is the EMF constant. The input voltages are paired such that $V_1 = V_2 = V_l$ and $V_3 = V_4 = V_r$, where V_l and V_r are the voltages controlling the left and right motors of the rover, respectively. The electrical component of the motor interacts with the mechanical component through the current and the motor speed Ω_j, the dynamics of which are described by

$$\dot{\Omega}_j = \frac{1}{J_m}\left(K_t i_j - b\Omega_j - \xi\Omega_j\right) \tag{17}$$

where J_m is the motor moment of inertia, K_t is the torque constant, b is the viscous torque constant and ζ is the base friction coefficient, representing the friction between the wheel and the ground.

Each motor generates a torque which is proportional to the current running through it and is given by

$$\tau_j = K_t i_j \eta_j \quad (18)$$

Here, η is the efficiency of the motor. It is identified through empirical testing to have the form $\eta_j = \alpha i_j + \gamma$, where α and γ are constants. The resulting forward force is then given by
(10).

D. Control and Guidance System

A simple feedback control system is used to direct the rover through discrete waypoints. For a given waypoint $r_d = [x_d, y_d, 0]^T$, the desired surge velocity u_d is obtained using a simple proportional controller on the distance error in the surge axis x^B $u_d = K_{p,xy}(x_d - x)\cos\psi + (y_d - y)\sin\psi$ (19)

where the surge velocity command is limited to the range $0 \le u_d \le 0.4$ ms^{-1}.

TABLE I
Rigid-Body Properties

Property	Symbol	Value	Unit
Effective area in x-axis	A_x	0.0316	m2
Effective area in y-axis	A_y	0.0448	m2
Drag coefficient	C_d	0.89	–
Moment of inertia about x-axis	I_x	0.0140	kgm^2
Moment of inertia about y-axis	I_y	0.0252	kgm^2
Moment of inertia about z-axis	I_z	0.0334	kgm^2
Mass	m	2.148	kg
Radius of wheel	R_w	0.0635	m
Moment arm of wheel	r_w	0.1245	m
Coefficient of friction in x	σ_x	0.22	–
Coefficient of friction in y	σ_y	1.00	–
Coefficient of friction in z	σ_z	0.30	–
Coefficient of friction about x	σ_p	0.35	–
Coefficient of friction about y	σ_q	0.44	–
Coefficient of friction about z	σ_r	0.18	–

Pseudo-controls are used to shape the response in each controllable degree of freedom before they are mixed to provide the motor voltage signals. The surge pseudo-input u_{surge} is determined by the proportional-integral (PI) law

$$u_{surge} = K_{p,u}(u_d - u) + K_{i,u}\int (u_d - u)\, dt \quad (20)$$

while the yaw pseudo-input is specified by a PI law with velocity feedback on the heading rate r $u_{yaw} = K_{p,\psi}(\psi_d - \psi) + K_{i,\psi}\int (\psi_d - \psi)dt - K_{d,\psi} r$ (21)

where ψ_d is the desired heading of the rover, obtained from the relative positions of the rover and waypoint by

$$\psi_d = \arctan \frac{y_d - y}{x_d - x} \quad (22)$$

To minimise aggressive turning, the difference between the heading command and current heading is limited to the range $-10^\circ \le (\psi_d - \psi) \le 10^\circ$.

The voltage commands to each pair of motors are obtained from the pseudo-controls by solving the relationship

$$V_l \; 1 \; 1 \; -1 \; u_{surge} = - \quad (23)$$

$$V_r \quad r_w \quad r_w \quad u_{yaw}$$

IV. Simulation Testing of Fault Detection Methods

It is clear that the rover model described in the previous section is not linear. This is in contrast to the generic linear model given in (2) and used to justify the InvSim approach to input residual generation. The ability to generate input residuals and use them in fault detection and isolation is thus investigated in simulation, using the *Genisa* InvSim algorithm. The rover vehicle is represented by the model in Section III and is susceptible to both faults and disturbances. This model is denoted the *system*. The software model used in the model-based and InvSim-based fault detection algorithms is identical to this but lacks any faults or disturbances. This is simply denoted the *model*. Both models use the properties described in Tables I–III.

Two cases are considered. First, an additive output fault is applied to the rover system. Output residuals are generated using model-based fault detection and used to isolate the location and severity of the fault. This provides the empirical proof for (3). The second case considers an input fault. Input residuals are generated using InvSim and their abilities to detect and isolate the fault compared with those of output residuals.

TABLE II
Motor Properties

Property	Symbol	Value	Unit
Viscous torque	b	0.008	Nm
Moment of inertia of motor	J_m	0.005	kgm^2
Torque constant	K_t	0.35	NmA^{-1}
EMF constant	K_e	0.35	Vrad^{-1} s^{-1}
Inductance of circuit	L	0.1	H
Resistance in circuit	R	4	Ω
Gradient for efficiency curve	α	−0.133	A^{-1}
Offset for efficiency curve	γ	0.6	–
Base friction on wheel	ξ	0.002	Nmsrad^{-1}

TABLE III Controller Gains

Property	Symbol	Value
Position controller proportional gain	$K_{p,xy}$	1
Velocity controller proportional gain	$K_{p,u}$	10
Velocity controller integral gain	$K_{i,u}$	0.1
Heading controller proportional gain	$K_{p,\psi}$	4
Heading controller integral gain	$K_{i,\psi}$	0
Heading controller velocity gain	$K_{d,\psi}$	1

The rover is simulated as operating on a flat, level terrain. This has the effect of reducing the rigid-body motion to three degrees of freedom: surge, sway and yaw. The outputs of the system are thus taken to be perfect measurements of the states in these directions, that is

$$y = [u,v,r]^T \quad (24)$$

The rover is directed along a path between discrete waypoints by the feedback control system described in Section III-D. When a fault or disturbance occurs, this has the effect of perturbing the rover's motion along the desired path. The path of the rover is shown in Fig. 8 for fault-free motion, a sensor fault, an actuator fault and a disturbance force. It is apparent in this instance that the faults and disturbance have a small effect on the vehicle motion which is largely compensated for by the controller. It is still of import, however, to detect and isolate the faults in the event that they are more severe and have greater impact on the closed-loop response.

A. Case 1: Fault in Heading Rate Output

An additive fault in the heading rate output $f_r = 15^\circ\,s^{-1}$ is simulated as occurring at $T_f = 20$ s. With reference to (2), the output fault vector is then $f_y = [0,0,f_r]^T$. Model-based fault detection is used to provide a model output y_m which is compared to the simulated system output y in Fig. 9. From (3), the heading rate output residual r_r will be of the form

$$r_r = \begin{cases} 0 & \text{if } t < T_f \\ f_r & \text{if } t \geq T_f \end{cases} \quad (25)$$

This result is validated in Fig. 10, which shows a step change in r_r from zero to $15^\circ\,s^{-1}$ after 20 s. The residuals for the surge and sway velocities are both zero for the duration of the simulation, as expected, and are not shown.

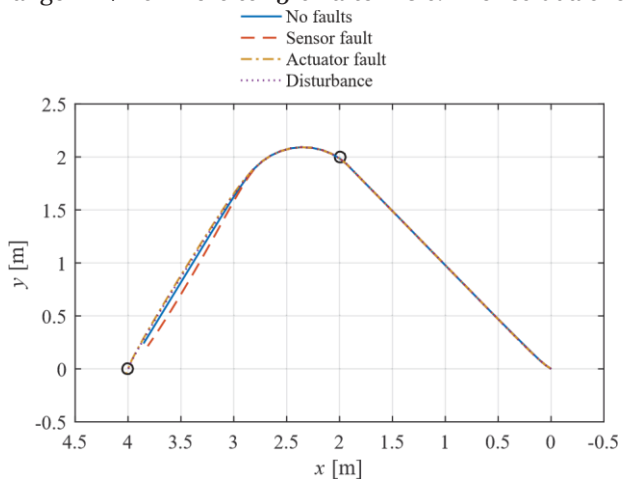


Fig. 8 Comparison of trajectories for rover with no faults, a sensor fault and an actuator fault

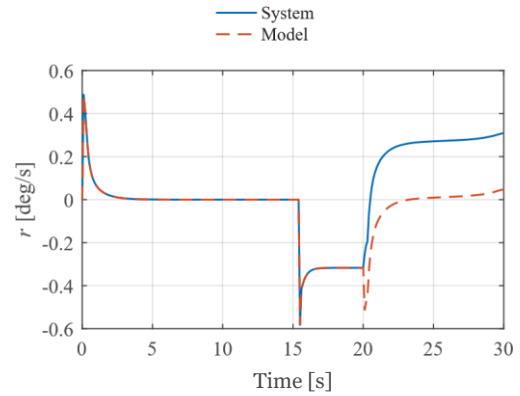


Fig. 9 Comparison of system heading rate measurement with equivalent output from mathematical model for a fault in the heading rate output

The absence of any non-zero residual in the surge and sway outputs, combined with the discontinuous shape of the heading rate residual, provides a strong case that a single fault isolated to the heading rate output has occurred. The heading rate output fault is thus trivial to detect using a static threshold. It may then be isolated and recovered from, by applying a correction of $-15^\circ\,s^{-1}$ to the heading rate output channel.

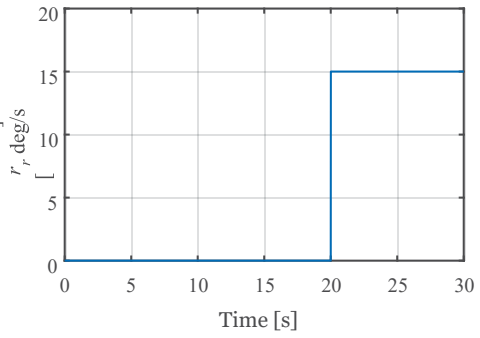


Fig. 10 Residual in heading rate output with fault in same channel

B. Case 2: Fault in Voltage Signal Received by Left-Hand Motors

An additive fault in the left-hand actuator input $f_{Vl} = 1$ V is simulated as occurring at $T_f = 20$ s. The input fault vector is thus $f_u = [f_{Vl},0]^T$. Using model-based fault detection, (3) shows that the output residuals will be of the form

$$r_y = \begin{cases} [0,0,0]^T & \text{if } t < T_f \\ F(f_u,t) & \text{if } t \geq T_f \end{cases} \quad (26)$$

where F describes the propagation of the input fault through the system.

The output residuals resulting from simulation testing of the input fault are shown in Fig. 11. For comparison, these

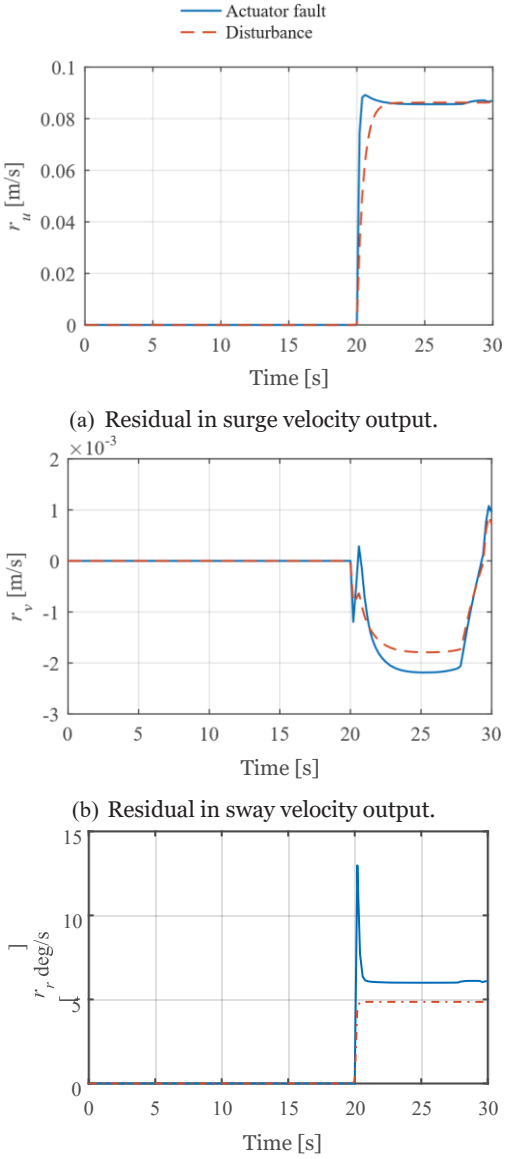
residuals are plotted against those resulting from a separate simulation with a disturbance force $F_d = [0.4,0,0]^T$ N, applied at position $r_d = [0,-0.1,0]^T$ m and time $T_d = 20$ s. It is apparent that the single input fault manifests itself in all three output residuals, making it difficult to isolate. The disturbance similarly produces a response in all three residuals. While it is clear from viewing the residuals that the effects of the fault and disturbance differ, a static threshold would be unable to discriminate between them. Additionally, there is no indication as to the magnitude of the fault, unlike the case with the output fault.

In an effort to yield further information on the input fault and disturbance, InvSim is used to generate input residuals. From (4), it is anticipated that the input residuals will be of the form

$$r_u = \begin{cases} [0,0]^T & \text{if } t < T_f \\ f_u & \text{if } t \geq T_f \end{cases} \quad (27)$$

Fig. 12 shows the results of the input residual generation using InvSim. It is clear that neither input residual satisfies the conditions describes in (27). Both residuals show perturbations at approximately 0.5 and 15.5 seconds. These perturbations correspond to manoeuvres by the rover at the beginning of the simulation and as it passes through a waypoint. They may therefore be attributed to sudden changes in the rover inputs and outputs. Smoothness in the trajectory of the system is an important consideration in InvSim [9]. A discontinuous path through the waypoints could thus be responsible for these perturbations.

These perturbations may therefore be neglected in favour of the clear discontinuity occurring at the 20 s mark in r_{Vl}. Again, this is not a perfect step change as suggested by (27). However, accounting for the presence of further perturbations, the residual can be stated to quite clearly show both the shape and magnitude of the fault f_{Vl}. Conversely, the residual r_{Vr} continues to demonstrate only small perturbations after the occurrence of the fault. In this case, a small static threshold on each input residual would indicate the presence of a fault in the left-hand actuators only, while the shape of the residual argues that the fault has occurred at the input itself. Compare these results to the effects of a disturbance force, shown also in Fig. 12. The left-hand residual r_{Vl} shows a longer transient phase for the disturbance than the fault, indicating that it does not occur at the input, but later in the system. The right-hand residual r_{Vr} also shows a small persistent change after the disturbance has occurred, indicating that the disturbance is also coupled with the right-hand actuators. The presence of a persistent non-zero residual in r_{Vr} may also be used to discriminate the disturbance from the input fault.



(a) Residual in surge velocity output.

(b) Residual in sway velocity output.

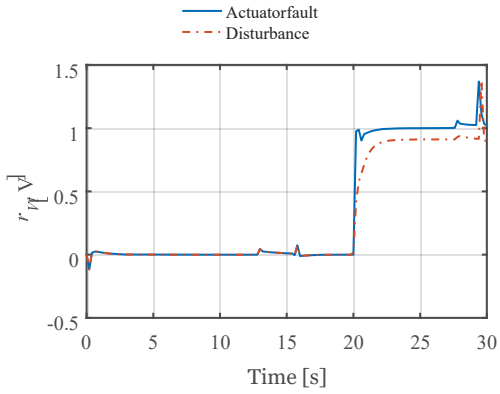
(c) Residual in heading rate output.

Fig. 11 Output residuals generated using conventional model-based fault detection on system with input fault

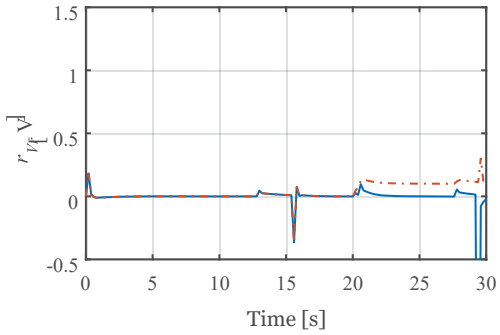
C. Comparison of Methodologies

These results may be summarised by constructing a table of residuals, shown in Table IV. Here 0 denotes no residual, + denotes a small positive residual and ++ denotes a large positive residual. For the output fault f_r, it is immediately obvious that the fault affects the heading rate output in isolation and may therefore be stated to occur in that output. For the input fault, large residuals are generated in both surge velocity and heading rate outputs, providing no

clear indication for the location of the fault. Conversely, the input residuals



(a) Residual in left-hand motor voltage signal



(b) Residual in right-hand motor voltage signal.

Fig. 12 Residuals from InvSim-based fault detection on system with actuator

fault

TABLE IV
Residuals for Both Sensor and Actuator Faults

	f_r	f_{Vl}	F_d
r_u	0	++	++
r_v	0	+	+
r_r	++	++	++
r_{Vl}		++	++
r_{Vr}		0	+

clearly isolate the fault to the left-hand actuators. Finally, while the output residuals have identical entries in the table for the input fault and disturbance, the input residuals do not. This allows use of the input residuals to discriminate between an input fault and environmental disturbance, demonstrating the benefit of inverse simulation in this application.

V. Conclusion

It is clear from the previous section that inverse simulation may be used to generate input residuals for a system, in a manner similar to the generation of output residuals using a system model. These input residuals have been shown to provide clear and unambiguous information on the location, severity and time of a fault at the input to the rover system. In contrast, the output residuals are unable to provide much information beyond the fact that a fault has occurred somewhere in the system prior to the output. While advanced techniques such as structured residuals or adaptive thresholds may allow isolation of an input fault using the system outputs, InvSim provides a conceptually simpler approach: an additive fault at the input manifests in the residual as a step change with magnitude equal to that of the fault. It is then far simpler to draw a conclusion on the nature of this fault from the input residuals shown in Fig. 12 than the output residuals given in Fig. 11. Additionally, the input residuals generated using InvSim may be combined with any output residuals to provide more comprehensive information on any faults or disturbances in the system. Table IV highlights this benefit to a degree.

With reference to the use of inverse simulation specifically, (4) demonstrated the analytical proof of detecting faults at the input using input residuals. In simple systems, obtaining the inverted model $G^{-}{}_{m}{}^{1}(s)$ may be trivial. In the case of a complex, non-linear system such as the presented rover model, the system may only be inverted numerically. The *Genisa* InvSim algorithm provides a stable, generic method by which to achieve this inversion.

References

[1] S. Kassel, “Lunokhod-1 Soviet lunar surface vehicle,” DARPA, Tech. Rep. September, 1971.

[2] M. B. Quadrelli, L. J. Wood, J. E. Riedel, M. C. McHenry, M. Aung, L. A. Cangahuala, R. A. Volpe, P. M. Beauchamp, and J. A. Cutts, “Guidance, navigation and control technology assessment for future planetary science missions,” *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 7, pp. 1165–1186, 2015.

[3] R. Isermann, *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*. Berlin: Springer-Verlag, 2006.

[4] R. Dearden, T. Willeke, R. Simmons, V. Verma, F. Hutter, and S. Thrun, “Real-time fault detection and situational awareness for rovers: Report on the Mars technology program task,” in *Proceedings of the IEEE Aerospace Conference*, vol. 2, Big Sky, MT, 2004, pp. 826–840.

[5] V. Verma, G. Gordon, R. Simmons, and S. Thrun, “Tractable particle filters for rover fault diagnosis,” *IEEE Robotics & Automation Magazine*, vol. 11, pp. 56–66, 2004.

[6] K. Ferguson, “Towards a better understanding of the flight mechanics of compound helicopter configurations,” PhD thesis, University of Glasgow, November 2015.

[7] 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, pp. 920–926, 1991.

[8] 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.

[9] D. 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.

[10] D. J. Murray-Smith, “Inverse simulation and analysis of underwater vehicle dynamics using feedback principles,” *Mathematical and Computer Modelling of Dynamical Systems*, vol. 20, no. 1, pp. 45–65, 2014.

[11] 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.

[12] K. Worrall, D. Thomson, and E. McGookin, "Application of inverse simulation to a wheeled mobile robot," in *Proceedings of the 6th International Conference on Automation, Robotics and Applications (ICARA 2015)*, Queenstown, February 2015.

[13] 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.

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

[15] K. J. Worrall, "Guidance and search algorithms for mobile robots: Application and analysis within the context of urban search and rescue," PhD thesis, University of Glasgow, 2008.

[16] K. J. Worrall and E. W. McGookin, "A mathematical model of a Lego differential drive robot," in *Proceedings of the 6th UKACC Control Conference*, Glasgow, 2006.