AUTONOMOUS PLANETARY ROVER CONTROL USING INVERSE SIMULATION

Kevin Worrall(1), Douglas Thomson(1), Euan McGookin(1), Thaleia Flessa(1)

(1)University of Glasgow, Glasgow, G12 8QQ, UK, Email: kevin.worrall@glasgow.ac.uk

ABSTRACT

With extended autonomous capability planetary rovers will be able to carry out more exploration. This paper presents a method of control that provides such autonomy. This controller is based on Inverse Simulation, which considers the path to be taken and the dynamics of the rover. The principle of Inverse Simulation is that the time history for the desired trajectory is used as an input to the Inverse Simulation which outputs the required control values over the course of the manoeuvre to be carried out. The resulting control signal can then be applied to the rover to make it manoeuvre in the desired manner. This method of control provides very accurate position and orientation control of the rover while considering the limitations of the rover and the actuators used.

1. INTRODUCTION

Inverse Simulation is a method by which the control inputs necessary for a given system to respond in a defined manner can be calculated. In this way, given an appropriate mathematical model, the control inputs necessary for a given system to respond in a defined manner can be directly calculated. It is an iterative process where step changes in the various controls are applied until the predicted response matches the predefined response [1]. In the context of this work, the desired path for the rover is determined. This path will provide the inputs to define the desired trajectory for the Inverse Simulation, which in turn will generate the required guidance commands to follow the trajectory.

Fig. 1 provides a diagrammatic view of the applications of Inverse Simulation and shows the multiple uses of Inverse Simulation for a single application.

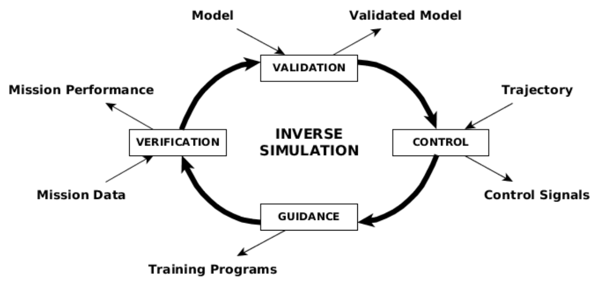

Figure 1. Overview of Inverse Simulation

The use of Inverse Simulation has been predominantly within aerospace with much of the work carried out on helicopters [1]. The application of Inverse Simulation to helicopters has been used to produce control signals for specific flight manoeuvres [1]. Recently, Inverse Simulation has been applied to manoeuvre control for Autonomous Underwater Vehicles [2] and mobile robots [3]. The latter has shown that the trajectory taken by the robot can be accurately controlled and matches both the desired path and performance requirements.

Inverse Simulation has yielded benefits in other areas. It has been applied in pilot training where the pilot is provided with the calculated control inputs for the current manoeuvre and is expected to match them [1]. This process enables the pilot to enhance their skills. Another application for Inverse Simulation is in the validation of mathematical models [4]. With the reliance on computer based simulation the validation of the models used is important. Inverse Simulation can be used to validate models by applying the output data from the real system to the Inverse Simulation and comparing the control outputs to those that were applied. If the values match the model used within the Inverse Simulation are valid, if the values do not match then Inverse Simulation can be used to further investigate the differences.

The use of Inverse Simulation in planetary rovers will increase the autonomous control over the rover while ensuring that mission objectives are met. In addition to this, since the actuator and rover dynamics are considered, the life of the rover can be extended as control parameters are calculated with the limitations of both considered. As the mission progresses and degradation in systems occurs the Inverse Simulation can be updated, by updating the mathematical model used, to compensate for the degradation. This will extend the operation lifespan of the rover and thus its mission, which results in better value for money.

This work presents the Integration Method of Inverse Simulation, where the mathematical model is used in a conventional simulation structure within the Inverse Simulation algorithm [1-6]. The Integration Method uses numerical integration to calculate the require control parameters [1-6]. This method is presented in Section 2 and discussed. A description of the mathematical model of the rover is provided in Section 3. Section 4 describes how the Inverse Simulation is applied to the chosen rover with each stage of the algorithm discussed. One major step when applying Inverse Simulation is the calculation of the desired trajectory. In this work an algorithm is provided that calculates the desired trajectory based on a series of way-points. This desired trajectory, based on the desired forward and yaw velocities at each time step, becomes the input to the Inverse Simulation. A series of tests were carried out to test the operation of the algorithm. An overview of the tests and the results produced are provided in Section 5 with Section 6 providing results from a practical implementation. The results show that Inverse Simulation is a suitable method of control and will provide a means of accurately controlling the desired position and orientation of a mobile robot while considering both rover and actuator limitations.

2. OVERVIEW OF INVERSE SIMULATION

This paper presents the Integration Method of Inverse Simulation. As mentioned above, this method of Inverse Simulation uses numerical integration. Other methods of Inverse Simulation are out-with the scope of this paper and the interested reader is guided to [1]. This implementation of the Integration Method uses the Newton-Raphson method to solve the forward simulation.

Inverse Simulation is an algorithm that will generate a time history of control signals based on a time history of a desired trajectory. The following is the procedure used:

1. The Present Desired Trajectory and the previous control signal are presented to a Newton-Raphson loop. If this is the first pass then an estimated control signal is selected.

2. A series of perturbed signals are generated. The perturbed signals are based on the control signals supplied. Each control signal is slightly adjusted to provide a series of new control signals.

3. Each of the perturbed control signals are applied to the model and the output compared to the desired.

4. A Jacobian is created using the result calculated from each run. The inverse of this Jacobian is the calculated.

5. Using Equ. 1, the next control signals to be tested are calculated. (1)

   $$u_n = u - \left(J^{-1} \times e\right) \quad (1)$$

   here u_n a vector of the calculated control signals, u the vector of the current control signals, J^{-1} the inverse Jacobian and e represents the error between the current control signal model output and the desired model output.

6. u_n is then applied to the model with the output compared to the desired. If the difference is within a set tolerance then u_n is saved and the algorithm loops back to 1 until the complete trajectory has been processed. If the difference is not within a tolerance then u_n is used as the control signal to be perturbed. Steps 2 to 6 are repeated until the difference is within tolerance or the maximum number of iterations around the loop has been exceeded.

There are various parameters that are required to be set; time-step for the time histories, the tolerance in the difference between the actual and desired and the maximum number of iterations. For this work these values are set to be 0.001s, 0.000005 and 30 respectively.

Further information on the theoretical side of Inverse Simulation can be found in [1, 5].

3. MODEL OF A 4WD ROBOT

Inverse Simulation relies upon an accurate mathematical model. An overview of the model used in this work is provided which has been validated in previous work [6]. Further Details of this model can be found in [3, 7-9].

A four wheel robot was selected to be modelled. This robot has two wheels on each side and relies upon differential drive for steering. An image representing the robot, with appropriate frames of reference, is shown in Fig. 2, reproduced from [3].

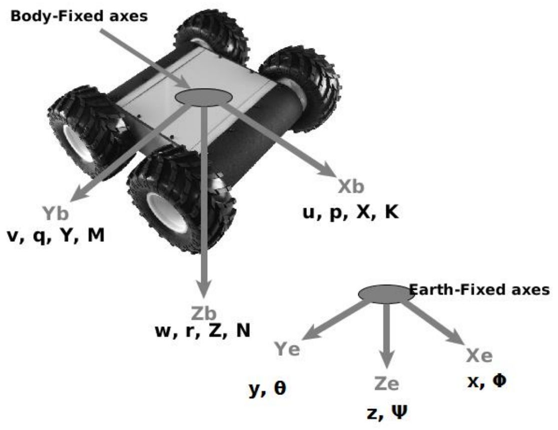

Figure 2. Modelled robot with frames of reference. Reproduced from [3]

The complete state space model can be represented as [3, 7]:

$$\begin{bmatrix} \dot{\boldsymbol{v}} \\ \dot{\boldsymbol{\eta}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M}^{-1}\{-(\boldsymbol{C}(\boldsymbol{v}) + \mathbf{D}(\boldsymbol{v}))\boldsymbol{v} - \mathrm{g}(\boldsymbol{\eta}) + \boldsymbol{\tau}\} \\ \mathbf{J}(\boldsymbol{\eta})\boldsymbol{v} \end{bmatrix} \quad (2)$$

which describes the rigid body dynamics and the kinematics of the robot. Here $\boldsymbol{v}$ is the velocity vector, $\boldsymbol{M}$ is the mass and inertia matrix, $\boldsymbol{C}(\boldsymbol{v})$ is the Coriolis matrix, which together represent the rigid model dynamics. $\mathbf{D}(\boldsymbol{v})$ is the damping matrix, $g(\boldsymbol{\eta})$ represents the gravitational forces and moments, with $\boldsymbol{\eta}$ representing the orientation in the inertial frame and $\boldsymbol{\tau}$ is a vector representing the force and torque inputs [3, 7], with $J(\boldsymbol{\eta})$ the transformation matrix relating the body-fixed frame to the Earth-fixed frame. Vector $\boldsymbol{\tau}$ is (3):

$$\boldsymbol{\tau} = [X\ Y\ Z\ K\ M\ N]^T \tag{3}$$

with X being the surge force, Y the sway force, Z the heave force, K the roll moment, M the pitch moment and N the yaw moment. In the configuration used only surge and yaw are controllable.

$\boldsymbol{v}$ can be represented as (4):

$$\boldsymbol{v} = [u\ v\ w\ p\ q\ r]^T \tag{4}$$

Where u, v, w are the surge, sway and heave velocities respectively and p, q, r are the roll, pitch and yaw rates. Vector η is defined as (5):

$$\boldsymbol{\eta} = [x\ y\ z\ \phi\ \theta\ \psi]^T \tag{5}$$

Here x, y, z are the spatial coordinates and ϕ, θ, ψ are the roll, pitch, yaw angles respectively. For any control algorithm the input to the model needs to be well defined. This is also the case for Inverse Simulation. The algorithm alters the input to the model for each iteration. This means that an understanding of the control inputs is required. For this particular model the inputs used are the wheel torques. Within the model the torque created by each wheel is used to calculate the Surge force (force acting along the Xb axis) and the Yaw torque (torque acting about the Zb axis). To calculate the surge force, the force generated by each wheel is calculated from the torque and these are then summed together. To calculate the yaw torque the following is used [3, 7]:

$$N = ((F_{fl} + F_{bl}) - (F_{fr} + F_{br})) \times r_m \tag{6}$$

with F_{fl} being the force generated by the front left wheel, F_{bl} the force generated by the back left wheel, F_{fr} being the force generated by the front right wheel, F_{br} the force generated by the back right wheel and r_m the moment arm which is the distance from the line of the centre of gravity lies on, [3, 7].

4. IMPLEMENTATION OF INVERSE SIMULATION

With the Inverse Simulation algorithm provided and the mathematical model derived the next step is to provide a method of generating appropriate time histories describing the accelerations, velocities and displacements of the robot.
To generate the required time histories the first step is to define the path that is to be followed. The algorithm developed accepts a series of way-points, defined by XY coordinates, as inputs. It is assumed that the robot will stop at each way-point, turn on the spot, and then travel forward again. Using these the distance to travel between each is calculated as is the required angle the robot is to be at. In previous work [1] the method used has been to generate time histories based on a 7th order polynomial. For short manoeuvres over a short time period this method can be used. However, this method becomes impractical for longer time periods, where there may be an expectation for the rover to travel at a constant speed. The current method assumes that there is a constant acceleration up to the midpoint of the manoeuvre then a constant deceleration from the mid-point to the end of the manoeuvres. The method developed here accepts as input the required constant speed and the required time for acceleration and deceleration. Using this information and the distances and angles calculated from the way points the accelerations along the x-axis and the accelerations about the z-axis are calculated, assuming a time step of 0.001s. During the acceleration and deceleration phase the 7th order polynomial described in [1] is used. At all other times a constant velocity is assumed resulting in a zero value for acceleration. The time histories generated are then integrated to provide the velocities which are then further integrated to provide the displacements. For example, the robot is commanded to go forward 1m with a constant speed of 0.1ms^{-1}. The acceleration time histories generated can be seen in Fig. 3, with Fig. 4 providing the velocities plots and Fig 5 the displacements.

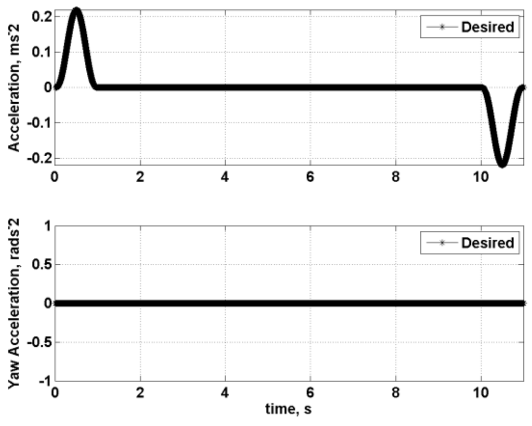

Figure 3. Accelerations

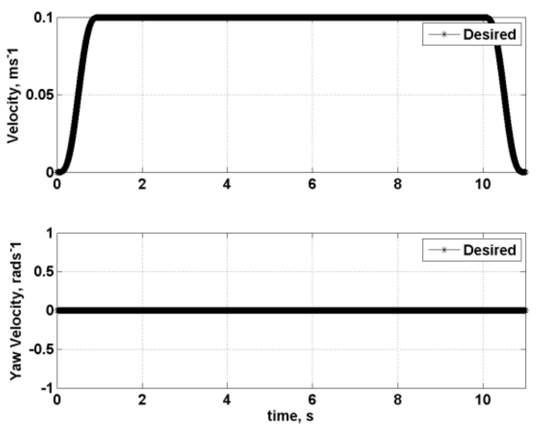

Figure 4. Velocities

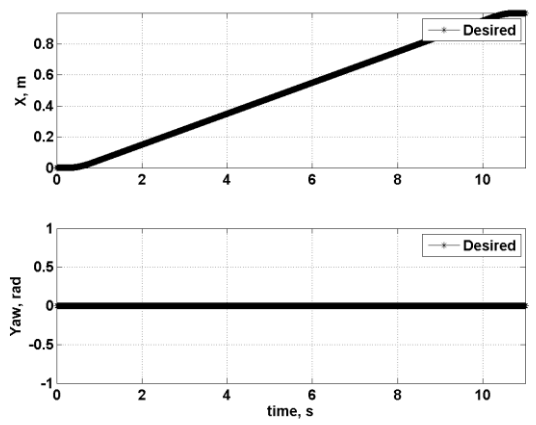

Figure 5. Displacements

It can be seen that the final displacement is 1m and that, after the acceleration phase, the forward velocity was maintained at $0.1ms^{-1}$.

The definition of the trajectory allows the set-up of the Inverse Simulation to commence. To do this the actual parameters being controlled need to be selected. In this work the surge velocity (along the Xb axis) and the Yaw velocity (about the Zb axis) are to be controlled. To achieve this the inputs to the Inverse Simulation are the desired surge and yaw velocities. By controlling these two parameters any motion can be achieved.

For the Inverse Simulation to run perturbations are carried out on the control signal. As described in [3] the perturbations for this robot need to be calculated as pairs. The reason for this is that the model accepts four inputs, one for each motor. There are two motors on each side and for proper operation the motors on the same side should be sent the same control signal. This reduces the number of inputs to two. To achieve this the perturbations shown in Table 1 (reproduced from [3]) are used to achieve the controllable movements. The value that is added and subtracted is based on the current control signal and the time-step of the simulation. This value should be representative of the

Table 1. Perturbations used (Reproduced from [3])

Motion	**Left Side Perturbation Sign**	**Right Side Perturbation Sign**
Forward	+	+
Backwards	-	-
Left Turn	-	+
Right Turn	+	-

change in the control input that can be achieved within the time-step of the system.

With the perturbations defined the Jacobian needs to be defined using the following equation [3]:

$$\mathbf{J} = \begin{bmatrix} \frac{f(x,u)_2 - f(x,u)_3}{2u_L\delta t} & \frac{f(x,u)_4 - f(x,u)_5}{2u_R\delta t} \\ \frac{f(x,r)_2 - f(x,r)_3}{2u_L\delta t} & \frac{f(x,r)_4 - f(x,r)_5}{2u_R\delta t} \end{bmatrix} \tag{7}$$

Here $f(x,u)_i$ represents the forward velocity result for perturbation i, $f(x,r)_i$ represents the rotational velocity for perturbation i, u_L and u_R are the previous control signals and δt is the time-step. It should be noted that Perturbation 1 is the unaltered control signal.

5. SIMULATION RESULTS

A series of experiments were designed with each one based on a sequence of way-points. Each experiment involves a manoeuvre or path to follow. Each manoeuvre is split into a number of forward and rotational stages. The manoeuvres selected test multiple aspects of the Inverses Simulations ability to generate control signals involving long distances with constant speed, positive to negative coordinates and crossing the +180° to -180° point. The timescale for each experiment is dependent on the selected constant speed and the distances required to be traversed.

Using the Inverse Simulation developed the required control signals to be supplied to the rover were calculated. The control signals were then applied to a forward simulation of the rover to show that the resulting motion is as desired. Each test had a forward velocity of $0.1ms^{-1}$ and rotational velocity of $0.1°s^{-1}$ and the start position for each test was (0, 0). Tables 2 and 3 provide the mean error between the desired and actual linear and rotational velocities. The standard deviation for each result is also provided.

The results show that there is a minimal deviation between the desired and actual. This shows that Inverse Simulation is able to generate accurate control signals for the rover. Further to this the deviation from the desired is minimal showing that very accurate control can be achieved using Inverse Simulation. This will result in highly accurate trajectory following performance. The path and control signals for the clover, zigzag and Long distance experiments are provided (Fig. 6-12) to show the paths set and typical control signals generated. Fig 6 shows the path travelled for the clover manoeuvre with Fig 7 the control signals representing the control signals. During the forward motion the signals match. This shows that all motors are running in the same direction. When the robot turns the control signals generated are opposite to one another. This is consistent with a differentially steered rover. The output from the zigzag and long distance tests provide similar results.

Table 2. Simulation Results – Linear Velocities

Experiment	Linear Velocity Mean Error
Forward 1m	$0.0000\text{x}10^{-4}$ $\sigma = 0.5410\text{x}10^{-4}$
Left Turn	$0.0001\text{x}10^{-4}$ $\sigma = 0.4487\text{x}10^{-4}$
Right Turn	$0.0002\text{x}10^{-4}$ $\sigma = 0.4493\text{x}10^{-4}$
Left-Right Turn	$0.0001\text{x}10^{-4}$ $\sigma = 0.4268\text{x}10^{-4}$
Square	$-0.0001\text{x}10^{-4}$ $\sigma = 0.4139\text{x}10^{-4}$
Figure of 8	$0.0002\text{x}10^{-4}$ $\sigma = 0.3846\text{x}10^{-4}$
Clover	$0.0002\text{x}10^{-4}$ $\sigma = 0.2953\text{x}10^{-4}$
ZigZag	$0.0000\text{x}10^{-4}$ $\sigma = 0.3411\text{x}10^{-4}$
Long Distance	$0.014\text{x}10^{-4}$ $\sigma = 0.2843\text{x}10^{-4}$

Table 3. Simulation Results – Rotational Velocities

Experiment	Error Rotational Velocities
Forward 1m	$0.0039\text{x}10^{-5}$ $\sigma = 0.6580\text{x}10^{-5}$
Left Turn	$-0.0045\text{x}10^{-4}$ $\sigma = 0.5644\text{x}10^{-4}$
Right Turn	$-0.0052\text{x}10^{-4}$ $\sigma = 0.5674\text{x}10^{-4}$
Left-Right Turn	$-0.0031\text{x}10^{-4}$ $\sigma = 0.6187\text{x}10^{-4}$
Square	$-0.0023\text{x}10^{-4}$ $\sigma = 0.6033\text{x}10^{-4}$
Figure of 8	$-0.0026\text{x}10^{-4}$ $\sigma = 0.6437\text{x}10^{-4}$
Clover	$-0.0015\text{x}10^{-4}$ $\sigma = 0.5324\text{x}10^{-4}$
ZigZag	$-0.0025\text{x}10^{-4}$ $\sigma = 0.5434\text{x}10^{-4}$
Long Distance	$0.0003\text{x}10^{-6}$ $\sigma = 0.5157\text{x}10^{-4}$

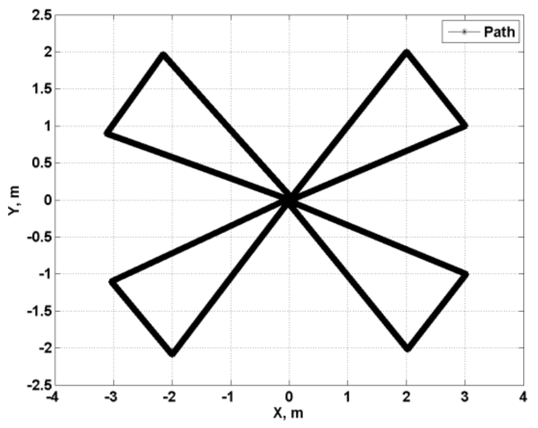

Figure 6. Path for Clover

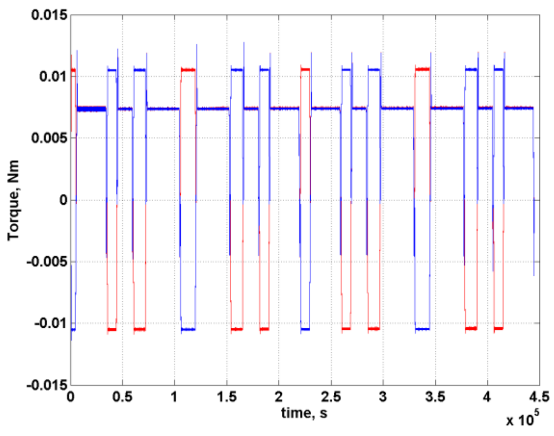

Figure 7. Control signals for Clover with the red line representing the right side control signal and the blue line representing the left side control signals.

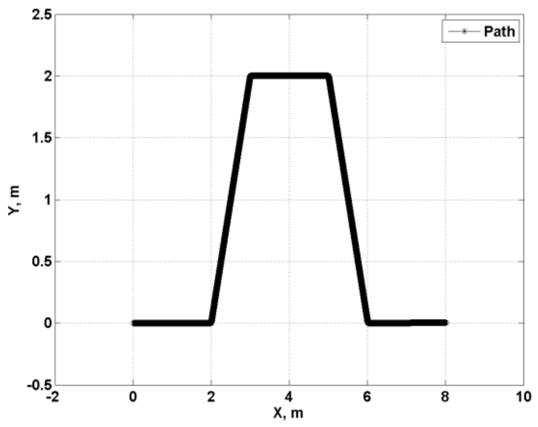

Figure 8. Path for ZigZag

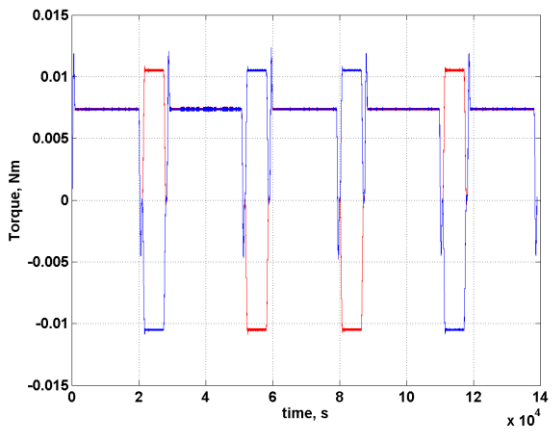

Figure 9. Control signals for ZigZag with the red line representing the right side control signal and the blue line representing the left side control signals.

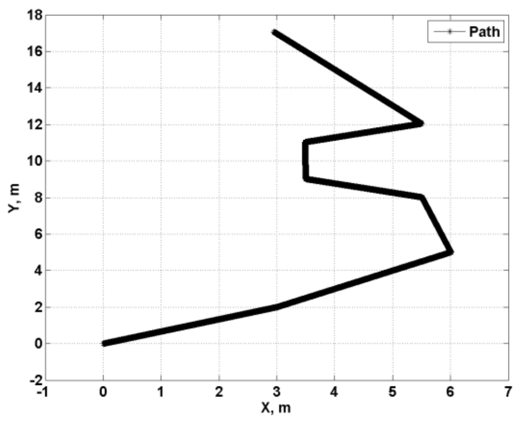

Figure 10. Path for Long Distance

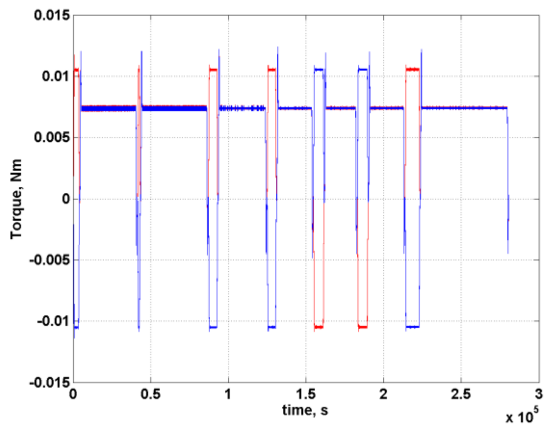

Figure 11. Control signals for Long Distance with the red line representing the right side control signal and the blue line representing the left side control signals.

6. PRACTICAL RESULTS

The simulated robot is based on an existing usable rover. The model created has been validated and the results from the simulation show that the control inputs calculated are usable; this would mean that the control inputs can be applied to the robot and the same results will be shown. To demonstrate this a number of minor changes are required due to the simulation required wheel torques as inputs but the rover requires wheel speeds in RPM. In order to provide a suitable conversion, experimental data was used to derive an empirical relationship between the wheel speeds and the wheel torques. This yielded the following equation:

$$\tau = -2\times10^{-7}\times RPM^{2} + 0.00045\times RPM \tag{8}$$

The Inverse Simulation was then run with a time-step of 0.001s. However when applied to the rover the time-step becomes 0.05s. This time-step is more representative of the control signals the rover controller can generate. The selected test manoeuvre was forward 1m with a forward velocity of 0.1ms^{-1}. To allow additional data to be captured the rover rests for 1s moves forward 1m then rests for 1s. The Inverse Simulation was run and the output checked in simulation. The control signals generated are shown in Fig. 12.

With the successful completion within the simulation a file containing the control inputs was created and loaded onto the rover. The average result over tens runs can be found in Table 4. An example from the encoder data is provided in Fig. 13. This figure shows how the rover moved.

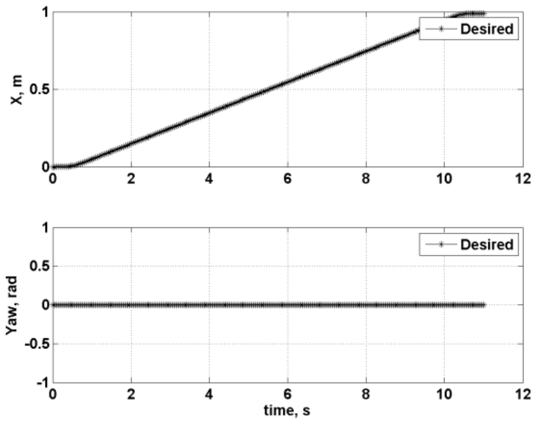

Figure 12. Control Signals generated by the Inverse Simulation

Table 4. Experimental Results

	Measured (m)	Encoders (m)
Average	1.00290	1.03560
Error	0.00290	0.03560
Standard Deviation	0.00226	0.00049
Min'	0.99900	1.03500
Max'	1.00700	1.03600

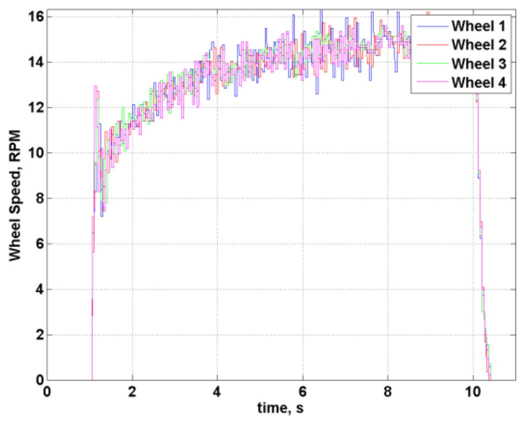

Figure 13. Encoder Data for a single run.

It can be seen from the results presented that the rover has traversed the required distance with minimal error. Each data file from the rover has 2200 elements with a logging interval of 0.005ms resulting in a time of 9s (11s -2 x 2s pauses) to carry out the manoeuvre. This equates to 0.106ms^{-1} which is the requested constant speed. This data shows that Inverse Simulation can be used for the control of a rover and the results achieved are highly accurate.

7. FUTURE APPLICATIONS

There have been many space related applications where Inverse Simulation has been identified as a potentially useful tool. The application to a basic rover has been shown. In addition to rover control, the rendezvous and docking of space assets is another area that could benefit from Inverse Simulation. During the rendezvous and docking process there are known trajectories that the space assets are to take. With this information, an Inverse Simulation can be created to provide the required control inputs for the space assets. The requirements for planetary landers to follow set trajectories to land on objects of interest could also benefit from the use Inverse Simulation. The control inputs for the desired landing trajectory could be calculated and applied to the system. A further application of Inverse Simulation that covers multiple areas is the use of Inverse Simulation as a method of failure detection, isolation and recovery. A reference set of control signals is generated and the system outputs can be compared. This will allow a method of verification for procedures carried out.

8. CONCLUSION

This paper has provided an overview of Inverse Simulation and provided an example of autonomous rover control using Inverse Simulation. The basic operating principles of Inverse Simulation on a rover has been presented. The results provided show that the use of Inverse Simulation as a method of generating control signals provides highly accurate path following. The extension of this work to a practical demonstration show that the work is applicable.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the UK Space Agency for funding this work via a CREST2 award.

REFERENCES

1. Thomson, D. and Bradley, R., (2006), Inverse simulation as a tool for flight dynamics research-Principles and applications, *Progress in Aerospace Sciences* 42, pp. 174-210.
2. Murray-Smith. D.J., (2012), Inverse simulation of an underwater vehicle model using feedback principles. *MATHMOD 2012: 7th Vienna International Conference on Mathematical Modelling*, Austria
3. Worrall, K., Thomson, D., and McGookin, E. (2015), Application of Inverse Simulation to a Wheeled Mobile Robot. In: *6th International Conference on Automation, Robotics and Applications (ICARA 2015)*, Queenstown, New Zealand
4. Bradley, .R, Murray-Smith, D.J, and Thomson, D., (1990), Validation of Helicopter mathematical models. *Transactions of the Institute of Measurement and Control*, vol. 12, 4: pp. 186-196.
5. Murray-Smith, D.J. (2000), The inverse simulation approach: a focused review of methods and applications, *Mathematics and Computers in Simulation 53*, pp. 239-247.
6. Hess, R.A., and Gao, C., (1993), A generalized algorithm for inverse simulation applied to helicopter manoeuvring flight. *Journal of the American Helicopter Society*, Volume 38, Number 4, pp. 3-15(13)
7. Worrall, K., (2008), *Guidance and Search Algorithms for Mobile Robots: Application and Analysis within the Context of Urban Search and Rescue*, PhD Thesis, University of Glasgow
8. Fossen, T.I., (2002), *Marine Control Systems: Guidance, Navigation, and Control of Ships, Rigs and Underwater Vehicles*, Marine Cybernetics,
9. Worrall, K. and McGookin, E., (2006), A Mathematical Model of a Lego Differential Drive Robot, *Proc' of 6th UKACC*, Sept', Glasgow