72nd International Astronautical Congress (IAC), Dubai, United Arab Emirates, 25-29 October 2021.

IAC-21,C1,8,12,x64025

Attitude Control of a Nanosatellite using Inverse Simulation

Robert Gordon [a], **Kevin Worrall** [b], **Matteo Ceriotti** [c]

[a] James Watt School of Engineering, University of Glasgow, r.gordon.5@research.gla.ac.uk
[b] James Watt School of Engineering, University of Glasgow, kevin.worrall@glasgow.ac.uk
[c] James Watt School of Engineering, University of Glasgow, matteo.ceriotti@glasgow.ac.uk

Abstract

Modern nanosatellites are being utilised more widely for missions that require high pointing accuracy of their instrumentation. Therefore, attitude control methods that can ensure pointing requirements are always met are needed. This paper explores the feasibility, performance and practicality of applying existing Inverse Simulation (InvSim) techniques to the attitude control problem. A series of different attitude manoeuvres are used to show the versatility of InvSim. Specific scenarios of an inertial pointing and spin-axis pointing rigid body 3U nanosatellite, with four reaction wheel actuators, are investigated. Performance of the InvSim controller is compared to Proportional-Integral-Derivative (PID) control through consideration of the pointing error, peak actuator power and total actuator energy of the attitude manoeuvres. InvSim is successfully implemented as an attitude controller, with comparable performance to traditional PID control in most cases, and performing better in the presence of higher nonlinear dynamics. In addition, two specific use cases are explored to show the superior practicality and versatility of InvSim, and its possible use as a design tool/aid.

Nomenclature

$\boldsymbol{u}$	Control vector
T	Control timestep
$\boldsymbol{y}$	Controlled states vector
$\boldsymbol{y}_d$	Desired controlled states vector
$\boldsymbol{\omega}_b$	Body rate vector
$\mathbf{J}_b$	Body inertia matrix
$\boldsymbol{\tau}_a$	Equivalent actuator torque
$\boldsymbol{h}_w$	Reaction wheel momentum vector
$\dot{\boldsymbol{h}}_w$	Reaction wheel torque vector
$\mathbf{A}_w$	Reaction wheel actuator distribution matrix
$\hat{\boldsymbol{w}}_m$	Reaction wheel unit vector
T_w	Reaction wheel time constant
u_{sat}	Control torque saturation
$h_{w,\text{sat}}$	Reaction wheel momentum saturation
ϕ, θ, ψ	Euler angles
$\mathbf{A}_{zyx}$	Attitude matrix
$\mathcal{F}^b, \mathcal{F}^i$	Body and inertial reference frames
$\boldsymbol{\epsilon}$	Controlled states error
$\mathbf{J}_n$	Jacobian
m	Number of controls
p	Number of controlled states
$\Delta\boldsymbol{u}$	Control perturbation vector
Δu_{min}	Minimum control perturbation
$\boldsymbol{\epsilon}_{\text{tol}}$	Tolerance of Newton-Raphson loop
dt	Integration timestep
$\mathbf{P}_{\text{wps}}$	Matrix of desired trajectory way-points
t_{move}	Time-to-move
t_{hold}	Time-to-hold
ω_{spin}	Spin-axis spin rate
$C(s)$	PID control law
$G(s)$	Plant
K_p, K_i, K_d	PID controller gains
$\boldsymbol{q}$	Quaternion
ϑ_ϵ	Pointing error
P	Actuator power
E_{total}	Total actuator energy

1. Introduction

The advent of nano and pico satellites has allowed for greater and more affordable access to space [1]. These small satellite platforms are being more widely used for complex missions using a variety of instrumentation, traditionally reserved for larger satellites [2, 3]. In many cases, these instruments have restrictive pointing requirements to adequately perform their tasks, making the need for accurate, stable and efficient attitude control methods of high importance.

Traditional attitude control methods such as Proportional-Integral-Derivative (PID) control allow for a stable, accurate solution to be found through linearisation and tuning of gains around a steady-state condition. This solution can perform well while operating close to the linearisation point [4]. However, if the mission requirements requires diverging significantly from the steady-state condition then

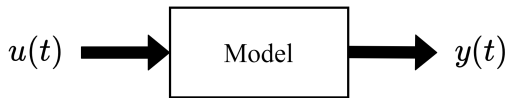


Fig. 1 Forward simulation

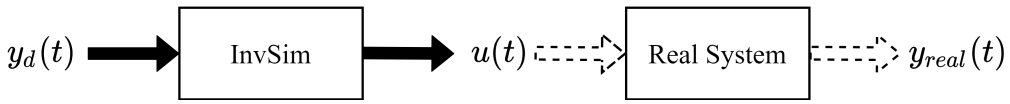


Fig. 2 Inverse simulation controller

re-tuning or gain scheduling may be required [5]. Additionally a PID controller is "unintelligent", meaning that it will always attempt to follow the commanded attitude even if it is unrealisable due to system dynamics or actuator saturation. In cases where pointing of nanosatellite instrumentation is mission critical and needs to be ensured, such an "unintelligent" approach may not be sufficient. Inverse simulation (InvSim), which is proposed in this work, provides an alternative "intelligent" solution that can ensure pointing constraints are met throughout an attitude slew manoeuvre.

Conventional forward simulations map a set of input controls $\boldsymbol{u}$ to a set of corresponding output controlled states $\boldsymbol{y}$ through integration of a mathematical model. This forward simulation, as seen in fig. 1, allows for detailed study and analysis of the dynamics of a system when subjected to a time-series of inputs/controls.

Inverse Simulation (InvSim) conversely maps the controlled states $\boldsymbol{y}$ to corresponding inputs controls $\boldsymbol{u}$. This allows for a time-series of desired controlled states $\boldsymbol{y}_d$ to be mapped to the required inputs $\boldsymbol{u}$ needed to achieve those controlled states. When placed into the configuration shown in fig. 2, InvSim can be used as an offline, open-loop controller that will produce the required controls to follow a trajectory defined by the time-series $\boldsymbol{y}_d(t)$.

InvSim techniques have already been widely utilised within research fields such as rotorcraft [6–10] and fixed-wing aircraft [11–14]. There has been limited use of InvSim within the space field such as applications in the control of a four-wheeled planetary rover [15, 16] and orbital rendezvous and docking [17]. There are two main methodologies for InvSim: the *Differential method* and the *Integral method*.

The differential method was the first to be proposed and is primarily an analytical approach involving converting the differential equations into a series of algebraic equations that can be reduced and solved iteratively. This however is model-specific, with knowledge of the plant being key to the derivation of the solution. Unique solutions have been developed previously for vehicles such as fixed wing aircraft [12] and helicopters [6], however no general differential solution exists. The solution process also makes use of numerical time differentiation which is known throughout literature to suffer from poor accuracy and instability [18].

A new InvSim method was first posed by Hess and Gao in the early 1990's [7, 8, 14] based on numerical integration methods known to be highly accurate and stable. The integral method, as it became known, allowed for a general solution to be formed, not requiring any inside knowledge into the plant, therefore working with a black box model. This method does however require substantially greater computation power when compared to its differential counterpart [19]. This makes the integral method less suitable for online real-time deployment as substantial computational power would be required on the nanosatellite's hardware. Instead the solution can be calculated beforehand and then executed by the vehicle to perform the desired manoeuvre. The general integral method is now the most common method due to its ease of implementation, versatility and stable solution. This will be the methodology implemented in the design of the attitude controller detailed in this paper.

This paper will investigate the feasibility and performance of InvSim as an attitude control method for inertial and spin-axis pointing of a nanosatellite. The paper will model the nanosatellite as a rigid body, with purely numerical simulations being used as a proof of concept. Additionally, it is assumed that perfect modelling of the nanosatellite is possible and that no parametric or non-parametric uncertainties exist.

The rest of this paper is organised as follows: section 2 will outline the rigid body mathematical model used to test the InvSim attitude controller. The integral InvSim algorithm used to develop the attitude controller is fully described in section 3. Section 4 describes two scenarios and a variety attitude paths used to test the validity and performance of the InvSim controller. Two PID controllers, one for each scenario, are developed in section 5.1 to allow for comparisons in performance and practicality to be made to the InvSim controller developed. The specific quantitative metrics used to draw comparisons between the InvSim and PID controllers are defined in section 5.2. The results and discussion follows in section 6 which includes the performance, feasibility and practicality of

using each controller design. Finally section 7 summarises the conclusions of the paper.

2. Mathematical Model

The nanosatellite is represented as a rigid body, equipped with a set of m reaction-wheel actuators, and modelled through the following equations of motion:

$$\frac{d\boldsymbol{\omega}_b}{dt} = \mathbf{J}_b^{-1}\left[\boldsymbol{\tau}_a - \boldsymbol{\omega}_b \times (\mathbf{J}_b\boldsymbol{\omega}_b)\right] \tag{1}$$

$$\boldsymbol{\tau}_a = -\mathbf{A}_w\dot{\boldsymbol{h}}_w - \boldsymbol{\omega}_b \times (\mathbf{A}_w\boldsymbol{h}_w) \tag{2}$$

$$\mathbf{A}_w = [\hat{\boldsymbol{w}}_1\hat{\boldsymbol{w}}_2 \cdots \hat{\boldsymbol{w}}_m] \tag{3}$$

$$\frac{d\dot{\boldsymbol{h}}_w}{dt} = \frac{\boldsymbol{u} - \dot{\boldsymbol{h}}_w}{T_w} \tag{4}$$

$$|\boldsymbol{u}| < u_{\text{sat}}, \quad |\boldsymbol{h}_w| < h_{w,\text{sat}} \tag{5}$$

The Euler rotational equation (eq. (1)) gives the time derivative of the body rates, $\boldsymbol{\omega}_b$, defined within the body axis set $\mathcal{F}^b$ and relative to the inertial frame $\mathcal{F}^i$. $\mathbf{J}_b$ is the inertia tensor of the rigid-body nanosatellite and $\boldsymbol{\tau}_a$ is the effective actuator torque supplied by the set of reaction wheels, as defined in eq. (2). The reaction wheel torque $\dot{\boldsymbol{h}}_w$, and angular momenta $\boldsymbol{h}_w$ of each wheel are mapped to the body axis through the actuator distribution matrix $\mathbf{A}_w$. The actuator distribution matrix (eq. (3)) is comprised of m number of unit column vectors $\hat{\boldsymbol{w}}_1, \hat{\boldsymbol{w}}_2, \ldots \hat{\boldsymbol{w}}_m$ each pointing along the spinning axis of a reaction wheel within the body axis of the nanosatellite. The effective torque also includes the coupling term $\boldsymbol{\omega}_b \times (\mathbf{A}_w\boldsymbol{h}_w)$ which is caused by the interaction between the reaction wheel's angular momentum, $\boldsymbol{h}_w$, and the nanosatellite's body rate. The control signal, $\boldsymbol{u}$, affects the time derivative of the wheel torque through the first order dynamics, seen in eq. (4), characterised by the wheel's time constant T_w. The actuator saturation applied to the commanded control torque u_{sat} and the wheel momentum $h_{w,\text{sat}}$ is also included in the model (eq. (5)).

The Z-Y-X asymmetric Euler set, defined with Euler angles $[\phi\ \theta\ \psi]^T$ defines the attitude matrix $\mathbf{A}_{zyx}$:

$$\mathbf{A}_{zyx} = \mathbf{A}_x(\phi)\mathbf{A}_y(\theta)\mathbf{A}_z(\psi) \tag{6}$$

The attitude is defined as the rotation from the inertial axis set $\mathcal{F}^i$ to the nanosatellite's body fixed axis $\mathcal{F}^b$. The corresponding Euler kinematics are given:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \tan\theta\sin\phi & \tan\theta\cos\phi \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sec\theta\sin\phi & \sec\theta\cos\phi \end{bmatrix} \begin{bmatrix} \omega_{b,x} \\ \omega_{b,y} \\ \omega_{b,z} \end{bmatrix} \tag{7}$$

Where $[\dot{\phi}\ \dot{\theta}\ \dot{\psi}]^T$ are the Euler angle rates, $[\phi\ \theta\ \psi]^T$ are the Euler angles and $\boldsymbol{\omega}_b$ is the nanosatellite's body rate.

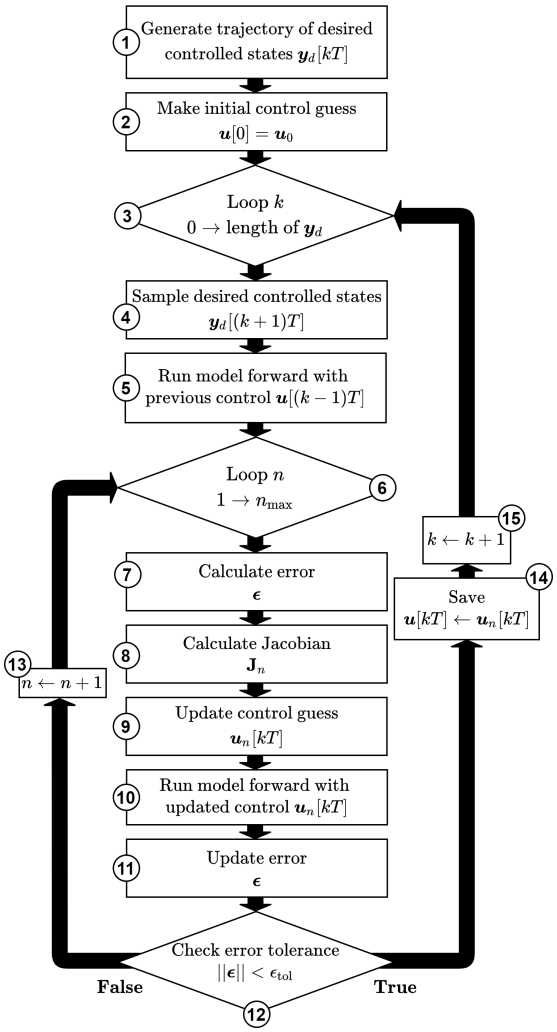


Fig. 3 Integral inverse simulation algorithm

3. Methodology

The InvSim integral method is detailed in a flow chart in fig. 3 with each step numbered for reference. Step 1 generates the desired controlled states that the InvSim controlled is required to follow. Choice of controlled states can be made depending on the specific mission requirements and can also have significant impact on performance of the algorithm. This will be explored later in section 4 when specific test cases are set out.

An initial guess for the control solution at $t = 0$, $\boldsymbol{u}_0$, is made as a starting point for the algorithm to begin solving the first control timestep. The algorithm now enters the outer loop (step 3) which will iterate through the timeseries of desired controlled states and solve the control vector $\boldsymbol{u}[kT]$ for each control timestep kT.

The objective of the algorithm is to find the control, $\boldsymbol{u}[kT]$, that will be applied to the actuators for T seconds driving the controlled states $\boldsymbol{y}$ to match the desired controlled states, $\boldsymbol{y}_d$, at time $(k+1)T$, giving the required constraint:

$$\boldsymbol{y}[(k+1)T] = \boldsymbol{y}_d[(k+1)T] \tag{8}$$

First, the next set of controlled states, $\boldsymbol{y}[(k+1)T]$, is sampled from the desired trajectory (step 4) and can be considered the current target for the InvSim algorithm. The model is integrated forward in time using the control solution from the previous timestep, $\boldsymbol{u}[(k-1)T]$, as an input (step 5) and the resulting controlled states, $\boldsymbol{y}$, are obtained. If this is the first timestep being solved for, then the initial control guess, $\boldsymbol{u}_0$, is used.

Step 6 is the start of the iterative Newton-Raphson loop, or the inner loop, which will calculate the solution to the constraint seen in eq. (8). First, the error between the actual and desired controlled states, $\boldsymbol{\epsilon}$, is calculated (step 7):

$$\boldsymbol{\epsilon} = \boldsymbol{y}[(k+1)T] - \boldsymbol{y}_d[(k+1)T] \tag{9}$$

Step 8 calculates the Jacobian, $\mathbf{J}_n$, of first order partial derivatives of each controlled state, y_i, with respect to each control, u_j, for p number of controlled states and m number of controls:

$$\mathbf{J}_n = \begin{bmatrix} \frac{\delta y_1}{\delta u_1} & \frac{\delta y_1}{\delta u_2} & \cdots & \frac{\delta y_1}{\delta u_m} \\ \frac{\delta y_2}{\delta u_1} & \frac{\delta y_2}{\delta u_2} & \cdots & \frac{\delta y_2}{\delta u_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\delta y_p}{\delta u_1} & \frac{\delta y_p}{\delta u_2} & \cdots & \frac{\delta y_p}{\delta u_m} \end{bmatrix} \tag{10}$$

The partial derivatives are calculated through numerical differencing:

$$\left.\frac{dy_i}{du_j}\right|_{k+1} = \frac{y_i[u_j + \Delta u_j]|_{k+1} - y_i[u_j - \Delta u_j]|_{k+1}}{2\Delta u_j} \tag{11}$$

where Δu_j is a perturbation applied to the control vector. The perturbed control signals, $u_j \pm \Delta u_j$, are integrated forward through the model and will produce perturbations in the resultant controlled states, $y_i \pm \Delta y_i$. These can then be used in eq. (11) to obtain the partial derivatives of each controlled state, y_i, with respect to each control, u_j. The vector of control perturbations is calculated as a fraction of the previous control solution:

$$\Delta \boldsymbol{u} = T\boldsymbol{u}[(k-1)T] \tag{12}$$

where T is the control timestep. A minimum perturbation size is ensured through the constraint:

$$\Delta u_j > \Delta u_{\min} \tag{13}$$

where $\Delta u_{\min}$ is the minimum perturbation size. This must be significantly large to sufficiently perturb the controlled states and avoid rounding errors through similar differencing in eq. (11). To ensure that the perturbed control signals do not exceed the actuator saturation of the model (section 2) the following two additional constraints are applied:

$$u_j + \Delta u_j < u_{\text{sat}} \tag{14}$$

$$u_j - \Delta u_j < u_{\text{sat}} \tag{15}$$

Once the Jacobian has been calculated, the control guess can be updated using the inverse Jacobian $\boldsymbol{J}_n^{-1}$ through (step 9):

$$\boldsymbol{u}_n[kT] = \boldsymbol{u}_{n-1}[kT] + \boldsymbol{J}_n^{-1}\boldsymbol{\epsilon} \tag{16}$$

where $\boldsymbol{u}_{n-1}[kT]$ is the previous control guess.

If the number of controlled states, p, is equal to the number of controls, m, then the Jacobian is a square matrix, the inverse of which can be calculated explicitly. However, in the case that the Jacobian is non-square, $p \neq m$, then the Moore-Penrose pseudo-inverse must be used [20]. Using the updated control guess, $\boldsymbol{u}_n[kT]$, the model is integrated forward to obtain corresponding updated values for the controlled states, $\boldsymbol{y}[(k+1)T]$ (step 10). The error between the desired controlled states and the actual controlled states, $\boldsymbol{\epsilon}$, is updated, eq. (9) (step 11), and then checked against a tolerance $||\boldsymbol{\epsilon}|| < \epsilon_{\text{tol}}$ (step 12). If the error satisfies the tolerance, then the current control guess $\boldsymbol{u}_n[kT]$ is accepted and saved (step 14) before the algorithm iterates to the next control timestep (step 15) and the solution process begins again (step 3). If the tolerance is not met, then the Newton-Raphson iterates (step 13) to find a new updated control guess (step 6). The Newton-Raphson loop continues to iterate until the tolerance, ϵ_{tol}, is met or a maximum number of Newton-Raphson iterations is reached, $n_{\max}$.

The control timestep T is the most important parameter of the InvSim algorithm to be selected. T must be sufficiently short to allow for a smooth resultant control signal $\boldsymbol{u}[kT]$ while long enough to allow for proper propagation of the controlled states $\boldsymbol{y}$ through the model to ensure numerical stability in the partial derivatives of the Jacobian. The integration timestep, dt, used to solve the mathematical model should satisfy $dt << T$ to ensure an accurate model. Reducing ϵ_{tol} and increasing $n_{\max}$ can allow for higher accuracy of the Newton-Raphson loop, particularly in the presence of nonlinearities in the system. Finally the initial control guess $\boldsymbol{u}_0$ can also be chosen, however in most cases simply setting this to zero works well.

4. Test Scenarios

Two scenarios are used to show the validity and relative performance of the InvSim control algorithm in different operating conditions. The first scenario simply considers inertial pointing of the rigid-body's x-axis, $\boldsymbol{x}_b$, along which the nanosatellite's instrumentation can be considered to be mounted. The path traced by the x-axis can be defined in spherical coordinates through the pitch (θ) and yaw (ψ) Euler angles as defined in section 2. The second scenario considers the same pointing of the x-axis while simultaneously attempting to hold a constant spin rate ω_{spin} around the x-axis. This would be applicable to a spin-stabilised nanosatellite with instruments mounted along the stabilised x-axis. The spin-axis pointing scenario also allows for the InvSim controller to be tested in the presence of higher coupling in the dynamics caused by the significant constant spin rate.

4.1. Desired Trajectories

In both scenarios the InvSim controller will be required to follow four paths defined in the $\theta - \psi$ plane and can be seen in fig. 4. Each path is defined through a set of piece-wise 4th order polynomials:

$$\text{Position:} \quad x = at^4 + bt^3 + ct^2 + dt + e \tag{17a}$$

$$\text{Velocity:} \quad \dot{x} = 4at^3 + 3bt^2 + 2ct + d \tag{17b}$$

$$\text{Acceleration:} \quad \ddot{x} = 12at^2 + 6bt + 2c \tag{17c}$$

$$\text{Jerk:} \quad \dddot{x} = 24at + 6b \tag{17d}$$

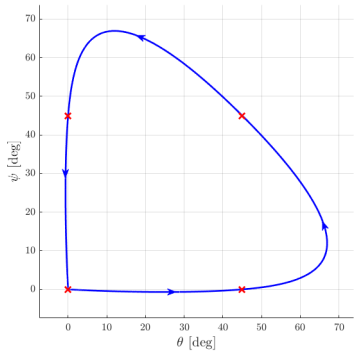
Where a, b, c, d and e are coefficients that must be solved using the following series of constraints:

- Continuity of the position (eq. (17a)), velocity (eq. (17b)) and acceleration (eq. (17c)) between each piece-wise polynomial. This ensures a realisable, continuous resultant control signal.
- Set of way-points $\mathbf{P}_{\text{wps}}$ giving the position in the $\theta - \psi$ plane at the end of each piece-wise polynomial:

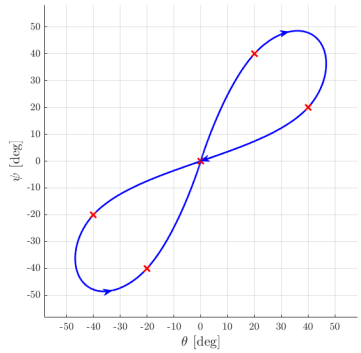
$$\mathbf{P}_{\text{wps}} = \begin{bmatrix} \theta_1 & \psi_1 \\ \theta_2 & \psi_2 \\ \vdots & \vdots \\ \theta_f & \psi_f \end{bmatrix} \tag{18}$$

  Which can be seen in fig. 4.
- Minimisation of discontinuity in the jerk (eq. (17d)) is also included to ensure a unique solution is found.
- The time given for completion of each piece-wise polynomial or the time-to-move, t_{move}.

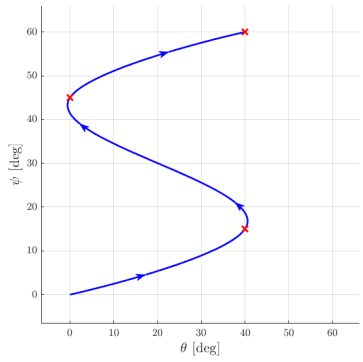
Additionally once each trajectory is completed the final way-point is held for a time t_{hold}, to show the capability of the InvSim controller to hold a constant attitude. The values of time-to-move, t_{move}, time-to-hold, t_{hold}, and pointing axis (x-axis) spin rate, ω_{spin}, for the spin-axis pointing scenario, can be seen given in table 1.



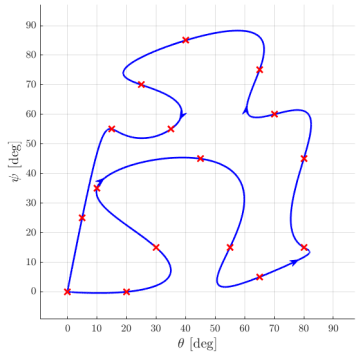
(a) 45°box



(b) Figure of 8



(c) Slalom

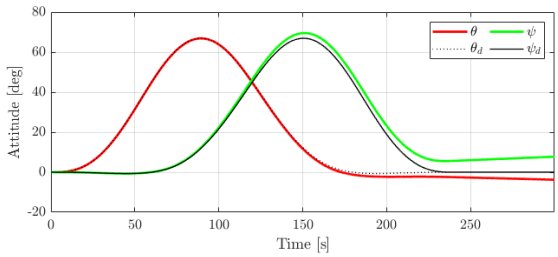


(d) Long path

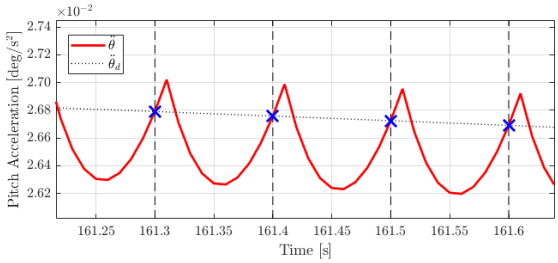
Fig. 4 Trajectory paths

Table 1 Test scenario parameters

Time-to-move, t_{move}	60 s
Time-to-hold, t_{hold}	60 s
Spin-axis spin rate, ω_{spin}	3 °/s



(a) Attitude



(b) Pitch acceleration oscillations

Fig. 5 Acceleration controlled state oscillations

4.2. Choice of Controlled States

The choice of using position, velocity or acceleration profiles for the set of desired controlled states ($\boldsymbol{y}_d$) has significant impact on the performance and stability of the InvSim algorithm.

Perturbations in the controls will have a greater and quicker effect on higher order derivatives of the trajectory such as the velocity and to a greater extent the acceleration [9]. This results in higher numerical stability when calculating the partial derivatives (eq. (11)) of the Jacobian (eq. (10)) and thus velocity and acceleration profiles are a more attractive option for the desired controlled states $\boldsymbol{y}_d$.

However, it was found that using Euler accelerations caused divergence between control timesteps kT in the presence of highly-coupled dynamics, i.e. high body rate manoeuvres. Fig. 5b shows divergence from the desired pitch acceleration between successive control timesteps kT. Note that the InvSim algorithm is performing as intended, matching the desired acceleration at each timestep kT. However, the errors in the acceleration profile accumulate over time causing substantial divergences from the desired path as seen in fig. 5a. This issue can be mitigated by reducing the size of the control timestep T, so long as the frequency of control commands ($f_c = 1/T$) can be realised by the actuators. This would also require further reduction of the integration timestep dt to maintain a stable InvSim solution, which in turn vastly increases computation time; therefore, velocity profiles are the best compromise for accuracy and stability of the InvSim algorithm for both the inertial and spin-axis pointing scenarios.

For the inertial pointing scenario, the desired controlled states for the InvSim algorithm can therefore simply be defined as the Euler angle velocities:

$$\boldsymbol{y}_d[kT] = \left[\dot{\phi}_d[kT] \;\; \dot{\theta}_d[kT] \;\; \dot{\psi}_d[kT]\right]^T \tag{19}$$

Note that in this scenario the desired Euler roll rate, $\dot{\phi}_d$, will always be zero as the trajectory path is only defined in the $\theta - \psi$ plane.

For the spin-axis pointing scenario, the desired Euler roll rate, $\dot{\phi}_d$, is be replaced with the desired rate around the body axis $(\omega_{b,x})_d$, as a constant spinning velocity is to be maintained while the trajectory path is followed:

$$\boldsymbol{y}_d[kT] = \left[(\omega_{b,x})_d \;\; \dot{\theta}_d[kT] \;\; \dot{\psi}_d[kT]\right]^T \tag{20}$$

with the constant spinning rate is set to ω_{spin}:

$$(\omega_{b,x})_d = \omega_{\text{spin}} \tag{21}$$

which has been defined previously in table 1.

5. Performance Comparison

5.1. PID Control

Two PID control schemes are used for the inertial and spin-axis pointing scenarios using the general feedback control law defined in the Laplace domain:

$$C(s) = K_p + \frac{K_i}{s} + K_d s \tag{22}$$

Tuning of the controller gains K_p, K_i and K_d was completed using the MATLAB PID Tuning Algorithm. The algorithm aims to achieve the following objectives:

- *Closed-loop stability:* output remains bounded for a bounded input
- *Adequate performance:* fast response to changes in the reference or disturbances
- *Adequate robustness:* sufficient gain and phase margin to remain robust in the face of modelling errors and variations

The tuning algorithm selects a crossover frequency (bandwidth), ω_c, based on the system dynamics to ensure a fast response while maintaining a phase margin of 60° for robustness.

In the inertial pointing scenario, three single-input-single-output (SISO) PID controllers are employed to follow the Euler angle pitch and yaw references, θ_d and ψ_d, from

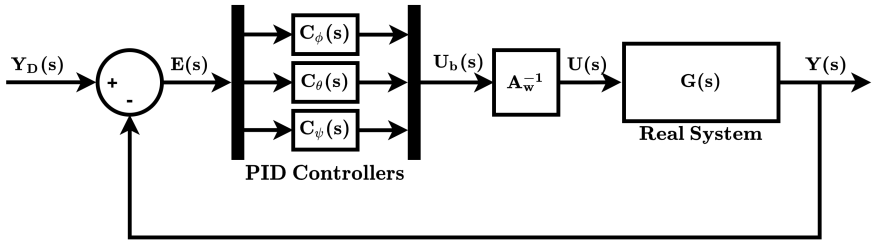


Fig. 6 PID feedback controller (inertial pointing)

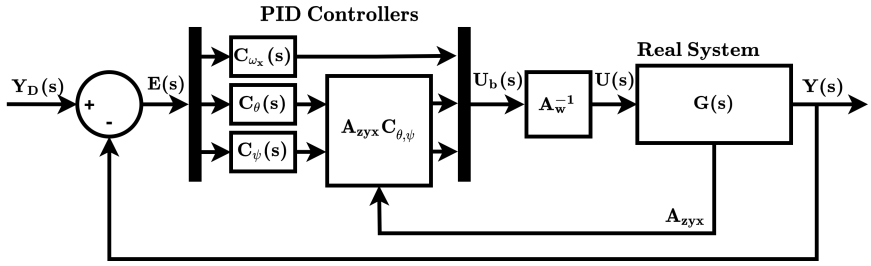


Fig. 7 PID feedback controller (spin-axis pointing)

the desired trajectories seen in section 4.1 while holding a constant roll angle, $\phi_d = 0°$. The control signals produced are approximated to be relative to the nanosatellite's body axis frame $\mathcal{F}^b$ and therefore are simply mapped to the set of reaction wheel actuators using the actuator distribution matrix $\mathbf{A}_w$, as can be seen in fig. 6.

The PID control architecture for the spin-axis pointing nanosatellite can be seen in fig. 7. It also includes three SISO PID controllers for the x-axis spin rate $\omega_{b,x}$ and Euler angles θ and ψ.

Due to the constant spin rate $\omega_{b,x}$ the direction of the nanosatellite's actuators is constantly changing with respect to the inertial axis set $\mathcal{F}^i$. The Euler angles, $[\phi\ \theta\ \psi]^T$, used to defined the trajectory are in respect to the inertial frame, $\mathcal{F}^i$. Therefore, the θ and ψ controllers require their control signals to first be transformed into the body axis set $\mathcal{F}^b$ before they can be applied to the actuators using $\mathbf{A}_w$. This is done by taking the attitude direction cosine matrix $\mathbf{A}_{\text{zyx}}$ and applying it to the signals from the $\mathbf{C}_\theta(s)$ and $\mathbf{C}_\psi(s)$ controllers.

5.2. Performance Metrics

For both the InvSim and PID cases the performance will be measured through the following quantitative metrics:

- Pointing error
- Peak control power
- Total control energy

The pointing error is defined as the shortest rotation required to move between the desired pointing direction and the actual pointing direction (i.e the x-axis direction). As mentioned previously (section 4) the attitude of the pointing x-axis of the nanosatellite is defined through the spherical coordinates by the θ and ψ Euler angles. Quaternions are utilised as a convenient way to calculate this angular pointing error [21].

Quaternions of the desired pitch and yaw rotations, $\boldsymbol{q}(\psi_d, \theta_d)$, and actual pitch and yaw rotations, $\boldsymbol{q}(\psi, \theta)$, can be constructed and then combined to get the error quaternion:

$$\boldsymbol{q}_\epsilon = \boldsymbol{q}^*(\psi_d, \theta_d) \odot \boldsymbol{q}(\psi, \theta) \tag{23}$$

where $\boldsymbol{q}_\epsilon$ is the error quaternion, $\boldsymbol{q}^*(\psi_d, \theta_d)$ is the conjugate of the desired pitch-yaw quaternion and $\odot$ is the quaternion multiplication operator [21].

The angular pointing error can then be extracted from the scalar component of the pointing error quaternion:

$$\vartheta_\epsilon = 2\cos^{-1}(q_{\epsilon,4}) \tag{24}$$

where ϑ_ϵ is the angular pointing error and $q_{\epsilon,4}$ is the scalar component of the pointing error quaternion $\boldsymbol{q}_\epsilon$.

The control power, P, for each individual reaction wheel of the nanosatellite can be found using the magnitudes of the torque, τ_{w_i}, and speed of the wheel, ω_{w_i}. This can be redefined in terms of the angular momentum, h_{w_i}, torque, $\dot{h}_{w_i}$, and inertia of each wheel, J_w. The total control power is found summing the power of all m wheels:

$$P = \sum_{i=1}^{m} |\tau_{w_i}\omega_{w_i}| = \sum_{i=1}^{m} \frac{|\dot{h}_{w_i} h_{w_i}|}{J_w} \tag{25}$$

The total control energy, E, used can be obtained integrating the control power, P, along the entire trajectory:

$$E_{\text{total}} = \int_0^{t_f} P \, dt \tag{26}$$

6. Results and Discussion

6.1. Parameters

The parameters for the mathematical model are given in table 3. For the actuator saturation u_{sat} and $h_{w,\text{sat}}$ parameters were assumed to be that of the *Hyperion RW10* reaction wheel. The time-constant, T_w, was assumed to be that of the *O.C.E Technologies RW270*. The nanosatellite's body inertia matrix, $\mathbf{J}_b$, was assumed to be that of the *UKube-1* 3U Cubesat [22]. The actuator distribution matrix, $\mathbf{A}_w$, is the *NASA Standard 4-wheel configuration* [21] as seen in fig. 8. The reaction wheel inertia, J_w, was estimated based on the maximum angular momentum and maximum speed of the *Hyperion RW10* reaction wheel. The same integration timestep dt is used for both the InvSim and PID control examples.

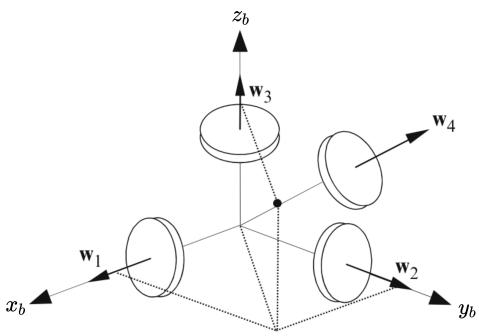


Fig. 8 NASA standard 4-wheel configuration [21]

The parameters used for the InvSim algorithm can be seen in table 2. As previously mentioned in section 4.2, T was chosen to balance stability of the algorithm and speed of computation while achieving a relatively smooth control signal. The integration timestep $dt << T$ to ensure accuracy of the mathematical model.

The tolerance, ϵ_{tol}, and maximum iterations, $n_{\max}$, allow for the accuracy of the Newton-Raphson loop to be altered. However, it was found that the current rigid body model is highly linear in most cases, meaning that very few iterations of the loop are required to achieve an accurate solution. Furthermore it was found that further reduction of ϵ_{tol} tends to result in the Newton-Raphson loop oscillating around the solution leading to unnecessary additional computational time, with no improvement in the result. In these cases, the number of iterations tends to reach the limit, $n_{\max}$, on every timestep even if limit is increased. Therefore, unless more nonlinear dynamics are present, any reduction of the tolerance or increase in the maximum iterations will not result in an improved solution, but will increase computational time.

Table 2 Inverse simulation parameters

Control timestep, T	0.1 s
Initial control guess, $\boldsymbol{u}_0$	$[0 \quad 0 \quad 0]^T$ Nm
Maximum Newton-Raphson iterations, $n_{\max}$	60
Minimum control perturbation, $\Delta u_{\min}$	1×10^{-10} Nm
Newton-Raphson tolerance, $\boldsymbol{\epsilon}_{\text{tol}}$	1×10^{-12} rad/s

In addition, it was found for the rigid body model used, and the rest-to-rest manoeuvres being followed, that a solution can converge an initial control guess $\boldsymbol{u}_0$ of zero. The minimum control perturbation, $\Delta u_{\min}$, is only important for the first few timesteps when the magnitude of the control signal is near zero. The value is chosen to ensure significant change in the controlled states to maintain a stable Jacobian.

The gains for both PID control schemes were tuned using the MATLAB PID Tuning Algorithm, as mentioned in section 5.1. The resulting tuned gains, phase margins and bandwidths of each controller are shown in table 4.

6.2. Performance

The InvSim controller illustrated in the previous sections was able to follow all of the desired paths (fig. 4) in both the inertial and spin-axis pointing scenarios. Examples of the resultant control signal and dynamic responses of the *long path* in the inertial and spin-axis pointing scenarios can be seen in figs. 9 and 10 respectively. It can be seen in both scenarios that the resultant control response is smooth and the desired controlled states are followed successfully.

Figs. 11 and 12 compare the pointing error between the InvSim and PID controllers for all paths and in both the inertial and spin-axis pointing scenarios. First, it can be noted in both the inertial and spin-axis pointing scenarios that PID, unlike the InvSim controller, has final pointing errors of zero. This is due to the online/offline deployment of each controller. The PID controller is deployed online with feedback allowing for the integral component of the controller to attenuate steady-state errors to zero. The InvSim controller conversely is deployed offline and therefore does not benefit from any feedback in real-time. Numerical errors are uncompensated for and result in divergence over time from the desired trajectory. Increasing the time that the final attitude position is held for will cause further

Table 3 Model parameters

Reaction wheel time-constant, T_w	0.1 s
Control torque saturation, u_{sat}	0.1×10^{-3} Nm
Reaction wheel momentum saturation, $h_{w,\,\text{sat}}$	6×10^{-3} Nms
Satellite inertia matrix, $\mathbf{J}_b$	$\begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.0506 & 0 \\ 0 & 0 & 0.0506 \end{bmatrix}$ kgm^2
Actuator Distribution matrix, $\mathbf{A}_w$	$\begin{bmatrix} 1 & 0 & 0 & 1/\sqrt{3} \\ 0 & 1 & 0 & 1/\sqrt{3} \\ 0 & 0 & 1 & 1/\sqrt{3} \end{bmatrix}$
Reaction wheel inertia, J_w	3.82×10^{-6} kgm^2
Integration timestep, dt	0.01 s

Table 4 PID controller parameters

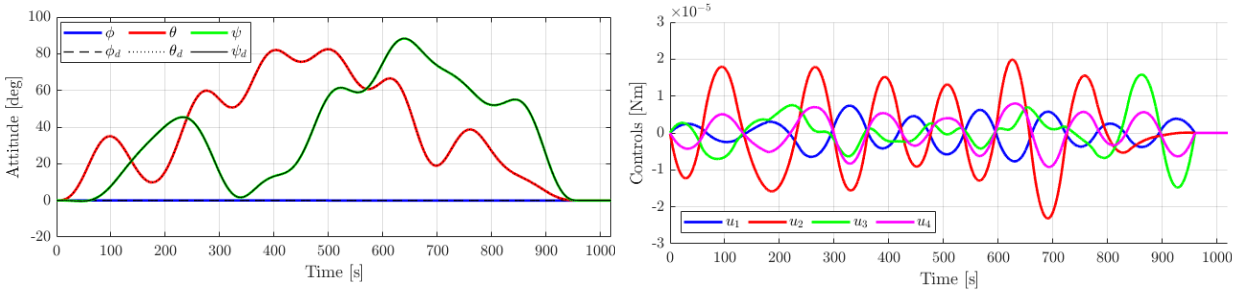
Inertial pointing nanosatellite					
	Phase Margin	Bandwidth	K_p	K_i	K_d
PID(ϕ)	60°	4.68 rad/s	0.0063	0.0003	0.0387
PID(θ)	60°	4.74 rad/s	0.1349	0.0098	0.2042
PID(ψ)	60°	4.68 rad/s	0.1760	0.0130	0.2096
Spin-axis pointing nanosatellite					
	Phase Margin	Bandwidth	K_p	K_i	K_d
PID($\omega_{b,x}$)	60°	17.73 rad/s	0.1954	0.3357	0.0128
PID(θ)	60°	0.885 rad/s	0.0149	0.0016	0.0351
PID(ψ)	60°	0.885 rad/s	0.0148	0.0016	0.0349

divergence when using the InvSim controller and increase to the final errors. This would suggest that in real-world deployment of an InvSim controller, where parametric and non-parametric model uncertainties exist, real-time feedback will need to be added to the control solution.

It can be seen in fig. 11 that the pointing error across the duration of the trajectories (tracking error) is greater for the InvSim controller for the most part. This again can be attributed to the lack of feedback in the InvSim controller design implemented in this paper. However, it should be noted that the tracking error for the InvSim across all trajectories in the inertial pointing scenario does not exceed 0.1°. The tracking errors for the spin-axis pointing scenario (fig. 12) are for the most part significantly lower with the InvSim controller compared to PID. An exception can be seen in fig. 12b for the *figure of 8* path where the pointing error diverges greatly when compared to the PID example. The InvSim controller outperforms the PID controller in the *long path* (fig. 12d) with the PID controller reaching a pointing error of over 3°, while the InvSim pointing error does not exceed much over 0.5°.

Additionally, fig. 13 shows how well both control schemes are able to hold the constant spin rate, ω_{spin}, around the pointing axis while completing the trajectory. It can be seen that the InvSim controller is better at maintaining the constant spin rate, however the PID also performs adequately reaching a maximum error of 1.7×10^{-3}°/s during the *long path*. The greater performance comparatively of the InvSim controller over the PID example in the spin-axis pointing scenario is most likely due to the greater nonlinearities introduced through the coupled dynamics. PID control is inherently linear and therefore is less well equipped to deal with the nonlinear dynamics seen with the spin-axis pointing nanosatellite. InvSim however through its iterative Newton-Raphson loop can easily solve highly nonlinear systems. However, as the dynamics become more nonlinear, the number of iterations the Newton-Raphson loop requires to find a solution will increase as will computation time.

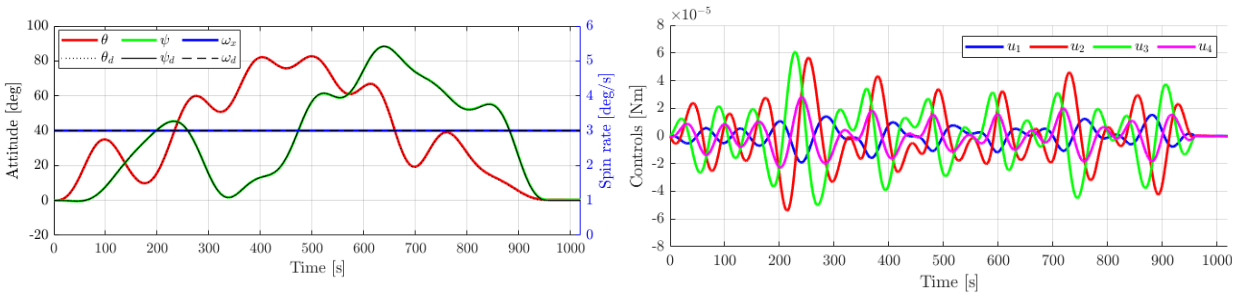
Figs. 14 and 15 show the peak actuator power and



(a) Attitude

(b) Controls

Fig. 9 InvSim - Long path (inertial pointing)



(a) Attitude and spinning rate

(b) Controls

Fig. 10 InvSim - Long path (spin-axis pointing)

total actuator energy used for all of the trajectories in the inertial and spin-axis pointing scenarios respectively. It can be seen that both the peak actuator power and total control energy in both scenarios are almost identical. The only case where there seems to be a somewhat significant difference is in the spin-axis pointing *long path* case where the total actuator energy used by the InvSim controller is 0.043 J less than in the PID example.

It should be noted when comparing the performances that the PID controller's control signal is continuous whereas the InvSim controller has a control timestep T of 0.1s. As the integration timestep in both cases is 0.01s it can be seen that the InvSim controller is following the same desired trajectory using 10 times less control actions. Given sufficient computational power and time the control timestep T for the InvSim controller could be reduced and improve performance.

6.3. Practicality

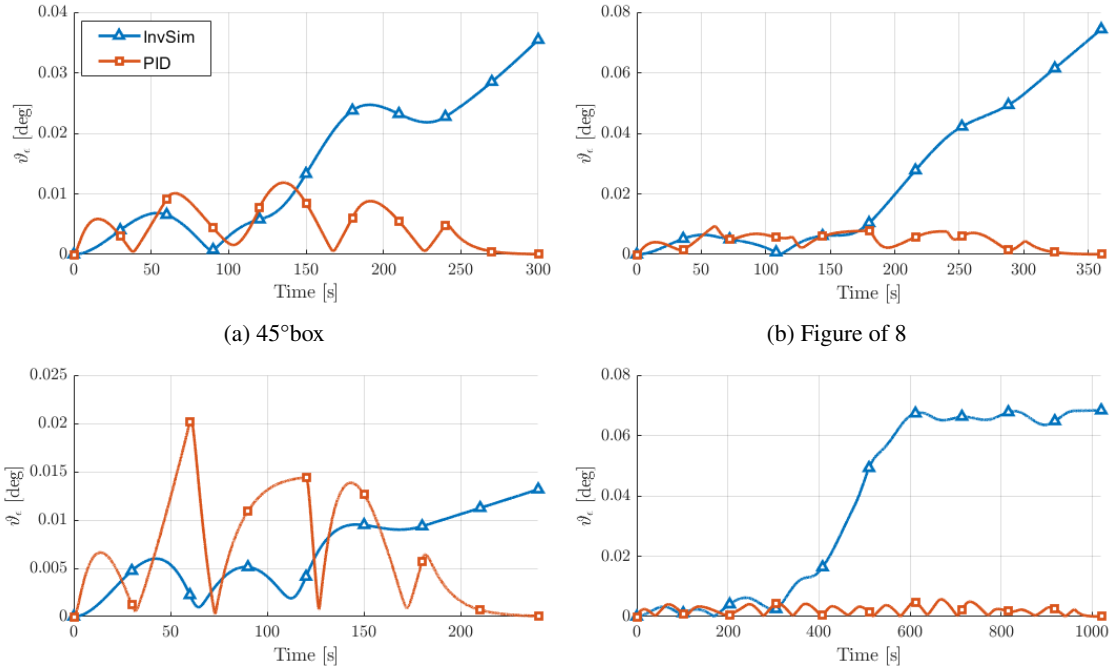
The most noticeable difference between both controllers in the spin-axis pointing scenario is the ease of implementation. When moving from the inertial pointing to the spin-axis pointing scenario, the only change required for the InvSim controller is to redefine the controlled states that the algorithm is required to follow. Instead, the PID controller used in the inertial pointing scenario cannot be easily applied to the spin-axis pointing problem. Comparison of figs. 6 and 7 shows that significant additions were required to allow for a stable PID control scheme to be derived. Each SISO PID controller also required re-tuning to ensure good performance.

Any changes to the nanosatellite itself requires no modification to the InvSim controller other than tweaking of the mathematical model being used. A PID controller on the other hand requires at minimum a re-tuning of the control gains or more drastic alterations as seen with the spin-axis pointing scenario previously. This lends the InvSim controller as a possible tool to aid in the design process of a nanosatellite allowing for changes to be made efficiently without the control system needing to be modified.

An example of this can be illustrated through considering the case where the inertia matrix of the nanosatellite is altered from that seen in table 3 to:

$$\mathbf{J}_b = \begin{bmatrix} 0.005 & 0 & 0 \\ 0 & 0.004 & 0 \\ 0 & 0 & 0.002 \end{bmatrix} \text{kgm}^2 \tag{27}$$

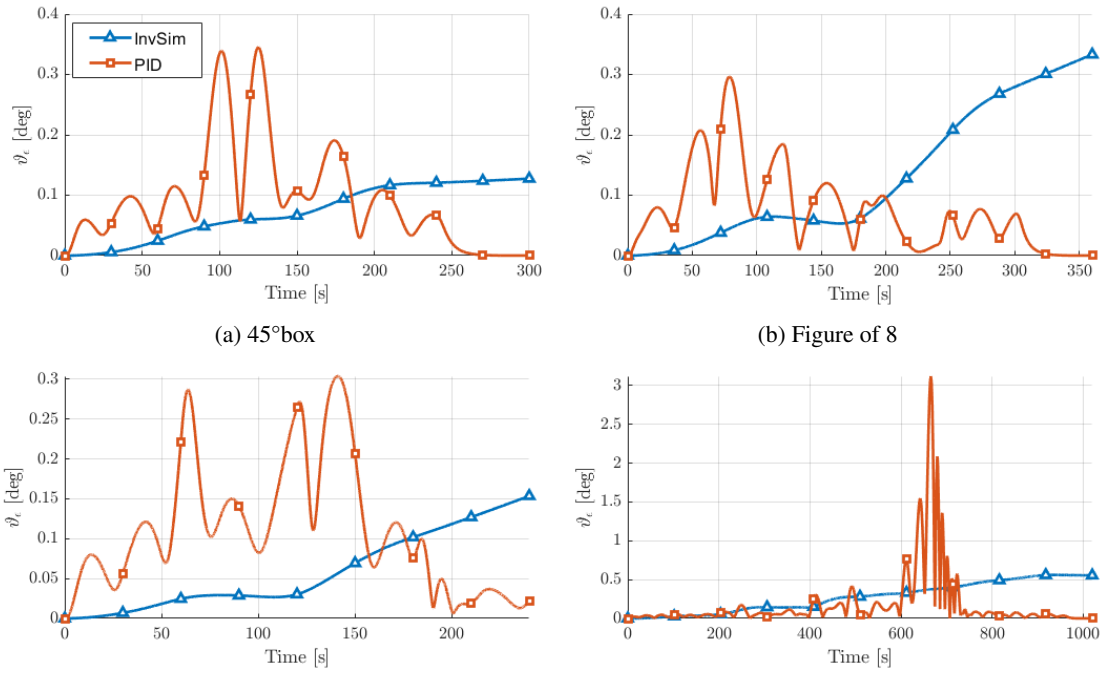
The *45°box* path and the inertial pointing scenario is used to compare the performance of the PID controller tuned previously (table 4) and the InvSim controller. The PID controller has not been re-tuned and the only change made



(a) 45°box (b) Figure of 8

(c) Slalom (d) Long path

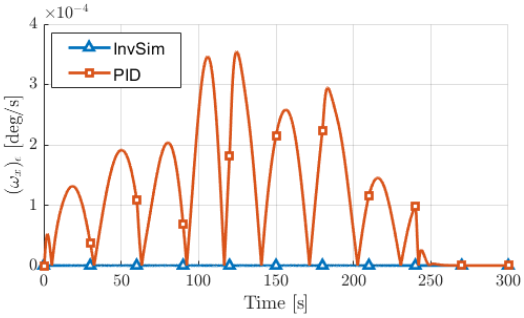
Fig. 11 Inertial pointing scenario pointing error



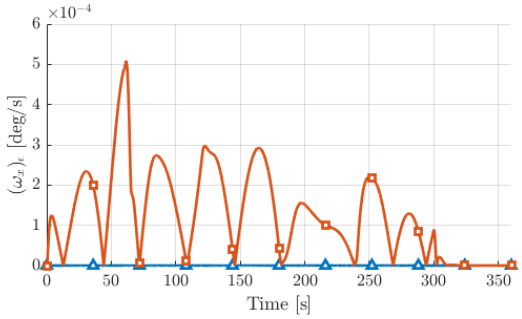
(a) 45°box (b) Figure of 8

(c) Slalom (d) Long path

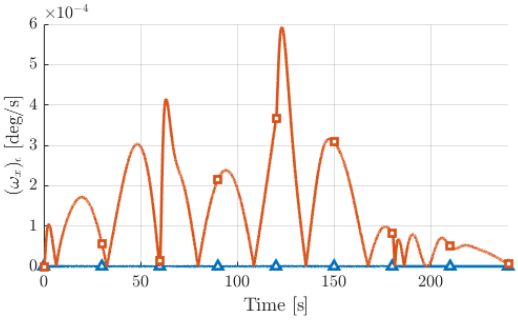
Fig. 12 Spin-axis pointing scenario pointing error



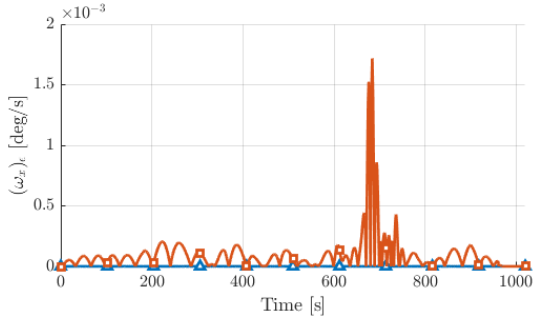
(a) 45°box



(b) Figure of 8

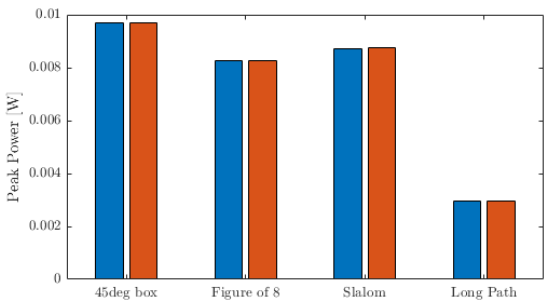


(c) Slalom

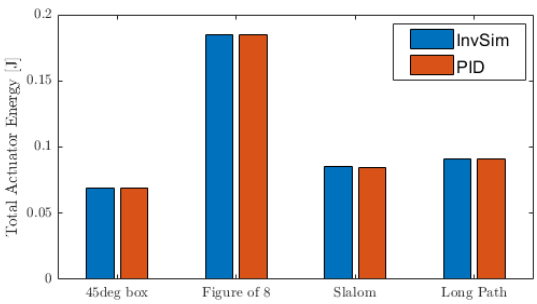


(d) Long path

Fig. 13 Spin-axis pointing scenario spin rate error

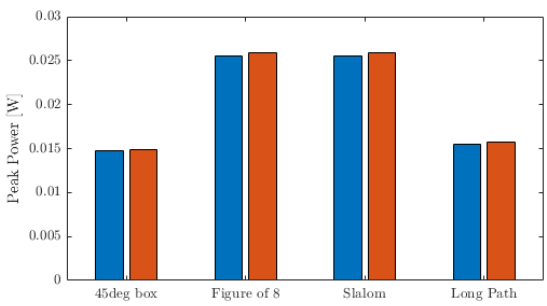


(a) Peak actuator power

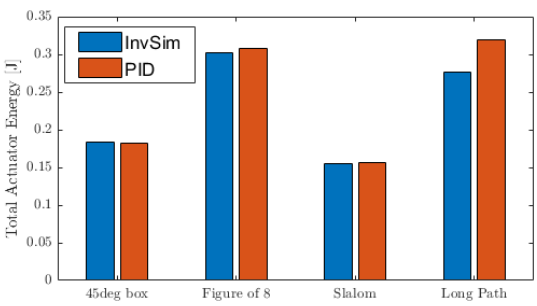


(b) Total actuator energy

Fig. 14 Inertial pointing scenario actuator power and energy

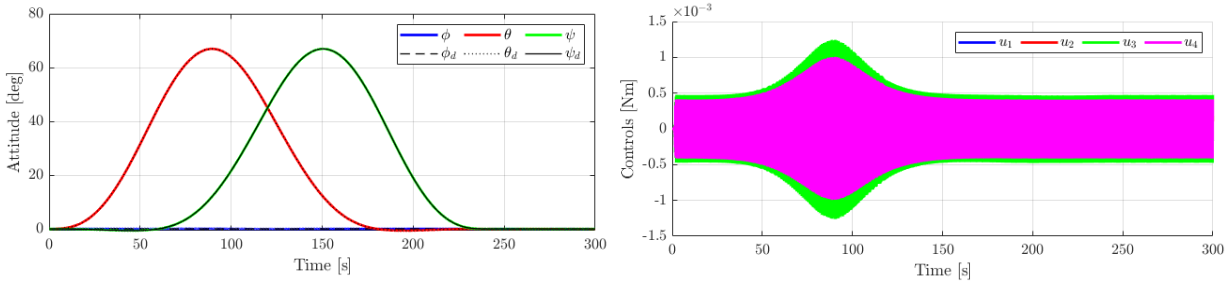


(a) Peak actuator power



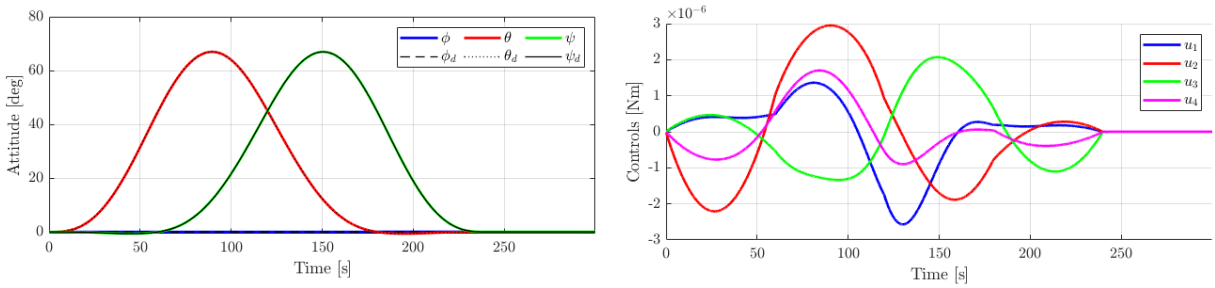
(b) Total actuator energy

Fig. 15 Spin-axis pointing scenario actuator Power and energy



(a) Attitude (b) Controls

Fig. 16 PID - Inertia changed, no re-tuning



(a) Attitude (b) Controls

Fig. 17 InvSim - Inertia changed

to the InvSim controller is to update the inertia matrix used in the mathematical model.

Fig. 16a shows that the PID controller is still capable of following the desired trajectory despite no re-tuning of the controller gains. However the resultant control signal that the PID controller produces (fig. 16b) contains high frequency oscillations that are highly undesirable. Not only would this control signal most likely not be realisable by the actuators but the peak actuator power is 5622% that of the peak power of the InvSim controller and the total actuator energy is 3473% that of the InvSim controller.

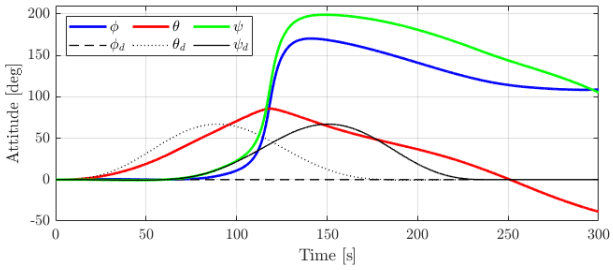
As can be seen in fig. 17 the InvSim controller follows the desired trajectory and produces a smooth control signal. It can therefore be seen that performance of the controller can easily be ensured even as the design of the nanosatellite itself is altered. The PID controller conversely requires complete re-tuning of the controller gains every time an alteration to the nanosatellite design is made. This shows the versatility and practicality of using InvSim not only as an attitude controller but also as a design aid. Introduction of InvSim as the chosen controller removes the need for any control design iterations to be made as changes are made the rest of the nanosatellite's design.

The InvSim algorithm is also capable of rejecting and flagging desired trajectories given that are unrealisable due to the nanosatellite's dynamics and/or actuator size. A PID controller however is unintelligent when given an unrealisable trajectory and will instead attempt to follow it as best it can. This could result in the pointing requirements of the nanosatellite not being met and cause the nanosatellite to be unsuccessful in its mission. This allows for InvSim to also be an informative tool within the iterative design process of a nanosatellite. Not only does the InvSim controller allow for easy changes to the other subsystems of the nanosatellite, as mentioned previously, but it can also actively provide useful feedback on whether the design can achieve the required instrument pointing requirements.

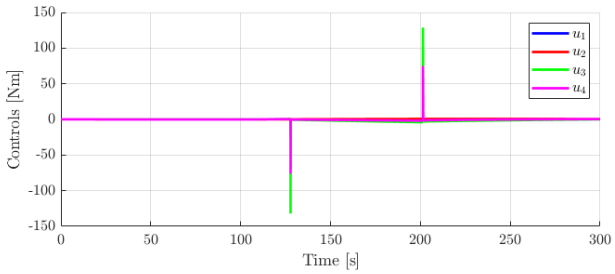
This can be illustrated by increasing the nanosatellite's inertia matrix to:

$$\mathbf{J}_b = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \text{kgm}^2 \tag{28}$$

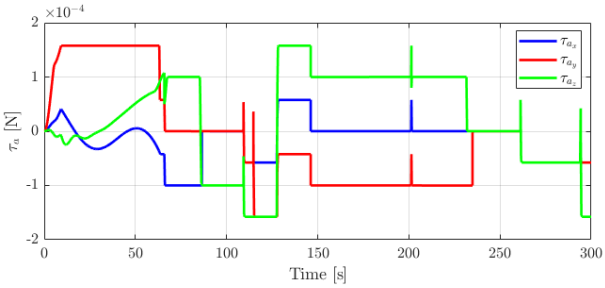
Both controllers are required to follow the *45°box* path in the inertial pointing scenario. The increased inertia causes the required control torque to exceed the actuator saturation, u_{sat} and $h_{w\,,\text{sat}}$, seen in table 3. The InvSim algorithm is able to detect that this saturation is reached and throws an error message indicating that the desired trajectory given is not realisable with the current system dynamics and/or actuator power. This information can be used within the design process to inform engineers that larger actuators, alterations to other subsystems, a redesign of the desired trajectory or a combination of all these



(a) Attitude



(b) Controls



(c) Equivalent actuator torque

Fig. 18 PID - Actuator saturation

changes is required. Additionally the InvSim algorithm will indicate that the trajectory is not possible before it begins to execute the manoeuvre. This could be especially important when deviation from the trajectory would result in catastrophic mission failure. This could be conceivable in the case where there is little margin in the power budget of the nanosatellite and miss-pointing of solar-panels could result in loss of the spacecraft.

The PID controller however is unable to deal with actuator saturation in such an ‘intelligent” way. Instead when actuator saturation is reached, as can be seen through the equivalent actuator torque in fig. 18c, the PID controller simply demands more control effort which can not be supplied (fig. 18b). The resultant attitude response can be seen in fig. 18a which shows complete failure of the PID controller to follow the desired trajectory.

7. Conclusions

In conclusion it has been shown that the integral Inverse Simulation (InvSim) algorithm can successfully be used to produce an attitude control system for a rigid body nanosatellite. Peak power and total actuator energy across all paths and in both the inertial and spin-axis pointing scenarios for the InvSim controller is almost identical to the Proportional-Integral-Derivative (PID) control example. Lack of any feedback within the InvSim controller results in the inability to attenuate the pointing error to zero and therefore a significant final pointing error that will continue to diverge exists.

Tracking pointing error performance of the InvSim controller when compared to PID control is mixed. In the inertial pointing scenario it is clear that the PID controller outperforms InvSim due to the highly linear dynamics. However, in the spin-axis pointing scenario where more nonlinear dynamics exist the InvSim controller maintains a lower tracking error in several of the paths, in particular the *long path*. Additionally InvSim performs slightly better when maintaining the constant spinning rate around the pointing axis in the spin-axis pointing scenario.

It has been shown however that the greatest advantage of using InvSim as a control solution is the practicality and versatility of its implementation and use. InvSim is particularly applicable within an iterative design process where changes to the nanosatellite design affecting the dynamics are constantly being made. InvSim does not require any laborious re-tuning of control gains when changes are made to the nanosatellite design in order to maintain control performance. Instead the changes simply need to be included in the mathematical model being used to drive the inverse solution. Also, in the presence of highly nonlinear dynamics a PID control scheme would require some sort of gain-scheduling between linearisation points to perform adequately which is not required when using InvSim.

Additionally, InvSim is capable of informing engineers when the given desired trajectory is not possible with the current nanosatellite design. This will give useful information suggesting that larger actuators, redesign of the desired trajectory or other changes need to be made.

The InvSim algorithm is calculated offline so it will not attempt to follow a trajectory until it has been deemed possible to follow. Conversely PID control would try to follow the trajectory the best it can, either with degraded performance or complete failure.

Overall InvSim provides unique practical properties over other traditional control methods such as PID. It's performance is adequate, however real-world implementation will require inclusion of some form of feedback to deal with parametric and non-parametric model uncertainties as-well as disturbances.

References

[1] M. N. Sweeting, "Modern Small Satellites-Changing the Economics of Space," *Proceedings of the IEEE*, vol. 106, no. 3, pp. 343–361, 2018.

[2] D. Selva and D. Krejci, "A survey and assessment of the capabilities of Cubesats for Earth observation," *Acta Astronautica*, vol. 74, pp. 50–68, 2012, ISSN: 0094-5765.

[3] A. Poghosyan and A. Golkar, "CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions," *Progress in Aerospace Sciences*, vol. 88, pp. 59–83, 2017, ISSN: 0376-0421.

[4] G. C. Goodwin, S. F. Graebe, M. E. Salgado, *et al.*, *Control system design*. Prentice Hall New Jersey, 2001, vol. 240.

[5] M. A. Johnson and M. H. Moradi, *PID control*. Springer, 2005.

[6] D. Thomson and R. Bradley, "Development and verification of an algorithm for helicopter inverse simulation," *Vertica*, vol. 14, no. 2, pp. 185–200, 1990.

[7] R. A. Hess, C. Gao, and S. H. Wang, "Generalized technique for inverse simulation applied to aircraft maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 14, no. 5, pp. 920–926, Sep. 1991.

[8] R. A. Hess and C. Gao, "Generalized algorithm for inverse simulation applied to helicopter maneuvering flight," *Journal of the American Helicopter Society*, vol. 38, no. 4, pp. 3–15, 1993, ISSN: 00028711.

[9] S. Rutherford and D. G. Thomson, "Improved methodology for inverse simulation," *The Aeronautical Journal (1968)*, vol. 100, no. 993, pp. 79–86, 1996, ISSN: 0001-9240.

[10] G. Avanzini, D. Thomson, and A. Torasso, "Model predictive control architecture for rotorcraft inverse simulation," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 1, pp. 207–217, Dec. 2013, ISSN: 15333884.

[11] R. T. Jones, "A Simplified Application of the Method of Operators to the Calculation of Disturbed Motions of an Airplane," Tech. Rep., 1936.

[12] O. Kato and I. Sugiura, "An interpretation of airplane general motion and contol as inverse problem," *Journal of Guidance, Control, and Dynamics*, vol. 9, no. 2, pp. 198–204, Mar. 1986.

[13] O. Kato, "Attitude projection method for analyzing large-amplitude airplane maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 13, no. 1, pp. 22–29, Jan. 1990.

[14] C. Gao and R. A. Hess, "Inverse simulation of large-amplitude aircraft maneuvers," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 4, pp. 733–737, Jul. 1993.

[15] K. Worrall, D. Thomson, and E. Mcgookin, "Application of Inverse Simulation to a wheeled mobile robot," in *ICARA 2015 - Proceedings of the 2015 6th International Conference on Automation, Robotics and Applications*, 2015.

[16] K. J. Worrall, D. G. Thomson, E. W. McGookin, and T. Flessa, "Autonomous Planetary Rover Control using Inverse Simulation," in *13th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2015)*, 2015.

[17] W. Zhou, H. Wang, G. Tang, and S. Guo, "Inverse simulation system for manual-controlled rendezvous and docking based on artificial neural network," *Advances in Space Research*, vol. 58, no. 6, pp. 938–949, 2016, ISSN: 0273-1177.

[18] K. LIN, P. LU, and M. SMITH, "The numerical errors in inverse simulation," in *Flight Simulation and Technologies*, ser. Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics, Aug. 1993.

[19] D. Thomson and R. Bradley, *Inverse simulation as a tool for flight dynamics research-Principles and applications*, 2006.

[20] J. C. A. Barata and M. S. Hussein, "The Moore–Penrose pseudoinverse: A tutorial review of the theory," *Brazilian Journal of Physics*, vol. 42, no. 1-2, pp. 146–165, 2012.

[21] F. L. Markley and J. L. Crassidis, *Fundamentals of spacecraft attitude determination and control*. New York, NY: Springer New York, 2014, pp. 1–486, ISBN: 9781493908028.

[22] J. Kim and K. Worrall, "Sun tracking controller for UKube-1 using magnetic torquer only*," *IFAC Proceedings Volumes*, vol. 46, no. 19, pp. 541–546, 2013, ISSN: 1474-6670.