Many-Revolution Earth-Centred Solar-Sail Trajectory Optimisation Using Differential Dynamic Programming

Leemans, G.; Carzana, L.; Heiligers, M.J.

DOI
10.2514/6.2022-1776

Publication date
2022

Document Version
Final published version

Published in
AIAA SCITECH 2022 Forum

AIAA SciTech Forum
January 3-7, 2022, San Diego, CA & Virtual
AIAA SCITECH 2022 Forum

10.2514/6.2022-1776

Many-revolution Earth-centred solar-sail trajectory optimisation using differential dynamic programming

G. Leemans*, L. Carzana† and J. Heiligers ‡
Delft University of Technology, 2600 AA, Delft, The Netherlands

This work demonstrates the usability of differential dynamic programming (DDP) to obtain optimal Earth-centred solar-sail trajectories. The dynamical model is implemented as a two-body problem, augmented with an ideal solar-sail reflectance model and accounts for eclipses. The numerical performance of the optimisation algorithm is enhanced by integrating the sailcraft state in modified equinoctial elements and performing a Sundman transformation to change the independent variable from time to the true anomaly. The DDP algorithm is proven to be robust for trajectories extending up to 500 revolutions and, compared to known locally optimal steering laws, allows to obtain equally optimal solutions. The latter is demonstrated in this paper through a set of test cases that range from theoretical scenarios to realistic mission applications, including increasing the specific orbital energy of NASA's upcoming ACS3 mission. Additionally, the algorithm's ability to cope with different optimisation settings, perturbing accelerations and constraints is demonstrated.

Nomenclature

α	=	Sail cone angle [rad]	e	=	Eccentricity
β	=	Solar-sail lightness number	F	=	Dynamics matrix
γ	=	Smoothed sunlight fraction	$f, g, h, \tilde{k}$	=	Modified equinoctial elements
Δ	=	Trust region radius	G	=	Scaling matrix
δ	=	Sail clock angle [rad]	H	=	Heaviside function
ϵ_1	=	Reduction ratio tolerance	$\tilde{h}$	=	Logistic growth rate
ϵ_2	=	Optimality tolerance	k	=	Stage number
η	=	Sundman transformation variable	i	=	Inclination [rad]
κ	=	Reduction ratio tolerance	J	=	Cost function
Λ	=	Sundman-transformed dynamics matrix	l	=	True longitude [rad]
λ	=	Lagrange multiplier	M	=	Scaling matrix
μ	=	Gravitational parameter [km^3/s^2]	m	=	Mass [kg]
ν	=	True anomaly [rad]	N	=	Number of stages
ρ	=	Reduction ratio	$\hat{\boldsymbol{n}}$	=	Unit normal vector
Σ	=	Penalty matrix	p	=	Semi-latus rectum [km]
Φ	=	State transition matrix	$\boldsymbol{p}$	=	Reference vector
ϕ	=	Objective function	q	=	Auxiliary modified equinoctial element
$\boldsymbol{\psi}$	=	Terminal constraints	R	=	Radius [km]
Ω	=	Right ascension of the ascending node [rad]	$\boldsymbol{r}$	=	Position vector [km]
ω	=	Argument of periapsis [rad]	$\boldsymbol{r_s}$	=	Sun-sail vector [km]
A	=	Feed-forward control matrix	T_{es}	=	Sidereal day [s]
a	=	Semi-major axis [km]	t	=	Time [s]
B	=	Feedback control matrix	$\boldsymbol{u}$	=	Control vector
C	=	Sail-centred reference frame	u	=	Control parameter
D	=	Feedback control matrix	$\boldsymbol{X}$	=	Augmented state vector

*Graduate student, Faculty of Aerospace Engineering, g.leemans@student.tudelft.nl
†PhD candidate, Faculty of Aerospace Engineering, l.carzana@tudelft.nl
‡Assistant Professor, Faculty of Aerospace Engineering, m.j.heiligers@tudelft.nl

I. Introduction

Solar sailing is a form of low-thrust space propulsion that uses a large, thin, mirror-like structure to produce thrust by reflecting solar photons. The thrusting capabilities of a solar sail are therefore only limited by the lifetime of the sail itself; there is no dependency on available propellant. Hence, solar sailing is particularly interesting for missions of long duration or missions requiring continuous thrust. Examples include the use of a solar sail to mitigate space debris in the geostationary ring [1, 2], Earth-escape trajectories [3–5], and non-Keplerian orbits [6, 7]. The concept of solar sailing has been successfully demonstrated by missions like IKAROS (JAXA, 2010) [8] and various solar-sail missions are scheduled for the near-future, including NASA's Advanced Composite Solar Sail System (ACS3) mission, which is scheduled for launch in 2022 [9].

Since the acceleration that the solar sail imposes on the spacecraft is very small, solar-sail trajectories are often optimised for the time of flight. Global time-optimal solutions have been found for *interplanetary* solar-sail trajectories with the use of optimisation algorithms such as evolutionary neurocontrol [10]. However, global optima of *planet-centred* trajectories have not yet been found. Due to the small solar-sail acceleration, many revolutions around the central body are often necessary before the desired state is reached. This makes solar-sail planet-centred trajectory optimisation more complex, as it leads to a sensitive and high-dimensional problem [11]. Optimisation of these trajectories has historically been performed using (semi-)analytical techniques, resulting in locally optimal solutions. For example, control laws have been developed that maximise the rate of change of a single orbital element, such as the semi-major axis or the inclination [12]. However, when more than one orbital element is to be changed, the problem becomes more complicated. Solutions have been proposed where the trajectory is split into different arcs, where in each arc the rate of change of a different orbital element is maximised [13]. Other solutions implement a blended control law, where individual control laws are combined with weights, where the magnitude of the weights is based on the relative importance of changing each orbital element during that particular phase of the trajectory [14].

An optimisation method that has never been explored for planet-centred solar-sail trajectory optimisation is differential dynamic programming (DDP). DDP is a second-order gradient-based optimisation routine that has the benefit that it scales linearly with the number of control variables [15], which is a property that makes DDP interesting for large optimisation problems. DDP discretises the spacecraft's trajectory into stages, where at each stage a local, quadratic model of the cost is minimised [16]. DDP has already been successfully applied to finding optimal Earth-centred trajectories using solar electric propulsion (SEP) [11], where optimal solutions were found for a transfer from geostationary transer orbit to geostationary equatorial orbit (GEO) consisting of up to 2000 revolutions.

This work aims to obtain locally optimal Earth-centred solar-sail trajectories through the use of DDP. In addition, the capability of monotonic basin hopping (MBH) to find globally optimal solutions, to improve on the locally optimal solution found by DDP, will be investigated. MBH tries to find the global optimum by iteratively executing the DDP algorithm with a slightly perturbed initial guess [17]. As such, this work will extend the applicability of DDP to many-revolution Earth-centred solar-sail trajectories and proof its performance in terms of optimal solution found compared to known locally optimal solutions. For a variety of test cases, the optimal solar-sail attitude will be found to maximise the specific orbital energy. Test cases range from theoretical scenarios to realistic mission applications, including NASA's upcoming ACS3 mission.

II. Dynamical model

The dynamics of the satellite are defined in an Earth-centred equatorial inertial reference frame, ECI(x,y,z) with the x-axis towards the vernal equinox, the z-axis perpendicular to the equatorial plane and the y-axis completing the right-handed reference frame. DDP requires the spacecraft's state to be represented as an augmented state vector [18], where the spacecraft state vector is augmented with the control vector as

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x} & \boldsymbol{u} \end{bmatrix}^T = \begin{bmatrix} \boldsymbol{r} & \dot{\boldsymbol{r}} & \boldsymbol{u} \end{bmatrix}^T = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & t & \alpha & \delta \end{bmatrix}^T \tag{1}$$

In Eq. 1 $\boldsymbol{r}$ is the position vector of the satellite, $\boldsymbol{u}$ is the control vector consisting of the two sail control parameters, α and δ, which will be elaborated upon in Section II.B, $\dot{\square}$ represents the derivative with respect to time and t is the current time. The derivative of the augmented state vector is composed of various terms:

$$\dot{\boldsymbol{X}} = \begin{bmatrix} \dot{\boldsymbol{r}} & \ddot{\boldsymbol{r}}_{\oplus} + \ddot{\boldsymbol{r}}_S + \ddot{\boldsymbol{r}}_{J_2} + \ddot{\boldsymbol{r}}_{J_3} + \ddot{\boldsymbol{r}}_{J_4} + \ddot{\boldsymbol{r}}_{\leftmoon} + \ddot{\boldsymbol{r}}_{\odot} & 1 & \dot{\alpha} & \dot{\delta} \end{bmatrix}^T \tag{2}$$

The terms for the acceleration in the second column of the vector in Eq. 2 are, from left to right, point-mass gravity from the Earth, the solar-sail acceleration, accelerations resulting from irregularities in the Earth's gravity field (J_2, J_3 and J_4), lunar gravity and solar gravity. These individual contributions will be expanded on in the coming sections.

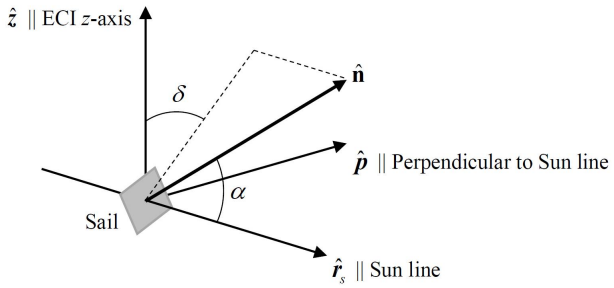


Fig. 1 Solar-sail attitude angles in frame C.

Unless stated otherwise, J2000 is used as the initial epoch for propagation. An important perturbing acceleration missing in Eq. 2 is the atmospheric drag. DDP optimisation requires the dynamical model to be twice differentiable, which becomes difficult when atmospheric drag is included. Density models most often consist of tabulated data, which are not differentiable [19]. The implementation of an accurate and twice differentiable drag model is therefore left for future research.

A. Two-body dynamics

First and foremost, the largest acceleration acting on the satellite is the point-mass gravity from the Earth [19]. This acceleration can be written as

$$\ddot{\boldsymbol{r}}_{\oplus} = -\frac{\mu_{\oplus}}{r^3}\boldsymbol{r} \tag{3}$$

where $\mu_{\oplus}$ is the Earth's gravitational parameter and $r = \|\boldsymbol{r}\|$. Table 1 provides the values of the dynamical parameters used.

B. Solar-sail acceleration

The solar-sail acceleration is modelled by assuming an ideal solar-sail reflectance model, which implies that all incoming radiation is specularly reflected and the acceleration acts along the vector normal to the sail surface, $\hat{\boldsymbol{n}}$. Other solar-sail force models account for absorption, diffuse reflection and thermal emission of the incoming radiation (optical force model), sail billowing and degradation [12, 20, 21]. The difference between using the ideal and optical force model has been demonstrated in Reference [20] for an Earth-Mars transfer. The results show an increase of $12 - 13\%$ in time of flight when using the optical model compared to the ideal model. The optical force model is not considered in this work, but similar orders of magnitude for the increase in time of flight may be expected when doing so in future work.

The direction of the sail normal vector, $\hat{\boldsymbol{n}}$, is defined in frame $C(\hat{\boldsymbol{r}}_s, \hat{\boldsymbol{p}}, \hat{\boldsymbol{z}})$, see Figure 1. Reference frame C has its origin at the sailcraft, the $\hat{\boldsymbol{r}}_s$ axis points in the anti-Sun direction, the $\hat{\boldsymbol{z}}$ axis in the direction of the ECI z-axis, and the third axis, $\hat{\boldsymbol{p}}$, completes the right-handed orthogonal reference frame. The sail normal vector is defined by two angles, see again Figure 1: the cone and clock angles, where the cone angle, α, is defined as the angle between the sail normal and the direction of sunlight. The clock angle, δ, is defined as the angle between the z-axis of the ECI reference frame and the projection of the normal vector onto the plane perpendicular to the Sun-line.

The normal vector of the sail, $\hat{\boldsymbol{n}}$, is defined in frame C as

$$\begin{aligned} \boldsymbol{n}|_C &= \begin{bmatrix} \cos\alpha & \sin\alpha\cos\delta & \sin\alpha\sin\delta \end{bmatrix}^T \\ \text{s.t.} \quad \alpha &\in [-\pi/2,\ \pi/2] \\ \delta &\in [-2\pi,\ 2\pi] \end{aligned} \tag{4}$$

An intrinsic property of the solar sail is that the solar-sail acceleration has no component in the direction of the Sun. To model this property, the cone angle is restricted to a range between $-\pi/2$ and $\pi/2$. The clock angle can in principle obtain any value, but is for practical reasons restricted between -2π and 2π. For inclusion in Eq. 2, the solar-sail acceleration is to be found in the ECI reference frame, and therefore the normal vector needs to be transformed from reference frame C to the ECI frame according to:

$$\boldsymbol{n} = \begin{bmatrix} \hat{\boldsymbol{r}}_s & \hat{\boldsymbol{z}} & \hat{\boldsymbol{z}} \times \hat{\boldsymbol{r}}_s \end{bmatrix} \boldsymbol{n}|_C \tag{5}$$

The resulting solar-sail acceleration is determined as [12]

$$\ddot{\boldsymbol{r}}_S = \gamma\beta\frac{\mu_\odot}{r_s^2}(\hat{\boldsymbol{r}}_s \cdot \boldsymbol{n})^2\boldsymbol{n} \tag{6}$$

where γ is the smoothed sunlight fraction, β is the solar-sail lightness number, $\mu_\odot$ the Sun's gravitational parameter and $\boldsymbol{r}_s$ is the position vector from the Sun towards the sailcraft. The solar-sail lightness number is defined as the ratio of the solar radiation pressure to the solar gravitational acceleration and is used as a metric for the thrusting capabilities of the sail. During eclipse conditions, the sail does not produce any acceleration. Eclipses are modelled using a low-precision solar ephemeris model [19] and a Heaviside-based sunlight fraction approach [22]. The sunlight fraction is equal to zero when the satellite is in eclipse, and equal to one when the satellite is in sunlight. Only eclipses with the Earth as the occulting body are taken into account.

C. Non-spherical Earth

Zonal harmonics coefficients, J_i with $i = 2, 3, ...$, are used to model the irregularities of the Earth's gravity field. The perturbing acceleration due to the J_2, J_3 and J_4 coefficients are accounted for and are equal to [23]

$$\ddot{\boldsymbol{r}}_{J_2} = -\frac{3\mu_\oplus J_2 R_\oplus^2}{2r^5}\left[x\left(1-5\tfrac{z^2}{r^2}\right) \quad y\left(1-5\tfrac{z^2}{r^2}\right) \quad z\left(3-5\tfrac{z^2}{r^2}\right)\right]^T \tag{7a}$$

$$\ddot{\boldsymbol{r}}_{J_3} = -\frac{15\mu_\oplus J_3 R_\oplus^3}{2r^7}\left[xz\left(1-\tfrac{3z^2}{r^2}\right) \quad yz\left(1-\tfrac{3z^2}{r^2}\right) \quad z^2\left(2-\tfrac{r^2}{5z^2}-\tfrac{3z^2}{r^2}\right)\right]^T \tag{7b}$$

$$\ddot{\boldsymbol{r}}_{J_4} = -\frac{35\mu_\oplus J_4 R_\oplus^4}{2r^9}\left[xz^2\left(\tfrac{3}{2}-\tfrac{9z^2}{4r^2}-\tfrac{3r^2}{28z^2}\right) \quad yz^2\left(\tfrac{3}{2}-\tfrac{9z^2}{4r^2}-\tfrac{3r^2}{28z^2}\right) \quad z^3\left(\tfrac{5}{2}-\tfrac{9}{4}\tfrac{z^2}{r^2}-\tfrac{15}{28}\tfrac{r^2}{z^2}\right)\right]^T \tag{7c}$$

where $R_\oplus$ is the Earth's radius. Values for the zonal harmonics coefficients are given in Table 1.

D. Third-body Perturbations

The accelerations resulting from the point-mass gravity from the Moon and the Sun are the largest third-body accelerations acting on an Earth-orbiting spacecraft and are accounted for. The position vector from the third-body to the satellite is written as [19]

$$\boldsymbol{r}_{\leftmoon/SC} = \boldsymbol{r} - \boldsymbol{r}_{TB} \tag{8}$$

where $\boldsymbol{r}_{TB}$ is the position vector of the third body. The contribution of the acceleration caused by the third body on the spacecraft can then be defined as

$$\ddot{\boldsymbol{r}}_{TB} = -\mu_{TB}\left[\frac{x-x_{TB}}{\|\boldsymbol{r}-\boldsymbol{r}_{TB}\|^3}+\frac{x_{TB}}{x_{TB}^3} \quad \frac{y-y_{TB}}{\|\boldsymbol{r}-\boldsymbol{r}_{TB}\|^3}+\frac{y_{TB}}{y_{TB}^3} \quad \frac{z-z_{TB}}{\|\boldsymbol{r}-\boldsymbol{r}_{TB}\|^3}+\frac{z_{TB}}{z_{TB}^3}\right]^T \tag{9}$$

where μ_{TB} is the gravitational parameter of the third body (see Table 1 for the values for the Moon and Sun).

To determine these perturbing accelerations, the position of the Moon and Sun needs to be known in the ECI frame at each epoch. To obtain these ephemeris data, analytical expressions for low-precision solar and lunar coordinates are used (see Reference [19], Section 3.3.1). The model to obtain these coordinates assumes unperturbed motion of the Earth around the Sun, whereas for the motion of the Moon around the Earth the influence of terrestrial and solar gravity are taken into account.

Table 1 Dynamical model parameters [23, 24].

Parameter	Value	Parameter	Value
$\mu_\oplus$	3.986004×10^{14} m^3/s^2	J_2	1.082626×10^{-3}
$\mu_\odot$	1.327124×10^{20} m^3/s^2	J_3	-2.533×10^{-6}
$\mu_\leftmoon$	4.902801×10^{12} m^3/s^2	J_4	-1.6186×10^{-6}
$R_\oplus$	63781363 m		

E. Modified equinoctial elements

The equations of motion (EOMs) are integrated in modified equinoctial elements (MEE). Integrating a set of orbit elements has benefits to integrating a Cartesian state, since the number of rapidly changing elements can be reduced. A formulation of the EOMs that only has one single rapidly changing element, like the Keplerian model, but does not experience the singularities that are present in the Keplerian model, are the MEE [25]. Five variables describe the shape and orientation of the orbit, while one variable describes the location of the spacecraft within the orbit. This latter variable is chosen as the true longitude. The MEE are defined as

$$p = a(1-e^2),\ f = e\cos(\omega+\Omega),\ g = e\sin(\omega+\Omega),\ h = \tan\left(\frac{i}{2}\right)\cos(\Omega),\ \tilde{k} = \tan\left(\frac{i}{2}\right)\sin(\Omega),\ l = \omega+\Omega+\nu \tag{10}$$

where p is the semi-latus rectum, f and g describe the orbit eccentricity, whereas h and $\tilde{k}$ describe the orbit inclination and l is the true longitude. The MEE are defined using Kepler elements, a, i, e, ω, Ω and ν, which are the semi-major axis, inclination, eccentricity, argument of periapsis, right ascension of the ascending node and the true anomaly, respectively. The augmented state vector in MEE form becomes

$$\boldsymbol{X}_{\text{MEE}} = \begin{bmatrix} p & f & g & h & \tilde{k} & l & t & \alpha & \delta \end{bmatrix}^T \tag{11}$$

Integrating the dynamics in MEE requires the derivative of the augmented state vector first to be evaluated in Cartesian elements (see Eq. 2), after which the derivative can be transformed to MEE by the method in Reference [26] (Section 2.1).

F. Sundman transformation

Using MEE has proven to be even more advantageous when using a Sundman transformation to change the independent variable for integration from time to the true anomaly, ν [11]. This Sundman transformation is given as

$$dt = \frac{1}{\sqrt{\mu_\oplus p}}\left(\frac{p}{q}\right)^2 d\nu \tag{12}$$

where q is an auxiliary MEE, which is defined as

$$q = 1 + f\cos(l) + g\sin(l) \tag{13}$$

The derivative of the augmented state vector is then transformed as

$$\mathring{\boldsymbol{X}}_{\text{MEE}} = \frac{\partial \boldsymbol{X}_{\text{MEE}}}{\partial \nu} = \frac{\partial \boldsymbol{X}_{\text{MEE}}}{\partial t}\frac{\partial t}{\partial \nu} = \dot{\boldsymbol{X}}_{\text{MEE}}\frac{\partial t}{\partial \nu} \tag{14}$$

where $\partial t/\partial \nu$ is obtained from Eq. 12. Note the different dot above the augmented state vector ($\mathring{\boldsymbol{X}}$ versus $\dot{\boldsymbol{X}}$), where $\mathring{\square}$ indicates the Sundman-transformed derivative.

III. Differential dynamic programming

This section provides the definition of the DDP method and details on its implementation. The algorithm provided here is based on the DDP algorithm in Reference [18], for the particular case of a single-phase trajectory without local stage costs.

Like any low-thrust trajectory optimisation algorithm, DDP seeks the minimum of a cost function $J(\boldsymbol{x}, \boldsymbol{u}, t)$, where $\boldsymbol{x}(t)$ is a state trajectory and $\boldsymbol{u}(t)$ is a control schedule. The general workflow of DDP is explained as follows. First, the spacecraft's trajectory is forward integrated with a given set of controls. Then, the trajectory is discretised into stages, where at each stage a local, quadratic model of the cost is minimised. The method of Dynamic Programming is based on Bellman's principle of optimality: "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision." [27]. Following this principle, the algorithm finds the control variables for the final stage that correspond to the best final solution. When these control variables are found, the algorithm repeats this task for the penultimate stage, until it has optimised all individual stages. This process is called the backward sweep. With the newly found control variables, the problem is forward integrated, after which the same procedure is initiated until the optimisation algorithm has converged. The remainder of this section will elaborate on the different steps taken by DDP, but first the notation is elaborated upon in the next section.

A. Notation

The DDP algorithm consists of a number of equations in which scalars, vectors, matrices and third-orders tensors make their appearance. This section covers the notation that is used throughout the upcoming sections. Similar as in Section II, boldface is used to indicate vectors, whereas superscripts are used to indicate the entries in a vector, matrix or tensor. For example, consider the augmented state vector in MEE (note that the MEE subscript is dropped for brevity) as

$$\boldsymbol{X} = \begin{bmatrix} p & f & g & h & \tilde{k} & l & t & \alpha & \delta \end{bmatrix}^T \tag{15}$$

The i^{th} entry of $\boldsymbol{X}$ is written as $\boldsymbol{X}^i$, so $\boldsymbol{X}^3$ is equal to g when counting starts from one. The same procedure holds for matrices and third-order tensors. For example, consider the following matrix:

$$Y = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \tag{16}$$

The $(i, a)^{\text{th}}$ entry of Y is written as $Y^{i,a}$, so $Y^{2,3}$ is equal to 6.

Subscripts are used for indexing and for indicating the derivative, for example:

$$J_X = \frac{\partial J}{\partial \boldsymbol{X}} \tag{17}$$

where J_X is the derivative of the cost, J, with respect to the augmented state vector, $\boldsymbol{X}$. When the augmented state vector, $\boldsymbol{X}$, has size $n \times 1$, then J_X is also of size $n \times 1$. Note that in this case no boldface is used for the subscript. An additional subscript, $J_{X,k}$, is used to indicate the value of the gradient evaluated at stage k. Second-order derivatives are defined similarly as

$$J_{X\lambda} = \frac{\partial^2 J}{\partial \boldsymbol{X} \partial \boldsymbol{\lambda}} \tag{18}$$

where $J_{X\lambda}$ is the second-order derivative of the cost, J, with respect to the augmented state vector, $\boldsymbol{X}$, and the vector of Lagrange multipliers, $\boldsymbol{\lambda}$. When the augmented state vector, $\boldsymbol{X}$, has size $n \times 1$ and the vector containing the Lagrange multipliers, $\boldsymbol{\lambda}$, has size $m \times 1$, then $J_{X\lambda}$ is of size $n \times m$.

Einstein summation is used to indicate summation over repeated indices. As an example, consider the simple dot product:

$$\boldsymbol{Y} \cdot \boldsymbol{Y}^T = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T = 1 \times 1 + 2 \times 2 + 3 \times 3 = 14 \tag{19}$$

Instead of writing the dot product, Einstein notation can be used. The product is then written as

$$\boldsymbol{Y}^{\gamma_1} \boldsymbol{Y}^{\gamma_1} = \sum_{\gamma_1=1}^{3} \boldsymbol{Y}^{\gamma_1} \times \boldsymbol{Y}^{\gamma_1} = 1 \times 1 + 2 \times 2 + 3 \times 3 = 14 \tag{20}$$

where γ_1 is a dummy index indicating that summation should take place over repeating values of the dummy index. Matrix multiplication can also be written using Einstein notation as

$$\dot{\Phi}^{i,a} = F^{i,\gamma_1} \Phi^{\gamma_1,a} = \sum_{\gamma_1=1}^{m} F^{i,\gamma_1} \Phi^{\gamma_1,a} \tag{21}$$

where $\dot{\Phi}$ is the derivative of the state transition matrix, F is the dynamics matrix (see Section III.D), Φ is the state transition matrix and m is the size of the matrix. The dummy index, γ_1, indicates that the $(i, a)^{\text{th}}$ entry of $\dot{\Phi}$ is found by summing the products of the γ_1^{th} entry of the i^{th} row of F with the γ_1^{th} entry of the a^{th} column of Φ, from $\gamma_1 = 1$ to m. Multiple dummy indices, for example γ_1 and γ_2, may be used to indicate a double summation.

B. Forward sweep

The optimisation procedure starts with the forward sweep, which integrates the trajectory with a given set of controls. For the first iteration, this is the initial guess, while subsequent iterations update the control according to $\boldsymbol{u} = \bar{\boldsymbol{u}} + \delta \boldsymbol{u}$, where $\bar{\boldsymbol{u}}$ is the control of the previous successful iteration (see Section III.G for how a successful iteration is defined) and $\delta \boldsymbol{u}$ is the control update calculated in the backward sweep of the algorithm (see Section III.D). If the resulting trajectory meets the optimality criteria, the procedure is finished and the algorithm is said to have converged. Otherwise, a new control update is calculated in the backward sweep.

The dynamics are integrated using a fixed timestep DOPRI8 integrator, where 100 steps are taken per revolution around the Earth regardless of the shape of the trajectory. State transition matrices (STMs) have to be integrated along the state, which is best carried out using an integrator with a fixed step size [28]. When a variable step size integrator

would be used, the state transition matrices will not account for part of the changes in the trajectory that are induced by the different time steps taken.

C. Augmented Lagrangian cost function

The cost function, which is to be minimised, is written as

$$J = \phi + \boldsymbol{\lambda}^T \boldsymbol{\psi} + \boldsymbol{\psi}^T \boldsymbol{\Sigma} \boldsymbol{\psi} \tag{22}$$

where ϕ is the objective that is to be minimised, $\boldsymbol{\lambda}$ the vector of Lagrange multipliers, $\boldsymbol{\psi}$ the vector with terminal constraints and $\boldsymbol{\Sigma}$ the penalty matrix to place additional weight on each constraint. The Lagrange multipliers are initialised at zero and updated with every iteration of the algorithm, see Section III.E.

D. Backward sweep

The backward sweep is the step of the algorithm where the control update is determined. This step solves the sequence of subproblems that minimises the cost-to-go from stage $k = N-1, N-2, ..., 0$. The cost-to-go is defined as the cost incurred from the current stage to the final destination [18]. The optimal cost-to-go at each stage is written as

$$J_k^* = \min_{\delta \boldsymbol{u}_k} [J_k] \tag{23}$$

where J_k is the cost-to-go from stage k until the end of the trajectory. The superscript asterisk denotes the optimal value, so J_k^* is the optimal cost-to-go from stage k until the end of the trajectory. The solution to Eq. 23 is the optimal control update $\delta \boldsymbol{u}_k^*$. The cost function is approximated with a second-order Taylor series expansion around the states, controls and Lagrange multipliers. By taking the derivative of this expansion with respect to the controls and setting it to zero, an unconstrained feedback control law for the optimal control update is found as

$$\begin{cases} \delta \boldsymbol{u}_k^* = A_k + B_k \delta \boldsymbol{x}_k + D_k \delta \boldsymbol{\lambda} \\ A_k = -J_{uu,k}^{-1} J_{u,k} \\ B_k = -J_{uu,k}^{-1} J_{ux,k} \\ D_k = -J_{uu,k}^{-1} J_{u\lambda,k} \end{cases} \tag{24}$$

where $\delta \boldsymbol{\lambda}$ is the Lagrange multiplier update, see Section III.E, A is the feedforward control matrix, and B and D are the feedback control matrices. It is important to note that the subscript x is used, not X. The derivatives with respect to the state, $\boldsymbol{x}$, and the controls, $\boldsymbol{u}$, are a part of the derivative with respect to the complete augmented state vector, $\boldsymbol{X}$, as

$$J_X = \begin{bmatrix} J_x & J_u \end{bmatrix}^T = \begin{bmatrix} \frac{\partial J}{\partial p} & \frac{\partial J}{\partial f} & \frac{\partial J}{\partial g} & \frac{\partial J}{\partial h} & \frac{\partial J}{\partial \tilde{k}} & \frac{\partial J}{\partial l} & \frac{\partial J}{\partial t} & | & \frac{\partial J}{\partial \alpha} & \frac{\partial J}{\partial \delta} \end{bmatrix}^T \tag{25}$$

Note that J_{uu} and J_{ux} are submatrices of J_{XX}, following the same analogy as for J_X in Eq. 25. The submatrices for Eq. 24 are calculated and stored for use during the forward sweep of the algorithm. Note that $\delta \boldsymbol{u}_k^*$ cannot be determined directly, since it is dependent on $\delta \boldsymbol{x}_k$. This variable is the change in state due to the changed control, which is calculated during the forward integration of the state. A trust-region method limits the size of the hessian, $J_{uu,k}$, such that the resulting change in state, $\delta \boldsymbol{x}_k$, remains within the valid region of the quadratic model, see Section III.F. The remainder of the backward sweep is focused on finding the required matrices for Eq. 24. These matrices can be found using the cost-to-go derivatives with respect to the augmented state, and Lagrange multipliers as

$$J_{X,k} = \Phi^T J_{X,k+1}^* \tag{26a}$$

$$J_{\lambda,k} = J_{\lambda,k+1}^* \tag{26b}$$

$$J_{XX,k}^{i,a} = J_{XX,k+1}^{*\gamma_1,\gamma_2} \Phi^{\gamma_1,i} \Phi^{\gamma_2,a} + J_{X,k+1}^{*,\gamma_1} \Phi^{\gamma_1,ia} \tag{26c}$$

$$J_{\lambda\lambda,k} = J_{\lambda\lambda,k+1}^* \tag{26d}$$

$$J_{X\lambda,k} = \Phi^T J_{X\lambda,k+1}^* \tag{26e}$$

where $\Phi^{i,a}$ is the state transition matrix and $\Phi^{i,ab}$ is the second-order state transition tensor, both will be referred to as the STMs. The derivatives of the cost-to-go at stage k in Eq. 26 are dependent on the derivatives of the optimal cost-to-go at stage $k+1$. Details on how the stage's optimal cost-to-go derivatives are calculated are provided at the end of this section. Some additional information per subequation is provided below.

- For the derivatives with respect to the Lagrange multipliers in Eqs. 26b and 26d, the derivatives at stage k are equal to the optimal derivatives at stage $k+1$. In addition, the observant reader might notice that $J_{\lambda,k}$ and $J_{\lambda\lambda,k}$ in Eqs. 26b and 26d are not directly needed in Eq. 24. However, these two derivatives are required to calculate the update of the Lagrange multipliers, $\delta\boldsymbol{\lambda}$, see Section III.E.
- For the derivatives where the state, $\boldsymbol{x}$, is involved (Eqs. 26a, 26c and 26e), the optimal cost-to-go derivatives from stage $k+1$ are mapped to stage k with the use of the STMs.
- In Eq. 26c Einstein notation (as introduced in Section III.A) is used to indicate summation over two dummy indices, γ_1 and γ_2, which is used to find the hessian, $J_{XX,k}$. For clarification, this equation can alternatively be written as

$$J^{i,a}_{XX,k} = \sum_{\gamma_1=1}^{m}\sum_{\gamma_2=1}^{m} J^{*\gamma_1,\gamma_2}_{XX,k+1}\Phi^{\gamma_1,i}\Phi^{\gamma_2,a} + J^{*,\gamma_1}_{X,k+1}\Phi^{\gamma_1,ia} \tag{27}$$

  where m is the size of $\boldsymbol{X}$.

Evaluating the STMs is the most computationally expensive part of the algorithm, and is performed in parallel using OpenMP [29]. Parallel computation is possible since the state at each point in time is already known from the forward sweep. In order to determine the STMs, the dynamics matrix and tensor are defined as

$$F^{i,a} = \frac{\partial \dot{\boldsymbol{X}}^i}{\partial \boldsymbol{X}^a} \tag{28a}$$

$$F^{i,ab} = \frac{\partial^2 \dot{\boldsymbol{X}}^i}{\partial \boldsymbol{X}^a \partial \boldsymbol{X}^b} \tag{28b}$$

where the $(i,a)^{\text{th}}$ entry of the dynamics tensor, $F^{i,a}$, corresponds to the derivative of the i^{th} entry of $\dot{\boldsymbol{X}}$ with respect to the a^{th} entry of $\boldsymbol{X}$. For the third-order dynamics tensor, $F^{i,ab}$, the $(i,ab)^{\text{th}}$ entry of the tensor corresponds to the second-order derivative of the i^{th} entry of $\dot{\boldsymbol{X}}$ with respect to the a^{th} and b^{th} entry of $\boldsymbol{X}$. The entries of the dynamics matrix and tensor are found analytically*.

The previously proposed Sundman transformation requires the dynamics matrix and tensor to be transformed. As stated in Section II.F and particularly in Eq. 14, the Sundman transformation is performed by multiplying the derivative of the augmented state vector with $\eta = dt/d\nu$.

The first and second derivatives of η with respect to the augmented state vector are written as

$$\eta^i_X = \frac{\partial \eta}{\partial \boldsymbol{X}^i} \tag{29a}$$

$$\eta^{i,a}_{XX} = \frac{\partial^2 \eta}{\partial \boldsymbol{X}^i \partial \boldsymbol{X}^a} \tag{29b}$$

The dynamics matrix and tensor can then be transformed as

$$\Lambda^{i,a} = F^{i,a}\eta + \dot{X}^i\eta^a_X \tag{30a}$$

$$\Lambda^{i,ab} = F^{i,ab}\eta + F^{i,a}\eta^b_X + F^{i,b}\eta^a_X + \dot{X}^i\eta^{a,b}_{XX} \tag{30b}$$

where $\Lambda^{i,a}$ and $\Lambda^{i,ab}$ are the Sundman-transformed dynamics matrix and tensor, respectively. Similar to the computation of the derivative of the augmented state vector in Section II.F, the dynamics matrix and tensor are both first determined in Cartesian elements, after which they are transformed to MEE, see Reference [11] (Appendix A). Finally, the differential equations that need to be solved to find the STMs (again using Einstein notation) are:

$$\dot{\Phi}^{i,a} = \Lambda^{i,\gamma_1}\Phi^{\gamma_1,a} \tag{31a}$$

$$\dot{\Phi}^{i,ab} = \Lambda^{i,\gamma_1}\Phi^{\gamma_1,ab} + \Lambda^{i,\gamma_1\gamma_2}\Phi^{\gamma_1,a}\Phi^{\gamma_2,b} \tag{31b}$$

This set of differential equations can easily be solved by knowing the boundary conditions $\Phi(t_k,t_k)^{i,a} = \boldsymbol{I}$ and $\Phi(t_k,t_k)^{i,ab} = \boldsymbol{0}$, where $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{0}$ a matrix of zeroes. In other words, at the beginning of each stage the state transition matrix is equal to the identity matrix and the state transition tensor is equal to zero. The STMs are then evaluated at the end of each stage using a single step of the DOPRI8 integration routine.

* Maple worksheets available at: `https://github.com/gijsleemans1/Analytical_derivatives`. Accessed: 9/6/2021

Now that it is clear how to obtain the STMs in Eq. 26, the derivatives of the cost-to-go at stage k in Eq. 26 can be calculated, assuming that the optimal cost-to-go derivatives from stage $k+1$ are known. Afterwards, the optimal cost-to-go derivatives from the current stage, k, are calculated as

$$ER_k = ER_{k+1} + J_{u,k}^T A_k + \frac{1}{2} A_k^T J_{uu,k} A_k \tag{32a}$$

$$J_{x,k}^{*T} = J_{x,k}^T + J_{u,k}^T B_k + A_k^T J_{uu,k} B_k + A_k^T J_{ux,k} \tag{32b}$$

$$J_{\lambda,k}^{*T} = J_{\lambda,k}^T + J_{u,k}^T D_k + A_k^T J_{uu,k} D_k + A_k^T J_{u\lambda,k} \tag{32c}$$

$$J_{xx,k}^{*T} = J_{xx,k}^T + B_k J_{uu,k}^T B_k + B_k^T J_{ux,k} + J_{ux,k} B_k \tag{32d}$$

$$J_{\lambda\lambda,k}^{*T} = J_{\lambda\lambda,k}^T + D_k J_{uu,k}^T D_k + D_k^T J_{u\lambda,k} + J_{u\lambda,k} D_k \tag{32e}$$

$$J_{x\lambda,k}^{*T} = J_{x\lambda,k}^T + B_k J_{uu,k}^T D_k + B_k^T J_{u\lambda,k} + J_{ux,k} D_k \tag{32f}$$

where ER_k is the expected reduction in cost at stage k. The expected reduction is not needed to calculate the control update in Eq. 24, but is used to determine whether an iteration of the DDP algorithm has been successful or not, see Section III.G. At the final stage, the optimal and nominal costs are equivalent, i.e. $J_N^* = J$. This means that the derivatives of the optimal cost-to-go are equal to the derivatives of the augmented Lagrangian, which can be found analytically*. Note that there is no expected reduction at the final stage.

To summarise, the backward sweep starts by calculating the derivatives of the optimal cost-to-go at the final stage analytically and the evaluation of the STMs at all stages. Then, an iterative approach is started to calculate the current stage's cost-to-go derivatives (see Eq. 26), the feedback and feedforward matrices (see Eq. 24), and optimal cost-to-go derivatives (see Eq. 32). After all the aforementioned terms have been calculated for stage k, the algorithm does the same for stage $k-1$. This process is repeated until the algorithm has evaluated the feedback and feedforward matrices for the first stage.

E. Lagrange multiplier update

The Lagrange multipliers are updated once per iteration of the algorithm, after the backward sweep has been completed, as

$$\delta\boldsymbol{\lambda} = -J_{\lambda\lambda}^{-1} J_\lambda \tag{33}$$

where $J_{\lambda\lambda} = J_{\lambda\lambda,0}^*$ and $J_\lambda = J_{\lambda,0}^*$, indicating the derivatives are evaluated at stage $k = 0$. Also, the expected reduction has to be updated as

$$ER_0 = ER_0 + J_\lambda^T \delta\boldsymbol{\lambda} + \frac{1}{2} \delta\boldsymbol{\lambda}^T J_{\lambda\lambda} \delta\boldsymbol{\lambda} \tag{34}$$

where ER_0 is the expected reduction at stage $k = 0$.

F. Trust region quadratic subproblem

As was stated in Section III.D, the trust region quadratic subproblem (TRQP) restricts the size of the hessian, $J_{uu,k}$, so that the resulting change in state, $\delta\boldsymbol{x}_k$, remains within the valid region of the quadratic model. This sub-algorithm is employed at each stage as soon as $J_{uu,k}$ is calculated in Eq. 26c. The expected reduction (see Eq. 32a), A_k, B_k and D_k (see Eq. 24) are all updated if $J_{uu,k}$ is changed by the TRQP. The trust region subproblem is formulated as

$$\min_{\delta u_k} J_{u,k} \delta\boldsymbol{u}_k + \frac{1}{2} \delta\boldsymbol{u}_k^T J_{uu,k} \delta\boldsymbol{u}_k \quad \text{such that} \quad \|G\delta\boldsymbol{u}_k\| \leq \Delta \tag{35}$$

where G is a scaling matrix, which was set to the identity matrix throughout this work, $\|\square\|$ is the Euclidean norm and Δ is the trust region radius. Many different methods for solving the TRQP exist in literature. The methods of Reference [30] have been proven to be robust for solving the TRQP in DDP. In particular, algorithm 7.3.4. (with the enhancement of introducing a scaling matrix, M), has been used to solve the TRQP in DDP [11, 18]. This method is from hereon referred to as the "classical TRQP". According to Reference [18], a less rigorous and efficient approach is arbitrary hessian shifting, from hereon referred to as the "simple TRQP". In the simple TRQP, the same user-defined value is added to the diagonal of the hessian at every stage. If the resulting control update is outside the trust region, the value on the diagonal is changed such that the control update lies exactly on the boundary of the trust region. The TRQP

not only restricts J_{uu}, but also $J_{\lambda\lambda}$. However, there is one notable difference between the TRQP for the control update and the multiplier update: J_{uu} is required to be positive definite, while $J_{\lambda\lambda}$ is required to be negative definite. So, the control parameter subproblem is defined as TRQP(J_u,J_{uu},Δ) and the multiplier subproblem as TRQP($-J_\lambda$,$-J_{\lambda\lambda}$,Δ).

G. Iterations

After the backward sweep has been completed, it is determined whether the iteration should be accepted or rejected. Therefore, a reduction ratio is calculated as

$$\rho = \frac{\delta J}{ER_0} \tag{36}$$

where δJ is the difference in final cost between successive iterations and ER_0 is the expected reduction at the first stage. When $\rho \approx 1$, the model is valid because the reduction in cost is as predicted. The iteration is then accepted and for the next iteration the trust-region is enlarged. If ρ is not close to one, the iteration should be rejected and the trust-region should be reduced for the next iteration. The STMs will not have to be calculated again when the iteration is rejected, since the state will not have changed because the controls remain unchanged. The algorithm can immediately proceed from the backward sweep onwards. The size of the trust region for the subsequent iteration is calculated as

$$\Delta_{\#p+1} = \begin{cases} \min\left((1+\kappa)\Delta_{\#p}, \Delta_{max}\right), & \text{if } \rho \in (1-\epsilon_1, 1+\epsilon_1) \\ \max\left((1-\kappa)\Delta_{\#p}, \Delta_{min}\right), & \text{otherwise} \end{cases} \tag{37}$$

where $\#p$ is the iteration counter, κ determines by how much the trust region is enlarged or shrunk during each iteration, Δ_{max} and Δ_{min} are the maximum and minimum trust region radii, respectively, and ϵ_1 gives the tolerance on ρ which is deemed acceptable. The complete algorithm is said to have converged when the expected reduction of the latest iteration is smaller than the optimality tolerance, ϵ_2.

H. Monotonic basin hopping

Classical dynamic programming algorithms minimise the cost function by discretising the stages and controls, which has the consequence that the algorithm satisfies the conditions for global optimality [18]. The downsides of these classical algorithms is that they have large storage requirements due to the "curse of dimensionality". Differential dynamic programming overcomes this issue by introducing the quadratic trust region, forcing the solution to be inside a given corridor around a reference solution, thereby reducing the dimension of the search space. This has the consequence that DDP can only be considered to find locally optimal solutions. DDP can be paired with a global search method, such as monotonic basin hopping (MBH), to find globally optimal solutions [11].

MBH works as follows [17]: first, the inner optimisation algorithm, DDP, runs with an initial guess. The result of the optimisation is a set of control variables that results in a locally optimal trajectory. This set of control variables is randomly perturbed within the control bounds, and fed to the optimisation algorithm as a new initial guess. If the optimisation yields a trajectory with lower cost, the iteration is accepted, the perturbed control becomes the new control and the procedure is repeated. If the new trajectory has a higher cost, the iteration is rejected and the original controls are again randomly perturbed. This is repeated until a stopping criterion is met.

I. Control Bounds

Control parameters can be limited by control bounds $u^L \leq u \leq u^U$, where u^L is a lower bound and u^U is an upper bound of the control parameter u. The original DDP algorithm treats control bounds with a null-space method [31]. This work proposes a different method of incorporating control bounds, namely a Heaviside-based approach. The Heaviside function is approximated with the smooth logistic function as

$$H\{j\} \approx \frac{1}{2} + \frac{1}{2}\tanh\left(\tilde{h}j\right) \tag{38}$$

where $\tilde{h}$ is a tuning parameter that determines the steepness of the ascent and descent of the Heaviside function. Increasing $\tilde{h}$ too much will lead to very large derivatives at the bounds and the algorithm becoming unstable. Throughout this work, $\tilde{h}$ was set to 100. The new control parameter u_{new}, calculated from the original u with the combination of Heaviside functions is:

$$u_{new} = u + (u_L - u) \times H\{-\tilde{h}(u - u_L)\} + (u_U - u) \times H\{\tilde{h}(u - u_U)\} \tag{39}$$

When the control is within the control bounds, both Heaviside functions will equate to zero and $u_{new} = u$. If the control is smaller than the lower bound, $u < u^L$, the Heaviside term $H\{-\tilde{h}(u - u_L)\}$ becomes equal to one, which will add the difference between the lower bound and the current control value, $u_L - u$, to the current control, resulting in the control being exactly on the lower bound. The same reasoning holds for when the control is larger than the upper bound, $u > u^U$, the Heaviside term $H\{\tilde{h}(u - u_U)\}$ becomes equal to one, which will add the difference between the upper bound and the current control value, $u_U - u$, which is a negative value, to the current control. This results in the control being exactly on the upper bound.

IV. Results

This section presents the results through a set of five test cases to demonstrate the applicability of DDP to optimise Earth-centred solar-sail trajectories as well as demonstrating the impact on the performance of DDP depending on the complexity of the dynamics considered, the settings of the algorithm and the inclusion of constraints. The first test case in Section IV.A is included for validation purposes and to compare the performance of the DDP optimisation approach proposed in this paper to the locally optimal results known from the literature. Subsequently, in Section IV.B the case is extended to more complex dynamics and a thorough analysis of various settings used for the DDP algorithm is given. Section IV.C demonstrates the algorithm's performance on a real-life example in the form of NASA's upcoming ACS3 mission. Constraints are implemented that force the ACS3 sailcraft to keep its Sun-synchronous attribute during orbit raising. While all test cases in Sections IV.A-IV.C are optimised over the course of only three orbital revolutions, Sections IV.D and IV.E extend these optimisation to many revolutions around Earth. Note that for all test cases the specific orbital energy is optimised.

A. Two-dimensional geostationary orbit: three orbital revolutions

This first test case concerns the optimisation of a spacecraft in geostationary orbit (42, 241 km altitude, zero inclination and eccentricity) over the course of three orbital revolutions. Since no constraints are enforced, the augmented Lagrangian cost function can be written as

$$J = \phi = -\left(\frac{\dot{r}^2}{2} - \frac{\mu_\oplus}{r}\right) \tag{40}$$

Note the minus sign in Eq. 40, since the specific orbital energy is to be maximised, whereas the cost function is minimised by the DDP algorithm. A two-dimensional dynamical model is assumed where the direction of sunlight is assumed to coincide with the orbital plane and the Sun is located at a distance of one astronomical unit (AU), fixed along the negative x-axis of the ECI frame. The position vector of the Sun is thus equal to

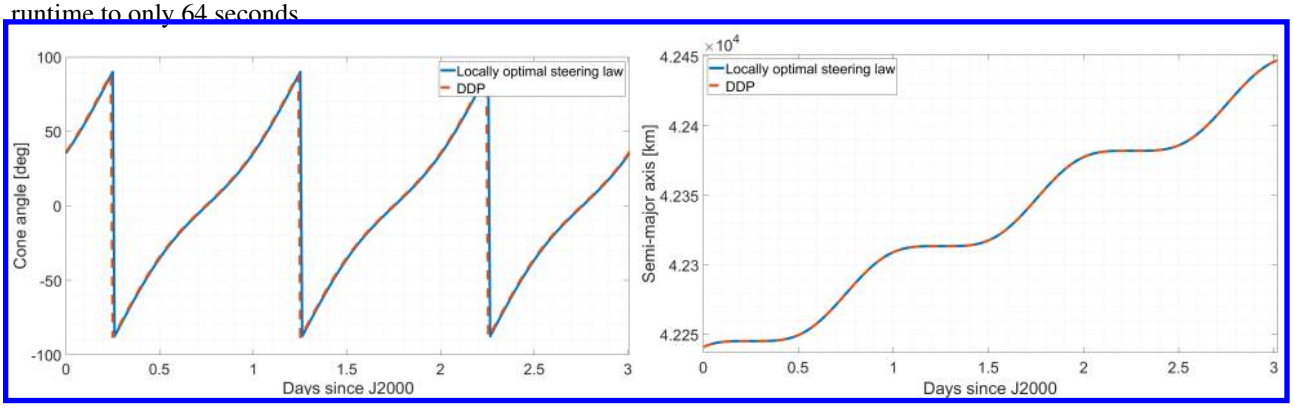
$$\boldsymbol{r}_\odot = \begin{bmatrix} -1AU & 0 & 0 \end{bmatrix} \tag{41}$$

The control is then solely determined by the sail cone angle, α, as the clock angle, δ, is fixed at 90°. For the cone angle, an initial guess of 0 is used across all stages. For now, only point-mass gravity from the Earth and the solar-sail acceleration itself are assumed to act on the sailcraft, i.e., the following terms in Eq. 2 equate to zero: $\ddot{\boldsymbol{r}}_{J_2} = \ddot{\boldsymbol{r}}_{J_3} = \ddot{\boldsymbol{r}}_{J_4} = \ddot{\boldsymbol{r}}_{\leftmoon} = \ddot{\boldsymbol{r}}_\odot = 0$. These assumptions and simplifications are made to allow validation of the DDP algorithm. The output of the DDP algorithm is compared to the outcome following from a locally optimal steering law from Reference [12]. A state-of-the-art solar-sail lightness number of $\beta = 0.011$ is used.

The resulting control profile and evolution of the semi-major axis are shown in Figure 2. The control profiles found by DDP and the locally optimal steering law are very similar, although small differences are visible. The final semi-major axis found by DDP is approximately 35 m smaller than the one found by the locally optimal steering law. Both solutions are local optima, but the solution found by the steering law is a better local optimum. The difference in semi-major axis could be explained by the fact that for this simple two-dimensional case the locally optimal steering law results in the theoretical maximum semi-major axis gain and the solution from the DDP algorithm only gets close to this result because of the approximation of the cost function by a second-order Taylor series expansion. With a stricter optimality tolerance (see Section IV.B.1 for the nominal DDP settings used throughout this paper), the DDP algorithm might be able to find an increase in the semi-major axis that converges towards the steering law solution. Since the DDP algorithm finds a very similar control profile compared to the locally optimal steering law, it can be concluded that the solution is a valid optimum. The DDP solution comes at the cost of a computational time of 853 seconds, compared to a runtime of approximately 0.2 seconds for the steering law solution[†]. When more complex and longer-duration solar-sail

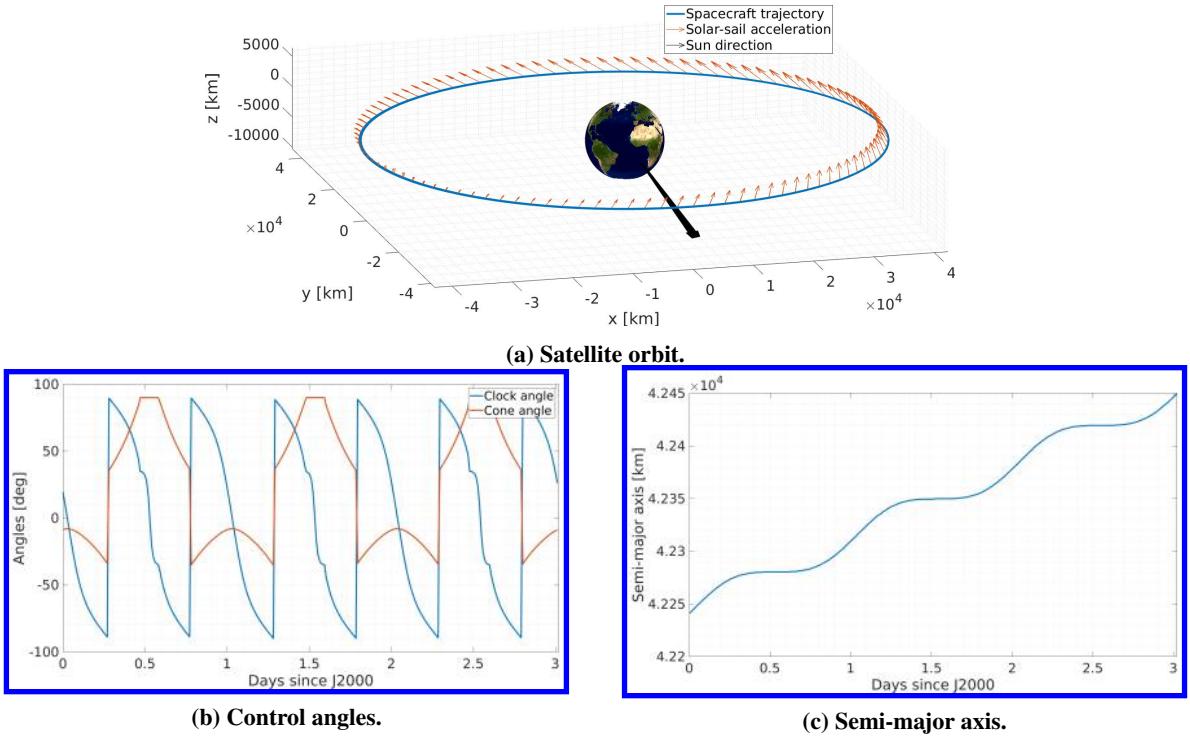
[†]Computation on personal laptop CPU: Quad-core Intel(R) Core(TM) i7-10610U @1.80 GHz

scenarios are considered, such as the ones in the next four subsections, the increase in computational time is expected to be justified. The improvement to the optimal solution is expected to be proportionally larger when more complex problems are evaluated. Furthermore, contrary to the locally optimal steering law approach, DDP allows to include terminal and path-inequality constraints in the optimisation problem. Finally, not that significant reductions in runtime can be achieved when using the locally optimal steering law as initial guess for the DDP algorithm, which reduces the runtime to only 64 seconds.



(a) Control angles. **(b) Semi-major axis.**

Fig. 2 Two-dimensional geostationary orbit: comparison with locally optimal steering law [12].



(a) Satellite orbit.

(b) Control angles. **(c) Semi-major axis.**

Fig. 3 Three-dimensional geostationary orbit: three orbital revolutions.

B. Three-dimensional geostationary orbit: three orbital revolutions

As a first extension, the problem of the previous section is expanded into three dimensions by removing the assumption on the direction of sunlight to be fixed and planar. Instead, the Sun's position is retrieved from ephemeris data, as explained in Section II.D. Also, a more realistic lightness number of 0.011 is assumed. The clock angle is no longer fixed to 90°, but is optimised by the DDP algorithm from an initial guess of zero at every stage. All other case details remain the same as those described in Section IV.A.

The resulting trajectory is shown in Figure 3a, where also the direction towards the Sun and the solar-sail acceleration are visualised. The optimal control profile, see Figure 3b, shows that both the cone and clock angles vary periodically

with every revolution of the satellite around the Earth. The resulting semi-major axis increase from Figure 3c behaves similar to the increase in the two-dimensional case, although there is a shift in phase, since the phasing of the Sun and spacecraft is different when using the true ephemeris of the Sun. Furthermore, the magnitude of the increase of the semi-major axis is different as a different solar-sail lightness number is used.

1. DDP algorithm settings

Various settings of the DDP algorithm can be tuned, as explained in Section III. The influence of changing these settings for the current problem is investigated to find out how robust the DDP algorithm is to different settings and what the best settings are for the remaining test cases. In this section, the effect of changing the reduction ratio tolerance, ϵ_1, the optimality tolerance, ϵ_2, the initial trust region radius, Δ_0, the maximum trust region radius, $\Delta_{\max}$, the minimum trust region radius, $\Delta_{\min}$, and the trust region scaling factor, κ, is assessed. For each setting three different values are used. The combination of settings which are used and their resulting semi-major axis increase, as well as the algorithm's runtime, are presented in Table 2.

Two settings seem to have the largest influence on the semi-major axis increase: the optimality tolerance, ϵ_2, and the minimum trust region radius, $\Delta_{\min}$, which were both defined in Section III.G. The smaller the optimality tolerance, the larger the final semi-major axis. This is as expected, since the optimality tolerance is the convergence criterion for the DDP algorithm. When a smaller value is used, the algorithm will iterate longer to find a more optimal trajectory. Any minimum trust region radius larger than zero leads to suboptimal trajectories. The trust region should be allowed to shrink when unsuccessful iterations take place, but it cannot become smaller than the minimum trust region radius. The trust region scaling factor, κ, also has a small effect on the optimality of the trajectory, for which an optimal value of 0.01 was found. The differences in semi-major axis resulting from varying the other settings are minor.

In general, the differences in runtime for different settings is small, the only large differences are observed for different optimality tolerances, where a smaller tolerance will lead to a longer runtime, and for varying the initial trust region radius, where the algorithm converges quickly to a suboptimal solution. The settings that result in the largest semi-major axis are: $\epsilon_1 = 0.1$ $\epsilon_2 = 1 \times 10^{-7}$, $\Delta_0 = \pi/2$, $\Delta_{\max} = \pi/2$, $\Delta_{\min} = 0$, and $\kappa = 0.01$. However, it is likely that by making the optimality tolerance, ϵ_2, even smaller, the resulting semi-major axis will become even larger. However, this increase in optimality will come at the cost of an increase in computational time. The aforementioned settings will be used in the remainder of this work, with the exception of the optimality tolerance, for which $\epsilon_2 = 1 \times 10^{-6}$ will be used. The difference in final semi-major axis between using $\epsilon_2 = 1 \times 10^{-6}$ and $\epsilon_2 = 1 \times 10^{-7}$ is very small, but comes at the cost of an increase in runtime of approximately 65%.

Table 2 Three-dimensional geostationary orbit: effect of varying DDP settings.

ϵ_1	ϵ_2	Δ_0	$\Delta_{\max}$	$\Delta_{\min}$	κ	Δa [km]	Runtime [s] [a]
0.05	1×10^{-6}	π	π	0	0.05	208.848116	271.9
0.2	1×10^{-6}	π	π	0	0.05	208.843899	240.3
0.5	1×10^{-6}	π	π	0	0.05	208.845164	241.9
0.1	1×10^{-5}	π	π	0	0.05	208.767372	131.0
0.1	1×10^{-6}	π	π	0	0.05	208.850688	213.6
0.1	1×10^{-7}	π	π	0	0.05	208.864644	351.5
0.1	1×10^{-6}	$\pi/2$	π	0	0.05	208.851152	214.1
0.1	1×10^{-6}	$3\pi/2$	π	0	0.05	208.850688	213.7
0.1	1×10^{-6}	2π	π	0	0.05	208.850688	215.0
0.1	1×10^{-6}	π	$\pi/2$	0	0.05	208.851109	213.7
0.1	1×10^{-6}	π	$3\pi/2$	0	0.05	208.851109	214.1
0.1	1×10^{-6}	π	2π	0	0.05	208.850013	213.9
0.1	1×10^{-6}	π	π	0.05	0.05	208.815523	160.0
0.1	1×10^{-6}	π	π	0.1	0.05	113.530786	6.8
0.1	1×10^{-6}	π	π	0.2	0.05	113.530786	6.3
0.1	1×10^{-6}	π	π	0	0.01	208.851657	219.8
0.1	1×10^{-6}	π	π	0	0.1	208.850182	217.9
0.1	1×10^{-6}	π	π	0	0.2	208.846977	222.2

[a] Computation was executed on sixteen threads of an Intel(R) Xeon(R) CPU E5-2683 v3 @2.00GHz.

There are more settings that can be changed than just the settings of the DDP algorithm itself. In Section III.F, two different trust region algorithms were discussed, the "classical TRQP" and the "simple TRQP". The classical TRQP is tuned by changing the value of the scaling matrix, M, whereas the simple TRQP is tuned by changing the magnitude of the hessian shift. The results of using the different algorithms with different settings are presented in Table 3. There are significant differences in runtime between the simple and classical algorithms: the simple algorithm converges multiple times faster, and it produces more optimal trajectories. It is interesting to note that the setting that converges fastest, is also the setting resulting in the most optimal trajectory. Therefore, in the remainder of this work, the simple TRQP with a hessian shift of 0.01 is used.

Table 3 Three-dimensional geostationary orbit: effect of varying TRQP settings.

TRQP	Scaling/Shift	Δa [km]	Runtime [s]
Simple	1×10^{-1}	208.736761	1347.2
Simple	1×10^{-2}	208.851109	223.1
Simple	1×10^{-3}	208.676761	692.1
Simple	1×10^{-4}	208.670943	750.7
Classical	1×10^{-3}	208.016431	2254.4
Classical	1×10^{-4}	208.466405	2455.0
Classical	1×10^{-5}	206.779676	4957.1
Classical	1×10^{-6}	206.632903	6358.9

Finally, the effect of using MBH has been investigated. Different combinations of settings for the chance and magnitude of the perturbation on the control variables were used. However, MBH was hardly able to improve on the solution found by DDP. In the cases where it did improve, the improvement was only very minor and came at the cost of a significant increase in computational time.

2. Effect of perturbing accelerations

So far, the only accelerations acting on the sailcraft have been point-mass gravity from the Earth and the solar-sail acceleration. The effect of including other perturbing accelerations in the dynamical model is shown in Table 4. The third-body perturbation from the Moon produces the largest effect; a decrease in the final semi-major axis of approximately 0.8 km is observed. Note that the magnitude of the semi-major axis decrease due to lunar gravity will most likely change when a different initial epoch is considered, since the acceleration acting on the satellite is dependent on the instantaneous position of the Moon. The effect of considering J_2, J_3, J_4 and solar gravity is very small, where J_2 has some effect, but considerably less than the gravitational acceleration from the Moon. There is no significant impact on the runtime. With all perturbing accelerations taken into account, a final semi-major axis increase of 208.05 km is obtained, which is significant considering that only three orbital revolutions are simulated. Therefore, it appears that a current state-of-the-art solar sail is a viable method of propulsion for orbit raising at geostationary altitude.

Table 4 Three-dimensional geostationary orbit: effect of varying dynamical model settings.

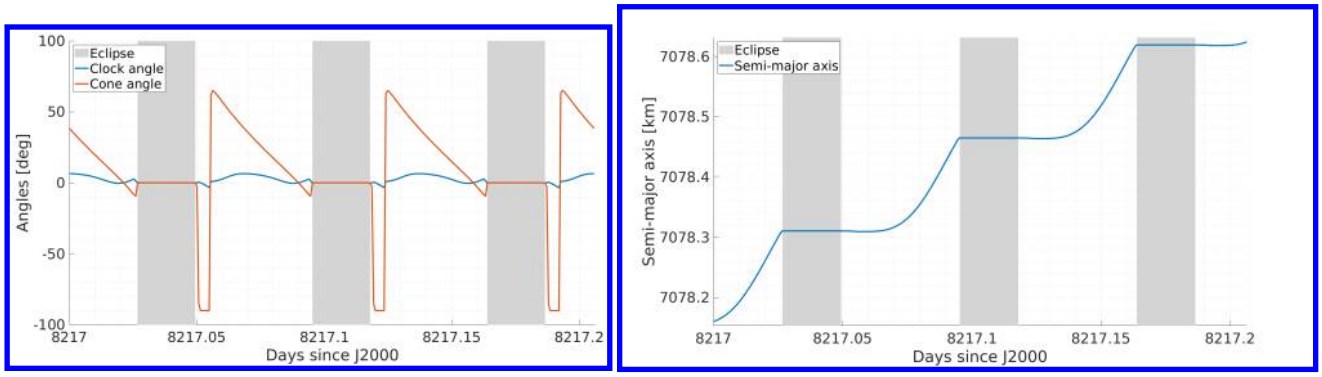
Perturbations	Δa [km]	Runtime [s]
None	208.851109	223.1
Moon	208.062980	232.8
J_2 and Moon	208.051849	227.3
J_2, J_3, J_4, Moon and Sun	208.049825	242.5

C. ACS3 mission: three orbital revolutions

For the analysis in this section, the sail characteristics from NASA's upcoming ACS3 mission are considered. It is assumed that the sailcraft will be launched into a 700 km altitude noon-midnight Sun-synchronous orbit (inclination $i = 98.19°$), with a solar-sail characteristic acceleration of 0.05 mm/s^2 (corresponding to a solar-sail lightness number of 0.0084)[‡]. The scheduled launch date is assumed to be June 1st 2022, which means that the sail will be operational from approximately July 1st 2022 onwards[‡]. Using the aforementioned initial conditions, the specific orbital energy of

[‡]W.K. Wilkie (NASA Langley Research Center), personal communication, May 2021

the satellite is optimised for a total of three revolutions. Since no constraints are considered yet, the cost function is identical to Eq. 40. The resulting control profile and semi-major axis increase are visualised in Figure 4. Considering the noon-midnight orientation of the Sun-synchronous orbit, the sailcraft spends significant amounts of time in eclipse, which is clearly visible in Figure 4: during eclipse, the semi-major axis does not increase and the cone and clock angles are set to zero.



(a) Control angles. **(b) Semi-major axis.**

Fig. 4 ACS3 mission: three orbital revolutions.

For the results presented in Figure 4, the only accelerations acting on the sailcraft have been point-mass gravity from the Earth and the solar-sail acceleration (including eclipses). The effect of including other perturbing accelerations in the dynamical model is shown in Table 5. The J_2 effect has the greatest impact on the final semi-major axis. The other contributions of the non-spherical Earth, J_3 and J_4, have some effect, whereas the inclusion of lunar and solar gravity makes no significant difference. Interestingly, the algorithm converges quicker when J_2 is included, whereas it takes longer with the other perturbing accelerations.

Table 5 ACS3 mission: effect of varying dynamical model settings.

Perturbations	Δa [km]	Runtime [s]
None	0.464314	260.9
J_2	0.388996	237.0
J_2, J_3 and J_4	0.385446	303.4
J_2, J_3, J_4, Moon and Sun	0.385513	304.6

1. Constrained DDP optimisation

This test case is used to demonstrate that DDP can deal with terminal constraints by implementing a terminal constraint to the cost function, which is minimised alongside the original objective. As stated previously, the sailcraft is placed in a Sun-synchronous orbit. However, as soon as the semi-major axis of the orbit changes, the Sun-synchronous attribute is lost. Only by simultaneously changing the inclination of the orbit can the orbit remain Sun-synchronous. The relation between the inclination and semi-major axis for a Sun-synchronous orbit is [32]:

$$\dot{\Omega} = -3\pi \frac{J_2 R_\oplus^2}{p^2} \cos(i) \frac{1}{T} = \frac{2\pi}{T_{\text{es}}} \tag{42}$$

where T is the orbital period of the satellite and T_{es} is the length of a sidereal day. When both sides of Eq. 42 are equal, the orbit is Sun-synchronous. The constraint is formulated by requiring that the difference between the left- and right-hand sides of Eq. 42 is as close as possible to zero. Then, by assuming a near-circular orbit, the constraint becomes

$$\psi = \cos(i) + \frac{4\pi/T_{ES}}{3J_2 R_\oplus^2 \sqrt{\mu_\oplus}} a^{7/2} \tag{43}$$

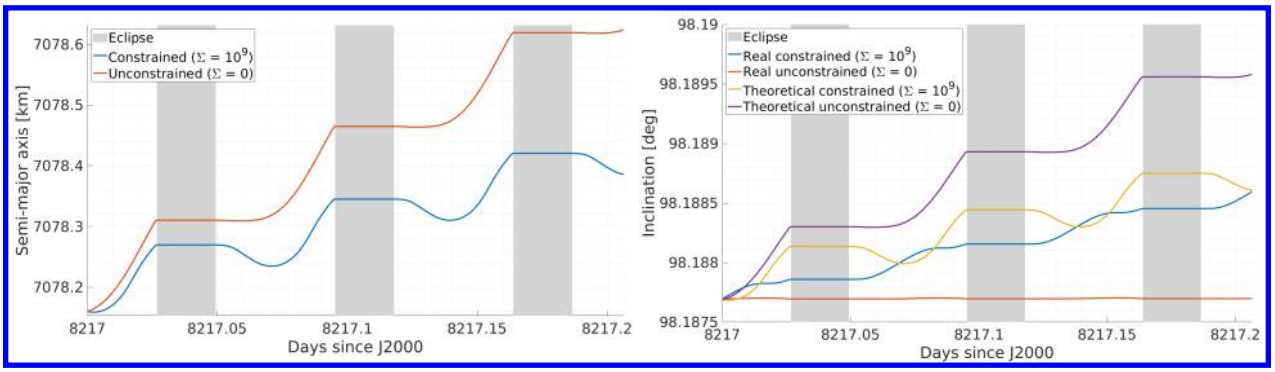
The complete cost function can then be written as

$$J = -\left(\frac{\dot{r}^2}{2} - \frac{\mu_\oplus}{r}\right) + \boldsymbol{\lambda}^T \left(\cos(i) + \frac{4\pi/T_{ES}}{3J_2 R_\oplus^2 \sqrt{\mu_\oplus}} a^{7/2}\right) + \left(\cos(i) + \frac{4\pi/T_{ES}}{3J_2 R_\oplus^2 \sqrt{\mu_\oplus}} a^{7/2}\right)^2 \boldsymbol{\Sigma} \tag{44}$$

When the orbit is Sun-synchronous, the value of the constraint will be equal to zero. When the orbit is not Sun-synchronous, the constraint will have a nonzero value and the DDP algorithm will adjust the control variables to minimise the constraint violation. Weights can be added to individual constraints in the augmented cost function using the penalty matrix, Σ. Increasing the weight on the inclination constraint will force the final inclination to be closer to the inclination that is required for Sun-synchronous conditions, but will lead to a smaller final semi-major axis, see Table 6. Here, Δi is the "real" inclination change resulting from the optimal trajectory found by DDP, whereas Δi_t is the "theoretical" inclination change that would be required for the final orbit to be Sun-synchronous. If the DDP algorithm is able to keep the trajectory under Sun-synchronous conditions, these "real" and "theoretical" inclinations are the same. Contrary, when no constraint is implemented, the "real" inclination will not change, since the DDP algorithm is solely focused on maximising the semi-major axis. The closer the "real" and "theoretical" inclination are to each other, the larger $\Delta i/\Delta i_t$, and the closer the final orbit is to Sun-synchronous conditions, but the smaller the final semi-major axis, since part of the sail acceleration is used for changing the inclination. In addition, the runtime of the algorithm until convergence increases with increasing weights. The evolution of the semi-major axis and inclination for the unconstrained and constrained ($\Sigma = 1 \times 10^9$) cases are shown in Figure 5a and 5b, respectively. It is clear that the final semi-major axis for the constrained case is smaller. Furthermore, the inclination for the constrained case stays close to the theoretical value, while for the unconstrained case the inclination hardly changes.

Table 6 ACS3 mission: effect of varying weights on inclination constraint.

Σ	Δa [km]	Δi [deg]	Δi_t [deg]	$\Delta i/\Delta i_t$ [%]	Runtime [s]
0×10^6	0.464208	3.49504×10^{-6}	0.00189234	0.18	264.0
5×10^6	0.444366	0.000312147	0.00181144	17.2	490.1
10×10^6	0.392154	0.000562415	0.00159857	35.2	675.0
75×10^6	0.257702	0.000862530	0.00105042	82.1	1044.6
150×10^6	0.240958	0.000882698	0.000982161	89.9	1138.8
1000×10^6	0.225821	0.0009001161	0.000920452	97.8	1342.0



(a) Semi-major axis comparison. **(b) Inclination comparison.**

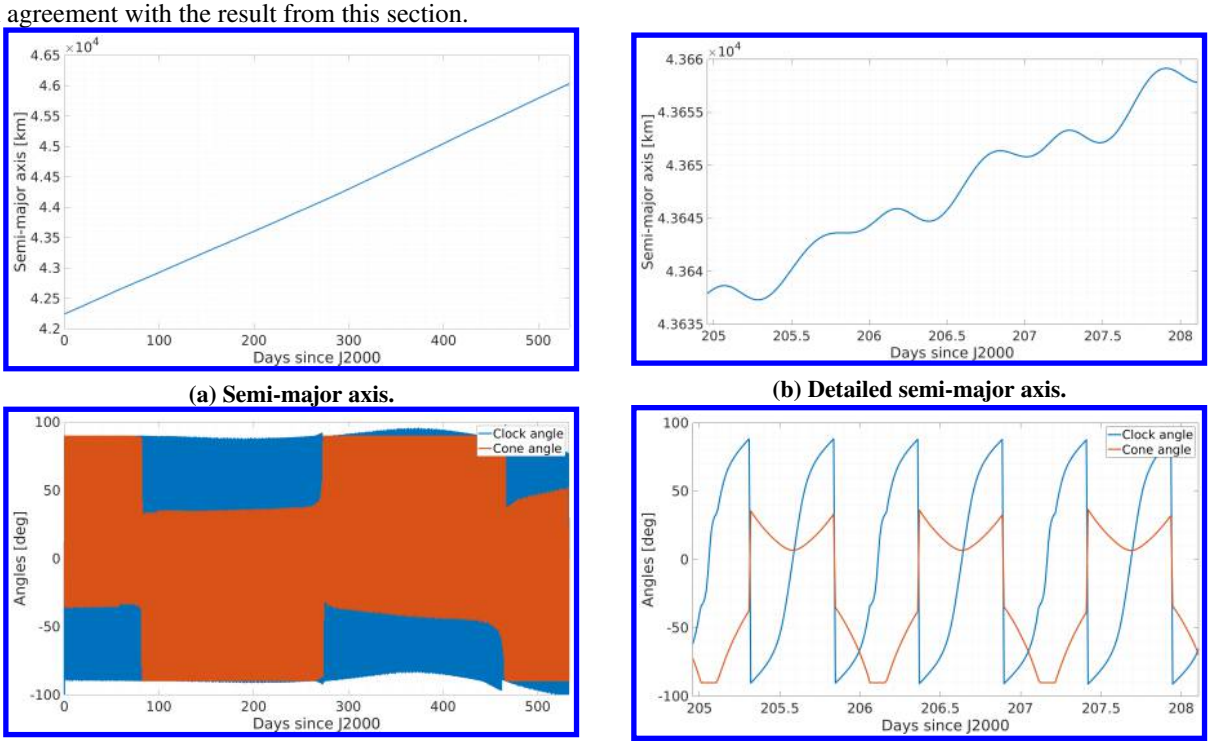
Fig. 5 ACS3 mission: constrained Sun-synchronous inclination, three orbital revolutions.

D. Three-dimensional geostationary orbit: 500 orbital revolutions.

Previously, in Section IV.B, a three-dimensional geostationary orbit-raising case was investigated for a duration of three orbital revolutions. Here, the case is extended to 500 orbital revolutions. In addition, a smaller, more realistic solar-sail lightness number is used, $\beta = 0.0011$. Again, for validation purposes, the optimisation is first executed when only the point-mass gravity of the Earth is considered and the results are compared to the results from the previously introduced locally optimal steering law [12]. The final semi-major axis increase found by the locally optimal steering law is marginally larger than the increase found by DDP (less than one kilometre over a total semi-major axis increase

of 3822 km). It can therefore once again be concluded that the solution found by DDP is a valid optimum. As stated before, it could be that the DDP algorithm finds a better solution when a stricter optimality tolerance is used.

From here on, all perturbing accelerations, J_2, J_3, J_4, solar gravity, lunar gravity and eclipses are taken into account. The resulting control angles and semi-major axis increase are shown in Figure 6. Though this figure does not provide details for individual revolutions, it does clearly show that the control profile adapts itself to the moving position of the Sun. Furthermore, the shape of the control profile for the first three revolutions was found to be exactly the same as that found in Section IV.B. The figures in the right column of Figure 6 show a detail of the figures in the left column after approximately 200 days have passed. The controls here are similar to the controls seen in Figure 3b, but mirrored due to the difference in the Sun's position. However, the semi-major axis increase shows a different pattern. In the three-revolution case without perturbations, the semi-major axis would always increase or remain unchanged, whereas for the 500 revolution case also a decrease in semi-major axis is observed. This behaviour can be attributed to the perturbations acting on the sailcraft, of which the lunar gravity is the predominant factor. A final semi-major axis increase of 3793.1 km is found. This increase is very large, especially considering that a solar-sail lightness number of $\beta = 0.0011$ was used, which is an order of magnitude smaller than the current state-of-the art. In Section IV.B the conclusion was drawn that a solar sail is a viable form of propulsion for orbit raising at geostationary altitude, which is in agreement with the result from this section.



(a) Semi-major axis.

(b) Detailed semi-major axis.

(c) Control angles.

(d) Detailed control angles.

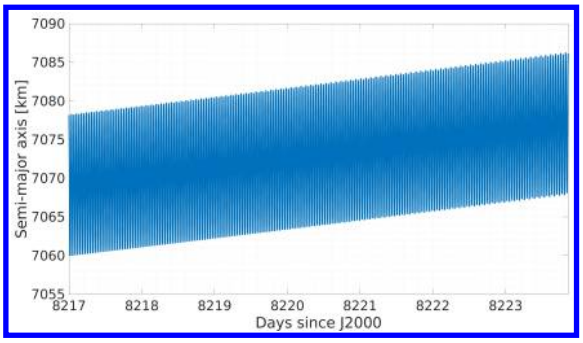
Fig. 6 Three-dimensional geostationary orbit: 500 orbital revolutions.

E. ACS3 mission: 100 orbital revolutions

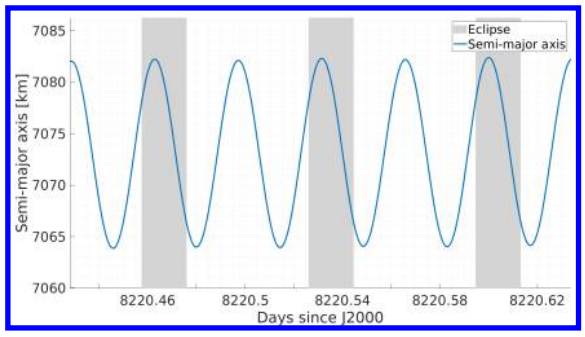
Previously, in Section IV.C, the ACS3 mission was investigated for a mission duration of three orbital revolutions. Here, the case is extended to 100 orbital revolutions. Furthermore, all perturbing accelerations (J_2, J_3, J_4, solar gravity, lunar gravity) and the Sun-synchronous inclination constraint ($\Sigma = 75 \times 10^6$) are taken into account. The resulting control angles, semi-major axis increase and inclination profile are shown in Figure 7. The figures in the right column again show a detail of the figures in the left column. A final semi-major axis increase of 7.85 km is found and the inclination constraint is met at $\Delta i/\Delta i_t = 99.4\%$. Considering that different weights have been used in the cost function for the three and 100 revolution cases, this result suggests that the optimal value for the weight is problem-dependent. Table 6 indeed shows that for the three-revolution case the constraint is met at 82.1% for a weight of $\Sigma = 75 \times 10^6$. The semi-major axis increase which is achieved by the sail is significantly smaller than the increase found at geostationary altitude in Section IV.D, but the time in which the increase is achieved is also very different. Here, the almost eight kilometre increase is achieved in approximately seven days, compared to over 500 days for the geostationary case.

Additionally, the perturbing accelerations have a different magnitude and the Sun-synchronous constraint negatively affects the semi-major axis increase. When taking these more challenging conditions into account, the solar sail is still able to achieve a significant increase in semi-major axis while the sailcraft is kept under Sun-synchronous conditions.

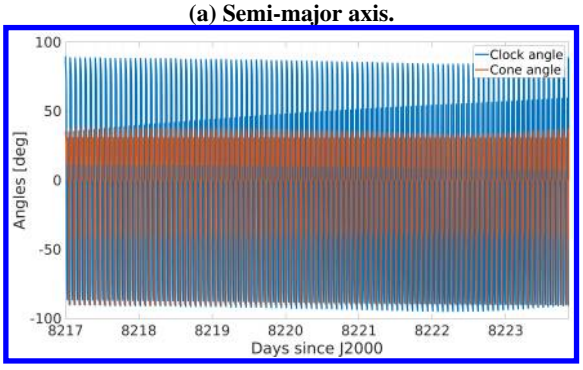
Several attempts have been made to increase the number of revolutions to one month of elapsed time, which would equate to approximately 452 orbital revolutions. However, no successful iterations were found by DDP with the settings mentioned in Section IV.B. Increasing the reduction ratio tolerance, ρ, leads to the algorithm finding successful iterations and eventually to convergence, but the results show discontinuities in the controls and the algorithm is not able to meet the constraints. Attempts to use the result of an unconstrained optimisation as initial guess to the constrained optimisation also proved unsuccessful. Careful tuning of both the weight on the constraint and the optimisation settings might be needed to find a combination of settings that works for the many-revolution constrained optimisation case, which is deemed beyond the scope of this paper but is recommended for future investigations into the applicability of DDP for many-revolution constraint optimisation.



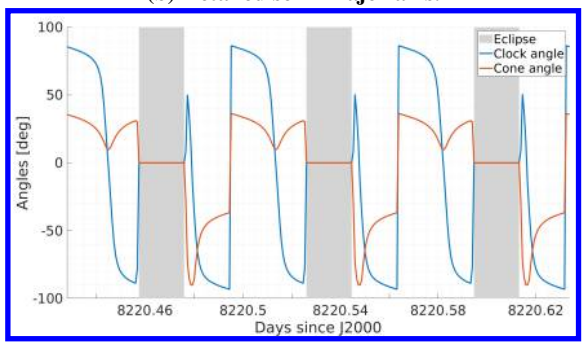
(a) Semi-major axis.



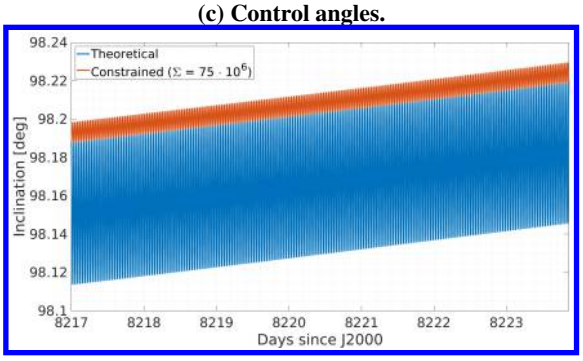
(b) Detailed semi-major axis.



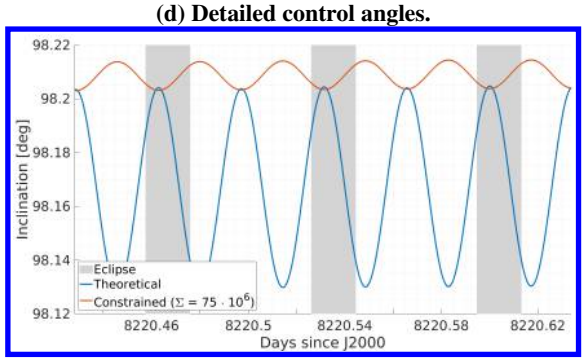
(c) Control angles.



(d) Detailed control angles.



(e) Inclination comparison.



(f) Detailed inclination comparison.

Fig. 7 ACS3 mission: 100 orbital revolutions.

V. Conclusions

In this work, the use of differential dynamic programming (DDP) to find optimal Earth-centred solar-sail trajectories has been investigated. For the two-dimensional case of orbit raising at geostationary altitude and starting from a zero-initial guess for the controls, the DDP algorithm has been proven to find very similar solutions compared to locally optimal steering laws. The DDP solution comes at the cost of an increased computational time, but DDP also has the benefit of being able to solve more complex and constrained Earth-centred solar-sail optimisation problems. The algorithm performs similarly optimal for a three-dimensional case (i.e., considering the actual obliquity of the ecliptic). Significant increases in the semi-major axis of the orbit were obtained (208 km after three revolutions) with a state-of-the-art solar-sail lightness number, indicating that a solar sail is a viable form of propulsion for orbit raising at geostationary altitude.

The effect of using different settings for the DDP algorithm has been investigated. The optimality tolerance and minimum trust region radius influence the final result the most. Different algorithms to solve the trust region quadratic subproblem were also tested. A hessian shifting technique obtained the most optimal results while also requiring the least computational time. Monotonic basin hopping (MBH) has been used to find globally optimal solutions with DDP in the inner loop of the algorithm. However, MBH did not significantly improve on the trajectory found by DDP.

Optimal orbit-raising solutions were also found for NASA's upcoming ACS3 mission. Additionally, constraints were successfully implemented to force the sailcraft to keep its Sun-synchronous attribute during the orbit raising manoeuvre. Alongside the sail's ability to keep the sailcraft under Sun-synchronous conditions, a notable increase in the semi-major axis of 7.85 km was obtained after 100 orbital revolutions.

The effect of considering perturbing accelerations, such as J_2, J_3, J_4, lunar and solar gravity, on the optimisation algorithm has been investigated. Irregardless of which perturbations are included in the dynamical model, DDP is able to find optimal trajectories.

To conclude, DDP is a viable tool for constrained and unconstrained many-revolution Earth-centred solar-sail trajectory optimisation.

References

[1] Kelly, P., and Bevilacqua, R., "Geostationary debris mitigation using minimum time solar sail trajectories with eclipse constraints," *Optimal Control Applications and Methods*, Vol. 42, No. 1, 2021, pp. 279–304. https://doi.org/10.1002/oca.2676.

[2] Kelly, P., and Bevilacqua, R., "An optimized analytical solution for geostationary debris removal using solar sails," *Acta Astronautica*, Vol. 162, 2019, pp. 72–86. https://doi.org/10.1016/j.actaastro.2019.05.055.

[3] Macdonald, M., and McInnes, C. R., "Realistic Earth Escape Strategies for Solar Sailing," *Journal of Guidance, Control and Dynamics*, Vol. 28, No. 2, 2005. https://doi.org/10.2514/1.5165.

[4] Green, A. J., "Optimal escape trajectory from a high earth orbit by use of solar radiation pressure." Ph.D. thesis, Massachusetts Institute of Technology, 1977.

[5] Guerrant, D., and Heaton, A., "Earth escape capabilities of the heliogyro solar sail," *Advances in the Astronautical Sciences*, Vol. 150, 2014, pp. 639–658.

[6] Heiligers, J., Ceriotti, M., McInnes, C. R., and Biggs, J. D., "Displaced Geostationary Orbit Design Using Hybrid Sail Propulsion," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 6, 2011. https://doi.org/10.2514/1.53807.

[7] McKay, R. J., Macdonald, M., Biggs, J., and McInnes, C., "Survey of Highly Non-Keplerian Orbits with Low-Thrust Propulsion," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 3, 2011, pp. 645–666. https://doi.org/10.2514/1.52133.

[8] Tsuda, Y., Mori, O., Funase, R., Sawada, H., Yamamoto, T., Saiki, T., Endo, T., Yonekura, K., Hoshino, H., and Kawaguchi, J., "Achievement of IKAROS — Japanese deep space solar sail demonstration mission," *Acta Astronautica*, Vol. 82, No. 2, 2013. https://doi.org/10.1016/j.actaastro.2012.03.032.

[9] Wilkie, W. K., "Overview of the NASA Advanced Composite Solar Sail System (ACS3) Technology Demonstration Project," *AIAA Scitech 2021 Forum*, 2021. https://doi.org/10.2514/6.2021-1260.

[10] Dachwald, B., "Optimization of very-low-thrust trajectories using evolutionary neurocontrol," *Acta Astronautica*, Vol. 57, 2005. https://doi.org/10.1016/j.actaastro.2005.03.004.

[11] Aziz, J. D., "Low-Thrust Many-Revolution Trajectory Optimization," , 2018. URL https://scholar.colorado.edu/concern/graduate_thesis_or_dissertations/jq085k18x.

[12] McInnes, C. R., *Solar sailing: technology, dynamics and mission applications*, Springer Science & Business Media, 2004.

[13] Gao, Y., "Near-Optimal Very Low-Thrust Earth-Orbit Transfers and Guidance Schemes," *Journal of Guidance, Control and Dynamics*, Vol. 30, No. 2, 2007. https://doi.org/10.2514/1.24836.

[14] Macdonald, M., and McInnes, C. R., "Analytical Control Laws for Planet-Centered Solar Sailing," *Journal of Guidance, Control and Dynamics*, Vol. 28, No. 5, 2005. https://doi.org/DOI:10.2514/1.11400.

[15] Jacobson, D. H., and Mayne, D. Q., "Differential dynamic programming," 1970.

[16] Aziz, J. D., Parker, J. S., Scheeres, D. J., and Englander, J. A., "Low-thrust many-revolution trajectory optimization via differential dynamic programming and a sundman transformation," *The Journal of the Astronautical Sciences*, Vol. 65, No. 2, 2018, pp. 205–228.

[17] Leary, R. H., "Global optimization on funneling landscapes," *Journal of Global Optimization*, Vol. 18, No. 4, 2000, pp. 367–383.

[18] Lantoine, G., and Russell, R. P., "A Hybrid Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems. Part 1: Theory," *Journal of Optimization Theory and Applications*, Vol. 154, 2012. https://doi.org/10.1007/s10957-012-0039-0.

[19] Montenbruck, O., and Gill, E., *Satellite Orbits, Models, Methods, Applications*, Springer, 2001.

[20] Mengali, G., and Quarta, A. A., "Optimal Three-Dimensional Interplanetary Rendezvous Using Nonideal Solar Sail," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 1, 2005. https://doi.org/10.2514/1.8325.

[21] Dachwald, B., MacDonald, M., McInnes, C. R., Mengali, G., and Quarta, A. A., "Impact of Optical Degradation on Solar Sail Mission Performance," *Journal of Spacecraft and Rockets*, Vol. 44, No. 4, 2007. https://doi.org/10.2514/1.21432.

[22] Aziz, J., Scheeres, D., Parker, J., and Englander, J., "A smoothed eclipse model for solar electric propulsion trajectory optimization," *Transactions of the Japan Society for Aeronautical and Space Sciences, Aerospace Technology Japan*, 2019, pp. 17–181. https://doi.org/10.2322/tastj.17.181.

[23] Kéchichian, J. A., "Inclusion of Higher Order Harmonics in the Modeling of Optimal Low-Thrust Orbit Transfer," *The Journal of the Astronautical Sciences*, Vol. 56, No. 1, 2008. https://doi.org/0.1007/BF03256541.

[24] Wakker, K. F., *Fundamentals of Astrodynamics*, Institutional Repository Library Delft University of Technology, 2015.

[25] Walker, M. J. H., Ireland, B., and Owens, J., "A set of modified equinoctial elements," *Celestial Mechanics*, Vol. 36, 1985.

[26] Betts, J. T., "Optimal low–thrust orbit transfers with eclipsing," *Optimal Control Applications and Methods*, Vol. 36, No. 2, 2015, pp. 218–240. https://doi.org/https://doi.org/10.1002/oca.2111.

[27] Bellman, R., and Kalaba, R. E., *Dynamic programming and modern control theory*, Vol. 81, Citeseer, 1965.

[28] Pellegrini, E., and Russel, R. P., "On the Computation and Accuracy of Trajectory State Transition Matrices," *Journal of Guidance, Control and Dynamcis*, Vol. 39, 2016. https://doi.org/10.2514/1.G001920.

[29] Dagum, L., and Menon, R., "OpenMP: an industry standard API for shared-memory programming," *IEEE computational science and engineering*, Vol. 5, No. 1, 1998, pp. 46–55.

[30] Conn, A. R., Gould, N. I., and Toint, P. L., *Trust region methods*, SIAM, 2000.

[31] Lantoine, G., and Russell, R. P., "A Hybrid Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems. Part 2: Application," *Journal of Optimization Theory and Applications*, Vol. 154, 2012. https://doi.org/10.1007/s10957-012-0038-1.

[32] Wertz, J. R., *Spacecraft Orbit and Attitude Systems, Orbit & Constellation Design & Management*, Microcosm Press and Springer, 2009.