Evolutionary Optimisation of a Flexible-Launcher Simple Adaptive Control System

Mooij, Erwin

DOI
10.1007/978-3-031-24812-2_9

Publication date
2023

Document Version
Final published version

Published in
Springer Optimization and Its Applications

Evolutionary Optimisation of a Flexible-Launcher Simple Adaptive Control System

Erwin Mooij

1 Introduction

From the early days in aeronautical engineering, where aircraft design problems have ranged from wing divergence and control reversal to dynamic flutter calculations for avoiding wing failure, static, and dynamic aeroelasticity issues have caused many control challenges and even loss of (fighter) aircraft during high-speed manoeuvring (Schwanz and Cerra 1984). Not only can aircraft suffer from aeroelastic effects, but also (small) conventional launch systems, which have long and slender bodies, may suffer from an unwanted coupling between the rigid body and its flexible modes. Even more so than for aircraft, this is not an isolated problem. During the launcher's flight, it uses a large amount of oxidiser and fuel, giving rise to large changes in mass properties and thus the flexible response. Because also the operational and atmospheric environment varies significantly, the entire flight profile should be examined rather than a single worst-case point, to identify the stability and controllability characteristics of the launch vehicle.

To control such a (very) non-linear system, robust non-linear control systems are required to stabilise the system and respond to both modelled and unmodelled disturbances. Two (non-linear) controllers that can potentially handle the aforementioned perturbations are an Incremental Non-linear Dynamic Inversion (INDI) controller and a system based on Simple Adaptive Control (SAC). Amongst others, INDI controllers have shown robust performance when applied to quadrotors (Smeur et al. 2016), both in a simulation environment and during flight tests, as well as hydraulic robot motion control (Huang et al. 2019). The alternative candidate, SAC, has shown good performance for a variety of applications in the fields of

E. Mooij (✉)
Delft University of Technology, Faculty of Aerospace Engineering, Delft, The Netherlands
e-mail: e.mooij@tudelft.nl

G. Fasano, J. D. Pintér (eds.), *Modeling and Optimization in Space Engineering*,
Springer Optimization and Its Applications 200,
https://doi.org/10.1007/978-3-031-24812-2_9

autopilot design (Barkana 2004), space-telescope control (Mehiel and Balas 2004), flexible structures (Barkana 2016), entry systems (Mooij 2018) and satellites with flexible appendages (Gransden and Mooij 2018).

The focus of this chapter is, however, not to design the best possible control system, but to develop a methodology to improve the performance of a controller with multiple design parameters. The design of linear control systems, such as the traditional proportional, integral, and derivative (PID) controllers, can be easily done, using, for instance, Bode plots, Nyquist criteria, etc. For non-linear controllers, the design criteria are much harder to formulate. Having concrete performance indices, though, would facilitate the use of optimisation algorithms that can then be used to improve the controller performance. To show the benefit of such a methodology, we will centre it around the design of a SAC system, as it has many more design parameters and would show the benefit of the proposed methodology.

Due to the nature of the control system design, performance optimisation is relatively difficult for the following reasons: (i) a large number of design parameters, (ii) a complex system, (iii) no analytical gradient information, (iv) a large parameter search space, (v) non-linear constraints and (vi) multiple objectives that may be conflicting. One (global) optimisation technique that may prove useful for this particular problem is the one based on evolutionary strategies. These have arisen from the desire to model the biological processes of natural selection and population genetics, with the original aim of designing autonomous learning and decision-making systems (Holland 1975). Evolutionary algorithms, and their binary counterparts genetic algorithms, have found widespread use in engineering systems (Zalzala and Fleming 1997), *e.g.,* aerodynamic plan-form design, optimal motion of industrial robot arms, and the design of VLSI layouts. Also, in the field of control engineering, many applications can be found: Tanaka and Chuang (1995) applied a genetic algorithm in combination with a neural network to the scheduling of linear controllers for the X-29, whereas Menon et al. (1995) used genetic programming for the synthesis of a non-linear flight control system of a high-performance aircraft. Fleming and Purshouse (2001) have given an extensive overview of genetic algorithms in control systems engineering: applications include controller design and system identification, as well as fault diagnosis, stability analysis and sensor–actuator placement.

It is stressed that the proposed design methodology could well be connected to other (multi-objective) optimisation algorithms, such as particle swarm optimisation, differential evolution or ant-colony optimisation, to name but just a few. The evolutionary algorithm has been selected as a showcase, mainly for its ease of implementation, not claiming that it is the best method to use. For similar reasons as the selection of the simple adaptive control system, we want to present a design approach for control systems with multiple design parameters and potentially conflicting performance objectives, thereby highlighting several steps in the design process.

As a reference, the two-stage PacAstro launcher for small payloads up to 225 kg has been selected for its availability of some geometrical and structural data.[1] The launcher is treated as a flexible beam with lumped masses to account for the subsystems and the fuel. Its design and modelling has been extensively discussed in earlier work, see, for instance, the work by Mooij and Gransden (2016), and will only be summarised here to have some insight in the system that we will be working with.

The layout of the rest of this chapter is as follows. In Sect. 2, the simulation model is introduced, *i.e.,* the pitch-plane state-space model of a flexible launcher. In Sect. 3, the control system design is covered, starting with the definition of performance metrics to be used in the (numerical) optimisation procedure, then followed by the theory on simple adaptive control systems, and concluded by a summary of the implementation and application of this control system to the flexible launcher. The optimisation problem is formulated in Sect. 4, including a top-level description of the evolutionary algorithm. Section 5 presents the results, divided into those obtained with a single-objective and multi-objective approach. Section 6 concludes this chapter with some final remarks.

2 Pitch-plane State-space Model of Flexible Launcher

For a first analysis towards investigating the stability and control characteristics of flexible launchers, it suffices to consider the (linearised) pitch-plane motion only. Mooij and Gransden (2016) describe a state-space model, derived for the error dynamics of a flexible launcher, and the configuration for which is shown in Fig. 1. For that error dynamics model, input is a modal description as a function of current mass, the normal-load and pitch-moment distribution, and, of course, the flight conditions. The mass matrix is created using a consistent formulation for a linearised beam element. Furthermore, the launcher is assumed to move with a steady-state velocity u_0, and the local deformation is determined by the combination of thrust, T, gravity, mg_d, aerodynamic normal force, N, and aerodynamic pitch moment, M.

In its general form, the system equation of this state-space model is given by

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{1}$$

with $\mathbf{A}$ and $\mathbf{B}$ being the system and control matrix, respectively. Due to the different nature of groups of state variables, it makes sense to partition $\mathbf{A}$ and $\mathbf{B}$ into sub-

[1] PacAstro was a US transportation service company, formed in 1990, to provide low-cost transportation of small satellites to Low Earth Orbit for approximately $5 million per launch using proven technology (Fleeter et al. 1992). Unfortunately, the launcher never came to operation despite several engine tests and three launch contracts, due to the lack of development funding. The company ceased to be in 1997. In Appendix A, some geometrical and mass properties have been provided.

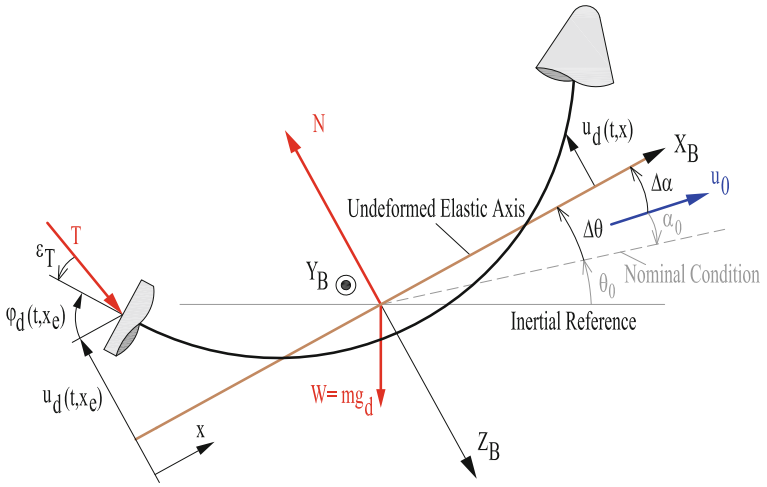

Fig. 1 Flexible vehicle definitions

matrices representing the rigid-body motion, the engine dynamics, and the flexible-body motion, thereby identifying the coupling terms between the different sets. The corresponding state-space matrices are then written as

$$\mathbf{A} = \begin{bmatrix} \mathbf{A_{RR}} & \mathbf{A_{RE}} & \mathbf{A_{RF}} \\ \mathbf{A_{ER}} & \mathbf{A_{EE}} & \mathbf{A_{EF}} \\ \mathbf{A_{FR}} & \mathbf{A_{FE}} & \mathbf{A_{FF}} \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} \mathbf{B_R} \\ \mathbf{B_E} \\ \mathbf{B_F} \end{pmatrix} \tag{2}$$

with

1. R for the rigid-body states angle of attack,[2] α, pitch angle, θ, and pitch rate, q
2. E for the engine states $\ddot{\varepsilon}_T$ (angular acceleration), $\dot{\varepsilon}_T$ (angular velocity), and ε_T (the angular position or swivel angle). These states originate from the assumption that the engine is modelled as an electro-hydraulic servo system, represented by a third-order transfer function.
3. F for the flexible-body states $\dot{\eta}_i$ and η_i for mode i. The total number of states in this group depends on how many bending modes n_f are taken into account.

[2] Pitch-plane translational motion is defined by u_0 and the vertical velocity, w. However, to study the rotational motion for a single point in the trajectory, it makes more sense to use the angle of attack, α, which can be derived from the (small) w through the relation

$$\Delta\alpha = \frac{\Delta w}{u_0} \quad \Rightarrow \quad \Delta\dot{\alpha} = \frac{\Delta\dot{w}}{u_0} \tag{3}$$

The state vector, $\mathbf{x}$, is thus given by $\mathbf{x}^{\mathrm{T}} = \left(\alpha\, \theta\, q\, \ddot{\varepsilon}_T\, \dot{\varepsilon}_T\, \varepsilon_T\, \dot{\eta}_1\, \eta_1\, \ldots\, \dot{\eta}_{n_f}\, \eta_{n_f}\right)^{\mathrm{T}}$. The only control is the commanded swivel angle, so $u = \varepsilon_{T,c}$. The corresponding sub-matrices, derived by Mooij and Gransden (2016), are given in Appendix B.

The engine is considered to be an electro-hydraulic servo system, approximated by a third-order system (Rolland Collette 1967),

$$\left(s^3 + 2\zeta_e\omega_e s^2 + \omega_e^2 s + K_e\omega_e^2\right)\varepsilon_T = K_e\omega_e^2\varepsilon_{T,c} \tag{4}$$

with defining parameters ω_e and ζ_e, the natural frequency and damping of the engine dynamics, and a gain, K_e, an amplification factor that improves the response (time). The values used in this study are $\omega_e = 50\,\mathrm{rad/s}$, $\zeta_e = 0.7$, and $K_e = 15$. Given the above transfer function means that the acceleration derivative ($\dddot{\varepsilon}_T$) is excited by $K_e\omega_e^2\varepsilon_{T,c}$, with the latter parameter being the commanded swivel angle. In case of a significant attitude correction, $\varepsilon_{T,c}$ may be deflected at its limit value (of $\pm 6°$), which means that $\dddot{\varepsilon}_T = 3927\,\mathrm{rad/s^3}$. Consequently, even after a mere 0.01 s, the acceleration $\ddot{\varepsilon}_T$ will be about 40 rad/s^2. It is clear that therefore some (mechanical) limits should be imposed on the engine states. As mentioned, $\varepsilon_{T,max} = 6°$, but the values for the maximum acceleration are not known for the PacAstro. Sutton and Biblarz (2017) mention a value of $\ddot{\varepsilon}_T = 30\,\mathrm{rad/s^2}$ and $\dot{\varepsilon}_T = 20\,\mathrm{rad/s}$ for the Space Shuttle main engines. Even though these engines may be heavier than the one of the current study, they give a good indication. Considering smaller (and lighter) nozzles, the acceleration limit is put to $\ddot{\varepsilon}_{T,max} = 50\,\mathrm{rad/s^2}$.

In Fig. 2, the reference trajectory of the PacAstro until first-stage burnout ($t_f = 126\,\mathrm{s}$, $h_f = 67.7\,\mathrm{km}$) is plotted. The figure shows an almost linear increase in velocity with altitude and, similarly, also in Mach number up to a maximum value of $M = 8.8$. The dynamic pressure peaks at $\bar{q} = 43.5\,\mathrm{kPa}$ at around $h = 11.1\,\mathrm{km}$ (time of maximum dynamic pressure (TMDP), $t = 63\,\mathrm{s}$, $M = 1.83$). With an initial

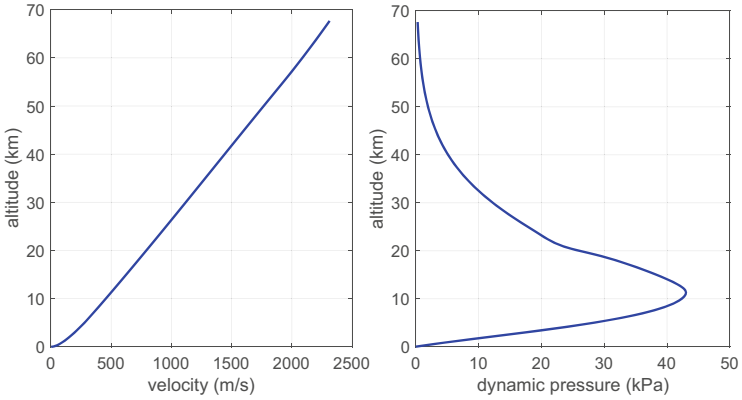

Fig. 2 PacAstro reference trajectory until first-stage burnout

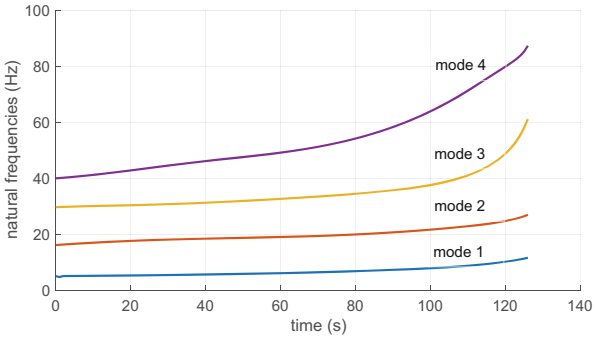

Fig. 3 Natural frequencies of the launch vehicle during flight

mass of $m_0 = 32{,}475\,\text{kg}$ and a constant mass flow of $\dot{m} = -186.6\,\text{kg/s}$, the final mass is $m_f = 8963.4\,\text{kg}$.

The normalised bending modes, necessary to find the in-flight bending deformations of the longitudinal axis, were calculated in Matlab® using an in-house finite-element mesher and solver. At $t = 0\,\text{s}$, the four lowest initial natural frequencies are 5.2, 16.2, 29.7, and 40.0 Hz, respectively, whereas their variation as a function of the flight time is shown in Fig. 3.

To generate an N-degree-of-freedom approximate differential equation model for a continuous system, the displacement of the continuous system is expanded as a linear combination of N prescribed shape functions. In other words, the deformation $u(x,t)$ is approximated by

$$u_d(x,t) = \sum_{i=1}^{N} \phi_i(x)\eta_i(t) \tag{5}$$

where x is the spatial coordinate, t is the time, $\phi_i(x)$ is the ith assumed mode shape, $\eta_i(t)$ is the ith generalised coordinate, and N is the number of terms or modes that are included in the approximation. The rotation $\varphi_d(x,t)$ of (an element of) the structure is given by

$$\varphi_d(x,t) = -\sum_{i=0}^{N} \sigma_i(x)\eta_i(t) \tag{6}$$

with $\sigma_i(x) = -\frac{\mathrm{d}\phi_i(x)}{\mathrm{d}x}$. In the current research, for the mode shapes, ϕ_i, the eigenvectors, derived from the finite-element model's mass and stiffness matrices, will be used.

3 Control System Design

3.1 Introduction

As benchmark, we assume a feedback law with proportional and derivative gains of $K_p = 2.8$ (on θ) and $K_d = 0.9\,\text{s}$ (on q) (Mooij and Gransden 2016). This design is based on a closed-loop rigid-body requirement of $3\,\text{rad/s} \leq \omega_r \leq 8\,\text{rad/s}$, with a damping factor of $\zeta \approx 0.7$, and designed for the point of maximum dynamic pressure ($t = 63\,\text{s}$).[3]

The Bode plot for the elastic system is given in Fig. 4 for the moment of maximum dynamic pressure. It shows that the elastic mode may pose a problem while controlling high-frequency oscillations due to, for instance, turbulence. It is clear that perturbations will be amplified while controlling an error in the pitch angle (by using the engine swivel angle). However, in case the deformations remain small, the problems will most likely remain limited. At its natural frequency of 37.3 rad/s, the bending mode spikes. The second bending mode spikes at a frequency around 105 rad/s and will probably have marginal to no effect on the control.

Section 3.2 will summarise the design elements of the selected robust control system that should be able to counter the (non-linear) effects of engine dynamics and flexible modes.

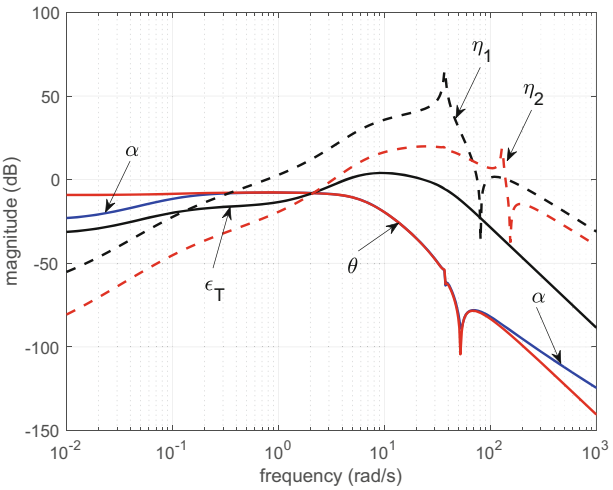

Fig. 4 Bode plots for the time of maximum dynamic pressure ($t = 63\,\text{s}$)

[3] The achieved closed-loop natural frequencies for rigid body and engine were $\omega_{r,cl} = 4.9\,\text{rad/s}$ (rigid body) and $\omega_{e,cl}$ 37.1 rad/s (engine), with damping factors $\zeta_{cl} = 0.75$ and $\zeta_e = 0.64$.

To allow for comparison of the different control system designs, the performance of a controller can be judged by several metrics. For the current control system design, we may look at the minimum state deviation of the plant with respect to the guidance commands. Another objective in the design could be to minimise the control effort that is required to influence the launcher's behaviour. For instance, in the case of launcher control system design, these two objectives can be expressed as the integrated pitch-angle deviation and the integrated swivel angle (equivalent to, for instance, the total hydraulic power required), and are given by

$$\sum_{\theta_{err}} = \int_0^t |\theta_c(t) - \theta_p(t)| \mathrm{d}t \qquad \sum_{\varepsilon_T} = \int_0^t |\varepsilon_T(t)| \mathrm{d}t \tag{7}$$

A graphic representation of the above metrics is shown in Fig. 5a, represented by the grey areas enclosed by the curves. It may be obvious that both individual areas should be as small as possible for optimal controller performance, which means that their numerical equivalent can be used to evaluate different controller designs. In the given example, $\sum_{\theta_{err}} = 6.55°\text{s}$ and $\sum_{\varepsilon_T} = 17.21°\text{s}$.

Another metric could be the oscillatory behaviour of either the state or control variables. Oscillations in the control may not only be energy expensive and a burden on the hardware, and it could also lead to instabilities. To detect oscillations or otherwise discrete changes in the controls, the cumulative moving standard deviation can be used. For a subset j of n_s out of a total of N samples of an arbitrary control signal u, the moving mean is defined as $\bar{y}_j = \frac{1}{n_s} \sum_{i=j}^{j+n_s-1} u_i$. Here, j will run from $j = 1+n_s/2$ to N-$n_s/2$, so each subsequent subset will shift by only one sample. Let the squared deviation from this mean be defined as $s_{u,j} = (u_{j+n_s/2} - \bar{y}_j)^2$, which represents the value at the midpoint of subset j. The cumulative standard deviation, F_u, for subset j is then

$$F_{u_j} = \sqrt{\frac{1}{N - n_s - 1} \sum_{k=1}^{j} s_k} \tag{8}$$

Figure 5b shows the oscillation pattern of the swivel angle for two (poor) controller performances. The cumulative standard deviation increases more rapidly when a discrete jump occurs or when there is an interval with persistent oscillations. As a metric, the grey area under the curve can be used, which, while minimised, would lead to a smoother behaviour. For the two cases shown, the numerical values are $F_{\varepsilon_T} = 36.2°$ and $116.9°$, respectively.

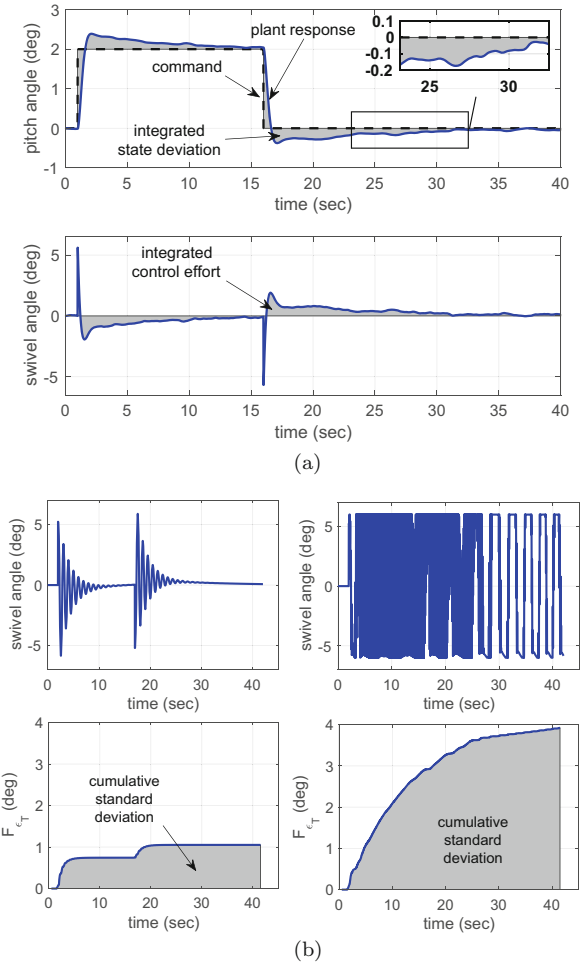

Fig. 5 Controller performance indices. (**a**) Integrated state deviation and control effort: $\sum_{\theta_{err}} = 6.55°\text{s}$ and $\sum_{\varepsilon_T} = 17.21°\text{s}$. (**b**) Oscillation metrics: $F_{\varepsilon_T} = 36.2°$ (left) and $116.9°$ (right)

3.2 Simple Adaptive Control

The concept of so-called simple adaptive control (SAC) is based on the principle of tracking the output of a reference model (Kaufman et al. 1998). Therefore, this system could also be classified as a model reference adaptive control (MRAC) system, although a principal difference from the original MRAC is that full state knowledge of the plant to be controlled is not required. A schematic overview of

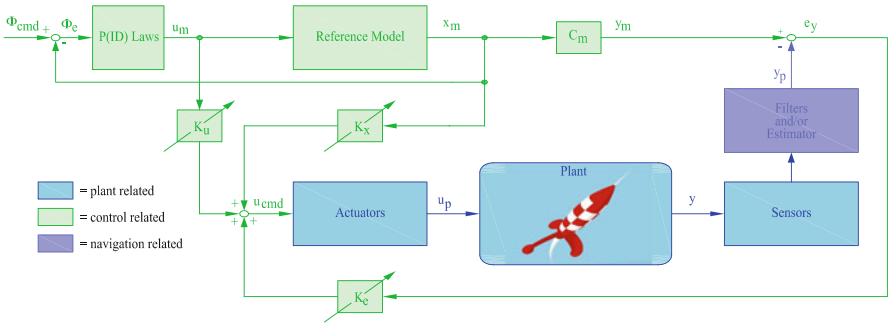

Fig. 6 Basic architecture of a simple adaptive control algorithm

a simple adaptive controller is shown in Fig. 6. The (single-actuator) control law is given by

$$u_p(t) = \mathbf{K_r}(t)\mathbf{r}(t) \tag{9}$$

where $\mathbf{r}(t) = \left(e_y(t)\ \mathbf{x_m^T}(t)\ u_m^T(t)\right)^T$ and $\mathbf{K_r}(t) = \left(K_e(t)\ \mathbf{K_x^T}(t)\ K_u(t)\right)$. It can be seen that the model input $\mathbf{u_m}$ and model state $\mathbf{x_m}$ are required to form part of the input signal $\mathbf{u_p}$ to the plant. Moreover, the so-called output error e_y serves as a feedback quantity to form the third element that composes u_p. The three gains, *i.e.*, K_e, $\mathbf{K_x}$, and K_u, are adaptive.

To compute the adaptive gains, $\mathbf{K_r}$ is defined to be the sum of a proportional and an integral component:

$$\mathbf{K_r}(t) = \mathbf{K_p}(t) + \mathbf{K_i}(t) \tag{10}$$

with

$$\mathbf{K_p}(t) = e_y(t)\mathbf{r}^T(t)\mathbf{T_p} \tag{11}$$

$$\mathbf{K_i}(t) = \mathbf{K_{i,0}} + \int_0^t e_y\mathbf{r}^T(t)\mathbf{T_i}\mathrm{d}t \tag{12}$$

In Eqs. (11) and (12), the weighting matrices $\mathbf{T_p}$ and $\mathbf{T_i}$ are positive semi-definite and positive definite, respectively. Note that the proportional gain component has a direct influence on the transient tracking behaviour but is strictly speaking not required to enforce asymptotic tracking, as $\mathbf{T_p}$ can be zero; tracking is guaranteed by the integral gain. To improve the transient response by only using an integral gain, a constant gain value has been added to $\mathbf{K_i}$. An advantage over the use of the proportional gain is that this constant value is independent of e_y and is therefore non-zero even if e_y is zero.

One way to improve the damping of the system is to include the error derivatives in the output error vector and apply a form of PD error scaling. The attitude controller aims at simultaneously reducing both the pitch angle, θ, and the pitch rate, q (which equals $\dot{\theta}$). This means that if errors in both are added together (with a proper scaling), the combined error would be similar to the output of a PD controller. And we know of such controller that the D term improves stability and damping of the control. Thus, the generic formulation of the output error becomes in that case

$$e_y = \mathbf{K}\left(\mathbf{y_m} - \mathbf{y_p}\right) = \mathbf{K}\left(\mathbf{C_m}\mathbf{x_m}(t) - \mathbf{C_p}(\mathbf{x_p}, t)\mathbf{x_p}(t)\right) \tag{13}$$

where $\mathbf{K}$ includes the appropriate ratio of adding the proportional and derivative signals together.

So far, an ideal environment has been considered. To cope with environmental disturbances, such as wind gust and turbulence, that lead to a persistent non-zero error and therefore to a continuous change in the integral gain $\mathbf{K_i}$, a robust design can be applied to adjust the integral gain and preventing it from reaching very high values. The integral term of Eq. (11) is adjusted as follows:

$$\dot{\mathbf{K}}_\mathbf{i} = e_y(t)\mathbf{r}^{\mathrm{T}}\mathbf{T_i} - \sigma_i \mathbf{K_i}(t) \tag{14}$$

Without the σ_i-term, $\mathbf{K_i}(t)$ is a perfect integrator and may steadily increase (and even diverge) whenever perfect output-following is not possible. Including the σ_i-term, $\mathbf{K_i}(t)$ is obtained from a first-order filtering of $e_y(t)\mathbf{r}^{\mathrm{T}}\mathbf{T_i}$ and, therefore, cannot diverge, unless the output error diverges.

3.3 Implementation

The current application of SAC focuses on a flexible launch vehicle with third-order engine dynamics. As a reference model, a simplified model is chosen: a rigid representation of the same launcher, stabilised by a PD controller and ideal engine dynamics. The reference model includes pitch angle and pitch rate only, contrary to the rigid-body model given by Eq. (B.1). In this way, the model is insensitive to angle-of-attack perturbations and will provide a more stable model that is easier to follow. The reference model is excited by the output error, *i.e.,* the current difference between model output and plant output, and transformed to equivalent model state errors. If at every control sample the difference between the current state error and the one of the previous sample is added to the model state vector, the model controller will bring this error back to zero. In that way, the signals u_m and $\mathbf{x_m}$ are created. Together with e_y, Eq. (13), the vector $\mathbf{r}$ is composed, and the plant input u_p can be calculated according to Eq. (9), after having calculated $\mathbf{K_p}$ and $\mathbf{K_i}$, Eqs. (11) and (12).

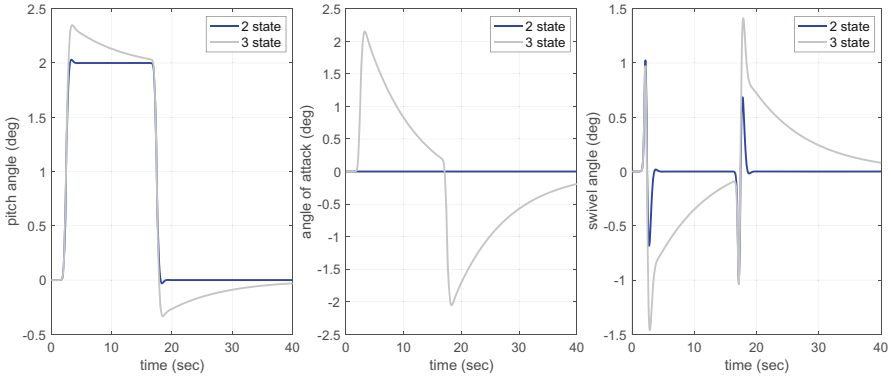

Fig. 7 Reference-model response to a 2° step command in pitch angle: difference between the two-state and three-state models (includes angle of attack)

So, the reference model is given by the reduced rigid-body model, and the full form of which is given in Appendix B:

$$\begin{pmatrix} \dot{\theta}_m \\ \dot{q}_m \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \frac{C_{m_q} \bar{q} S_{ref} d_{ref}}{I_{yy}} \end{bmatrix} \begin{pmatrix} \theta_m \\ q_m \end{pmatrix} + \begin{bmatrix} 0 \\ \frac{L_e T}{I_{yy}} \end{bmatrix} \varepsilon_{T,m} \tag{15}$$

so $\mathbf{x_m} = \begin{pmatrix} \theta_m & q_m \end{pmatrix}^{\mathrm{T}}$ and $u_m = \varepsilon_{T,m}$. The reference model is controlled by the benchmark PD controller, with $K_p = 2.8$ on the pitch-angle error and $K_d = 0.9$ on the pitch-rate error. The attitude command that the launcher has to follow is now enforced on the reference model, which will provide a smooth transient response that the plant will try to follow. The output error, e_y, will be formed as $e_y = K_p(\theta_m - \theta_p) + K_d(q_m - q_p)$, with the initial setting for the gains to be the same as for the reference-model controller. It is noted that both the model-controller gains and the output-error gains could be part of the optimisation process and do not necessarily have to be the same. However, initial runs showed that using the selected gains gives a good model performance. To avoid having an excessive number of design variables, we will keep these values for now. Finally, the nominal controller frequency is set to 100 Hz. This may seem like a rather large value, but it is, unfortunately, one of the characteristics of SAC (Messer et al. 1994).

Figure 7 shows the transient response of the reference model, while being subjected to a pitch-angle command of $\theta_c = 2°$ for the time of maximum dynamic pressure ($t = 63$ s). For each plot, two curves are shown, *i.e.,* one for the model with two states, given by Eq. (15), and one for a three-state model. The latter model also includes the angle of attack, see Eq. (B.1) for the corresponding state matrix. The difference in response is immediately clear. Due to the strong coupling between angle of attack and pitch motion, a large angle of attack is induced. Because the launcher is unstable ($C_{m_\alpha} > 0$), it takes a lot longer and more control effort to stabilise the system. The two-state model settles almost immediately, and this is

exactly the kind of behaviour that we want the plant to have, thus confirming the choice of reference model.

The design parameters of the adaptive controller are the weighting matrices, $\mathbf{T_p}$ and $\mathbf{T_i}$, the initial values of the integral gain, $\mathbf{K_{i,0}}$, and, as a safeguard against diverging output errors, the filter parameter, σ_i. The full form of either $\mathbf{T_p}$ or $\mathbf{T_i}$ is given by

$$\mathbf{T} = \begin{bmatrix} T_{e_y e_y} & T_{e_y \theta_m} & T_{e_y q_m} & T_{e_y \varepsilon_{T,m}} \\ T_{\theta_m e_y} & T_{\theta_m \theta_m} & T_{\theta_m q_m} & T_{\theta_m \varepsilon_{T,m}} \\ T_{q_m e_y} & T_{q_m \theta_m} & T_{q_m q_m} & T_{q_m \varepsilon_{T,m}} \\ T_{\varepsilon_{T,m} e_y} & T_{\varepsilon_{T,m} \theta_m} & T_{\varepsilon_{T,m} q_m} & T_{\varepsilon_{T,m} \varepsilon_{T,m}} \end{bmatrix} \tag{16}$$

which represents 16 design parameters per matrix. However, it is common practise to restrict to diagonal matrices, and thus only four parameters per matrix remain. With four initial gain values, $\mathbf{K_{i,0}}$, and a single filter parameter σ_i associated with e_y, the total number of design parameters can vary between 13 and 37.[4]

4 Optimisation Problem

Ever since its conception in the 1970s, there has been a non-wavering interest in problem-solving systems based on the principles of evolution: such systems maintain a population of potential solutions and include some selection process, which is based on the fitness of individuals, and some "genetic" operators that allow for the creation of new individuals. Almost each genetic (operating on binary strings) or evolutionary (operating on "real-life" parameters, *e.g.,* floating-point variables) algorithm has the following structure (Goldberg 1989, Michalewicz 1996):

```
begin
    initialise population P(0) with N individuals;
    evaluate P(0) and assign fitness to individuals;
    t := 1;
    repeat
        select P(t) from P(t − 1);
        recombine P(t) through crossover and mutation;
        evaluate P(t) and assign fitness to individuals;
        t := t + 1;
    until (termination condition)
end
```

[4] The minimum number could actually be four, as the principal requirement that $\mathbf{T_i}$ is positive definite dictates four diagonal elements larger than zero. The positive semi-definite condition of $\mathbf{T_p}$ could in principle lead to $\mathbf{T_p} = \mathbf{0}$. All other parameters can be non-zero to improve the performance but can be zero in the baseline algorithm.

The above pseudo-code represents a probabilistic algorithm, which maintains a population of individuals, $P(t) = \{\mathbf{x_1^t} \cdots \mathbf{x_N^t}\}$ for iteration t. Each individual $\mathbf{x_i^t}$, which represents a potential solution to the problem at hand, is evaluated to give some measure of its "fitness." A new population is formed by selecting the more fit individuals. Some members of the new population undergo transformations by means of genetic operators: unary transformations (mutation), which create new individuals by a small change in a single individual, and higher order transformations (crossover), which create new individuals by combining parts from two or more individuals. The termination condition can be a predefined number of iterations (also called generations) or a convergence criterion.

The applied operators in this study are a combination of simple, arithmetic and heuristic crossover, and multi non-uniform mutation. A detailed description of these (and other) operators can be found in the book by Michalewicz (1996). Here, we will restrict to a brief description. In simple crossover, two parents p_1 and p_2 are selected, and a simple single-point crossover is applied. This means that an independent parameter in p_1 is selected at random and one child is formed by taking the first part of p_1 (up to the selected parameter) and the second part of p_2 (onwards from the selected parameter). A second child is formed by the two remaining parts. Arithmetic crossover takes two parents p_1 and p_2 and performs an interpolation over a random distance along the line formed by the two parents. Heuristic crossover, finally, takes two parents p_1 and p_2 and performs an extrapolation along the line formed by the two parents outward in the direction of the better parent.

Non-uniform mutation randomly selects one variable x_j of an individual p_i and sets it equal to a non-uniform random number, with a (random) dependency on the generation number. This latter Gaussian distribution starts wide and narrows to a point distribution as the current generation approaches the maximum generation. In multi-non-uniform mutation, this operator is applied to each independent variable of the chosen individual.

For the selection method, we use stochastic universal sampling (SUS) (Baker 1987), which is some variation of roulette wheel selection (RWS) in the sense that the fitness determines the probability that the individual is selected. However, contrary to RWS where only one individual is selected, in SUS up to N individuals can be selected with multiple equally spaced pointers. By applying SUS, clustering of individuals in subsequent generations (genetic drift) will be reduced.

The fitness Φ_i of an individual i can either be directly or indirectly derived from its objective function(s). In single-objective optimisation, any one of the above defined performance metrics can be used directly. In case two or more objective functions are used, a trade-off has to be made if improving one objective will result in the simultaneous degradation of the other. From the available methods to derive a single fitness value from multiple objectives, the concept of Pareto ranking is used (Cvetković 2000, Fonseca 1995).

Pareto-based fitness assignment was first proposed by Goldberg (1989), who suggested the use of non-domination ranking and selection to move a population towards the Pareto front in a multi-objective optimisation problem. Fonseca (1995)

discusses a variation of this ranking method, which will be used in this study and is summarised below.

The problem under consideration is to simultaneously minimise the k components of a vector function $\mathbf{f} = (f_1, f_2, \ldots, f_k)$, each of which is a function of the design variables $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. The problem has usually no unique solution, but a set of equally efficient (or non-inferior) alternative solutions, which together form the so-called Pareto-optimal set or Pareto front when they are plotted in relation to the inferior solutions. For a minimisation problem, a vector $\mathbf{f} = (f_1, f_2, \ldots, f_k)$ is said to be inferior to $\mathbf{g} = (g_1, g_2, \ldots, g_k)$ if $\mathbf{g}$ is partially less than $\mathbf{f}$, *i.e.,*

$$\forall \mathbf{i} \in \{1, 2, \ldots, k\},\ g_i \leq f_i \wedge \exists \mathbf{i} \in \{1, 2, \ldots, k\},\ g_i < f_i \tag{17}$$

Consider now an individual $\mathbf{x_f}$ at generation t with corresponding objective vector $\mathbf{f}$, and let $\mathbf{r_f}^{(t)}$ be the number of individuals in the current population which are preferable (or superior) to it. The current position of $\mathbf{x_f}$ in the individuals' rank is then given by

$$\text{rank}(\mathbf{x_f}, t) = r_f^{(t)} \tag{18}$$

Note that by definition the individuals that form the Pareto front all have rank zero. The rank-based fitness assignment is further developed as follows:

1. Sort population according to rank.
2. Assign fitness Φ by interpolating from the best individual to the worst according to some function. In the current study, an exponential function of the form

$$\Phi\left(r_f^{(t)}\right) = s\rho^{r_f^{(t)}} \tag{19}$$

   is used, where s is the relative fitness ($s > 1$), whereas the parameter ρ should fulfil

$$\sum_{i=0}^{N-1} = \frac{N}{s} \tag{20}$$

   with N being the population size. In the limit case of N approaching infinity, the above power series equals $\frac{1}{1-N}$, which yields $\rho = 1 - \frac{s}{N}$.
3. Average the fitness assigned to individuals with the same rank, to sample them at the same rate while keeping the global population fitness constant.

5 Results

In this section, we present the results of the optimisation study, where different combinations of design variables will be explored. Note that the results are in support to present (and discuss) the design methodology for control systems with more than just a few design parameters. All simulations are run for a single point in the trajectory, *i.e.,* the point of maximum dynamic pressure ($t = 63$ s). We will not try to get the best possible design and establish its robustness by doing an extensive sensitivity analysis. To do so would require quite some more space than we have available in this chapter.

To begin with, in Sect. 5.1, the parameter ranges will be established by doing a simple Monte Carlo simulation. Inspecting the results and varying the ranges will quickly lead to a design space that is not too large (and non-linear) and will allow the optimisation algorithm to find an optimal design. In case the design space is too large, convergence may become an issue for some evolutionary algorithms.

Section 5.2 will present the results for a single-objective optimisation. There, the focus is on minimising the integrated pitch-angle deviation, without looking at the associated control effort and possible oscillations in either state or control. Finally, in Sect. 5.3, the exercise is repeated by optimising multiple objectives simultaneously.

5.1 Design Space Exploration

The design space exploration will be done for the rigid launcher, as the optimisation later will be carried out for a so-called nominal (= ideal) system. The effect of engine dynamics and flexible modes will be studied later, after an optimal control system design has been established.

Assuming diagonal $\mathbf{T_p}$ and $\mathbf{T_i}$, four initial integral gains, and one filter parameter, the 13 parameters are varied according to a uniform distribution. After some trial and error, the final ranges that were determined are the following:

1. $\mathbf{T_p}$: $T_{e_y e_y} \in [0,400]$, $T_{\theta_m \theta_m} \in [0,800]$, $T_{q_m q_m} \in [0,1600]$, and $T_{\varepsilon_{T,m} \varepsilon_{T,m}} \in [0,1600]$.
2. $\mathbf{T_i}$: $T_{e_y e_y} \in [0.15,60]$, $T_{\theta_m \theta_m} \in [25,10000]$, $T_{q_m q_m} \in [1.5,600]$, and $T_{\varepsilon_{T,m} \varepsilon_{T,m}} \in [1,400]$.
3. $\mathbf{K_{i,0}}$: each gain is sampled from the interval [0, 5].
4. σ_i is sampled from [0, 2].

A total of $N = 2500$ runs were executed, using the Mersenne Twister random generator, initialised with random seed 0.

In Fig. 8, the four objectives, integrated state deviation, $\sum_{\theta_{err}}$, integrated control effort, $\sum_{\varepsilon_T}$, and the two corresponding oscillatory indices, F_θ and F_{ε_T}, have been plotted. It is obvious that there is a large variation in performance, but it appears

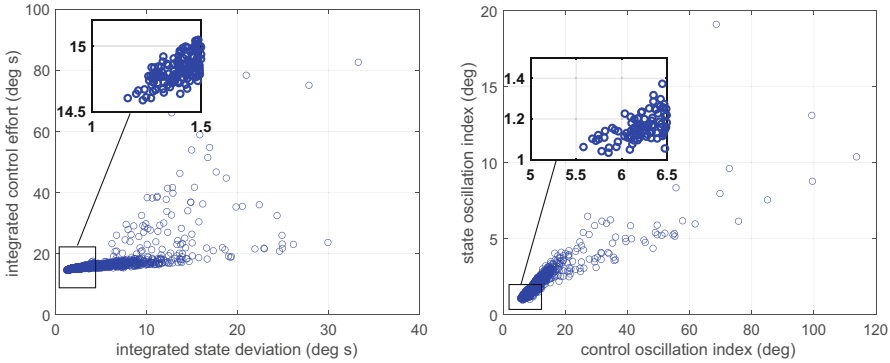

Fig. 8 Design space exploration: four objectives, $N = 2500$ Monte Carlo runs ($t = 63$ s)

that for each of the objectives there are quite small values. Just looking at the two plots indicates a dense band of points that seem to converge in the direction of some minimum value. Obviously, the integrated control effort can never reach zero because that would mean that the launcher is not controlled, immediately leading to a large integrated pitch-angle error. However, the two oscillation indices could converge to values close to zero, as that would imply a very smooth control. Note, though, that even a relatively smooth change of the control or state would already increase these objectives, so they could never be exactly zero.

It should also be noted that a small value in one objective does not necessarily mean a small value for all other objectives as well. In other words, the Pareto front in a four-dimensional objective space does not necessarily lie close to the bottom-left corner (the *origin*). In Sect. 5.3, the Pareto concept will be discussed in more detail.

Now that we have established the ranges of the independent parameters, we can move on to the next step, *i.e.,* the single-objective optimisation.

5.2 *Single-objective Optimisation*

Before any optimisation process can start, it is important to *tune* the algorithm, *i.e.,* to determine the (optimal) values of algorithm-specific settings, *in relation to the problem at hand*. The optimisation of the integrated state deviation by means of the evolutionary algorithm is done for 30 generations, with an initial population size of 50 individuals. The settings (and operators) we found to work well for the problem at hand were:

- Stochastic universal sampling as selection method, with the number of spins being 1

- Arithmetic crossover, with the total number of crossovers[5] being 40 and the number of retries to obtain a valid design with respect to the constraints being 10 (if no valid design is obtained, the objective value is penalised with a large value of 10^5).
- Multi-non-uniform mutation, with the number of mutations being 10, the maximum number of generations, $G_{max} = 30$, and the shape parameter, $b = 3$

To eliminate the random effect, the runs were repeated with different random seeds. We found that the solutions converged to (almost) the same value; nonetheless, re-running the optimisation multiple times should be standard practise.

For the first batch, the objective is the integrated state deviation, with the reference-model pitch angle being the target value. Figure 9a shows the average objective value, $\bar{y}$, as a function of generation number. Also included are the $\bar{y} \pm 1\text{-}\sigma$ lines, to show the spread of individuals in objective space as a function of the successive generations.[6] The results show a quick convergence, with reaching the best value after about eight generations. The response curves for the best individual are shown in Fig. 9b. The plant responds a bit faster than the reference model, which is most likely due to the perturbing angle of attack. As we mentioned, this perturbation is absent in the reference model. However, it induces a deviation between reference-model and plant pitch angle, which leads to larger gains to anticipate on the induced error. The global response of the plant becomes thus a bit more "aggressive." As the plant is faster, it will also slow down a bit quicker, thus becoming slower than the reference model close to the set point. Finally, the swivel angle remains non-zero for a much longer time than would be suggested by the practically zero error in the pitch angle. This is easily explained by the fact that the angle of attack needs quite a long time to be "pushed back" to zero.

In case we change the objective to be minimised to the integrated state deviation with respect to the guidance command,[7] *i.e.,* the step command, the results shown in Fig. 10a, b are obtained. At first sight, the results seem to be similar, but some small differences can be seen. In the first place, the converged minimum value is a bit smaller and convergence is slightly faster. In the second place, because the response is even faster than before, there is a slight overshoot of the plant pitch angle.

To compare the two final designs, in Table 1, the design parameters are listed. It is clear that both designs are not the same, although with some imagination one

[5] A crossover involves two parents and will produce two children. This means that given a number of crossovers and depending on their fitness (and selection), the population size is not constant and could be even double of the one with what we started.

[6] This figure shows that there is a significant spread in the initial (= first) generation, implying a good coverage of the design space. However, it is important to realise that a large standard deviation can also be caused by just a few outliers, not necessarily in the direction of the optimum.

[7] Note that the adaptive algorithm still aims at tracking the reference model. However, since the optimiser is forced to follow the guidance command, the resulting response is somewhat of a compromise: the optimiser pushes forward to the guidance command, whereas the controller pulls back towards the reference model.

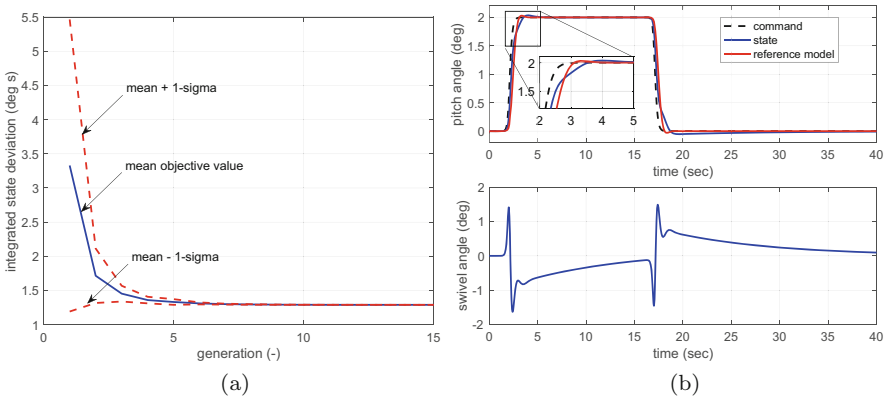

Fig. 9 Single-objective optimisation (reference-model tracking): mean fitness as a function of generation number and best individual's response. (**a**) Average fitness. (**b**) Best individual

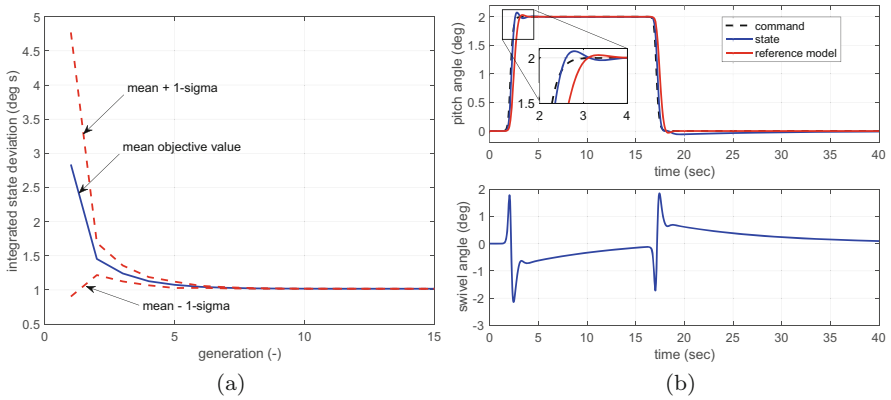

Fig. 10 Single-objective optimisation (command-value tracking): mean fitness as a function of generation number and best individual's response. (**a**) Average fitness. (**b**) Best individual

could say that the order of magnitude is the same. Conclusion from this is that there will be multiple solutions possible, leading to more or less the same performance, *i.e.,* there are possibly many local optima. That means that it is not always easy for the optimiser to find *the* optimal solution. Concerning the adaptive algorithm, it may be possible that minute changes in the design parameters have no effect on the performance; only relatively large variations will show in a change in performance.

Until now we have kept both the reference-model-controller gains, K_p and K_d, and the output-error gains, Eq. (13), constant and in both cases the same. However, as they are directly related to both the reference model and the plant response, it may be interesting to see whether the optimiser will also keep the gains constant (and the same). We will therefore add four more independent parameters: $\mathbf{K_m} =$

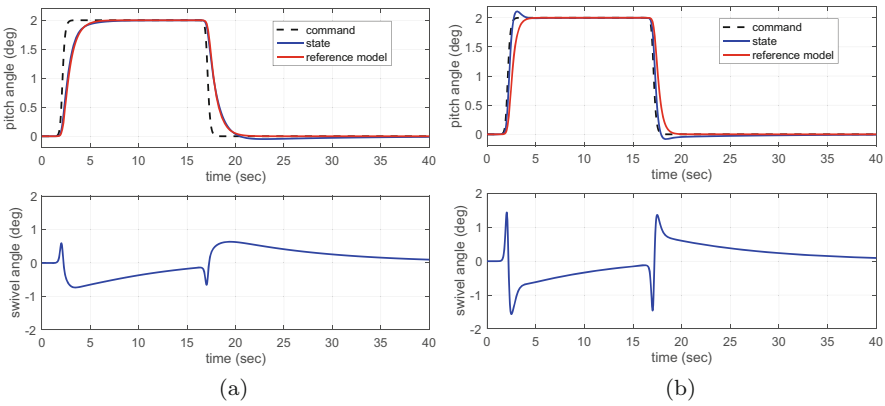

Fig. 11 Single-objective optimisation including with reference-model gains and output gains: response plots. (**a**) Reference-model tracking. (**b**) Guidance-command tracking

Table 1 Optimal design parameters simple adaptive controller (single-objective optimisation)

	Reference-model tracking				Guidance-command tracking			
	T_p	T_i	$K_{i,0}$	σ_i	T_p	T_i	$K_{i,0}$	σ_i
e_y	290	45	1.37	4.43	317	33	1.31	3.96
θ_m	670	5952	0.0	1.43	698	4248	0.0	1.48
q_m	737	279	0.0	0.63	419	151	0.0	3.20
$\varepsilon_{T,m}$	1238	246	0.0	1.02	803	125	0.0	1.59

$\left(K_{m,p}\ K_{m,d}\right)^{\mathrm{T}}$, with each gain varied over the range [0.5,5], and the output-error gains, $\mathbf{K_e} = \left(K_{e,p}\ K_{e,d}\right)^{\mathrm{T}}$, varied over the range [0.5,4].

With otherwise the same settings, the optimiser is run twice, once for reference-model tracking and once for guidance-command tracking. The response curves for the optimal designs are shown in Fig. 11, with corresponding gain values $\mathbf{K_m} = (3.50, 2.79)^{\mathrm{T}}$ and $\mathbf{K_e} = (2.84, 2.58)^{\mathrm{T}}$ for the former and $\mathbf{K_m} = (3.06, 1.53)^{\mathrm{T}}$ and $\mathbf{K_e} = (3.31, 1.20)^{\mathrm{T}}$ for the latter. Tracking the reference model leads to a more relaxed response and a much lower control effort. However, one should not forget that the reference-model response should meet with the requirements set to the attitude controller design. Thus, it might be that the response is too slow. A separate optimisation of the reference model is thus always required. In case of guidance-command tracking, one can also see that the reference model is quicker to act, with a steeper transient response. Comparing the results with those of Figs. 9b and 10b does show a smoother response, so it would be a good idea to keep the gains as design variables.

Comparing $\mathbf{K_m}$ and $\mathbf{K_e}$, they are obviously not the same, as we had assumed in our initial runs. Also, if we compare the gain sets between the two designs, they are not the same. Inspecting the other design parameters, they differ as well from the values listed in Table 1. All in all, we can conclude that the design space and its relation with the objective space is a complex one, which should be kept in mind.

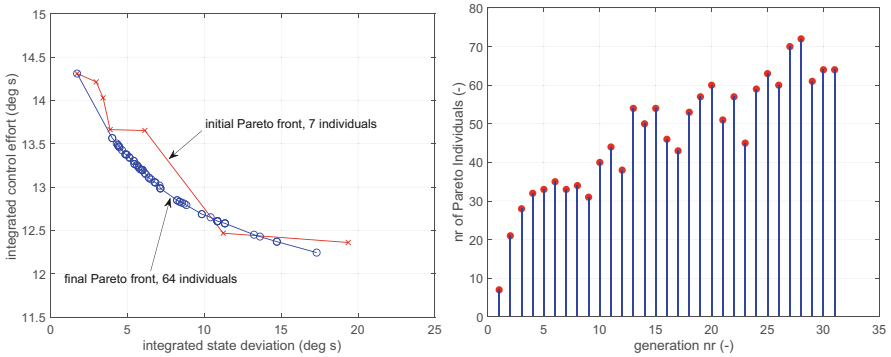

Fig. 12 Multi-objective optimisation (command-value tracking), rigid launcher. Initial population size 50 individuals. (**a**) Pareto front, initial and final. (**b**) Number of Pareto individuals

What we saw from the above results is that a smaller integrated state deviation means that the plant is closer in following the reference model or the guidance command. However, as found by Messer et al. (1994), this may indicate that the control effort is ever increasing. Comparing the two profiles of the swivel angle, we see that while tracking the command the swivel angle is indeed slightly larger. Therefore, it would be wise to include minimisation of the integrated control effort in the optimisation process. This will be addressed in the next section.

5.3 Multi-objective Optimisation

From the single-objective optimisation results, we found that a closer model or command-following would require more control effort, which was not taken into account in the optimisation process. Furthermore, we established that the design parameters to be included are $\mathbf{T_p}$, $\mathbf{T_i}$, $\mathbf{K_{i,0}}$, σ_i, $\mathbf{K_m}$, and $\mathbf{K_e}$, representing a total of 17 design parameters.

For our first batch of simulations, we will include the integrated control effort, $\sum_{\varepsilon_T}$, besides the integrated state deviation, $\sum_{\theta_{err}}$. With otherwise the same algorithm settings, the results shown in Figs. 12 and 13 are obtained. Two initial population sizes have been used, *i.e.,* 50 and 200, to try and find out which minimum population size we can use. Figure 12a shows the progression of the Pareto front for the 50-individual initial population. The progression of the front is clear, *i.e.,* in the midrange an improvement in $\sum_{\theta_{err}}$ of more than 10% is achieved. Also, the initial front was quite sparse, whereas the final front is much denser and seems to have converged quite well.

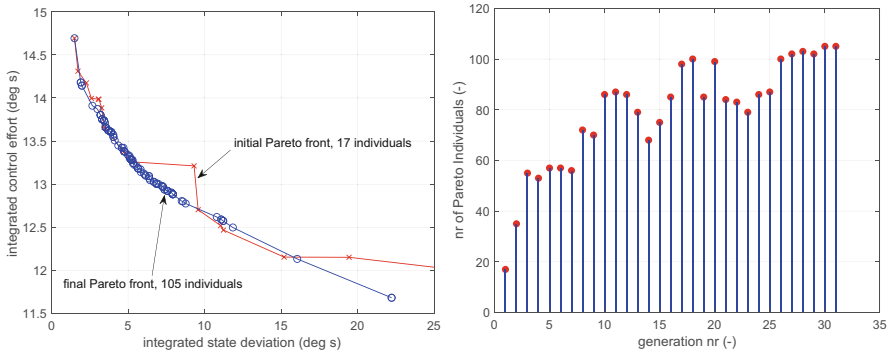

Fig. 13 Multi-objective optimisation (command-value tracking), rigid launcher. Initial population size 50 individuals. (**a**) Pareto front, initial and final. (**b**) Number of Pareto individuals

The total range of variation for either objective is 2°s for the control effort and 17°s for the state deviation. Each of the Pareto individuals represents a(n)(sub-) optimal control system design: selecting a better control effort has obviously an adverse effect on the state deviation, and *vice versa*, as was already established from the single-objective optimisation runs. In case either or both have a minimum/maximum requirement, a plot like this will allow for the selection of the right design.

One should not forget, however, that for any of these so-called heuristic methods it cannot be proven that the Pareto front has indeed converged to its "theoretical" value and that the global optimum has been found. It could be, after all, that the algorithm got stuck in a local minimum. A way to check this is to let the algorithm run for more generations, but looking at the "clustered" condition of the final front—as well as the total number of individuals in the front, see Fig. 12b—it does not seem likely that more progress can be made. A second check could be to do a localised small-scale Monte Carlo run around each Pareto individual, with, for instance, a 5% variation of the original range of the design parameters. Having done this, only showed a marginal shift of the front, so with this method it is as far as we can go. It is curious to note, though, that one individual of the initial front is ahead of the final front, so the conclusions are not black and white. It would be good to keep track of all fronts and compose a final Pareto front from all individual fronts. In that respect, we will include *elitism* for the remainder of the runs, *i.e.,* the best individual for each objective is carried over to the next generation, to avoid losing out on potentially good individuals.

Figure 13 shows the same plots, but now for an initial population size of 200. Apart from the much larger number of individuals in the Pareto fronts, the final front is more stretched out, and at the lower right corner, it is also more rotated towards the origin. In the mid-region, the values seem to be more or less the same, but at the edges a better performance in the individual objectives is found. In conclusion, for the number of design parameters, a population size of 50 seems on the low side.

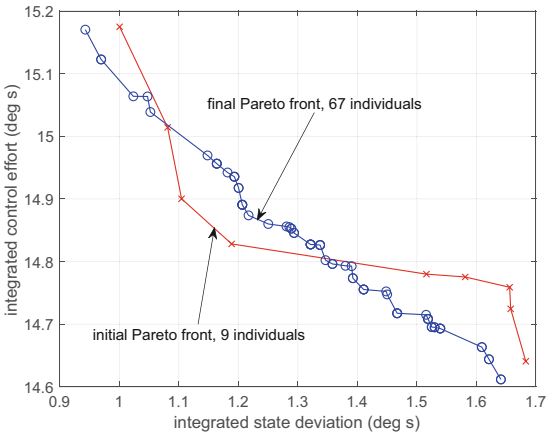

Fig. 14 Multi-objective optimisation, including off-diagonal elements in $\mathbf{T_p}$, rigid launcher. Initial population size 200 individuals

However, for mid-level performance individuals, it is sufficient. The advantage is, of course, that with a smaller initial population (and correspondingly a smaller number of crossovers and mutations), the CPU load of the algorithm is far less.

One additional test is done, to verify whether off-diagonal elements in one (or both) of the weighting matrices has an advantage. Since there are 10 more design parameters per matrix, we decided to extend only $\mathbf{T_p}$, as the integral gains already have a (constant) initial value. The range of the off-diagonal elements is taken as $\pm 10\%$ of the corresponding diagonal element in that row. Any $\mathbf{T_p}$ that is not positive semi-definite, taken as a constraint, is penalised with a very large objective value and effectively "killed off." Doing the run, with an initial population of 200 to account for the 25 design parameters, resulted in the plot of Fig. 14a. Some interesting aspects came forward. Despite the large variation in performance in the initial population, the converged Pareto front only occupies a small region in the objective space. The integrated state deviation has decreased to quite low values, albeit at the expense of a slightly larger control effort. The final Pareto front could have progressed more, as is evident from the initial Pareto front, that intersects with the final one.[8] The conclusion is that including off-diagonal elements may be beneficial and should be studied in more detail. Because of the different focus of this chapter, unfortunately we have to leave it at that.

[8] These results show that the *elitism* operator should not be limited to keeping only the best values for each objective, as has been implemented here, but should potentially include the complete Pareto front. A good algorithm to do so has been proposed by Tan et al. (2003), involving Tabu search. This method keeps an archive of the latest Pareto individuals, and in each successive generation the current Pareto front is removed from the population to maintain search diversity and added to the archive. From this archive, the newly dominated individuals are removed such that the absolute best Pareto front is maintained. It is possible to insert a few Pareto individuals back into the population, *e.g.,* every fifth generation, as it may help convergence to the (global) optimum.

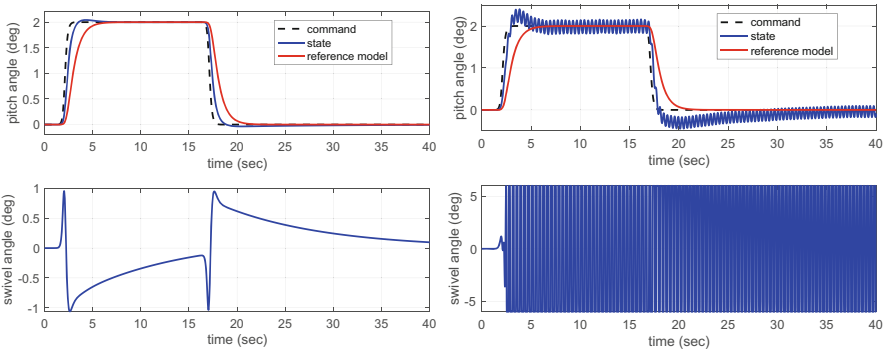

Fig. 15 Controller designed for a rigid body applied to one with engine dynamics. Controller frequency 100 Hz. (**a**) Rigid-body response. (**b**) Rigid body with engine dynamics

So far, we have been studying the response of the rigid launcher. Let us now include engine dynamics and see how this changes the picture. In principle, we would expect that if the engine dynamics is sufficiently fast—which should be the case with $\omega_e = 50$ rad/s—adding the dynamics will not change the response. However, upon inspection, selecting a Pareto individual from the final front with small state deviation and slightly larger control effort (such that engine dynamics will be excited a bit more), this is not the case. For the rigid system, the response is smooth (Fig. 15a), with performance indices given by $\sum_{\theta_{err}} = 1.73°\text{s}$, $\sum_{\varepsilon_T} = 14.31°\text{s}$, $F_\theta = 1.28°\text{s}$, and $F_{\varepsilon_T} = 3.50°$. The value of the oscillation indices is mainly driven by the discrete changes in the system. Changing to a rigid launcher with engine dynamics, the response deteriorates significantly, with performance indices of $\sum_{\theta_{err}} = 6.09°\text{s}$, $\sum_{\varepsilon_T} = 211.86°\text{s}$, $F_\theta = 2.78°\text{s}$, and $F_{\varepsilon_T} = 126.37°$. Looking at the corresponding response in Fig. 15b, the high oscillation index means that the system is in a high-frequency bang–bang state.

The SAC system is in principle a high-gain system, which will amplify the noise that enters the system. From the theoretical model of a rigid launcher with engine dynamics (Eq. (B.2)), we find there is a (strong) coupling between the engine and the angle of attack and, through the aerodynamics, the pitch rate and thus pitch angle. Having fast dynamics in the system requires a high controller frequency, as was also confirmed by Messer et al. (1994). They found, for their experimental setup of a suspended mass system, that for successful realisation of the controller a sampling frequency 40–80 times the Nyquist frequency is required. Going back to the results we got, it could lead to two changes in the approach: the control system

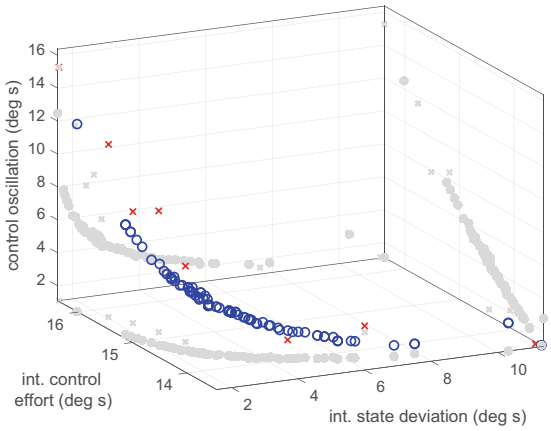

Fig. 16 Pareto front for three-objective optimisation, rigid launcher with engine dynamics. Initial population size 100 individuals

has to be re-optimised, but now with engine dynamics included, or the frequency of the controller is significantly increased, and then the system is re-optimised. Being able to find a good design with controller frequency set to 100 Hz would be the best, of course, in terms of on-board computer load. Unfortunately, a proper design could not be found. Thus, we will increase the frequency to 500 Hz and add the swivel oscillation index to the objectives.

The results of the three-objective optimisation are shown in Fig. 16. What is immediately clear is that there is an almost linear relationship between the integrated control effort and the control oscillation. That means that in principle we can do with one of the two, most notably the control oscillation, because lowering the oscillation seems to lower the control effort as well. This was indeed confirmed by doing a two-objective optimisation with $\sum_{\theta_{err}}$ and F_{ε_T}, which gave almost the same final Pareto front.[9]

From the obtained results, we will now select a design with a low oscillation index, which, unfortunately, will give us a large pitch-angle deviation. In a way this makes sense because a slower control response would induce smaller oscillations that are then less amplified. But a slower response would also yield a larger state deviation. Running the design does indeed show that the oscillations have almost disappeared. The optimisation process has thus led to a conceptual design of the control system that forms a good basis for further refinement. To show the final result, we will run the simulation once more, with the same settings for the controller, but now also including the first and second flexible modes. The result can be seen in Fig. 17.

Figure 17a shows the pitch-angle response, which is on the slow side, as expected. Even though the reference model tracks the command very well (in part

[9] The alternative of keeping the integrated control effort as objective was not as effective to reduce the oscillation index, though.

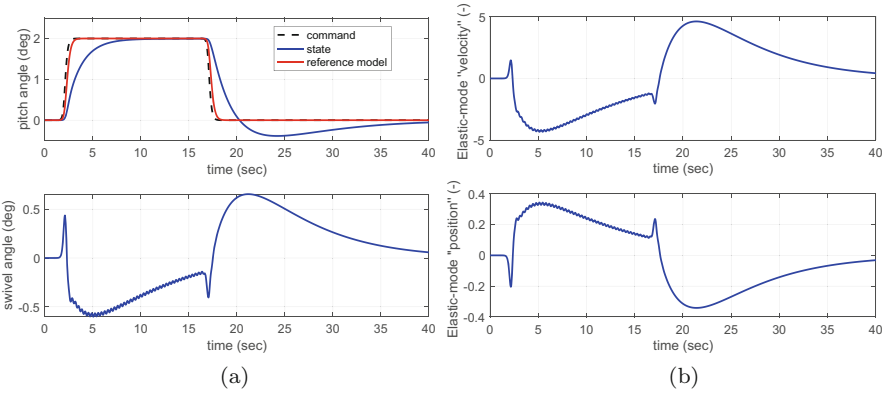

Fig. 17 Final controller design applied to flexible launcher with engine dynamics. (**a**) Rigid-body response. (**b**) First flexible mode

due to the high frequency), the plant response lags behind due to the lower swivel command. Because the swivel oscillations are virtually absent, also the flexible modes are not more excited than what is due to the rigid-body coupling (Fig. 17b). The corresponding performance indices are $\sum_{\dot{\eta}_{f,1}} = 94.7\,\text{s}$, $\sum_{\eta_{f,1}} = 7.4\,\text{s}$, $F_{\dot{\eta}_{f,1}} = 5.1\,\text{s}$, and $F_{\eta_{f,1}} = 0.6\,\text{s}$.[10]

6 Concluding Remarks

In this chapter, a methodology has been presented to design and analyse control systems with multiple design parameters and possibly conflicting objectives. These objectives have been formulated in a numerical way, *i.e.,* the integrated state deviation, integrated control effort, and an oscillation index using a moving-average technique. Because of such a formulation, numerical optimisation can be applied to go through a large number of designs and converge towards an optimum. As an example of the methodology, an evolutionary algorithm has been applied to the design of a simple adaptive control system.

Design space exploration helps to establish the bounds on the independent variables. Too large bounds can lead to a random behaviour of the optimisation algorithm, and with too narrow bounds the algorithm may miss the global optimum

[10] For comparison, if the simulation with results shown in Fig. 15b would be repeated for a flexible launcher, the indices would be $\sum_{\dot{\eta}_{f,1}} = 774.5\,\text{s}$, $\sum_{\eta_{f,1}} = 75.2\,\text{s}$, $F_{\dot{\eta}_{f,1}} = 650.1\,\text{s}$, and $F_{\eta_{f,1}} = 53.5\,\text{s}$. These very high values indicate severe deflections/vibrations, which could lead to hardware damage or even a complete breakup of the launcher.

and get stuck in a local one. The first step in the design process was to use single-objective optimisation. This led to a small integrated state deviation, but without guarantee that the control effort would be low. Adding this objective and executing the next step in the design process gave a Pareto front with converged close-to-optimal solutions. Finally, by adding also a control oscillation index as third objective, an attempt was made to lower the oscillatory behaviour of the control. It was observed that the same could be achieved by using only the integrated state deviation and the oscillation index.

In terms of controller performance, the results worked well enough for the rigid launcher. However, when engine dynamics was added, the same controller could not stabilise the system, and a strong oscillatory behaviour was the result. This could be solved by increasing the controller frequency and redoing the optimisation by including the aforementioned oscillation index. The final design was practically oscillation free and worked also well on the flexible launcher.

The introduced control system design methodology was shown to work well in a conceptual design environment, where baseline controllers (with multiple design parameters and thus quite a number of design degrees of freedom) can be quickly analysed, and their performance improved. Depending on the complexity of the design space, convergence will sometimes quickly come to a stop. Local refinement by using a limited Monte Carlo analysis around the Pareto solutions could serve as a confirmation for convergence. To account for the random effect, it is advised to always redo the optimisation with multiple seeds to initialise the random generator.

As recommendations for future work, one could think of confirming the versatility of this methodology by designing different control systems, possibly using different (and more complex) design criteria. Because the evolutionary algorithm was applied without a proper trade-off, other (global) optimisation techniques could be tried to improve the efficiency of the methodology.

Zooming in on the application at hand, it should be verified what the computational load means for an on-board implementation. Requiring a controller frequency of 100 or even 500 Hz is perhaps not a trivial requirement. For the flexible launcher, it would be interesting to study whether the same controller compared to one for the rigid launcher comes out of the optimisation process and whether the use of a different reference model may lead to an improved performance. Finally, the controlled flight from launch to payload separation should be verified to ensure that the controller always meets the performance requirements, taking transient effects due to the rapidly changing flight conditions, aerodynamic properties, and depletion of the fuel and oxidiser tanks into account. In principle, the adaptive controller should be able to adapt to those changes, but re-optimising the controller parameters for different conditions and using some form of interpolation in between design points may ultimately yield the best possible performance.

Appendix A: Pac Astro Mass Properties and Geometry

Tables A.1 and A.2 show the mechanical properties and additional masses of the launch vehicle model.

Table A.1 Mechanical properties of the launch vehicle structural model

Section	End coordinate [m]	Area [m^2]	Thickness [mm]	Moment of inertia [m^4]	Mass [kg]	Young's Modulus [GPa]	Density [kg/m^3]
Aft stage 1	3.07	3.93e−3	0.69	1.63e−3	30.46	72.4	2740
Fuel 1	6.03	1.51e−2	2.64	6.28e−3	230.91	73.8	2710
Intertank 1	8.63	1.78e−2	3.10	7.37e−3	102.69	72.4	2740
LOX 1	14.36	1.98e−2	3.46	8.23e−3	546.42	73.8	2710
Forward stage 1	16.79	1.48e−2	2.59	6.16e−3	67.30	72.4	2740
Aft stage 2	17.91	1.48e−2	2.59	6.16e−3	67.30	72.4	2740
Fuel 2	18.30	9.45e−3	1.65	3.92e−3	16.29	73.8	2710
Intertank 2	19.87	1.28e−2	2.24	5.33e−3	55.21	72.4	2740
LOX 2	20.96	1.03e−2	1.79	4.27e−3	49.62	73.8	2710
Forward stage 2	21.97	9.16e−3	1.60	3.81e−3	25.37	72.4	2740
Fairing	22.97	8.36e−3	1.46	3.47e−3	22.70	113	4430
Frustrum	25.58	7.04e−3	1.23	2.93e−3	53.54	113	4430
Nose	25.77						

Table A.2 Additional masses of the launch vehicle model (excluding fuel masses)

	Stage 1		Stage 2	
Subsystem	Mass [kg]	Location [m]	Mass [kg]	Location [m]
Engine	225	1.54	60	16.79
Thrust structures	55	2.20	20	21.46
Gimbal system	80	2.20	20	17.35
Pressurant	130	7.50	30	19.87
Valves and lines	130	7.50	50	19.00
GNC electronics			40	21.97
Payload adapter			20	22.47
Payload			225	22.97

Appendix B: State-space Matrices

The rigid-body sub-matrices (including coupling terms), $\mathbf{A_{RR}}$, $\mathbf{A_{RE}}$, and $\mathbf{A_{RF}}$ are given by

$$\mathbf{A_{RR}} = \begin{bmatrix} -\frac{C_{N_\alpha}\bar{q}S_{ref}}{mu_0} & -\frac{g_d \sin\theta_0}{u_0} & \frac{C_{N_q}\bar{q}S_{ref}}{mu_0}+1 \\ 0 & 0 & 1 \\ \frac{C_{m_\alpha}\bar{q}S_{ref}d_{ref}}{I_{yy}} & 0 & \frac{C_{m_q}\bar{q}S_{ref}d_{ref}}{I_{yy}} \end{bmatrix} \tag{B.1}$$

$$\mathbf{A_{RE}} = \begin{bmatrix} \frac{m_e \Delta L_e}{mu_0} & 0 & \frac{T}{mu_0} \\ 0 & 0 & 0 \\ \frac{m_e L_e \Delta L_e + I_e}{I_{yy}} & 0 & \frac{L_e T}{I_{yy}} \end{bmatrix} \tag{B.2}$$

$$\mathbf{A_{RF}} = \begin{bmatrix} a_{\alpha,\dot{\eta}_1} & a_{\alpha,\eta_1} & \dots & a_{\alpha,\dot{\eta}_N} & a_{\alpha,\eta_N} \\ 0 & 0 & \dots & 0 & 0 \\ a_{q,\dot{\eta}_1} & a_{q,\eta_1} & \dots & a_{q,\dot{\eta}_N} & a_{q,\eta_N} \end{bmatrix} \tag{B.3}$$

with, for $i = 1, \dots, n_f$:

$$a_{\alpha,\dot{\eta}_i} = -C_{N_{\dot{\eta}_i}}\bar{q}S_{ref} \qquad a_{\alpha,\eta_i} = -\frac{C_{N_{\eta_i}}\bar{q}S_{ref} - T\sigma_i(x_e)}{mu_0}$$

$$a_{q,\dot{\eta}_i} = \frac{C_{q,\eta_i}\bar{q}S_{ref}d_{ref}}{I_{yy}} \qquad a_{q,\eta_i} = \frac{C_{m_{\eta_i}}\bar{q}S_{ref}d_{ref} - L_e T\sigma_i(x_e) - T\phi_i(x_e)}{I_{yy}}$$

In the above equations, m and I_{yy} are the (current) mass and moment of inertia of the launcher, m_e and I_e are the mass and moment of inertia of the engine, and $\Delta L_{cm,e}$ is the distance from gimbal point to centre of mass of the engine. C_{N_α} and C_{N_q} are the normal force gradients with respect to α and q, and C_{m_α} and C_{m_q} are the corresponding pitch-moment gradients. Due to the bending of the launcher frame, local aerodynamic force and moment effects are introduced through the gradients $C_{N_{\eta_i}}$, $C_{N_{\dot{\eta}_i}}$, C_{q,η_i}, and $C_{m_{\eta_i}}$, and the details of which are provided by Mooij and Gransden (2016). Finally, $\phi_i(x)$ and $\sigma_i(x)$ are the modal-mass normalised i^{th} bending shape and slope at location x, in this case the engine location x_e.

The engine is modelled as third-order transfer function, with input parameters ω_e, ζ_e, and K_e, which are the natural frequency and damping of the engine dynamics and an amplification gain, respectively. The 3×3 engine sub-matrix, $\mathbf{A_{EE}}$, is defined to be

$$\mathbf{A_{EE}} = \begin{bmatrix} -2\zeta_e\omega_e & -\omega_e^2 & -K_e\omega_e^2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{B.4}$$

where the corresponding coupling matrices are zero, *i.e.,* $\mathbf{A_{ER}} = \mathbf{A_{EF}} = \mathbf{0}$.

Each bending motion depends on the generalised force for that specific motion. This generalised force is found by multiplying all the external loads with the eigenvector of that mode. As before, the external loads are a function of the bending motion and the position along the vehicle. Note that the subscripts i and j below both indicate a flexible mode, up to the maximum of n_f. So, for $\mathbf{A_{FR}}$, $\mathbf{A_{FE}}$ and $\mathbf{A_{FF}}$, we have

$$\mathbf{A_{FR}} = \begin{bmatrix} a_{\dot{\eta}_1,\alpha} & a_{\dot{\eta}_1,\theta} & a_{\dot{\eta}_1,q} \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ a_{\dot{\eta}_{n_f},\alpha} & a_{\dot{\eta}_N,\theta} & a_{\dot{\eta}_{n_f},q} \\ 0 & 0 & 0 \end{bmatrix} \tag{B.5}$$

with

$$a_{\dot{\eta}_i,\alpha} = -\bar{q}S_{ref}\int_0^{L_{tot}} C'_{N_\alpha}\phi_i(x)\mathrm{d}x \qquad a_{\dot{\eta}_i,\theta} = -g_d\sin\theta_0\int_0^{L_{tot}} \phi_i(x)m(x)\mathrm{d}x$$

$$a_{\dot{\eta}_i,q} = -\frac{\bar{q}S_{ref}}{u_0}\int_0^{L_{tot}} (x - x_{cm})C'_{N_\alpha}\phi_i(x)\mathrm{d}x$$

$$\mathbf{A_{FE}} = \begin{bmatrix} a_{\dot{\eta}_1,\ddot{\varepsilon}_T} & 0 & a_{\dot{\eta}_1,\varepsilon_T} \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ a_{\dot{\eta}_{n_f},\ddot{\varepsilon}_T} & 0 & a_{\dot{\eta}_{n_f},\varepsilon_T} \\ 0 & 0 & 0 \end{bmatrix} \tag{B.6}$$

with

$$a_{\dot{\eta}_i,\ddot{\varepsilon}_T} = m_e\Delta L_{cm,e}\phi_i(x_e) + I_e\sigma_i(x_e) \qquad a_{\dot{\eta}_i,\varepsilon_T} = T\phi_i(x_e)$$

$$\mathbf{A_{FF}} = \begin{bmatrix} a_{\dot{\eta}_1,\dot{\eta}_1} & a_{\dot{\eta}_1,\eta_1} & \dots & a_{\dot{\eta}_1,\dot{\eta}_{n_f}} & a_{\dot{\eta}_1,\eta_{n_f}} \\ a_{\eta_1,\dot{\eta}_1} & a_{\eta_1,\eta_1} & \dots & a_{\eta_1,\dot{\eta}_{n_f}} & a_{\eta_1,\eta_{n_f}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{\dot{\eta}_{n_f},\dot{\eta}_1} & a_{\dot{\eta}_{n_f},\eta_1} & \dots & a_{\dot{\eta}_{n_f},\dot{\eta}_{n_f}} & a_{\dot{\eta}_N,\eta_N} \\ a_{\eta_{n_f},\dot{\eta}_1} & a_{\eta_{n_f},\eta_1} & \dots & a_{\eta_{n_f},\dot{\eta}_{n_f}} & a_{\eta_{n_f},\eta_{n_f}} \end{bmatrix} \tag{B.7}$$

with, for $i \neq j$:

$$a_{\dot{\eta}_i,\dot{\eta}_j} = -\frac{\bar{q}S_{ref}}{u_0}\int_0^{L_{tot}} \phi_i(x) C'_{N_\alpha} \phi_j(x)\mathrm{d}x$$

$$a_{\dot{\eta}_i,\eta_j} = -\bar{q}S_{ref}\int_0^{L_{tot}} \phi_i(x) C'_{N_\alpha} \sigma_j(x)\mathrm{d}x - T\phi_i(x_e)\sigma_j(x_e)$$

$$a_{\eta_i,\dot{\eta}_j} = a_{\eta_i,\eta_j} = 0$$

and for $i = j$

$$a_{\dot{\eta}_i,\dot{\eta}_i} = a_{\dot{\eta}_i,\dot{\eta}_i} - 2\zeta_{f,i}\omega_{f,i}^2 \qquad a_{\dot{\eta}_i,\eta_i} = a_{\dot{\eta}_i,\eta_i} - \omega_{f,i}^2$$

$$a_{\eta_i,\dot{\eta}_i} = 1 \qquad a_{\eta_i,\eta_i} = 0$$

Lastly, to complete the model description, the components of **B** are stated:

$$\mathbf{B_R} = \mathbf{B_F} = \mathbf{0} \tag{B.8}$$

and

$$\mathbf{B_E} = \begin{pmatrix} K_e\omega_e^2 \\ 0 \\ 0 \end{pmatrix} \tag{B.9}$$

References

Baker, J.E., "Reducing Bias and Inefficiency in the Selection Algorithm", *Proceedings of the Second International Conference on Genetic Algorithms and their Application*, pp. 14–21, Hillsdale, New Jersey, 1987.

Barkana, I., "Classical and Simple Adaptive Control for Non-Minimum Phase Autopilot Design", *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 4, pp. 631–638, July-August 2004.

Barkana, I., "Parallel Feedforward and Simple Adaptive Control of Flexible Structures: First-Order Pole Instead of Collocated Velocity Sensors?", *Journal of Aerospace Engineering*, Vol. 29, Issue 2, March 2016.

Cvetković, D., "Evolutionary multi-objective decision support systems for conceptual design", PhD thesis, University of Plymouth, 2000.

Fleeter, R., Mcloughlin, F. and Mills, R., "A Low-Cost Expendable Launch Vehicle for 500-Pound Class Satellites", Marketing brochure, PacAstro, Herndon, VA, 26 May 1992.

Fleming, P.J. and Purshouse, R.C., "Genetic algorithms in control system engineering", Research Report no. 789, Department of Automatic Control and Systems Engineering, University of Sheffield, May 2001.

Fonseca, C.M.M. de, "Multiobjective Genetic Algorithms with Applications to Control Engineering Problems", PhD thesis, Department of Automatic Control and Systems Engineering, University of Sheffield, UK, 1995.

Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., 1989.

Gransden, D.I. and Mooij, E., "Control Recovery of a Satellite with Flexible Appendages after Space Debris Impact", AIAA-2018-2099, *AIAA SciTech Forum, Guidance, Navigation, and Control Conference*, Kissimmee, FL, 8–12 January 2018.

Holland, J.H., *Adaptation in natural and artificial systems*, The University of Michigan Press, Ann Arbor, MI, 1975.

Huang, Y., Pool, D.M., Stroosma, O. and Chu, Q., "Long-Stroke Hydraulic Robot Motion Control with Incremental Nonlinear Dynamic Inversion", *IEEE/ASME Transactions on Mechatronics*, Vol. 24, No. 1, February 2019, pp. 3044–314.

Kaufman, H., Barkana, I. and Sobel, K., *Direct adaptive control algorithms: Theory and applications*, Second edition, Springer-Verlag, New York, 1998.

Mehiel, E.A. and Balas, M.J., "Adaptive Control for a Deployable Optical Telescope", AIAA-2004-5222, From: *AIAA Guidance, Navigation, and Control Conference*, Providence, RI, August 16–19, 2004.

Menon, P.K., Yousefpor, M., Lam, T. and Steinberg, M.L., "Nonlinear flight control system synthesis using genetic programming", AIAA-1995-3224, *AIAA Guidance, Navigation, and Control Conference*, Baltimore, MD, August 7–10, 1995.

Messer, R.S., Haftka, R.T. and Cudney, H.H., "Cost of Model Reference Adaptive Control: analysis, experiments, and optimization", *Journal of Guidance, Control and Dynamics*, vol. 17, no. 5, pp. 975–982, Sep.–Oct. 1994.

Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, third edition, 1996.

Mooij, E., "Simple Adaptive Bank-Reversal Control for Non-linear Winged Re-entry Vehicles", *Mathematics In Engineering, Science And Aerospace*, MESA, Vol. 9, No. 1, 2018, pp. 85–110.

Mooij, E., and Gransden, D. I., "The Impact of Aeroelastic Effects on the controllability of Conventional Launch Vehicles", *Proceedings of the* 67^{th} *IAC Conference*, Guadalajara, Mexico, September, 2016.

Rolland Collette, J.G., "Analysis and Design of Space Vehicle Flight Control Systems, Volume XI, Component Dynamics", NASA CR-830, 1967.

Schwanz, R.C. and Cerra, J.J., "Dynamic modeling uncertainty affecting control system design", AIAA-1984-1057, From: *AIAA Dynamics Specialists Conference*, Palm Springs, CA, May 17–18, 1984.

Smeur, E.J.J., Chu, Q. and De Croon, G.C.H.E., "Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles", *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 3, March 2016, pp. 450–460.

Sutton, G.P. and Biblarz, O., *Rocket Propulsion Elements*, 9^{th} edition, John Wiley & Sons, Inc., 2017.

Tan, K.C., Khor, E.F., Lee, T.H., and Yang, Y.J., "A Tabu-based Exploratory Evolutionary Algorithm for Multiobjective Optimisation", *Artificial Intelligence Review*, Vol. 19, pp. 231–260, 2003.

Tanaka, T. and Chuang, C.-H., "Scheduling of linear controllers for X-29 by neural network and genetic algorithm", AIAA-1995-3270, *AIAA Guidance, Navigation, and Control*, Baltimore, MD, August 7–10, 1995.

Zalzala, A.M.S. and Fleming, P.J. (Eds.), Genetic algorithms in engineering systems, IEE Control Engineering Series 55, London, 1997.