Real-Time Optimal Control Using TransWORHP and WORHP Zen

Matthias Knauer and Christof Büskens

Abstract In many industrial applications solutions of optimal control problems are used, where the need for low computational times outweighs the need for absolute optimality. For solutions of fully discretized optimal control problems we propose two methods to approximate the solutions of problems with modified parameter values in real-time by using sensitivity derivatives.

We use TransWORHP to transcribe an optimal control problem to a sparse nonlinear programming problem, which will be solved using our NLP solver WORHP. For this nominal solution sensitivity derivatives can be computed with respect to any system parameter using WORHP Zen. On NLP level, the sensitivity derivatives allow to perform correction steps for changes in the system parameters. This concept can be transferred to discretized optimal control problems using, e.g., the sensitivity derivatives of the boundary condition or of the discretized differential equations. The quality and applicability of both methods are illustrated by a trajectory planning problem in the context of the planar restricted problem of three bodies. In both methods the sensitivity derivatives can be used to give numerical validations of the theoretically expected convergence behaviour.

1 Introduction

Numerical solution methods for optimal control problems are used widely and successfully in many academic and industrial applications. The quality of a computed optimal solution depends on the used mathematical model and on the knowledge available at the start of the optimization process. If the solution is applied to operate a real system, perturbations will always occur. The initial state might be different than expected, or the environment might have changed since the solution

M. Knauer (✉) · C. Büskens
Universität Bremen, Zentrum für Technomathematik, Bremen, Germany
e-mail: knauer@math.uni-bremen.de; bueskens@math.uni-bremen.de

G. Fasano, J. D. Pintér (eds.), *Modeling and Optimization in Space Engineering*, Springer Optimization and Its Applications 144,
https://doi.org/10.1007/978-3-030-10501-3_9

was computed. Of course, an updated optimal control problem can be solved now to handle the new situation.

In many cases, however, the solution of the updated optimal control problem would take too much time, even if the old solution could be used as a good initial guess. Different strategies have been developed to generate feasible trajectories within limited computation time:

Feedback control. Based on the dynamic system, a Riccati controller can be used to compensate for occurring perturbations and bring the states back on track, see [11].

Model predictive control. By considering only a finite time horizon for the optimal control problem, computation times can be reduced. In this method, any type of changing constraints can be considered, see [8, 10].

Sensitivity updates. Only cheap matrix–vector multiplications are needed to update the solution using sensitivity derivatives, see [5].

In this chapter, we will propose two methods for sensitivity updates for fully discretized optimal control problems. Using an example we will discuss the applicability and quality of both methods. We use TransWORHP to solve optimal control problems and WORHP Zen to generate the accompanying sensitivity derivatives. Both tools were developed especially for the NLP solver WORHP.

2 Optimal Control

How do I have to operate a system to guide it from an initial state to a final state without overstressing? This question, which applies to a wide range of problems (from landing trajectories of spaceships to medication of patients), is the template for optimal control problems:

$$\begin{aligned} \min_{x,u,t_f} \ & \phi(x(t_f)) + \int_0^{t_f} f_0(x(t), u(t))\, dt \\ \text{s.t.} \quad & \dot{x}(t) = f(x(t), u(t)), \quad t \in [0; t_f] \\ & x(0) = x_0 \\ & x(t_f) = x_f \\ & g(x(t), u(t)) \leq 0, \quad t \in [0; t_f] \end{aligned} \tag{1}$$

Here, a control function $u \in C_p^0([0; t_f], \mathbb{R}^{n_u})$ and a state function $x \in C_p^1([0; t_f], \mathbb{R}^{n_x})$ have to be determined for a fixed or free final time t_f. The dynamic behaviour of the system is modelled using a function $f \in C^1(\mathbb{R}^{n_x} \times \mathbb{R}^{n_u}, \mathbb{R}^{n_x})$. $C^j(I, \mathbb{R}^n)$ denotes the class of j times continuously differentiable functions from a domain $I \subset \mathbb{R}^k$ to $\mathbb{R}^n$, and $C_p^j(I, \mathbb{R}^n) \subset C^{j-1}(I, \mathbb{R}^n)$ the class of j times piecewise continuously differentiable functions.

For simplicity, let's for now allow only completely given initial and final states x_0 resp. x_f. The control $u(t)$ and the state $x(t)$ have to comply with inequality constraints using the function $g \in C^1(\mathbb{R}^{n_x} \times \mathbb{R}^{n_u}, \mathbb{R}^{n_g})$.

The optimal solution $(x^\star(t), u^\star(t))$ has to minimize the objective function consisting of the final term $\phi \in C^1(\mathbb{R}^{n_x}; \mathbb{R})$ and the integral of $f_0 \in C^1(\mathbb{R}^{n_x}, \mathbb{R}^{n_u})$. Note that the integral term can always be omitted by introducing a new differential equation

$$\dot{x}_+(t) = f_0(x(t), u(t)), \quad x_+(0) = 0 \tag{2}$$

and using $\phi(x(t_f)) + x_+(t_f)$ in the objective. Hence, we can assume $f_0 \equiv 0$ in the following.

2.1 Discretization

The infinite dimensional optimal control problem (1) can be approximated by a nonlinear programming problem (NLP) consisting of a large, but only finite number of parameters. See [2] or also [13] for a comparison of different methods. The so-called direct methods replace the continuous time interval $[0; t_f]$ by discrete grid points

$$0 = t_1 \le t_2 \le \cdots \le t_\ell = t_f, \quad \ell \in \mathbb{N} \tag{3}$$

Likewise, the control function $u(t)$ and the state function $x(t)$ are replaced by vectors of discrete values

$$u = (u^1, \ldots, u^\ell)^T, \quad x = (x^1, \ldots, x^\ell)^T \tag{4}$$

where $u^i \approx u(t_i)$ and $x^i \approx x(t_i)$.

Finally, the trapezoidal method is applied to the system of ordinary differential equations in (1), resulting in the discretized version of the optimal control problem, where $h_i := t_{i+1} - t_i$, $i = 1, \ldots, \ell - 1$ and $f^i = f(x^i, u^i)$:

$$\begin{aligned} \min_{x,u,t_f} \ & \phi(x^\ell) \\ \text{s.t.} \quad & x^{i+1} = x^i + h_i \tfrac{f^i + f^{i+1}}{2}, \quad i = 1, \ldots, \ell - 1 \\ & x^1 = x_0 \\ & x^\ell = x_f \\ & g(x^i, u^i) \le 0, \quad i = 1, \ldots, \ell \end{aligned} \tag{5}$$

For full discretization the discretized variables x and u can be grouped to a vector of optimization variables $z = (x, u) \in \mathbb{R}^{(n_x + n_u)\cdot \ell}$. In case of a free final time, t_f is also inserted into the vector z. The (in-)equality constraints of (5) can be grouped together as well. Alternatively, for shooting methods the discretized variables x are

only considered in the vector of optimization variables z for selected shooting points and are computed using Runge–Kutta schemes elsewhere. In both cases, (5) was led back to an NLP with n optimization variables and m constraints. Using the function $F \in C^1(\mathbb{R}^n, \mathbb{R})$ for the objective, and collecting all m_e equality and $m - m_e$ inequality constraints in $G \in C^1(\mathbb{R}^n, \mathbb{R}^m)$, we get:

$$\begin{aligned} \min_{z \in \mathbb{R}^n} \ & F(z) \\ \text{s.t.} \ & G_i(z) = 0, \quad i = 1, \ldots, m_e \\ & G_i(z) \leq 0, \quad i = m_e + 1, \ldots, m \end{aligned} \tag{6}$$

2.2 Numerical Solution

Sequential quadratic programming methods (SQP) can be used to solve (6) efficiently. Starting from an initial guess they approximate the solution of the nonlinear problem by a sequence of solutions of quadratic problems (QP). The ESA NLP solver WORHP implements a sparse SQP method to solve systems with millions of variables under millions of constraints, see [6]. Obviously, also for smaller problems the sparsity of the derivative matrices can be exploited to keep computation times small. Fully discretized problems (5) naturally yield highly structured NLPs.

To solve an optimal control problem (1) the software library TransWORHP generates the finite dimensional problem (5) and uses WORHP to solve it, see [9]. Different transcription schemes are available in TransWORHP:

Full discretization. The user can currently choose between Euler's method, the trapezoidal rule and Hermite–Simpson.

Multiple shooting. All discretized control variables, but only selected state variables are optimized. This method can be used with any Runge–Kutta method or Runge–Kutta–Fehlberg method for integration.

Pseudospectral methods. The discretized control and state variables are interpreted as control points of a polynomial of high order resulting in smooth functions.

TransWORHP was especially designed to solve problems consisting of multiple phases. The user can define a set of optimal control problems and connect their discretized versions to one large NLP.

3 Parametric Nonlinear Programming

If the functions F and G in (6) additionally depend on a parameter $p \in \mathbb{N}^{n_p}$, a parametric nonlinear programming problem is formulated. After an optimal solution was found for a fixed $p = p_0$, it is known, which constraints are active or inactive.

Without restriction the inactive inequality constraints can be removed from the problem, and only $m_a \leq m$ active constraints G^a remain:

$$\begin{aligned} &\min_{z \in \mathbb{R}^n} \; F(z, p) \\ &\text{s.t.} \;\; G_i^a(z, p) = 0, \quad i = 1, \ldots, m_a \end{aligned} \tag{7}$$

For an optimal solution of (7) for a fixed $p = p_0$, the tool WORHP Zen can be used to compute sensitivity derivatives $\frac{dz}{dp}(p_0)$. The proof of the sensitivity theorem, as given, for example, by Fiacco et al. [7], shows how the sensitivity derivatives can be computed without measurable computational cost by solving this system:

$$\begin{pmatrix} \nabla_z^2 L & \nabla_z G^{a\,T} \\ \nabla_z G^a & 0 \end{pmatrix} \cdot \begin{pmatrix} \frac{dz}{dp}(p_0) \\ \frac{d\lambda}{dp}(p_0) \end{pmatrix} = - \begin{pmatrix} \nabla_{zp} L \\ \nabla_p G^a \end{pmatrix} \tag{8}$$

Here, the Lagrange multipliers $\lambda \in \mathbb{R}^{m_a}$ and the Lagrange function

$$L(z, \lambda, p) := F(z, p) + \lambda^T G^a(z, p) \tag{9}$$

are used. If perturbations occur, i.e. $p \neq p_0$, the sensitivity derivatives can be used for real-time approximations of the solution of a perturbed problem:

$$z^{[1]} := z(p_0) + \frac{dz}{dp}(p_0) \cdot (p - p_0) \tag{10}$$

After this *pre-correction step* the constraint $G_i^a(z^{[1]}) \leq 0$ might not be fulfilled anymore and $z^{[1]}$ has to be considered as an infeasible solution. However for a sequence of *post-correction steps*, we additionally consider linear perturbations $q \in \mathbb{N}^m$ in the constraints of (7):

$$\begin{aligned} &\min_{z \in \mathbb{R}^n} \; F(z, p) \\ &\text{s.t.} \;\; G_i^a(z, p) + q = 0, \quad i = 1, \ldots, m_a \end{aligned} \tag{11}$$

For $q = 0$ the problems (7) and (11) coincide. The measured violation of the constraints, q, can be reduced iteratively for $i = 1, 2, 3, \ldots$:

$$z^{[i+1]} := z^{[i]} + \frac{dz}{dq}(0) \cdot q \tag{12}$$

Please note that after solving (11) the sensitivity derivatives $\frac{dz}{dq}(0)$ can be computed as in (8) for a simpler right-hand side, as $\nabla_{zq} L = 0$ and $\nabla_q (G^a + q) = I$.

Some important properties of these correction steps are proven in [4].

Theorem 1 (Convergence of Repeated Correction Method) *Let the necessary and sufficient optimality conditions of first and second order be fulfilled, and* $F, G^a \in C^3$.

Then there exists $U(p_0)$, $v \in \mathrm{Ker}(\nabla_z G^a(z(p_0), p_0))$, $\|v\| = \mathcal{O}(\|p - p_0\|^2)$ *such that for all* $p \in U(p_0)$, $i = 2, 3, 4, \ldots$:

$$\begin{aligned} \|z(p) - z^{[i]}\| &= \|v\| + \mathcal{O}(\|p - p_0\|^3) \\ \|F(z(p), p) - F(z^{[i]}, p)\| &= \mathcal{O}(\|p - p_0\|^3) \\ \|G^a(z^{[i]}, p)\| &= \mathcal{O}(\|p - p_0\|^{i+1}) \end{aligned} \tag{13}$$

The last equation ensures the convergence to a feasible solution, as

$$\|G^a(z^{[\infty]}, p)\| = 0 \tag{14}$$

4 Real-Time Optimal Control

Sensitivity derivatives for parametric optimal control problems with a perturbation parameter p in any function of (1) can be straightforwardly determined, at least when considering the discretized version (5). Further, the results of Theorem 1 can be transferred to optimal control problems and the sensitivity derivatives can be used for iteratively reducing errors in boundary conditions or path constraints while preserving optimality [3]. After solving the fully discretized optimal control problem for a nominal value $p = p_0$ and preparing the sensitivity differentials $\frac{dz}{dp}(p_0)$, approximative solutions for perturbations $p \neq p_0$ can be computed in real-time.

For comprehensibility, we restrict the presentation of the following methods to perturbations in the initial point and want to ensure that boundary conditions in the final point hold.

4.1 Using Sensitivity Derivatives of Boundary Condition

This method can be applied to fully discretized optimal control problems as well as to shooting methods:

$$\begin{aligned} \min_{x,u,t_f} \ & \phi(x^\ell) \\ \text{s.t.} \quad & x^{j+1} = x^j + h_j \tfrac{f^j + f^{j+1}}{2}, \quad j = 1, \ldots, \ell - 1 \\ & x^1 = x_0 + p \\ & x^\ell = x_f + q \\ & g(x^j, u^j) \leq 0, \quad j = 1, \ldots, \ell \end{aligned} \tag{15}$$

Initialize iteration counter $i := 1$
Update control to compensate initial perturbation:

$$u^{[1]} := u(p_0) + \frac{du}{dp}(p_0) \cdot (p - p_0)$$

Compute state $x^{[1]}$ by integration using $u^{[1]}$
Measure final state deviation:

$$\Delta q^{[i]} := x^{[i],\ell} - x_f$$

while $\|\Delta q^{[i]}\|_\infty > \varepsilon$ **do**
  Update control:

$$u^{[i+1]} := u^{[i]} + \frac{du}{dq}(0) \cdot \Delta q^{[i]}$$

  Compute state $x^{[i+1]}$ by integration using $u^{[i+1]}$
  Update final state deviation $\Delta q^{[i+1]}$
  Update iteration counter $i \leftarrow i + 1$
end

Algorithm 1: Real-time optimal control using sensitivity derivatives of boundary condition

To handle perturbations p at the initial point in the pre-correction step, the sensitivity derivative $\frac{du}{dp}(p_0) \in \mathbb{R}^{(n_u \cdot \ell) \times n_x}$ is needed. To ensure feasibility of the final point in the post-correction steps, an additional perturbation q is introduced to compute the sensitivity derivative $\frac{du}{dq}(0) \in \mathbb{R}^{(n_u \cdot \ell) \times n_x}$.

Algorithm 1 uses the derivatives $\frac{du}{dp}(p_0)$ and $\frac{du}{dq}(0)$ to generate an approximative solution of the perturbed optimal control problem in real-time.

To integrate the state in Algorithm 1 starting from $x^{[i],1} = x_0 + p$, the trapezoidal method has to be implemented. However, this requires the solution of implicit equations.

4.2 Using Sensitivity Derivatives of Discretized Differential Equations

This method only applies to fully discretized optimal control problems:

$$\begin{aligned} \min_{x,u,t_f} \ & \phi(x^\ell) \\ \text{s.t.} \ & x^{j+1} = x^j + h_j \tfrac{f^j + f^{j+1}}{2} + q^j, \quad j = 1, \ldots, \ell - 1 \\ & x^1 = x_0 + p \\ & x^\ell = x_f \\ & g(x^j, u^j) \le 0, \quad j = 1, \ldots, \ell \end{aligned} \tag{16}$$

Initialize iteration counter $i := 1$
Update control and state to compensate initial perturbation:

$$u^{[1]} := u(p_0) + \frac{du}{dp}(p_0) \cdot (p - p_0)$$

$$x^{[1]} := x(p_0) + \frac{dx}{dp}(p_0) \cdot (p - p_0)$$

Measure deviations in discretized differential equations:

$$\Delta q^{[i]} := \left(x^{[i],j+1} - x^{[i],j} - h_j \frac{f^{[i],j} + f^{[i],j+1}}{2} \right)_{j=1,\dots,\ell-1}$$

while $\|\Delta q^{[i]}\|_\infty > \varepsilon$ **do**
    Update control and state:

$$u^{[i+1]} := u^{[i]} + \frac{du}{dq}(0) \cdot \Delta q^{[i]}$$

$$x^{[i+1]} := x^{[i]} + \frac{dx}{dq}(0) \cdot \Delta q^{[i]}$$

    Update deviation in discretized differential equations $\Delta q^{[i+1]}$
    Update iteration counter $i \leftarrow i + 1$
end

Algorithm 2: Real-time optimal control using sensitivity derivatives of discretized differential equations

To handle perturbations p at the initial point in the pre-correction step, the sensitivity derivatives $\frac{du}{dp}(p_0) \in \mathbb{R}^{(n_u \cdot \ell) \times n_x}$ and $\frac{dx}{dp}(p_0) \in \mathbb{R}^{(n_x \cdot \ell) \times n_x}$ are needed. After the pre-correction step the final point is still feasible, as the optimization variable x^ℓ is forced to equal x_f and hence the sensitivity of x^ℓ with respect to any perturbations is zero. However, the equations containing the discretized differential equations will not hold anymore. To enforce them in the post-correction steps, an additional perturbation $q = (q^j)_{j=1,\dots,\ell-1}$ is used to provide the sensitivity derivatives $\frac{du}{dq}(0) \in \mathbb{R}^{(n_u \cdot \ell) \times (n_x \cdot (\ell-1))}$ and $\frac{dx}{dq}(0) \in \mathbb{R}^{(n_x \cdot \ell) \times (n_x \cdot (\ell-1))}$.

Algorithm 2 uses the derivatives $\frac{du}{dp}(p_0)$, $\frac{dx}{dp}(p_0)$, $\frac{dx}{dq}(0)$ and $\frac{du}{dq}(0)$ to generate an approximative solution of the perturbed optimal control problem in real-time only using matrix–vector multiplications. We used the abbreviation $f^{[i],j} = f(x^{[i],j}, u^{[i],j})$. In case of a free final time t_f, the derivatives $\frac{dt_f}{dp}(p_0)$ and $\frac{dt_f}{dq}(0)$ have to be considered analogously for both methods. This is necessary, as the final time is expected to change for perturbed problems.

5 Numerical Results

The functionality of Algorithms 1 and 2 will be illustrated by the restricted problem of three bodies [1], which we expand to an optimal control problem.

5.1 Restricted Problem of Three Bodies

Two bodies with finite masses m_1 and m_2 are orbiting around their center of mass. The restricted problem of three bodies studies the motion of a test body with infinitesimal mass in the gravitational field of the masses m_1 and m_2. In particular the test body does not influence the other bodies.

The standard scaling of this problem is also suitable for optimization:

- The masses of the two bodies are scaled such that their sum is 1. Without restriction let $m_2 = \mu \leq \frac{1}{2}$. Then we get $m_1 = 1 - \mu$.
- The constant distance between the finite masses is $a = 1$.
- The time scale is chosen such that the gravitational constant is $G = 1$.
- The mean motion n of the masses depends on their period T and hence

$$n = \frac{2\pi}{T} = \sqrt{G\frac{(1-\mu)+\mu}{a^3}} = 1 \tag{17}$$

The coordinate system to describe the motion of the test body is defined such that the origin is located in the center of mass of the system, and the axes are rotating together with the masses in the x–y-plane such that the x-axis always goes through the masses m_1 and m_2. For simplicity, we will only consider the planar restricted problem of three bodies. The fixed positions of the masses m_1 and m_2 in the rotating coordinate system are $(x_1, 0)$ and $(x_2, 0)$, respectively. We will denote the distance of the test body to the mass m_i by

$$r_i = \sqrt{(x - x_i)^2 + y^2}, \quad i = 1, 2 \tag{18}$$

Then, the motion of the test body is subject to this system of second order differential equations:

$$\begin{aligned} \ddot{x} - 2\dot{y} - x &= -(1-\mu)\frac{x - x_1}{r_1^3} - \mu\frac{x - x_2}{r_2^3} \\ \ddot{y} + 2\dot{x} - y &= -(1-\mu)\frac{y - y_1}{r_1^3} - \mu\frac{y - y_2}{r_2^3} \end{aligned} \tag{19}$$

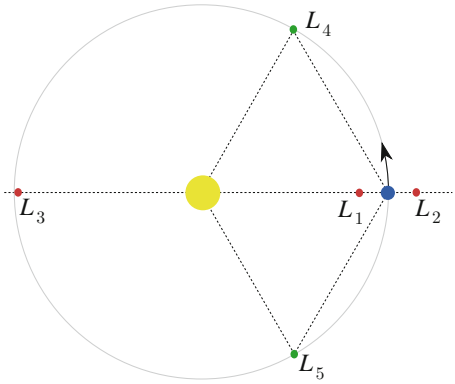

Figure 1 Lagrangian points L_1–L_5 in the restricted problem of three bodies

To ensure that the center of mass is in the origin, $x_1 = -\mu$ and $x_2 = 1 - \mu$ have to be fulfilled. The Lagrangian points L_i are the five stationary solutions of (19). L_1, L_2 and L_3 are located on the x-axis. L_4 and L_5 form equilateral triangles with the masses m_1 and m_2, as shown in Figure 1. If $\mu < \frac{1}{2} - \sqrt{\frac{23}{108}}$ it can be shown that L_4 and L_5 are stable in the linearized system.

As exemplary optimal control problem we will discuss the controlled motion of the test body.

5.2 *Optimal Control for a Spaceship*

As an illustrative example the task is to generate trajectories for a spaceship flying from a point p_0 close to L_5 to a point p_f close to L_4:

$$p_0 = (0.5, -0.866), \quad p_f = (0.5, 0.866) \tag{20}$$

The acceleration achieved by the thrusters is considered as the vector of control functions $u = (u_x, u_y)$ in the optimal control problem. Here, the control u_x is acting in the direction of x and the control u_y is acting in the direction of y.

The vector of state functions $x = (p_x, p_y, v_x, v_y)$ collects the positions in directions x and y and their velocities, which we have to introduce when writing (19) as a first order system. To generate a maneuver between two rest positions, we set

$$v_0 = (0, 0), \quad v_f = (0, 0) \tag{21}$$

In the notation of (1), let us define the trajectory planning problem as follows:

$$\begin{aligned}
\min_{x,u,t_f} \; & t_f + \omega \int_0^{t_f} u_x^2 + u_y^2 \, dt \\
\text{s.t.} \quad & \dot{p}_x = v_x \\
& \dot{p}_y = v_y \\
& \dot{v}_x = 2v_y + p_x - \frac{(1-\mu)(p_x - x_1)}{r_1^3} - \frac{\mu(p_x - x_2)}{r_2^3} + u_x \\
& \dot{v}_y = -2v_x + p_y - \frac{(1-\mu)p_y}{r_1^3} - \frac{\mu p_y}{r_2^3} + u_y \\
& (p_x, p_y)(0) = p_0 \\
& (p_x, p_y)(t_f) = p_f \\
& (v_x, v_y)(0) = v_0 \\
& (v_x, v_y)(t_f) = v_f
\end{aligned} \tag{22}$$

The objective is to minimize the process time t_f as well as the control input (or 'energy') as an integral term. For simplicity we use the weighting parameter $\omega = 1$. The mass of the smaller body is $\mu = 0.01$.

5.3 Reference Solution with TransWORHP

In Figure 2 the unperturbed solution is shown in blue for $\ell = 101$ discrete points. The task was to find a trajectory from p_0 to p_f. The spaceship can gain velocity by being attracted to the larger mass m_1, close to the origin. The resulting process time is $t_f = 2.46672$ and $E = 1.13076$ is a measure for the control input.

If the initial position is perturbed—in our example we modify the position in direction y by 0.1 and the velocity in the same direction by 0.1—the final position will not be reached if the same control is applied, as the red curve (perturbed integration) in Figure 2 shows. The slightly smaller initial distance to the masses accelerates the spaceship more than planned during optimization.

However, solving the optimal control problem from the perturbed initial position and velocity

$$\widetilde{p}_0 = (0.5, -0.866 + 0.1), \quad \widetilde{v}_0 = (0, 0 + 0.1) \tag{23}$$

we will fulfil the final condition again, as shown in green in Figure 2. As the distance from the perturbed initial point to the final point is shorter, and as the perturbed initial velocity points in the right direction, we get a faster process with $t_f = 2.22978$ and need less energy $E = 1.04937$. The controls for the unperturbed and perturbed solution are shown in Figure 3.

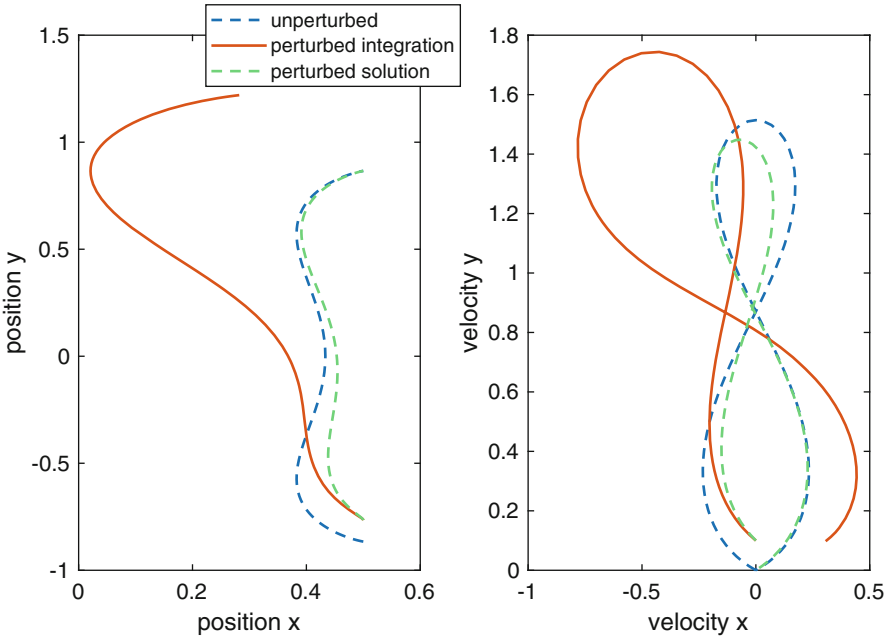

Figure 2 Position and velocity for the optimal solution from the unperturbed and the perturbed initial position

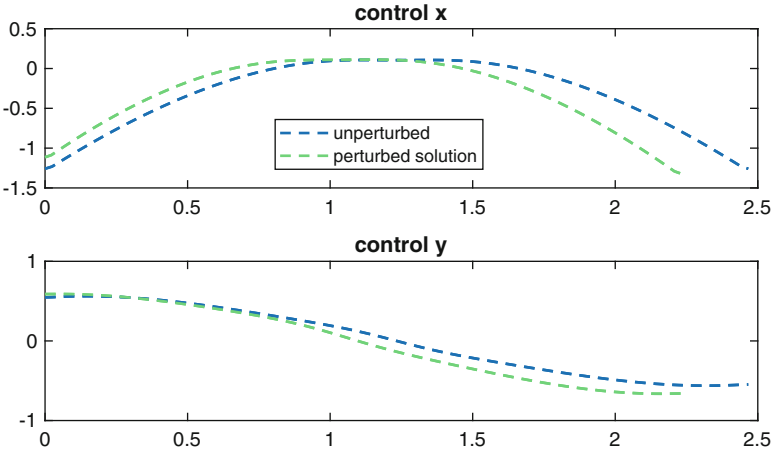

Figure 3 Controls for the optimal solution from the unperturbed and the perturbed initial position

All solutions for (22) have been computed with TransWORHP. Full discretization with the trapezoidal method requires $n = (n_x + n_u)\ell + 1 = 708$ variables in the NLP. Note that $n_x = 5$, as the integral term in the objective function, is implemented as an additional differential equation. One optimization variable is used for the free

Table 1 Dependency pattern for differential equations as used by TransWORHP

		States					Controls	
		p_x	p_y	v_x	v_y	E	u_x	u_y
Differential equations	$\dot{p}_x$			×				
	$\dot{p}_y$				×			
	$\dot{v}_x$	×	×		×		×	
	$\dot{v}_y$	×	×	×				×
	$\dot{E}$						×	×

final time. After optimization $m = n_x(\ell - 1) = 500$ equality constraints for the discretized differential equations have to be fulfilled.

The objective function only consists of two NLP variables. Hence the gradient only has two non-zero entries out of n (with the values 1 and ω). The Jacobian of the constraints could store up to $n \cdot m = 354{,}000$ entries. However, due to the discretization in (5) each constraint only depends on two adjacent points. Table 1 shows that each differential equation only depends on few states and controls. This pattern can be used to provide a Jacobian with only 3900 non-zero entries. We use finite differences to compute them (although it would also be possible analytically). The Hessian also takes advantage of the fact that constraints only depend on two adjacent points. Using finite differences again, we only have to compute 3536 out of $\frac{n(n+1)}{2} = 250{,}986$ entries.

We used a reasonable initial guess for the process time $t_f = 2.5$ and for the controls

$$u_x = 0.5 - 0.38t, \quad u_y = -0.6(t - 1.3)^2$$

and provided an initial guess for the states by integration. An optimal solution was found using TransWORHP in 10 NLP iterations with a precision of 10^{-6} for feasibility and optimality. The high level of sparsity allowed the computation in approximately 5 s on a standard pc using standard WORHP settings.

5.4 Real-Time Solution Using WORHP Zen

Instead of solving the optimal control problem again in the case of a perturbation, we will now apply the algorithms of Sect. 4 to approximate the optimal solution in less time.

In the method described in Sect. 4.1 the pre-correction step considering the measured perturbation in the initial position brings the spaceship closer to the final position, but still does not reach it. The final position in direction y is almost reached, whereas in direction x there is a big gap, see the yellow curves in Figure 4. After some post-correction steps, however, the desired final position is reached precisely (purple curves). In Table 2 the iterative behaviour of the objective function

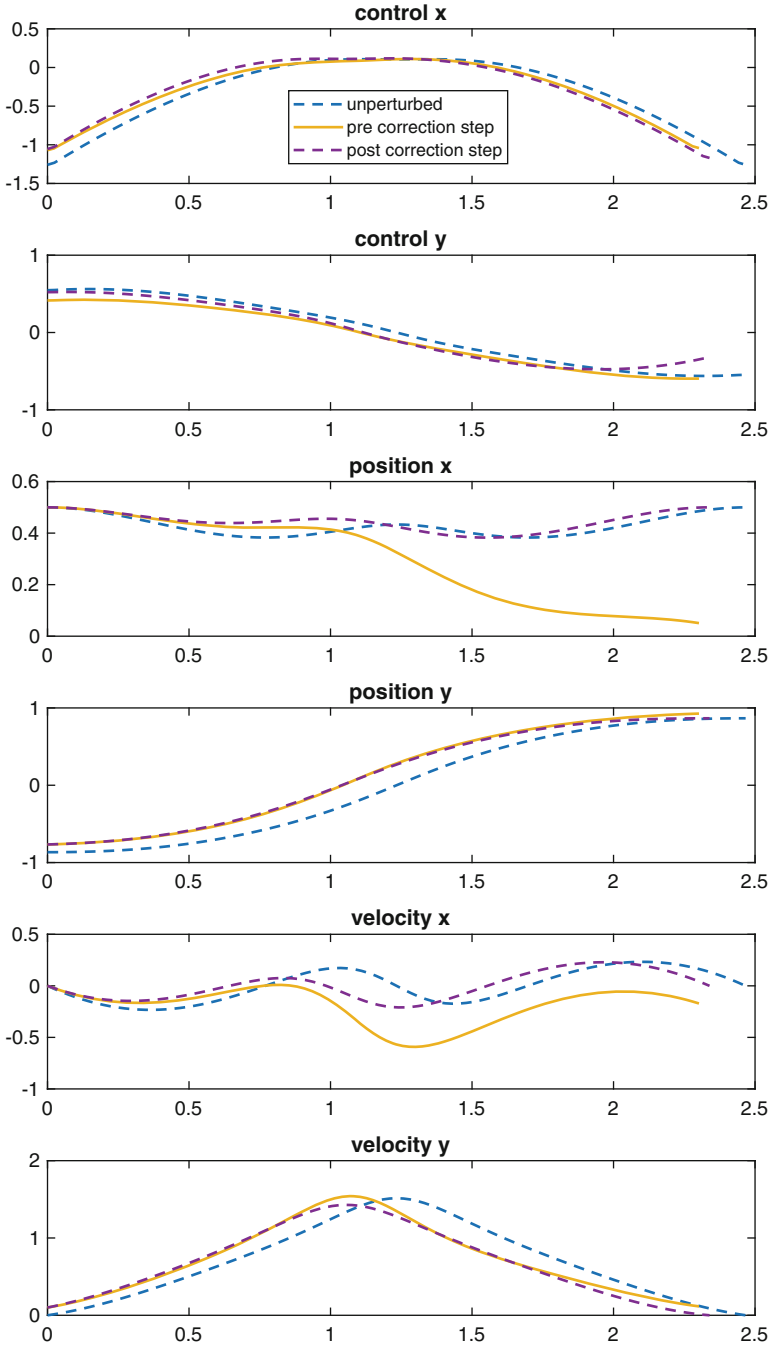

Figure 4 Real-time optimal control for the spaceship using sensitivity derivatives of boundary condition

Table 2 Process time t_f, energy term and final state deviation for real-time optimal control for the spaceship using sensitivity derivatives of boundary condition

Iteration	t_f	E	$\|\Delta q^{[i]}\|_\infty$
0	2.46672	1.13076	0.5256863
1	2.30186	0.77673	0.4980371
2	2.34467	1.19887	0.1383581
3	2.33899	0.83768	0.0231891
4	2.33976	0.84755	0.0030709
5	2.33917	0.84558	0.0015629
6	2.33919	0.84594	0.0001827
7	2.33917	0.84585	0.0000756
8	2.33917	0.84586	0.0000132
9	2.33917	0.84585	0.0000041
10	2.33916	0.84585	0.0000009

and the constraints during the correction steps is shown. The free process time t_f is also adjusted during the iteration. Starting with an initial velocity towards our target results in a shorter process time. Iteration 0 is the solution computed for the perturbed initial position. In iteration 1 the pre-correction step was applied, and the following iterations refer to the post-correction steps.

On the other side, the pre-correction step of the method described in Sect. 4.2 already reaches the final position and velocity exactly as shown in Figure 5. However, the updated controls and states do not fulfil the system of discretized differential equations anymore, as shown in Figure 6. This gap can be closed during the post-correction steps. In Figure 6 after the fourth post-correction step the deviation can't be distinguished from zero anymore.

The process time is also adjusted for this method. Comparing the results for both methods in Tables 2 and 3 we get a smaller process time for the second method, at a higher energy cost.

5.5 Comparison with Convergence Theorem

In Theorem 1 an exponential convergence rate for the constraints (13) was shown. This can be observed in the last column of Table 2 and in Figure 7 for the method described in Sect. 4.1. In every iteration, the measured deviation in the final point is reduced by the same magnitude.

Analogously, Table 3 and Figure 8 visualize the convergence rate for the method described in Sect. 4.2. A remarkable fact when using the post-correction steps: solutions can be found which hold the system of differential equations with higher accuracy than the original solution, where only 10^{-6} was demanded.

Furthermore, Theorem 1 states in (13) that the objective function maintains optimality depending on the deviation $\|p - p_0\|^3$. To show this numerically, we use

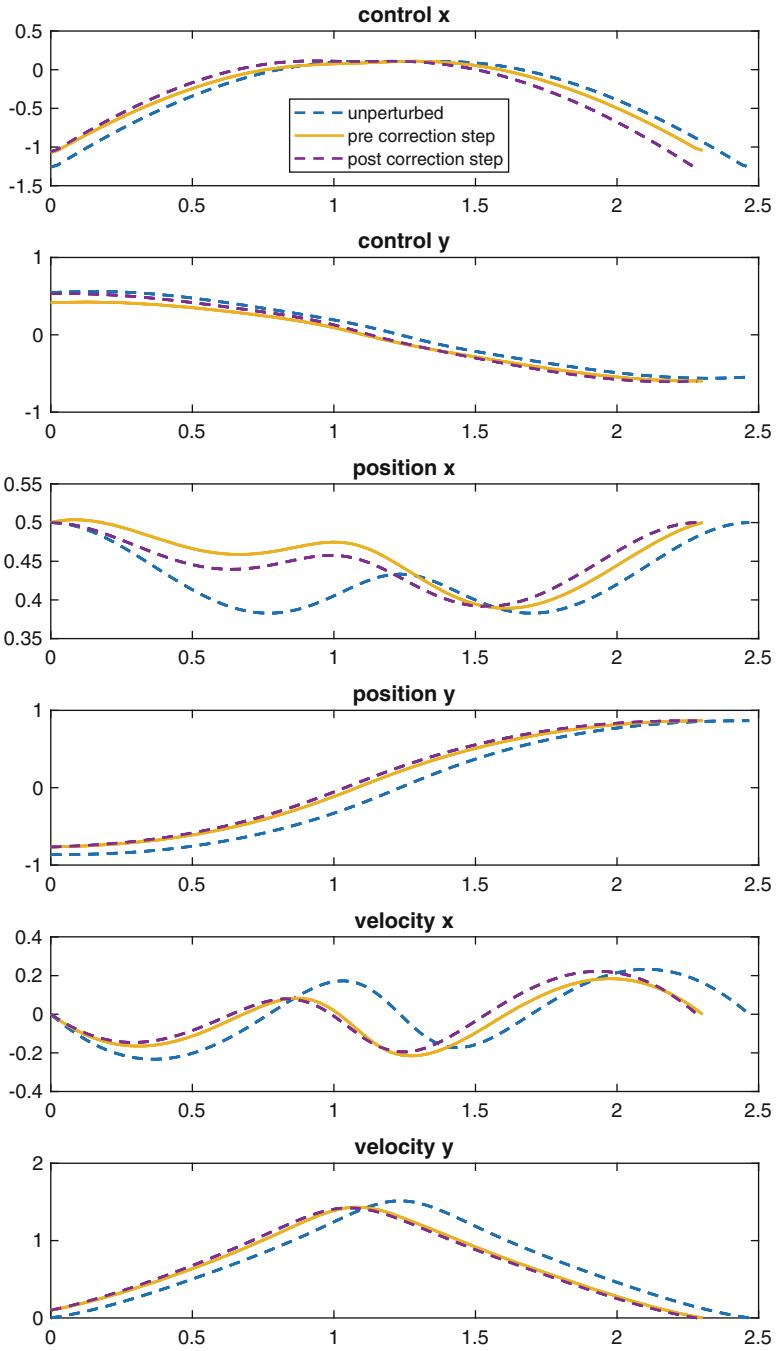

Figure 5 Real-time optimal control for the spaceship using sensitivity derivatives of discretized differential equation

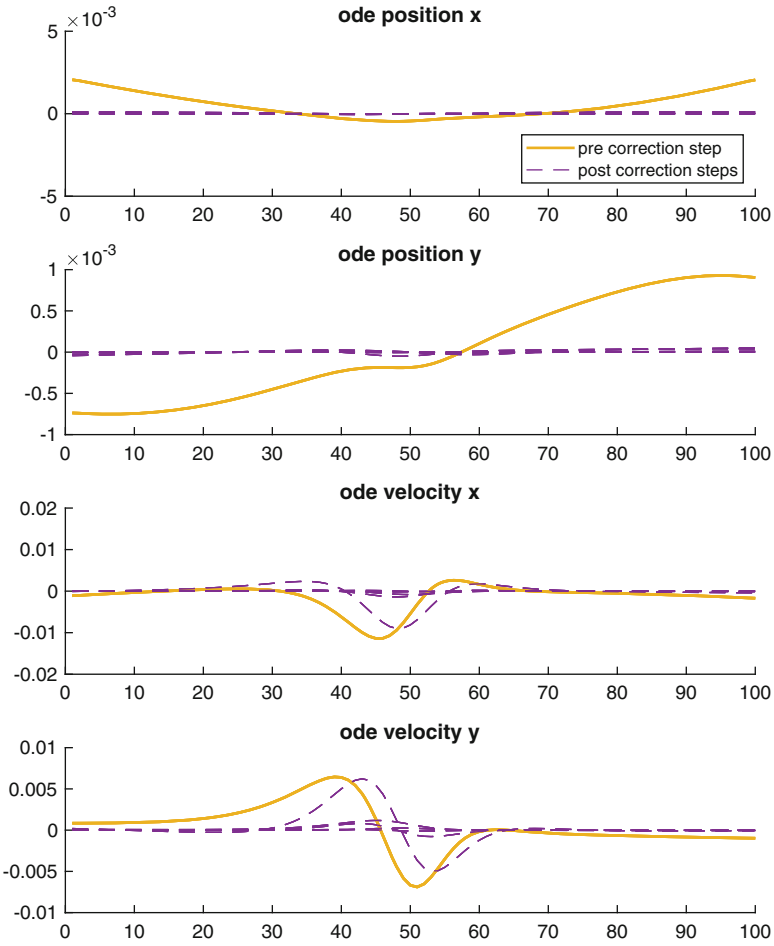

Figure 6 Deviation $\Delta q^{[i],j}$ in discretized differential equations for $j = 1, \ldots, \ell - 1$ ($\ell = 101$) during correction steps

the same unperturbed solution as in Sect. 5.3 together with the sensitivity derivatives and consider perturbations $\Delta y_0 \in [-0.4, 0.4]$ in the initial position of p_y.

For the method described in Sect. 4.1, Figure 9 shows that the feasible solution after the post-correction steps is a good approximation of the perturbed optimal solution for a large range of Δy_0. For the objective function (sum of process time and energy) a cubic behaviour was expected. However, the individual components of the objective function act differently.

As expected, smaller perturbations require less iterations. The number of iterations is reflected also directly in the computational time of 0.5–2 s using MATLAB. Obviously, the trapezoidal method takes the largest part of the high computational cost.

Table 3 Process time t_f, energy term and deviations in discretized differential equations for real-time optimal control for the spaceship using sensitivity derivatives of discretized differential equation

Iteration	t_f	E	$\|\Delta q^{[i]}\|_\infty$
0	2.46672	1.13076	0.1414214
1	2.30186	0.77673	0.0113876
2	2.29206	0.90077	0.0089981
3	2.28513	0.92540	0.0011791
4	2.28319	0.93618	0.0013948
5	2.28328	0.93942	0.0002601
6	2.28275	0.94005	0.0001871
7	2.28292	0.94058	0.0000506
8	2.28280	0.94056	0.0000245
9	2.28284	0.94065	0.0000086
10	2.28282	0.94063	0.0000033
11	2.28282	0.94065	0.0000015
12	2.28282	0.94065	0.0000006

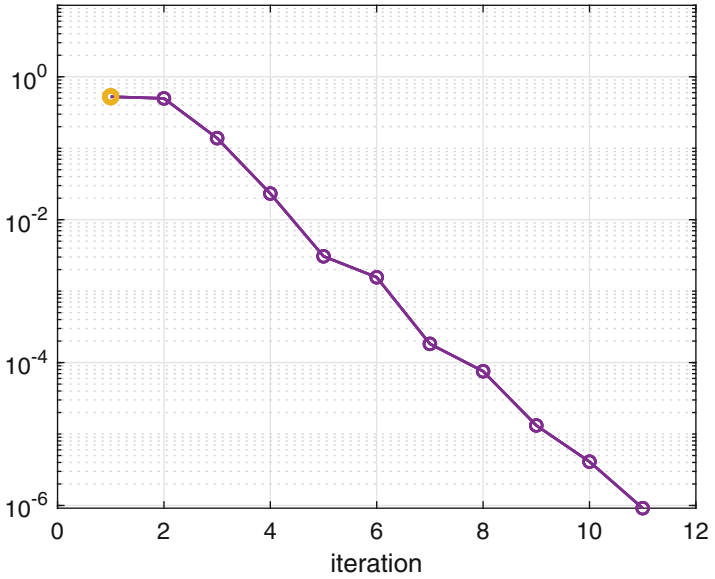

Figure 7 Convergence of $\|\Delta q^{[i]}\|_\infty$ in final point during correction steps

Analogously Figure 10 shows the results for the method described in Sect. 4.2. The post-correction steps yield better approximations for the objective function than for the first method. Even if the number of iterations is higher (for larger deviations) than for the first method, the computational time of under 0.06 s (using MATLAB again) is remarkably smaller as the trapezoidal method is not called any more.

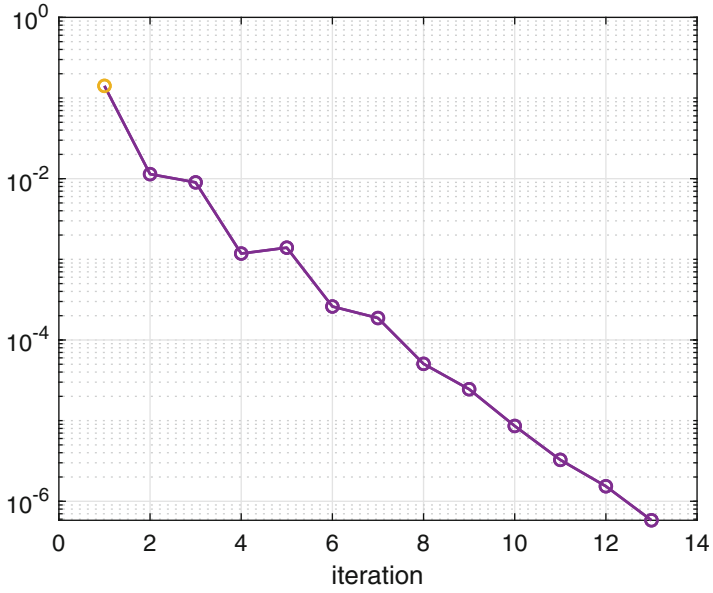

Figure 8 Convergence $\|\Delta q^{[i]}\|_\infty$ in discretized differential equations during correction steps

6 Conclusion

Sensitivity derivatives can be used to generate sub-optimal solutions of perturbed optimal control problems. Constraints, which have been active in the solution of the unperturbed problem, can be iteratively fulfilled. The optimality of the sub-optimal solution depends on the amount of perturbation.

We presented two methods to ensure feasibility in the post-correction steps: First, by using sensitivity derivatives of the boundary condition only small matrices of sensitivity derivatives were needed. However, the states have to be integrated for updated controls to evaluate the deviation. For implicit integration methods as the trapezoidal method, this would require costful operations. Second, by using sensitivity derivatives of discretized differential equations larger matrices of sensitivity derivatives were needed. Apart from function evaluations of f this method only uses matrix–vector multiplications to update the states and controls.

If only selected states have to be transferred to a motor controller, the pre-correction step of the second method already provides an acceptable solution.

As active constraints on the controls would stay active after the correction steps, optimality can't be guaranteed any more. On the other hand, the presented algorithms would not handle it, if inactive constraints on the controls would get active. However, by trimming the controls to the feasible region, the algorithm can be adjusted.

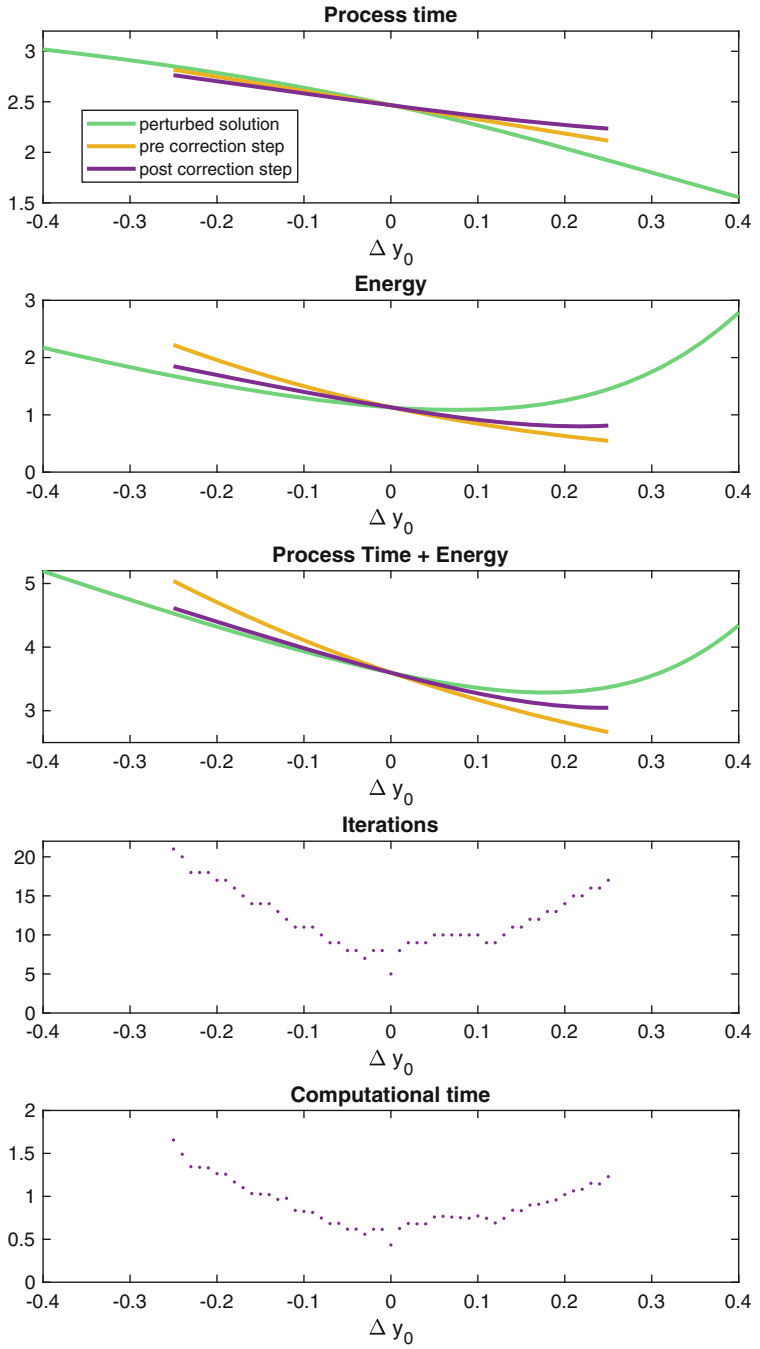

Figure 9 Approximation of the objective function for the perturbed trajectory using sensitivity derivatives of boundary condition

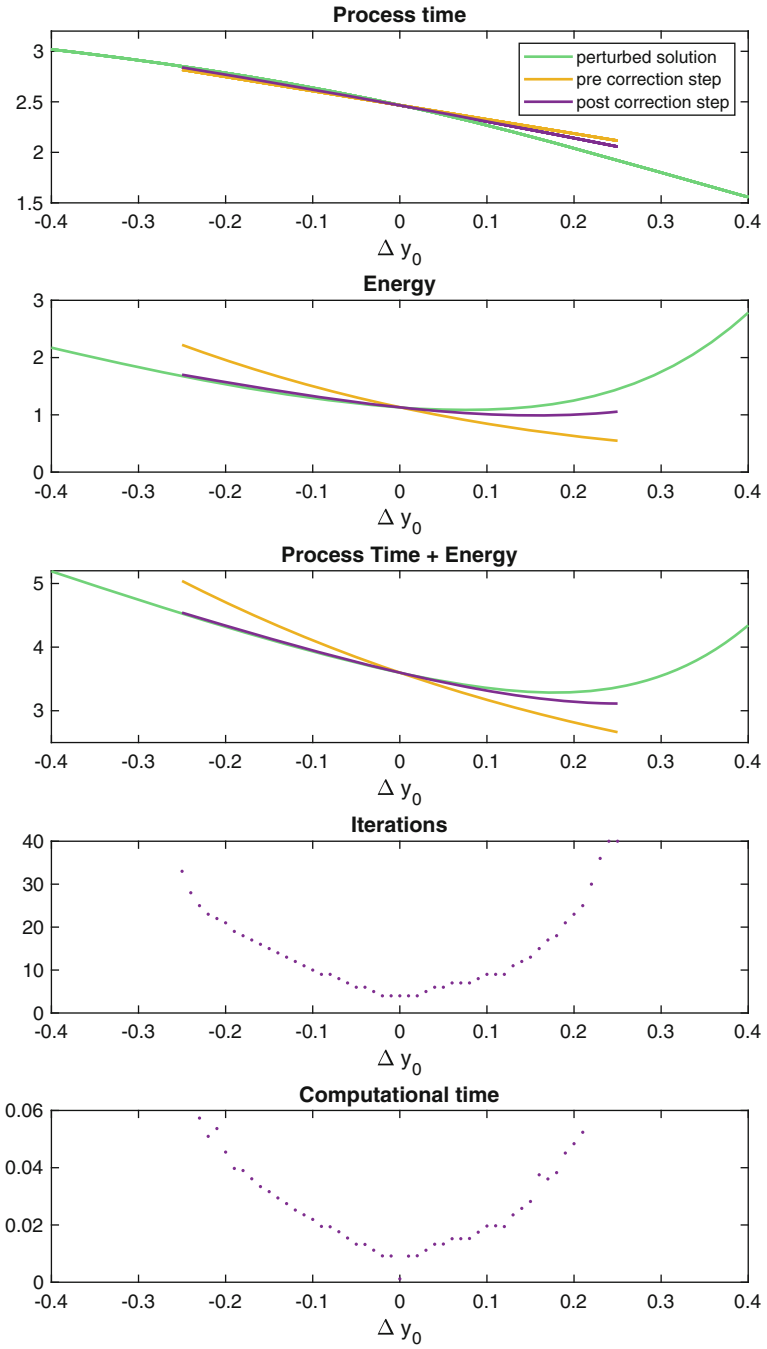

Figure 10 Approximation of the objective function for the perturbed trajectory using sensitivity derivatives of discretized differential equation

The hidden potential of sensitivity derivatives is still large. To improve handling of state perturbations at any time during a manoeuver in the pre-correction step, [12] applied similar techniques to efficient on-board optimization for a Mars entry. However, large data sets for sensitivity derivatives are needed in any of these methods.

References

1. Battin, R.H.: An Introduction to the Mathematics and Methods of Astrodynamics, Revised Edition. AIAA Education Series (1999)
2. Betts, J.T.: Practical Methods for Optimal Control Using Nonlinear Programming. SIAM, Philadelphia (2001)
3. Büskens, C.: Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerprozesse mit Steuer- und Zustandsbeschränkungen. Ph.D. thesis, Universität Münster, Institut für Numerische Mathematik, Münster (1998)
4. Büskens, C.: Real-time optimization and real-time optimal control of parameter-perturbed problems. Habilitation, Universität Bayreuth (2002)
5. Büskens, C., Maurer, H.: SQP-methods for solving optimal control problems with control and state constraints: adjoint variables, sensitivity analysis and real-time control. J. Comput. Appl. Math. **120**, 85–108 (2000)
6. Büskens, C., Wassel, D.: The ESA NLP solver WORHP. In: Modeling and Optimization in Space Engineering, pp. 85–110. Springer, New York (2013)
7. Fiacco, A.V.: Introduction to Sensitivity and Stability Analysis in Nonlinear Programming. Mathematics in Science and Engineering, vol. 165. Academic Press, New York (1983)
8. Grüne, L., Pannek, J.: Nonlinear Model Predictive Control: Theory and Applications. Springer, London (2011)
9. Knauer, M., Büskens, C.: From WORHP to TransWORHP. In: Proceedings of the 5th International Conference on Astrodynamics Tools and Techniques (2012)
10. Knauer, M., Büskens, C.: Processing user input in tracking problems using model predictive control. IFAC-PapersOnLine **50**(1), 9846–9851 (2017)
11. Roenneke, A.J., Cornwell, P.J.: Trajectory control for a low-lift re-entry vehicle. J. Guid. Control. Dyn. **16**(5), 927–933 (1993)
12. Seelbinder, D.: On-board trajectory computation for mars atmospheric entry based on parametric sensitivity analysis of optimal control problems. Ph.D. thesis, Universität Bremen (2017)
13. Topputo, F., Zhang, C.: Survey of direct transcription for low-thrust space trajectory optimization with applications. Abstr. Appl. Anal. **2014**, 15 pp. (2014). Article ID 851720