# Gauss–Newton meets PANOC: A fast and globally convergent algorithm for nonlinear optimal control ⋆

**Pieter Pas** * **Andreas Themelis** ** **Panagiotis Patrinos** *

* *Department of Electrical Engineering (ESAT–STADIUS), KU Leuven, Kasteelpark Arenberg 10, Leuven, 3001, Belgium (e-mail: {pieter.pas,panos.patrinos}@esat.kuleuven.be)*
** *Faculty of Information Science and Electrical Engineering (ISEE), Kyushu University, 744 Motooka, Nishi-ku, 819-0395 Fukuoka, Japan (e-mail: andreas.themelis@ees.kyushu-u.ac.jp)*

**Abstract:** PANOC is an algorithm for nonconvex optimization that has recently gained popularity in real-time control applications due to its fast, global convergence. The present work proposes a variant of PANOC that makes use of Gauss–Newton directions to accelerate the method. Furthermore, we show that when applied to optimal control problems, the computation of this Gauss–Newton step can be cast as a linear quadratic regulator (LQR) problem, allowing for an efficient solution through the Riccati recursion. Finally, we demonstrate that the proposed algorithm is more than twice as fast as the traditional L–BFGS variant of PANOC when applied to an optimal control benchmark problem, and that the performance scales favorably with increasing horizon length.

*Keywords:* Numerical methods for optimal control, Nonconvex optimization, Gauss–Newton, Linear quadratic regulator, Model predictive control

## 1. INTRODUCTION

The ever increasing scale and complexity of models used in optimal control applications necessitate the development of efficient numerical solvers for large-scale, nonconvex optimization. One such solver is PANOC, the *Proximal Averaged Newton-type method for Optimality Conditions* (Stella et al., 2017), which has proven successful in real-time model predictive control (MPC) applications (Sathya et al., 2018; Small et al., 2019; Lindqvist et al., 2022). Various implementations are available, in C++ (Pas et al., 2022), Rust (Sopasakis et al., 2020), and Julia (Stella, 2017). The appeal of an algorithm like PANOC is that it enjoys fast convergence thanks to its Newton-type directions, without giving up any theoretic guarantees about global convergence (De Marchi and Themelis, 2022).

In the original PANOC publication, the limited-memory BFGS (L–BFGS) method was used to generate fast Newton-type directions. In (Pas et al., 2022), the structure of box-constrained problems was exploited to apply L–BFGS more effectively by reducing the size of the Newton system after eliminating active constraints. The present work continues the search for faster and more effective Newton-type directions by exploiting the specific structure of optimal control problems (OCPs).

The remainder of this paper is structured as follows. Section 2 explores a linear Newton approximation (LNA) of the fixed-point residual mapping that lies at the core of PANOC. By using a Gauss–Newton (GN) approximation, the high computational cost of evaluating second-order derivatives is avoided. In Section 3, we go on to apply this Gauss–Newton variant of PANOC to an input-constrained, nonconvex optimal control problem, and show that the computation of the GN step corresponds to the solution of an equality-constrained linear quadratic regulator (LQR) problem. Section 4 solves this problem efficiently using the Riccati recursion. Pseudocode for the algorithm is provided, with a brief discussion of the computational cost. The performance of the resulting solver is validated in Section 5, where it is applied to a challenging model predictive control benchmark. We report a speedup by a factor of two compared to the L–BFGS version of PANOC. Finally, Section 6 concludes with a recapitulation of the main results and a discussion of future work.

### 1.1 Notation

Let $[a, b]$ denote the closed interval from $a$ to $b$, then $\mathbb{N}_{[i,j]} \triangleq [i, j] \cap \mathbb{N}$. $\overline{\mathbb{R}} \triangleq \mathbb{R} \cup \{+\infty\}$ is the set of extended real values. $x_i$ refers to the $i$'th component of $x \in \mathbb{R}^n$. Given an index set $\mathcal{I} = \{i_1, \ldots, i_m\} \subseteq \mathbb{N}_{[1,n]}$, we use

⋆ This work is supported by: the Research Foundation Flanders (FWO) PhD grant No. 11M9523N and research projects G081222N, G033822N, G0A0920N; European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 953348; Research Council KU Leuven C1 project No. C14/18/068; Fonds de la Recherche Scientifique – FNRS; Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS project No. 30468160 (SeLMA); and the Japan Society for the Promotion of Science (JSPS) KAKENHI grant JP21K17710.

$x_{\mathcal{I}} = (x_{i_1}, \ldots, x_{i_m})$. Given a matrix $A \in \mathbb{R}^{n\times m}$, $A_{[\mathcal{I},\mathcal{J}]} \in \mathbb{R}^{\#\mathcal{I}\times\#\mathcal{J}}$ denotes the matrix that consists of all elements of $A$ with row indices in $\mathcal{I}$ and column indices in $\mathcal{J}$; a dot is used to denote all indices, e.g. $A_{[\mathcal{I},\cdot]}$ selects the complete rows of $A$ with row indices in $\mathcal{I}$. In the context of OCPs, the vector $u \in \mathbb{R}^{Nn_u}$ without superscript refers to the concatenation of vectors $u^k \in \mathbb{R}^{n_u}$ for all time steps $k$ in the horizon. Given a positive definite matrix $R$, define the $R$-norm as $\|x\|_R \triangleq \sqrt{x^\top R\, x}$; $\|x\|$ refers to the Euclidean norm. The indicator function $\delta_U$ of a set $U$ is zero for points in $U$ and $+\infty$ otherwise. The proximal operator of a function $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ is defined as $\mathbf{prox}_g(x) \triangleq \arg\min_w \left\{g(w)+\frac{1}{2}\|w-x\|^2\right\}$, with the projection onto $U$ as special case, $\mathbf{\Pi}_U(x) \triangleq \mathbf{prox}_{\delta_U}(x)$. (Rockafellar and Wets, 2004, §1.G). Denote the distance between a point $x$ and a closed set $D$ by $\mathbf{dist}_D(x) \triangleq \|x - \mathbf{\Pi}_D(x)\|$. Let $f : \mathbb{R}^n \to \mathbb{R}^p$ and $g : \mathbb{R}^m \to \mathbb{R}^q$, then $(f \times g) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p \times \mathbb{R}^q : (x, y) \mapsto (f(x), g(y))$ is their Cartesian product, and if $p = q$, their reduced sum is defined as $(f \oplus g) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p : (x, y) \mapsto f(x)+g(y)$. For a function $F : \mathbb{R}^n \to \mathbb{R}^m$, denote its Jacobian matrix by $\mathrm{J}_F : \mathbb{R}^n \to \mathbb{R}^{m\times n}$; a superscript is used to refer to the variables with respect to which to differentiate, e.g. $\mathrm{J}_F^x \triangleq \frac{\partial F}{\partial x} = (\nabla_x F)^\top$. The Clarke generalized Jacobian of $F$ is denoted by $\partial_C F$ (Clarke, 1990), and for a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, define the generalized Hessian matrix as $\partial^2 f \triangleq \partial_C(\nabla f)$.

## 2. GAUSS–NEWTON ACCELERATION OF PANOC

We consider optimization problems of the general form

$$\underset{u}{\mathbf{minimize}} \quad \psi(u) + g(u), \tag{P}$$

where $\psi : \mathbb{R}^n \to \mathbb{R}$ has a locally Lipschitz-continuous gradient but is not necessarily convex, and where $g(u) : \mathbb{R}^n \to \overline{\mathbb{R}}$ is proper, lower semicontinuous, and $\gamma_g$-prox-bounded, but possibly nonsmooth and nonconvex. Problems of this form can be tackled using the proximal gradient method, or accelerated variants thereof, such as the PANOC algorithm (Stella et al., 2017; De Marchi and Themelis, 2022).

### *2.1 Linear Newton approximations for PANOC*

Local solutions to (P) correspond to fixed points of the *forward-backward operator* $T_\gamma(u) \triangleq \mathbf{prox}_{\gamma g}\big(u-\gamma\nabla\psi(u)\big)$, and are characterized by the nonlinear inclusion $0 \in R_\gamma(u)$, where $R_\gamma \triangleq \gamma^{-1}(\mathrm{Id}-T_\gamma)$ is the *fixed-point residual* of $T_\gamma$. Traditionally, PANOC applies the L–BFGS quasi-Newton method to this root-finding problem to achieve fast convergence. A line search over the *forward-backward envelope* $\varphi_\gamma^{\mathrm{FB}}$ is used as a globalization strategy.

This paper explores alternative directions to accelerate PANOC by studying generalized Jacobians to construct a *linear Newton approximation* (LNA) (Facchinei and Pang, 2003) of the fixed-point residual $R_\gamma$.

*Proposition 1.* (LNA scheme for $R_\gamma$)
Suppose that $\nabla\psi$ is semismooth around $\bar{u} \in \mathbb{R}^n$ and that $\mathbf{prox}_{\gamma g}$ with $\gamma > 0$ is semismooth at $\bar{u} - \gamma\nabla\psi(\bar{u})$. Then,

$$H_\gamma(u) \triangleq \gamma^{-1}\mathrm{I} - B(u)\big(\gamma^{-1}\mathrm{I} - \partial^2\psi(u)\big), \tag{1}$$

where $B(u) = \partial_C\, \mathbf{prox}_{\gamma g}\big(u - \gamma\nabla\psi(u)\big)$ and $\partial^2\psi(u) = \partial_C\big(\nabla\psi(u)\big)$, furnishes an LNA scheme for $R_\gamma$ at $\bar{u}$. (Patrinos and Bemporad, 2013, Lem. 6) (Patrinos et al., 2014, Prop. 3.7) (Themelis et al., 2019, §15.4.13)

**Proof.** Because of the semismoothness of $\mathbf{prox}_{\gamma g}$ and $\nabla\psi$, $B(u)$ is an LNA scheme for $\mathbf{prox}_{\gamma g}$ at $\bar{u} - \gamma\nabla\psi(\bar{u})$, and $\mathrm{I} - \gamma\partial^2\psi(u) = \partial_C\big(u - \gamma\nabla\psi(u)\big)$ is an LNA scheme for $\mathrm{Id} - \gamma\nabla\psi$ at $\bar{u}$. By (Facchinei and Pang, 2003, Thm. 7.5.17), the product $B(u)\big(\gamma^{-1}\mathrm{I} - \partial^2\psi(u)\big)$ is an LNA scheme for the composition $T_\gamma = \mathbf{prox}_{\gamma g} \circ (\mathrm{Id} - \gamma\nabla\psi)$ at $\bar{u}$. □

This proposition motivates using a solution $\Delta u$ of the Newton system $H_\gamma(\bar{u})\,\Delta u = -R_\gamma(\bar{u})$ as an update direction for PANOC, using the LNA around the current iterate $\bar{u}$.

### *2.2 Structured PANOC*

In the case where the nonsmooth term $g$ in (P) is the indicator of a rectangular box $U$, $\mathbf{prox}_g$ is a separable projection. This structure can be exploited to reduce the dimension of the Newton system (Pas et al., 2022, §III).

Decompose the box $U \triangleq \bigtimes_{i=1}^n U_i$ as a Cartesian product of one-dimensional intervals. Then, $B(u) = \partial_C\, \mathbf{\Pi}_U\big(u - \gamma\nabla\psi(u)\big)$ is a set of diagonal matrices with

$$B(u)_{ii} \in \begin{cases} \{0\} & \text{if } u_i - \gamma\nabla_i\psi(u) \notin U_i, \\ \{1\} & \text{if } u_i - \gamma\nabla_i\psi(u) \in \mathbf{int}\, U_i, \\ [0,1] & \text{if } u_i - \gamma\nabla_i\psi(u) \in \mathbf{bdry}\, U_i. \end{cases} \tag{2}$$

Motivated by these different cases, let us define the index sets $\mathcal{K}(u) \triangleq \big\{i \in \mathbb{N}_{[1,\,n]} \;\big|\; u_i - \gamma\nabla_i\psi(u) \notin \mathbf{int}\, U_i\big\}$ and $\mathcal{J}(u) \triangleq \big\{i \in \mathbb{N}_{[1,\,n]} \;\big|\; u_i - \gamma\nabla_i\psi(u) \in \mathbf{int}\, U_i\big\}$ of active and inactive constraints respectively, and choose $\hat{B}(u) \in B(u)$, defining $\hat{B}(u)_{ii} \triangleq 0$ if $i \in \mathcal{K}(u)$ and $\hat{B}(u)_{ii} \triangleq 1$ if $i \in \mathcal{J}(u)$.

By permutation of (1), the Newton step $\Delta u$ at a point $\bar{u}$ can then be computed by solving the system

$$\begin{cases} \Delta u_{\mathcal{K}} = \bar{u}_{\mathcal{K}} - T_\gamma(\bar{u})_{\mathcal{K}}, \\ \partial^2_{\mathcal{J}\mathcal{J}}\psi(\bar{u})\,\Delta u_{\mathcal{J}} = -\nabla_{\mathcal{J}}\psi(\bar{u}) - \partial^2_{\mathcal{J}\mathcal{K}}\psi(\bar{u})\,\Delta u_{\mathcal{K}}. \end{cases} \tag{3}$$

### *2.3 Gauss–Newton approximation*

We will now specialize to problems where the smooth term is a composition $\psi(u) \triangleq \ell\big(F(u)\big)$ of $\ell : \mathbb{R}^m \to \mathbb{R}$ convex and $F : \mathbb{R}^n \to \mathbb{R}^m$. Considering the computational cost of evaluating and factorizing the second-order derivatives of $\psi$, the proposed method approximates (3) using the Gauss–Newton matrix $\hat{\nabla}^2_{\mathrm{GN}} \triangleq \mathrm{J}_F(u)^\top\, \partial^2\ell\big(F(u)\big)\, \mathrm{J}_F(u)$ (Schraudolph, 2002, §3).

*Remark 2.* For $\psi \in C^2$, we have $\nabla^2\psi = \hat{\nabla}^2_{\mathrm{GN}} + \delta^2_{\mathrm{GN}}$ with $\delta^2_{\mathrm{GN}}(u) \triangleq \sum_{i=1}^m \nabla_i\,\ell\big(F(u)\big)\,\nabla^2 F_i(u)$. If the function $F$ is linear around a solution $u^\star$, or if $F(u^\star)$ is a stationary point of $\ell$, the error term $\delta^2_{\mathrm{GN}}$ vanishes, and the Gauss–Newton approximation approaches the true Hessian matrix of $\psi$.

Substituting $\partial^2\psi$ by $\hat{\nabla}^2_{\mathrm{GN}}$ in (3) and writing the solution to the resulting system as the solution of an equality constrained quadratic program yields

$$\begin{aligned} \underset{\Delta u}{\mathbf{minimize}} \quad & \tfrac{1}{2}\,\Delta u^\top \hat{\nabla}^2_{\mathrm{GN}}(\bar{u})\,\Delta u + \nabla\psi(\bar{u})^\top\Delta u \\ \mathbf{subject\ to} \quad & \Delta u_{\mathcal{K}} = u_{\mathcal{K}} - T_\gamma(\bar{u})_{\mathcal{K}}. \end{aligned} \tag{GN-QP}$$

The following sections explore methods for efficiently solving this Gauss–Newton QP (GN-QP) by making use of

$$\begin{aligned}
\underset{\Delta x,\Delta u}{\textbf{minimize}} \quad & \tfrac{1}{2}\sum_{k=0}^{N-1}\begin{pmatrix}\Delta x^k\\ \Delta u^k\end{pmatrix}^\top\begin{pmatrix}Q_k & S_k^\top\\ S_k & R_k\end{pmatrix}\begin{pmatrix}\Delta x^k\\ \Delta u^k\end{pmatrix} + \tfrac{1}{2}\left(\Delta x^N\right)^\top Q_N\left(\Delta x^N\right) + \sum_{k=0}^{N-1}\begin{pmatrix}q^k\\ r^k\end{pmatrix}^\top\begin{pmatrix}\Delta x^k\\ \Delta u^k\end{pmatrix} + \left(q^N\right)^\top\left(\Delta x^N\right)\\
\textbf{subject to} \quad & \Delta x^0 = 0\\
& \Delta x^{k+1} = A_k\Delta x^k + B_k\Delta u^k \qquad (0\le k<N)\\
& \Delta u_{\mathcal{K}} = u_{\mathcal{K}} - T_\gamma(u)_{\mathcal{K}}
\end{aligned} \tag{P-ELQR}$$

$$\begin{aligned}
\underset{\Delta x,\Delta u_{\mathcal{J}}}{\textbf{minimize}} \quad & \tfrac{1}{2}\sum_{k=0}^{N-1}\begin{pmatrix}\Delta x^k\\ \Delta u_{\mathcal{J}}^k\end{pmatrix}^\top\begin{pmatrix}Q_k & \hat{S}_k^\top\\ \hat{S}_k & \hat{R}_k\end{pmatrix}\begin{pmatrix}\Delta x^k\\ \Delta u_{\mathcal{J}}^k\end{pmatrix} + \tfrac{1}{2}\left(\Delta x^N\right)^\top Q_N\left(\Delta x^N\right) + \sum_{k=0}^{N-1}\begin{pmatrix}\hat{q}^k\\ \hat{r}^k\end{pmatrix}^\top\begin{pmatrix}\Delta x^k\\ \Delta u_{\mathcal{J}}^k\end{pmatrix} + \left(\hat{q}^N\right)^\top\left(\Delta x^N\right)\\
\textbf{subject to} \quad & \Delta x^0 = 0\\
& \Delta x^{k+1} = A_k\Delta x^k + \hat{B}_k\Delta u_{\mathcal{J}}^k + \hat{c}_k \qquad (0\le k<N)
\end{aligned} \tag{P-LQR}$$

the particular structure of finite-horizon optimal control problems. The Gauss–Newton step $\Delta u$ can then be used as an accelerated direction for PANOC.

## 3. OPTIMAL CONTROL

This section explores how optimal control problems fit into the optimization framework from the previous section, and how their specific structure can be exploited to compute Gauss–Newton directions efficiently.

### *3.1 Problem formulation*

Consider the following general formulation of a nonlinear optimal control problem with finite horizon $N$.

$$\begin{aligned}
\underset{u,x}{\textbf{minimize}} \quad & \sum_{k=0}^{N-1}\ell_k\big(h_k(x^k,u^k)\big) + \ell_N\big(h_N(x^N)\big)\\
\textbf{subject to} \quad & u\in U\\
& x^0 = x_{\text{init}}\\
& x^{k+1} = f(x^k,u^k) \qquad (0\le k<N)
\end{aligned} \tag{OCP}$$

The function $f:\mathbb{R}^{n_x}\times\mathbb{R}^{n_u}\to\mathbb{R}^{n_x}$ models the discrete-time, nonlinear dynamics of the system, with initial state $x_{\text{init}}$. The functions $h_k:\mathbb{R}^{n_x}\times\mathbb{R}^{n_u}\to\mathbb{R}^{n_y^k}$ for $0\le k<N$ and $h_N:\mathbb{R}^{n_x}\to\mathbb{R}^{n_y^N}$ can be used to represent the (possibly time-varying) output mapping of the system, and the convex functions $\ell_k:\mathbb{R}^{n_y^k}\to\mathbb{R}$ and $\ell_N:\mathbb{R}^{n_y^N}\to\mathbb{R}$ define the stage costs and the terminal cost respectively.

The problem (OCP) can be transcribed into formulation (P) as follows. Recursively define the state transition function $\Phi^k$ as $\Phi^0(u)\triangleq x_{\text{init}}$ and $\Phi^{k+1}(u)\triangleq f\big(\Phi^k(u),u^k\big)$. Define $G$ as the function that maps a sequence of inputs to the interleaved states and inputs over the horizon, $G(u) = \big(\Phi^0(u),\, u_0,\, \Phi^1(u),\, u_1,\, \ldots,\, \Phi^N(u)\big)$. Using this definition, the *single-shooting* or *sequential* formulation of problem (OCP) is an instance of (P), with $\ell=\ell_0\oplus\cdots\oplus\ell_N$, $h=h_0\times\cdots\times h_N$, $F=h\circ G$, $\psi=\ell\circ F$ and $g=\delta_U$, i.e.

$$\begin{aligned}
\underset{u}{\textbf{minimize}} \quad & \ell\big(h\big(G(u)\big)\big)\\
\textbf{subject to} \quad & u\in U.
\end{aligned} \tag{SS-OCP}$$

### *3.2 Gauss–Newton approximations for optimal control*

By specializing the Gauss–Newton QP (GN-QP) for this class of optimal control problems, and by exploiting the separable structure of the objective function, the Gauss–Newton step can be shown to be the solution to the equality-constrained, finite-horizon, linear quadratic regulator problem (P-ELQR) at the top of this page. For the sake of readability, we defined the following variables.

$$\begin{aligned}
&\bar{x}^k \triangleq \Phi^k(\bar{u}) && \hbar^k \triangleq h_k(\bar{x}^k,\bar{u}^k)\\
&q^k \triangleq \mathrm{J}^x_{h_k}(\bar{x}^k,\bar{u}^k)^\top\nabla\ell_k(\hbar^k) && r^k \triangleq \mathrm{J}^u_{h_k}(\bar{x}^k,\bar{u}^k)^\top\nabla\ell_k(\hbar^k)\\
&A_k \triangleq \mathrm{J}^x_f(\bar{x}^k,\bar{u}^k) && B_k \triangleq \mathrm{J}^u_f(\bar{x}^k,\bar{u}^k)\\
&Q_k \triangleq \mathrm{J}^x_{h_k}(\bar{x}^k,\bar{u}^k)^\top\,\partial^2\ell_k(\hbar^k)\,\mathrm{J}^x_{h_k}(\bar{x}^k,\bar{u}^k)\\
&S_k \triangleq \mathrm{J}^u_{h_k}(\bar{x}^k,\bar{u}^k)^\top\,\partial^2\ell_k(\hbar^k)\,\mathrm{J}^x_{h_k}(\bar{x}^k,\bar{u}^k)\\
&R_k \triangleq \mathrm{J}^u_{h_k}(\bar{x}^k,\bar{u}^k)^\top\,\partial^2\ell_k(\hbar^k)\,\mathrm{J}^u_{h_k}(\bar{x}^k,\bar{u}^k)
\end{aligned} \tag{6}$$

In order to transform (P-ELQR) into a standard linear quadratic regulator formulation, eliminate the fixed variables $u_{\mathcal{K}}$. The result is the problem (P-LQR), where we used the following definitions.

$$\begin{aligned}
&\hat{q}_k \triangleq q^k + S_k^\top{}_{[\,\cdot\,,\mathcal{K}]}\,u^k_{\mathcal{K}}, && \hat{r}_k \triangleq r^k_{\mathcal{J}} + R_{k[\mathcal{J},\mathcal{K}]}\,u^k_{\mathcal{K}}, \qquad (7)\\
&\hat{S}_k \triangleq S_{k[\mathcal{J},\,\cdot\,]}, \;\; \hat{R}_k \triangleq R_{k[\mathcal{J},\mathcal{J}]}, \;\; \hat{B}_k \triangleq B_{k[\,\cdot\,,\mathcal{J}]}, \;\; \hat{c}_k \triangleq B_{k[\,\cdot\,,\mathcal{K}]}\,u^k_{\mathcal{K}}.
\end{aligned}$$

*Remark 3.* In the absence of box constraints, we have $\mathcal{K}=\emptyset$, and the algorithm reduces to the iterative linear quadratic regulator (ILQR) method for nonlinear MPC of (Li and Todorov, 2004) with a line search.

### *3.3 Handling state constraints*

Consider a standard state-constrained finite-horizon optimal control problem of the following form.

$$\begin{aligned}
\underset{u,x}{\textbf{minimize}} \quad & \tfrac{1}{2}\sum_{k=0}^{N-1}\left[\left\|x^k - x_{\mathrm{r}}\right\|_Q^2 + \left\|u^k - u_{\mathrm{r}}\right\|_R^2\right]\\
& + \tfrac{1}{2}\left\|x^N - x_{\mathrm{r}}\right\|_{Q_N}^2\\
\textbf{subject to} \quad & u\in U\\
& x^0 = x_{\text{init}}\\
& x^{k+1} = f(x^k,u^k) \qquad (0\le k<N)\\
& c_k(x^k)\in D_k \qquad (0\le k\le N)
\end{aligned} \tag{SC-OCP}$$

The costs are now given by the weighted squared distance to the reference state $x_{\mathrm{r}}$ and reference input $u_{\mathrm{r}}$. A smooth, possibly nonlinear function $c_k$ of the states enables the representation of general equality and inequality constraints, by constraining its image to the box $D_k$.

It is common practice to relax the state constraints by means of a penalty method, moving the constraints to

the objective function as quadratic penalty terms. For example, $\frac{\mu}{2}\,\mathbf{dist}^2_{D_k}\big(c_k(x^k)\big)$, for sufficiently large $\mu > 0$.

Such a soft-constrained optimal control problem fits into the framework of (SS-OCP) by defining

$$\begin{aligned} \ell_k(x,u,z) &\triangleq \tfrac{1}{2}\left\|x - x_\mathrm{r}\right\|^2_Q + \tfrac{1}{2}\left\|u - u_\mathrm{r}\right\|^2_R + \tfrac{\mu_k}{2}\,\mathbf{dist}^2_{D_k}(z), \\ \ell_N(x,z) &\triangleq \tfrac{1}{2}\left\|x - x_\mathrm{r}\right\|^2_{Q_N} + \tfrac{\mu_N}{2}\,\mathbf{dist}^2_{D_N}(z), \\ h_k(x,u) &\triangleq \big(x,\ u,\ c_k(x)\big), \quad h_N(x) \triangleq \big(x,\ c_N(x)\big). \end{aligned} \tag{8}$$

The cost $\ell$ is no longer twice differentiable, but its gradient $\nabla\ell$ is locally Lipschitz continuous, and hence its Clarke generalized Jacobian $\partial^2\ell$ is well defined and nonempty (Facchinei and Pang, 2003, Prop. 7.1.4). Additionally, the gradient is semismooth, so Proposition 1 applies.

The following proposition gives a sufficient condition for the solution to the Gauss–Newton QP (GN-QP) to be uniquely defined.

*Proposition 4.* If the cost matrix $R$ is positive definite, $Q$ is positive semidefinite, and $\mu_k \geq 0$ for all $k$, then the Gauss–Newton matrix $\hat{\nabla}^2_\mathrm{GN}$ for the soft-constrained optimal control problem is positive definite.

**Proof.** By algebraic manipulations of $\hat{\nabla}^2_\mathrm{GN}$.
Because of the block-diagonal structure of $\partial^2\ell$ and $\mathrm{J}_h$, their product $L \triangleq \mathrm{J}_h^\top \partial^2\ell\, \mathrm{J}_h$ is also block-diagonal, with blocks of the form $\mathbf{blkdiag}(Q + C_k^\top M_k C_k,\ R) \succeq 0$, where $C_k \triangleq \mathrm{J}_{c_k}(x^k)$ and $M_k \in \partial^2\big(\frac{\mu}{2}\,\mathbf{dist}^2_{D_k}(c_k(x^k))\big)$. Because of the structure of $G$ (it includes the identity map of $u$), the block rows of $\mathrm{J}_G(u)$ that correspond to the inputs have full rank (they contain $n_u \times n_u$ identity matrices) and line up with the positive definite blocks $R$ in $L$. Hence, the full product $\hat{\nabla}^2_\mathrm{GN} = \mathrm{J}_G(u)^\top L\, \mathrm{J}_G(u)$ is positive definite. □

## 4. ALGORITHMIC DETAILS

We will now describe an algorithm for efficiently solving (P-LQR) to obtain the Gauss–Newton step $\Delta u$ that can be used to accelerate PANOC.

The PANOC$^+$ method from (De Marchi and Themelis, 2022) is given in Algorithm 1. It has been specialized to use the GN step $\Delta u$ derived in Section 2. Unlike the original version of PANOC$^+$ with an L–BFGS accelerator, a GN step can be computed from the very first iteration.

### *4.1 Solution of the LQR problem*

The Gauss–Newton step $\Delta u$ can be computed as the solution to (P-LQR) using LQR factorization and LQR solution routines based on the Riccati recursion (Rawlings et al., 2017, §8.8.3), (Patrinos and Bemporad, 2014, Alg. 3-4). These routines, specialized to the problem at hand, are listed in Algorithms 2 and 3.

An important observation is that the cost for the computation of the Gauss–Newton direction using these routines scales *linearly* with the horizon length $N$. In the worst case, when $\mathcal{K}(\bar{u}) = \emptyset$, Algorithm 2 requires the factorization of $N$ matrices of size $n_u \times n_u$ and some matrix products. In contrast, general direct solution methods for system (3) require a single factorization of a much larger $n_u N \times n_u N$ matrix, with a cost that scales *cubically* with $N$.

**Algorithm 1:** PANOC$^+$ (De Marchi and Themelis, 2022, Algorithm 2) with Gauss–Newton acceleration

**In:** initial guess $u^{(0)}$, initial step size $\gamma_0 > 0$, parameters $\alpha, \beta \in (0,1)$
**Out:** $u^\star$

$\hat{u}^{(0)} \leftarrow T_{\gamma_0}(u^{(0)}), \quad p^{(0)} \leftarrow \hat{u}^{(0)} - u^{(0)}, \quad \nu \leftarrow 1$
**while** Stopping criterion not satisfied for $u^{(\nu-1)}$
  Compute $\Delta u$ from (GN-QP) with $\bar{u} \triangleq u^{(\nu-1)}$
  $\gamma_\nu \leftarrow \gamma_{\nu-1}, \quad \tau \leftarrow 1$
  ▷
  $u^{(\nu)} \leftarrow u^{(\nu-1)} + (1-\tau)\,p^{(\nu-1)} + \tau\,\Delta u$
  $\hat{u}^{(\nu)} \leftarrow T_{\gamma_\nu}(u^{(\nu)}), \quad p^{(\nu)} \leftarrow \hat{u}^{(\nu)} - u^{(\nu)}$
  **if** $\psi(\hat{u}^{(\nu)}) > \psi(u^{(\nu)}) + \nabla\psi(u^{(\nu)})^\top p^{(\nu)} + \frac{\alpha}{2\gamma_\nu}\left\|p^{(\nu)}\right\|^2$
    $\gamma_\nu \leftarrow \gamma_\nu/2, \quad \tau \leftarrow 1$ and go to ▷
  **if** $\varphi^\mathrm{FB}_{\gamma_\nu}(u^{(\nu)}) > \varphi^\mathrm{FB}_{\gamma_{\nu-1}}(u^{(\nu-1)}) - \beta\frac{1-\alpha}{2\gamma_{\nu-1}}\left\|p^{(\nu-1)}\right\|^2$
    $\tau \leftarrow \tau/2$ and go to ▷
  $\nu \leftarrow \nu + 1$
$u^\star \leftarrow T_{\gamma_{\nu-1}}(u^{(\nu-1)})$

**Algorithm 2:** LQR factor

**In:** $Q_k, \hat{S}_k, \hat{R}_k, \hat{q}_k, \hat{r}_k, A_k, \hat{B}_k, \hat{c}_k$ **Out:** $K_k, e_k$
$P_N \leftarrow Q_N, \quad s_N \leftarrow \hat{q}_N$
**for** $k = N-1, \ldots, 0$
  $\bar{R} \leftarrow \hat{R}_k + \hat{B}_k^\top P_{k+1} \hat{B}_k$
  $\bar{S} \leftarrow \hat{S}_k + \hat{B}_k^\top P_{k+1} A_k$
  $y \leftarrow P_{k+1}\hat{c}_k + s_{k+1}$
  $K_k \leftarrow -\bar{R}^{-1}\bar{S}$
  $e_k \leftarrow -\bar{R}^{-1}(\hat{B}_k^\top y + \hat{r}_k)$
  $s_k \leftarrow \bar{S}^\top e_k + A_k^\top y + \hat{q}_k$
  $P_k \leftarrow Q_k + A_k^\top P_{k+1} A_k + \bar{S}^\top K_k$

**Algorithm 3:** LQR solve

**In:** $A_k, B_k, K_k, e_k, \Delta u_\mathcal{K}$ **Out:** $\Delta u_\mathcal{J}, \Delta x$
$\Delta x^0 \leftarrow 0$
**for** $k = 0, \ldots, N-1$
  $\Delta u^k_\mathcal{J} \leftarrow K_k \Delta x^k + e_k$
  $\Delta x^{k+1} \leftarrow A_k \Delta x^k + B_k \Delta u^k$

### *4.2 Practical considerations*

Far away from the solution, the Gauss–Newton model might not approximate the true function well, and the GN step might not perform much better than an L–BFGS step. Considering the significant difference in computational cost between Gauss–Newton and L–BFGS (the former requires evaluation of the Jacobians of the dynamics, matrix factorizations and multiplications, whereas the latter only requires a limited number of vector operations), we propose to only compute the GN step every $k_\mathrm{GN} \geq 1$ iterations. In between, cheaper structured PANOC L–BFGS steps are used (Pas et al., 2022, §III). When eventually a GN step is accepted by the line search with step size $\tau = 1$, the algorithm continues to take GN steps, for as long as they keep getting accepted with unit step size. Using this technique, the algorithm initially maintains

a low cost per iteration, and eventually enjoys the fast local convergence of the more expensive GN iterations. This will be corroborated experimentally in the following section.

## 5. EXPERIMENTAL RESULTS

In this section, the PANOC algorithm with Gauss–Newton acceleration is applied to a nonlinear, input-constrained model predictive control problem, and its performance is compared to the approximate structured PANOC algorithm with L–BFGS acceleration from (Pas et al., 2022). As a benchmark, we consider the optimal control of a *"chain of masses connected by springs"* described by (Wirsching et al., 2006). One side of the chain is fixed, and the other side is attached to an actuator. A disturbance is applied to the system, and the controller aims to bring the chain back to a steady state, with the actuator at a given target position. The velocity of the actuator is limited to 1 m/s along each axis. Unless specified otherwise, we use the parameter values listed in (Wirsching et al., 2006).

CasADi (Andersson et al., 2019) is used to model and discretize the problem using a fourth-order Runge–Kutta integrator, and to generate C code for evaluating the required problem functions and their derivatives. These functions are then used in an optimized C++ implementation of Algorithms 1–3, based on ALPAQA (Pas, 2021). [1]

### 5.1 Number of iterations

In a first experiment, the convergence in terms of the number of iterations is compared for the PANOC algorithm with Gauss–Newton acceleration as described in this publication, and for the structured PANOC algorithm with L–BFGS acceleration without the off-diagonal Hessian–vector term from (Pas et al., 2022). For the GN accelerator, the parameter $k_{\mathrm{GN}}$ from Section 4.2 is set to one (i.e. a GN step is computed on each PANOC iteration). The L–BFGS memory is set to 40, equal to the length of the horizon. Figure 1 shows the convergence of the two algorithms when applied to an OCP using the chain model. Initially, they both perform similarly, but after around 20 iterations, the GN directions are accepted with unit step size, enabling very fast linear convergence.

Note that a similar graph in terms of solver run time would look quite different: even though the progress per iteration is comparable for the first 20 iterations, the computational cost per iteration for the GN accelerator is around ten times higher than for the L–BFGS accelerator. This can be greatly improved by increasing $k_{\mathrm{GN}}$.

### 5.2 Run time in function of horizon length

In a second experiment, we explore the effect of the horizon length on the solver run time. For each horizon length between $N = 10$ and $N = 45$, 256 optimal control problems are composed, each with a different initial state $x_{\mathrm{init}}$, generated by applying uniformly random inputs in $[-1, 1]$ for five time steps. We set $k_{\mathrm{GN}}$ to 30, and select the L–BFGS memory to equal the horizon length $N$. The solvers declare convergence when $\left\|u^{(\nu)} - \mathbf{\Pi}_U\left(u^{(\nu)} - \nabla\psi(u^{(\nu)})\right)\right\| \leq 10^{-10}$. Run times for structured PANOC with L–BFGS and for PANOC with GN acceleration are reported in Figure 2. The solver with GN acceleration is more than twice as fast as the L–BFGS variant, and the run time scales close to linearly with the horizon length $N$.

[1] The Python source code to reproduce the results can be found at `github.com/kul-optec/panoc-gauss-newton-ifac-experiments`. The experiments were carried out using a Core i7-7700HQ@2.8 GHz.

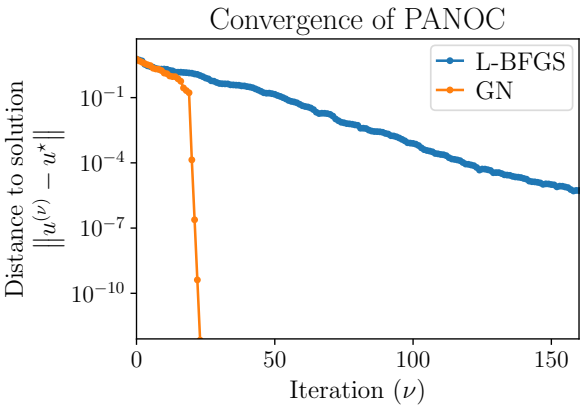



Fig. 1. Comparison of the convergence of structured PANOC with L–BFGS and PANOC with the proposed Gauss–Newton accelerator ($k_{\mathrm{GN}} = 1$), when applied to the chain OCP.

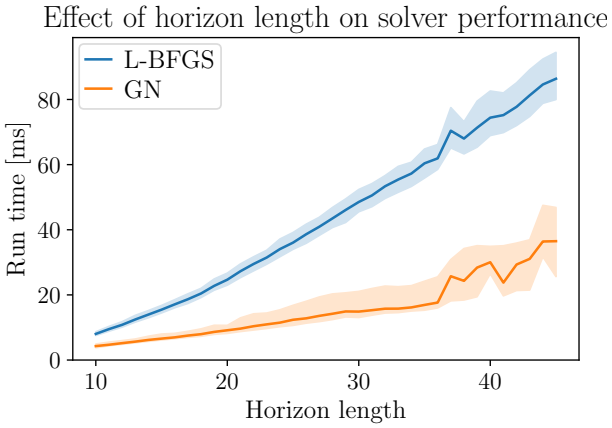



Fig. 2. Median solver run time over the 256 test problems for each horizon length, for structured PANOC with L–BFGS and PANOC with the Gauss–Newton accelerator ($k_{\mathrm{GN}} = 30$). The shaded area indicates the P10 and P90 percentiles.

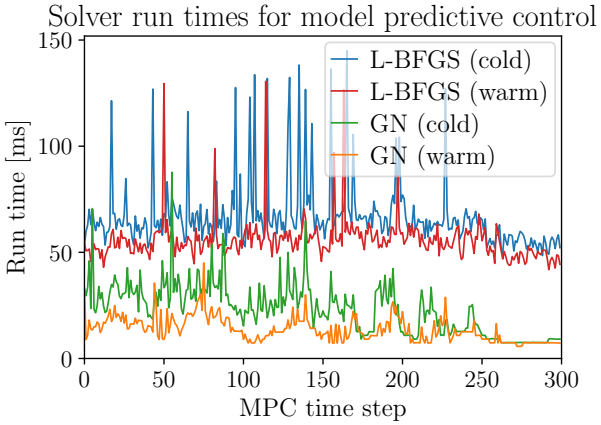



Fig. 3. Solver run times for structured PANOC with L–BFGS and PANOC with the Gauss–Newton accelerator ($k_{\mathrm{GN}} = 10$) when applied to the chain MPC problem. The *cold* label indicates an initial guess of zero, for *warm* it is the shifted previous solution.

### 5.3 Model predictive control

Finally, both solvers are applied in a closed-loop controller. A disturbance of $[-1, 1, 1]$ m/s is applied for five time steps, and the system with the MPC controller is simulated for one minute. The run times of the two solvers described earlier are reported in Figure 3. The GN solver (with $k_{\mathrm{GN}} = 10$) outperforms the L–BFGS-based solver in terms of both average and worst-case run time. The fast local convergence of GN is especially noticeable when the initial guess is close to the solution, e.g. by warm starting the solver using the shifted solution from the previous time step, and when the system starts to settle near the end of the simulation. For reference, the popular `Ipopt` solver (Wächter and Biegler, 2006) requires around 1.7 seconds to solve the first OCP (without JIT compilation), compared to just 30 ms for the PANOC solver with GN acceleration.

## 6. CONCLUSION

In this paper, we extended the PANOC algorithm to enable acceleration using Gauss–Newton directions. We showed how the structure of optimal control problems can be exploited to efficiently compute these Gauss–Newton directions using the Riccati recursion, in such a way that the computational cost scales linearly with the horizon length. Performance of the proposed method was then compared to a previous variant of PANOC: we reported a speedup by a factor of two for a challenging optimal control benchmark problem.

An open-source C++ implementation of the solver is available in the ALPAQA GitHub repository. (Pas, 2021). Further performance improvements could be achieved by exploiting the sparsity of the Jacobians, or by employing tailored linear algebra routines (Frison et al., 2018).

## REFERENCES


Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2019). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1), 1–36.

Clarke, F. (1990). *Optimization and Nonsmooth Analysis*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics.

De Marchi, A. and Themelis, A. (2022). Proximal gradient algorithms under local Lipschitz gradient continuity. *Journal of Optimization Theory and Applications*, 194(3), 771–794.

Facchinei, F. and Pang, J.S. (2003). *Finite-Dimensional Variational Inequalities and Complementarity Problems*, volume II. Springer.

Frison, G., Kouzoupis, D., Zanelli, A., and Diehl, M. (2018). BLASFEO: Basic linear algebra subroutines for embedded optimization. *ACM Trans. Math. Softw.*, 44, 42:1–42:30.

Li, W. and Todorov, E. (2004). Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO*.

Lindqvist, B., Mansouri, S.S., Haluška, J., and Nikolakopoulos, G. (2022). Reactive navigation of an unmanned aerial vehicle with perception-based obstacle avoidance constraints. *IEEE Transactions on Control Systems Technology*, 30(5), 1847–1862.

Pas, P. (2021). ALPAQA: A matrix-free solver for nonlinear MPC and large-scale nonconvex optimization. URL https://github.com/kul-optec/alpaqa.

Pas, P., Schuurmans, M., and Patrinos, P. (2022). ALPAQA: A matrix-free solver for nonlinear MPC and large-scale nonconvex optimization. In *2022 European Control Conference (ECC)*, 417–422.

Patrinos, P. and Bemporad, A. (2013). Proximal Newton methods for convex composite optimization. In *52nd IEEE Conference on Decision and Control*, 2358–2363.

Patrinos, P. and Bemporad, A. (2014). An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *IEEE Transactions on Automatic Control*, 59(1), 18–33.

Patrinos, P., Stella, L., and Bemporad, A. (2014). Forward-backward truncated Newton methods for convex composite optimization.

Rawlings, J., Mayne, D., and Diehl, M. (2017). *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing.

Rockafellar, R.T. and Wets, R.J.B. (2004). *Variational analysis*. Grundlehren der mathematischen Wissenschaften 317. Springer, Berlin.

Sathya, A., Sopasakis, P., Van Parys, R., Themelis, A., Pipeleers, G., and Patrinos, P. (2018). Embedded nonlinear model predictive control for obstacle avoidance using PANOC. In *2018 European Control Conference (ECC)*, 1523–1528.

Schraudolph, N.N. (2002). Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7), 1723–1738.

Small, E., Sopasakis, P., Fresk, E., Patrinos, P., and Nikolakopoulos, G. (2019). Aerial navigation in obstructed environments with embedded nonlinear model predictive control. In *2019 18th European Control Conference (ECC)*, 3556–3563.

Sopasakis, P., Fresk, E., and Patrinos, P. (2020). OpEn: Code Generation for Embedded Nonconvex Optimization. *IFAC-PapersOnLine*, 53(2), 6548–6554. 21st IFAC World Congress.

Stella, L. (2017). ProximalAlgorithms.jl: Proximal algorithms for nonsmooth optimization in Julia. URL https://github.com/JuliaFirstOrder/ProximalAlgorithms.jl.

Stella, L., Themelis, A., Sopasakis, P., and Patrinos, P. (2017). A simple and efficient algorithm for nonlinear model predictive control. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 1939–1944.

Themelis, A., Ahookhosh, M., and Patrinos, P. (2019). On the acceleration of forward-backward splitting via an inexact Newton method. In H.H. Bauschke, R.S. Burachik, and D.R. Luke (eds.), *Splitting Algorithms, Modern Operator Theory, and Applications*, 363–412. Springer International Publishing, Cham.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1), 25–57.

Wirsching, L., Bock, H.G., and Diehl, M. (2006). Fast NMPC of a chain of masses connected by springs. In *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*, 591–596.