Proceedings of the ASME 2019
Dynamic Systems and Control Conference
DSCC2019
October 8-11, 2019, Park City, Utah, USA

DSCC2019-9067

THE RESTRICTED NEWTON METHOD FOR FAST NONLINEAR MODEL PREDICTIVE CONTROL

Anson Maitland *
Chi Jin
John McPhee
Department of Systems Design Engineering
University of Waterloo
Waterloo, ON, N2L 3G1
Canada

ABSTRACT

We introduce the Restricted Newton's Method (RNM), a basic optimization method, to accelerate model predictive control turnaround times. RNM is a hybrid of Newton's method (NM) and gradient descent (GD) that can be used as a building block in nonlinear programming. The two parameters of RNM are the subspace on which we restrict the Newton steps and the maximal size of the GD step. We present a convergence analysis of RNM and demonstrate how these parameters can be selected for MPC applications using simple machine learning methods. This leads to two parameter selection strategies with different convergence behaviour. Lastly, we demonstrate the utility of RNM on a sample autonomous vehicle problem with promising results.

INTRODUCTION

Model predictive control (MPC) is a feedback-based control strategy that provides a tractable method to approximate a solution to a real world optimal control problem [1, 2]. In its most basic form MPC uses the most recent measurement (or estimate) of the plant's state and a plant model to predict the future states of the plant under some control policy, and then optimizes that control policy with respect to a chosen cost. This cost captures the desired objective of our controller but can also be chosen to satisfy stability as well as other technical concerns [3]. An MPC operates under the receding horizon principle by optimizing the control policy over some future horizon, implements only the first controller action, and repeats this process at each timestep with an updated measurement of the plant's state. From a theoretical perspective an MPC is a method to approximate a dynamic programming solution over a finite horizon [4].

There are two approaches to formulating the finite horizon optimal control problem of an MPC: direct and indirect methods [5–7]. In this paper we focus on the direct approach (which is the most popular) where the problem is first discretized and then transcribed directly to a constrained finite-dimensional optimization problem. This approach generally leads to a parameterized nonlinear programming problem of the form:

$$\begin{aligned} &\min_{\mathbf{X},\mathbf{U}} J(\mathbf{X},\mathbf{U};\mathbf{x}_0) \\ \text{subject to} \quad &\mathbf{x}_{k+1} = \bar{\mathbf{f}}(\mathbf{x}_k,\mathbf{u}_k;\Delta t),\ k=0,\dots,H-1 \\ &\mathbf{h}(\mathbf{X},\mathbf{U};\mathbf{x}_0) \leq 0. \end{aligned} \tag{1}$$

The ingredients of this problem are: the latest state feedback $\mathbf{x}_0$, a discrete plant model $\bar{\mathbf{f}}$ based on the sampling timestep Δt, a cost function J and a constraint vector $\mathbf{h}$ that captures constraints on the plant states or control actions, such as actuator saturation limits. The states and controls over the prediction horizon of H steps are denoted by $\mathbf{X} = [\mathbf{x}_0,\dots,\mathbf{x}_H]$ and $\mathbf{U} = [\mathbf{u}_0,\dots,\mathbf{u}_{H-1}]$, respec-

*Address all correspondence to this author, email: anson.maitland@uwaterloo.ca

tively. The control policy $\mathbf{U}$ is the primary variable of interest since the states are constrained by the model dynamics.

Efficiently solving NLP (1) is of primary concern when it comes to deploying an MPC on a real-world system. NLPs are NP-hard and require computationally expensive methods to solve [8, 9]. The real-time constraint and limited computational resources of a real-world system further increases the challenge for MPC deployment. Significant efforts in the MPC community have been devoted to generating fast algorithms and efficient automatic code generation schemes for MPC deployment. For example, the Real-Time Iteration scheme [10] is a tailored algorithm for MPC, CVXGEN [11] and FORCES [12, 13] are code generation tools that can deliver tailored interior point solvers and the ACADO Toolkit [14] can automatically generate efficient C code for MPC. By exploiting both the problem structure and knowledge of the embedded hardware one can achieve significant improvements in computation time.

We present a new quasi-Newton method, called the Restricted Newton Method (RNM), that can be used in place of Newton's Method (NM) within an NLP solver. This method is a hybrid of NM and Gradient Descent (GD). In summary, the updates of RNM are composed of two orthogonal components: the first is a Newton step restricted to a subspace and the second is a descent step along the perpendicular subspace. The choice of subspace and the maximal GD step length are parameters of the method. We present two strategies for selecting these parameters using basic machine learning methods.

Newton type methods that utilize projected Hessians to solve NLPs have been around for sometime. These methods focus on equality constrained problems and the selection of appropriate Hessian approximations on the complement of the null space of the constraint manifold [15–19]. In the RNM we introduce a linear constraint manifold artificially and include an orthogonal gradient step. In the work presented here we use an exact projected Hessian, but utilizing approximate ones, as is mostly done in the early work, opens up avenues for future work. One distinguishing feature of the RNM is the freedom to choose the constraint subspace. We present two methods to select this parameter nearly optimally (in terms of convergence rate) since the RNM would otherwise prove counterproductive to our goal of reducing the turnaround times (TATs) of a MPC.

This paper is organized as follows: we first present the RNM algorithm and a convergence analysis. We then discuss two parameter selection strategies that can be used to determine the convergence behaviour of RNM and demonstrate the effect of the parameters on a low-dimensional convex problem. Lastly, we apply the RNM method and parameter selection strategies to an autonomous vehicle problem with promising results.

THE RESTRICTED NEWTON METHOD

The Restricted Newton Method (RNM) is an optimization method that is a hybrid of Newton's Method (NM) and Gradient Descent (GD). Depending on the dimension of the subspace we can recover either the full Newton step or the standard GD step. In the following we consider the problem:

$$\min_{\mathbf{z}} \ \mathscr{L}(\mathbf{z}) \tag{2}$$

where $\mathbf{z} \in \mathbb{R}^l$. This problem represents the final Lagrangian minimization problem of an NLP.

ALGORITHM

The RNM is parameterized by two elements $\{\Pi_r \in \mathbb{R}^{l\times r} | \Pi_r^T \Pi_r = \mathbf{I}\}$ a matrix whose columns span a subspace of dimension r (Π_r can be thought of as the first r columns of some orthogonal matrix Π) and $\gamma > 0$ the maximal length of the GD step. We denote the orthogonal complement of Π_r by $\Pi_r^\perp \in \mathbb{R}^{l\times l-r}$. The RNM, displayed in Algorithm 1, begins at an initial guess $\mathbf{z}^{(0)}$ and proceeds until the termination criteria are satisfied.

Algorithm 1 Restricted Newton Method

procedure RNM($\mathbf{z}^{(0)}; \Pi_r, \gamma$)
  $k \leftarrow 0$
  repeat
    $\Delta\mathbf{z}_-^{(k)} \leftarrow -\Pi_r[\Pi_r^T \nabla^2 \mathscr{L}(\mathbf{z}^{(k)})\Pi_r]^{-1}\Pi_r^T \nabla \mathscr{L}(\mathbf{z}^{(k)})$
    $\Delta\mathbf{z}_\perp^{(k)} \leftarrow -(\mathbf{I} - \Pi_r\Pi_r^T)\nabla \mathscr{L}(\mathbf{z}^{(k)})$
    $\Delta\mathbf{z}_{RNM}^{(k)} \leftarrow \Delta\mathbf{z}_-^{(k)} + \gamma\Delta\mathbf{z}_\perp^{(k)}$
    $t^{(k)} \leftarrow$ LineSearch
    $\mathbf{z}^{(k+1)} \leftarrow \mathbf{z}^{(k)} + t^{(k)}\Delta\mathbf{z}_{RNM}^{(k)}$
    $k \leftarrow k+1$
  until Termination
  return $\mathbf{z}^{(k)}$
end procedure

Provided the LineSearch procedure returns a t satisfying the Wolfe conditions:

$$\mathscr{L}(\mathbf{z}^{(k)} + t^{(k)}\Delta\mathbf{z}_{RNM}^{(k)}) \leq \mathscr{L}(\mathbf{z}^{(k)}) + c_1 t^{(k)} \nabla \mathscr{L}(\mathbf{z}^{(k)})^T \Delta\mathbf{z}_{RNM}^{(k)} \tag{3}$$

$$\nabla \mathscr{L}(\mathbf{z}^{(k)} + t^{(k)}\Delta\mathbf{z}_{RNM}^{(k)})^T \Delta\mathbf{z}_{RNM}^{(k)} \geq c_2 \nabla \mathscr{L}(\mathbf{z}^{(k)})^T \Delta\mathbf{z}_{RNM}^{(k)} \tag{4}$$

for some $0 < c_1 < c_2 < 1$, then Algorithm 1 is globally convergent under mild assumptions. The following is a corollary of Zoutendijk's Theorem, see [20].

Theorem 0.1. (Global Convergence) *Suppose $\mathscr{L}$ is bounded below in $\mathbb{R}^l$ and continuously differentiable in an open set Z containing the level set $L := \{\mathbf{z} | \mathscr{L}(\mathbf{z}) \leq \mathscr{L}(\mathbf{z}^{(0)})\}$ where $\mathbf{z}^{(0)}$ is the initial guess and that $\nabla\mathscr{L}$ is Lipschitz continuous on Z. Further, if $\Pi_r^T \nabla^2 \mathscr{L}(\mathbf{z}^{(k)})\Pi_r$ are positive definite with a uniformly bounded condition number and $t^{(k)}$ satisfy the Wolfe conditions* (3) *&* (4) *then Algorithm 1 is globally convergent.*

If we include the stronger assumption that $\mathscr{L}$ is strongly convex so that $m\mathbf{I} \prec \nabla^2\mathscr{L} \prec M\mathbf{I}$ for some $0 < m < M$ then Algorithm 1 is convergent to a global minimum.

The RNM is a quasi-Newton method for which we can write the update as

$$\begin{aligned}\Delta\mathbf{z}_{RNM} &= -[\Pi_r[\Pi_r^T\nabla^2\mathscr{L}\Pi_r]^{-1}\Pi_r^T - \gamma(\mathbf{I} - \Pi_r\Pi_r^T)]\nabla\mathscr{L} \\ &= -[\Pi_r[[\Pi_r^T\nabla^2\mathscr{L}\Pi_r]^{-1} - \gamma\mathbf{I}]\Pi_r^T + \gamma\mathbf{I}]\nabla\mathscr{L} \\ &= -\Gamma(\Pi_r, \gamma)\nabla\mathscr{L}\end{aligned}$$

where $\Gamma \in \mathbb{R}^{l\times l}$ is the approximate inverse Hessian. We note that if $r = l$ so $\Pi_r = \Pi$ we recover the Newton step

$$\begin{aligned}\Gamma(\Pi, \gamma) &= [\Pi[[\Pi^T\nabla^2\mathscr{L}\Pi]^{-1} - \gamma\mathbf{I}]\Pi^T + \gamma\mathbf{I}] \\ &= [\nabla^2\mathscr{L}]^{-1}\end{aligned}$$

and if $r = 0$ so $\Pi_r = \emptyset$ we recover the gradient descent step $\Gamma(\emptyset, \gamma) = -\gamma\mathbf{I}$.

As compared to NM the computational savings of RNM comes from reducing the dimension of the linear solve step from l to r. However, we note that the RNM does add the overhead of computing a number of matrix products meaning that achieving any computational savings is dependent on the size of r. Provided $r \ll l$ then the asymptotic cost of the equivalent linear solve component in RNM is $O(rl^2)$ operations coming from the required matrix products compared to the $O(l^3)$ operations of the linear solve in NM. We note both methods have a shared cost of computing the gradient and Hessian once per iteration.

CONVERGENCE ANALYSIS

Assuming $\mathscr{L}$ is strongly convex and we use a backtracking line search in place of `LineSearch` then it is straightforward to show Algorithm 1 has at least a linear rate of convergence since one can view RNM as a steepest descent method in the $\Gamma^{1/2}$ norm [21]. Under some special choices of the RNM parameters we can prove superlinear convergence as a corollary of Theorem 3.5 in [20].

Theorem 0.2. (Superlinear Convergence) *Suppose $\mathscr{L} : \mathbb{R}^l \to \mathbb{R}$ is three times continuously differentiable and the sequence $\{\mathbf{z}^{(k)}\}$ generated by Algorithm 1, where the $t^{(k)}$ satisfy Wolfe's conditions* (3) *&* (4) *with $c_1 \leq 1/2$, converges to $\mathbf{z}^*$ such that $\nabla\mathscr{L}(\mathbf{z}^*) = 0$ and $\nabla^2\mathscr{L}(\mathbf{z}^*) \succ 0$. If Π, r, γ satisfy:*

$$\Pi_r^{\perp T}\nabla^2\mathscr{L}(\mathbf{z}^*)\Pi_r^{\perp} = \gamma^{-1}\mathbf{I} \tag{5}$$

then Algorithm 1 converges to $\mathbf{z}^*$ superlinearly with $t^{(k)} = 1$ for all k sufficiently large.

This result tells us that if $\Pi_r^{\perp}$ diagonalizes the last $l - r$ columns of the Hessian at the optimum and γ equals the corresponding eigenvalues, then the RNM steps approach full Newton steps in the limit resulting in superlinear convergence. We now do some analysis to discuss the convergence rate of RNM in a more general setting and what makes it advantageous for realtime MPC.

First, we consider the decrease in objective function value over a single RNM step. For brevity we denote the current iterate $\mathbf{z}$ and its successor $\mathbf{z}^+$. Further, we denote $\nabla\mathscr{L}_- = \Pi_r^T\nabla\mathscr{L}$ and $\nabla\mathscr{L}_\perp = \Pi_r^{\perp T}\nabla\mathscr{L}$. Assume $\|\nabla\mathscr{L}_-(\mathbf{z})\| \geq \eta$ for some $\eta > 0$ [1]. The backtracking line search exit condition is:

$$\begin{aligned}\mathscr{L}(\mathbf{z} + t\Delta\mathbf{z}_{RNM}) &\leq \mathscr{L}(\mathbf{z}) + \alpha t\nabla\mathscr{L}(\mathbf{z})^T\Delta\mathbf{z}_{RNM} \\ &\leq \mathscr{L}(\mathbf{z}) - \alpha t\lambda_-^2(1 + \kappa^2/\gamma)\end{aligned}$$

where $\alpha \in (0, 1/2), \beta \in (0, 1)$ are the line search parameters, $\lambda_-^2 = \nabla\mathscr{L}_-^T\mathbf{H}_1^{-1}\nabla\mathscr{L}_-$ and $\gamma^2\|\nabla\mathscr{L}_\perp\|^2 = \kappa^2\lambda_-^2$ for some $\kappa > 0$. The backtracking line search updates $t \leftarrow \beta t$ until the exit condition is satisfied. Strong convexity implies

$$\begin{aligned}\mathscr{L}(\mathbf{z} + t\Delta\mathbf{z}_{RNM}) &\leq \mathscr{L}(\mathbf{z}) + t\nabla\mathscr{L}(\mathbf{z})^T\Delta\mathbf{z}_{RNM} + \frac{M}{2}t^2\|\Delta\mathbf{z}_{RNM}\|^2 \\ &\leq \mathscr{L}(\mathbf{z}) - \lambda_-^2[t(1 + \kappa^2/\gamma) - \frac{M}{2}t^2(1/m + \kappa^2)]\end{aligned}$$

so the step size $\tilde{t} = \frac{m}{M}\frac{1+\kappa^2/\gamma}{1+m\kappa^2}$ satisfies the exit condition of the line search. Thus,

$$\begin{aligned}\mathscr{L}(\mathbf{z}^+) - \mathscr{L}(\mathbf{z}) &\leq -\alpha\beta\frac{m}{M}\frac{1 + \kappa^2/\gamma}{1 + m\kappa^2}\lambda_-^2(1 + \kappa^2/\gamma) \\ &\leq -\alpha\beta\eta^2\frac{m}{M^2}\frac{(1 + \kappa^2/\gamma)^2}{1 + m\kappa^2}\end{aligned}$$

since $\lambda_-^2 \geq (1/M)\|\nabla\mathscr{L}_-\|^2$. This decrease can be made largest by maximizing $\frac{(1+\kappa^2/\gamma)^2}{1+m\kappa^2}$ which occurs at $\gamma = 1/m$.

From the above analysis we have a basic criterion for trying to select good values of γ. In practise, m is often unknown but it can be approximated with adequate data.

[1] All norms are 2 norms unless stated otherwise.

The following analysis illustrates RNM's practical advantage when it comes to fast MPC. We shall see that, despite RNM's slow overall convergence rate, initially it can perform near quadratically on the subspace Π_r. Thus in realtime applications where the number of iterations of a solver may be upper bounded RNM can still achieve desirable results.

Assuming $\nabla^2\mathscr{L}$ is Lipschitz continuous with constant L so that

$$\|\nabla^2\mathscr{L}(\mathbf{z})-\nabla^2\mathscr{L}(\mathbf{z}')\|\leq L\|\mathbf{z}-\mathbf{z}'\|$$

for all $\mathbf{z},\mathbf{z}'$ in a neighbourhood of the minimum $\mathbf{z}^*$ and that $t=1$ satisfies the backtracking line search criterion then,

$$\begin{aligned}\|\nabla\mathscr{L}_-(\mathbf{z}^+)\| &= \|\Pi_r^T\nabla\mathscr{L}(\mathbf{z}+\Delta\mathbf{z}_{RNM})\|\\ &= \|\int_0^1\Pi_r^T\nabla^2\mathscr{L}(\mathbf{z}+t\Delta\mathbf{z}_{RNM})\Delta\mathbf{z}_{RNM}dt\\ &\quad-\Pi_r^T\Gamma^{-1}(\mathbf{z})\Delta\mathbf{z}_{RNM}\|\\ &\leq\int_0^1\|\Pi_r^T[\nabla^2\mathscr{L}(\mathbf{z}+t\Delta\mathbf{z}_{RNM})-\nabla^2\mathscr{L}(\mathbf{z})]\Delta\mathbf{z}_{RNM}\|dt\\ &\quad+\|\Pi_r^T[\nabla^2\mathscr{L}(\mathbf{z})-\Gamma^{-1}(\mathbf{z})]\Delta\mathbf{z}_{RNM}\|\\ &\leq(1+\nu^2)\frac{L}{2m^2}\|\nabla\mathscr{L}_-(\mathbf{z})\|^2+\nu\frac{M}{m}\|\nabla\mathscr{L}_-(\mathbf{z})\|\end{aligned}$$

where we have used the relations

$$\begin{aligned}\|\Pi_r^T[\nabla^2\mathscr{L}-\Gamma^{-1}]\Delta\mathbf{z}_{RNM}\|&\leq\gamma M\|\nabla\mathscr{L}_\perp\|\\ \|\Delta\mathbf{z}_{RNM}\|^2&\leq\frac{1}{m^2}\|\nabla\mathscr{L}_-\|^2+\gamma^2\|\nabla\mathscr{L}_\perp\|^2\\ \gamma^2\|\nabla\mathscr{L}_\perp\|^2&=\frac{\nu^2}{m^2}\|\nabla\mathscr{L}_-\|^2\end{aligned}$$

for some $\nu>0$. From the last inequality we can see that if ν is small (meaning that the $\gamma\Delta\mathbf{z}_\perp$ is negligible compared to $\Delta\mathbf{z}_-$) then the gradient in the chosen subspace shrinks quadratically provided $\|\nabla\mathscr{L}_-\|<\frac{m^2}{L}$. As ν increases (or as $\gamma\Delta\mathbf{z}_\perp$ increases in relative size) this quadratic convergence behaviour is diminished.

The preceding analysis illustrates the effect that the choice of subspace Π_r can have on the convergence behaviour of RNM. Essentially, Π_r is providing good conditioning of the gradient steps in some directions but not others since Π_r selects over which space the Hessian acts. A challenge for using RNM is selecting Π_r.

PARAMETER SELECTION

The convergence behaviour of RNM is dependent on the parameters Π_r,γ. In the previous section we presented a simple suggestion for selecting γ. We now propose a method for selecting these parameters offline. This method is essentially a feature extraction applied to MPC simulation data. Importantly, we shall see that the required training data can be rather impoverished in comparison to typical machine learning applications.

The basic algorithm is as follows:

1. Construct a snapshot matrix $\mathbf{S}\in\mathbb{R}^{l\times d(N)}$ by running the MPC over representative control scenarios with $N>l$ timesteps.
2. Apply a feature extraction method to the columns of $\mathbf{S}$ in order to extract $\Pi_r\in\mathbb{R}^{l\times r}$.

We now discuss two strategies for applying the above algorithm. Strategy A is to select $\mathbf{Z}=\begin{pmatrix}\Delta\mathbf{z}(t_1)&\cdots&\Delta\mathbf{z}(t_N)\end{pmatrix}\in\mathbb{R}^{l\times N}$ where $\Delta\mathbf{z}(t_i)=\mathbf{z}^*(t_i)-\mathbf{z}^*(t_{i-1})$ as the snapshot matrix. Then in the second step compute the singular value decomposition of the snapshot matrix $\mathbf{S}=\Pi\Sigma\mathbf{V}^T$ and then truncate Π to its first $r<l$ columns to form $\Pi_r\in\mathbb{R}^{l\times r}$. This strategy selects a Π_r that will achieve the greatest initial quadratic convergence behaviour. This basic method has been used by the authors in [22, 23] for MPC acceleration. In practice, a good value for γ was found by manual tuning and is typically small. It is known that γ is related to the singular values of the snapshot matrix $\mathbf{Z}$.

In strategy B we approximate the requirements of Theorem 0.2 by choosing $\Omega=\begin{pmatrix}\mathbf{O}(t_1)&\cdots&\mathbf{O}(t_N)\end{pmatrix}\in\mathbb{R}^{l\times rN}$ where $\mathbf{O}(t_i)\in\mathbb{R}^{l\times l}$ is the matrix composed of the ordered eigenvectors of $\nabla^2\mathscr{L}(\mathbf{z}^*(t_i))$ as our snapshot matrix. Then by applying a cluster analysis algorithm to the columns of Ω we can extract the r vectors that best capture the r largest eigenvectors. We can further use the eigenvalues corresponding to the snapshot data in order to approximate γ.

We demonstrate the role of the RNM parameter choices with the following sample problem. Consider the following two-dimensional convex optimization problem:

$$\min f(\mathbf{z})=e^{z_1+3z_2}+e^{z_1-3z_2}+e^{-z_1}. \tag{6}$$

We solve this problem using NM, GD and RNM with initial guess $\mathbf{z}^{(0)}=\begin{pmatrix}-1.054&0.707\end{pmatrix}^T$. Because this is a two-dimensional problem, we can parameterize the RNM subspace $\Pi_1=\begin{pmatrix}\sin(\theta)&\cos(\theta)\end{pmatrix}^T$ where $0\leq\theta<2\pi$ is the angle measured counter clockwise from the z_1 axis. We compare 2 versions of RNM against NM and GD in Fig. 1. These 2 versions approximate the parameter selections of strategies A and B. In all cases we used a backtracking linesearch with the same parameters.

In Fig. 2 we compare the convergence rates of the different solutions. By construction $\mathbf{z}^{(0)}-\mathbf{z}^*=\Pi_1(3\pi/4)$. Thus RNM with $\theta=3\pi/4,\gamma=0$ will lead to the exact solution with quadratic convergence. For small values of γ there will be orthogonal steps taken leading to a loss of quadratic convergence once the size of the steps along $\Pi_1(3\pi/4)$ reaches a sufficiently

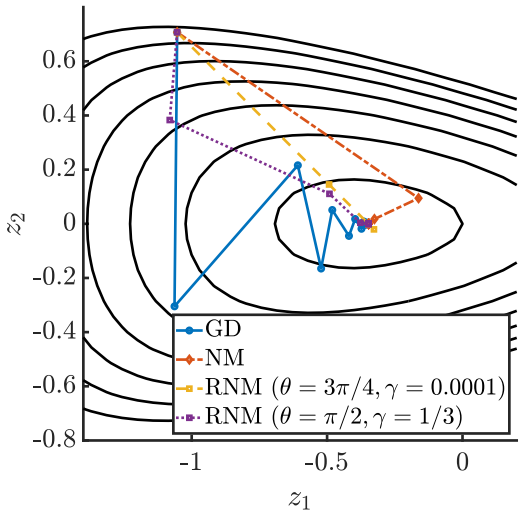


FIGURE 1: Iterates of optimization methods for problem (6). The contours of f are shown in sold black.

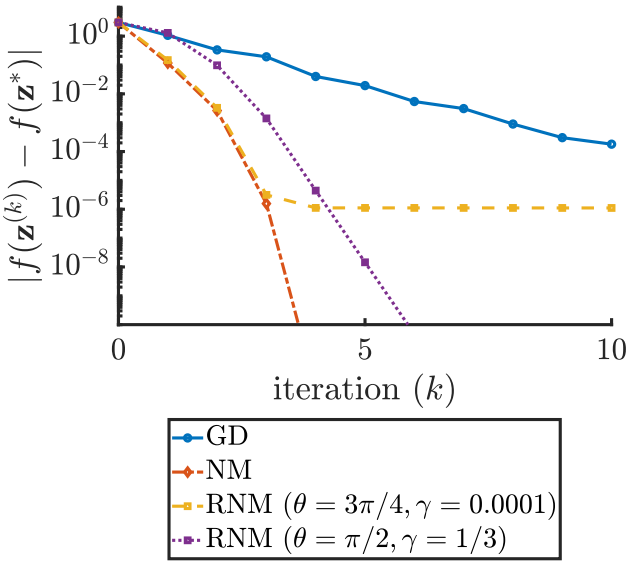
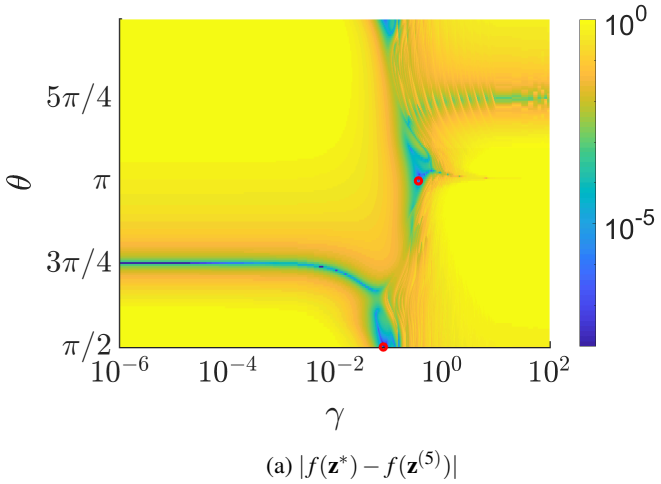
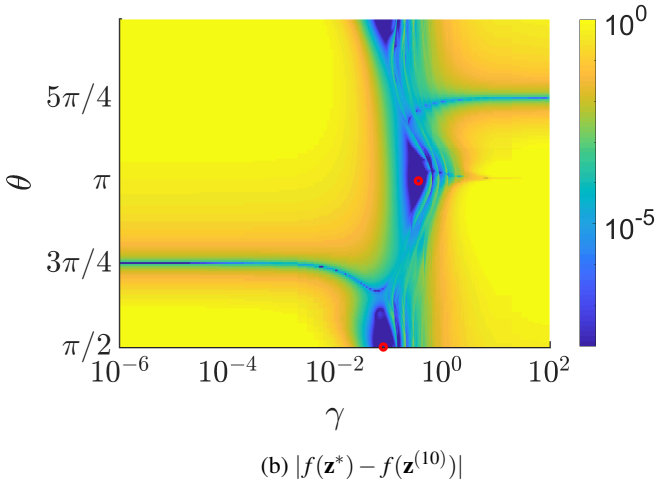


FIGURE 2: Convergence rate of optimization methods for problem (6).

small size, as discussed in the convergence analysis of the preceding section. This is observed in Fig. 2 after 3 iterations. From Fig. 1 we can also see that the steps are dominantly along the $\Pi_1(3\pi/4)$ direction. Further, as $\nabla^2 f(\mathbf{z}^*) = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$ where $\lambda_1 = 2.828, \lambda_2 = 12.728$ are the eigenvalues, then RNM with $\theta = \pi/2, \gamma = \lambda_1^{-1}$ satisfies the conditions of Theorem 0.2. In Fig. 2 we can observe the superlinear convergence of this RNM



(a) $|f(\mathbf{z}^*) - f(\mathbf{z}^{(5)})|$



(b) $|f(\mathbf{z}^*) - f(\mathbf{z}^{(10)})|$

FIGURE 3: Role of parameters in RNM convergence for problem (6) at 5 and 10 iterations, respectively.

method.

In Fig. 3 we display the effect of the parameters on RNM convergence with darker colours representing faster convergence. From Fig. 3a we can see that $\theta = 3\pi/4$ with small γ converges the fastest and further that this rapid convergence is highly sensitive to θ. After more iterations we can see from Fig. 3b that other parameter choices are able to catch up to the initially rapidly converging case $\theta = 3\pi/4$ with small γ. In particular we notice the parameter choices near $\theta = \pi/2, \gamma = \lambda_1^{-1}$ and $\theta = \pi, \gamma = \lambda_2^{-1}$ (highlighted by the red circles) yield best results. These points are those satisfying Theorem 0.2. Notably, the superlinear convergence behaviour is not as sensitive to the parameter choices as the initial quadratic convergence case, $\theta = 3\pi/4, \gamma = 0$. Interestingly, we note that the subspace perpendicular to $\theta = 3\pi/4$ yields more rapid convergence for large γ. This is because in this special case the gradient steps will

be taken directly towards the optimum leading to rapid (but not quadratic) convergence.

AUTONOMOUS VEHICLE PROBLEM

The goal of this MPC is to track a reference trajectory specified in terms of vehicle position and velocity. For the vehicle model we used a nonlinear, bicycle model that captures the planar motion of a vehicle assuming the pitch, heave, and roll are negligible. The model includes simple mappings representing the motor to relate the pedal inputs to wheel torques, and Pacejka tires to capture nonlinear slip dynamics [24]. The three control inputs are steering angle rate $w_\delta \in [-0.5, 0.5]$ (rad/s), total braking force $F_B \in [0, 15000]$ (N) and throttle $\phi \in [0,1](-)$. The model states of interest are the absolute position of the vehicle's center of gravity (x,y) (m) and speed v (m/s). We denote the state and controls by $\mathbf{x} = \begin{pmatrix} x & y & v & \beta & \psi & w_z & \delta \end{pmatrix}^T$ and $\mathbf{u} = \begin{pmatrix} w_\delta & F_B & \phi \end{pmatrix}^T$, respectively, where the hitherto unmentioned states are side slip angle β (rad), yaw angle ψ (rad), rate of change of yaw w_z (rad/s) and steering angle δ (rad). The model's equations of motion are

$$
\begin{aligned}
\dot{x} &= v\cos(\psi - \beta) \\
\dot{y} &= v\sin(\psi - \beta) \\
\dot{v} &= \frac{1}{m}[(F_{lr} - F_{Ax})\cos\beta + F_{lf}\cos(\delta+\beta) \\
&\qquad - (F_{sr} - F_{Ay})\sin\beta - F_{sf}\sin(\delta+\beta)] \\
\dot{\beta} &= w_z - \frac{1}{mv}[(F_{lr} - F_{Ax})\sin\beta + F_{lf}\sin(\delta+\beta) \\
&\qquad + (F_{sr} - F_{Ay})\cos\beta + F_{sf}\cos(\delta+\beta)] \\
\dot{\psi} &= w_z \\
\dot{w}_z &= \frac{1}{I_{zz}}\left(F_{sf} l_f \cos\delta - F_{sr} l_r - F_{Ay} e_{SP} + F_{lf} l_f \sin\delta\right) \\
\dot{\delta} &= w_\delta
\end{aligned}
\tag{7}
$$

where the forces are given by

$$
\begin{aligned}
F_{sf} &= D_f \sin(C_f \arctan(B_f\alpha_f - E_f(B_f\alpha_f - \arctan(B_f\alpha_f)))) \\
F_{sr} &= D_r \sin(C_r \arctan(B_r\alpha_r - E_r(B_r\alpha_r - \arctan(B_r\alpha_r)))) \\
F_{Ax} &= \frac{1}{2} c_w \rho A v^2 + D_x \\
F_{Ay} &= D_y \\
F_{lf} &= -\frac{2}{3} F_B - f_R \frac{m l_r g}{l_f + l_r} \\
F_{lr} &= \frac{\zeta M_{mot}}{R} - \frac{1}{3} F_B - f_R \frac{m l_f g}{l_f + l_r}
\end{aligned}
$$

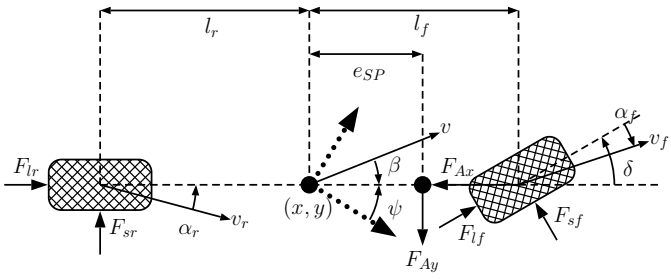


FIGURE 4: Vehicle model schematic.

and the intermediate terms are

$$
\begin{aligned}
\alpha_f &= \delta - \arctan\left(\frac{l_f w_z - v\sin\beta}{v\cos\beta}\right) \\
\alpha_r &= \arctan\left(\frac{l_r w_z + v\sin\beta}{v\cos\beta}\right) \\
M_{mot} &= (1 - e^{-3\phi})\left(-37.8 + 1.54\frac{\zeta v}{R} - 0.0019\left(\frac{\zeta v}{R}\right)^2\right) \\
&\qquad + e^{-3\phi}\left(-34.9 - 0.04775\frac{\zeta v}{R}\right) \\
f_R &= 9\times10^{-3} + 7.2\times10^{-5} v + 5.038848\times10^{-10} v^4
\end{aligned}
$$

which are the front slip angle, rear slip angle, motor torque and speed-dependent rolling resistance, respectively. The parameter $\zeta = 5.2003$ comes from a fixed gear input. We display a schematic of the model in Fig. 4. More model details including the parameters $(m, I_{zz}, l_{f,r}, e_{SP}, B_{f,r}, C_{f,r}, D_{f,r}, E_{f,r}, c_W, \rho, A, g, R)$ can be found in [25, 26]. The changes to the published model are a fixed gear input and the inclusion of body forces $D_{x,y}$ to model external disturbances, such as wind.

To formulate our MPC we utilized a direct transcription approach with control parameterization [27]. At each timestep our MPC solves the following NLP

$$
\mathbf{U}^* = \underset{\mathbf{U}\in\mathscr{U}}{\arg\min}\ J(\mathbf{U}; \mathbf{x}_0, \mathbf{X}^r) \tag{8}
$$

where $\mathbf{U} = [\mathbf{u}_0, \ldots, \mathbf{u}_{H-1}]$ is the control sequence over the prediction horizon comprised of H timesteps, $\mathscr{U}$ captures the control input limits, $\mathbf{x}_0$ is the most recent measurement (or estimate) of the vehicle's state and $\mathbf{X}^r = [\mathbf{x}_1^r, \ldots, \mathbf{x}_H^r]$ is the reference trajectory over the horizon. We select the cost to be

$$
J(\mathbf{U}; \mathbf{x}_0, \mathbf{X}^r) = \sum_{k=1}^{H} \|\mathbf{x}_k - \mathbf{x}_k^r\|_{\mathbf{Q}}^2 + \|\mathbf{u}_{k-1}\|_{\mathbf{R}}^2 + \sum_{k=1}^{H-1} \|\Delta\mathbf{u}_k\|_{\Delta\mathbf{R}}^2
$$

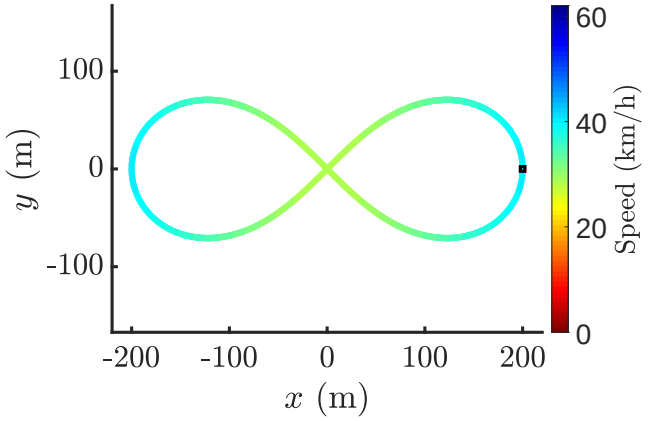


FIGURE 5: Reference trajectory for snapshot data generation. Trajectory begins and ends at □.

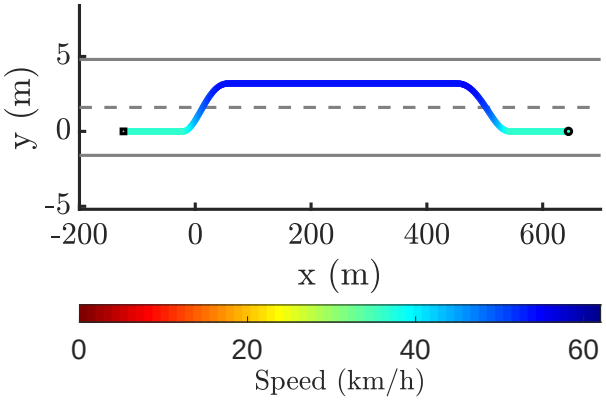


FIGURE 6: Reference trajectory for overtaking maneuver. Trajectory begins on the left at □ and ends at ○.

where $\mathbf{x}_k = \mathbf{x}_k(\mathbf{U};\mathbf{x}_0), k \geq 1$ are predicted vehicle states defined using an explicit Euler discretization of the nominal vehicle model

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \tag{9}$$

with Δt being the timestep, $\mathbf{f}$ is the vector field of the vehicle model that captures (7), $\Delta\mathbf{u}_k = (\mathbf{u}_k - \mathbf{u}_{k-1})/\Delta t$ and $\mathbf{Q}, \mathbf{R}, \Delta\mathbf{R}$ are tunable diagonal weighting matrices. This choice of cost function balances two objectives: to minimize the deviation from the reference trajectory and to minimize the inputs and their rates of action. This MPC design does not attempt to model the disturbance forces.

In our implementation we used the symbolic single-shooting approach to form the problem Lagrangian and generate its exact derivatives in Maple [23]. This yields an NLP of minimal dimension by incorporating the state constraints within the cost. Optimized code to compute the gradient and Hessian were converted to MEX files and utilized by our solvers in Matlab. We selected $H = 10$, $\Delta t = 0.05$s.

In Fig. 5 we display the reference trajectory used to generate our snapshot data. This is a gentle figure eight maneuver that begins and ends at $(200, 0)$. External forces $D_{x,y} \sim 100 \times \mathcal{N}(\text{N})$ where $\mathcal{N}$ represents the standard normal distribution, were applied to the model during the snapshot generation simulation. Only a single run of this reference trajectory using an undamped NM was used to generate all the snapshot data used in the following sections. In comparison to typical machine learning type methods this training data is significantly smaller than what is commonly needed to achieve desirable results.

In Fig. 6 we display the reference trajectory used to compare an MPC using RNM. This trajectory is an overtaking maneuver at urban driving speeds. External forces of $D_{x,y} \sim 10 \times \mathcal{N}(\text{N})$ were applied during the different simulations.

We compared different MPCs using two versions of the RNM strategy against a baseline NM solver. The first selected Π_r using strategy A and the second using strategy B. We denote these solutions RNM A and RNM B. In both cases γ was tuned manually to yield desirable results based on an approximation of $1/m$. For RNM A with $\gamma = 5 \times 10^{-7}$ we were able to achieve good results with $r = 2$ and for RNM B with $\gamma = 2 \times 10^{-7}$ we were able to use $r = 4$. This is rather significant reductions as the original problem is 30 dimensional. Selecting r by analytical means is an open problem. The preceding values of r were selected as the smallest values that did not cause the MPC to fail. In the interest of reduced TATs we fixed the number of iterations at 5 and set $t^{(k)} = 1$ in all cases.

From Fig. 7 we can observe the excellent tracking behaviour of the MPCs using RNM as compared to the one using NM. Relative to the lane width (3.2 m) or the mean speed of the reference trajectory (46.4 km/h) the RNM results are all within 1% of the baseline solution.

In Fig. 8 we display the different control responses of the different MPCs. One can clearly see the different control responses about the lane change portions of the maneuver. These portions of the reference are also where the tracking response differs most greatly. The steering rate inputs are quite similar but the throttle/brake response differs significantly. This is largely due to the selection of Π_r since the control response of the RNM solutions are almost restricted to the column space of Π_r. We can see that RNM B has more variance in its throttle/brake response compared to RNM A. This is most likely due to the different levels of dimensional reduction achieved in each case.

In our implementation we took advantage of the small values

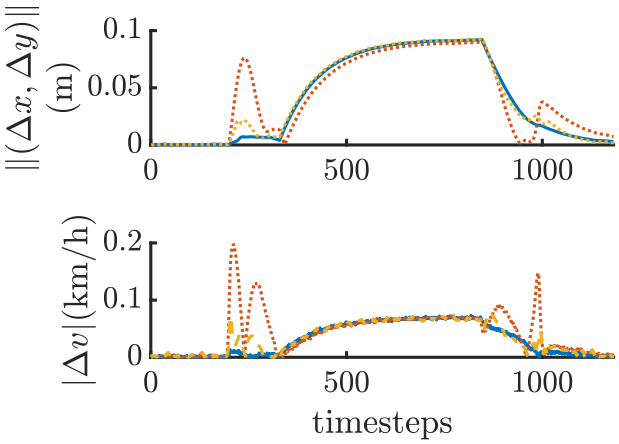


FIGURE 7: Comparison of tracking error between MPCs using NM (solid blue line), RNM A (red dotted line) and RNM B (yellow dashed line).

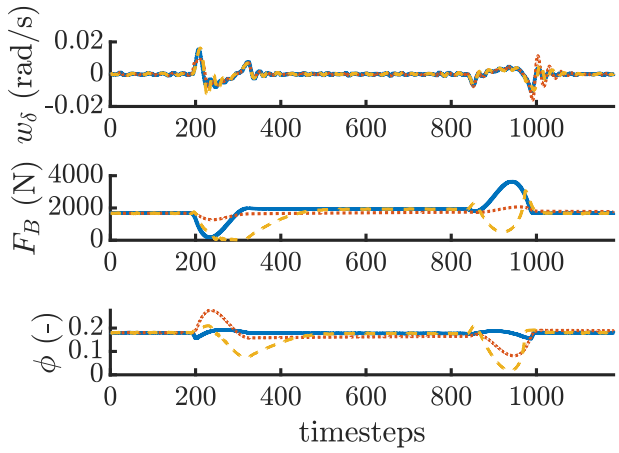


FIGURE 8: Comparison of control inputs between MPCs using NM (solid blue line), RNM A (red dotted line) and RNM B (yellow dashed line).

of r and replaced the linear solve of RNM by a symbolic linear solver that could be optimized along with the matrix operations in order to generate code to compute $\Delta \mathbf{z}_{RNM}$. The computation times are summarized in Tab. 1. All computation times were measured on a desktop computer with an Intel(R) Core(TM) i7-4790 CPU running MATLAB 2019a. For the baseline NM we used the LAPACK `matrixDivide` operation converted to a MEX function. The computational cost in this case is solving a 30×30 linear system. The computation times of RNM A and RNM B include the symbolic linear solve step as well as the matrix multiplications. These matrix multiplications dominate

	Computation Time (μs)	Acceleration
NM	29.0	–
RNM A	13.1	2.22×
RNM B	22.8	1.28×

TABLE 1: Computational cost of solver step (reported value is the median value of > 1000 sample computations).

the cost of RNM since the symbolic linear solve for $r \leq 4$ takes $< 3\mu$s. For this problem, computing the gradient and Hessian takes $\approx 50\mu$s. The overall savings per iteration is thus only a fraction of the accelerations reported in Tab. 1.

CONCLUSIONS

The RNM is a low-level optimization method that can be incorporated within NLP methods for NMPC applications. The convergence behaviour of RNM is highly dependent on the selection of two parameters: a linear constraint manifold and a step size. A reduction in computational complexity can be effected by judicious selection of these parameters as compared to a standard NM. We outline two strategies to select these parameters using basic machine learning techniques.

Future work on this method should focus on improving the parameter selection methods. In particular, can we find an analytic method to determine the dimension of the subspace r? And how sensitive are the solutions to Π and γ? This is a component of the larger question of the robustness of NMPC using RNM to the choice of snapshot. More NMPC applications should be examined to answer these questions — particularly those of larger dimension.

New parameter selection methods should also be considered. For example, if we enforce the constraint that Π_r is sparse, such as a move blocking matrix, then further computational savings can be had since the necessary matrix multiplications of RNM would be cheaper. Another possibility would be to allow the RNM parameters be functions of the optimization problem parameters or even the optimization variables themselves. In this way the parameters would no longer be constant and the machine learning problem would be significantly more challenging. One possibility to tackling this problem would be to use nonlinear machine learning techniques in order to learn the manifold whose tangent space at $\mathbf{z}'$ is spanned by a desirable subspace $\Pi_r(\mathbf{z}')$.

It is the authors' hope that this is a step forwards in the fruitful intersection of machine learning and optimization for MPC.

ACKNOWLEDGMENT

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Ontario Centres of Excellence (OCE) and Maplesoft.

REFERENCES

[1] Del Re, L., Allgöwer, F., Glielmo, L., Guardiola, C., and Kolmanovsky, I., 2010. "Automotive model predictive control". *Lecture Notes in Control and Information Science*.

[2] Qin, S. J., and Badgwell, T. A., 1997. "An overview of industrial model predictive control technology". In AIche Symposium Series, Vol. 93, New York, NY: American Institute of Chemical Engineers, 1971-c2002., pp. 232–256.

[3] Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O., 2000. "Constrained model predictive control: Stability and optimality". *Automatica,* ***36***(6), pp. 789–814.

[4] Rawlings, J. B., and Mayne, D. Q., 2009. *Model predictive control: Theory and design*. Nob Hill Pub.

[5] Rao, A. V., 2014. "Trajectory optimization: a survey". In *Optimization and Optimal Control in Automotive Systems*. Springer, pp. 3–21.

[6] Biegler, L. T., 2007. "An overview of simultaneous strategies for dynamic optimization". *Chemical Engineering and Processing: Process Intensification,* ***46***(11), pp. 1043–1053.

[7] Kameswaran, S., and Biegler, L. T., 2006. "Simultaneous dynamic optimization strategies: Recent advances and challenges". *Computers and Chemical Engineering,* ***30***(10), pp. 1560–1575.

[8] Garey, M. R., and Johnson, D. S., 2002. *Computers and Intractability*, Vol. 29. wh freeman New York.

[9] Murty, K. G., and Kabadi, S. N., 1987. "Some NP-complete problems in quadratic and nonlinear programming". *Mathematical Programming,* ***39***(2), pp. 117–129.

[10] Diehl, M., Bock, H. G., and Schlöder, J. P., 2005. "A real-time iteration scheme for nonlinear optimization in optimal feedback control". *SIAM Journal on Control and Optimization,* ***43***(5), pp. 1714–1736.

[11] Mattingley, J., and Boyd, S., 2009. "Automatic code generation for real-time convex optimization". *Convex Optimization in Signal Processing and Communications*, pp. 1–41.

[12] Domahidi, A., Zgraggen, A. U., Zeilinger, M. N., Morari, M., and Jones, C. N., 2012. "Efficient interior point methods for multistage problems arising in receding horizon control". In Decision and Control (CDC), 2012 IEEE 51st Annual Conference on, IEEE, pp. 668–674.

[13] Zanelli, A., Domahidi, A., Jerez, J., and Morari, M., 2017. "FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs". *International Journal of Control*, pp. 1–17.

[14] Houska, B., Ferreau, H. J., and Diehl, M., 2011. "Acado toolkitan open-source framework for automatic control and dynamic optimization". *Optimal Control Applications and Methods,* ***32***(3), pp. 298–312.

[15] Coleman, T. F., and Conn, A. R., 1982. "Nonlinear programming via an exact penalty function: Global analysis". *Mathematical Programming,* ***24***(1), pp. 137–161.

[16] Coleman, T. F., and Conn, A. R., 1982. "Nonlinear programming via an exact penalty function: asymptotic analysis". *Mathematical Programming,* ***24***(1), pp. 123–136.

[17] Coleman, T. F., and Conn, A. R., 1984. "On the local convergence of a quasi-Newton method for the nonlinear programming problem". *SIAM Journal on Numerical Analysis,* ***21***(4), pp. 755–769.

[18] Nocedal, J., and Overton, M. L., 1985. "Projected Hessian updating algorithms for nonlinearly constrained optimization". *SIAM Journal on Numerical Analysis,* ***22***(5), pp. 821–850.

[19] Byrd, R. H., 1990. "On the convergence of constrained optimization methods with accurate Hessian information on a subspace". *SIAM Journal on Numerical Analysis,* ***27***(1), pp. 141–153.

[20] Wright, S. J., and Nocedal, J., 1999. *Numerical Optimization*, Vol. 2. Springer New York.

[21] Boyd, S., and Vandenberghe, L., 2004. *Convex Optimization*. Cambridge university press.

[22] Maitland, A., and McPhee, J., 2017. "Improving model predictive controller turnaround time using restricted Lagrangians". In Decision and Control (CDC), 2017 IEEE 56th Annual Conference on, IEEE, pp. 3811–3816.

[23] Maitland, A., and McPhee, J., submitted. "Accelerating model predictive control turnaround times using restricted Lagrangians". *IEEE Transactions on Control Systems Technology*.

[24] Pacejka, H. B., and Bakker, E., 1992. "The magic formula tyre model". *Vehicle System Dynamics,* ***21***(S1), pp. 1–18.

[25] Gerdts, M., 2005. "Solving mixed-integer optimal control problems by branch&bound: a case study from automobile test-driving with gear shift". *Optimal Control Applications and Methods,* ***26***(1), pp. 1–18.

[26] Kirches, C., Sager, S., Bock, H. G., and Schlöder, J. P., 2010. "Time-optimal control of automobile test drives with gear shifts". *Optimal Control Applications and Methods,* ***31***(2), pp. 137–153.

[27] Rao, A. V., 2009. "A survey of numerical methods for optimal control". *Advances in the Astronautical Sciences,* ***135***(1), pp. 497–528.