SPOCK: A proximal method for multistage risk-averse optimal control problems ⋆

Alexander Bodard [**] **Ruairi Moran** [*] **Mathijs Schuurmans** [**]
Panagiotis Patrinos [**] **Pantelis Sopasakis** [*]

[*] *Queen's University Belfast, School of Electronics, Electrical Engineering and Computer Science, BT9 5AH, Northern Ireland, UK. (emails: {**rmoran05**, **p.sopasakis**}**@qub.ac.uk**)*
[**] *KU Leuven, Department of Electrical Engineering, Kasteelpark Arenberg 10, 3001 Leuven, Belgium. (emails: {**alexander.bodard**, **mathijs.schuurmans**, **panos.patrinos**}**@esat.kuleuven.be**)*

Abstract: Risk-averse optimal control problems have gained a lot of attention in the last decade, mostly due to their attractive mathematical properties and practical importance. They can be seen as an interpolation between stochastic and robust optimal control approaches, allowing the designer to trade-off performance for robustness and vice-versa. Due to their stochastic nature, risk-averse problems are of a very large scale, involving millions of decision variables, which poses a challenge in terms of efficient computation. In this work, we propose a splitting for general risk-averse problems involving linear dynamical systems and show how to efficiently compute iterates on a GPU-enabled hardware. Moreover, we propose SPOCK — a new algorithm that utilizes the proposed splitting and takes advantage of the SuperMann scheme combined with fast directions from Anderson's acceleration method for enhanced convergence speed. We implement SPOCK in `Julia` as an open-source solver, which is amenable to warm-starting and massive parallelization.

Keywords: Stochastic optimal control problems, large-scale optimization problems, modeling for control optimization, risk-averse optimal control.

1. BACKGROUND AND MOTIVATION

There are two main ways to formulate optimal control problems (OCPs) for uncertain systems: (i) the *stochastic/risk-neutral* (Cinquemani et al., 2011) approach where we assume, naively, that we have perfect information about the distributions of the involved random disturbances, and (ii) the *robust/minimax* approach where we ignore any statistical information that is usually available (Rawlings et al., 2017, Ch. 3). Another way is the risk-averse formulation, which takes into account *inexact* and *data-driven* statistical information (Wang and Chapman, 2022) using risk measures. A risk measure quantifies the magnitude of the right tail of a random cost, allowing the designer to choose a balance between the risk-neutral and the worst-case scenario. We restrict our interest to *coherent* risk measures (Shapiro et al., 2021) because of the desirable mathematical properties they possess.

⋆ The first two authors contributed equally to this paper. The work of R. Moran and P. Sopasakis has received support by EPSRC under the Queen's Collaborative Award "BotDozer: GPU-accelerated model predictive control for autonomous heavy equipment", which is co-funded by EquipmentShare. The work of A. Bodard, M. Schuurmans and P. Patrinos has received support by: the Research Foundation Flanders (FWO) research projects G081222N, G033822N, G0A0920N; EU'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 and the Fonds Wetenschappelijk Onderzoek - Vlaanderen under EOS project no 30468160 (SeLMA).

The challenge that multistage risk-averse optimal control problems (RAOCPs) bring, is that their cost functions involve the composition of several nonsmooth risk measures.

Risk measures have been widely used in practice and are gradually becoming popular, e.g., in reinforcement learning (Chow et al., 2017, 2015), in MPC (Sopasakis et al., 2019a) for microgrids (Hans et al., 2019), collision avoidance (Dixit et al., 2022; Schuurmans et al., 2020), energy management systems (Maree et al., 2021) and air-ground rendezvous (Barsi Haberfeld, 2021).

RAOCPs are typically solved using stochastic dual dynamic programming (Pereira and Pinto, 1991; Shapiro, 2011) approaches such as (da Costa and Leclère, 2021; Shapiro et al., 2013), however, these turn out to be slow. Authors in Sopasakis et al. (2019c) proposed an approach that allows us to use more efficient out-of-the-box optimization methods and software such as GUROBI (Gurobi Optimization, LLC, 2018) and MOSEK (Andersen and Andersen, 2000). These solvers use interior-point methods (Nemirovski and Todd, 2008), which usually do not scale well with the size of the problem and are hard to warm-start (Yildirim and Wright, 2002). However, risk-averse problems possess a rich structure that we can exploit to devise very efficient and massively parallelizable methods that can run on lockstep architectures such as GPUs (Oh et al., 2008).

We propose a look to first-order methods because they have simple iterates that are amenable to parallelization. In addition, the nonsmooth nature of the cost functions of risk-averse problems prohibits the use of any method that requires gradients. The Chambolle-Pock method (Chambolle and Pock, 2011) is chosen because it is gradient-free, it scales well with the problem size, it is easy to warm-start, it has a provably decent $\mathcal{O}(1/k)$ convergence rate which is similar to the celebrated alternating direction method of multipliers (ADMM) (Parikh and Boyd, 2014), and it is simpler than ADMM as it requires only two invocations of the problem's linear operator instead of four. Furthermore, we can accelerate the Chambolle-Pock method by combining it with SuperMann (Themelis and Patrinos, 2019), a Newton-type algorithm for finding fixed points of nonexpansive operators. We propose the resulting algorithm, SuperMann with Chambolle-Pock (SPOCK).

Due to these advantages, proximal algorithms have caught the attention of researchers in the area of control (Stella et al., 2017; Stathopoulos et al., 2016; O'Donoghue et al., 2013). Still, GPU computation is mostly underused in optimal control problems. Recent successful attempts in stochastic control are (Chowdhury and Subramani, 2022; Sampathirao et al., 2015).

The contributions of this paper are, (i) we propose a problem splitting, for general risk-averse problems involving linear dynamical systems, that allows us to apply the Chambolle Pock method and SuperMann, and (ii) an implementation of a new solver in `Julia`, and simulations that demonstrate the effectiveness of the method. The novel open-source solver SPOCK is amenable to warm-starting and massive parallelization.

2. NOTATION

Let $\mathbb{N}_{[k_1,k_2]}$ denote the integers in $[k_1, k_2]$, let 1_n and 0_n refer to n-dimensional column vectors of ones and zeroes respectively and let I_n be understood as the n-dimensional identity matrix. For $z \in \mathbb{R}^n$ let $[z]_+ = \mathbf{max}\{0, z\}$, where the max is taken element-wise. We denote the transpose of a matrix A by $A^\intercal$. The adjoint of a linear operator $L : \mathbb{R}^n \to \mathbb{R}^m$ is the operator $L^* : \mathbb{R}^m \to \mathbb{R}^n$ that satisfies $y^\intercal Lx = x^\intercal L^* y$ for all $x \in \mathbb{R}^n, y \in \mathbb{R}^m$. The dual cone $\mathcal{K}^*$ of a closed convex cone $\mathcal{K} \subseteq \mathbb{R}^n$ is the set $\mathcal{K}^* = \{y \in \mathbb{R}^n | y^\intercal x \geq 0, \forall x \in \mathcal{K}\}$. The relative interior of $\mathcal{K}$ is denoted by $\mathbf{ri}\ \mathcal{K}$. We denote a second order cone of dimension d by SOC_d and a translated second order cone by $\mathrm{SOC}_d + a$ where $a \in \mathbb{R}^d$. The indicator of a convex set $\mathcal{K}$ is defined as $\delta_\mathcal{K}(x) = 0$, if $x \in \mathcal{K}$, and $\delta_\mathcal{K}(x) = \infty$, otherwise. The set of fixed points of an operator $T : \mathbb{R}^n \to \mathbb{R}^n$ is denoted $\mathbf{fix}\ T = \{v \in \mathbb{R}^n \mid v = T(v)\}$. Define $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$. A function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is called *lower semicontinuous* (lsc) if its sublevel sets, $\{x \mid f(x) \leq \alpha\}$, are closed. The domain of a function f is $\mathbf{dom}\, f = \{x \mid f(x) < \infty\}$. The subdifferential of a convex function f is $\partial f(x) = \{u \in \mathbb{R}^n \mid \forall y \in \mathbb{R}^n, (y - x)^\intercal u + f(x) \leq f(y)\}$. The proximal operator of a proper lsc convex function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ with parameter $\alpha > 0$ is

$$\mathbf{prox}_{\alpha f}(x) = \underset{v}{\mathbf{arg\,min}} \left\{ f(v) + \tfrac{1}{2\alpha}\|v - x\|_2^2 \right\}. \tag{1}$$

Note that superscript i denotes scenario tree nodes, superscript (k) denotes algorithm iterations, and subscript t denotes time steps.

3. THE CHAMBOLLE-POCK METHOD

The Chambolle-Pock (CP) method is a proximal method that can be used to solve optimization problems of the following form (Ryu and Boyd, 2016, p.32)

$$\mathbb{P} : \underset{z \in \mathbb{R}^{n_z}}{\mathbf{minimize}}\ f(z) + g(Lz), \tag{2}$$

where $L : \mathbb{R}^{n_z} \to \mathbb{R}^{n_\eta}$ is a linear operator, and f and g are proper closed convex functions on $\mathbb{R}^{n_z}$ and $\mathbb{R}^{n_\eta}$, respectively. We assume strong duality, that is, $\mathbf{ri\,dom}\, f \cap \mathbf{ri\,dom}\, g(L) \neq \emptyset$, throughout this paper. The primal-dual optimality conditions of (2) consist in determining a pair of $z \in \mathbb{R}^{n_z}$, $\eta \in \mathbb{R}^{n_\eta}$ such that

$$0 \in \partial f(z) + L^*\eta, \tag{3a}$$
$$Lz \in \partial g^*(\eta). \tag{3b}$$

Our objective is to determine an approximate solution (z, η) such that

$$\xi_p \in \partial f(z) + L^*\eta, \tag{4a}$$
$$Lz + \xi_d \in \partial g^*(\eta), \tag{4b}$$

for some $\xi_p \in \mathbb{R}^{n_z}$ and $\xi_d \in \mathbb{R}^{n_\eta}$ of sufficiently small norm.

The CP method (Chambolle and Pock, 2011) recursively applies the firmly nonexpansive (FNE) operator T, where $(z^{(k+1)}, \eta^{(k+1)}) = T(z^{(k)}, \eta^{(k)})$, that is,

$$\begin{bmatrix} z^{(k+1)} \\ \eta^{(k+1)} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{prox}_{\alpha f}(z^{(k)} - \alpha L^* \eta^{(k)}) \\ \mathbf{prox}_{\alpha g^*}(\eta^{(k)} + \alpha L(2z^{(k+1)} - z^{(k)})) \end{bmatrix}}_{T(z^{(k)}, \eta^{(k)})}, \tag{5}$$

which is a generalized proximal point method with preconditioning operator

$$M(z, \eta) := \begin{bmatrix} I & -\alpha L^* \\ -\alpha L & I \end{bmatrix} \begin{bmatrix} z \\ \eta \end{bmatrix}. \tag{6}$$

The CP method converges if a solution exists and $0 < \alpha\|L\| < 1$, where $\|L\|$ is the operator norm $\|L\| = \mathbf{max}\{\|Lz\| : z \in \mathbb{R}^{n_z}, \|z\| \leq 1\}$.

We define the residual operator $R := \mathrm{id} - T$, where id is the identity operator. We may partition the residual into its primal and dual parts as $R^{(k)} = (z^{(k)}, \eta^{(k)}) - T(z^{(k)}, \eta^{(k)}) = (r_z^{(k)}, r_\eta^{(k)})$. We can show that the quantities $\xi_p^{(k)} = \frac{1}{\alpha} r_z^{(k)} - L^* r_\eta^{(k)}$, and $\xi_d^{(k)} = \frac{1}{\alpha} r_\eta^{(k)} - L r_z^{(k)}$ satisfy (4). We define $\xi_{(k)} = \mathbf{max}(\|\xi_p^{(k)}\|_\infty, \|\xi_d^{(k)}\|_\infty)$ and terminate when $\xi_{(k)} \leq \mathbf{max}(\epsilon_{\mathrm{abs}}, \epsilon_{\mathrm{rel}} \xi_{(0)})$, for some absolute and relative tolerances ϵ_{abs} and ϵ_{rel}, respectively. This termination criterion is akin to that used in (Sopasakis et al., 2019b).

The CP oracle consists of $\mathbf{prox}_{\alpha f}$, $\mathbf{prox}_{\alpha g^*}$, L and L^*. The CP algorithm has similar convergence properties to the celebrated ADMM, but requires only two invocations of L and L^* instead of four.

4. MULTISTAGE NESTED RISK-AVERSE OCPS

In this section we introduce the risk-averse optimal control problem on a scenario tree. We consider the following discrete-time linear dynamical system

$$x_{t+1} = A(w_t)x_t + B(w_t)u_t, \tag{7}$$

for $t \in \mathbb{N}_{[0,N-1]}$, with state variable $x_t \in \mathbb{R}^{n_x}$ and control input $u_t \in \mathbb{R}^{n_u}$. The random disturbance $w_t \in \{1, \dots, W\}$ is finite valued; for example, it can be an iid process or a Markov chain.

4.1 Scenario trees

A scenario tree is a representation of the dynamics of the system in Equation (7) over a finite number of stages given that the control actions are determined based on the system states in a causal fashion (Høyland and Wallace, 2001; Dupačová et al., 2000), as depicted in Fig. 1. Each time period is called a *stage*, t, and the number of time periods is called the *horizon*, N. Each possible realization of Equation (7) at each stage is called a *node*. We enumerate the nodes of a tree with i, where $i = 0$ is the *root* node which corresponds to the initial state of the system. The nodes at subsequent stages for $t \in \mathbb{N}_{[0,N]}$ are denoted by $\mathbf{nodes}(t)$. Let $\mathbf{nodes}(t_1, t_2) = \cup_{t=t_1}^{t_2} \mathbf{nodes}(t)$, where $0 \le t_1 \le t_2 \le N$. The unique *ancestor* of a node $i \in \mathbf{nodes}(1, N)$ is denoted by $\mathbf{anc}(i)$ and the set of *children* of $i \in \mathbf{nodes}(t)$ for $t \in \mathbb{N}_{[0,N-1]}$ is $\mathbf{ch}(i) \subseteq \mathbf{nodes}(t+1)$. Each node is associated with a probability $\pi^i > 0$ of occurring, where $\sum_{i \in \mathbf{nodes}(t)} \pi^i = 1$ for $t \in \mathbb{N}_{[0,N]}$.

Dynamics. The system dynamics on the scenario tree is described by

$$x^{i+} = A^{i+} x^i + B^{i+} u^i, \tag{8}$$

where $i \in \mathbf{nodes}(0, N-1)$ and $i_+ \in \mathbf{ch}(i)$. The state x^0 at the root node is assumed to be known.

Constraints. Here we assume that the states and inputs must satisfy the random convex constraints

$$\Gamma_x^i x^i + \Gamma_u^i u^i \in C^i, \tag{9a}$$

$$\Gamma_N^j x^j \in C_N^j, \tag{9b}$$

for $i \in \mathbf{nodes}(0, N-1)$ and $j \in \mathbf{nodes}(N)$, where C^i and C_N^j are closed convex sets, Γ_x^i and Γ_u^i are the state-input constraint matrices and Γ_N^j is the terminal state constraint matrix.

Quadratic costs. In this paper we consider quadratic cost functions. On the scenario tree, for $i \in \mathbf{nodes}(0, N-1)$ and $i_+ \in \mathbf{ch}(i)$, every edge of the tree — that connects the nodes i and i_+ — is associated with a stage cost function $\ell(x^i, u^i, w^{i_+}) = x^{i\intercal} Q^{i_+} x^i + u^{i\intercal} R^{i_+} u^i$. For $j \in \mathbf{nodes}(N)$, every leaf node is associated with a terminal cost $\ell_N(x^j, w^j) = x^{j\intercal} Q_N^j x^j$.

4.2 Risk measures

Given a discrete sample space $\Omega = \{1, \dots, n\}$ and a probability vector $\pi \in \mathbb{R}^n$, a random cost $Z : \Omega \to \mathbb{R}$ can be identified by a vector $Z = (Z^1, \dots, Z^n) \in \mathbb{R}^n$. The expectation of Z with respect to π is $\mathbb{E}^\pi[Z] = \pi^\intercal Z$, and the maximum of Z is defined as $\mathbf{max}\, Z = \mathbf{max}\{Z^i, i \in \mathbb{N}_{[1,n]}\}$.

A risk measure can be used to reflect the inexact knowledge of the probability measure. A risk measure is an operator, $\rho : \mathbb{R}^n \to \mathbb{R}$, that maps a random cost $Z \in \mathbb{R}^n$ to a characteristic index $\rho(Z)$ that quantifies the magnitude of its right tail. A risk measure is said to be *coherent*

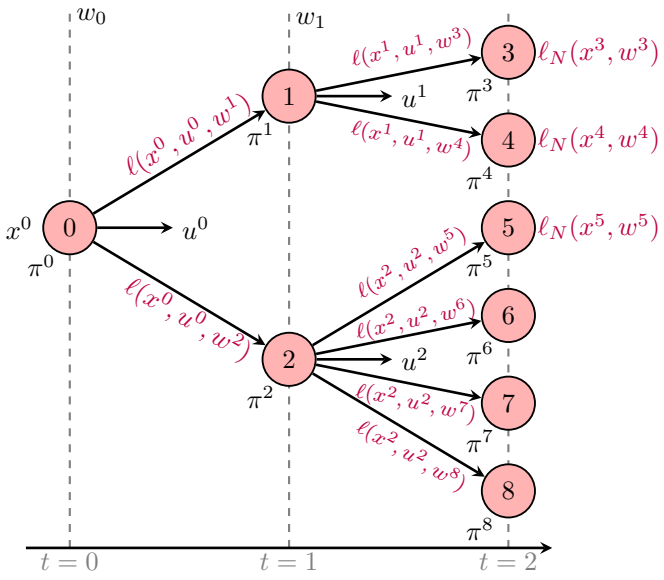

Fig. 1. Discrete RAOCP described by a scenario tree.

(Shapiro et al., 2021, Sec. 6.3) if (i) it is convex, (ii) it is monotone, that is, $\rho(Z) \le \rho(Z')$ whenever $Z \le Z'$ element-wise, (iii) ρ is translation equivariant, that is, $\rho(Z + c1_n) = \rho(Z) + c$ for all $Z \in \mathbb{R}^n$ and $c \in \mathbb{R}$, (iv) ρ is positively homogeneous. Coherent risk measures admit the following dual representation

$$\rho(Z) = \max_{\mu \in \mathcal{A}} \mathbb{E}^\mu[Z], \tag{10}$$

where the set $\mathcal{A}$ is a closed convex set known as the *ambiguity set* of ρ. An ambiguity set is a quantifiable way of indicating the uncertainty about the true probability distribution of π. Equation (10) suggests that coherent risk measures can be seen as the worst-case expectations of Z, with respect to a probability vector μ that is taken from $\mathcal{A}$. Clearly, $\mathbb{E}^\pi$ is a coherent risk measure with $\mathcal{A} = \{\pi\}$ and $\mathbf{max}$ is also coherent with $\mathcal{A}$ being the probability simplex.

A popular coherent risk measure is the average value-at-risk with parameter $a \in [0, 1]$, denoted by $\mathrm{AV@R}_a$. It is defined as

$$\mathrm{AV@R}_a[Z] = \begin{cases} \min\limits_{t \in \mathbb{R}^n} \left\{ t + \frac{1}{a} \mathbb{E}^\pi [Z - t]_+ \right\}, & a \neq 0, \\ \mathbf{max}[Z], & a = 0. \end{cases} \tag{11}$$

The ambiguity set of $\mathrm{AV@R}_a$ is

$$\mathcal{A}_a^{\mathrm{avar}}(\pi) = \left\{ \mu \in \mathbb{R}^n \,\middle|\, \sum_{i=1}^n \mu^i = 1, 0 \le \mu^i \le \tfrac{\pi^i}{a} \right\}. \tag{12}$$

4.3 Conic representation of risk measures

Coherent risk measures also admit the following *conic representation*

$$\rho[Z] = \max_{\mu \in \mathbb{R}^n, \nu \in \mathbb{R}^{n_\nu}} \{ \mu^\intercal Z \mid b - E\mu - F\nu \in \mathcal{K} \}, \tag{13}$$

where $\mathcal{K}$ is a closed convex cone. Provided strong duality holds (i.e., if there exist μ and ν so $b - E\mu - F\nu \in \mathbf{ri}\mathcal{K}$), the dual form of the risk measure in Equation (13) is

$$\rho[Z] = \min_{y} \{ y^\intercal b \mid E^\intercal y = Z, F^\intercal y = 0, y \in \mathcal{K}^* \}. \tag{14}$$

For AV@R$_a$, $n_\nu = 0$, $b = [\pi^\intercal \ 0_n^\intercal \ 1]^\intercal$, $E = [aI_n \ -I_n \ 1_n]^\intercal$, and $\mathcal{K} = \mathbb{R}^{2n}_+ \times \{0\}$.

4.4 Conditional risk mappings

Every node $i \in \mathbf{nodes}(t+1), t \in \mathbb{N}_{[0,N-1]}$ is associated with a cost value $Z^i = \ell(x^{\mathbf{anc}(i)}, u^{\mathbf{anc}(i)}, w^i)$. For each $t \in \mathbb{N}_{[0,N-1]}$ we define a random variable $Z_t := (Z^i)_{i \in \mathbf{nodes}(t+1)}$ on the probability space $\mathbf{nodes}(t+1)$. Every node $j \in \mathbf{nodes}(N)$ is associated with a cost value $Z^j = \ell(x^j, w^j)$. We define the random variable $Z_N := (Z^j)_{j \in \mathbf{nodes}(N)}$ on the probability space $\mathbf{nodes}(N)$.

We define $Z^{[i]} := (Z^{i_+})_{i_+ \in \mathbf{ch}(i)}, i \in \mathbf{nodes}(0, N-1)$. This partitions the variable $Z_t = (Z^{[i]})_{i \in \mathbf{nodes}(t)}$ into groups of nodes which share a common ancestor.

Let $\rho : \mathbb{R}^{|\mathbf{ch}(i)|} \to \mathbb{R}$ be risk measures on the probability space $\mathbf{ch}(i)$. For every stage $t \in \mathbb{N}_{[0,N-1]}$ we may define a conditional risk mapping at stage t, $\rho_{|t} : \mathbb{R}^{|\mathbf{nodes}(t+1)|} \to \mathbb{R}^{|\mathbf{nodes}(t)|}$, as follows (Shapiro et al., 2021, Sec. 6.8.2)

$$\rho_{|t}[Z_t] := \left(\rho^i[Z^{[i]}]\right)_{i \in \mathbf{nodes}(t)}. \tag{15}$$

5. PROPOSED SPLITTING

5.1 Original problem

A risk-averse optimal control problem with horizon N is defined via the following multistage nested formulation (Shapiro et al., 2021, Sec. 6.8.1)

$$V^\star := \inf_{u_0} \rho_{|0}\Big[Z_0 + \inf_{u_1} \rho_{|1}\Big[Z_1 + \cdots + \inf_{u_{N-1}} \rho_{|N-1}\big[Z_{N-1} + Z_N\big]\cdots\Big]\Big], \tag{16}$$

subject to (8) and (9). Note that the infima in (16) are taken element-wise and with respect to the control laws $u_t := (u^i)_{i \in \mathbf{nodes}(t)}$ for $t \in \mathbb{N}_{[0,N-1]}$. This problem is decomposed using *risk-infimum interchangeability* and *epigraphical relaxation* in (Sopasakis et al., 2019c), allowing us to cast the original problem as the following minimization problem with conic constraints

$$\underset{\substack{(x^i)_i,(x^j)_j,(u^i)_i,(y^i)_i,\\ (\tau^{i_+})_{i_+},(s^i)_i,(s^j)_j}}{\mathbf{minimize}} \quad s^0 \tag{17a}$$

$$\mathbf{subject\ to}\ x^0 = x, x^{i_+} = A^{i_+}x^i + B^{i_+}u^i, \tag{17b}$$

$$y^i \succcurlyeq_{(\mathcal{K}^i)^*} 0, y^{i\intercal}b^i \le s^i, \tag{17c}$$

$$E^{i\intercal}y^i = \tau^{[i]} + s^{[i]}, F^{i\intercal}y^i = 0, \tag{17d}$$

$$\ell(x^i, u^i, w^{i_+}) \le \tau^{i_+}, \ell_N(x^j, w^j) \le s^j, \tag{17e}$$

$$\Gamma^i_x x^i + \Gamma^i_u u^i \in C^i, \Gamma^j_N x^j \in C^j_N, \tag{17f}$$

for $t \in \mathbb{N}_{[0,N-1]}, i \in \mathbf{nodes}(t), i_+ \in \mathbf{ch}(i)$, and $j \in \mathbf{nodes}(N)$, where $(\cdot)^{[i]} = ((\cdot)^{i_+})_{i_+ \in \mathbf{ch}(i)}$, τ and s are slack variables, and x is the initial state.

5.2 Operator splitting

Define $z = (s^0, z_1, z_2)$, where $z_1 = ((x^{i'})_{i'}, (u^i)_i)$, and $z_2 = (y^i, \tau^{[i]}, s^{[i]})$, for $i' \in \mathbf{nodes}(0, N)$ and $i \in \mathbf{nodes}(0, N-1)$. We define the sets

$$\mathcal{S}_1 := \left\{ z_1 \;\middle|\; \begin{array}{l} x^0 - x = 0, \\ x^{i_+} - A^{i_+}x^i - B^{i_+}u^i = 0, \\ \text{for } i \in \mathbf{nodes}(0, N-1), i_+ \in \mathbf{ch}(i). \end{array} \right\}, \tag{18}$$

$$\mathcal{S}_2 := \prod_{i \in \mathbf{nodes}\,(0,N-1)} \underbrace{\ker \begin{bmatrix} E^{i\intercal} & -I & -I \\ F^{i\intercal} & 0 & 0 \end{bmatrix}}_{\mathcal{S}_2^i}, \tag{19}$$

$$\begin{aligned} \mathcal{S}_3 := &\prod_{i \in \mathbf{nodes}\,(0,N-1)} \left((\mathcal{K}^i)^* \times \mathbb{R}_+ \times C^i\right) \\ &\times \prod_{i \in \mathbf{nodes}\,(1,N)} \left(\mathrm{SOC}_{n_x+n_u+2} + \left[0^\intercal_{n_x+n_u} \ \tfrac{1}{2} \ -\tfrac{1}{2}\right]^\intercal\right) \\ &\times \prod_{j \in \mathbf{nodes}\,(N)} \left(C^j \times \left(\mathrm{SOC}_{n_x+2} + \left[0^\intercal_{n_x} \ \tfrac{1}{2} \ -\tfrac{1}{2}\right]^\intercal\right)\right). \end{aligned} \tag{20}$$

Next, we define a linear operator L that maps z to

$$\begin{aligned} \eta := &((y^i, s^i - b^{i\intercal}y^i, \Gamma^i_x x^i + \Gamma^i_u u^i)_{i \in \mathbf{nodes}(0,N-1)}, \\ &((Q^i)^{1/2}x^{\mathbf{anc}(i)}, (R^i)^{1/2}u^{\mathbf{anc}(i)}, \tfrac{1}{2}\tau^i, \tfrac{1}{2}\tau^i)_{i \in \mathbf{nodes}(1,N)}, \\ &(\Gamma^j_N x^j, ((Q^j_N)^{1/2}x^j, \tfrac{1}{2}s^j, \tfrac{1}{2}s^j))_{j \in \mathbf{nodes}(N)}). \end{aligned} \tag{21}$$

Remark that L can be made blockdiagonal by permuting its columns, such that each block corresponds to the constraints imposed on a single node. The permutation operation does not change the matrix norm, and the norm of a blockdiagonal matrix equals the largest norm of its constituting blocks. Therefore, $\|L\|$ does not scale with the prediction horizon N and, importantly, neither does the CP step size α.

We can now write Problem (17) as

$$\underset{z}{\mathbf{minimize}} \ s^0 \tag{22a}$$

$$\mathbf{subject\ to}\ z_1 \in \mathcal{S}_1, z_2 \in \mathcal{S}_2, \eta \in \mathcal{S}_3, \tag{22b}$$

which is equivalent to

$$\underset{z}{\mathbf{minimize}} \ f(z) + g(Lz), \tag{23}$$

where

$$f(z) := s^0 + \delta_{\mathcal{S}_1}(z_1) + \delta_{\mathcal{S}_2}(z_2), \tag{24a}$$

$$g(\eta) := \delta_{\mathcal{S}_3}(\eta). \tag{24b}$$

5.3 Proximal operator of f

From Equation (24a), the separable sum property (Parikh and Boyd, 2014, Sec. 2.1), and the fact that $\mathbf{prox}_{\alpha\delta_{\mathcal{S}}}(\cdot) = \mathbf{proj}_{\mathcal{S}}(\cdot)$, we can compute the proximal operator of f by independently computing $\mathbf{prox}_{\alpha \mathrm{id}}(s^0)$ (Parikh and Boyd, 2014, Sec. 6.1.1), $\mathbf{proj}_{\mathcal{S}_1}(z_1)$, and $\mathbf{proj}_{\mathcal{S}_2}(z_2)$.

Projection on $\mathcal{S}_1$. The projection of $\bar{z}_1$ on $\mathcal{S}_1$ is the minimizer of the problem

$$\underset{z_1 \in \mathcal{S}_1}{\mathbf{minimize}} \Bigg\{ \underbrace{\sum_{i \in \mathbf{nodes}(0,N-1)} \tfrac{1}{2}\left(\|x^i - \bar{x}^i\|^2 + \|u^i - \bar{u}^i\|^2\right)}_{\text{stage-wise sq. distances}} + \underbrace{\sum_{j \in \mathbf{nodes}(N)} \tfrac{1}{2}\|x^j - \bar{x}^j\|^2}_{\text{terminal stage sq. distances}} \Bigg\}. \tag{25}$$

This follows the structure of a finite horizon linear-quadratic optimal control (FHOC) problem, which can

be solved using the dynamic programming (DP) method. The method for computing $\mathbf{proj}_{\mathcal{S}_1}(\bar{z}_1)$ comprises the offline Algorithm 1, and the online Algorithm 2. The offline algorithm is run once at the start and the online algorithm is then run every time $\mathbf{proj}_{\mathcal{S}_1}(\bar{z}_1)$ is computed.

Algorithm 1 Projection on $\mathcal{S}_1$: Offline

Require: the system matrices $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, and the prediction horizon N
Ensure: Matrices $(K^i)_i$, $(P^i)_i$, $(\widetilde{R}^i)_i$ and $(\bar{A}^i)_i$
  for all $i \in \mathbf{nodes}(N)$ **in parallel do**
    $P^i \leftarrow I_{n_x}$
  for $t = 0, 1, \ldots, N-1$ **do**
    for all $i \in \mathbf{nodes}(N-(t+1))$ **in parallel do**
      $\widetilde{P}^{i_+} \leftarrow B^{i_+\intercal} P^{i_+}$, $i_+ \in \mathbf{ch}(i)$
      $\widetilde{R}^i \leftarrow I_{n_u} + \sum_{i_+ \in \mathbf{ch}(i)} \widetilde{P}^{i_+} B^{i_+}$ and determine its Cholesky decomposition
      $K^i \leftarrow -(\widetilde{R}^i)^{-1} \sum_{i_+ \in \mathbf{ch}(i)} \widetilde{P}^{i_+} A^{i_+}$
      $\bar{A}^{i_+} \leftarrow A^{i_+} + B^{i_+} K^i$, $i_+ \in \mathbf{ch}(i)$
      $P^i \leftarrow I_{n_x} + K^{i\intercal} K^i + \sum_{i_+ \in \mathbf{ch}(i)} \bar{A}^{i_+\intercal} P^{i_+} \bar{A}^{i_+}$

Algorithm 2 Projection on $\mathcal{S}_1$: Online

Require: Matrices computed by Algorithm 1, the vector $\bar{z}_1$ as defined in Sec. 5.2, and the initial state x^0
Ensure: Projection on $\mathcal{S}_1$ at $\bar{z}_1$
  for all $i \in \mathbf{nodes}(N)$ **in parallel do**
    $q_0^i \leftarrow -\bar{x}^i$
  for $t = 0, 1, \ldots, N-1$ **do**
    for all $i \in \mathbf{nodes}(N-(t+1))$ **in parallel do**
      $d_{t+1}^i \leftarrow (\widetilde{R}^i)^{-1} \left(\bar{u}^i - \sum_{i_+ \in \mathbf{ch}(i)} B^{i_+\intercal} q_t^{i_+} \right)$ using the Cholesky decomposition of $\widetilde{R}^i$
      $q_{t+1}^i \leftarrow K^{i\intercal} \left(d^i - \bar{u}^i \right) - \bar{x}^i + \sum_{i_+ \in \mathbf{ch}(i)} \bar{A}^{i_+\intercal} \left(P^{i_+} B^{i_+} d^i + q^{i_+} \right)$
  $x^0 \leftarrow x$
  for $t = 0, 1, \ldots, N-1$ **do**
    for all $i \in \mathbf{nodes}(t), i_+ \in \mathbf{ch}(i)$ **in parallel do**
      $u^i \leftarrow K^i x^i + d^i$
      $x^{i_+} \leftarrow A^{i_+} x^i + B^{i_+} u^i$
  return $z_1 = \mathbf{proj}_{\mathcal{S}_1}(\bar{z}_1)$.

Projection on $\mathcal{S}_2$. Since $\mathcal{S}_2 = \prod_{i \in \mathbf{nodes}(0,N-1)} \mathcal{S}_2^i$, the projection on $\mathcal{S}_2$ is the projection on the kernel of a matrix of typically small dimensions, i.e., the projection on a linear space. This can either be solved very efficiently and accurately using a numerical method, or, by precomputing a pseudoinverse. Note that the projections for each scenario tree node can be computed in parallel.

5.4 Proximal operator of g^*

We start with the proximal operator of g with parameter α, where $\mathbf{prox}_{\alpha g}(\eta) = \mathbf{proj}_{\mathcal{S}_3}(\eta)$. Then, by the extended Moreau decomposition (Beck, 2017, Thm.6.45)

$$\mathbf{prox}_{\alpha g^*}(\eta) = \eta - \mathbf{proj}_{\mathcal{S}_3}\left(\tfrac{\eta}{\alpha}\right). \tag{26}$$

The set $\mathcal{S}_3$ is the Cartesian product of low-dimensional (translated) convex cones (cf. (20)), projections on which can be carried out independently.

6. SPOCK: SUPERMANN AND CHAMBOLLE-POCK

To accelerate the CP method we combine it with SuperMann (Themelis and Patrinos, 2019) and call the resulting ensemble SPOCK (see Algorithm 3). SuperMann is an algorithmic framework that augments the classical Krasnosel'skiĭ-Mann iteration of an FNE operator with a line search to yield a method that enjoys the same global convergence properties, and under certain weak assumptions converges superlinearly. The FNE operator for SPOCK is T defined by Equation (5). We combine SuperMann with Anderson's acceleration (AA) method (Anderson, 1965) to compute fast quasi-Newtonian directions. The most computationally demanding parts of the algorithm are the operators L and L^*, which are computed in T.

Let us first define $c_z^{(k)} := z^{(k)} - z^{(k+1)}$, and likewise we define $c_\eta^{(k)}$. We denote a primal update direction by d_z, and a dual update direction by d_η. Now we can define $v^{(k)} := (z^{(k)}, \eta^{(k)})$, $c^{(k)} := (c_z^{(k)}, c_\eta^{(k)})$, and $d^{(k)} := (d_z^{(k)}, d_\eta^{(k)})$.

6.1 SuperMann

SuperMann is a Newton-type algorithm that finds a fixed-point $v^\star \in \mathbf{fix}\, T$ by finding a zero of R. The framework involves two extragradient-type updates of the general form

$$v_{K1}^{(k+1)} = v^{(k)} + \tau_{(k)} d^{(k)}, \tag{27a}$$

$$v_{K2}^{(k+1)} = v^{(k)} - \bar{\tau}_{(k)} R(v_{K1}^{(k+1)}), \tag{27b}$$

where $d^{(k)}$ are fast, e.g., quasi-Newtonian, directions and the step sizes $\tau_{(k)}$ and $\bar{\tau}_{(k)}$ are chosen so that SuperMann enjoys the same global convergence properties as the classical Krasnosel'skiĭ-Mann scheme. Note that when the directions satisfy a Dennis-Moré condition, SuperMann converges superlinearly under mild assumptions that include metric subregularity of the residual.

At each step we perform a backtracking line search on $\tau_{(k)}$ until we either trigger an "educated" *K1* update, (fast convergence, cf. (27a)) or a "safeguard" *K2* update, (global convergence, cf. (27b)) as shown in Algorithm 3. The *K2* update can be interpreted as a projection of $v^{(k)}$ on a hyperplane generated by $v_{K1}^{(k+1)}$ (which separates $\mathbf{fix}\, T$ from $v^{(k)}$), therefore guaranteeing that every iterate $v^{(k+1)}$ moves closer to $\mathbf{fix}\, T$. SuperMann also checks for a sufficient decrease of the norm of the residual $\|R(v^{(k)})\|$ at each step, which triggers a "blind" *K0* update of the form $v_{K0}^{(k+1)} = v^{(k)} + d^{(k)}$. This does not require line search iterations.

6.2 Anderson's acceleration

Quasi-Newtonian directions can be computed according to the general rule

$$d^{(k)} = -B_{(k)}^{-1} c^{(k)} = -H_{(k)} c^{(k)}, \tag{28}$$

where the invertible linear operators $H_{(k)}$ are updated according to certain low-rank updates so as to satisfy certain secant conditions, starting from an initial operator $H_{(0)}$. AA enforces a multi-secant condition (Fang and Saad, 2009; Walker and Ni, 2011). For each iteration k,

we update two buffers of length m: one for the differences of z and η between iterations,

$$M_{(k)}^{\mathrm{P}} = \left[v^{(k)} - v^{(k-1)} \cdots v^{(k-m+2)} - v^{(k-m+1)}\right], \quad (29)$$

and one for the differences of c_z and c_η between iterations,

$$M_{(k)}^{\mathrm{R}} = \left[c^{(k)} - c^{(k-1)} \cdots c^{(k-m+2)} - c^{(k-m+1)}\right]. \quad (30)$$

Directions are computed by

$$d^{(k)} = -c^{(k)} - (M_{(k)}^{\mathrm{P}} - M_{(k)}^{\mathrm{R}})\gamma^{(k)}, \quad (31)$$

where $\gamma^{(k)}$ is the least squares solution to the linear system $M_{(k)}^R \gamma^{(k)} = c^{(k)}$, that is, $\gamma^{(k)}$ solves

$$\underset{\gamma^{(k)}}{\textbf{minimize}} \|M_{(k)}^R \gamma^{(k)} - c^{(k)}\|^2, \quad (32)$$

which can be solved by QR factorization of the matrix $M_{(k)}^R$ (which may be updated at every iteration (Walker and Ni, 2011)). In practice AA performs well with short memory lengths between 3 and 10.

Algorithm 3 SPOCK algorithm for RAOCPs

Require: problem data, $z^{(0)}$ and $\eta^{(0)}$, tolerances $\epsilon_{\text{abs}} > 0$ and $\epsilon_{\text{rel}} > 0$, α such that $0 < \alpha\|L\| < 1$, $\mathbb{N} \ni m > 0$, $c_0, c_1, c_2 \in [0, 1)$, $\beta, \sigma \in (0, 1)$, and $\lambda \in (0, 2)$.
Ensure: approximate solution of RAOCP

1: Invoke Algorithm 1
2: $r^{(0)} \leftarrow v^{(0)} - T(v^{(0)})$
3: $\zeta_{(0)} \leftarrow ((r_z^{(0)})^\intercal (r_z^{(0)} - \alpha L^* r_\eta^{(0)}) + (r_\eta^{(0)})^\intercal (r_\eta^{(0)} - \alpha L r_z^{(0)}))^{1/2}$
4: $\omega_{\text{safe}} \leftarrow \zeta_{(0)}$, $k \leftarrow 0$
5: **if** termination criteria (Sec. 3) are satisfied **then**
6: **return** $v^{(k)}$
7: Use AA to compute an update direction, $d^{(k)}$ (Sec. 6.2)
8: $r^{(k)} \leftarrow v^{(k)} - T(v^{(k)})$
9: $\omega_{(k)} \leftarrow ((r_z^{(k)})^\intercal (r_z^{(k)} - \alpha L^* r_\eta^{(k)}) + (r_\eta^{(k)})^\intercal (r_\eta^{(k)} - \alpha L r_z^{(k)}))^{1/2}$
10: **if** $\omega_{(k)} \le c_0 \zeta_{(k)}$ **then**
11: $v^{(k+1)} \leftarrow v^{(k)} + d^{(k)}$, $\zeta_{(k+1)} \leftarrow \omega_{(k)}$, **goto** line 20 *(K0)*
12: $\zeta_{(k+1)} \leftarrow \zeta_{(k)}$, $\tau \leftarrow 1$
13: $\tilde{v}^{(k)} \leftarrow v^{(k)} + \tau d^{(k)}$, $\tilde{r}^{(k)} \leftarrow \tilde{v}^{(k)} - T(\tilde{v}^{(k)})$
14: $\tilde{\omega}_{(k)} \leftarrow ((\tilde{r}_z^{(k)})^\intercal (\tilde{r}_z^{(k)} - \alpha L^* \tilde{r}_\eta^{(k)}) + (\tilde{r}_\eta^{(k)})^\intercal (\tilde{r}_\eta^{(k)} - \alpha L \tilde{r}_z^{(k)}))^{1/2}$
15: **if** $\omega_{(k)} \le \omega_{\text{safe}}$ **and** $\tilde{\omega}_{(k)} \le c_1 \omega_{(k)}$ **then**
16: $v^{(k+1)} \leftarrow \tilde{v}^{(k)}$, $\omega_{\text{safe}} \leftarrow \tilde{\omega}_{(k)} + c_2^k$, **goto** line 20 *(K1)*
17: $\rho_{(k)} \leftarrow \tilde{\omega}_{(k)}^2 - 2\alpha (\tilde{r}^{(k)})^\intercal M (\tilde{v}^{(k)} - v^{(k)})$
18: **if** $\rho_{(k)} \ge \sigma \tilde{\omega}_{(k)} \omega_{(k)}$ **then** $v^{(k+1)} \leftarrow v^{(k)} - \dfrac{\lambda \rho_{(k)}}{\tilde{\omega}_{(k)}^2} \tilde{r}^{(k)}$ *(K2)*
19: **else** $\tau \leftarrow \beta\tau$, **goto** line 13
20: $k \leftarrow k + 1$, **goto** line 5

7. SIMULATION RESULTS

Consider a linear system $x_{t+1} = A(w_t)x_t + B(w_t)u_t$ where $A(w)$ is the tridiagonal matrix defined by

$$A_{j,j}(w) = 1 + \frac{w-1}{W}\left(1 + \frac{j-1}{n_x}\right), \quad j \in \mathbb{N}_{[1,n_x]}, \quad (33a)$$

$$A_{j,j-1}(w) = A_{j-1,j}(w) = 0.01, \quad j \in \mathbb{N}_{[2,n_x]}, \quad (33b)$$

and $B(w) = I_{n_u}$ for $w \in \mathbb{N}_{[1,W]}$. This system is inspired by (Schuurmans and Patrinos, 2021; Dean et al., 2020) and models the temperature of n_x servers in a data center. The state $x_t \in \mathbb{R}^{n_x}$ corresponds to the deviations from some nominal temperature, and the input $u_t \in \mathbb{R}^{n_u}$ models the amount of heating ($u_t \ge 0$) or cooling ($u_t < 0$) that is applied to each server. The diagonal elements (33a), and the sub- and super-diagonal elements (33b) of $A(w)$ model the heat generated by each individual machine, and the heat transfer between adjacent machines, respectively. The disturbance w represents the load on the system; if $w = 1$, the servers are idle and no additional heat is generated. If $w = W$, all servers are running at full capacity and a maximum amount of heat is generated. Note that server j produces more heat than server $j-1$ under the same load.

We use the quadratic stage and terminal costs from Sec. 4.1 where $Q^{i_+} = Q_N^j = I_{n_x}$ and $R^{i_+} = 10 I_{n_u}$ for $i \in \mathbf{nodes}(0, N-1), i_+ \in \mathbf{ch}(i)$ and $j \in \mathbf{nodes}(N)$. We also impose the constraints $\|x^i\|_\infty \le 1$ for $i \in \mathbf{nodes}(1, N)$ and $\|u^i\|_\infty \le 1.5$ for $i \in \mathbf{nodes}(0, N-1)$. Each node has an associated AV@R$_a$ with $a = 0.95$. This section compares SPOCK [1] to existing solvers when solving the RAOCP (16) of horizon N for the above example. Remark that the number of scenarios in this problem, W^N, grows exponentially with the prediction horizon, N.

In what follows, we fix $W = 2$ and consider reference probabilities $\pi^i = (0.3, 0.7)^\intercal$, for $i \in \mathbf{nodes}(0, N-1)$. The results for SPOCK are obtained using the SuperMann parameters from (Themelis and Patrinos, 2019, Sec.VI.D). Quasi-Newton directions are generated by AA with $m = 3$. The other solvers all solve the tractable reformulation (17) through the JUMP interface (Dunning et al., 2017).

7.1 SuperMann acceleration

To illustrate the benefit of applying SuperMann, we compare the performance of the vanilla CP algorithm with the SPOCK solver. Consider system (33) with $N = 7$, $n_x = 5$, and $x^0 = 0.1 \cdot 1_{n_x}$.

Fig. 2 shows the residual value ξ from Sec. 3 against the number of calls to the operator L for both CP and SPOCK, with tolerances $\epsilon_{\text{abs}} = \epsilon_{\text{rel}} = 10^{-5}$. It demonstrates that SuperMann requires considerably fewer calls compared to vanilla CP, as the former converges with 571 calls, and the latter with 3159. Furthermore, Fig. 2 suggests that this difference becomes more pronounced as the tolerances decrease.

7.2 Open-loop simulation

We now evaluate SPOCK's performance by comparing it to both open-source solvers (IPOPT (Wächter and Biegler, 2006), SEDUMI (Sturm, 1999), COSMO (Garstka et al., 2021)) and proprietary alternatives (MOSEK (Andersen and Andersen, 2000), GUROBI (Gurobi Optimization, LLC, 2018)). COSMO and SEDUMI are conic solvers, while IPOPT, MOSEK and GUROBI are interior-point solvers.

Consider system (33) with $n_x = 50$, $x^0 = 0.1 \cdot 1_{n_x}$, and $\epsilon_{\text{abs}} = \epsilon_{\text{rel}} = 10^{-3}$. Fig. 3 shows the execution time of the aforementioned solvers against the prediction horizon N. The number of iterations of SPOCK in this simulation is shown in Table 1. Once a solver reaches the run time threshold of 150 s, the horizon N is no longer increased for that solver. The performances of

[1] A sequential implementation is available at https://github.com/kul-optec/spock.jl.

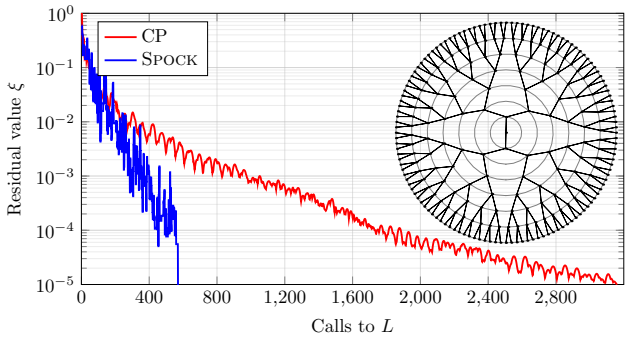

Fig. 2. Residual value ξ against the number of calls of the linear operator for the vanilla Chambolle-Pock algorithm (CP) and SPOCK, for $N = 7, n_x = 5$. A visualization of the scenario tree is shown, in which the root node is at the center and each concentric circle corresponds to a time stage.

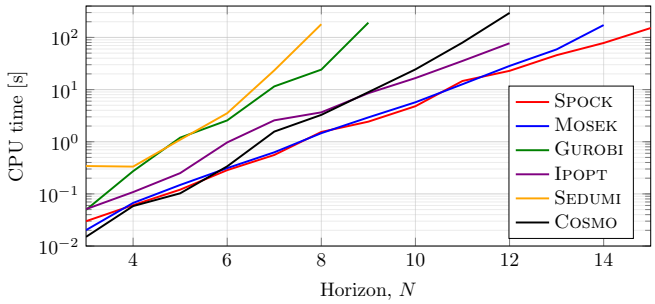

Fig. 3. Execution time against the prediction horizon N, for $n_x = n_u = 50$. The number of scenarios is 2^N.

SEDUMI and GUROBI appear to rapidly degrade as the prediction horizon increases. IPOPT and COSMO perform decently for moderate problem sizes. However, for large values of N, the execution time of COSMO increases at a considerably higher rate than that of SPOCK, and IPOPT even runs out of memory for $N = 12$. SPOCK performs competitively with MOSEK for smaller problem sizes, and even outperforms it by approx. a factor 2 for $N = 14$. Further experiments have indicated that eventually MOSEK runs out of memory when $N = 16$, whereas SPOCK does not. Altogether, these results are highly promising, considering that the computationally demanding steps of the SPOCK solver can be massively parallelized.

Table 1. SPOCK iterations corresponding to Fig. 3.

Horizon, N	3	6	9	12	15
Num. iterations	105	128	110	109	105

7.3 Closed-loop simulation

Most of the solvers we compared SPOCK to use interior-point methods and are, therefore, difficult to warm-start. However, within the context of MPC, a reasonable initial guess of the solution is typically available from the previous time step. We demonstrate that this can be leveraged to a far greater extent by the SPOCK solver than by the compared alternatives, by carrying out an MPC simulation

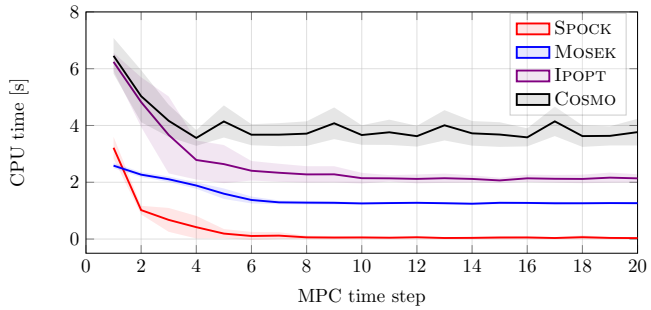

Fig. 4. Average execution time for successive MPC time steps over 15 random realizations with $N = 10$ and $n_x = n_u = 20$. SPOCK and COSMO exploit warm-starts, the interior-point solvers do not.

of 20 steps, in which we iteratively solve the RAOCP for successive initial states. Warm-starts are performed by using the primal and dual solutions at one MPC step to initialize the primal and dual vectors in the next MPC step. We consider system (33) with $N = 10$, $n_x = 20$, $\epsilon_{\text{abs}} = \epsilon_{\text{rel}} = 10^{-3}$ and the initial state in the first MPC step $x^0 = 0.1 \cdot 1_{n_x}$.

Fig. 4 compares the average execution time of the best performing solvers mentioned in Sec. 7.2, for successive MPC time steps. SPOCK and COSMO exploit warm-starts, the interior-point solvers do not. In the first MPC step, SPOCK performs better than IPOPT and COSMO, and slightly slower than MOSEK. However, warm-starts greatly reduce the required number of iterations for SPOCK, such that it outperforms all the other solvers in closed loop.

8. CONCLUSIONS AND FUTURE WORK

We proposed a splitting scheme for general risk-averse problems tailored to proximal methods by exploiting the problem structure. The splitting is used in our proposed algorithm SPOCK which is amenable to warm-starting and massive parallelization. SPOCK takes advantage of the SuperMann scheme combined with fast directions from Anderson's acceleration method to enhance the convergence speed. We implemented SPOCK in `Julia` as an open-source solver, which is available on GitHub at https://github.com/kul-optec/spock.jl. Our simulations demonstrate: the benefit of SPOCK over the vanilla CP method; that the solver, even without parallelization, outperforms proprietary solvers such as MOSEK and GUROBI for large problem sizes; and, that when warm-starting is used in the context of MPC, SPOCK outperforms the compared solvers by a considerable margin. In the future, we plan to harness the parallelization capabilities of GPUs by implementing SPOCK in CUDA-C++.

REFERENCES

Andersen, E.D. and Andersen, K.D. (2000). The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High performance optimization*, 197–232. Springer.

Anderson, D.G. (1965). Iterative procedures for nonlinear integral equations. *JACM*, 12(4), 547–560.

Barsi Haberfeld, G. (2021). *Efficient algorithms for risk-averse air-ground rendezvous missions.* Ph.D. thesis, University of Illinois at Urbana-Champaign.

Beck, A. (2017). *First-order methods in optimization*. SIAM.

Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40(1), 120–145.

Chow, Y., Ghavamzadeh, M., Janson, L., and Pavone, M. (2017). Risk-constrained reinforcement learning with percentile risk criteria. *JMLR*, 18(1), 6070–6120.

Chow, Y., Tamar, A., Mannor, S., and Pavone, M. (2015). Risk-sensitive and robust decision-making: a CVaR optimization approach. *Adv. neural inf. proc. sys.*, 28.

Chowdhury, R. and Subramani, D. (2022). Optimal path planning of autonomous marine vehicles in stochastic dynamic ocean flows using a GPU-accelerated algorithm. *IEEE Journal of Oceanic Engineering*.

Cinquemani, E., Agarwal, M., Chatterjee, D., and Lygeros, J. (2011). Convexity and convex approximations of discrete-time stochastic control problems with constraints. *Automatica*, 47(9), 2082–2087.

da Costa, B.F.P. and Leclère, V. (2021). Dual SDDP for risk-averse multistage stochastic programs. *arXiv preprint arXiv:2107.10930*.

Dean, S., Mania, H., Matni, N., Recht, B., and Tu, S. (2020). On the sample complexity of the linear quadratic regulator. *Found. Comp. Math.*, 20(4), 633–679.

Dixit, A., Ahmadi, M., and Burdick, J.W. (2022). Risk-averse receding horizon motion planning. *arXiv preprint arXiv:2204.09596*.

Dunning, I., Huchette, J., and Lubin, M. (2017). Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2), 295–320.

Dupačová, J., Consigli, G., and Wallace, S.W. (2000). Scenarios for multistage stochastic programs. *Annals of operations research*, 100(1), 25–53.

Fang, H.r. and Saad, Y. (2009). Two classes of multisecant methods for nonlinear acceleration. *Numerical linear algebra with applications*, 16(3), 197–221.

Garstka, M., Cannon, M., and Goulart, P. (2021). COSMO: A conic operator splitting method for convex conic problems. *JOTA*, 190(3), 779–810.

Gurobi Optimization, LLC (2018). Gurobi optimizer reference manual.

Hans, C.A., Sopasakis, P., Raisch, J., Reincke-Collon, C., and Patrinos, P. (2019). Risk-averse model predictive operation control of islanded microgrids. *IEEE TCST*, 28(6), 2136–2151.

Høyland, K. and Wallace, S.W. (2001). Generating scenario trees for multistage decision problems. *Management science*, 47(2), 295–307.

Maree, J.P., Gros, S., and Lakshmanan, V. (2021). Low-complexity risk-averse MPC for EMS. In *IEEE Smart Grid Comm*, 358–363. IEEE.

Nemirovski, A.S. and Todd, M.J. (2008). Interior-point methods for optimization. *Acta Numerica*, 17, 191–234.

O'Donoghue, B., Stathopoulos, G., and Boyd, S. (2013). A splitting method for optimal control. *IEEE Transactions on Control Systems Technology*, 21(6), 2432–2442.

Oh, J., Hwang, S.J., Nguyen, H.G., Kim, A., Kim, S.W., Kim, C., and Kim, J.K. (2008). Exploiting thread-level parallelism in lockstep execution by partially duplicating a single pipeline. *ETRI journal*, 30(4), 576–586.

Parikh, N. and Boyd, S. (2014). Proximal algorithms. *FnT in Optimization*, 1(3), 127–239.

Pereira, M.V. and Pinto, L.M. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1), 359–375.

Rawlings, J.B., Mayne, D.Q., and Diehl, M. (2017). *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI.

Ryu, E.K. and Boyd, S. (2016). Primer on monotone operator methods. *Appl. Comput. Math*, 15(1), 3–43.

Sampathirao, A.K., Sopasakis, P., Bemporad, A., and Patrinos, P. (2015). Distributed solution of stochastic optimal control problems on GPUs. In *CDC*, 7183–7188.

Schuurmans, M., Katriniok, A., Tseng, H.E., and Patrinos, P. (2020). Learning-based risk-averse model predictive control for adaptive cruise control with stochastic driver models. *IFAC-PapersOnLine*, 53(2), 15128–15133.

Schuurmans, M. and Patrinos, P. (2021). A general framework for learning-based distributionally robust MPC of Markov jump systems. *arXiv e-prints*, arXiv–2106.

Shapiro, A. (2011). Analysis of stochastic dual dynamic programming method. *EJOR*, 209(1), 63–72.

Shapiro, A., Dentcheva, D., and Ruszczynski, A. (2021). *Lectures on stochastic programming: modeling and theory*. SIAM.

Shapiro, A., Tekaya, W., da Costa, J.P., and Soares, M.P. (2013). Risk neutral and risk averse stochastic dual dynamic programming method. *EJOR*, 224(2), 375–391.

Sopasakis, P., Herceg, D., Bemporad, A., and Patrinos, P. (2019a). Risk-averse model predictive control. *Automatica*, 100, 281–288.

Sopasakis, P., Menounou, K., and Patrinos, P. (2019b). Superscs: fast and accurate large-scale conic optimization. In *ECC*, 1500–1505. IEEE.

Sopasakis, P., Schuurmans, M., and Patrinos, P. (2019c). Risk-averse risk-constrained optimal control. In *ECC*, 375–380. IEEE.

Stathopoulos, G., Shukla, H., Szucs, A., Pu, Y., Jones, C.N., et al. (2016). Operator splitting methods in control. *FnT in Systems and Control*, 3(3), 249–362.

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

Sturm, J.F. (1999). Using sedumi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1-4), 625–653.

Themelis, A. and Patrinos, P. (2019). SuperMann: a superlinearly convergent algorithm for finding fixed points of nonexpansive operators. *IEEE TACON*, 64(12), 4875–4890.

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.

Walker, H.F. and Ni, P. (2011). Anderson acceleration for fixed-point iterations. *SINUM*, 49(4), 1715–1735.

Wang, Y. and Chapman, M.P. (2022). Risk-averse autonomous systems: a brief history and recent developments from the perspective of optimal control. *Artificial Intelligence*, 103743.

Yildirim, E.A. and Wright, S.J. (2002). Warm-start strategies in interior-point methods for linear programming. *SIOPT*, 12(3), 782–810.