ORIGINAL PAPER

Optimization techniques for tree-structured nonlinear problems

Jens Hübner[1] · Martin Schmidt[2] · Marc C. Steinbach[3]

Abstract

Robust model predictive control approaches and other applications lead to nonlinear optimization problems defined on (scenario) trees. We present structure-preserving Quasi-Newton update formulas as well as structured inertia correction techniques that allow to solve these problems by interior-point methods with specialized KKT solvers for tree-structured optimization problems. The same type of KKT solvers could be used in active-set based SQP methods. The viability of our approach is demonstrated by two robust control problems.

Keywords Nonlinear stochastic optimization · Interior-point methods · Structured Quasi-Newton updates · Structured inertia correction · Robust model predictive control

Mathematics Subject Classification 90-08 · 90C06 · 90C15 · 90C30 · 90C51

1 Introduction

This paper addresses nonlinear optimization problems (NLPs) with an underlying tree topology. The prototypical example are multistage stochastic optimization problems. These are computationally expensive because they involve some random process

✉ Martin Schmidt
martin.schmidt@uni-trier.de

Jens Hübner
info@jhuebner.de

Marc C. Steinbach
mcs@ifam.uni-hannover.de

1   HaCon Ingenieurgesellschaft mbH, Lister Str. 15, 30163 Hannover, Germany

2   Department of Mathematics, Trier University, Universitätsring 15, 54296 Trier, Germany

3   Institute of Applied Mathematics, Leibniz Universität Hannover, Welfengarten 1, 30167 Hannover, Germany

whose discretization yields a scenario tree that grows exponentially with the length of the planning horizon. Primal and dual decomposition methods are specially tailored to linear and mildly nonlinear convex multistage stochastic optimization problems. Moreover, they extend well to the mixed-integer case. See Birge and Louveaux (2011), Kall and Wallace (1994) for an overview of algorithmic approaches and of stochastic optimization in general. To tackle highly nonlinear convex and nonconvex NLPs, we consider interior-point and SQP methods. Interior-point methods (IPMs) are also efficient on linear or quadratic problems, while SQP methods extend well to mixed-integer problems. In both cases the key to efficiency is the algebraic structure of the arising KKT systems rather than the stochastic structure. Therefore, we allow arbitrary trees instead of only scenario trees. All methods mentioned above are amenable to parallelization due to the rich structure of the underlying tree.

In Steinbach (2002) and the references therein, the third author has developed suitable formulations and a sequential interior-point approach for convex tree-structured NLPs with polyhedral constraints, although with parallelization and with the fully nonlinear and nonconvex case in mind. Other early interior-point approaches for stochastic optimization include Berger et al. (1995), Carpenter et al. (1993), Czyzyk et al. (1995), Jessup et al. (1994), Schweitzer (1998). For more recent structure-exploiting parallel algorithms see, e.g., Blomvall (2003), Blomvall and Lindberg (2002), Gondzio and Grothey (2006, 2007, 2009), Lubin et al. (2012). Based on the dissertation (Hübner 2016), a massively distributed implementation of our algorithm has been presented in Hübner et al. (2017). Here we address the extension to fully nonlinear and nonconvex problems. In particular, we present proper NLP formulations, structure-preserving Quasi-Newton updates, and tailored inertia correction techniques. Although all presented techniques possess a straightforward distributed implementation, we will not address this in detail.

The paper is structured as follows. In Sect. 2 we briefly introduce our approach and extend the problem formulations and KKT systems of Steinbach (2002) to the general NLP case. Section 3 presents (partially) separable Quasi-Newton update formulas that preserve the problem-specific block structure. The proper specialization of standard inertia correction techniques to this structure is discussed in Sect. 4. In Sect. 5, finally, two case studies from robust process control that lead to multistage stochastic NLPs with ODE dynamics serve as proof of concept for our algorithmic techniques.

2 Tree-sparse optimization

In this paper, we consider the integrated modeling and solution framework for tree-structured convex NLPs from Steinbach (2002) that we refer to as "tree-sparse". It consists of natural model formulations that have favorable regularity properties and block-level sparsity admitting $O(|V|)$ KKT solution algorithms, where $|V|$ is the number of tree nodes. There are three standard forms that cover, respectively, the general case with no further structure and two more specialized control formulations with different stochastic interpretations. The NLPs that we study below have linearizations whose polyhedral constraints match precisely the structure considered in Steinbach (2002).

For the following, let $V = \{0, \ldots, N\}$ denote the node set of a tree rooted in 0. Given a node $j \in V$, we denote the set of successors by $S(j)$, the predecessor by $i = \pi(j)$ (if $j \neq 0$), and the path to the root by $\Pi(j) = (j, \pi(j), \ldots, 0)$. The level of j is the path length, i.e., $t(j) = |\Pi(j)| - 1$. Finally, L_t stands for the set of level t nodes and L for the set of leaves. In a scenario tree we have $L = L_T$ where T is the tree's depth, and every node j has a probability p_j such that $\sum_{j \in L_t} p_j = 1$ for every t.

2.1 Tree-sparse NLPs

Consider a general NLP with equality constraints, range inequality constraints, and simple bounds in the form

$$\min_{y} \quad \phi(y) \quad \text{s.t.} \quad c_{\mathcal{E}}(y) = 0, \quad c_{\mathcal{R}}(y) \geq 0, \quad y \in [y^-, y^+]. \tag{1}$$

We call the NLP *tree-sparse* if it satisfies certain separability properties. Given a tree with vertex set V and variable vector $y = (y_j)_{j \in V}$, the objective ϕ and range constraints $c_{\mathcal{R}}$ have to be separable with respect to the node variables y_j, and the equality constraints $c_{\mathcal{E}}$ split into dynamic constraints with a Markov structure (y_j depends only on its predecessor y_i) and separable *global* constraints. We distinguish three variants of tree-sparse NLPs depending on the type of dynamic constraints: *implicit* tree-sparse NLPs have implicit dynamics while *outgoing* and *incoming* tree-sparse NLPs with $y_j = (x_j, u_j)$ and $y_j = (u_j, x_j)$, respectively, have explicit dynamics in control form where the current state x_j depends on (x_i, u_i) or (x_i, u_j). In the following, we state these three NLP variants without further explanation. All details can be found in Steinbach (2002).

The implicit tree-sparse NLP reads

$$\min_{y} \quad \sum_{j \in V} \phi_j(y_j) \tag{2a}$$

$$\text{s.t.} \quad h_j(y_i) - g_j(y_j) = 0, \quad j \in V, \tag{2b}$$

$$r_j(y_j) \geq 0, \quad\quad\quad j \in V, \tag{2c}$$

$$y_j \in [y_j^-, y_j^+], \quad\quad j \in V, \tag{2d}$$

$$\sum_{j \in V} e_j(y_j) = 0, \tag{2e}$$

the outgoing tree-sparse NLP is given by

$$\min_{x,u} \quad \sum_{j \in V} \phi_j(x_j, u_j) \tag{3a}$$

$$\text{s.t.} \quad h_j(x_i, u_i) - x_j = 0, \quad\quad\quad j \in V, \tag{3b}$$

$$r_j(x_j, u_j) \geq 0, \quad\quad\quad\quad j \in V, \tag{3c}$$

$$x_j \in [x_j^-, x_j^+], \ u_j \in [u_j^-, u_j^+], \quad j \in V, \tag{3d}$$

$$\sum_{j \in V} e_j(x_j, u_j) = 0, \tag{3e}$$

and the incoming tree-sparse NLP reads

$$\min_{u,x} \sum_{j \in V} \left(\phi_{ij}(x_i, u_j) + \phi_j(x_j) \right) \tag{4a}$$

$$\text{s.t.} \quad h_j(x_i, u_j) - x_j = 0, \qquad\qquad j \in V, \tag{4b}$$

$$r_{ij}(x_i, u_j) \geq 0, \ r_j(x_j) \geq 0, \qquad j \in V, \tag{4c}$$

$$u_j \in [u_j^-, u_j^+], \ x_j \in [x_j^-, x_j^+], \quad j \in V, \tag{4d}$$

$$\sum_{j \in V} \left(e_{ij}(x_i, u_j) + e_j(x_j) \right) = 0. \tag{4e}$$

At the root, $j = 0$, the preceding variable y_i is empty and the dynamic constraints reduce to respective initial conditions $g_0(y_0) = h_0$, $x_0 = h_0$, and $x_0 = h_0(u_0)$.

In addition to global equality constraints one might also consider *local* equality constraints as in Steinbach (2002). In the above NLPs these would take the respective forms

$$e_j^l(y_j) = 0, \quad j \in V, \qquad\qquad \text{(implicit)} \tag{5a}$$

$$e_j^l(x_j, u_j) = 0, \quad j \in V, \qquad\qquad \text{(outgoing)} \tag{5b}$$

$$e_{ij}^l(x_i, u_j) = 0, \quad e_j^l(x_j) = 0, \quad j \in V. \qquad \text{(incoming)} \tag{5c}$$

To avoid unnecessary technical complexity we assume that local constraints are modeled as global constraints. Specific discussions will be added where the distinction is relevant. Again for simplicity we consider range constraints in the form $c_{\mathcal{R}}(y) \geq 0$ rather than lower and upper constraints as in Steinbach (2002), $c_{\mathcal{R}}(y) \in [r^-, r^+]$.

2.2 Tree-sparse KKT systems

We solve the NLP (1) by an interior-point method, handle bounds directly, and convert range constraints to $c_{\mathcal{R}}(y) - s = 0$ using slacks $s \geq 0$. Then, every iteration leads to a KKT system of the following form, where slack increments Δs have already been eliminated:

$$\begin{bmatrix} H + \Phi & G^T & F^T & (F^r)^T \\ G & & & \\ F & & & \\ F^r & & & -\Psi^{-1} \end{bmatrix} \begin{pmatrix} \Delta y \\ -\Delta \lambda \\ -\Delta \mu \\ -\Delta v \end{pmatrix} = - \begin{pmatrix} f \\ h \\ e_V \\ \psi \end{pmatrix}. \tag{6}$$

Here G, F, F^r correspond to dynamic, global, and range constraints, respectively, and $\Phi \geq 0, \Psi > 0$ are diagonal barrier matrices. Elimination of Δv from the last equation, $F^r \Delta y + \Psi^{-1} \Delta v = -\psi$, then yields a system of the form

$$\begin{bmatrix} \bar{H} & G^T & F^T \\ G & & \\ F & & \end{bmatrix} \begin{pmatrix} \Delta y \\ -\Delta \lambda \\ -\Delta \mu \end{pmatrix} = - \begin{pmatrix} f \\ h \\ e_V \end{pmatrix}, \tag{7}$$

where $\bar{H} = H + \Phi + (F^r)^T \Psi (F^r)$. If we solve the NLP by an active-set based SQP method, every active-set sub-iteration also produces a KKT system of the form (7), except that the barrier matrices vanish (yielding $\bar{H} = H$) and that F may contain additional rows from active inequality constraints. It is an essential feature of all three tree-sparse NLP forms that the additional terms in $\bar{H}$ (IPM) or in F (SQP) preserve the original block structure.

Next we consider the separable or partially separable Lagrangians of the three tree-sparse NLP forms and the resulting specializations of the KKT system (6), which are all straightforward extensions of the material in Steinbach (2002). The specific forms of the Lagrangians are needed for the Quasi-Newton updates in Sect. 3 whereas the KKT systems are needed for the inertia correction in Sect. 4.

2.2.1 Implicit tree-sparse KKT system

The Lagrangian of the implicit tree-sparse NLP (with $\eta_j = (\eta_j^-, \eta_j^+)$ and h_0 constant) reads

$$\mathcal{L}(y, \lambda, v, \mu, \eta) = -\lambda_0^T h_0 + \sum_{j \in V} \mathcal{L}_j(y_j, \lambda_j, \lambda_{S(j)}, v_j, \mu, \eta_j) \tag{8}$$

where

$$\begin{aligned} \mathcal{L}_j(y_j, \lambda_j, \lambda_{S(j)}, v_j, \mu, \eta_j) = \phi_j(y_j) + \lambda_j^T g_j(y_j) - \sum_{k \in S(j)} \lambda_k^T h_k(y_j) \\ - v_j^T r_j(y_j) - \mu^T e_j(y_j) \\ - (\eta_j^-)^T (y_j - y_j^-) - (\eta_j^+)^T (y_j^+ - y_j). \end{aligned}$$

With barrier parameter β and $e = (1, \ldots, 1)^T$ in suitable dimension, the resulting barrier matrices and vectors are

$$\begin{aligned} \Phi_j &= \mathrm{Diag}(y_j^+ - y_j)^{-1} \mathrm{Diag}(\eta_j^+) + \mathrm{Diag}(y_j - y_j^-)^{-1} \mathrm{Diag}(\eta_j^-), \\ \varphi_j &= \eta_j^+ - \eta_j^- - \beta \left(\mathrm{Diag}(y_j^+ - y_j)^{-1} - \mathrm{Diag}(y_j - y_j^-)^{-1} \right) e, \\ \Psi_j &= \mathrm{Diag}(r_j)^{-1} \mathrm{Diag}(v_j), \\ \psi_j &= r_j - \beta \mathrm{Diag}(v_j)^{-1} e. \end{aligned}$$

Let ζ generically denote the dual variables appearing in the Lagrangian $\mathcal{L}$ or its components $\mathcal{L}_j$. Then the relevant partial derivatives of the Lagrangian read

$$\begin{aligned} H_j &= \nabla^2_{y_j y_j} \mathcal{L}_j(y_j, \zeta), & f_j &= \nabla_{y_j} \mathcal{L}_j(y_j, \zeta), \\ G_j &= \nabla_{y_i} h_j(y_i), & P_j &= \nabla_{y_j} g_j(y_j), \end{aligned}$$

$$F_j = \nabla_{y_j} e_j(y_j),$$
$$F_j^r = \nabla_{y_j} r_j(y_j),$$

and we further abbreviate

$$\hat{H}_j = H_j + \Phi_j, \qquad \hat{f}_j = f_j - \varphi_j, \qquad e_V = \sum_{j \in V} e_j(y_j).$$

This yields the following KKT system (6) in node-wise representation, where we omit Δ's on all increments and write y_j, λ_j, v_j, μ for simplicity:

$$\hat{H}_j y_j + P_j^T \lambda_j - \sum_{k \in S(j)} G_k^T \lambda_k - F_j^T \mu - (F_j^r)^T v_j + \hat{f}_j = 0, \quad j \in V, \qquad (9a)$$

$$G_j y_i - P_j y_j + (h_j - g_j) = 0, \quad j \in V, \qquad (9b)$$

$$F_j^r y_j + \Psi_j^{-1} v_j + \psi_j = 0, \quad j \in V, \qquad (9c)$$

$$\sum_{j \in V} F_j y_j + e_V = 0. \qquad (9d)$$

The construction of H, G, F, F^r and Φ, Ψ from the node contributions proceeds exactly as in Steinbach (2002), also for the following explicit tree-sparse NLP variants.

2.2.2 Outgoing tree-sparse KKT system

The Lagrangian of the outgoing tree-sparse NLP (with $\xi_j = (\xi_j^-, \xi_j^+)$ and h_0 again constant) reads

$$\mathcal{L}(x, u, \lambda, v, \mu, \eta, \xi) = -\lambda_0^T h_0 + \sum_{j \in V} \mathcal{L}_j(x_j, u_j, \lambda_j, \lambda_{S(j)}, v_j, \mu, \eta_j, \xi_j) \qquad (10)$$

with

$$\mathcal{L}_j(x_j, u_j, \lambda_j, \lambda_{S(j)}, v_j, \mu, \eta_j, \xi_j) = \phi_j(x_j, u_j) + \lambda_j^T x_j - \sum_{k \in S(j)} \lambda_k^T h_k(x_j, u_j)$$

$$- v_j^T r_j(x_j, u_j) - \mu^T e_j(x_j, u_j)$$
$$- (\eta_j^-)^T (x_j - x_j^-) - (\eta_j^+)^T (x_j^+ - x_j)$$
$$- (\xi_j^-)^T (u_j - u_j^-) - (\xi_j^+)^T (u_j^+ - u_j).$$

Here, the barrier matrices and vectors are

$$\Phi_j^x = \text{Diag}(x_j^+ - x_j)^{-1} \text{Diag}(\eta_j^+) + \text{Diag}(x_j - x_j^-)^{-1} \text{Diag}(\eta_j^-),$$
$$\Phi_j^u = \text{Diag}(u_j^+ - u_j)^{-1} \text{Diag}(\xi_j^+) + \text{Diag}(u_j - u_j^-)^{-1} \text{Diag}(\xi_j^-),$$

$$\varphi_j^x = \eta_j^+ - \eta_j^- - \beta \left(\mathrm{Diag}(x_j^+ - x_j)^{-1} - \mathrm{Diag}(x_j - x_j^-)^{-1} \right) e,$$

$$\varphi_j^u = \xi_j^+ - \xi_j^- - \beta \left(\mathrm{Diag}(u_j^+ - u_j)^{-1} - \mathrm{Diag}(u_j - u_j^-)^{-1} \right) e,$$

$$\Psi_j = \mathrm{Diag}(r_j)^{-1} \mathrm{Diag}(v_j),$$

$$\psi_j = r_j - \beta \mathrm{Diag}(v_j)^{-1} e,$$

and the partial derivatives of the Lagrangian read

$$H_j = \nabla_{x_j x_j}^2 \mathcal{L}_j(x_j, u_j, \varsigma), \tag{11a}$$

$$J_j = \nabla_{x_j u_j}^2 \mathcal{L}_j(x_j, u_j, \varsigma), \qquad K_j = \nabla_{u_j u_j}^2 \mathcal{L}_j(x_j, u_j, \varsigma), \tag{11b}$$

$$f_j = \nabla_{x_j} \mathcal{L}_j(x_j, u_j, \varsigma), \qquad d_j = \nabla_{u_j} \mathcal{L}_j(x_j, u_j, \varsigma), \tag{11c}$$

$$G_j = \nabla_{x_i} h_j(x_i, u_i), \qquad E_j = \nabla_{u_i} h_j(x_i, u_i), \tag{11d}$$

$$F_j^r = \nabla_{x_j} r_j(x_j, u_j), \qquad D_j^r = \nabla_{u_j} r_j(x_j, u_j), \tag{11e}$$

$$F_j = \nabla_{x_j} e_j(x_j, u_j), \qquad D_j = \nabla_{u_j} e_j(x_j, u_j). \tag{11f}$$

Thus, with

$$\hat{H}_j = H_j + \Phi_j^x, \qquad \hat{K}_j = K_j + \Phi_j^u,$$

$$\hat{f}_j = f_j - \varphi_j^x, \qquad \hat{d}_j = d_j - \varphi_j^u,$$

$$\bar{h}_j = h_j(x_i, u_i) - x_j, \qquad e_V = \sum_{j \in V} e_j(x_j, u_j)$$

we obtain the KKT system (6) in node-wise representation, again with Δ's omitted:

$$\hat{H}_j x_j + J_j^T u_j + \lambda_j - \sum_{k \in S(j)} G_k^T \lambda_k - F_j^T \mu - (F_j^r)^T v_j + \hat{f}_j = 0, \quad j \in V, \tag{12a}$$

$$J_j x_j + \hat{K}_j u_j - \sum_{k \in S(j)} E_k^T \lambda_k - D_j^T \mu - (D_j^r)^T v_j + \hat{d}_j = 0, \quad j \in V, \tag{12b}$$

$$G_j x_i + E_j u_i - x_j + \bar{h}_j = 0, \quad j \in V, \tag{12c}$$

$$F_j^r x_j + D_j^r u_j + \Psi_j^{-1} v_j + \psi_j = 0, \quad j \in V, \tag{12d}$$

$$\sum_{j \in V} F_j x_j + D_j u_j + e_V = 0. \tag{12e}$$

2.2.3 Incoming tree-sparse KKT system

For the incoming tree-sparse NLP the Lagrangian finally reads

$$\mathcal{L}(x, u, \lambda, v, \mu, \eta, \xi) = \sum_{j \in V} \left(\mathcal{L}_j(x_j, \lambda_j, v_j, \mu, \eta_j) + \mathcal{L}_{ij}(x_i, u_j, \lambda_j, v_{ij}, \mu, \xi_j) \right)$$

$$\tag{13}$$

with

$$\mathcal{L}_j(x_j, \lambda_j, v_j, \mu, \eta_j) = \phi_j(x_j) + \lambda_j^T x_j - v_j^T r_j(x_j) - \mu^T e_j(x_j)$$
$$- (\eta_j^-)^T (x_j - x_j^-) - (\eta_j^+)^T (x_j^+ - x_j)$$

and

$$\mathcal{L}_{ij}(x_i, u_j, \lambda_j, v_{ij}, \mu, \xi_j) = \phi_{ij}(x_i, u_j) - \lambda_j^T h_j(x_i, u_j)$$
$$- v_{ij}^T r_{ij}(x_i, u_j) - \mu^T e_{ij}(x_i, u_j)$$
$$- (\xi_j^-)^T (u_j - u_j^-) - (\xi_j^+)^T (u_j^+ - u_j).$$

Here, the barrier matrices and vectors are

$$\Phi_j^x = \mathrm{Diag}(x_j^+ - x_j)^{-1} \mathrm{Diag}(\eta_j^+) + \mathrm{Diag}(x_j - x_j^-)^{-1} \mathrm{Diag}(\eta_j^-),$$
$$\Phi_j^u = \mathrm{Diag}(u_j^+ - u_j)^{-1} \mathrm{Diag}(\xi_j^+) + \mathrm{Diag}(u_j - u_j^-)^{-1} \mathrm{Diag}(\xi_j^-),$$
$$\varphi_j^x = \eta_j^+ - \eta_j^- - \beta \left(\mathrm{Diag}(x_j^+ - x_j)^{-1} - \mathrm{Diag}(x_j - x_j^-)^{-1} \right) e,$$
$$\varphi_j^u = \xi_j^+ - \xi_j^- - \beta \left(\mathrm{Diag}(u_j^+ - u_j)^{-1} - \mathrm{Diag}(u_j - u_j^-)^{-1} \right) e,$$
$$\Psi_{ij} = \mathrm{Diag}(r_{ij})^{-1} \mathrm{Diag}(v_{ij}),$$
$$\psi_{ij} = r_{ij} - \beta \mathrm{Diag}(v_{ij})^{-1} e,$$
$$\Psi_j = \mathrm{Diag}(r_j)^{-1} \mathrm{Diag}(v_j),$$
$$\psi_j = r_j - \beta \mathrm{Diag}(v_j)^{-1} e,$$

and the partial derivatives of the Lagrangian read

$$K_j = \nabla^2_{u_j u_j} \mathcal{L}_{ij}(x_i, u_j, \zeta), \quad J_j = \nabla^2_{x_i u_j} \mathcal{L}_{ij}(x_i, u_j, \zeta), \tag{14a}$$

$$H_{ij} = \nabla^2_{x_i x_i} \mathcal{L}_{ij}(x_i, u_j, \zeta), \quad H_j = \nabla^2_{x_j x_j} \mathcal{L}_j(x_j, \zeta) + \sum_{k \in S(j)} H_{jk}, \tag{14b}$$

$$d_j = \nabla_{u_j} \mathcal{L}_{ij}(x_i, u_j, \zeta), \quad f_j = \nabla_{x_j} \mathcal{L}_j(x_j, \zeta) + \sum_{k \in S(j)} \nabla_{x_j} \mathcal{L}_{jk}(x_j, u_k, \zeta), \tag{14c}$$

$$E_j = \nabla_{u_j} h_j(x_i, u_j), \quad G_j = \nabla_{x_i} h_j(x_i, u_j), \tag{14d}$$

$$F_{ij}^r = \nabla_{x_i} r_{ij}(x_i, u_j), \tag{14e}$$

$$D_j^r = \nabla_{u_j} r_{ij}(x_i, u_j), \quad F_j^r = \nabla_{x_j} r_j(x_j), \tag{14f}$$

$$D_j = \nabla_{u_j} e_{ij}(x_i, u_j), \quad F_j = \nabla_{x_j} e_j(x_j) + \sum_{k \in S(j)} \nabla_{x_j} e_{jk}(x_j, u_k). \tag{14g}$$

Again, with

$$\hat{K}_j = K_j + \Phi_j^u, \qquad\qquad \hat{H}_j = H_j + \Phi_j^x,$$

$$\hat{d}_j = d_j - \varphi_j^u, \qquad\qquad \hat{f}_j = f_j - \varphi_j^x,$$

$$\bar{h}_j = h_j(x_i, u_j) - x_j, \qquad e_V = \sum_{j \in V} \left(e_{ij}(x_i, u_j) + e_j(x_j) \right)$$

we obtain the KKT system (6) in node-wise representation with Δ's omitted:

$$J_j x_i + \hat{K}_j u_j - E_j^T \lambda_j - (D_j^r)^T v_{ij} - D_j^T \mu + \hat{d}_j = 0, \quad j \in V, \qquad (15a)$$

$$\hat{H}_j x_j + \sum_{k \in S(j)} J_k^T u_k + \lambda_j - \sum_{k \in S(j)} G_k^T \lambda_k \qquad\qquad (15b)$$

$$- \sum_{k \in S(j)} (F_{jk}^r)^T v_{jk} - (F_j^r)^T v_j - F_j^T \mu + \hat{f}_j = 0, \quad j \in V, \qquad (15c)$$

$$G_j x_i + E_j u_j - x_j + \bar{h}_j = 0, \quad j \in V, \qquad (15d)$$

$$F_{ij}^r x_i + D_j^r u_j + \Psi_{ij}^{-1} v_{ij} + \psi_{ij} = 0, \quad j \in V, \qquad (15e)$$

$$F_j^r x_j + \Psi_j^{-1} v_j + \psi_j = 0, \quad j \in V, \qquad (15f)$$

$$\sum_{j \in V} D_j u_j + \sum_{j \in V} F_j x_j + e_V = 0. \qquad (15g)$$

3 Tree-sparse Quasi-Newton updates

Standard interior-point methods are second-order methods, i.e., they use local information given by the Hessian of the Lagrangian $\mathcal{L}$. In many applications, the evaluation of second-order derivatives is prohibitively expensive and one is interested in using Quasi-Newton updates that approximate the Hessian.

For the following let an initial point $y^{(0)}$ and an initial approximation $B^{(0)}$ of the Hessian $\nabla_{yy}^2 \mathcal{L}(y, \zeta)$ be given, where ζ again denotes all relevant dual variables. Then, omitting the current iteration index k and denoting iteration $k + 1$ by superscript "+", three standard approaches of updating the Hessian approximations are the symmetric rank-two BFGS update rule

$$B^+ = B - \frac{B s s^T B}{s^T B s} + \frac{g g^T}{g^T s}, \qquad (16)$$

the symmetric rank-one (SR1) update formula

$$B^+ = B + \frac{r r^T}{r^T s}, \qquad (17)$$

and the somewhat less popular PSB rule

$$B^+ = B + \frac{r s^T + s r^T}{s^T s} - \frac{(r^T s) s s^T}{(s^T s)^2}, \qquad (18)$$

where the iteration data is given by

$$s := y^+ - y, \quad g := \nabla\mathcal{L}(y^+, \zeta^+) - \nabla\mathcal{L}(y, \zeta^+), \quad r := g - Bs. \tag{19}$$

An overview of Quasi-Newton methods for unconstrained and constrained optimization can be found, e.g., in Dennis and Moré (1977), Dennis and Schnabel (1996), Fletcher (2013), Nocedal and Wright (2006) and the references therein. General-purpose implementations of algorithms using Quasi-Newton approaches for constrained nonlinear optimization include, e.g., KNITRO (Byrd et al. 2006, 2000) and SNOPT (Gill et al. 2002).

3.1 Quasi-Newton methods for tree-sparse problems

Standard update formulas like (16)–(18) produce dense Hessian approximations that destroy the specific block structure of the tree-sparse NLPs. Quasi-Newton methods in large-scale optimization should generally not alter the sparsity pattern too much, i.e., the Hessian update strategy should keep additional fill-in at a minimum level. Such sparse Quasi-Newton approaches for unconstrained and constrained optimization are considered repeatedly in the literature; see, e.g., Fletcher (1995), Gill et al. (1984), Liu and Nocedal (1989), Lucia (1983), Powell and Toint (1981), Toint (1981).

In fact, the tree-sparse NLPs are designed such that their block structure is entirely preserved by suitable block-sparse Hessian approximations. The key idea is to apply the update formulas node-wise rather than globally, which goes back to multiple shooting SQP methods for deterministic optimal control problems (Bock and Plitt 1985). As an additional benefit the updates have much higher rank.

Before we explicitly discuss tree-sparse Quasi-Newton updates for specific control forms we briefly review the concept of (partially) separable functions. A function $\varphi \colon \mathbb{R}^n \to \mathbb{R}$ is called *partially separable* if it can be written as

$$\varphi(y) = \sum_{i=1}^{M} \varphi_i(y_{I(i)}),$$

where each function φ_i depends only on a subset of the variables that is indexed by $I(i) \subseteq \{1, \dots, n\}$ for $i = 1, \dots, M$. We call the function φ *(completely) separable* if, in addition, $I(i) \cap I(j) = \emptyset$ for all $i \neq j$.

Quasi-Newton methods for arbitrary partially separable functions have first been developed in Griewank and Toint (1982a, b).

3.2 Tree-sparse Hessian updates for outgoing control problems

For tree-sparse NLPs in outgoing control form the Lagrangian (10),

$$\mathcal{L}(x, u, \lambda, v, \mu, \eta, \xi) = -\lambda_0^T h_0 + \sum_{j \in V} \mathcal{L}_j(x_j, u_j, \lambda_j, \lambda_{S(j)}, v_j, \mu, \eta_j, \xi_j),$$

is completely separable with respect to the primal node variables, $y_j = (x_j, u_j)$, $j \in V$. Thus, its Hessian is block-diagonal, and each block $\nabla^2_{y_j y_j} \mathcal{L}_j(y_j, \zeta)$ can be approximated individually by some B_j obtained with the same update formula based on

$$
\begin{aligned}
s_j &= y_j^+ - y_j, \\
g_j &= \nabla_{y_j} \mathcal{L}_j(y_j^+, \zeta^+) - \nabla_{y_j} \mathcal{L}_j(y_j, \zeta^+), \\
r_j &= g_j - B_j s_j.
\end{aligned}
$$

Here, B_j approximates the diagonal block (*without* barrier terms)

$$
\nabla^2_{y_j y_j} \mathcal{L}_j(y_j, \zeta) = \begin{bmatrix} H_j & J_j^T \\ J_j & K_j \end{bmatrix},
\tag{20}
$$

where the subblocks H_j, J_j, and K_j are given in (11). The overall Hessian approximation (again without barrier terms) is then given by

$$
\nabla^2_{yy} \mathcal{L}(y, \zeta) \approx \mathrm{Diag}(\{B_j\}_{j \in V}).
$$

Finally note that, since the Lagrangian (8) of the implicit NLP form is also completely separable, it admits analogous individual updates of the diagonal blocks $\nabla^2_{y_j y_j} \mathcal{L}_j(y_j, \zeta) = H_j$. No communication is needed in the distributed code.

3.3 Tree-sparse Hessian updates for incoming control problems

The Lagrangian of tree-sparse NLPs in incoming control form,

$$
\mathcal{L}(x, u, \lambda, v, \mu, \eta, \xi) = \sum_{j \in V} \left(\mathcal{L}_{ij}(x_i, u_j, \lambda_j, v_{ij}, \mu, \xi_j) + \mathcal{L}_j(x_j, \lambda_j, v_j, \mu, \eta_j) \right),
$$

is composed of two types of node functions, $\mathcal{L}_{ij}$ and $\mathcal{L}_j$ as given in (13). In contrast to the outgoing control case, this Lagrangian is only partially separable with respect to the primal node variables $y_j = (u_j, x_j)$. Thus, by setting $y_{ij} = (x_i, u_j)$, the overall Hessian $\nabla^2_{yy} \mathcal{L}$ will be approximated by the sum of updates $B_{ij} \approx \nabla^2_{y_{ij} y_{ij}} \mathcal{L}_{ij}$ and $B_j \approx \nabla^2_{x_j x_j} \mathcal{L}_j$, $j \in V$, whose subblocks overlap in part as detailed below.

We obtain the node-wise approximations

$$
B_{ij} \approx \nabla^2_{y_{ij} y_{ij}} \mathcal{L}_{ij}(x_i, u_j, \zeta) = \begin{bmatrix} H_{ij} & J_j^T \\ J_j & K_j \end{bmatrix},
\tag{21}
$$

where H_{ij}, J_j, and K_j are defined in (14). In a distributed code, B_{ij} has to be sent to the process that holds B_i. The approximations of the Hessians of $\mathcal{L}_j$ are given by

$$
B_j \approx \nabla^2_{x_j x_j} \mathcal{L}_j(x_j, \zeta) =: \tilde{H}_j.
\tag{22}
$$

The approximations B_{ij} and B_j are updated by applying, for instance, one of the update rules (16)–(18) where the respective differences of the iterates are

$$s_{ij} = \begin{pmatrix} x_i^+ - x_i \\ u_j^+ - u_j \end{pmatrix}, \quad s_j = x_j^+ - x_j, \tag{23}$$

the corresponding differences of the gradients of the Lagrangian are

$$g_{ij} = \begin{pmatrix} \nabla_{x_i}\mathcal{L}_{ij}(x_i^+, u_j^+, \zeta^+) - \nabla_{x_i}\mathcal{L}_{ij}(x_i, u_j, \zeta^+) \\ \nabla_{u_j}\mathcal{L}_{ij}(x_i^+, u_j^+, \zeta^+) - \nabla_{u_j}\mathcal{L}_{ij}(x_i, u_j, \zeta^+) \end{pmatrix}, \tag{24a}$$

$$g_j = \nabla_{x_j}\mathcal{L}_j(x_j^+, \zeta^+) - \nabla_{x_j}\mathcal{L}_j(x_j, \zeta^+), \tag{24b}$$

and, finally, $r_{ij} = g_{ij} - B_{ij}s_{ij}$ and $r_j = g_j - B_j s_j$.

To illustrate the resulting block structure, we consider a simple tree with root 0 and successors $S(0) = \{1, 2\}$. In this case, the Hessian of the Lagrangian (13) has the form

$$\nabla_{yy}^2 \mathcal{L}(y, \zeta) = \begin{bmatrix} K_0 & & & & & \\ & H_0 & J_1^T & & J_2^T & \\ & J_1 & K_1 & & & \\ & & & H_1 & & \\ & J_2 & & & K_2 & \\ & & & & & H_2 \end{bmatrix}$$

with variables $(u_0, x_0, u_1, x_1, u_2, x_2)$ and

$$H_0 = \tilde{H}_0 + H_{01} + H_{02}, \quad H_1 = \tilde{H}_1, \quad H_2 = \tilde{H}_2. \tag{25}$$

Hence, the subblocks K_j and $\tilde{H}_j$ are placed on the diagonal corresponding to the respective variables u_j and x_j. Moreover, each subblock H_{ij} is added to $\tilde{H}_i$, and J_j and J_j^T are placed on the anti-diagonals corresponding to the variable pair (x_i, u_j).

4 Tree-sparse inertia correction

Recall that the search direction in an interior-point method for NLP (1) is computed from system (7). Hence the reduced KKT matrix has to be invertible. In addition, to guarantee a descent direction, the Hessian $\bar{H}$ has to be positive definite on the null-space of the constraints matrix. These conditions are satisfied if and only if the reduced KKT matrix

$$\Omega = \begin{bmatrix} \bar{H} & C^T \\ C & 0 \end{bmatrix} \quad \text{with} \quad \bar{H} \in \mathbb{R}^{n \times n}, \quad C = \begin{bmatrix} G \\ F \end{bmatrix} \in \mathbb{R}^{m \times n} \tag{26}$$

has n positive and m negative eigenvalues (Nocedal and Wright 2006), i.e., inertia(Ω) = $(n, m, 0)$. Therefore, when necessary, the inertia condition is enforced by replacing Ω with

$$\Omega(\delta_c, \delta_r) = \begin{bmatrix} \bar{H} + \delta_c I & C^T \\ C & -\delta_r I \end{bmatrix}, \quad \delta_c, \delta_r \geq 0, \tag{27}$$

as follows. If C (and hence Ω) is rank-deficient, the associated zero eigenvalues are shifted into the negative region by choosing a fixed small value $\delta_r > 0$ (*regularization*). Then, if $\Omega(0, \delta_r)$ has less than n positive eigenvalues, δ_c is increased repeatedly until $\Omega(\delta_c, \delta_r)$ has the desired inertia (*convexification*). For details see Schmidt (2013), Vanderbei and Shanno (1997), Wächter and Biegler (2006).

For the inertia correction of the tree-sparse NLPs, additional structural properties of the associated factorizations of Ω come into play. First, in the *implicit* tree-sparse KKT system the general *regularization* $-\delta_r I$ in (27) splits into independent regularizations for each dynamics equation (9b) and for the global constraints (9d), which are applied to certain symmetric Schur complement blocks $Y_j \geq 0$ and $X_\emptyset \geq 0$ (see Steinbach 2001) by factorizing $Y_j + \delta_r I > 0$ (or $X_\emptyset + \delta_r I > 0$) unless $Y_j > 0$ (or $X_\emptyset > 0$). Thus, any rank deficiencies in C are handled locally, and the local refactorizations do not require any communication in the distributed code.

On the other hand, the general regularization $-\delta_r I$ would destroy the block structure of the tree-sparse KKT systems in outgoing or incoming *control form* since in both cases the factorization makes use of the fact that the matrix blocks to the right of G and below G^T are zero. These factorizations can only handle a regularization of the global constraints, not the dynamics,

$$\Omega^*(\delta_c, \delta_r) = \begin{bmatrix} \bar{H} + \delta_c I & G^T & F^T \\ G & 0 & 0 \\ F & 0 & -\delta_r I \end{bmatrix}. \tag{28}$$

While at first thought this might appear as a drawback, it is actually an advantage: with explicit dynamics (3b) or (4b), the term $-x_j$ creates identity blocks $-I$ along the diagonal of G. Hence G has always full rank by construction, and a regularization needs only be considered for F. A closer look at the tree-sparse KKT solution algorithms reveals that full rank of F is equivalent to $X_\emptyset > 0$ (where $X_\emptyset$ can be shown to be the Schur complement of the projection of F on $N(G)$, with $X_\emptyset \geq 0$). This is checked in the very last block operation: the Cholesky factorization of $X_\emptyset$, cf. step 18 of Table 1 for the incoming control case. Thus, if a regularization is required, only $X_\emptyset + \delta_r I > 0$ needs to be refactorized while the by far more expensive initial part of the factorization can be retained. Again there is no extra communication.

Second, to preserve sparsity, each tree-sparse factorization needs a stricter condition than the inertia condition above: while rank$(C) = m$ is required as before, $\bar{H}$ must be positive definite on the null space $N(G)$ rather than on $N(C) = N(G) \cap N(F)$. This is because all three factorizations form the Schur complement $X_\emptyset$ of the projection of F on $N(G)$. The stricter condition can easily be incorporated into the *convexification* heuristic since positive definiteness of $\bar{H} + \delta_c I$ on $N(G)$ is checked during the factorization. Of course, in problems without global constraints we have $F \in \mathbb{R}^{0 \times m}$ and there is no difference.

The structural details of the tree-sparse convexification are more involved than for the regularization. Positive definiteness of $\bar{H} + \delta_c I$ on $N(G)$ is equivalent to positive

Table 1 Operations of KKT solution algorithm for incoming control

	Factorization ↓	Inward subst. ↓	Outward subst. ↑
1:	$K_j \mathrel{+}= \Phi_j^u$		$-v_{ij} \leftarrow \Psi_{ij}(-v_{ij})$
2:	$H_j \mathrel{+}= \Phi_j^x$		$-v_j \leftarrow \Psi_j(-v_j)$
3:	$K_j \mathrel{+}= D_j^{r\,T} \Psi_{ij} D_j^r$	$d_j \mathrel{+}= D_j^{r\,T} \Psi_{ij} r_{ij}$	$-v_{ij} \mathrel{+}= D_j^r u_j$
4:	$J_j \mathrel{+}= D_j^{r\,T} \Psi_{ij} F_{ij}^r$		
5:	$H_i \mathrel{+}= F_{ij}^{r\,T} \Psi_{ij} F_{ij}^r$	$f_i \mathrel{+}= F_{ij}^{r\,T} \Psi_{ij} r_{ij}$	$-v_{ij} \mathrel{+}= F_{ij}^r x_i$
6:	$H_j \mathrel{+}= F_j^{r\,T} \Psi_j F_j^r$	$f_j \mathrel{+}= F_j^{r\,T} \Psi_j r_j$	$-v_j \mathrel{+}= F_j^r x_j$
7:	$K_j \mathrel{+}= E_j^T H_j E_j$	$d_j \mathrel{+}= E_j^T(H_j h_j + f_j)$	$-\lambda_j \mathrel{+}= H_j x_j$
8:	$K_j \rightarrow L_j L_j^T$		
9:	$J_j \mathrel{+}= E_j^T H_j G_j$		$x_j \mathrel{+}= E_j u_j$
10:	$H_i \mathrel{+}= G_j^T H_j G_j$	$f_i \mathrel{+}= G_j^T(H_j h_j + f_j)$	$x_j \mathrel{+}= G_j x_i$
11:	$F_i \mathrel{+}= F_j G_j$	$e_V \mathrel{+}= F_j h_j$	$-\lambda_j \mathrel{+}= F_j^T(\mu)$
12:	$D_j \mathrel{+}= F_j E_j$		
13:	$D_j \leftarrow D_j L_j^{-T}$	$h_j \leftrightarrow f_j$	
14:	$J_j \leftarrow L_j^{-1} J_j$	$d_j \leftarrow L_j^{-1} d_j$	$u_j \leftarrow -L_j^{-T} u_j$
15:	$H_i \mathrel{-}= J_j^T J_j$	$f_i \mathrel{-}= J_j^T d_j$	$u_j \mathrel{+}= J_j x_j$
16:	$F_i \mathrel{-}= D_j J_j$		
17:	$X_V \mathrel{+}= D_j D_j^T$	$e_V \mathrel{-}= D_j d_j$	$u_j \mathrel{+}= D_j^T(-\mu)$
18:	$X_\emptyset \rightarrow LL^T$	$-\mu = L^{-1} e_\emptyset$	$-\mu \leftarrow L^{-T}(-\mu)$

In step 18, $X_\emptyset$, $e_\emptyset$ denote X_V, e_V after finishing ("removing") every node $j \in V$ in step 17. See Steinbach (2002) for details

definiteness of properly modified blocks K_j in every node, see steps 3 and 7 in Table 1. Thus, if this does not hold, the Cholesky factorization of some modified block K_j will fail (step 8), and $\Omega^*(\delta_c, \delta_r)$ has to be refactorized from scratch with an increased value of δ_c.

In order to avoid refactorizations of the entire matrix, different convexification parameters δ_{cj} can be used in each node j to modify the local Hessian blocks,

$$(H_j, K_j) \rightarrow (H_j + \delta_{cj} I, K_j + \delta_{cj} I) \quad \text{or} \quad (H_j, K_j) \rightarrow (H_j, K_j + \delta_{cj} I). \quad (29)$$

This way, only local refactorizations are needed. However, in contrast to the uniform shift of all eigenvalues of $\bar{H}$ by δ_c, individual local shifts δ_{cj} may produce a highly inhomogeneous change of the subproblem's geometry. As a remedy, one may also consider combined global and local convexifications, as suggested in Hübner (2016). This allows for a wide range of possible heuristics that we do not want to explore here.

Let us finally discuss the inertia correction with local equality constraints. In all tree-sparse factorizations, these constraints are eliminated by local projections at the very beginning of the algorithm, under the assumption that the Jacobians of e_{ij}^l, e_j^l

in (5) have full rank. This produces a projected KKT system without local constraints which has precisely the form (9), (12), or (15). If any of the local constraints may require a regularization, this procedure has to be modified as follows. All potentially rank-deficient local constraints are modeled as global constraints. However, it turns out that each of them augments the Schur complement $X_\emptyset \geq 0$ mentioned above with an independent diagonal block $X_j^l \geq 0$. Thus, similar to the implicit case, the general regularization splits into independent local regularizations that are applied immediately when reaching the respective block by factorizing $X_j^l + \delta_r I > 0$ (or $X_\emptyset + \delta_r I > 0$) unless $X_j^l > 0$ (or $X_\emptyset > 0$). Local regularizations are again automatically independent, require no communication, and in contrast to the convexifications a refactorization of $\Omega^*(\delta_c, \delta_r)$ from scratch is never needed.

5 Case studies

The main goal of this section is to show that the tree-sparse Hessian updates and the tree-sparse inertia correction work efficiently: they provide natural extensions of the existing tree-sparse algorithms to deal with nonconvexity and with unavailable or expensive second-order derivatives.

In Sect. 5.1 and 5.2 we consider robust moving horizon control problems for a double integrator and a bioreactor, respectively. These examples are both nonconvex. The second problem does not provide explicit evaluations of the Hessian of the Lagrangian, so we use a Quasi-Newton approach based on the tree-sparse Hessian updates of Sect. 3. All optimization problems are solved using the interior-point code Clean::IPM (Schmidt 2013) with a tree-sparse KKT solver. The KKT solver incorporates the proposed inertia correction heuristic of Sect. 4 to address nonconvexity.

Moving horizon controllers (MHC) compensate random disturbances of a process by measuring or estimating the current disturbance in regular intervals and solving a dynamic optimization problem over a certain *prediction horizon T* to determine the current corrective action. Robust MHC incorporate an explicit stochastic model of future disturbances up to some *stochastic horizon $T_s \leq T$* whereas standard MHC simply ignore future disturbances ($T_s = 0$).

5.1 Nonlinear double integrator

In this case study, we consider a moving horizon controller (MHC) for stabilizing a perturbed nonlinear double integrator. The dynamics is given by the discrete time model proposed in Lazar et al. (2008),

$$x_1(t + 1) = x_1(t) + x_2(t) + \frac{1}{40}\left(x_1(t)^2 + x_2(t)^2\right) + \frac{1}{2}u(t) + d(t), \quad (30a)$$

$$x_2(t + 1) = x_2(t) + \frac{1}{40}\left(x_1(t)^2 + x_2(t)^2\right) + u(t). \quad (30b)$$

Herein, the state (x_1, x_2) is driven by the control $u \in [-2, 2]$, and the first state x_1 is perturbed by some dynamic disturbance d with nominal value $d^{nom} = 0$. The task is
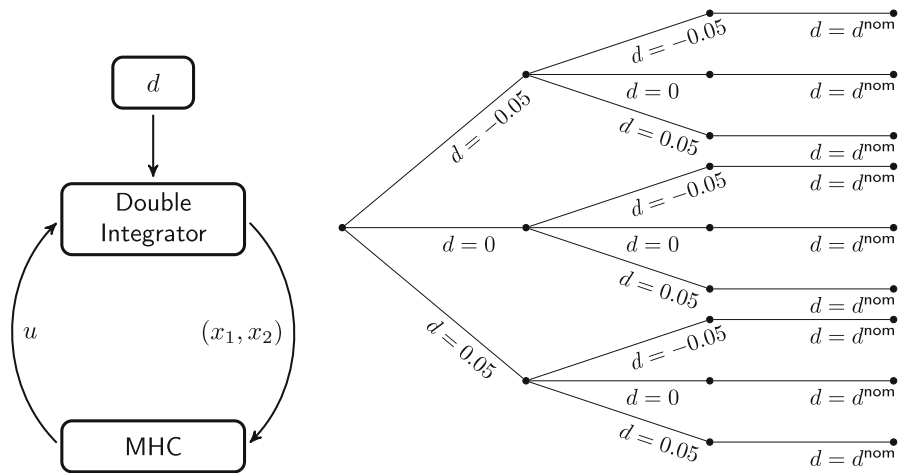
Fig. 1 Moving horizon control of the double integrator (left) and scenario tree with prediction horizon $T = 3$ and stochastic horizon $T_s = 2$ (right)

to keep the system close to its reference point, $(x_1^*, x_2^*) = (0, 0)$. In the absence of disturbances, the reference state (x_1^*, x_2^*) is a fixed point of the dynamics (30).

As in Lucia and Engell (2012), the disturbance may take one of three values, $d(t) \in \{-0.05, 0, 0.05\}$, with respective probabilities 0.2, 0.4, and 0.4. Again as in Lucia and Engell (2012), the objective is

$$\phi^L(x, u) = (x - x^*)^T Q(x - x^*) + Ru^2 \tag{31}$$

with $Q = I$ and $R = 0.15$, a standard quadratic tracking functional with control costs. At each sampling time, the double integrator receives a control signal $u(t)$ from the MHC and the random disturbance $d(t)$ is observed. The resulting new state $(x_1(t + 1), x_2(t + 1))$ is then sent back to the controller; see Fig. 1.

5.1.1 Tree-sparse formulation

The moving horizon optimization problem that is solved at each sampling time to determine the control signal u includes the dynamic model (30) and the cost (31). We formulate it as a tree-sparse NLP in outgoing control form with probability-weighted node objectives

$$\phi_j(x_j, u_j) = p_j(x_j^T Q x_j + Ru_j^2), \quad j \in V, \tag{32}$$

dynamics

$$h_j(x_i, u_i) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_i + \begin{pmatrix} 1/2 \\ 1 \end{pmatrix} u_i + \frac{\|x_i\|_2^2}{40} \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} d_j \\ 0 \end{pmatrix}, \quad j \in V \backslash \{0\}, \tag{33}$$

and simple control bounds

$$u_j \in [-2, 2], \quad j \in V. \tag{34}$$
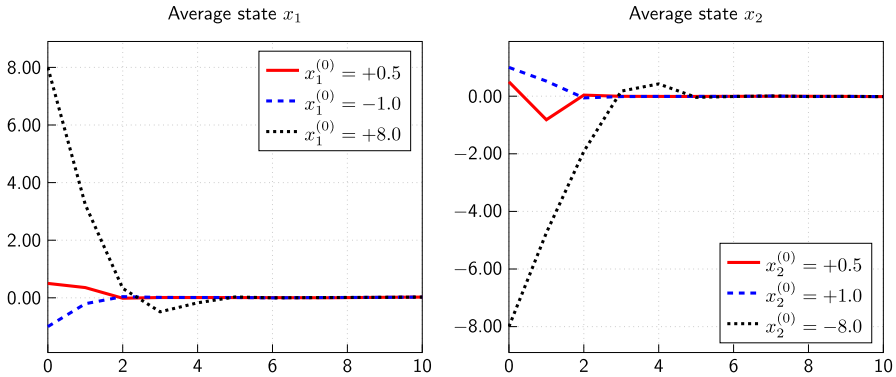
Fig. 2 Progress of average states of perturbed double integrator for three initial states with deterministic MHC ($T_s = 0$, $T = 3$)

The initial state $\hat{x}_0$ is given as $h_0 = \hat{x}_0$. A possible scenario tree is shown in Fig. 1.

5.1.2 Control towards reference point

In a first test, we evaluate the performance of the deterministic controller ($T_s = 0$) for bringing the system from three given initial states close to the reference point (x_1^*, x_2^*). The MHC uses the prediction horizon $T = 3$ and analytic second-order derivatives. We generate random disturbances at $t = 1, \ldots, 10$ and apply the MHC for 10 time steps. For each initial value, this is repeated 50 times with different series of random disturbances. Figure 2 shows the mean values of the states over the 10 time steps. For each of the three initial states, the mean values come very close to (x_1^*, x_2^*) within the first 5 time steps, which confirms the proper operation of the controller.

5.1.3 Hold the reference point with minimal costs

In the second test series, the performance of the robust controller is tested for keeping the perturbed system close to the initial state (x_1^*, x_2^*) over 20 time steps. Here we consider prediction horizons $T = 3$ and $T = 10$ with respective stochastic horizons $T_s \in \{0, 1, 2, 3\}$ and $T_s \in \{0, 1, 2, 3, 5, 7\}$. Again we use analytic second-order derivatives, and each test is run with the same set of 50 series of random disturbances at $t = 1, \ldots, 20$. Figure 3 illustrates the resulting averages of accumulated costs, states, and control over 20 steps. We observe that the 10 different controllers determined by (T, T_s) show visible differences in the mean values of u and x_2 only during the first five time steps (with an average control of zero at $t = 5$). Moreover, the length of the prediction horizon has no visible influence: the plots for $T = 3$ and $T = 10$ combined with $T_s \in \{0, 1, 2, 3\}$ look exactly identical. A close examination of the data reveals that differences after $t = 5$ and between $T = 3$ and $T = 10$ are indeed very small with values in the order of 10^{-4} to 10^{-3}.

The average accumulated costs (top of Fig. 3) demonstrate that the performance of the controller improves when uncertainties are included. Adding uncertainties in the
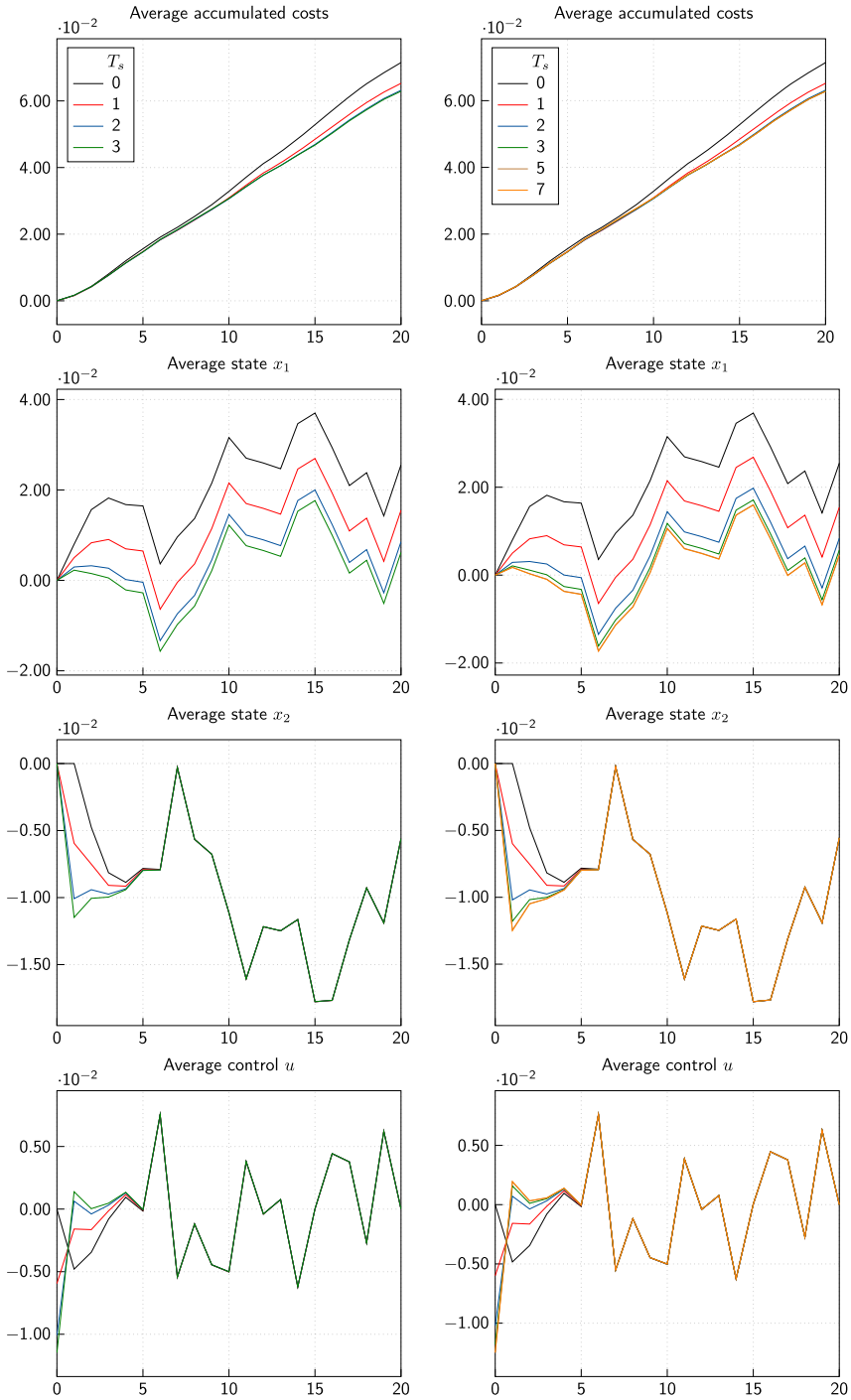
Fig. 3 Progress of double integrator: averages of accumulated costs, of states x_1, x_2, and of control u for prediction horizons $T = 3$ (left) and $T = 10$ (right) with several stochastic horizons T_s

first time step, i.e., increasing T_s from 0 to 1, reduces the resulting costs at the final step $t = 20$ by approximately 9%. Increasing the stochastic horizon to $T_s = 2$ leads to a further cost reduction of approximately 3%. Larger stochastic horizons ($T_s > 2$) have no significant effect on the costs. The cost reductions are obtained by balancing the disturbances in the first 5 time steps of the process. The control signal is adjusted to reduce the deviation of x_1 from the reference state $x_1^* = 0$. This reduces the costs due to deviations of x_1 and increases the costs due to deviations of x_2 and due to nonzero control signals. In other words, within the critical first 5 time steps, the incurred costs are "moved" from state x_1 to state x_2 and to the control u. This strategy pays off in the further process where the costs due to u and x_2 are identical for all considered values of T_s while the costs due to x_1 depend on its value at time step 5.

5.1.4 Exact Hessians versus approximations

Here we solve the robust control problem (32)–(34) with a fixed prediction horizon $T = 12$ and increasing stochastic horizons $T_s \in \{1, \ldots, 12\}$. We measure the total time for IPM solution and for evaluating the NLP data with either analytic Hessians or SR1 updates or PSB updates. Each test is run 50 times for different initial states, and the averages of IPM solution time, NLP evaluation time, and iteration counts are measured, see Fig. 4. The problem sizes are given in Table 2 together with total runtimes of the IPM with analytic Hessian evaluations. All computations are carried out on a single core of a workstation with 48 GiB of RAM and 12 X5675 cores running at 3.07 GHz.

Let us first consider the case of analytic Hessian evaluations. The average number of iterations is almost independent of T_s, which indicates a certain "scalability" of the stochastic model. The NLP evaluation time is lower than with both Quasi-Newton updates, thus analytic Hessians are cheaper than approximations in this problem. Finally we note that the constraints matrices cannot become rank-deficient and thus the regularization remains inactive for all 12 values of T_s and all 50 initial values.

With SR1 updates we observe the lowest iteration counts of all three Hessian versions on small problems ($T_s < 8$) but the highest iteration counts on large problems ($T_s > 9$). The convexification is again inactive in all runs, which indicates that the tree-sparse SR1 updates provide good Hessian approximations.

With PSB updates, the iteration count varies with T_s but remains significantly smaller than with analytic Hessians. This yields the smallest total runtimes on large problems ($T_s > 9$) although the NLP evaluation time per iteration is significantly larger than for analytic Hessians. Surprisingly, most runs require a convexification to solve the optimization problems.

5.2 Nonlinear bioreactor

This case study is concerned with a robust nonlinear moving horizon controller that keeps a bioreactor in a steady state of production. The problem is proposed as a benchmark in Ungar (1990) and has been studied, e.g., in Lucia and Engell (2012), Lucia et al. (2012). The plant consists of a continuous flow stirred tank reactor containing
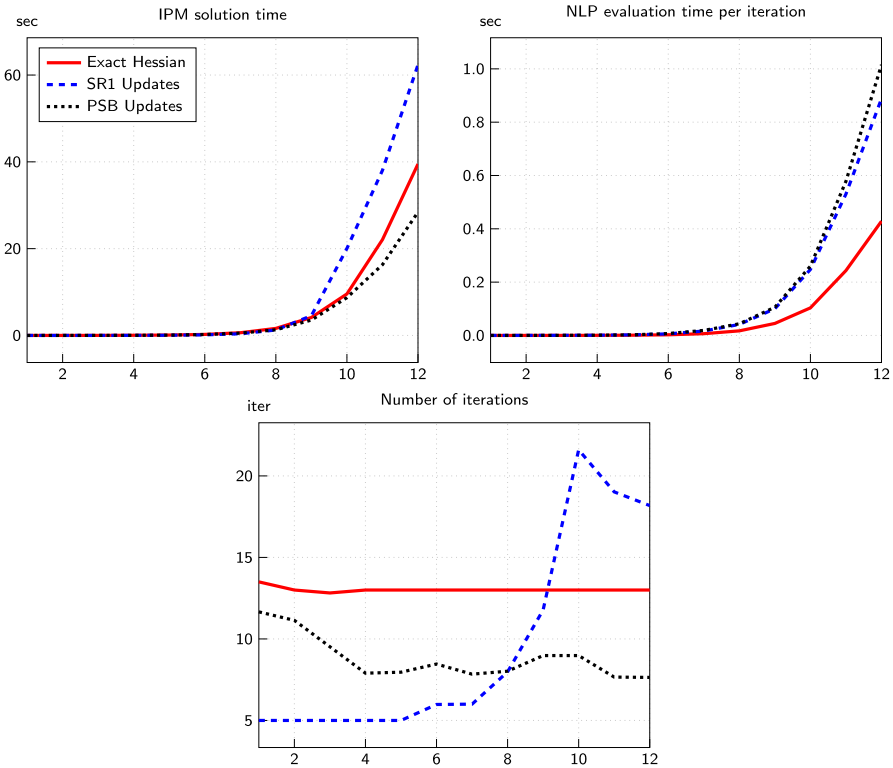
Fig. 4 Double integrator: average IPM performance vs.stochastic horizon T_s for different algorithm con-figurations

a mixture of water and cells. The latter consume nutrients and produce (desired and undesired) products as well as more cells. The volume of the mixture is constant and its composition is adjusted by a water stream that feeds nutrients into the tank at the inlet and that contains nutrients and cells at the outlet. The dynamic model of the bioreactor is given by the ODE system

$$\dot{x}_1 = f_1(x_1, x_2, v) = -x_1 v + x_1(1 - x_2)e^{x_2/\gamma}, \tag{35a}$$

$$\dot{x}_2 = f_2(x_1, x_2, v) = -x_2 v + x_1(1 - x_2)e^{x_2/\gamma}\frac{1 + \beta}{1 + \beta - x_2}, \tag{35b}$$

where x_1 and x_2 are dimensionless amounts of cell mass and nutrient, respectively, and v is the flow rate of the water stream. The respective physical bounds are

$$x_1(t), x_2(t) \in [0, 1], \quad v(t) \in [0, 2]. \tag{36}$$

The ODE system (35) describes the rates of change in the amounts of cells x_1 and nutrients x_2, respectively, that result from the respective amounts $-x_1 v$ and $-x_2 v$ leaving the tank and from the metabolism of the cells. The cell growth is represented

Table 2 Double integrator: problem sizes and average IPM solution times with analytic Hessian evaluations

T_s	Nodes	Scenarios	Variables	Equalities	Bounds	Time (s)
1	37	3	111	74	222	0.00
2	103	9	309	206	618	0.01
3	283	27	849	566	1698	0.01
4	769	81	2307	1538	4614	0.04
5	2065	243	6195	4130	12 390	0.09
6	5467	729	16 401	10,934	32,802	0.24
7	14,215	2187	42,645	28,430	85,290	0.62
8	36,085	6561	108,255	72,170	216,510	1.63
9	88,573	19,683	265,719	177,146	531,438	4.14
10	206,671	59,049	620,013	413,342	1,240,026	9.56
11	442,867	177,147	1,328,601	885,734	2,657,202	22.07
12	797,161	531,441	2,391,483	1,594,322	4,782,966	39.50

by $x_1(1 - x_2)e^{x_2/\gamma}$, where γ is the uncertain nutrient consumption parameter with nominal value $\gamma^{\mathrm{nom}} = 0.48$. The rate of cell growth β depends very mildly on the composition of the mixture in the tank. We regard it as constant, $\beta^{\mathrm{nom}} = 0.02$.

When feeding nutrients to the bioreactor with a constant flow rate v, the system has a Hopf bifurcation at a certain flow rate v^{H} that depends on the values of γ and β. For $v < v^{\mathrm{H}}$, System (35) stabilizes at a unique fixed point (x_1^*, x_2^*) whereas it becomes unstable for $v \geq v^{\mathrm{H}}$. For the nominal parameter values, the Hopf bifurcation occurs at the flow rate $v^{\mathrm{H}} = 0.829$. This value decreases with an increasing value of γ or a decreasing value of β as shown in Fig. 5.

The desired steady state of production, $(x_1^*, x_2^*) \approx (0.1236477, 0.8760318)$, is close to the Hopf bifurcation and is obtained for the constant flow rate $v^* = 0.769$ with nominal parameter values. The parameter γ is assumed to be normally distributed with small variance, $\gamma \sim N(\gamma^{\mathrm{nom}}, 0.005)$. As in Lucia and Engell (2012), standard quadratic costs are applied that penalize deviations from the reference state x_1^* with the factor 200 and changes of the flow rate with the factor 75.

5.2.1 Tree-sparse formulation

The optimization problem for the bioreactor is formulated as a tree-sparse NLP with incoming control (4). State variables $x_j \in \mathbb{R}^3$ consist of the two states x_1, x_2, and the flow rate v of the ODE system (35). The control $u_j \in \mathbb{R}$ models the *change* of the *piecewise constant* flow rate v at time $t(i)$, yielding dynamics $x_j = g_j(x_i, u_j) = (w_j(x_i, u_j), x_{i,3} + u_j)$ where

$$w_j(x_i, u_j) = \begin{pmatrix} x_{i,1} \\ x_{i,2} \end{pmatrix} + \int_{t(i)}^{t(j)} \begin{pmatrix} f_1(x_1(t), x_2(t), x_{i,3} + u_j) \\ f_2(x_1(t), x_2(t), x_{i,3} + u_j) \end{pmatrix} dt. \qquad (37)$$
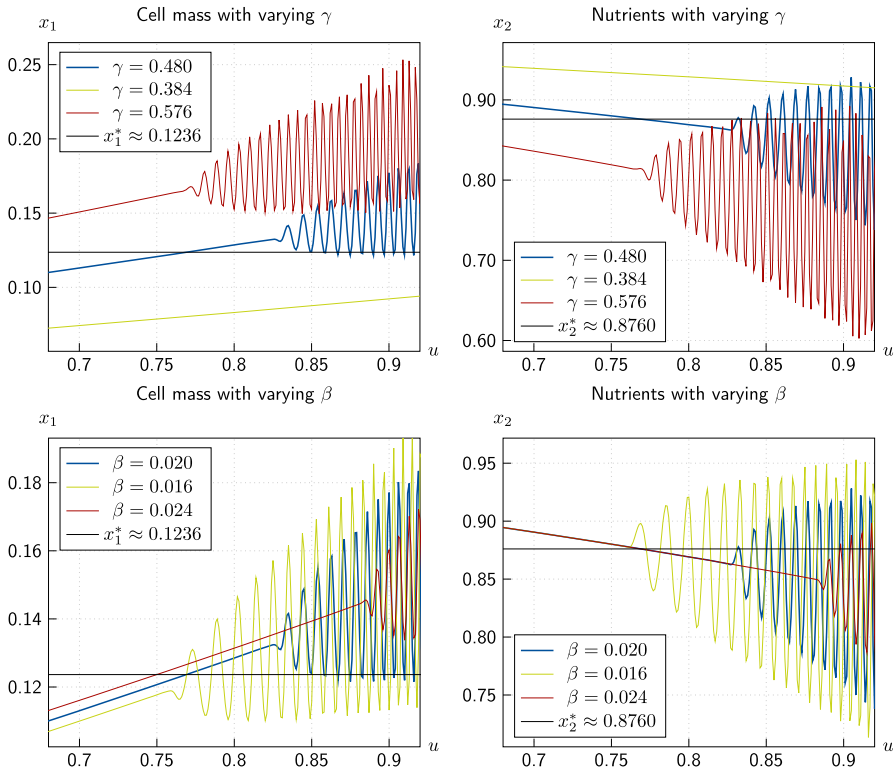
Fig. 5 Hopf bifurcations of the bioreactor for varying parameters γ and β

Here the flow rate is modeled as a state variable since its difference $u_j = x_{j,3} - x_{i,3}$ enters into the objective,

$$\phi_{ij}(x_i, u_j) = \frac{p_j}{2} R u_j^2, \quad \phi_j(x_j) = \frac{p_j}{2} (x_j - \bar{x}^*)^T Q (x_j - \bar{x}^*),$$

with $R = 75$, $Q = \text{Diag}(200, 0, 0)$, and $\bar{x}^* = (x_1^*, x_2^*, 0)$. Finally, physical bounds (36) are incorporated as simple state bounds, $x_j^- = (0, 0, 0)$ and $x_j^+ = (1, 1, 2)$.

It turns out that with increasing stochastic horizon T_s the robust control problem becomes extremely hard to solve due to the highly nonlinear dynamics. Therefore we use as scenario tree a simple fan ($T_s = 1$) with n_s scenarios, see Fig. 6.

5.2.2 Keep a steady state of production

In the following, we consider 13 instances of the benchmark problem differing in the prediction horizon T or in the number of scenarios n_s. The reactor is started in the reference state and run for 40 s with a sampling interval of 100 ms. At each sampling time, the plant receives a new control signal from the MHC and the random value of γ
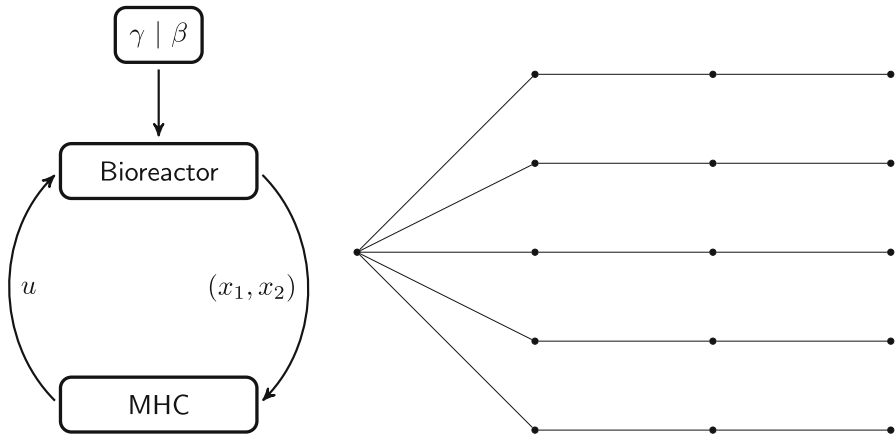
Fig. 6 Process control of the bioreactor (left) and scenario tree with prediction horizon $T = 3$ and $n_s = 5$ scenarios (right)

is sampled, see Fig. 6. Thus, each test run requires solving 400 optimization problems. Each of those tests is run 50 times with different series of random disturbances. We compute variances of the cell mass x_1 and averages of accumulated costs, of x_1, and of the flow rate v.

As in Sect. 5.1, the tree-sparse NLPs are solved using Clean::IPM where solutions of the IVP (37) are computed using a semi-implicit extrapolation method (Bader and Deuflhard 1983). The Hessians are approximated by tree-sparse SR1 updates, which turned out to be the most reliable choice here. All computations in this section have been executed on a single core of a workstation with 16 GiB of RAM and an Intel(R) Core i7-3770 quad-core processor running at 3.40 GHz.

In some test runs, the IPM does not converge for all 400 optimization problems, which demonstrates the difficulty of these problems. To keep the results comparable, we exclude all test runs that do not succeed in each of the 13 benchmark instances. In the end we are left with solutions for 28 of the 50 disturbance series.

The results are given in Figs. 7 and 8. The plots show averages of accumulated costs (upper left), of the cell mass x_1 (upper right), the variance of x_1 (lower left), and average flow rates (lower right) vs. time. Two situations are shown in the figures. First, in Fig. 7, the bioreactor is regulated using deterministic optimization problems ($n_s = 1$) and an increasing prediction horizon T. Second, in Fig. 8, the MHC uses stochastic optimization problems with fixed prediction horizon ($T = 5$) and a varying number of scenarios n_s. In both situations, increasing the free tree parameter (T or n_s) improves the performance of the controller. The average accumulated costs decrease monotonously, which is achieved by damping the effect of the perturbation on the cell mass x_1. After experiencing disturbances, x_1 oscillates about the reference state x_1^*. The amplitude of this oscillation is reduced by adjusting the flow rate v, which reduces the costs caused by the deviations of x_1.

The results in Fig. 7 show that a certain minimal prediction horizon is required to produce a significant effect. Using the value $T = 5$ reduces the costs by less than 5%
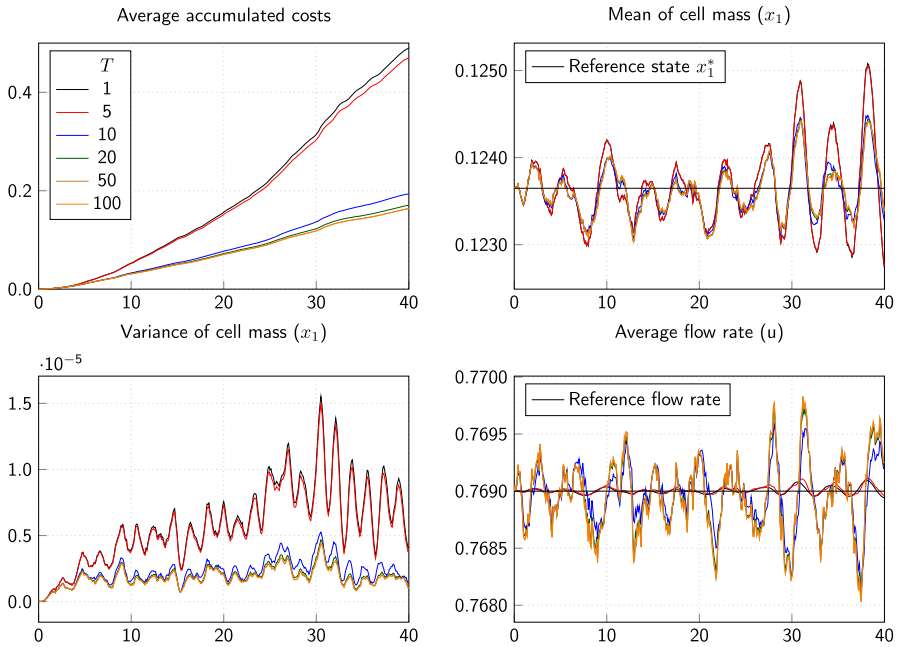
Fig. 7 Performance of the bioreactor for deterministic problems ($n_s = 1$) with increasing prediction horizons (T)
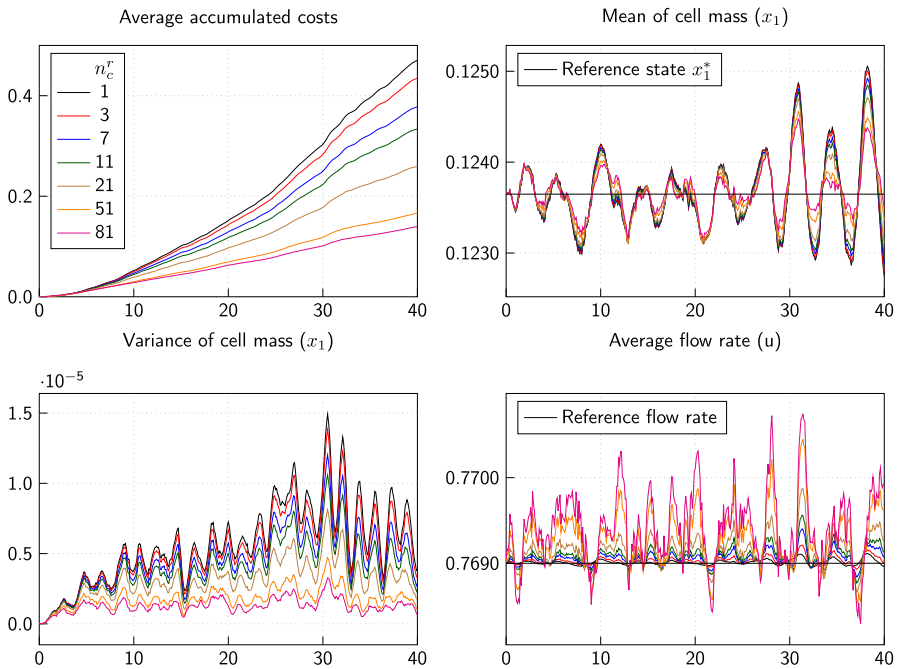

Fig. 8 Performance of the bioreactor for stochastic problems with fixed prediction horizon $T = 5$ and increasing number of scenarios n_s

Table 3 Problem sizes, IPM solution times (s) and NLP evaluation times (s) of the problems in Fig. 9

Case	T	n_s	Var.	Equ.	Bnd.	Iter.	IPM	NLP
1	100	1	404	303	606	8	0.015	0.011
2	5	51	1024	768	1536	9	0.037	0.028
3	10	3	124	93	186	6	0.005	0.003

Fig. 9 Bioreactor: large prediction horizon vs.large number of scenarios
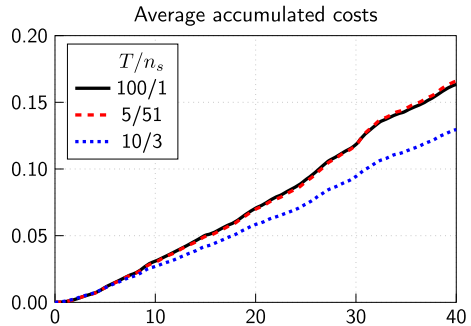
in comparison to $T = 1$ whereas the value $T = 10$ yields a reduction of more than 60%. However, increasing T further yields no significant improvement. The value $T = 20$, for instance, produces a reduction of only 65%.

The results in Fig. 8 show that the robust control approach with its explicit model of uncertainties improves the controller performance. The total costs are reduced by 7% with $n_s = 3$, and by 70% with $n_s = 81$. Of course, each additional scenario causes higher computational effort while the relative reduction of costs per scenario decreases. For instance, we have 3.5% reduction of costs per scenario for $n_s = 3$ and 0.88% per scenario for $n_s = 81$.

The results of Figs. 7 and 8 show that the goal of reducing the costs is achieved in both situations. However, increasing the number of scenarios n_s is computationally more expensive than enlarging the prediction horizon T, as can be seen in Table 3. The optimization problems of case 1 ($T = 100$ and $n_s = 1$) are significantly smaller and solved faster than those of case 2 ($T = 5$ and $n_s = 51$), although the two cases produce almost identical costs as shown in Fig. 9. Finally, case 3 in Table 3 represents a fair compromise between the length of the prediction horizon ($T = 10$) and the number of scenarios ($n_s = 3$). In Fig. 9 we see that this compromise produces the smallest costs and, moreover, has the smallest optimization problems among the ones listed in Table 3. It is solved 3 times faster than case 1 and 9 times faster than case 2. Thus we conclude that a suitably designed robust controller can be significantly more efficient than a deterministic controller.

6 Conclusion

We have seen that the concept of tree-sparse optimization problems (Steinbach 2002) extends directly from convex problems with polyhedral constraints to the fully non-

linear and nonconvex NLP case. The required structural assumptions are satisfied whenever the nonlinear constraint functions possess mild separability properties. Moreover, interior-point methods with a tree-sparse KKT solver are directly applicable to the general case by introducing straightforward structure-specific specializations to a standard inertia correction procedure. Finally, the Lagrangian is separable or partially separable in all three tree-sparse NLP variants so that the Hessian can be approximated by structure-preserving Quasi-Newton update formulas when second-order derivatives are unavailable or too expensive. The effectiveness of these techniques has been demonstrated on two robust model predictive control problems with ODE dynamics. Altogether, our problem formulation and our solution approach exploit the specific structure of tree-sparse problems in a natural way.

Although we have not demonstrated this here, it is easily seen that the distributed implementation presented in Hübner et al. (2017) extends as well to the NLP case considered in this paper: the function and derivative evaluations, possibly with Quasi-Newton updates, and the tree-sparse inertia correction parallelize similar or easier than the KKT solver. Our techniques are also directly applicable in an active set-based SQP framework such as Rose (2018): here we can again use the tree-sparse KKT solver, except that the implementation requires some overhead for the changing active set.

References

Bader G, Deuflhard P (1983) A semi-implicit midpoint rule for stiff systems of ordinary differential equations. Numer Math 41:373–398. https://doi.org/10.1007/BF01418331

Berger AJ, Mulvey JM, Rothberg E, Vanderbei RJ (1995) Solving multistage stochastic programs using tree dissection. Technical Report SOR-95-07. Princeton University, USA

Birge JR, Louveaux F (2011) Introduction to stochastic programming. Springer, Berlin. https://doi.org/10.1007/978-1-4614-0237-4

Blomvall J, Lindberg PO (2002) A Riccati-based primal interior point solver for multistage stochastic programming. Eur J Oper Rese 143(2):452–461. https://doi.org/10.1016/S0377-2217(02)00301-6

Blomvall J (2003) A multistage stochastic programming algorithm suitable for parallel computing. Parallel Comput 29(4):431–445. https://doi.org/10.1016/S0167-8191(03)00015-2

Bock HG, Plitt KJ (1985) A multiple shooting algorithm for direct solution of optimal control problems. In: Gertler J (ed) Proceedings of the 9th IFAC world congress, Budapest, Hungary, 1984, vol IX. Pergamon Press, Oxford, UK, pp 242–247. https://doi.org/10.1016/S1474-6670(17)61205-9

Byrd RH, Gilbert J-C, Nocedal J (2000) A trust region method based on interior point techniques for nonlinear programming. Math Program 89(1):149–185. https://doi.org/10.1007/PL00011391

Byrd RH, Nocedal J, Waltz RA (2006) KNITRO: an integrated package for nonlinear optimization. In: Large scale nonlinear optimization, 35–59, 2006. Springer, pp 35–59. https://doi.org/10.1007/0-387-30065-1_4

Carpenter TJ, Lustig IJ, Mulvey JM, Shanno DF (1993) Separable quadratic programming via a primal-dual interior point method and its use in a sequential procedure. ORSA J Comput 5(2):182–191. https://doi.org/10.1287/ijoc.5.2.182

Czyzyk J, Fourer R, Mehrotra S (1995) A study of the augmented system and column-splitting approaches for solving two-stage stochastic linear programs by interior-point methods. ORSA J Comput 7(4):474–490. https://doi.org/10.1287/ijoc.7.4.474

Dennis JE, Moré JJ (1977) Quasi-Newton methods, motivation and theory. SIAM Rev 19(1):46–89. https://doi.org/10.1137/1019005

Dennis JE, Schnabel RB (1996) Numerical methods for unconstrained optimization and nonlinear equations, vol 16. SIAM. https://doi.org/10.1137/1.9781611971200

Fletcher R (1995) An optimal positive definite update for sparse Hessian matrices. SIAM J Optim 5(1):192–218. https://doi.org/10.1137/0805010

Fletcher R (2013) Practical methods of optimization. Wiley, New York. https://doi.org/10.1002/9781118723203

Gill PE, Murray W, Saunders MA, Wright MH (1984) Sparse matrix methods in optimization. SIAM J Sci Stat Comput 5(3):562–589. https://doi.org/10.1137/0905041

Gill PE, Murray W, Saunders MS (2002) SNOPT: an SQP algorithm for large-scale constrained optimization. SIAM J Optim 12(4):979–1006. https://doi.org/10.1137/S1052623499350013

Gondzio J, Grothey A (2006) Direct solution of linear systems of size 109 arising in optimization with interior point methods. In: Wyrzykowski R, Dongarra J, Meyer N, Wasniewski J (eds) Parallel processing and applied mathematics, vol 3911. Lecture notes in computer science. Springer, Berlin, pp 513–525. https://doi.org/10.1007/11752578_62

Gondzio J, Grothey A (2007) Parallel interior-point solver for structured quadratic programs: application to financial planning problems. Ann Oper Res 152(1):319–339. https://doi.org/10.1007/s10479-006-0139-z

Gondzio J, Grothey A (2009) Exploiting structure in parallel implementation of interior point methods for optimization. Comput Manag Sci 6(2):135–160. https://doi.org/10.1007/s10287-008-0090-3

Griewank A, Toint PL (1982a) Partitioned variable metric updates for large structured optimization problems. Numer Math 39(1):119–137. https://doi.org/10.1007/BF01399316

Griewank A, Toint P (1982b) On the unconstrained optimization of partially separable functions. In: Powell M (ed) Nonlinear optimization 1981. Academic press, Cambridge, pp 301–312

Hübner J (2016) Distributed algorithms for nonlinear tree-sparse problems. Ph.D. Thesis. Gottfried Wilhelm Leibniz Universität Hannover

Hübner J, Steinbach MC, Schmidt M (2017) A distributed interior-point KKT solver for multistage stochastic optimization. INFORMS J Comput 29(4):612–630. https://doi.org/10.1287/ijoc.2017.0748

Jessup ER, Yang D, Zenios SA (1994) Parallel factorization of structured matrices arising in stochastic programming. SIAM J Optim 4(4):833–846. https://doi.org/10.1137/0804048

Kall P, Wallace SW (1994) Stochastic programming. Wiley-interscience series in systems and optimization. Wiley, New York

Lazar M, De La Peña DM, Heemels W, Alamo T (2008) On input-tostate stability of min-max nonlinear model predictive control. Syst Control Lett 57(1):39–48. https://doi.org/10.1016/j.sysconle.2007.06.013

Liu DC, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. Math Program 45(3):503–528. https://doi.org/10.1007/BF01589116

Lubin M, Petra CG, Anitescu M (2012) The parallel solution of dense saddle-point linear systems arising in stochastic programming. Optim Methods Softw 27(4–5):845–864. https://doi.org/10.1080/10556788.2011.602976

Lucia A (1983) An explicit Quasi-Newton update for sparse optimization calculations. Math Comput 40(161):317–322. http://www.jstor.org/stable/2007377

Lucia S, Engell S (2012) Multi-stage and two-stage robust nonlinear model predictive control. Nonlinear Model Predict Control 45(17):181–186. https://doi.org/10.3182/20120823-5-NL-3013.00015

Lucia S, Finkler T, Basak D, Engell S (2012) A new robust NMPC scheme and its application to a semi-batch reactor example. In: Proceedings of the international symposium on advanced control of chemical processes, pp 69–74. https://doi.org/10.3182/20120710-4-SG-2026.00035

Nocedal J, Wright SJ (2006) Numerical optimization, 2nd edn. Springer, Berlin. https://doi.org/10.1007/978-0-387-40065-5

Powell M, Toint PL (1981) The Shanno–Toint procedure for updating sparse symmetric matrices. IMA J Numer Anal 1(4):403–413. https://doi.org/10.1093/imanum/1.4.403

Rose D (2018) An elastic primal active-set method for structured QPs. Ph.D. Thesis. Leibniz Universität Hannover

Schmidt M (2013) A generic interior-point framework for nonsmooth and complementarity constrained nonlinear optimization. Ph.D. Thesis. Gottfried Wilhelm Leibniz Universität Hannover

Schweitzer E (1998) An interior random vector algorithm for multistage stochastic linear programs. SIAM J Optim 8(4):956–972. https://doi.org/10.1137/S105262349528456X

Steinbach MC (2001) Hierarchical sparsity in multistage convex stochastic programs. In: Uryasev SP, Pardalos PM (eds) Stochastic optimization: algorithms and applications. Kluwer Academic Publishers, Dordrecht, pp 385–410. https://doi.org/10.1007/978-1-4757-6594-6_16

Steinbach MC (2002) Tree-sparse convex programs. Math Methods Oper Res 56(3):347–376. https://doi.org/10.1007/s001860200227

Toint PL (1981) A note about sparsity exploiting quasi-Newton updates. Math Program 21(1):172–181. https://doi.org/10.1007/BF01584238

Ungar LH (1990) A bioreactor benchmark for adaptive network-based process control. In: Miller WT III, Sutton RS, Werbos PJ (eds) Neural networks for control. MIT Press, Cambridge, pp 387–402. http://dl.acm.org/citation.cfm?id=104204.104221

Vanderbei RJ, Shanno DF (1997) An interior-point algorithm for nonconvex nonlinear programming. Comput Optim Appl 13:231–252. https://doi.org/10.1023/A:1008677427361

Wächter A, Biegler LT (2006) On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math Program 106(1):25–57. https://doi.org/10.1007/s10107-004-0559-y