# Numerical Methods for Embedded Optimisation and their Implementation within the ACADO Toolkit

**Hans Joachim Ferreau[1], Boris Houska[1], Tom Kraus[2], Moritz Diehl[1]**

[1] Electrical Engineering Department (ESAT-SCD) and Optimisation in Engineering Center (OPTEC), K.U. Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium
e-mail: {joachim.ferreau,boris.houska,moritz.diehl}@esat.kuleuven.be

[2] BioSYST-MeBioS and OPTEC, K.U. Leuven, Kasteelpark Arenberg 30, 3001 Leuven, Belgium
e-mail: tom.kraus@biw.kuleuven.be

**Abstract.** *When nonlinear dynamic systems shall be controlled to perform certain tasks optimally, nonlinear optimal control problems have to be solved. Often it is even desired to solve such problems in real-time, possibly on embedded hardware. This occurs most prominently in the framework of model predictive control (MPC) of fast systems, e.g. in mechatronics or automotive engineering. We briefly review the state-of-the-art in nonlinear dynamic optimisation and point out the differences between direct approaches based on Sequential Quadratic Programming (SQP) and Interior-Point (IP) methods. We then review algorithmic ideas for embedded nonlinear optimisation, in particular the so-called real-time iteration.*

*These methods are recently implemented in the ACADO Toolkit, an open-source software environment and algorithm collection for Automatic Control and Dynamic Optimisation. The ACADO Toolkit provides a general framework for using a variety of algorithms for direct optimal control, including in particular embedded optimisation in form of model predictive control (MPC) as well as moving horizon estimation (MHE). ACADO Toolkit is implemented as self-contained C++ code whose object-oriented design allows for convenient coupling of existing optimisation packages and for extending it with user written optimisation routines.*

*We finally present simulation results using ACADO Toolkit for a challenging test problem, namely nonlinear MPC of power generating kite systems.*

**1. Introduction.** During the last decades the number of applications where control techniques based on dynamic optimisation lead to improved performance has rapidly increased. These techniques use a mathematical model in form of differential equations of the process to be controlled to predict its future behaviour and calculate optimised control actions. Often it is desired to perform this optimisation online, during the runtime of the process, to obtain a feedback controller. Within such advanced controllers, the numerical solution of optimal control problems is the main algorithmic step. Thus, efficient and reliable optimisation algorithms for performing this step, possibly on embedded hardware, are of great interest.

Searching the literature, we encounter a number of optimisation software packages for solving optimal control problems. Among them are the open-source package IPOPT [51, 52], which implements an interior point algorithm for the optimisation of large-scale differential algebraic systems discretised by collocation methods. It is written in C/C++ and Fortran, but uses modeling languages like AMPL or MATLAB for providing additional user interfaces and to allow automatic differentiation. Similar to IPOPT, the commercial MATLAB package PROPT [2] solves optimal control problems based on collocation techniques, while using existing NLP solvers such as KNITRO, CONOPT, SNOPT or CPLEX. The usage of MATLAB syntax makes PROPT quite user-friendly.

The proprietary package MUSCOD-II offers a different way for solving optimal control prob-

lems. It discretises the differential algebraic systems based on BDF or Runge-Kutta integration methods and uses Bock's direct multiple shooting [11]; sequential quadratic programming (SQP) is used for solving the resulting NLPs. Its highly efficient algorithms are implemented in C/C++ and Fortran, however, its software design makes it difficult to extend the code. Further existing packages based on shooting techniques are dsoa [22], OptCon [48], and NEWCON [45].

Each of the above packages has its particular strengths and all of them have proven successful for a specific range of applications. As they are all tailored to a certain choice of underlying numerical algorithms, it is usually problem dependent which one is most suited. Moreover, their specialised software design renders it difficult to combine algorithmic ideas from different packages or to extend them with new mathematical concepts. To overcome these drawbacks, the ACADO Toolkit has been designed to meet the following five key properties that are in the authors' opinion crucial for software packages for automatic control based on dynamic optimisation:

1. *Open-source*: The package needs to be freely available at least to academic users for allowing researchers to reproduce all results and to test own modifications. For this reason, the ACADO Toolkit is distributed under the GNU Lesser General Public Licence (LGPL), which even allows the package to be linked against proprietary software.

2. *User-friendliness*: The syntax to formulate optimal control problems should be as intuitive as possible. Given the fact that dynamic optimisation is more and more widely used in many different engineering applications, also non-experts should be able to formulate their control problems quickly. Therefore, the ACADO Toolkit makes intensive use of the object-oriented capabilities of C++, in particular operator overloading, that allows to state optimal control problems in a way that is very close to the usual mathematical syntax.

3. *Code extensibility*: The software design should allow to extend the package by linking existing algorithms and to implement new developments while avoiding code duplication. The ACADO Toolkit realises these requirements by a careful design of interfaces guaranteeing that almost all algorithmic parts can also be used in their well-documented stand-alone versions.

4. *Self-containedness*: The software must only depend on external packages if this is really inevitable; usage of external packages should be optional and the main package should provide a mode to run stand-alone. This feature is particularly crucial for applications on embedded hardware as they usually occur in model predictive controllers. Compiling external packages might often not be possible on the given hardware, or they can significantly increase the size of the executable as well as the software maintenance effort, and non-trivial software license issues may arise. To avoid these troubles, the ACADO Toolkit is written in a completely self-contained manner; the use of external packages for graphical output or sparse linear algebra operations is optional.

5. *Efficiency:* Of course, all these beforementioned properties may not seriously affect the efficiency of the implementation. Therefore, the ACADO Toolkit is designed such that basically all computational overhead only occurs during an offline initialisation step while the online computations remain fully efficient.

**1.1. Problem Formulation.** Throughout this paper we regard optimal control problems (OCPs) of the following general form that can be handled by the ACADO Toolkit:

$$\begin{aligned} \underset{x(\cdot),z(\cdot),u(\cdot),p}{\text{minimise}} \quad & \int_0^T L(\tau, x(\tau), z(\tau), u(\tau), p)\,\mathrm{d}\tau \\ & + M(x(T), z(T), p) \\ \text{subject to} \quad x(0) &= x_0 \\ \dot{x}(t) &= f(t, x(t), z(t), u(t), p) \\ 0 &= g(t, x(t), z(t), u(t), p) \\ 0 &\geq h(t, x(t), z(t), u(t), p) \\ 0 &\geq r(x(T), z(T), p) \end{aligned} \tag{1}$$

for all $t \in [0, T]$.

OCPs of this form comprise a dynamic system with differential states $x : \mathbb{R} \to \mathbb{R}^{n_x}$, a time varying control input $u : \mathbb{R} \to \mathbb{R}^{n_u}$, and time-constant parameters $p \in \mathbb{R}^{n_p}$. In some cases, the formulation of the dynamic system requires also algebraic states, which we denote by $z : \mathbb{R} \to \mathbb{R}^{n_z}$. The horizon length might either be fixed or is subject to optimisation. The algorithms implemented in ACADO `Toolkit` assume that the differential and algebraic right-hand side functions $f$ and $g$ are sufficiently smooth. Model predictive control (MPC) and moving horizon estimation (MHE) problems are special cases of this general formulation.

All direct optimal control algorithms start with transforming the continuous-time OCP of form (1) into a finite-dimensional nonlinear programming (NLP) problem. The so-called "simultaneous" approach comes in the form of direct collocation methods [50, 5] as well as in form of direct multiple shooting [11, 36]. Collocation schemes or efficient integrators are employed to arrive at the following discrete-time optimal control problem, which is a finite-dimensional NLP:

$$\begin{aligned} \underset{x,\,z,\,u}{\text{minimise}} \quad & \sum_{i=0}^{N-1} L_i(x_i, z_i, u_i) + E(x_N) \\ \text{subject to} \quad x_0 &= \bar{x}_0 \\ x_{i+1} &= f_i(x_i, z_i, u_i) \\ 0 &= g_i(x_i, z_i, u_i) \\ 0 &\geq h_i(x_i, z_i, u_i) \\ 0 &\geq r(x_N) \end{aligned} \tag{2}$$

for all $i = 0, \ldots, N-1$. In order to simplify the notation, we skipped possibly free parameters $p$ and $T$ from the formulation and will continue to do so in the presentation to follow. Note that problem formulation (2) is still fully general as these parameters could be covered by introducing auxiliary differential states to the system. The simultaneous approach addresses this nonlinear optimal control problem (2) by a Newton-type optimisation algorithm. The two steps – simulation and optimisation – are performed simultaneously.

Another approach, that emerged early in the nonlinear optimal control literature, is the so-called "sequential" approach to optimal control problems [46]. It is based on the observation that within problem (2), the initial value $x_0$ together with the system equations uniquely determine the values $x$ and $z$ if the controls $u$ are fixed. Thus, a simple system simulation yields implicit functions that satisfy the system equations for all $u$. By doing so, one arrives at a problem with strongly reduced variable space compared to the original problem, and it is thus an appealing idea to use the reduced problem within an optimisation procedure. At each optimisation iteration the two steps, system simulation and optimisation, are performed sequentially, one after the other.

The optimisation problem of the sequential approach has much less variables, but also less structure in the linear subproblems than the simultaneous approach. Even more important, the Newton-type optimisation procedure behaves quite differently for both approaches: typically, faster local convergence rates are observed for the simultaneous approach, in particular for unstable or highly nonlinear systems, because – intuitively speaking – the nonlinearity is equally distributed over the horizon.

**1.2. Outline.** The article is organised as follows: Section 2 briefly reviews Newton-type optimisation methods and how they can be applied to optimal control problems. Afterwards, Section 3 discusses several ideas to make these methods suitable for real-time applications by exploiting the fact that MPC requires the solution of a whole sequence of "neighbouring" problems. Section 4 describes how these real-time algorithms can be used within the open-source package ACADO `Toolkit`. Its main software modules and algorithmic features are sketched and ACADO `Toolkit`'s simulation environment for performing closed-loop simulations is presented. This environment is then used for setting up a nonlinear MPC simulation for a power generating kite system in Section 5; the numerical results are briefly discussed. Section 6 concludes and describes a couple of future developments of ACADO `Toolkit`.

## 2. Newton-Type Optimisation for Optimal Control

Newton's method for solving nonlinear equations of the form $Z(W) = 0$ starts with an initial guess $W^0$ and generates a series of iterates $W^k$ each solving a linearisation of the system at the previous iterate, i.e., for given $W^k$ the next iterate $W^{k+1}$ shall satisfy $Z(W^k)+\nabla Z(W^k)^T(W^{k+1}-W^k) = 0$. The hope is that the linearisations – that can be solved w.r.t. $W^{k+1}$ by standard linear algebra tools – are sufficiently good approximations of the original nonlinear system and that the iterates converge towards a solution $W^*$. If the Jacobian $\nabla Z(W^k)^T$ is computed exactly, Newton's method has locally a quadratic convergence rate, otherwise the convergence rate is slower but the iterations cheaper. This gives rise to the larger class of "Newton-type methods" and we refer to [14] for an excellent overview of them.

The discrete-time optimal control problem (2) is a specially structured instance of a generic NLP that has the form

$$\begin{aligned} &\underset{X}{\text{minimise}} \quad F(X) \\ &\text{subject to} \\ &\qquad 0 = G(X) \\ &\qquad 0 \geq H(X)\,. \end{aligned} \tag{3}$$

Under mild assumptions, any locally optimal solution $X^*$ of this problem has to satisfy the famous Karush-Kuhn-Tucker (KKT) conditions: there exist multiplier vectors $\lambda^*$ and $\mu^*$ so that the following equations hold:

$$\nabla_X \mathcal{L}(X^*, \lambda^*, \mu^*) = 0 \tag{4a}$$
$$G(X^*) = 0 \tag{4b}$$
$$H(X^*) \leq 0 \tag{4c}$$
$$\mu^* \geq 0 \tag{4d}$$
$$H_i(X^*)\,\mu_i^* = 0\,. \tag{4e}$$

for all $i = 1, \ldots, n_H$. Therein we have used the definition of the Lagrange function

$$\mathcal{L}(X, \lambda, \mu) = F(X) + G(X)^T\lambda + H(X)^T\mu\,. \tag{5}$$

All Newton-type optimisation methods try to find a point satisfying these conditions by using successive linearisations of the problem functions. Major differences exist, however, on how to treat the last conditions (4d)–(4e) that are due to the inequality constraints, and the two big families are Sequential Quadratic Programming (SQP) type methods and Interior Point (IP) methods.

**2.1. Sequential Quadratic Programming.** One variant to iteratively solve the KKT system is to linearise all nonlinear functions appearing in Eqs. (4a)–(4e). The resulting linear complementarity system can be interpreted as the KKT conditions of a quadratic program (QP)

$$\begin{aligned} &\underset{X}{\text{minimise}} \quad F^k_{\text{QP}}(X) \\ &\text{subject to} \\ &\qquad 0 = G(X^k) + \nabla G(X^k)^T(X - X^k) \\ &\qquad 0 \geq H(X^k) + \nabla H(X^k)^T(X - X^k) \end{aligned} \tag{6}$$

with objective function

$$\begin{aligned} F^k_{\text{QP}}(X) &= \nabla F(X^k)^T X \\ &+\tfrac{1}{2}(X - X^k)^T\nabla^2_X\mathcal{L}(X^k, \lambda^k, \mu^k)(X - X^k) \end{aligned}$$

In the case that the so-called Hessian matrix $\nabla^2_X\mathcal{L}(X^k, \lambda^k, \mu^k)$ is positive semi-definite, this QP is convex so that global solutions can be found reliably. This general approach to address the nonlinear optimisation problem is called Sequential Quadratic Programming (SQP). Apart from the presented "exact Hessian" SQP variant presented above, several other – and much more widely used – SQP variants exist, that make use of inexact Hessian or Jacobian matrices.

The Constrained (or Generalised) Gauss-Newton method is a particularly successful SQP variant that is based on approximations of the Hessian. It is applicable when the objective function is a sum of squares:

$$F(X) = \frac{1}{2}\|R(X)\|_2^2\,. \tag{7}$$

In this case, the Hessian can be approximated by

$$A_k = \nabla R(X^k)\nabla R(X^k)^T \tag{8}$$

and the corresponding QP objective is easily seen to be

$$F^k_{\text{QP}}(X) = \frac{1}{2}\|R(X^k) + \nabla R(X^k)^T(X - X^k)\|_2^2\,. \tag{9}$$

The constrained Gauss-Newton method has only linear convergence but often with a surprisingly fast contraction rate. The contraction rate is fast when the residual norm $\|R(X^*)\|$ is small or the problem functions $R, G, H$ have small second derivatives. It has been developed and extensively investigated by Bock and coworkers, see e.g. [10, 47] and is implemented in the packages PARFIT [10] and also as one variant within MUSCOD-II [15, 36].

**2.2. Interior Point Methods.** In contrast to SQP methods, an alternative way to address the solution of the KKT system is to replace the non-smooth KKT conditions in Eq. (4d)–(4e) by a smooth nonlinear approximation, with $\tau > 0$:

$$\begin{aligned} \nabla_X \mathcal{L}(X^*, \lambda^*, \mu^*) &= 0 && \text{(10a)} \\ G(X^*) &= 0 && \text{(10b)} \\ H_i(X^*)\, \mu_i^* &= \tau, \quad 1 \le i \le n_H\,. && \text{(10c)} \end{aligned}$$

This system is then solved with Newton's method. The obtained solution is not a solution to the original problem, but to the problem

$$\begin{aligned} &\underset{X}{\text{minimise}} \quad F(X) - \tau \textstyle\sum_{i=1}^{n_H} \log(-H_i(X)) \\ &\text{subject to} \qquad\qquad\qquad\qquad\qquad\qquad (11) \\ &\qquad 0 = G(X)\,. \end{aligned}$$

Thus, the solution is in the interior of the set described by the inequality constraints, and the closer to the true solution the smaller $\tau$ gets. The crucial feature of the family of "interior point methods" is the fact that, once a solution for a given $\tau$ is found, the parameter $\tau$ can be reduced without jeopardising convergence of Newton's method. After only a limited number of Newton iterations a quite accurate solution of the original NLP is obtained. We refer to the excellent textbooks [54, 12] for details. A widely used implementation of nonlinear interior point methods is the open-source code IPOPT [51].

**2.3. Solving the Linear Subproblems.** We mentioned earlier that discrete-time optimal control problems (2) are specially structured NLPs. In order to solve them efficiently, this structure has to be exploited when employing either of the before-mentioned solution approaches.

When an SQP-type method is used, the underlying quadratic programs (6) are linearised variants of the original nonlinear problem and thus inherit its special structure. In particular, the QP matrices occuring in the objective function and the constraints are block-(bi)diagonal, i.e. sparse, where the number of blocks corresponds to the number of intervals used to discretise the continuous-time problem. As most QP solvers, especially when they are based on an active-set method, work on dense matrices, an approach became popular that is often called "condensing" [11]. Condensing is based on a similar observation as the sequential approach, namely the fact that all system states are uniquely determined by their initial value and the control sequence. Thus, it is easily possible to use the equality constraints describing the dynamic system to eliminate all dynamic states from the QP. This leads to a usually much smaller QP, which comes at the expense of dense QP matrices. Condensing can be an efficient way to exploit the structure of the optimal control NLP when the number of states is significantly higher than that of the controls and if the number of discretisation intervals is small.

When solving the KKT systems (10) within an interior point method (similar ideas also exists for active-set methods), the sparsity can also be exploited by using a sparse linear system solver. The almost block diagonal structure of this linear system allows it to be efficiently factorised by a (discrete time) Riccati recursion. This was described for linear model predictive control in [44, 34], though the main idea is older [24, 49].

## 3. Real-Time Optimal Control Methods

For exploiting the fact that (nonlinear) MPC requires the solution of a whole sequence of "neighbouring" NLPs and not just a number of stand-alone problems, we have first the possibility to initialise subsequent problems efficiently based on previous information. In this section we introduce several concepts for such initialisations, in particular the important concept of NLP sensitivities. On the other hand, we will give an overview of specially tailored online algorithms for approx-

imately solving each NLP, that deliver on purpose inaccurate solutions and postpone calculations from one problem to the next.

**3.1. Online Initialisation.** A first and obvious way to transfer solution information from one solved MPC problem to the initialisation of the next one is based on the dynamic programming principle, also-called the principle of optimality of subarcs: Let us assume we have computed an optimal solution $(x_0^*, z_0^*, u_0^*, x_1^*, z_1^*, u_1^*, \ldots, x_N^*)$ of the MPC problem (2) starting with initial value $\bar{x}_0$. If we regard a shortened MPC problem without the first interval, which starts with the initial value $\bar{x}_1$ chosen to be $x_1^*$, then for this shortened problem the vector $(x_1^*, z_1^*, u_1^*, \ldots, x_N^*)$ is the optimal solution. Based on the expectation that the measured or observed true initial value for the shortened MPC problem differs not much from $x_1^*$ – i.e. we believe our prediction model and expect no disturbances – this "shrinking horizon" initialisation is canonical.

However, in the case of moving (finite) horizon problems, the horizon is not only shortened by removing the first interval, but also prolonged at the end by appending a new terminal interval – i.e. the horizon is moved forward in time. In the moving horizon case, the principle of optimality is thus not strictly applicable, and we have to think about how to initialise the appended new variables $z_N, u_N, x_{N+1}$. Often, they are obtained by setting $u_N := u_{N-1}$ or setting $u_N$ as the steady-state control. The states $z_N$ and $x_{N+1}$ are then obtained by forward simulation. We call this transformation of the variables from one problem to the next "shift initialisation". Though the shifted solution is not optimal for the new undisturbed problem, in the case of long horizon lengths $N$ we can expect the shifted solution to be a good initial guess for the new solution. Moreover, for most MPC schemes with stability guarantee (for an overview see e.g. [41]) there exists a canonical choice of $u_N$ that implies feasibility (but not optimality) of the shifted solution for the new, undisturbed problem. The shift initialisation is very often used e.g. in [38, 7, 42, 21].

**3.2. NLP Sensitivities.** In the shift initialisation discussed above we did assume that the new initial value corresponds to the model prediction. This is of course never the case, because exactly the fact that the initial state is subject to disturbances motivates the use of MPC. By far the most important change from one optimisation problem to the next one are thus the unpredictable changes in the initial value. The concept of parametric NLP sensitivities can be used to construct a guess for the new initial value.

To illustrate the idea, let us first regard the parametric root finding problem $R(\bar{x}_0, W) = 0$ that results from the necessary optimality conditions of an IP method, i.e. the system (10) in variables $W = (X, \lambda, \mu)$. In the MPC context, this system depends on the uncertain initial value $\bar{x}_0$. We denote the solution manifold by $W^*(\bar{x}_0)$. When we know the solution $W = W^*(\bar{x}_0)$ for a previous initial value $\bar{x}_0$ and want to compute the solution for a new initial value $\bar{x}_0'$, then a good solution predictor for $W^*(\bar{x}_0')$ is provided by $W' = W + \frac{dW^*}{d\bar{x}_0}(\bar{x}_0)(\bar{x}_0' - \bar{x}_0)$ where $\frac{dW^*}{d\bar{x}_0}(\bar{x}_0)$ is given by the implicit function theorem. An important practical observation is that an approximate tangential predictor can also be obtained when it is computed at a point $W$ that does not exactly lie on the solution manifold. This fact leads to a combination of a predictor and corrector step in one linear system [17, 55]. Unfortunately, the interior point solution manifold is strongly nonlinear at points where the active set changes, and the tangential predictor is not a good approximation when we linearise at such points.

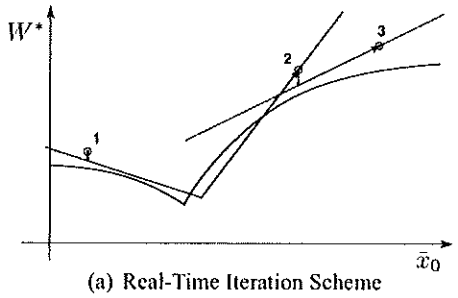
This is because the true NLP solution is not determined by a smooth root finding problem (10). Instead, it is a well-known fact from parametric optimisation that the solution manifold has smooth parts when the active set does not change (and bifurcations are excluded), but that nondifferentiable points occur whenever the active set changes [29]. In order to "jump" over these nonsmooth points to deliver better predictors than the IP predictors discussed before, we have to regard directional derivatives of the solution manifold. These "generalised tangential predictors" can be computed by suitable quadratic programs [29,

Thm 3.3.4]. The theoretical results can be made a practical algorithm by the following procedure proposed in [15]: first, we have to make sure that the parameter $\bar{x}_0$ enters the NLP linearly, which is automatically the case for simultaneous optimal control formulations. Second, we address the problem with an exact Hessian SQP method. Third, we just take our current solution guess $W$ for a problem $\bar{x}_0$, and then solve the QP subproblem for the new parameter value $\bar{x}_0'$, but initialised at $W$. It can be shown [15, Thm. 3.6] that this "initial value embedding" procedure delivers exactly the generalised tangential predictor when started at a solution $W = W^*(\bar{x}_0)$. It is important to remark that the predictor becomes approximately tangential when we do not start on the solution manifold or no exact Hessian matrix is used.

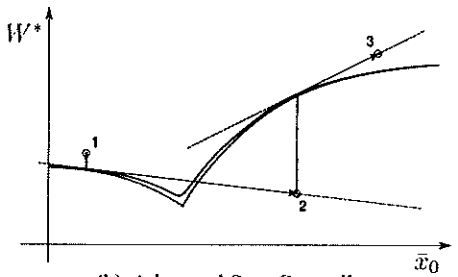
**3.3. Brief Overview of Online Optimisation for MPC.** We will now review a few of the approaches suggested in the literature in a personal and incomplete selection.

The *Newton-Type Controller* of Li and Biegler [37] was probably one of the first true online algorithms for MPC. It is based on a sequential optimal control formulation, thus it iterated in the space of controls only. It uses an SQP-type procedure with Gauss-Newton Hessian and line search, and in each sampling time, only one SQP iteration is performed. The transition from one problem to the next uses a shift of the controls. The result of each SQP iterate is used to give an approximate feedback to the plant. As a sequential scheme without tangential predictor, it seems to have had sometimes problems with nonlinear convergence, though closed-loop stability was proven for open-loop stable processes [38], and in principle, the theoretical nonlinear MPC stability analysis from [19] is applicable.

The *Real-Time Iteration Scheme* presented in [15, 17] also performs one SQP type iteration with Gauss-Newton Hessian per sampling time. However, it employs a simultaneous NLP parameterisation, Bock's direct multiple shooting method, with full derivatives and condensing. Moreover, it uses the generalised tangential predictor of the "initial value embedding" discussed in previous subsection to correct for the mismatch between the expected state $\bar{x}_0$ and the actual state $\bar{x}_0'$. For doing so, an inequality constrained QP is solved. The computations in each iteration are divided into a long "preparation phase", in which the system linearisation, elimination of algebraic variables and condensing are performed, and a much shorter "feedback phase" just one condensed QP is solved. Depending on the application, the feedback phase can be several orders of magnitude shorter than the feedback phase. The iterates of the scheme are visualised in Fig. 1(a).




(a) Real-Time Iteration Scheme




(b) Advanced Step Controller

**Fig. 1.** Subsequent solution approximations across an active-set change.

Note that only one system linearisation and one QP solution are performed in each sampling time, and that the QP corresponds to a linear MPC feedback along a time varying trajectory. In contrast to IP formulations, the real-time iteration scheme gives priority to active set changes and works well when the active set changes faster than the linearised system matrices. In the limiting case of a linear system model it gives the same feedback as linear MPC. Error bounds and closed loop stability of the scheme have been established for shrinking horizon problems in [16]

and for MPC with shifted and non-shifted initialisations in [19, 18].

The *Advanced Step Controller* by Zavala and Biegler [55] iterates a complete Newton-type IP procedure to convergence (with $\tau \to 0$) in each sampling time. In this respect, it is just like offline optimal control. However, two features qualify it as an online algorithm: first, it takes computational delay into account by solving an "advanced" problem with the expected state $\bar{x}_0$ as initial value – similar as in the real-time iterations with shift – and second, it applies the obtained solution not directly, but computes first the tangential predictor which is correcting for the differences between expected state $\bar{x}_0$ and the actual state $\bar{x}_0'$. Note that in contrast to the other online algorithms, several Newton iterations are performed during the "preparation phase". The tangential predictor is computed in the "feedback phase" by only one linear system solve based on the last Newton iteration's matrix factorisation. The IP predictor cannot "jump" over active set changes as easily as the SQP based predictor of the real-time iteration, see Fig. 1(b). Roughly speaking, the advanced step controller gives lower priority to sudden active set changes than to system nonlinearity. As the advanced step controller solves each expected problem exactly, classical MPC stability theory [41] can relatively easily be extended to this scheme [55].

## 4. ACADO Toolkit for Automatic Control and Dynamic Optimisation

After briefly reviewing a couple of dedicated numerical algorithms for real-time optimal control, we will now describe how they can be used within the open-source package ACADO `Toolkit`.

**4.1. Scope of the Software.** The open-source package ACADO `Toolkit 1.0` can deal with the following three important problem classes: First, offline dynamic optimisation problems, where the aim is to find an open-loop control which minimises a given objective functional. The second class are parameter and state estimation problems, where parameters or unknown control inputs should be identified by measuring an output of a given (nonlinear) dynamic system. The third class are combined online estimation and model predictive control problems, where parameterised dynamic optimisation problems have to be solved repeatedly to obtain a dynamic feedback control law.

The basic class of problems which can be solved with ACADO `Toolkit` are standard optimal control problems (OCPs) as stated in Section 1.1. The algorithms that are currently implemented in ACADO `Toolkit` assume that the right-hand side function $f$ is smooth or at least sufficiently often differentiable depending on which specific discretisation method is used. The same requirements should be satisfied for the algebraic right-hand side $g$. Moreover, we assume that the function $\frac{\partial g}{\partial z}$ is always regular, i.e. the index of the DAE should be one. The remaining functions, namely the Lagrange term $L$, the Mayer term $M$, the boundary constraint function $r$, as well the path constraint function $s$ are assumed to be at least twice continously differentiable in all their arguments.

An important subclass of optimal control problems, which requires special attention, are state and parameter estimation problems. This subclass can theoretically also be transformed into a problem of the form (1). However, as mentioned in Section 2.1, parameter estimation problems with least-squares objective terms can be treated with a specialised algorithm known under the name generalised Gauss-Newton method. The general parameter estimation problem formulation is as follows:

$$\begin{aligned} \underset{x(\cdot),z(\cdot),u(\cdot),p}{\text{minimise}} \quad & \sum_{i=1}^{N} \| h(t_i, x(t_i), z(t_i), u(t_i), p) - \eta_i \|^2_{S_i} \\ \text{subject to} \quad & \\ \dot{x}(t) &= f(t, x(t), z(t), u(t), p) \\ 0 &= g(t, x(t), z(t), u(t), p) \\ 0 &= r(x(0), x(T), p) \\ 0 &\geq s(t, x(t), z(t), u(t), p) \end{aligned} \tag{12}$$

for all $t \in [0, T]$. Here, $h$ is called a measurement function while $\eta_1, \ldots, \eta_N$ are the measurements taken at the time points $t_1, \ldots, t_N \in [0, T]$. Note that the least-squares term is in this formulation weighted with positive semi-definite weight-

ing matrices $S_1, \ldots, S_N$, which are typically the inverses of the variance covariance matrices associated with the measurement errors.

Finally, model based feedback control constitutes the third main problem class that can be tackled with ACADO Toolkit. It comprises two kinds of online dynamic optimisation problems: the Model Predictive Control (MPC) problem of finding optimal control actions to be fed back to the controlled process, and the Moving Horizon Estimation (MHE) problem of estimating the current process states using measurements of its outputs. The MPC problem is a special case of an (OCP) and is assumed to be stated in the following form:

$$\begin{aligned} \min_{\substack{x(\cdot),\, z(\cdot) \\ u(\cdot),\, p}} \quad & \int_{t_0}^{t_0+T} \|h(t, x(t), z(t), u(t), p) - \eta(t)\|_Q^2 \mathrm{d}\tau \\ & + \|m(T, x(T), p) - \eta(T)\|_P^2 \\ \text{s. t.} \quad & \\ x(t_0) &= x_0 \\ \dot{x}(t) &= f(t, x(t), z(t), u(t), p) \\ 0 &= g(t, x(t), z(t), u(t), p) \\ 0 &= r(x(T), z(T), p) \\ 0 &\geq s(t, x(t), z(t), u(t), p) \end{aligned} \tag{13}$$

for all $t \in [t_0, t_0 + T]$. In contrast to OCPs, MPC problems are assumed to be formulated on a fixed horizon $T$ and employing a tracking objective function that quadratically penalises deviations of the process outputs from a given reference trajectory $\eta$. Moreover, formulation (MPC) requires a fixed initial value $x_0$ for the differential states to be given.

In case not all differential states of the process can be measured directly, an estimate has to be obtained using an online state estimator. This is usually done by one of the many Kalman filter variants or by solving an MHE problem. The MHE problem has basically the same form as problem formulation (12). Both, the MPC and the MHE problem are solved repeatedly, during the runtime of the process, to yield a model and optimisation based feedback controller.

**4.2. Symbolic Expressions.** One fundamental requirement on an optimal control package is that functions such as objectives, right-hand sides of differential equations or constraint functions can be provided by the user in a convenient manner. Often this is achieved by letting the user link plain C functions to the package. However, ACADO Toolkit implements more powerful features: It uses symbolic expressions as a base class to build up complex model equations by making extensive use of the C++ class concept together with operator overloading. By doing so, all structural information about the functions is kept.

Listing 1: Dimension and convexity detection for symbolic functions

```
int main( ){

  DifferentialState x;
  Function          f;

  f << exp(x) + 1.0;
  f << x*x;

  printf("The dimension of f is");
  printf("%d.   \n", f.getDim() );

  if( f.isConvex() == TRUE )
      printf("Function f is convex.\n");

  return 0;
}
```

In order to illustrate this concept, we consider the ACADO tutorial code Listing 1. Running this simple piece of code shows that the dimension of the defined function f is two and that it depends on one differential state. Moreover, the convexity of the components of f is recognised. Due to operator overloading the syntax can be used as if we would write standard C/C++ code.

The use of symbolic expressions offers a couple of interesting features:

- *Automatic differentiation:* The symbolic notation of functions enables us to provide not only numeric but also automatic and symbolic differentiation. The automatic differentiation [8, 27, 28] is implemented in its forward as well as in the adjoint mode for first and (mixed) second order derivatives. Moreover, all expressions can symbolically be differentiated returning again an expression, like AD with source code transformation. This func-

tionality can be used recursively leading to arbitrary orders of symbolic differentiation.

- *Convexity detection:* As we have already illustrated in the example code, functions can be tested for convexity/concavity. The corresponding algorithmic routines are based on disciplined convex programming [25]. Note that the syntax for the routines is (almost) the same as in the MATLAB package CVX [26]. However the ACADO code is C++ based.
- *Code optimisation:* In the context of optimal control algorithms, right-hand side functions are typically evaluated many times. Thus, it is efficient to pre-optimise functions internally during the initialisation phase, which is done automatically within ACADO Toolkit. By doing so, sparsity patterns or specific structures of the constraints can be exploited to speed-up computation.
- *C Code Generation:* Each model function written in the ACADO notation can later also be exported in form of (optimised) standard C code.

**4.3. Software Modules and Algorithmic Features.** ACADO Toolkit is designed as a platform for new algorithmic extensions, but already implements a number of algorithmic components to solve optimal control problems in the form given in Section 4.1. These components are implemented within different abstract software modules to facilitate future extensions. We briefly sketch them one by one.

First, for the optimisation of dynamic systems based on single or multiple shooting methods [11], it is necessary to simulate ODE or DAE systems. Moreover, sensitivities of the state trajectory with respect to initial values, controls, etc. must be computed efficiently. For this aim, ACADO Toolkit comes along with state-of-the-art integration routines such as several Runge-Kutta methods as well as a BDF (backward differentiation formula) method which is used for stiff differential or differential algebraic equations. A particular feature of the ACADO BDF integrator, which is based on the algorithmic ideas in [3, 4], is that it can also deal with fully implicit differential algebraic equations of index 1 given in the form

$$\forall t \in [0,T]: \; F(\dot{y}(t), y(t), u(t), p, T) = 0\,, \quad (14)$$

where differential and algebraic states are merged into one state vector $y$.

Sensitivities can either be computed using internal numerical differentiation [10, 4] or (internal) automatic differentiation. However, for automatic differentiation the right hand side functions must be provided in the ACADO syntax. In order to provide consistent and self-contained C++ code, the integration routines have been implemented in cooperation with the class Function, which detects for example the sparsity patterns of the right-hand side functions. However, note that in the current release, ACADO Toolkit does only provide dense linear algebra routines. It also provides interfaces for external sparse linear algebra solvers, which can be linked and then be used e.g. within the BDF integrator. Tailored sparse linear algebra solvers will be made available in future versions.

Second, once an integrator for dynamic systems is avaliable, the original continuous optimal control problem can be discretised. The most simple strategy is to regard the simulation of the system as a function evaluation depending on the initial values, parameters, controls, etc. leading to a single shooting discretisation. ACADO Toolkit also implements multiple shooting methods, which out-perform single shooting methods in many cases [11, 35]. As an alternative to shooting techniques, collocation methods have attracted a lot of attention during the last decades [6, 52]. Here, the dynamic system is discretised at the level of the NLP, leading to quite large and sparse nonlinear programs. In ACADO Toolkit, collocation methods are currently under development and will be released soon.

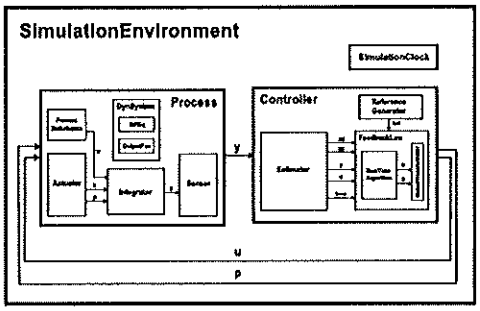
As described in Section 1.1, discretisation of the dynamic system transforms the continuous optimal control problem into a specially structured nonlinear program of form (3). Thus, specially tailored nonlinear optimisation algorithms form the third main software mod-

ule within ACADO Toolkit. Currently, ACADO Toolkit provides several SQP-type methods that can e.g. be based on BFGS Hessian approximations, as described in [43], or on Gauss-Newton methods [9]. In addition, line search globalisation routines [43, 30] as well as auto-initialisation techniques are implemented to make the optimisation routines more reliable. If multiple shooting is used for the discretisation, ACADO Toolkit exploits the structure via condensing techniques as described in Section 2.3. By default, the dense QP are solved using the open-source C++ software package qpOASES [1, 23].

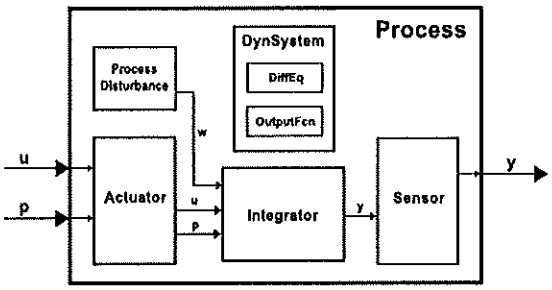
Together with the implementation of collocation methods, interior point methods, e.g. as described in [6], are currently under development. This comprises both the development of own code to keep ACADO Toolkit self-contained as well as linkage of existing open-source packages.

**4.4. Functionality for Model-Based Feedback Control.** Besides the before-mentioned features to solve offline optimal control problems, ACADO Toolkit provides a complete simulation environment for closed-loop simulations as illustrated in Figure 4.4. Its main components are the Process class for setting up realistic simulations of the process to be controlled and the Controller class for implementing the closed-loop controller. This controller can later be used stand-alone, e.g. on embedded hardware, for real-world feedback control applications.

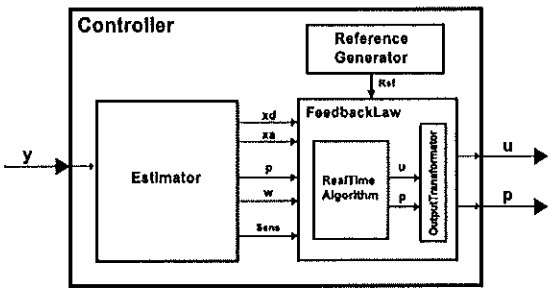
The Process class in depicted in Figure 2(b). Its most prominent members are a dynamic system, comprising a differential equation as well as an optional output function modelling the process, and an integrator capable of simulating these model equations. The simulation uses (optimised) control and parameter inputs from the controller, which might be subject to noise or delays that can be introduced via an (hypothetical) actuator. In addition, so-called process disturbances can be specified by the user for setting up arbitrary disturbance scenarios for the simulation. For example, in the powerkite example presented in Section 5 the process disturbance introduces wind disturbances on the flying kite. Finally, the outputs obtained by integrating the model equations can again be subject to noise or delays introduced via a (hypothetical) sensor. It is important to note that the model used for simulating the process does not need to be the same as specified within the optimal control formulations within the controller.




(a) Simulation Environment




(b) Process




(c) Controller

**Fig. 2.** The main components of ACADO Toolkit's simulation environment.

The design of the Controller class is illustrated in Figure 2(c) and consists of three major blocks: first, an online state/parameter estimator uses the outputs of the process to obtain estimates for the differential states as well as, if present, the parameters and disturbances together with sensitivity information. This estimator might either

be a moving horizon estimator or one of several implemented variants of the Kalman filter. Second, a so-called reference generator provides reference trajectories to the feedback law. These references can either be statically given by the user according to a desired simulation scenario or can be calculated dynamically based on information from the estimator. Finally, both the state/parameter estimates as well as the reference trajectories are used by the feedback law class to compute optimised controls and parameters. The feedback law can be something as simple as a linear state feedback but will usually obtain the controls from one of the real-time algorithms described in Section 3.3. Currently, only the real-time iteration scheme can be used but other real-time methods are under development.

While the `Process` is conceptionally thought to produce a continuous output stream, output of the `Controller` comes in sampled form. The sampling time is determined dynamically based on the sampling times given for estimator, reference generator and feedback law. Their sampling times are usually pre-defined by the user, however, this concept is currently extended to the case where these block dynamically vary their sampling times based on the computation time required to perform their respective tasks. Communication between `Process` and `Controller` is orchestrated by the `SimulationEnviroment`. It also features the simulation of computational delay, i.e. it can delay the control input to the `Process` by the amount of time the `Controller` took to determine the feedback control. This feature seems to be crucial for realistic closed-loop simulations of fast processes where the sampling time is not negligible compared to the settling time of the controlled process.

**5. Numerical Example: Nonlinear MPC of Power Generating Kite Systems** As a numerical example we present an optimization study for kites that produce wind energy by periodically pulling a generator on the ground while flying fast in a crosswind direction. This application – initially motivated by Loyd[39] in the 1980s – is described in detail in [32], cf. also [31, 13, 33, 53, 20].

The main concept for energy production with kites is that these kites periodically pull their cables to drive a generator on the ground, as it is shown in Figure 3. In this paper we consider the case that this generator is fixed on a tower of 60m height. Of course, every kite has to be pulled back at some point in time to achieve a periodic power generating cycle. In this paper we use a lift or drag control to reduce the kite's pulling force as described in [32].

**5.1. Model and Problem Description** The model is a system of 9 nonlinear differential equations and the controls are the change of the kite's roll angle, the change of its lift coefficient and the second time-derivative of the cable length. In our numerical example we attempt to follow a precalculated reference trajectory with the kite. This reference trajectory is a repetition of an optimal periodic loop of 18s being the solution of an offline optimal control problem achieving a maximal average power at the generator for one cycle. The objective function consists of 9 least-squares terms. The position of the kite, given in polar coordinates, and its velocity are the 6 states, which should follow the time-dependent reference trajectory. Furthermore, the three controls are penalized to provide for a decent control action. A "soft" zero terminal constraint is introduced in form of an extra least-squares term at the end node of the control/prediction horizon.

An important point in the considerations in [32] is that the wind is increasing with the altitude above the ground level and kites can use this powerful wind at high altitudes in contrast to conventional

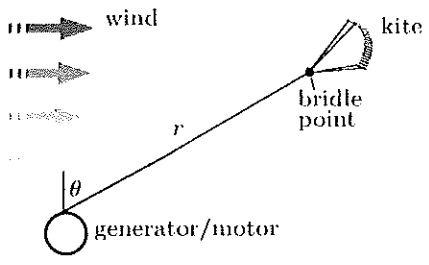



**Fig. 3.** Energy production with a single kite

windmills. Using the wind shear model, as also proposed in [40], the wind velocity $w$ is depending on the altitude $h$ over the ground:

$$w(h) = \frac{\ln\left(\frac{h}{h_r}\right)}{\ln\left(\frac{h_0}{h_r}\right)}(v_0 + w_v), \qquad (15)$$

where $v_0$ is the wind velocity at a suitable altitude $h_0$ and $h_r$ the roughness length. We also simulate a disturbance $w_v$ which is unknown to the optimization routine and makes the online control problem more challenging. Practically speaking, this disturbance can be interpreted as a sudden unforseen gust of wind or a diminishment of wind. In our simulations we consider the wind velocity at 100m to double from $10\frac{m}{s}$ to $20\frac{m}{s}$ for 6 s.

**5.2. Setting-Up the Simulation with ACADO Toolkit** To solve the corresponding MPC problem numerically, we use the ACADO `Toolkit` and combine the direct multiple shooting method and a generalized Gauss-Newton-algorithm.

The corresponding setup in ACADO `Toolkit` is shown in listing 2. After the definition of the model equations, which cannot be shown here in all detail, the online optimal control problem `OCP` is set up. The control/prediction horizon length is set to 10 s while the sampling time is fixed at 1 s. Note that the number of piecewise constant control intervals is 10, which is also set in the constructor of the `OCP`. The setup of the optimal control problem realizes the formulation (13) comprising the model response function $h$ and the end term $m$.

For the definition of the process a wind disturbance is read from a file to be simulated by the `SimulationEnvironment`. For the real-time algorithm we have set two options: The integrator tolerance is defined to be $10^{-6}$, while a KKT tolerance of $10^{-5}$ should be obtained during the online optimization.

**5.3. Results** In Figure 4 the numerical results are visualized. The kite is following its steady-state trajectory at first when no disturbance is present and $w_v = 0$. Then the disturbance $w_v = 10$ is active for 6 s and the kite is blown off its reference trajectory. However, when the disturbance

Listing 2: Setting up (simulated) process and feedback controller for powerkite NMPC simulation.

```
// SETUP OF THE MODEL
// ------------------
// ... definition of the kite model
// ... definition of model response h, m
// ... definition of weighting matrices P,Q


// SETTING UP OF THE OPTIMAL CONTROL PROBLEM:
// -----------------------------------------
const double tStart =  0.0;
const double tEnd   = 10.0;

OCP ocp( tStart, tEnd, 10 );
ocp.minimizeLSQ       ( Q, h, eta );
ocp.minimizeLSQEndTerm( P, m, eta );

ocp.subjectTo( kiteModel );

// ... definition of control and
//     state constraints


// SETTING UP THE (SIMULATED) PROCESS
// ----------------------------------
Process process( kiteModel );
process.setProcessDisturbance(
            "wind_disturbance.txt");

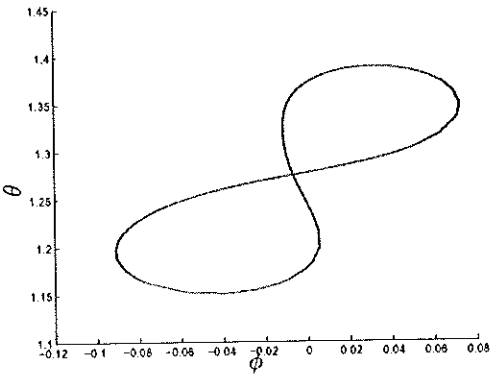

// SETTING UP THE NMPC CONTROLLER
// ------------------------------
RealTimeAlgorithm algorithm( ocp );
algorithm.set("IntegratorTolerance", 1e-6 );
algorithm.set("KKTtolerance"       , 1e-5 );

double samplingTime = 1.0;
PeriodicReferenceTrajectory reference;
reference=readFromFile("periodic_data.txt");
Controller controller( algorithm,
                       reference,
                       samplingTime );

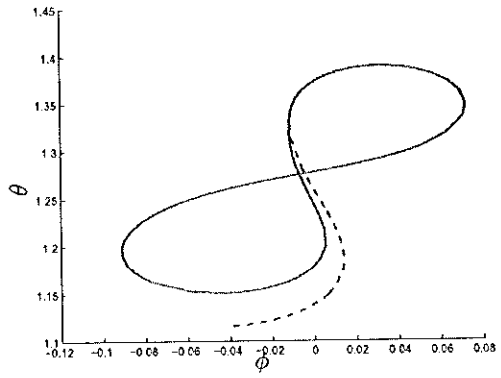

// RUN SIMULATION
// --------------
Vector x0("initial_value.txt");
double simStartTime =  0.0;
double simEndTime   = 90.0;

SimulationEnvironment sim( process,
                           controller );
sim.init( x0 );
sim.run( simStartTime,simEndTime );
```

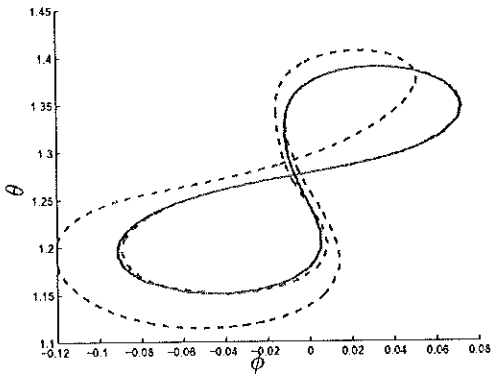
vanished the controller is able to bring the kite back again after approximatly one cycle duration. In general, NMPC can handle quite large disturbances $w_v$. No matter at which point of the loop the disturbance was applied, the kite always found its way back to its reference loop for disturbances




(a) steady-state trajectory




(b) trajectory during 6s disturbance




(c) trajectory stabilized

**Fig. 4.** angular position of kite (polar coordinates)

of this magnitude and length. This is demonstrating the robustness of the chosen approach. However, there is no theoretical guarantee that the optimization routine can always remain safely in its region of convergence, when the disturbances are too large.

Note that after the simulation of 90 s of flight time, which correspond to 5 cycles, a total energy of 234.7 MJ was gained.

## 6. Conclusions and Future Work

We reviewed a couple of important algorithmic ideas for embedded nonlinear optimisation and showed how they can be used within the `ACADO Toolkit`, a software environment and algorithm collection for automatic control and dynamic optimization. It is an open-source (GNU Lesser Public License) software package written in `C++` that is completely self-contained. However, its software design allows to easily extend the `ACADO Toolkit` with existing numerical optimization packages and its user-friendly syntax makes it very convenient to set up customized optimization problems. We sketched its key algorithmic features, in particular its simulation environment for setting up closed-loop MPC/MHE simulations. We demonstrated the usage of this environment for performing a nonlinear MPC simulation for a power generating kite system and briefly discussed the numerical results.

`ACADO Toolkit` is currently released in version `1.0` that implements the functionality described in this paper. However, several algorithmic extensions like collocation techniques, interior point methods, sequential convex programming methods, and multi-objective optimization tools are currently under development and will become part of future releases. Finally, we would like to invite external developers to realise their own algorithmic ideas within the open framework of `ACADO Toolkit`.

**Acknowledgements.** Research is supported by Research Council KUL: CoE EF/05/006 Optimization in Engineering (OPTEC), IOF-SCORES4CHEM, GOA/10/009 (MaNet), GOA/10/11, several PhD/postdoc and fellow grants; Flemish Government:

FWO: PhD/postdoc grants, projects G.0452.04, G.0499.04, G.0211.05, G.0226.06, G.0321.06, G.0302.07, G.0320.08, G.0558.08, G.0557.08, G.0588.09,G.0377.09, research communities (ICCoS, ANMMM, MLDM); IWT: PhD Grants, Belgian Federal Science Policy Office: IUAP P6/04; EU: ERNSI; FP7-HDMPC, FP7-EMBOCON, Contract Research: AMINAL. Other: Helmholtz-vICERP, COMET-ACCM. The first author holds a PhD fellowship of the Research Foundation – Flanders (FWO).

## References.

[1] qpOASES Homepage. http://www.qpOASES.org, 2007–2009.

[2] PROPT: Matlab Optimal Control Software (ODE,DAE). http://tomdyn.com, 2009.

[3] U.M. Ascher and L.R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential–Algebraic Equations*. SIAM, Philadelphia, 1998.

[4] I. Bauer. *Numerische Verfahren zur Lösung von Anfangswertaufgaben und zur Generierung von ersten und zweiten Ableitungen mit Anwendungen bei Optimierungsaufgaben in Chemie und Verfahrenstechnik*. PhD thesis, Universität Heidelberg, 1999.

[5] L.T. Biegler. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers and Chemical Engineering*, 8:243–248, 1984.

[6] L.T. Biegler. An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing*, 46:1043–1053, 2007.

[7] L.T. Biegler and J.B Rawlings. Optimization approaches to nonlinear model predictive control. In W.H. Ray and Y. Arkun, editors, *Proc. 4th International Conference on Chemical Process Control - CPC IV*, pages 543–571. AIChE, CACHE, 1991.

[8] C.H. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland. ADIFOR Generating derivative codes from Fortran programs. *Scientific Programming*, 1:11–29, 1992.

[9] H.G. Bock. Recent advances in parameter identification techniques for ODE. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations*. Birkhäuser, Boston, 1983.

[10] H.G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, volume 183 of *Bonner Mathematische Schriften*. Universität Bonn, Bonn, 1987.

[11] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, pages 243–247. Pergamon Press, 1984.

[12] S. Boyd and L. Vandenberghe. *Convex Optimization*. University Press, Cambridge, 2004.

[13] M. Canale, L. Fagiano, and M. Milanese. Power Kites for Wind Energy Generation - Fast Predictive Control of Tethered Airfoils. *IEEE Control Systems Magazine*, pages 25–38, 2007.

[14] P. Deuflhard. *Newton Methods for Nonlinear Problems*. Springer, New York, 2004.

[15] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*, volume 920 of *Fortschr.-Ber. VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik*. VDI Verlag, Düsseldorf, 2002. Download also at: http://www.ub.uni-heidelberg.de/archiv/1659/.

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

[17] M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and Nonlinear Model Predictive Control of Processes governed by differential-algebraic equations. *J. Proc. Contr.*, 12(4):577–585, 2002.

[18] M. Diehl, R. Findeisen, and F. Allgöwer. A Stabilizing Real-time Implementation of Nonlinear Model Predictive Control. In L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors, *Real-Time and Online PDE-Constrained Optimization*, pages 23–52. SIAM, 2007.

[19] M. Diehl, R. Findeisen, F. Allgöwer, H.G. Bock, and J.P. Schlöder. Nominal Stability of the Real-Time Iteration Scheme for Nonlinear Model Predictive Control. *IEE Proc.-Control Theory Appl.*, 152(3):296–308, 2005.

[20] M. Diehl and B. Houska. Windenergienutzung mit schnell fliegenden flugdrachen: eine herausforderung für die optimierung und regelung - wind power via fast flying kites: a challenge for optimization and control. *at-automatisierungstechnik*, 57(10):525–533, 2009.

[21] M. Diehl, L. Magni, and G. De Nicolao. Online NMPC of unstable periodic systems using approximate infinite horizon closed loop costing. *IFAC Annual Reviews in Control*, 28:37–45, 2004.

[22] B.C. Fabien. dsoa: The implementation of a dynamic system optimization algorithm. *Optimal Control Applications and Methods*, DOI: 10.1002/oca.898, 2009.

[23] H.J. Ferreau, H.G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.

[24] T. Glad and H. Johnson. A method for state and control constrained linear-quadratic control problems. In *Proceedings of the 9th IFAC World Congress, Budapest, Hungary*, pages 1583–1587, 1984.

[25] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs, recent advances in learning and control. *Lecture Notes in Control and Information Sciences, Springer*, pages 95–110, 2008.

[26] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming (web page and software). http://stanford.edu/ boyd/cvx, June 2009.

[27] A. Griewank. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, 2000.

[28] A. Griewank, D. Juedes, H. Mitev, J. Utke, O. Vogel, and A. Walther. ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++. Technical report, Technical University of Dresden, Institute of Scientific Computing and Institute of Geometry, 1999. Updated version of the paper published in *ACM Trans. Math. Software* 22, 1996, 131–167.

[29] J. Guddat, F. Guerra Vasquez, and H.T. Jongen. *Parametric Optimization: Singularities, Pathfollowing and Jumps*. Teubner, Stuttgart, 1990.

[30] S. P. Han. A Globally Convergent Method for Nonlinear Programming. *JOTA*, 22:297–310, 1977.

[31] B. Houska and M. Diehl. Optimal Control of Towing Kites. In *45th IEEE Conference on Control and Decision, San Diego*, pages 2693–2697, 2006. (CD-ROM).

[32] B. Houska and M. Diehl. Optimal Control for Power Generating Kites. In *Proc. 9th European Control Conference*, page 3560d'ż"3567, Kos, Greece,, 2007. (CD-ROM).

[33] A. Ilzhoefer, B. Houska, and M. Diehl. Nonlinear MPC of kites under varying wind conditions for a new class of large scale wind power generators. *International Journal of Robust and Nonlinear Control*, 17(17):1590–1599, 2007.

[34] J.B. Jorgensen, J.B. Rawlings, and S.B. Jorgensen. Numerical methods for large-scale moving horizon estimation and control. In *Proceedings of Int. Symposium on Dynamics and Control Process Systems (DYCOPS)*, 2004.

[35] D.B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*, volume 613 of *Fortschritt-Berichte VDI Reihe 3, Verfahrenstechnik*. VDI Verlag, Düsseldorf, 1999.

[36] D.B. Leineweber, I. Bauer, A.A.S. Schäfer, H.G. Bock, and J.P. Schlöder. An Efficient Multiple Shooting Based Reduced SQP Strategy for Large-Scale Dynamic Process Optimization (Parts I and II). *Computers and Chemical Engineering*, 27:157–174, 2003.

[37] W.C. Li and L.T. Biegler. Multistep, Newton-Type Control Strategies for Constrained Nonlinear Processes. *Chem. Eng. Res. Des.*, 67:562–577, 1989.

[38] W.C. Li and L.T. Biegler. Newton-Type Controllers for Constrained Nonlinear Processes with Uncertainty. *Industrial and Engineering Chemistry Research*, 29:1647–1657, 1990.

[39] M.L. Loyd. Crosswind Kite Power. *Journal of Energy*, 4(3):106–111, July 1980.

[40] MathWorks. A Wind Shear Model. http://www.mathworks.com/ acces/ hepdesk /help /toolbox /aeroblks /windshearmodel.html, 2006.

[41] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: stability and optimality. *automatica*, 26(6):789–814, 2000.

[42] A. M'hamdi, A. Helbig, O. Abel, and W. Marquardt. Newton-type Receding Horizon Control and State Estimation. In *Proc. 13rd IFAC World Congress*, pages 121–126, San Francisco, 1996.

[43] M.J.D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G.A. Watson, editor, *Numerical Analysis, Dundee 1977*, volume 630 of *Lecture Notes in Mathematics*, Berlin, 1978. Springer.

[44] C.V. Rao, S.J. Wright, and J.B. Rawlings. Application of Interior-Point Methods to Model Predictive Control. *Journal of Optimization Theory and Applications*, 99:723–757, 1998.

[45] A. Romanenko, N. Pedrosa, J. Leal, and L. Santos. *Seminario de Aplicaciones Industriales de Control Avanzado*, chapter A Linux Based Nonlinear Model Predictive Control Framework, p. 229, Madrid, Spain., pages 229–236. 2007.

[46] R.W.H. Sargent and G.R. Sullivan. The development of an efficient optimal control package. In J. Stoer, editor, *Proceedings of the 8th IFIP Conference on Optimization Techniques (1977), Part 2*, Heidelberg, 1978. Springer.

[47] J.P. Schlöder. *Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung*, volume 187 of *Bonner Mathematische Schriften*. Universität Bonn, Bonn, 1988.

[48] L.L. Simon, Z.K. Nagy, and K. Hungerbuehler. *Nonlinear Model Predictive Control*, volume 384 of *Lecture Notes in Control and Information Sciences*, chapter Swelling Constrained Control of an Industrial Batch Reactor Using a Dedicated NMPC Environment: OptCon, pages 531–539. Springer, 2009.

[49] M.C. Steinbach. A structured interior point SQP method for nonlinear optimal control problems. In R. Bulirsch and D. Kraft, editors, *Computation Optimal Control*, pages 213–222, Basel Boston Berlin, 1994. Birkhäuser.

[50] T.H. Tsang, D.M. Himmelblau, and T.F. Edgar. Optimal control via collocation and non-linear

programming. *International Journal on Control*, 21:763–768, 1975.

[51] A. Wächter. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University, 2002.

[52] A. Wächter and L. Biegler. IPOPT - an Interior Point OPTimizer. https://projects.coin-or.org/Ipopt, 2009.

[53] Paul Williams, Bas Lansdorp, and Wubbo Ockels. Optimal Crosswind Towing and Power Generation with Tethered Kites. *Journal of Guidance, Control, and Dynamics*, 31(1):81–92, January–February 2008.

[54] S.J. Wright. *Primal-Dual Interior-Point Methods*. SIAM Publications, Philadelphia, 1997.

[55] V. M. Zavala and L.T. Biegler. The Advanced Step NMPC Controller: Optimality, Stability and Robustness. *Automatica*, 45:86–93, 2009. (accepted for publication).