Dynamic optimization using adaptive control vector parameterization

Martin Schlegel, Klaus Stockmann, Thomas Binder, Wolfgang Marquardt *

Lehrstuhl für Prozesstechnik, RWTH Aachen University, D-52056 Aachen, Germany

Abstract

In this paper we present a method for the optimization of dynamic systems using problem-adapted discretizations. The method is based on the direct sequential or single-shooting approach, where the optimization problem is converted into a nonlinear programming problem by parameterization of the control profiles. A fully adaptive, problem-dependent parameterization is generated by repetitive solution of increasingly refined finite dimensional optimization problems. In each step of the proposed algorithm, the adaptation is based on a wavelet analysis of the solution profiles obtained in the previous step. The method is applied to several case study problems to demonstrate that the adaptive parameterization is more efficient and robust compared to a uniform parameterization of comparable accuracy.

Key words: dynamic optimization, sequential approach, adaptive mesh refinement, wavelets, state path constraints

1 Introduction

Dynamic optimization problems arise in many engineering applications. Typical examples from the field of process systems engineering include the design of trajectories for the optimal operation of batch and semi-batch reactors, or for continuous processes in transient phases such as grade transitions, start-up or shut-down. It is still a challenge to obtain a high-quality solution for such problems efficiently, especially when the problem formulation contains large-scale models, e.g. those stemming from industrial applications. However, even for problems with just small models where the computation time is not an

* Corresponding author: marquardt@lpt.rwth-aachen.de

Preprint submitted to Computers & Chemical Engineering 17 January 2004
issue, the solution quality, which can be obtained by employing numerical methods, is not always satisfactory.

One important reason for this fact is that the analytical solution of a dynamic optimization problem consists of one or more intervals, the so-called arcs (Bryson and Ho, 1975). The control variables are continuous and differentiable within each interval, but can jump from one interval to the next at the so-called switching points. Consequently, an optimal control profile might show a quite non-uniform frequency behavior over the optimization horizon. This fact still poses a challenge to all numerical solution methods. Typically, these methods employ some kind of discretization of the continuous quantities (control and state variables). The quality of the solution depends on the chosen discretization resolution, especially of the control variables. The solution quality can be insufficient, if the discretization does not properly reflect the solution structure. However, this structure is typically not known before the problem has been solved. Additional numerical problems can arise in intervals, where model states are constrained by a pre-specified bound (path constraints), or on so-called singular arcs, where the sensitivity of the objective function with respect to the control can be small.

These problems can be addressed by refinement methods, which adapt the discretization grids to the problem in some way. In this paper, a wavelet-based adaptive refinement algorithm is suggested, which is embedded into a direct single-shooting method. In Sections 2 and 3 we introduce the problem formulation and the employed solution method. Section 4 first presents a review of adaptation approaches known from the literature, and then elaborates the details of the proposed multi-scale control mesh adaptation framework. Section 5 briefly explains the software implementation of the concept. The features and benefits of the approach are illustrated by means of three numerical case studies in Section 6, before we conclude in Section 7.

2 Preliminaries

2.1 Problem formulation

We consider an optimal control problem formulation of the following form:

$$\min_{u,p,t_f} \Phi(x(t_f))$$  \hspace{1cm} (P1)
\[ M(x, t) \dot{x} = f(x, u, p, t), \quad t \in [t_0, t_f], \quad (1) \]
\[ 0 = x(t_0) - x_0, \quad (2) \]
\[ x^L \leq x(t) \leq x^U, \quad t \in [t_0, t_f], \quad (3) \]
\[ u^L \leq u(t) \leq u^U, \quad t \in [t_0, t_f], \quad (4) \]
\[ p^L \leq p \leq p^U, \quad (5) \]
\[ 0 \geq e(x(t_f)) . \quad (6) \]

In this formulation, \( x(t) \in \mathbb{R}^{n_x} \) denotes the vector of state variables which can be either of differential or algebraic nature, \( x_0 \) are the given consistent initial conditions. The time-dependent control variables \( u(t) \in \mathbb{R}^{n_u} \) and the unknown time-independent parameters \( p \in \mathbb{R}^{n_p} \) are the degrees of freedom for the optimization. The final time \( t_f \) can be either fixed or an unknown decision parameter, as well. The differential-algebraic (DAE) model is given by the equation system (1) in linear-implicit form. We only consider DAE systems with a differential index of less than or equal to one. The objective function \( \Phi \) is for simplicity formulated as a terminal cost criterion. Note that the more general formulation of an integral cost term can be easily converted into the above formulation by introducing an additional state variable.

Furthermore, path constraints can be formulated for the states (3), control variables (4) and time-independent parameters (5), with \( (\cdot)^L \) and \( (\cdot)^U \) denoting the corresponding lower and upper bounds. Finally, endpoint constraints (6) on the state variables can be employed. Typically, state path and endpoint constraints are not applied on all \( n_x \) state variables.

Note that the path constraints are formulated separately for each control, state and parameter variable, and that only simple bounds are considered. More complex constraints and combined state and control path constraints can be converted easily into this formulation by adding additional equations and auxiliary variables to the model.

### 2.2 Numerical solution methods

Solution strategies for dynamic optimization problems of the form (P1) can be classified into indirect and direct methods.

**Indirect methods** use the first-order necessary conditions from Pontryagin’s maximum principle (Pontryagin et al., 1962; Bryson and Ho, 1975) in order to reformulate the problem as a multi-point boundary value problem. Various techniques are available for solving such a problem. An overview of those methods has been given recently by Srinivasan et al. (2003).

Instead of solving the problem after a reformulation through the maximum
principle, direct methods solve the problem (P1) directly. The control variables $u(t)$ and the state variables $x(t)$ as well as their derivatives with respect to time $\dot{x}(t)$ appear as continuous quantities in this problem. The problem can be converted into a finite dimensional one, usually a nonlinear programming problem (NLP), which subsequently can be solved by a suitable numerical optimization algorithm. This is accomplished by discretization. In general, three different direct methods can be distinguished: the control vector parameterization (or sequential) approach (e.g. Kraft (1985)), the multiple-shooting approach (Bock and Plitt, 1984)) and the full discretization (or simultaneous) approach (e.g. Cuthrell and Biegler (1987)). These methods differ in how the continuous quantities are discretized and which of them are degrees of freedom in the optimization algorithm.

The control vector parameterization approach, also referred to as single-shooting or sequential approach, is the method used in this work and will be described in more detail in Section 3.

3 Control vector parameterization

In the control vector parameterization approach (Kraft, 1985) only the control variables $u(t)$ are discretized explicitly. The discretization parameters are the degrees of freedom for the optimization. The profiles for the state variables $x(t)$ are obtained by forward numerical integration of the DAE system (1) for a given input, which explains the term single-shooting approach. For the discretization of the control profiles $u_i(t)$ often piecewise polynomial approximations are applied. We use a B-spline representation for this purpose.

3.1 B-spline representation of control profiles

Spline functions are an important class of functions used frequently for interpolation and approximation purposes. In contrast to classical polynomial (Lagrange) interpolation a spline interpolation uses local polynomials of low order and connects them at grid points (knots) obeying certain smoothness conditions.

We build our control profile parameterization by means of $B$-splines (de Boor, 1978). On an arbitrarily chosen set of knots $T = \{\tau_1, \ldots, \tau_n\}$ the B-splines
\( \varphi_j^{(m)} \) of order \( m = 1, \ldots, n \) and \( j = 1, \ldots, n - m \) are defined recursively as

\[
\varphi_j^{(1)}(t) := \chi_{[\tau_j, \tau_{j+1}]}(t) = \begin{cases} 
1 & \text{if } \tau_j \leq t < \tau_{j+1} \\
0 & \text{else}
\end{cases}, \tag{7}
\]

\[
\varphi_j^{(m)}(t) := \frac{t - \tau_j}{\tau_{j+m-1} - \tau_j} \varphi_j^{(m-1)}(t) + \frac{\tau_{j+m} - t}{\tau_{j+m} - \tau_{j+1}} \varphi_j^{(m-1)}(t), \ m > 1 \tag{8}
\]

Note that the characteristic function (7) defines a piecewise constant discretization, whereas higher-order approximation can be obtained by recursive application of expression (8). Figure 1 shows the B-splines of orders \( m = 1, 2, 3 \).

![B-splines of orders m = 1, 2, 3](image)

Fig. 1. B-splines of orders \( m = 1, 2, 3 \)

The construction of B-splines implies that a B-spline of order \( m \) is \( C^{m-2} \) continuous at distinct grid points in \( T \). Therefore, besides piecewise constant \( (m = 1) \) and piecewise linear \( (m = 2) \) splines the cubic B-splines \( (m = 4) \) are most commonly applied, since they are twice continuously differentiable.

Special care has to be taken at the boundaries of the interval of interest. Note that, with the convention \( 0/0 = 0 \), Eq. (8) is also valid for coinciding knot points. This feature is used by defining auxiliary knots at each boundary. For the interpolation of a control profile on a given mesh of time points \( \Delta = \{t_0, \ldots, t_l\} \) with a B-spline basis of order \( m \) we require \( n = l + m - 1 \) knots, such that \( \tau_1 = \cdots = \tau_m \) and \( \tau_{n+1} = \cdots = \tau_{n+m} \) are multiple knots at the boundaries. The dimension of the required spline basis is equal to \( n \). For example, for a piecewise constant representation of a control profile with \( l \) intervals (i.e. \( l + 1 \) mesh points) we need \( n = l \) splines of type \( \varphi_j^{(1)} \), whereas a piecewise linear parameterization would require \( n = l + 1 \) splines of type \( \varphi_j^{(2)} \).

In order to allow for a flexible, problem-specific discretization, we choose a separate set \( \Delta_i = \{t_0, \ldots, t_l_i\} \) of discretization time points for each control variable \( u_i \). We then can expand the discretized control profile as a linear combination of B-splines as

\[
u_i(t) = \sum_{j=1}^{n_i} \hat{u}_{i,j} \varphi_j^{(m)} \tag{9}\]

\[5\]
or
\[
    u_i(t) = \hat{u}_{i,m}^T \Phi^m
\]
in vector notation. Note that for piecewise constant control profiles the values of the spline coefficients \( \hat{u}_{i,j} \) just correspond to the values of the control variable \( u_i \) in the specific interval, whereas the spline coefficients of a piecewise linear approximation are equal to the values of the control variable at the mesh points \( t \in \Delta_i \).

### 3.2 Reformulation as NLP

After discretization of the control profiles, the dynamic optimization problem (P1) can be reformulated as a nonlinear programming problem (NLP):

\[
\begin{align*}
    \min_{\hat{u}, p, t_f} & \quad \Phi = \Phi (x(\hat{u}, p, t_f)) \\
\text{s.t.} & \quad x^L \leq x(\hat{u}, p, t_j) \leq x^U, \quad \forall t_j \in \Delta_x, \\
& \quad u^L \leq \hat{u} \leq u^U, \\
& \quad p^L \leq p \leq p^U, \\
& \quad 0 \geq e(x(t_f)) .
\end{align*}
\]

The vector \( \hat{u} = [\hat{u}_i^T, \ldots, \hat{u}_{n_u}^T]^T \) contains all discretization parameters of the control variables. Since the control path constraints (12) are discretized already, their enforcement is straightforward, since the B-spline coefficients \( \hat{u}_i \) are equal to the values of the profile \( u_i \) at the specific mesh points for \( m \leq 2 \), or can be converted into those for \( m > 2 \). Note, however, that for the latter case it cannot be guaranteed that a control profile does not violate a path constraint in the interval between two mesh points.

The state variables \( x(\hat{u}, p, t) \) are obtained from a numerical integration of the DAE system for a given set of discretized control variables and time-independent parameters.

For evaluation of the state path constraints (11) we cannot directly use the continuous formulation from (P1), since the NLP can only handle a finite number of constraints. Therefore, the path constraints are evaluated pointwise at time points \( t_j \) contained in a set \( \Delta_x \), for example on the unified mesh of all control variables \( \Delta_x = \bigcup_{i=1}^{n_u} \Delta_i \).

Once the problem (P2) has been formulated through the discretization of the control vector, it can be solved by a suitable NLP solver. Typically, a sequential quadratic programming (SQP) method is used (Nocedal and Wright, 1999). The function values required by the solver are obtained by repeated numerical
integration of the model (1) in each function evaluation step of the NLP solver. The gradient information of the constraints and the objective function with respect to the decision variables can be obtained in several ways. Here, we use the explicit solution of the arising sensitivity equation systems, which is the method of choice in most sequential dynamic optimization algorithms (see e.g. Vassiliadis et al. (1994)).

3.3 State path constraints

Above we mentioned that it can be guaranteed for a B-spline discretization with \( m \leq 2 \), that the control path constraints are not violated at any time point \( t \in [t_0, t_f] \). However, this is not necessarily the case for state path constraints. Besides the point-wise enforcement as in equation (11), various other approaches have been suggested to tackle this problem in a single-shooting context. Among these are the introduction of slack variables and penalty functions either in the objective function or as additional endpoint constraints. Feehery and Barton (1998) give a comprehensive overview on these methods. They propose a method based on the equivalence of path-constrained and hybrid discrete/continuous dynamic optimization problems. However, in certain cases this method may require the use of second-order information, which is computationally expensive to obtain. For practical applications, the point-wise enforcement is simple to implement and typically has better convergence properties than the other methods. However, it is likely to suffer from constraint violations at time points not contained in \( \Delta_x \).

We would like to illustrate this phenomenon with the following artificial test problem

\[
\min_{u(t)} 1(t_f) \\
\text{subject to} \\
\dot{x} = x + u, \\
x(0) = 1, \\
x(t) = 1, \\
0 \leq t \leq 1.
\]

Obviously, this problem is just a feasibility problem with a state equality constraint \( x(t) = 1 \). The solution is \( u(t) = -1 \). However, if we parameterize \( u \) with just one piecewise linear function and evaluate the constraint at the grid points \( \Delta_x = \Delta = \{t_0, t_1\} \) \( (t_0 = 0 \text{ and } t_1 = 1) \), this corresponds to only demanding \( x(1) = 1 \). For this simple problem, an analytical solution for \( x(t) \) can be given as

\[
x(t) = e^t(\hat{u}_2 + 1) + t(\hat{u}_1 - \hat{u}_2) - \hat{u}_2,
\]

(11)

where \( \hat{u}_1 = u(0) \), \( \hat{u}_2 = u(1) \) are the discrete degrees of freedom. It is obvious now that by enforcing \( x(t) \) just at one single point (such as \( x(t) = 1 \)), \( \hat{u}_1 \) and \( \hat{u}_2 \)
are not uniquely determined by the expression (11). Rather, any combination of \( [\hat{u}_1, \hat{u}_2] \), which fulfills \( \hat{u}_1 = \hat{u}_2(2 - \epsilon) + 1 - \epsilon \) solves the problem. Figure 2 shows \( u \) and \( x \) for two such cases.

![Figure 2](image)

**Fig. 2.** Different linear control profiles leading to the same point-wise fulfillment of the path constraint.

Obviously the constraint is violated except at the grid points of \( u \). However, if we would enforce it in addition e.g. at \( t = 0.5 \) (\( \Delta_x = \{t_0, (t_0 + t_1)/2, t_1\}\)), the correct solution \( u(t) = -1 \) (leading to \( x(t) = 1 \)) would be found. Note that for a piecewise constant discretization in this example an enforcement at \( t = 1 \) would be sufficient to uniquely determine \( \hat{u}_1 \) and therefore \( u(t) \). This example can be viewed as a cut-out of an active path constraint in a real problem discretized with more than one interval. A typical observation is that for too coarse point-wise path constraint enforcement the remaining freedom in the choice of the control variables is used for a further reduction of the objective function at the price of intermediate path constraint violation (Schwartz, 1996). This often leads to oscillatory control profiles on arcs with active state path constraints.

Obviously, for general, possibly nonlinear problems just doubling the path-constraint resolution is not guaranteed to prevent an intermediate path constraint violation. Nevertheless, the amount of violation will always be reduced, and oscillations can be at least significantly damped. Therefore, we suggest the heuristic to (optionally) enforce the state path constraints at additional time points between the mesh points of the control discretization contained in \( \Delta_i \). To be specific, if we have the unified mesh \( \bigcup_{i=1}^{m_u} \Delta_i = \{t_0, \ldots, t_j, \ldots, t_{\Delta}\} \), we can enforce the state path constraints at the time points

\[
\Delta_x = \left\{ t_0, \frac{t_0 + t_1}{2}, t_1, \frac{t_1 + t_2}{2}, \ldots \right\}.
\]  (12)

This formulation realizes a path constraint resolution twice as fine as the control variable resolution. As our case studies show, this is a reasonable approach which works well in practice.
3.4 Resolution of the control variable profiles

In general, it is desirable to obtain a solution of problem (P2) which is close to the true solution of the original optimal control problem (P1). Many publications have discussed approximating a continuous infinite-dimensional optimal control problem by a discrete one. For example, Dontchev and Hager (2001) consider an Euler discretization of state-constrained optimal control problems and show that for a sufficiently small stepsize $h$ the Euler discretization has a solution at a distance of $O(h)$ from a continuous solution. Polak (1993) developed a theory of consistent approximations and showed that general control constrained optimal control problems can be consistently approximated by a sequence of NLPs, where the control profiles are approximated by piecewise constant functions and the Euler method is used for the discretization of the dynamic system. Later, these results have been extended to explicit Runge-Kutta methods (Schwartz, 1996).

Following these results, a very fine discretization mesh $\Delta_i$ for each control variable $u_i$ seems to be a reasonable choice in order to obtain an accurate solution of problem (P1). However, for practical problems this is not a feasible option mainly due to two reasons.

The first reason is numerical accuracy. If a control profile is represented by a very fine discretization of the form (9), the NLP solution algorithm has to deal with a large number of decision variables. However, as indicated before, typical control profiles may exhibit regions where there are no significant differences between the values of neighboring control vector parameters $\hat{u}_{i,j}$, for example in those parts of the control profile where it is governed by an active control path constraint ($u = u_{\text{min}}$ or $u = u_{\text{max}}$). In these regions, a fine discretization would not be required to reflect the true solution. But also in regions, where no constraints are active, namely on singular arcs, a too fine parameterization can be problematic. Since the sensitivity of the problem with respect to the control is typically low on singular arcs, this can lead to numerical difficulties such as ill-conditioning in the optimization of the discretized problem, because the NLP solver might not be able to distinguish between the influence of those parameters within some tolerance. Oscillatory control profiles in these regions are therefore often observed.

The second reason is computational efficiency, because the computational effort required for the solution of a dynamic optimization problem with the sequential method is strongly correlated to the cost for the sensitivity integration. There are efficient algorithms available, which exploit the special properties of the sensitivity system (e.g. Feehery et al. (1997)), but still the largest part of the computational effort is spent on the sensitivity analysis. Since the influence of a decision variable $\hat{u}_{i,j}$ on the states $x$ is limited to the
time region $t \geq t_j$, it is sufficient to solve each sensitivity equation system for the determination of $s_i := \frac{\partial x}{\partial u_{i,j}}$ on the time interval $[t_j, t_f]$. Nevertheless, the computational effort is increasing polynomially with the number of decision variables. This fact is a major bottleneck of the sequential approach. Schlegel et al. (2004) present a sensitivity integration method which can be advantageous for the solution of dynamic optimization problems with highly resolved control variable profiles, but also there the computational effort is proportional to the number of control parameters.

For these reasons, it is therefore clearly desirable to keep the number of decision variables as small as possible. On the other hand the solution obtained by solving the discretized problem (P2) should be close to the solution of the continuous problem (P1). This raises the question of an optimal selection of the discretization grids which balances the computational cost due to the number of parameterization functions with the desired approximation quality. In the following section, we present an adaptive refinement strategy, which generates efficient, problem-adapted meshes $\Delta_i$ for each control variable $u_i$.

4 Multi-scale control mesh adaptation

The problem of choosing optimal discretization grids for direct dynamic optimization problems has been tackled by several authors. One possible way is the movement of the discretization grid points. For a direct simultaneous method this approach has been followed by Cuthrell and Biegler (1987) and von Stryk (1995), where the length or spatial position of the collocation points are determined by the optimizer as additional decision variables. The algorithm described by Vassiliadis et al. (1994) uses a similar concept applied to a sequential method. Again, the number of parameterization intervals is fixed beforehand, but the length of these intervals are also additional degrees of freedom. The drawback of these methods is that the problem becomes more complex and difficult to solve. For example, a linear optimal control problem results in a nonlinear programming problem after discretization. Furthermore, the overall attainable accuracy might be limited, since the number of intervals has to be fixed in advance. In more recent work, again for a simultaneous solution method, the approach has been extended into a bi-level decomposition algorithm where the optimization problem is solved on a fixed mesh in an inner loop, while position and length of the intervals are placed in an outer loop (Tanartkit and Biegler, 1996; Tanartkit and Biegler, 1997).

Instead of grid point relocation, a second strategy focuses on grid point insertion. For example, Waldraff et al. (1997) applied a grid generation procedure based on curvature information of the optimal solution profiles obtained by a sequential method. Schwartz (1996) and Betts and Huffmann (1998) proposed
a grid refinement strategy based on a local error analysis of the differential equations, which are discretized in a simultaneous solution framework. Both, state and control variables are approximated on one mesh by trapezoidal discretization. Starting from a coarse grid, new mesh points are inserted if the local errors of the differential states for a fixed control are larger than a tolerance. Binder et al. (2001) and Binder (2002) also used a local error analysis of the DAE constraints in a simultaneous method. In contrast to the previously mentioned approaches this method allows different grids for state and control variables, both of which are discretized in a multi-scale framework employing wavelets.

In this paper, we present a similar approach applied to a sequential solution method. In order to generate the desired problem-adapted control variable mesh, optimization and a subsequent mesh refinement procedure are carried out in a repetitive procedure, starting from initially coarsely discretized control profiles $u^0_i(\Delta^0)$. For the derivation of consistent approximations, Polak (1993) used a sequence of successively refined dyadic grids for piecewise-constant discretized control profiles. Starting from a single interval, the number of grid points is continuously doubled. Conceptually, our method uses a similar sequence for refinement. However, in order to generate adapted grids, rather than doubling the resolution on the entire horizon, refinement is only carried out where necessary. For this purpose, in each refinement step $\ell$ the previous solution $u^{\ell-1}_i$ is inspected by an a-posteriori analysis. Based on this analysis the new discretization for the next optimization run is generated. Problem (P2) is then resolved on the improved discretization grid $\Delta^\ell_i$ where the interpolated old solution $u^{\ell-1}_i$ is used as an initial guess. Hence, (P2) is resolved repeatedly on different meshes with a (usually) increasing number of parameterization variables. The concepts presented in this paper are an extension and modification of previous work by Binder et al. (2000). We will go into more detail about the differences in Section 4.4.

The analysis of the control profiles consists of two parts: a) grid point elimination and b) grid point insertion. Since the proposed method is based on multi-scale concepts using wavelets, some fundamental properties of wavelets are sketched briefly in the following, before we proceed with the algorithmic details.

4.1 Wavelets

Wavelets are well-suited for a multi-scale analysis of functions, which is often called a multi-resolution analysis. A comprehensive description of wavelets and the theory of multi-resolution methods is beyond the scope of this paper. The reader is referred to the extensive literature in this field (e.g. Dahmen (1997)).
The key idea of a multiresolution method is to develop a hierarchical representation of a function with a collection of coefficients, each of which provides some local information about the frequency of the function. For simplicity, the illustration in this section focuses on the analysis of a generic function \( f \in L^2 \) on the interval \([0, 1]\). We will apply these concepts separately to each control profile \( u_i(t) \) later, but any reference to the index \( i \) is left out here for ease of notation. Furthermore, we restrict the discussion to piecewise constant and piecewise linear representations, though an extension to B-splines of arbitrary order is straightforward (Dahmen, 1997).

As a starting point, we consider a hierarchical decomposition of a given function by a multiresolution sequence of nested spaces \( S_{j_0} \subset \ldots \subset S_j \subset S_{j+1} \subset \ldots \), whose union is dense in \( L^2 \). The index \( j \) is used to denote a specific scale, with \( j_0 \) being the scale with the coarsest discretization. In Figure 3 this is shown for the piecewise constant approximation of a continuous function.

![Figure 3: Approximation of a function on various resolutions.](image)

The spaces \( S_j \) are spanned by compactly supported functions \( \varphi_{j,k} \). If we let \( \varphi_{j,k} \) being defined on the unit interval \([0, 1]\), they can be derived from a single generic basis function \( \varphi_{0,0} \) by dyadic dilations with powers of two and integer translations according to

\[
\varphi_{j,k}(t) = 2^{j/2} \varphi_{0,0}(2^j t - k), \quad k \in 0, \ldots, 2^j - 1.
\]

(13)

Here, the index \( j \) denotes the scale, which corresponds to the level of resolution, whereas \( k \) denotes the translation index on a specific scale.

For the case of piecewise constant functions, the generic function \( \varphi_{0,0} \) (also referred to as the scaling function) is just the characteristic function

\[
\varphi_{0,0}(t) = \chi_{[0,1]}(t) = \begin{cases} 
1 & \text{if } 0 \leq t < 1 \\
0 & \text{else}
\end{cases}
\]

(14)

The basis generated from this particular scaling function is called the Haar basis. It is equivalent to the B-spline \( \varphi_j^{(1)} \) on the interval \([0, 1]\). Similarly, we can also use higher order B-splines as scaling functions. Piecewise linear function representations can be generated from the hat basis, whose scaling function is just the B-spline \( \varphi_j^{(2)} \).

The definition interval (support) of a single-scale function \( \varphi_{j,k} \) has a length proportional to \( 2^{-j} \). Thus, \( 2^{-j} \) can be viewed as the mesh size for the space.
The so-called single-scale representation of the function $f$ on a scale $J$ can be written as

$$f_J = \sum_{k \in \mathcal{I}_J} c_{J,k} \varphi_{J,k}$$

with $\mathcal{I}_J$ denoting the scale dependent index set. We also can introduce a vector notation as

$$f_J = \mathbf{c}^T \Phi_{J,J}.$$  

This single-scale representation could be directly used for the approximation of the control profiles. However, by construction $\varphi_{J,k}$ is defined on an equidistant mesh corresponding to the resolution of scale $J$. In contrast, the spline basis used in (9) allows possibly different interval lengths corresponding to the chosen discretization mesh $\Delta$, which is a key requirement for mesh point adaptation. Therefore, we introduce the local single-scale function $\hat{\varphi}_{J,k}$. The local single-scale representation simply involves just the significant knot points of the function expansion, as it is depicted in Figure 4 for the Haar basis and the hat basis. Then, the function $f$ on scale $J$ can also be expressed as

$$f_J = \sum_{k \in \hat{\mathcal{I}}_J} \hat{c}_{J,k} \hat{\varphi}_{J,k} = \hat{\mathbf{c}}^T \hat{\Phi}_{J,J},$$

where $\hat{\mathcal{I}}_J \subseteq \mathcal{I}_J$ comprises a suitable subset of the translation indices. Obviously, $\hat{\mathcal{I}}_J$ always has to be adapted to the specific function to be approximated. Then, $f_J$ can be expressed by (17) with fewer parameters than by (15). Note
that (17) is equivalent to the spline representation (9), with the coefficient \( \hat{u}_j \) differing from \( \hat{c}_{j,k} \) just by the scaling factor \( 2^{j/2} \).

Now, given a multiresolution sequence, one way of updating an approximation \( f_j \in S_j \) to a function \( f_{j+1} \) is to add some detail information \( f_j + w_j = f_{j+1} \) where \( w_j \) belongs to some complement \( W_j \) of the space \( S_j \) in the next finer space, i.e. \( S_{j+1} = S_j \oplus W_j \). In other words, the space \( W_j \) contains the differences between the approximations in \( S_j \) and \( S_{j+1} \). An example for this concept is depicted in Figure 5. We can introduce a basis \( \{ \psi_{j,k} : k \in \mathcal{J}_j \} \) for \( W_j \). Then the detail \( w_j \) can be written as \( w_j = \sum_{k \in \mathcal{J}_j} d_{j,k} \psi_{j,k} \). With successive decomposition of \( S_{j+1} \) into \( S_{j+1} = S_{j_0} \oplus \sum_{k=j_0}^{J} W_k \), the function \( f_{J+1} \) is associated with a multi-scale representation

\[
f_{J+1} = \sum_{k \in \mathcal{J}_{j_0}} c_{j_0-1,k} \varphi_{j_0,k} + \sum_{l=j_0}^{J-1} \sum_{k \in \mathcal{J}_l} d_{l,k} \psi_{l,k} \tag{18}
\]

A representation of \( f \) in \( S_j \) is therefore equivalent to taking \( f_{j_0} \) and adding the differences contained in \( W_{j_0}, \ldots, W_j \). Under certain assumptions on the complements \( W_j \) the complement basis functions \( \psi_{j,k} \) are called wavelets (Dahmen, 1997). For the Haar basis, the wavelet is just given by

\[
\psi_{0,0}(t) = \begin{cases} 
1 & \text{if } 0 \leq t < 1/2 \\
-1 & \text{if } 1/2 \leq t < 1 \\
0 & \text{else}
\end{cases}
\tag{19}
\]

Figure 6 shows the scaling functions and the corresponding wavelets for the Haar basis and the hat basis.

The whole collection \( \{ \varphi_{j_0,k} : k \in \mathcal{J}_{j_0} \} \cup \{ \psi_{j,k} : k \in \mathcal{J}_j, j = j_0, j_0+1, \ldots \} \) is a basis for \( L_2 \). By introducing \( d_{j_0-1,k} := c_{j_0,k} \) and \( \psi_{j_0-1,k} := \varphi_{j_0,k} \) for ease of notation, we can decompose the function \( f \) on a level \( J \) as

\[
f_J = \sum_{j=j_0-1}^{J-1} \sum_{k \in \mathcal{J}_j} d_{j,k} \psi_{j,k} \tag{20}
\]
In vector form we write
\[ f_J = d^T \Psi_{A_J}, \]  
(21)
where \( A_J := \{(j, k) \mid j_0 - 1 \leq j \leq J - 1, k \in J_j\} \) is the index set of the full wavelet basis. Typically, one chooses \( j_0 \) as the lowest possible scale, which is 0 for the Haar basis and 1 for the hat basis.

A conversion of the single-scale formulation (16) to the wavelet representation (21) can be accomplished by applying the fast wavelet transformation (FWT) (Mallat, 1989)
\[ d = T c, \quad c = T^{-1} d, \]  
(22)
where the transformation operator \( T \) is stemming from a recursive application of special filter matrices.

The magnitude of a wavelet coefficients \( d_{j,k} \) is a measure for the local frequency content of the function. A graphical representation can be obtained by using the time-scale plot of a wavelet decomposition, where rectangular patches are used to represent a particular wavelet \( \psi_{j,k} \) in its coordinates in time and scale. The absolute value of \( d_{j,k} \) can be visualized by a specific color of the patch. Figure 7 shows a function and the time-scale plot of the corresponding wavelet decomposition. Some of the patches are denoted by their corresponding wavelet functions. Such a plot gives a useful visualization of the level of local detail contained in the analyzed function.

Wavelets are constructed such that the Euclidean norm of the wavelet coefficients of \( f \) is always proportional to the \( L_2 \) norm of \( f \) (norm equivalence). An
important property of the multi-scale representation is the *norm equivalence*

\[ \|f_J\|_{L^2} \sim \|a^T\|_{\ell^2} \quad (23) \]

Thus, discarding small coefficients \(d_{j,k}\) will cause only small changes in approximate representations of \(f\). Moreover, it can be shown for \(f\) being smooth on the support of \(\psi_{j,k}\) that the values \(|d_{j,k}|\) will be quite small. Therefore, functions that exhibit strong variations only locally, can be approximated very accurately by linear combinations of only relatively few wavelet basis functions corresponding to the *significant* coefficients \(d_{j,k}\). Denoting the set of the corresponding indices \((j,k)\) by \(\Lambda\), the finite basis \(\Psi_\Lambda := \{\psi_{j,k} : (j,k) \in \Lambda\}\) is expected to lead to a particularly *adapted* approximation to \(f\). We refer to \(\Psi_\Lambda\) as a *sparse wavelet basis* in contrast to the *full wavelet basis* \(\Psi_J\). Such a sparse wavelet representation can be converted into a local single-scale representation \((17)\) and vice versa:

\[ f_J = \hat{c}^T \Phi_J = d^T \Psi_\Lambda \quad (24) \]

This conversion can be accomplished simply by interpolation of \(\hat{c}\) on the equidistant grid in order to obtain the single-scale representation and then apply the transformation \((22)\) to obtain \(d\). Note that number of entries in the index sets \(\tilde{J}_J\) and \(\Lambda\) and therefore the size of the vectors \(\hat{c}\) and \(d\) is the same. This conversion is important in order to exploit the specific properties and advantages of the different representations. Both allow a sparse representation of the function, but the local single-scale representation allows simple point-wise function evaluations, whereas at every time point wavelets from different scales may contribute to the function, so that a point-wise evaluation in the wavelet representation is more involved. On the other hand, the wavelet representation is advantageous for deriving adaptive refinement procedures, as it is described in the following section. After having sketched fundamental properties of wavelets we can proceed to outline the algorithmic details of the refinement strategy.

---

1 The proportionality is valid for approximation orders \(m > 1\). For orthogonal wavelet bases such as the Haar basis, even a norm equality holds.
4.2 Refinement strategy

The concepts explained in Section 4.1 can be utilized for the a-posteriori analysis of the optimal solution $\hat{u}_i^\ell(\Delta^\ell)$ found in the refinement step $\ell$. The result of the analysis is a new discretization mesh $(\Delta_{i+1}^\ell)$ which is better adapted to the solution structure and which is used as the starting point for the subsequent optimization run. The analysis involves two steps, a grid point elimination and a grid point insertion step.

Both are based on a wavelet representation of the control profile. Therefore, the non-equidistant spline representation is first converted to the equivalent wavelet form along the following steps:

Algorithm 1 Conversion from time domain into (sparse) wavelet domain

Given: $u(t) = \hat{u}^T \Phi^{(m)}$, corresponding time grid $\Delta$.
Interpolate on finest mesh (scale $J$) $\to \hat{u}_J$
for $k = 1 \ldots 2^J$ do
  $c_{J,k} = u_{k,J} \cdot 2^{-J/2}$ \{Conversion into single-scale domain\}
end for
Perform fast wavelet transformation $\to d = Tc$, index set $\Lambda_J$.
Select wavelet coefficients corresponding to $\hat{d}$, index set $\hat{\Lambda}$.

After this conversion, the control profiles are available in the sparse wavelet domain as $u_i(t) = \hat{d}_i^T \Psi_{\hat{\Lambda}_i}$ for the subsequent analysis steps.

4.2.1 Grid point elimination

The idea of grid point elimination is to remove unnecessary grid points in the representation of $\hat{u}_i^\ell$, the optimal solution at refinement step $\ell$. The goal is to obtain a “minimum representation” of $\hat{u}_i^\ell$ in the sense that an approximation $\tilde{u}_i^\ell$ satisfies

$$\frac{\|\hat{u}_i^\ell - \tilde{u}_i^\ell\|_{L^2}}{\|\hat{u}_i^\ell\|_{L^2}} \leq \varepsilon'$$

with a given tolerance $\varepsilon'$.

This can easily be accomplished in the wavelet space by employing the norm equivalence (1), which allows us to simply neglect those elements of $\hat{d}_\Lambda^\ell$ which are smaller than a threshold $\varepsilon_c$ which uniquely depends on $\varepsilon'$. In the current implementation, the elimination threshold $\varepsilon_c$ is a user-specified parameter. The entries of the eliminated wavelets are collected in the index set $\hat{\Lambda}_i^\ell + 1 \subseteq \hat{\Lambda}_i^\ell$. The remaining entries are the significant coefficients.

\footnote{In the algorithm, the indices $i$, $\ell$ and $*$ have been left out for ease of notation.}
4.2.2 Grid point insertion

The key problem in grid refinement to better approximate \( u_i \) lies in identifying those areas of the profile which require a higher resolution. In the same way, as the wavelet analysis of the optimal solution can be used to eliminate unnecessary grid points, it also allows to determine those regions of the control profile, where a refinement is indicated.

As seen above, small wavelet coefficients in \( \mathbf{d}_i^\ell \) enable us to discard the corresponding mesh points since their contribution to the profile is negligible. On the other hand, large wavelet coefficients are significant for an accurate representation of the profile. The multi-scale property together with smoothness assumptions suggests that neighbors of wavelets with large (absolute) coefficients are good candidates for refinement. In this setting the wavelet functions \( \psi_{j,k} \) and \( \psi_{j,k+1} \) as well as the functions of the higher scale, \( \psi_{j+1,2k} \) and \( \psi_{j+1,2k+1} \), are referred to as neighbors of the wavelet function \( \psi_{j,k} \). This definition corresponds to the local neighborhood of \( \psi_{j,k} \) in the time-scale representation of wavelet functions. In this context, we call those wavelets free, whose corresponding wavelet coefficients are zero.

First, one takes the subset of \( \hat{A}_i^\ell \) that comprises those wavelets that have at least one free neighbor. This subset will be termed boundary and its elements boundary wavelets (this definition, again, corresponds to the time-scale representation of wavelets). The corresponding wavelet coefficients are denoted \( \hat{\mathbf{d}}_i^\ell \subseteq \hat{\mathbf{d}}_i^\ell \). Figure 8 shows a control profile, its time-scale representation and the boundary wavelets marked by \( \times \).

![Control profile with discretization time points \( \Delta \), corresponding time-scale plot and boundary wavelets \( \Psi_A \) (marked by \( \times \)).](image)

Now, a certain number of boundary wavelets possessing the largest absolute coefficients is marked. This number is chosen such that the \( \ell_2 \)-norm of the selected wavelets amounts to some pre-specified percentage \( \varepsilon_i \) of the norm of
the whole boundary as
\[
\| \tilde{d}_{\Lambda_i}^{\ell} \|_{\ell_2} = \varepsilon_i \| \tilde{d}_{\Lambda_i}^{\ell} \|_{\ell_2} .
\]  
(26)

For the example shown in Figure 8 and a value of \( \varepsilon_i = 0.9 \), we get those boundary wavelets marked in Figure 9 (left). Note that in this example also one wavelet function has been eliminated (marked with \( \circ \)), because the value of the coefficient \( d_{3,6} < \varepsilon_e \) (cf. Section 4.2.1).

Finally, the neighbors of the marked wavelets \( \tilde{\Psi}_{\Lambda_i}^{\ast \ell} \) are select, yielding an index set \( \tilde{\Lambda}_i^{\ell+1} \). In principle, the refinement can be done in horizontal (selecting possible free neighbors on the same scale) and/or vertical dimension (selecting free neighbors on the next higher scale(s)). A vertical refinement is always required in order to obtain areas with finer resolution. Tree-like structures, the so-called wavelet-trees are the result of a vertical refinement. The degree of vertical refinement \( \delta_v (1,2, \ldots \text{added scales}) \) is an additional degree of freedom.

4.2.3 Refined index set

During a refinement step always first a grid point elimination and secondly a grid point insertion step is carried out. It is ensured that no wavelets are added due to the insertion criterion, which have been previously removed already in the elimination step. Therefore, we construct the final refined index set as
\[
\hat{\Lambda}_i^{\ell+1} = \left( \hat{\Lambda}_i^{\ell} \cup \tilde{\Lambda}_i^{\ell+1} \right) \setminus \tilde{\Lambda}_i^{\ell+1} ,
\]  
(27)
i.e. we take the index set of the previous optimization solution, unite it with the set of the prospective new functions and exclude those eliminated. The index set \( \hat{\Lambda}_i^{\ell+1} \) can be converted into the corresponding time point set \( \Delta_i^{\ell+1} \), which is finally used in the upcoming optimization.
The time-scale plot of the new, adapted mesh for our example is shown in the right plot of Figure 9. Here, no horizontal refinement \((\delta_h = 0)\) and a vertical refinement of \(\delta_v = 1\) have been chosen, which is a reasonable choice proven in our studies.

In some cases it might happen that the grid point elimination and insertion criteria lead to contradictory conclusions. Consider for example the wavelet \(\psi_{2,3}\) in Figure 8, right. In the current parameterization it is not used (therefore the corresponding patch in the fourth row of the time-scale plot is not marked), but this might be the result of an elimination in the previous step, which fits well to the constant profile of \(u\) in that region (Figure 8, left). However, as can be seen in Figure 9, right, \(\psi_{2,3}\) is now added as potential parameterization function for the next run, after which it might get eliminated again, and so forth. In order to avoid such a cycling effect, it is assured by the algorithm that \(\Lambda_{i+1}\) does not contain indices of wavelets that have been already deleted during the grid point elimination phase of the refinement procedure.

4.2.4 Point-wise evaluation of state path constraints

In Section 3.3, we developed the heuristic to perform the point-wise evaluation of the path constraints on a mesh with the double resolution of the control profiles. However, for very fine control variable resolutions this would lead to a significant increase of the number of constraints (11) to be dealt with by the NLP solver. Therefore, it is reasonable to also integrate the grid for the state path constraint evaluation in the adaptation procedure. Since we solve a sequence of refined problems, it is possible to determine those regions in the control profile where it is governed by active state path constraints. This information then can be utilized to add additional points for intermediate point-wise path constraint enforcement only in those regions, and therefore limit the additional effort for the NLP solver to the extent necessary. Thereby, it is ensured that the resolution of the path constraint mesh is at least twice as fine as the proposed, new control profile grids. In concrete terms, we employ the following algorithm:
Algorithm 2: Addition of points for state path constraint evaluation

Given: $\Delta^\ell_k, \Delta_{i}^{\ell+1}, i = 1, \ldots, n_u$

$\Delta_{x}^{\ell+1} \leftarrow \Delta^\ell_k$

for $j = 1 \ldots n_{\Delta x}$ do

if $x^\ell(t_j) = x^k$ or $x^\ell(t_j) = x^U$ then

$t_{n}^{\ell n} \leftarrow 0.5 \cdot (t_{j-1} + t_j)$, $t_{r}^{\ell n} \leftarrow 0.5 \cdot (t_{j+1} + t_j)$

if $t_{n}^{\ell n} \in \Delta_{i}^{\ell+1}$ for any $i$ then

$t_{n}^{\ell n} \leftarrow 0.25t_{j-1} + 0.75t_j$

end if

if $t_{r}^{\ell n} \in \Delta_{i}^{\ell+1}$ for any $i$ then

$t_{r}^{\ell n} \leftarrow 0.25t_{j+1} + 0.75t_j$

end if

end if

$\Delta_{x}^{\ell+1} \leftarrow \Delta_{x}^{\ell+1} \cup \{t_{n}^{\ell n}, t_{r}^{\ell n}\}$

end for

This algorithm implies that the middle points of the left and right time intervals at any active state path constraint point are added to the new mesh $\Delta_{x}^{\ell+1}$ for the constraint evaluation. However, if these middle points are already contained in any of the new control constraints grids $\Delta_{i}^{\ell+1}, i = 1, \ldots, n_u$ (as a result from the grid point insertion) then again the middle of the active point and such a new point is added.

4.2.5 Convergence properties and stopping criterion

Another interesting question which arises is, how far convergence properties can be guaranteed for the successive refinement algorithm. Due to the grid refinement and the dyadic nature of the wavelet grids, the grid points contained in $\Delta^\ell_k$ are a subset of those contained in $\Delta_{i}^{\ell+1}$. Consequently, the same statement is valid for the degrees of freedom (due to the control profiles) in the NLP problems $\ell$ and $\ell + 1$. Therefore, the optimal objective value $\Phi^{\star \ell}$ is an upper bound for the subsequent objective value $\Phi^{\star \ell+1}$, provided converged solutions of the involved NLP problems have been found. However, this holds only true in general for problems without state path constrains (which are evaluated point-wise). As discussed in Section 3.3, it can happen that the solver finds a better optimum at the price of intermediate path constraint violations. Therefore, cases can occur, where the objective function value $\Phi^{\star \ell}$ is actually lower than the one obtained on a refined control mesh $\Phi^{\star \ell+1}$, because the amount of path constraint violation in iteration $\ell + 1$ might be smaller.

Based on these arguments, a heuristic stopping criterion for the algorithm can

\[ \text{This statement is valid in those cases where no elimination has been carried out. This is typically the case, if the refinement is started from a coarse initial mesh.} \]
be formulated based on checking the relative improvement of the objective function in each refinement step. If

\[
\frac{\Phi^{s\ell} - \Phi^{s\ell-1}}{\Phi^{s\ell}} \leq \varepsilon_{\Phi},
\]

where \(\varepsilon_{\Phi}\) is a user-specified tolerance, and

\[
\Phi^{s\ell} \leq \Phi^{s\ell-1},
\]

then the algorithm stops.

4.3 Summary of the adaptive algorithm

The specific details of the various elements of adaptive refinement algorithm have been described above. They are now summarized in order to present an overview of the algorithm:

Algorithm 3 Adaptive grid refinement algorithm

Choose initial control profiles and grids: \(u_0^i(t), \Delta_1^0, i = 1 \ldots n_u\)

Specify tolerances: \(\varepsilon_{\Phi}, \varepsilon_c, \varepsilon_i\)

Specify refinement level: \(\delta_h, \delta_v\)

Choose maximum number of refinement iterations: \(\ell_{\text{max}}\)

\[\Delta^0_x = \bigcup_{i=1}^{n_u} \Delta^0_i\]

for \(\ell = 0, 1, \ldots\) do

B-spline parameterization: \(u_\ell^i(t) = \hat{u}_\ell^iT \Phi_i^{(m)}\)

Solve optimization problem (P2\(\ell\)) on \(\Delta^\ell_i\), state path constraints evaluated at \(\Delta^\ell_x\)

\(\rightarrow\) optimal solution \(\hat{u}_\ell^i, p^i, t^i\), \(\Phi^\ell\)

if \(\left(\frac{\Phi^{s\ell} - \Phi^{s\ell-1}}{\Phi^{s\ell}}\right) > \varepsilon_{\Phi} \text{ or } \Phi^{s\ell} > \Phi^{s\ell-1} \text{ (for } \ell > 0 \text{ and } \ell < \ell_{\text{max}}\) then

Conversion into wavelet domain (Algorithm 1): \(\hat{u}_\ell^i, \Delta^\ell_i \rightarrow \hat{d}_\ell^i, \Lambda^\ell_i\)

i) Grid point elimination \(\rightarrow \hat{\Lambda}^\ell_{i+1}\)

ii) Grid point insertion \(\rightarrow \hat{\Lambda}^\ell_{i+1}\)

\(\rightarrow\) refined grids: \(\hat{\Delta}^\ell_{i+1}\)

iii) Refinement of path constraint grid (Algorithm 2) \(\rightarrow \hat{\Delta}^\ell_{x+1}\)

Interpolate profiles: \(u_\ell^{i+1} := \text{interp}(u_\ell^i, \hat{\Delta}^\ell_{i+1})\)

else

Exit

end if

end for

\(\rightarrow\) optimal solution on adaptive grid: \(u^*_i = u^\ell_i, \Phi^* = \Phi^{s\ell}\)

As shown in the algorithm, the optimal solution \(u^\ell_i\) found in iteration \(\ell\) is interpolated on the refined grid \(\Delta^\ell_{i+1}\). This interpolation is then used as the
starting point of the optimization run in iteration \( \ell + 1 \) and provides a good initial guess for the NLP solver.

It should also be mentioned, that the computational effort of the wavelet conversions and other grid refinement operations are negligible compared to the cost for one optimization step, i.e. an NLP solution with underlying state and sensitivity integrations.

4.4 Alternative approach: sensitivity-based refinement

The grid point elimination and insertion as described in Sections 4.2.1 and 4.2.2 are purely signal-based, in the sense that the control profile is refined based on an analysis treating it as a signal, not considering it as the solution of a dynamic optimization problem. This aspect is included in the algorithm through the repeated solution of the refined NLP problems. One might ask, whether it is possible to consider the optimization problem as such already in the determination of the new grid points. This approach has indeed been followed in previous work by Binder et al. (2000). In this contribution, the elimination step is carried out exactly in the same way as described in Section 4.2.1. However, instead of the signal-based insertion method proposed here, Binder et al. (2000) used a refinement criterion based on gradient information of the prospective new discretization functions. The analysis is based on the Lagrangian function of the optimal control problem (P1). After the solution of the discretized problem in refinement step \( \ell \), the gradients of the Lagrangian function at the optimal point with respect to potentially new parameterization functions (wavelets or single-scale functions) are computed. Those grid points, whose corresponding parameterization functions exhibit large Lagrangian gradients, are added to the mesh of problem \( \ell + 1 \). The method can be also interpreted as an application of the parametric sensitivity analysis of nonlinear programming problems (Fiacco, 1983), because the potential grid refinement can be formulated as a disturbance to the optimum. This method, however, is not easily applicable to problems with state and/or control path constraints, because the refinement might result in a change of the active constraint set and would therefore require a complicated analysis.

The signal-based analysis, however, works for path-constrained problems in a straightforward way. As we will show with our case studies, quite satisfactory results can be obtained by this method.
5 Implementation

The numerical concepts presented above have been implemented into the prototype software tool DyOS (DyOS, 2002). The implementation allows the DAE model (1) to be accessed via a so-called ESO (Equation Set Object) interface definition, as defined in the CAPE-OPEN project (Keeping and Pantelides, 1998). For example, the modeling and simulation package gPROMS (gPROMS, 2002) provides such an interface and has been used as the model server in our implementation. The implementation is following the lines described elsewhere (Schlegel et al., 2001).

DyOS allows the user to specify the control parameterization for each control variable separately, either as piecewise constant, piecewise linear or piecewise cubic spline representation, where the adaptive mesh refinement is currently only implemented for the constant and linear case. It is possible to provide initial control profiles with equidistant or non-equidistant profiles, for those cases where some initial knowledge about the control profiles is available. If adaptive refinement is chosen, the initial profiles are rounded to a dyadic grid, if they are not already specified in such a way. Furthermore, time-variant bounds on the control profiles can be specified. Note that all these settings can be selected independently for each of the $n_u$ control variables.

Currently, DyOS is interfaced to the SQP solvers NPSOL (Gill et al., 1986) and SNOPT (Gill et al., 1998), and to the Gauss-Newton solver DFNLP (Schittkowski, 2003) for parameter estimation applications. There is the choice of three different numerical integrators, DDASAC (Caracotsios and Stewart, 1985), DDASPK (Li and Petzold, 1999) or a modified version of LIMEX (Schlegel et al., 2004).

6 Case studies

In this section we present some numerical case studies. The adaptive refinement method is applied to several problems to demonstrate that an adaptive parameterization is more efficient and robust compared to a uniform parameterization of comparable accuracy.

6.1 Example 1: Semi-batch reactor problem

This problem considers the operation of an isothermal semi-batch reactor with parallel reactions and selectivity constraints. The details of the problem are
discussed by Srinivasan et al. (2003). The model consists of three differential and two algebraic variables and equations. The control variable is the feed rate to the reactor. The optimal control profile starts at the maximum, then moves to a singular arc and finally to the minimum.

The problem has been solved by the proposed adaptive refinement methods. The settings for the various solution parameters are listed in Table 1.

<table>
<thead>
<tr>
<th>NLP solver</th>
<th>NPSOL</th>
<th>opt. tol.</th>
<th>$10^{-5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>integrator</td>
<td>LIMEX</td>
<td>int. tol.</td>
<td>$10^{-7}$</td>
</tr>
<tr>
<td>$\varepsilon_\Phi$</td>
<td>$10^{-5}$</td>
<td>$\delta_h$</td>
<td>0</td>
</tr>
<tr>
<td>$\varepsilon_e$</td>
<td>$10^{-7}$</td>
<td>$\delta_e$</td>
<td>1</td>
</tr>
<tr>
<td>$\varepsilon_i$</td>
<td>0.9</td>
<td>( )</td>
<td>( )</td>
</tr>
</tbody>
</table>

Table 1
Settings for Example 1.

The problem has been solved starting from an equidistant parameterization with four piecewise constant splines. The Figures 10 and 11 show the optimal control profiles obtained for $\ell = 0, 1, 2$ and for $\ell = 3, 4, 5$, respectively. After five refinements, i.e. six optimizations, the stopping criterion was met. The profiles show nicely, how the solution is successively refined starting from the very coarse initial solution. Especially the corresponding wavelet coefficients given as time-scale plots show that indeed parameterization functions are inserted locally, where it is required. Note the two wavelet trees evolving around the two switching points.

Table 2 shows computational statistics for the different refinement iterations. The results allow the conclusion that the effort for further improving the solution quality in terms of the objective function value is growing with increasing updates. In other words, minor improvements in the objective function are only available for a significant additional effort. This can be also observed in Figure 12, where the evolution of the objective function value over the iteration counter as well as the corresponding computation times (per iterations and accumulated) are shown. On the other hand, the refinement algorithm provides intermediate solutions of reasonable quality already after the first few refinement steps at low computational cost.

As Figure 11 shows, the finest resolution obtained after five refinement steps is on scale 7 of the wavelet hierarchy, which would correspond to a mesh size of $2^7 = 128$ grid points, if an equidistant resolution would have been chosen. Therefore, a reference solution has been computed with 128 equidistantly spaced piecewise constant elements. The solution profile obtained is shown in Figure 13.
Fig. 10. Example 1: Solution profiles and wavelet coefficients for $\ell = 0, 1, 2$.

<table>
<thead>
<tr>
<th>$\ell$</th>
<th>CPU time</th>
<th>acc. CPU time</th>
<th>no. of dof</th>
<th>$\Phi^\ell$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.09</td>
<td>0.09</td>
<td>4</td>
<td>-0.4286537</td>
</tr>
<tr>
<td>1</td>
<td>0.20</td>
<td>0.30</td>
<td>8</td>
<td>-0.431923</td>
</tr>
<tr>
<td>2</td>
<td>0.59</td>
<td>0.89</td>
<td>14</td>
<td>-0.4316754</td>
</tr>
<tr>
<td>3</td>
<td>1.05</td>
<td>1.94</td>
<td>19</td>
<td>-0.4317106</td>
</tr>
<tr>
<td>4</td>
<td>2.03</td>
<td>3.97</td>
<td>30</td>
<td>-0.4317207</td>
</tr>
<tr>
<td>5</td>
<td>4.30</td>
<td>8.27</td>
<td>45</td>
<td>-0.4317212</td>
</tr>
<tr>
<td>eq.</td>
<td>29.00</td>
<td>29.00</td>
<td>128</td>
<td>-0.4317264</td>
</tr>
</tbody>
</table>

Table 2
Example 1: Computational statistics.
Fig. 11. Example 1: Solution profiles and wavelet coefficients for $\ell = 3, 4, 5$.

Fig. 12. Example 1: Evolution of the objective function value and the computation time.

Apparently, the profile looks slightly smoother than the best obtained by the refinement algorithm, which is due to the much larger number of degrees of freedom. That the adaptive solution is of high quality, though comprising much fewer degrees of freedom, becomes apparent when we compare the objective function value of $\Phi^*_{eq} = -0.431726$ (equidistant), which is only slightly better than the value $\Phi^*_{ad} = -0.431721$ obtained by the adaptive procedure.
A further point of interest is the comparison of the computation time required for both solutions. In Table 2, also the computational statistics for the equidistant solution are given. The results show that the solution time required for the equidistant solution is much higher, due to the fact the number of degrees of freedom is almost three times larger as the number on the highest level of refinement. Both problems have been started from the same initial guess, a constant value of $u = 1 \cdot 10^{-4}$. Although within the adaptive framework six optimization problems had to be solved, the overall solution time is still much lower. This is because the solution obtained in a refinement step $\ell$ is always used as the starting point for the optimization in step $\ell + 1$.

6.2 Example 2: Obstacle problem

The obstacle problem has been considered by Schwartz (1996). The problem has a control constraint and two state path constraints and can be formulated as

$$\min_{u(t)} 5x_1^2(t_f) + x_2^2(t_f)$$

s.t. \hspace{0.5cm} \dot{x}_1 = x_2 , \\
\hspace{0.5cm} \dot{x}_2 = u - 0.1(1 + 2x_1^2)x_1 , \\
\hspace{0.5cm} x_3 = 9(x_1 - 1)^2 + \left(\frac{x_2 - 0.4}{0.3}\right)^2 - 1 , \\
\hspace{0.5cm} x(0) = [1, 1, 3]^T , \\
\hspace{0.5cm} -1 \leq u \leq 1 , \\
\hspace{0.5cm} -0.8 \leq x_2(t) , \\
\hspace{0.5cm} 0 \leq x_3(t) , \\
\hspace{0.5cm} 0 \leq t \leq 2.9 .$$

The optimal control profile is first at the maximum value for a very short time and then jumps to the minimum. In a third arc, it is determined by the path
constraint on $x_2$, and in the last interval it goes to the maximum value. This example is well-suited to illustrate the effect of a point-wise evaluation of path constraints in conjunction with a piecewise-linear control parameterization and the adaptive refinement procedure.

First, we will have a look at the solution with the adaptive refinement method, starting from a piecewise-linear parameterization on an equidistant mesh with four intervals, with a constant initial guess of $u = 0$. The tolerances and adaptation parameters have been chosen according to Table 1, except that SNOPT has been chosen as the NLP solver for this problem. In the upper part of Table 3 the evolution of the solution over the refinement iterations is listed. In addition to Example 1 a column has been included which shows, how many (here adaptively distributed) points have been used for the point-wise evaluation of the path constraints. Figure 14 shows the optimal control profiles and the corresponding wavelet trees for $\ell = 0, 2, 4, 7$. Additionally, for each iteration the profile of the constrained state variable $x_2$ is shown, including an enlargement of the active part of the constraint. It can be observed that the path constraint violation (regions, where $x_2 < -0.8$) is quite significant in the first couple of iterations, especially in the very first. Since the refinement could not build up any knowledge about active regions in iteration $\ell = 0$, the path constraint is only evaluated at the rather coarse control mesh. However, in the course of the adaptation the algorithm accumulates additional points for the enforcement of the path constraint following the rule of guaranteeing at least a twice as fine resolution compared to the control mesh (cf. Section 4.2.4). Consequently, the constraint violation is more and more reduced, leading to a very accurate fulfillment of the constraint in the later refinement steps, as it can be seen in the lower two rows of Figure 14.

Figure 15 shows the evolution of the objective function value and the computation time over the iteration counter. It can be observed that the objective function value $\Phi^{*1}$ is significantly larger than $\Phi^{*0}$. This can be explained by the better fulfillment of the path constraints, leading to a smaller feasible region for the optimization in the later iterations, as discussed in Section 4.2.5. From iteration $\ell = 1$ on, the objective function value decreases continuously, like in Example 1.

The finest resolution of optimal profile at refinement iteration $\ell = 7$ would correspond to an equidistant profile of 512 intervals. However, since this is only the case at one point of the profile, we rather consider an equidistant solution with 256 intervals for comparison. Here, two cases are studied: a) the path constraints being evaluated on the same mesh as the control variables, i.e. at 256 points) and b), the path constraints being evaluated at 512 points. The solution for case a) is depicted in Figure 16, and for case b) in Figure 17. The computational statistics can be found in the lower part of Table 3. The solution profiles in Figure 16 nicely demonstrate the effect of oscillations due
Fig. 14. Example 2: Control profiles, wavelet coefficients and profile of $x_2$ for $\ell = 0, 2, 4, 7$.

Fig. 15. Example 2: Evolution of the objective function value and the computation time.
<table>
<thead>
<tr>
<th>( \ell )</th>
<th>CPU time</th>
<th>acc. CPU time</th>
<th>no. of dof</th>
<th>path</th>
<th>( \Phi^{\ell} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.13</td>
<td>0.13</td>
<td>5</td>
<td>4</td>
<td>0.03533</td>
</tr>
<tr>
<td>1</td>
<td>0.08</td>
<td>0.20</td>
<td>7</td>
<td>8</td>
<td>0.07610</td>
</tr>
<tr>
<td>2</td>
<td>0.16</td>
<td>0.36</td>
<td>13</td>
<td>17</td>
<td>0.03575</td>
</tr>
<tr>
<td>3</td>
<td>0.39</td>
<td>0.75</td>
<td>19</td>
<td>26</td>
<td>0.03334</td>
</tr>
<tr>
<td>4</td>
<td>0.41</td>
<td>1.16</td>
<td>25</td>
<td>41</td>
<td>0.03165</td>
</tr>
<tr>
<td>5</td>
<td>0.20</td>
<td>1.36</td>
<td>28</td>
<td>59</td>
<td>0.03118</td>
</tr>
<tr>
<td>6</td>
<td>0.42</td>
<td>1.78</td>
<td>33</td>
<td>85</td>
<td>0.03108</td>
</tr>
<tr>
<td>7</td>
<td>0.91</td>
<td>2.69</td>
<td>34</td>
<td>110</td>
<td>0.03107</td>
</tr>
<tr>
<td>eq.</td>
<td>28.55</td>
<td>28.55</td>
<td>257</td>
<td>256</td>
<td>0.03100</td>
</tr>
<tr>
<td>eq.</td>
<td>84.00</td>
<td>84.00</td>
<td>257</td>
<td>512</td>
<td>0.03103</td>
</tr>
<tr>
<td>eq.</td>
<td>0.66</td>
<td>0.66</td>
<td>35</td>
<td>34</td>
<td>0.03180</td>
</tr>
<tr>
<td>eq.</td>
<td>1.08</td>
<td>1.08</td>
<td>35</td>
<td>68</td>
<td>0.03180</td>
</tr>
</tbody>
</table>

Table 3
Example 2: Computational statistics.

In contrast to point-wise evaluation of path constraints on the same mesh as the control profiles. In contrast, the double resolution of the path constraints can prevent oscillations, as it can be seen in Figure 17.

Fig. 16. Example 2: Control profile, wavelet coefficients and profile of \( x_2 \) for the equidistant solution with 256 grid points, path constraint evaluated at 256 points.

Fig. 17. Example 2: Control profile, wavelet coefficients and profile of \( x_2 \) for the equidistant solution with 256 grid points, path constraints evaluated at 512 points.

In alternative to comparing the adaptive solution with very highly resolved
equidistant solutions like 256 intervals in this example, we also present a different way of comparison. The solution at refinement step \( \ell = 7 \) of the adaptive method contains 34 degrees of freedom\(^4\) (see Table 3). Assuming that no a-priori knowledge about the optimal control profile is available, we are interested in an equidistant solution with the same number of degrees of freedom. This is shown in Figure 18, for the case of point-wise path constraint evaluation on the same grid. Note that a mesh of 34 intervals does not fit on the dyadic wavelet grid, therefore a time-scale plot cannot be shown, here. However, the software implementation is able to handle arbitrary user-specified control discretizations beyond the adaptive framework. The profile for \( x_2 \) shows significant violations of the path constraint, leading to large oscillations in the control profile.

Fig. 18. Example 2: Control profile and profile of \( x_2 \) for the equidistant solution with 34 grid points, path constraints evaluated at 34 points.

However, even the solution with an enforcement of the path constraint twice as fine is better but still not satisfactory in this case, as Figure 19 shows. The computation time for this solution is with 1.08 CPU seconds (see Table 3) comparable to the accumulated CPU time after refinement iteration \( \ell = 4 \). The solution quality of this adaptive solution is much higher, as the objective function value and the third row (corresponding to \( \ell = 4 \)) in Figure 14 illustrate, although this solution requires only 25 degrees of freedom.

Fig. 19. Example 2: Control profile and profile of \( x_2 \) for the equidistant solution with 34 grid points, path constraints evaluated at 68 points.

\(^4\) Note that the number of degrees of freedom also corresponds to the number of patches in the time-scale plot.
6.3 Example 3: Nonlinear CSTR problem

This case study is taken from Canto et al. (2002). It considers the optimal operation of an isothermal continuous stirred tank reactor involving a system of five simultaneous chemical reactions. The reactor model contains eight differential equations. The problem formulation which we consider here involves three control variables, which are discretized using piecewise linear functions. There are no state path constraints. The adaptive refinement procedure has been started from equidistant grids with 8 intervals, (i.e. 9 linear degrees of freedom) for each control variable. The settings for the optimization and refinement are identical to those used for Example 1 (see Table 1), except that SNOPT has been employed as the NLP solver.

For lack of space we just present the solution profiles found after convergence of the refinement algorithm, which occurred after $\ell = 4$ iterations. The profiles exhibit a quite complex structure. Figure 20 shows the optimal profiles of the three control variables. The corresponding time-scale plots nicely demonstrate that different parameterization schemes for each control variable evolved during the adaptation procedure.

For reference, we also show the results of an 128-interval equidistant solution in Figure 21. Qualitatively, only slight differences from the adaptive profiles can be recognized.

In Figure 22, the objective function values and the corresponding computation times are plotted. A similar behavior as in Example 1 can be observed.

Finally, Table 4 shows the comparison of the statistics of the adaptive and equidistant solutions. Like in the previous examples we can note that basically the same solution can be achieved after a significantly shorter computation time. As a conclusion we can state that the adaptive algorithm can also be applied to problems with several controls variables in a straightforward manner, leading to separately adapted grids for each one.

As a conclusion, it is important to mention that the adaptive method automatically finds a suitable resolution of the mesh, whereas the resolution of an equidistant mesh must be specified a-priori.

7 Conclusions

In this paper, we presented an adaptive refinement concept for the solution of dynamic optimization problems. The method is embedded into a single-
Fig. 20. Example 3: Solution profiles and wavelet coefficients of the three control variables for $\ell = 4$.

<table>
<thead>
<tr>
<th>$\ell$</th>
<th>CPU time</th>
<th>CPU time</th>
<th>no. of dof</th>
<th>$\Phi^{*\ell}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.30</td>
<td>1.30</td>
<td>27</td>
<td>-20.09313</td>
</tr>
<tr>
<td>1</td>
<td>1.91</td>
<td>3.20</td>
<td>44</td>
<td>-20.09428</td>
</tr>
<tr>
<td>2</td>
<td>3.49</td>
<td>6.69</td>
<td>64</td>
<td>-20.09552</td>
</tr>
<tr>
<td>3</td>
<td>6.71</td>
<td>12.86</td>
<td>96</td>
<td>-20.09572</td>
</tr>
<tr>
<td>4</td>
<td>8.89</td>
<td>21.75</td>
<td>146</td>
<td>-20.09575</td>
</tr>
<tr>
<td>eq.</td>
<td>61.00</td>
<td>61.00</td>
<td>387</td>
<td>-20.09568</td>
</tr>
</tbody>
</table>

Table 4

Example 3: Computational statistics.

Using the shooting dynamic optimization framework. Starting from an initial coarse grid, the parameterization is successively refined based on a wavelet-based analysis of the optimal solution obtained in the preceding optimization step. The adaptive grids are generated automatically. With the help of this approach we can obtain economic and nonuniform grids on which the optimization might
Fig. 21. Example 3: Solution profiles and wavelet coefficients of the three control variables for an equidistant solution with 128 intervals.

Fig. 22. Example 3: Evolution of the objective function value and the computation time.

be evaluated at less computational expense than on a uniform mesh of comparable accuracy. With the help of three case studies with increasing complexity the properties and benefits of the proposed method could be illustrated.

The adaptive refinement method can be especially useful for those problems, where nothing is known about the solution structure a-priori. Obtaining a good solution quality in such cases often involves a manual, time consuming
iterative procedure of solving an optimization problem and subsequent re-
arrangement of the grids. This effort can be significantly reduced or saved
completely by using the adaptive method.

The question arises whether these favorable properties can be expected always. The adaptive method takes advantage of local differences in the time-frequency behavior of the solution. Therefore, significant reductions in computational cost are not likely to be observed in those cases, where the optimal solution does not reveal such local differences.

A special property of the adaptive method is that it can – at least to some extent – adjust the attainable solution accuracy to the available computation time. This can be an attractive feature for on-line applications such as control on moving horizons, where only a limited time for computation of an optimal solution is available, before the horizon is shifted. By means of the adaptive method, a reasonably accurate solution can be obtained after a short computation time. If still some time is available, this solution can be refined until the time is up.

Future extensions of the algorithm can make use of the fact that the coarse solutions do not necessarily need to be solved up to a high optimization tolerance, since they are being refined anyway. Therefore a successive tightening strategy for the objective function tolerance is conceivable.

Acknowledgements

This work has been partially funded by the European Commission under grant G1RD-CT-1999-00146 in the “INCOOP” project. The authors gratefully acknowledge fruitful discussions with A. Cruse, W. Dahmen, J.V. Kadam and J. Oldenburg.

References


