ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

Branch-and-Refine for Mixed-Integer Nonconvex Global Optimization

Sven Leyffer, Annick Sartenaer, and Emilie Wanufelle

Mathematics and Computer Science Division

Preprint ANL/MCS-P1547-0908

October 2, 2008

This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

Contents

Branch-and-Refine for Mixed-Integer Nonconvex Global Optimization

Sven Leyffer*, Annick Sartenaer†, and Emilie Wanufelle‡

September 27, 2008

Abstract

We propose a new global optimization method for solving mixed-integer nonlinear nonconvex optimization problems. The new method relaxes the nonconvex problem with a piecewise linear envelope by using the concept of special ordered sets. The relaxed problem is then successively improved by branch-and-refine, a variant of the branch-and-bound strategy that refines the discretization after branching. We establish convergence to a global optimum under mild assumptions and show that it generates tighter relaxations than other commonly used underestimators. The method is motivated by an application arising from power system analysis and has been developed in a general framework that can be extended to solve a large class of problems.

Keywords: Mixed-integer nonlinear and nonconvex programming, global optimization, piecewise envelopes, branch-and-bound, branch-and-refine.

AMS-MSC 2000: 90C11, 90C26, 90C30, 90C57

1 Introduction

We consider the solution of nonconvex mixed-integer nonlinear programs (MINLPs). The interest in solving nonconvex MINLPs is motivated by the large number of real-world applications that can be modeled as MINLPs, including gas network problems (Martin et al., 2006), nuclear core reload problems (Quist et al., 1998), trim loss minimization problems in the paper industry (Harjunkoski et al., 1998), and power system management. We are particularly interested in an optimal power flow (OPF) problem; see (Wanufelle, 2007, Chapter 2). Such a problem contains discrete variables modeling the ratio of voltage and the activity of the capacitor banks, and nonlinear equations. The equations involve trigonometric functions that arise in the definition of real and reactive power, making the OPF problems highly nonconvex. Problems of this form can be expressed as

$$\left\{\begin{array}{ll} \underset{x,y}{\text{minimize}} & g_0(x,y) \\ \text{subject to} & g_i(x,y) = 0,\ i = 1,\ldots,m \\ & x \in X,\ y \in Y \cap \mathbb{Z}^t,\ (x,y) \in P, \end{array}\right. \tag{P}$$

*Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA, leyffer@mcs.anl.gov

†Department of Mathematics, FUNDP - University of Namur, 5000 Namur, Belgium, annick.sartenaer@fundp.ac.be

‡Department of Mathematics, FUNDP - University of Namur, 5000 Namur, Belgium, emilie.wanufelle@fundp.ac.be

where $g_i : \mathbb{R}^{s+t} \to \mathbb{R},\ i = 0, \ldots, m$ are twice continuously differentiable and may be nonconvex and the polyhedral set P contains all the general linear constraints on the problem. The variables x and y denote the continuous and discrete variables, respectively, and the sets X and Y are defined by

$$X = \{x \in \mathbb{R}^s : l_x \leq x \leq u_x\} \quad \text{and} \quad Y = \left\{y \in \mathbb{R}^t : l_y \leq y \leq u_y\right\}. \tag{1.1}$$

In the remainder, we assume that the bounds l_x, u_x, l_y and u_y are finite. We also note that general nonlinear inequality constraints can be easily included in problem (P); see the end of Section 2 for more details. However, we concentrate here on equality constraints because the OPF problem contains only general nonlinear constraints of this form.

1.1 Solution Techniques for MINLP

Significant progress has been made in the solution of convex MINLPs in the past few decades with the emergence of methods such as outer approximations proposed by Duran and Grossmann (1986) (see also Fletcher and Leyffer (1994)), generalized Benders decomposition (Geoffrion, 1972), and LP/NLP-based branch-and-bound (Quesada and Grossmann, 1992) (see also Leyffer (1993)). Efficient solvers implementing these methods, like Bonmin (Bonami et al., 2005) or FilMINT (Abhishek et al., 2006) are now available. However, the treatment of nonconvex MINLPs still lags behind the progress in convex MINLPs. The combination of nonconvex and discrete variables increases the complexity of these problems. Convergence to a global optimum (or even to a feasible point, if such a point exists) cannot be guaranteed with local optimization methods. To ensure the convergence to a global optimum (or to prove that the problem is infeasible) requires global optimization methods.

There exist two general-purpose solvers for nonconvex MINLPs: BARON, developed by Tawarmalani and Sahinidis (2002), and αBB, developed by Adjiman et al. (1998). Both solvers replace the equality constraints by two inequalities and then construct outer approximations[1] of the inequality constraints. One drawback of using BARON is the fact that the trigonometric functions in our target application have only trivial outer approximations. In fact, BARON cannot be directly applied to problems with trigonometric functions. Bussieck (2004) reports numerical experience with GAMS/BARON on an OPF problem. The trigonometric functions are approximated by polynomials of degree 7 to "fool" BARON into accepting problems with trigonometric expressions. However, BARON is unable to solve this problem in a reasonable amount of time.

In this paper, we develop a new global optimization method that relaxes each function by a *linear* envelope. Our method can be seen as an extension of the special ordered sets (SOS) approximation method introduced by Beale and Tomlin (1970) and recently employed by Martin et al. (2006) to solve a nonconvex mixed integer problem arising in the management of gas networks. Our method is motivated by a nonconvex MINLP problem, but it can also be applied to continuous nonlinear and nonconvex problems.

The remainder of this paper is organized as follows. We first briefly review the SOS approximation method and discuss its disadvantages in the rest of this section. These pitfalls motivate two new techniques. In Section 2, we show that by developing SOS approximations on the computa-

[1] In the present context, the notion of "outer approximation" does not refer to the classical outer approximation methods mentioned above. But it means that the approximation is a valid relaxation for the approached function.

tional graph of the problem functions, we can avoid the exponential complexity of SOS variables. We illustrate how bounds can be propagated and how we exploit common subexpressions to obtain tighter approximations. In Section 3, we develop a new piecewise polyhedral approximation based on SOS approximation that allows us to give a global convergence result. We compare our new approximation to standard outer approximations, and we compute tight errors for a range of functions relevant to OPF problems. In Section 4, we present our branch-and-refine algorithm and establish the global convergence result.

1.2 SOS Approximation of Nonlinear Functions

In the 1970s, when methods for solving linear programs were much more advanced than methods for nonlinear programs, Beale and Tomlin (1970) developed special ordered set approximations to replace nonlinear functions by piecewise linear approximations. Recently, Martin et al. (2006) implemented a similar approach and investigated properties of the resulting polyhedra. Here, we briefly review SOS approximations and motivate the need for our new approach.

An SOS approximation of a function $h(z)$, $z = (x, y)$, defined on an n-dimensional space can be built as follows (see (Tomlin, 1981) and (Martin et al., 2006)): we choose p_q breakpoints in each component z_q, $1 \leq q \leq n$, and create a grid. The total number I_M of breakpoints in this grid is thus equal to

$$I_M = \prod_{q=1}^{n} p_q. \tag{1.2}$$

Let M be the set of breakpoints, referred to as z^k, $k \in I_M$, and let $h^k = h(z^k)$.

The domain $\otimes_{q=1}^{n}[l_{z_q}, u_{z_q}]$ on which $h(z)$ is approximated can be partitioned into S simplices, each of which is defined by at most $n+1$ breakpoints belonging to M. With the previous notations, the SOS approximation of $h(z)$ is given by

$$\tilde{h}(z) = \sum_{k \in I_M} \lambda^k z^k, \tag{1.3}$$

where the SOS variables λ^k associated with the breakpoints z^k satisfy

$$\sum_{k \in I_M} \lambda^k z^k = z, \tag{1.4a}$$

$$\sum_{k \in I_M} \lambda^k = 1, \quad 0 \leq \lambda^k, \quad k \in I_M, \tag{1.4b}$$

$$\text{at most } n+1 \ \lambda^k \text{ are nonzero and they correspond to an } n\text{-dimensional simplex.} \tag{1.4c}$$

Figure 1 illustrates an SOS approximation based on five breakpoints in each dimension for the bilinear product xy on $[-2, 2] \times [-2, 2]$. The decomposition of the domain into simplices is given on the left of the figure, while the associated SOS approximation is represented on the right. Actually, the SOS approximation is linear on each simplex and is given by the plane joining the three breakpoints defining the triangle.

By replacing all nonconvex expressions in (P) by an SOS approximation, we obtain an approximation to (P) in the form of a mixed integer linear program (MILP). The resulting MILP problem can be solved by using special branching tactics adapted to SOS variables.

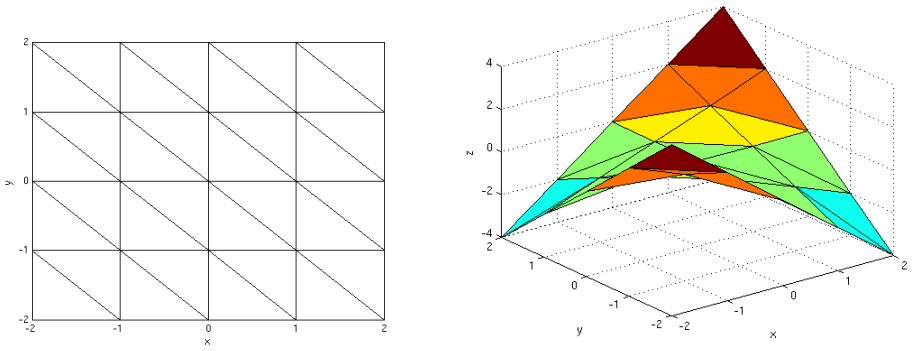

Figure 1: SOS approximation of xy (right) on the partitioned domain (left).

However, the SOS approximation method is not always efficient, and it did not converge on the OPF problem. In a preliminary implementation, the resulting MILP was wrongly declared infeasible, even though the nonlinear problem was feasible. This pitfall illustrated in Example 1.1, motivates the derivation of outer approximations.

Example 1.1 *Consider the following two equality constraints:*

$$c_1(x) \equiv \sin(x) = 0 \tag{1.5}$$

$$c_2(x) \equiv -0.35\ (x-\pi)^2 - 0.3 = 0. \tag{1.6}$$

There are two solutions to $c_1(x) = c_2(x)$ *in the domain* $[0, 2\pi]$*, namely,* $x_1 \simeq 3.492$ *and* $x_2 \simeq 4.541$*. Next, we construct an SOS approximation of* c_1 *and* c_2 *and observe that the feasible domain of this MILP is empty. This situation is illustrated in Figure 2.*

To avoid this adverse situation, we could add breakpoints to the SOS approximations or we could refine the approximations during the branch-and-bound process by adding new breakpoints. However, detecting the places where the approximation must be refined is not trivial. We present an alternative based on piecewise polyhedral envelopes in the next section.

A second disadvantage of the SOS approximation suggested in (Martin et al., 2006; Möller, 2004) is the large number of additional variables λ^k introduced. To approximate a general function $h(z)$ that depends on n variables, we introduce a mesh of size p^n, where p is the number of breakpoints, assumed to be identical in all dimensions. Thus, we need an exponential number, p^n, of new variables. In addition, structural information such as partial separability is not usually exploited by SOS approximations.

In the next two sections we show how these two difficulties can be overcome.

2 Decomposition of Nonlinear Functions

In this section we show how the exponential complexity of SOS approximations can be avoided. It is well known that most functions can be decomposed into unary and binary (one- and two-dimensional) functions. This decomposition is not unique and can be expressed in a computational

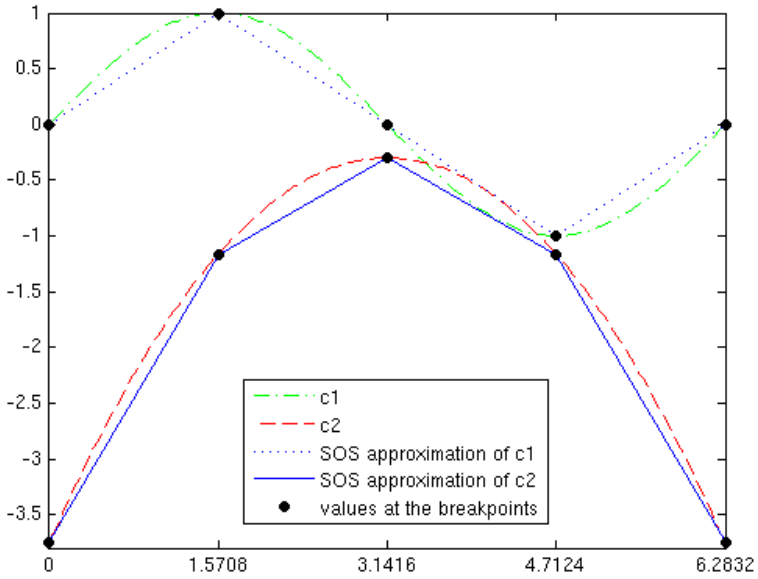

Figure 2: Example illustrating potentially empty feasible region for SOS approximation.

graph. We show how to propagate bounds through the computational graph and how common subexpressions in the graph can be exploited.

Consider a general nonlinear function $h(x, y)$. We introduce intermediate variables w_i as follows:

$$\begin{array}{lcll} w_j & = & x_j & j = 1, \ldots, s, \\ w_{s+j} & = & y_j & j = 1, \ldots, t, \\ w_{s+t+j} & = & h_j(w_{j_1}\{, w_{j_2}\}) & j_1, j_2 < s + t + j,\ j = 1, \ldots, K, \\ h(x, y) & = & w_{s+t+K}, & \end{array} \tag{2.1}$$

where the first $s + t$ assignments of problem to intermediate variables is done solely to simplify the notation. The functions h_j are unary (e.g., $\sin(w)$, in which case the second argument, w_{j_2}, is omitted) or binary functions (e.g., addition, multiplication).

Example 2.1 *Consider the nonlinear function*

$$h(x_1, x_2, x_3) = ax_1^2 + bx_1x_2\cos(x_3), \tag{2.2}$$

where x_j, $1 \le j \le 3$, are variables and a and b are parameters. A possible decomposition of $h(x)$ into unary and binary functions is

$$w_j = x_j, \quad j = 1, \ldots, 3, \tag{2.3a}$$
$$w_4 = w_1^2, \tag{2.3b}$$
$$w_5 = w_1w_2, \tag{2.3c}$$
$$w_6 = \cos(w_3), \tag{2.3d}$$
$$w_7 = w_5w_6, \tag{2.3e}$$
$$w_8 = aw_4 + bw_7 = h. \tag{2.3f}$$

Figure 3 represents the computational graph of the nonlinear function (2.2). In this figure, the bottom nodes correspond to the original variables x_j, while the top node represents the function after decomposition. The other nodes are associated to new variables w_j, allowing us to decompose the function. Each directed edge means that the source node is directly involved in the function represented at the target node. We note that this decomposition is not unique. For example, we could have chosen to use $w_6 = \cos(w_3)$, $w_7 = w_1 w_6$ and $w_8 = w_2 w_7$ instead.

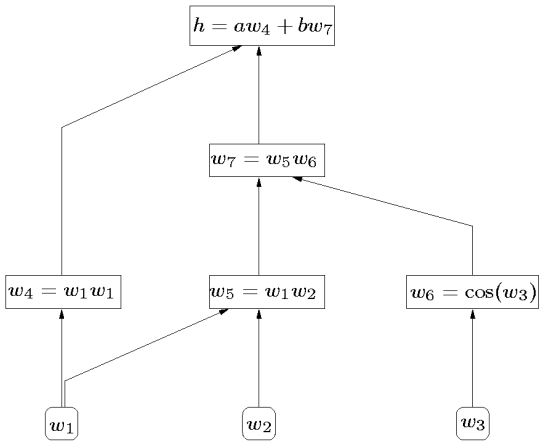

Figure 3: Decomposition of the nonlinear function (2.2).

Each nonlinear constraint (2.3) could now be approximated by using SOS approximations of dimension 1 or 2. By exploiting a decomposition into unary and binary functions, we can create approximations that avoid the exponential growth in the number of variables of standard SOS approximations. For example, a general nonlinear function $h(x)$ that depends on n variables $x_1, \ldots, x_n$ introduces p^n SOS variables. However, by exploiting the decomposition into unary and binary functions, we can approximate the same function with only $k_1 p + k_2 p^2$ additional SOS variables, where k_1 is the number of unary and k_2 is the number of binary functions, respectively. This approach is similar in spirit to automatic differentiation (Griewank, 2000).

Unfortunately, the reduction in the number of SOS variables also reduces the accuracy of the approximation. To see this effect, consider the trilinear function $h(x) = x_1 x_2 x_3$ defined on $[0, 4]^3$, and assume that we use three breakpoints in each dimension. The SOS-4 approximation of $h(x)$ at $x = (2, 2, 1)$ is exact. However, using the decomposition $w_1 = x_1 x_2$ and $h = w_1 x_3$, we can show that the value of the corresponding SO approximation is 0, rather than 4.

We can now reformulate the original nonconvex MINLP, (P), using SOS approximations of the nonlinear components that arise in the decomposition of the problem functions. We associate the intermediate variables with the problem function they represent through subscripts; that is, w_{ij} is the jth variable of the decomposition of $g_i(x, y)$. We denote the unary and binary components of each function $g_i(x, y)$ by $g_{ij}(w_{i,j_1}\{, w_{i,j_2}\})$, where the second argument is absent if g_{ij} is a unary

function. This gives rise to the equivalent MINLP problem:

$$\left\{\begin{array}{lll} \underset{x,y,w}{\text{minimize}} & w_{0,s+t+K_0} & \\ \text{subject to} & w_{ij} = x_j & i = 0,\ldots,m,\ j = 1,\ldots,s \\ & w_{i,s+j} = y_j & i = 0,\ldots,m,\ j = 1,\ldots,t \\ & w_{i,s+t+j} = g_{ij}(w_{i,j_1}\{,w_{i,j_2}\}) & i = 0,\ldots,m,\ j = 1,\ldots,K_i,\ j_1,j_2 < s+t+j \\ & x \in X, y \in Y \cap \mathbb{Z}^t, (x,y) \in P, w \in W, & \end{array}\right. \tag{D}$$

where W represents the bounds on the intermediate variables w_{ij} that are readily derived for a range of nonlinear functions.

Proposition 2.2 *Consider the functions $x^2, \sin(x), \cos(x)$, and xy arising in the OPF problem on the range $[l_x, u_x] \times [l_y, u_y]$. It follows that the following bounds can be derived on the range of these functions:*

x^2

$$\begin{array}{rcl} l_{x^2} & = & \left\{\begin{array}{ll} \min(l_x^2, u_x^2) & \text{if } 0 \notin [l_x, u_x], \\ 0 & \text{if } 0 \in [l_x, u_x], \end{array}\right. \\ u_{x^2} & = & \max(l_x^2, u_x^2), \end{array}$$

xy

$$\begin{array}{rcl} l_{xy} & = & \min(l_x l_y,\ l_x u_y,\ u_x l_y,\ u_x u_y), \\ u_{xy} & = & \max(l_x l_y,\ l_x u_y,\ u_x l_y,\ u_x u_y), \end{array}$$

$\sin(x)$

$$l_{\sin(x)} = \left\{\begin{array}{ll} -1 & \text{if } l_x \le \frac{3\pi}{2} \le u_x, \\ \min(\sin(l_x), \sin(u_x)) & \text{otherwise}, \end{array}\right.$$

$$u_{\sin(x)} = \left\{\begin{array}{ll} 1 & \text{if } l_x \le \frac{\pi}{2} \le u_x, \\ \max(\sin(l_x), \sin(u_x)) & \text{otherwise}, \end{array}\right.$$

$\cos(x)$

$$\begin{array}{rcl} l_{\cos(x)} & = & \left\{\begin{array}{ll} -1 & \text{if } l_x \le \pi \le u_x, \\ \min(\cos(l_x), \cos(u_x)) & \text{otherwise}, \end{array}\right. \\ u_{\cos(x)} & = & \max(\cos(l_x), \cos(u_x)), \end{array}$$

where we have assumed without loss of generality that the domain of trigonometric functions lies in $[0, 2\pi]$ (otherwise, we can simply exploit the periodicity of these functions).

The proof of this proposition can be found in Wanufelle (2007, Section 3.2.2). We can now obtain an SOS approximation problem for (D) by replacing each nonlinear term,

$$w_{i,s+t+j} = g_{ij}(w_{i,j_1}\{,w_{i,j_2}\}), \tag{2.4}$$

by its piecewise-linear SOS approximation,

$$\begin{aligned} w_{i,s+t+j} &= \sum_{k\in I_{ij}} \lambda_{ij}^k g_{ij}(w_{i,j_1}^k\{, w_{i,j_2}^k\}), \\ 1 &= \sum_{k\in I_{ij}} \lambda_{ij}^k, \quad \lambda_{ij}^k \geq 0 \; \forall k \in I_{ij} \\ w_{i,j_1} &= \sum_{k\in I_{ij}} \lambda_{ij}^k w_{i,j_1}^k, \; \{ \text{ and } w_{i,j_2} = \sum_{k\in I_{ij}} \lambda_{ij}^k w_{i,j_2}^k\}, \end{aligned} \tag{2.5}$$

where w_{ij}^k are the breakpoints, I_{ij} is the set of indices that describe the SOS, and λ_{ij}^k are the additional SOS variables. However, the SOS approximation still suffers from the potential infeasibility described in Example 1.1. The next section shows how we can avoid this pitfall by replacing the piecewise linear approximation by a piecewise polyhedral outer approximation.

We finish this section by observing that an additional advantage of the decomposition (D) is the fact that we can exploit common subexpressions. In our implementation, every common subexpression will be present only once in the SOS approximation. In addition, we can exploit the fact that some variables occur in both unary and binary functions to further reduce the number of SOS variables that are required. These simplifications are important because they keep the problem size small and manageable.

3 Outer Approximation with Special Ordered Sets

In this section we show how to avoid the pitfalls of Example 1.1, where an SOS approximation created an infeasible approximation. The basic idea is illustrated in Figure 4. Instead of replacing a function by its SOS approximation, we compute a tight polyhedral envelope on each subinterval. This gives rise to a piecewise polyhedral envelope. We start by building the envelopes, which need the computation of the maximum errors produced by the SOS approximations, and then show how these errors can be determined for the problem functions that appear in the OPF model.

3.1 Piecewise Polyhedral Envelopes

In our method, each nonlinear function appearing in the problem is replaced by a piecewise polyhedral envelope based on SOS. We refer to the resulting problem as the envelope problem. Figure 4 shows the piecewise linear domain based on an SOS approximation of $\sin(x)$ on $[0, 2\pi]$, which is represented by the area inside the continuous lines.

The piecewise envelope is readily derived. For example, consider the unary function $w_h = h(w)$, and let ϵ_{L_k} and ϵ_{U_k} be the maximum overestimation and underestimation error of the SOS approximation on the interval $[w^k, w^{k+1}]$,

$$\epsilon_{L_k} = \max_{w\in[w^k,w^{k+1}],\lambda^k+\lambda^{k+1}=1} \Big(0, \lambda^k h(w^k) + \lambda^{k+1} h(w^{k+1}) - h(w)\Big),$$

and

$$\epsilon_{U_k} = \max_{w\in[w^k,w^{k+1}],\lambda^k+\lambda^{k+1}=1} \Big(0, h(w) - \lambda^k h(w^k) - \lambda^{k+1} h(w^{k+1})\Big),$$

for $\lambda^k, \lambda^{k+1} \geq 0$. Then we can write the piecewise envelope as

$$\Big(\sum_{k=1}^{p} \lambda^k \Big(h(w^k) - \max(\epsilon_{L_{k-1}}, \epsilon_{L_k})\Big)\Big) \leq w_h \leq \Big(\sum_{k=1}^{p} \lambda^k \Big(h(w^k) + \max(\epsilon_{U_{k-1}}, \epsilon_{U_k})\Big)\Big), \tag{3.1}$$

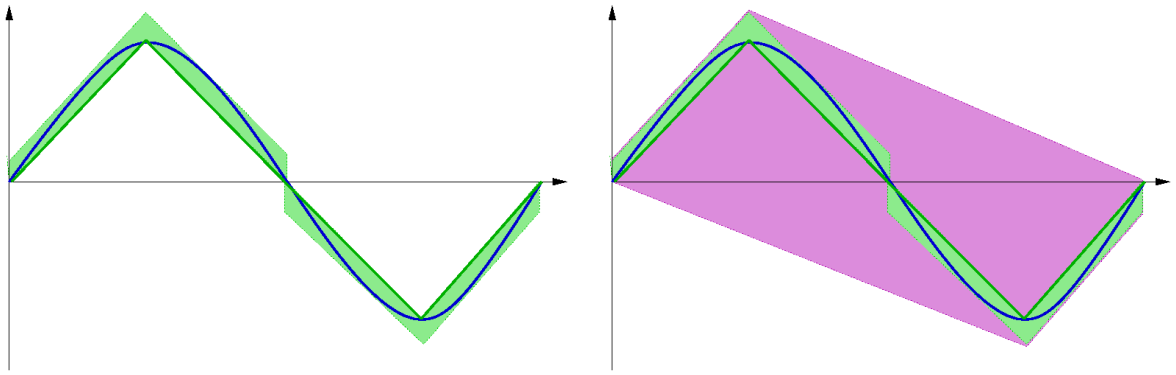

Figure 4: Piecewise polyhedral envelope for $\sin(x)$ on $[0, 2\pi]$ satisfying the SOS condition (left), and convex hull of SOS envelope (right).

where ϵ_{L_0}, ϵ_{L_p}, ϵ_{U_0}, and ϵ_{U_p} are set to zero. This expression can be tightened if we use the lambda method (e.g., (Williams, 2005)), which would require introducing additional binary variables. However, we prefer to avoid the introduction of additional binary variables.

3.2 Computation of the SOS Approximation Errors

In this section, we determine analytical forms for the approximation errors in (3.1). These errors depend on the functions that are approximated, and we compute them here for the functions that occur in the OPF problem, namely, x^2, $\sin(x)$, $\cos(x)$, and the bilinear product xy. Generalizations to other nonconvex functions are straightforward.

Proposition 3.1 *Let $[x^k, x^{k+1}]$ be a piece of the domain $[l_x, u_x]$ where the function x^2 is replaced by its SOS approximation. The maximum overestimation and underestimation approximation errors, ϵ_{x^2,L_k} and ϵ_{x^2,U_k} respectively, generated on this piece are*

$$\epsilon_{x^2,L_k} = \frac{(x^{k+1} - x^k)^2}{4} \quad \text{and} \quad \epsilon_{x^2,U_k} = 0. \tag{3.2}$$

The proof of this proposition can be found in (Wanufelle, 2007, Theorem 3.1).

We can further simplify the lower bound used in (3.1), by observing that for a uniform approximation with a fixed number p of equally spaced breakpoints, we obtain

$$\sum_{k=1}^{p} \lambda^k \max(\epsilon_{x^2,L_{k-1}}, \epsilon_{x^2,L_k}) = \frac{(u_x - l_x)^2}{4(p-1)^2}. \tag{3.3}$$

Indeed, in this case, the overestimation error ϵ_{x^2,L_k} is identical for each piece and corresponds to the right side of (3.3). This can be easily checked by using (1.4b).

While the approximation errors for the square functions depend on the size of the pieces rather than on the domain, the approximation errors for trigonometric functions have a different analytical expression that strongly depends on the domain over which the approximation is done. The following two propositions summarize these results; see (Wanufelle, 2007, Theorems 3.5 and 3.6) for a proof. We assume without loss of generality that the domain belongs to $[0, 2\pi]$.

Proposition 3.2 *Let $[x^k, x^{k+1}] \subseteq [0, 2\pi]$ be a piece of the domain where the function $\sin(x)$ is replaced by its SOS approximation defined by $\tilde{f}(x) = a_k x + b_k$, where*

$$a_k = \frac{\sin x^{k+1} - \sin x^k}{x^{k+1} - x^k} \quad \text{and} \quad b_k = \frac{x^{k+1} \sin x^k - x^k \sin x^{k+1}}{x^{k+1} - x^k}. \tag{3.4}$$

Then, the maximum overestimation and underestimation approximation errors, ϵ_{sin,L_k} and ϵ_{sin,U_k} respectively, generated on this piece are such that

$$\epsilon_{sin,L_k} = \begin{cases} a_k(2\pi - \arccos(a_k)) + b_k & \text{if } 2\pi - \arccos(a_k) \in [x^k, x^{k+1}], \\ \quad + \sin(\arccos(a_k)) & \\ 0 & \text{otherwise,} \end{cases}$$

$$\epsilon_{sin,U_k} = \begin{cases} -a_k \arccos(a_k) - b_k + \sin(\arccos(a_k)) & \text{if } \arccos(a_k) \in [x^k, x^{k+1}], \\ 0 & \text{otherwise.} \end{cases}$$

Proposition 3.3 *Let $[x^k, x^{k+1}] \subset [0, 2\pi]$ be a piece of the domain where the function $\cos(x)$ is replaced by its SOS approximation defined by $\tilde{f}(x) = c_k x + d_k$, where*

$$c_k = \frac{\cos x^{k+1} - \cos x^k}{x^{k+1} - x^k} \quad \text{and} \quad d_k = \frac{x^{k+1} \cos x^k - x^k \cos x^{k+1}}{x^{k+1} - x^k}. \tag{3.5}$$

Then, the maximum overestimation and underestimation approximation errors, ϵ_{cos,L_k} and ϵ_{cos,U_k}, respectively, generated on this piece are such that

$$\epsilon_{cos,L_k} = \begin{cases} c_k(\pi + \arcsin(c_k)) + d_k & \text{if } \pi + \arcsin(c_k) \in [x^k, x^{k+1}], \\ \quad + \cos(\arcsin(c_k)) & \\ 0 & \text{otherwise,} \end{cases}$$

$$\epsilon_{cos,U_k} = \begin{cases} c_k(\arcsin(c_k)) - d_k & \text{if } -\arcsin(c_k) \in [x^k, x^{k+1}], \\ \quad + \cos(\arcsin(c_k)) & \\ -c_k(2\pi - \arcsin(c_k)) - d_k & \text{if } 2\pi - \arcsin(c_k) \in [x^k, x^{k+1}], \\ \quad + \cos(\arcsin(c_k)) & \\ 0 & \text{otherwise.} \end{cases}$$

We now consider the bilinear product. This function has the advantage that the approximation given by constraints (1.3) to (1.4b) defines an envelope for xy as long as this function is approximated on a rectangle, as stated by the following result.

Theorem 3.4 *For every (x, y, xy) such that $l_x \leq x \leq u_x$ $(l_x < u_x)$ and $l_y \leq y \leq u_y$ $(l_y < u_y)$, there exists a unique convex combination of the four points $(l_x, l_y, l_x l_y)$, $(l_x, u_y, l_x u_y)$, $(u_x, l_y, u_x l_y)$ and $(u_x, u_y, u_x u_y)$; in other words, there exist $\lambda_i \geq 0$, $i = 1, \ldots, 4$, such that*

$$\begin{pmatrix} x \\ y \\ xy \end{pmatrix} = \lambda_1 \begin{pmatrix} l_x \\ l_y \\ l_x l_y \end{pmatrix} + \lambda_2 \begin{pmatrix} l_x \\ u_y \\ l_x u_y \end{pmatrix} + \lambda_3 \begin{pmatrix} u_x \\ l_y \\ u_x l_y \end{pmatrix} + \lambda_4 \begin{pmatrix} u_x \\ u_y \\ u_x u_y \end{pmatrix} \tag{3.6}$$

and

$$\sum_{i=1}^{4} \lambda_i = 1 \tag{3.7}$$

hold.

Proof. Let (x, y, xy) be a point such that $l_x \leq x \leq u_x$ ($l_x < u_x$) and $l_y \leq y \leq u_y$ ($l_y < u_y$). As a consequence, x can be expressed as a convex combination of the extreme points of $[l_x, u_x]$; that is, there exists some μ belonging to $[0, 1]$ such that

$$x = (1 - \mu)\ l_x + \mu\ u_x. \tag{3.8}$$

In the same way, there exists some ν belonging to $[0, 1]$ such that

$$y = (1 - \nu)\ l_y + \nu\ u_y. \tag{3.9}$$

By using the expressions of x and y in function of μ and ν, the bilinear product xy can be transformed into

$$xy = (1 - \mu)(1 - \nu)\ l_x l_y + (1 - \mu)\nu\ l_x u_y + \mu(1 - \nu)\ u_x l_y + \mu\nu\ u_x u_y. \tag{3.10}$$

Therefore, the values

$$\begin{aligned}
\lambda_1 &= (1 - \mu)(1 - \nu),\\
\lambda_2 &= (1 - \mu)\nu,\\
\lambda_3 &= \mu(1 - \nu),\\
\lambda_4 &= \mu\nu
\end{aligned}$$

satisfy the conditions (3.6) and (3.7) as well as the positivity condition. Substituting these values in (3.6), we observe that the first three equations of (3.6) are equivalent to (3.8), (3.9), and (3.10), respectively. Equation (3.7) follows by summing the λ_i's and factorizing. The positivity constraint on the λ_i also holds, because μ and ν belong to the interval $[0, 1]$. The convex combination is unique because it can be shown that the matrix

$$\begin{bmatrix}
l_x & l_x & u_x & u_x\\
l_y & u_y & l_y & u_y\\
l_x l_y & l_x u_y & u_x l_y & u_x u_y\\
1 & 1 & 1 & 1
\end{bmatrix} \tag{3.11}$$

is nonsingular. □

We note that (x, y, xy) is required to be written as a *convex combination* of the extreme points, which implies that four equations must be satisfied. Therefore, four points (and not three) are necessary to generate any point (x, y, xy). Furthermore, this theorem is not trivial because a positivity constraint is imposed on the coefficients of a convex combination.

As a direct consequence of this theorem, for each point (x, y), there exists a convex combination of λ^{ij} such that the bilinear product, $h(x) = xy$, equals its linear approximation given by (1.3) to (1.4b). Consequently, we do not have to introduce approximation errors in inequalities (3.1) to guarantee that the correct value of the bilinear product xy at a feasible point (x, y) can be produced, and we can write (3.1) as

$$w_{xy} = \sum_{i=1}^{p_x} \sum_{j=1}^{p_y} \lambda^{ij} x^i y^j. \tag{3.12}$$

We note, however, that we can no longer enforce the standard SOS condition because it would violate the result of Theorem 3.4. Below, we show how to branch efficiently on bilinear terms.

3.3 Comparison with Standard Outer Approximations

We now compare the quality of our outer approximations with standard ones from the literature. We start by considering the McCormick outer approximation w_{xy} of bilinear terms xy; see (McCormick, 1976). For our comparison, we use the concept of the convex hull of a set of points, that is, the smallest convex set containing this set of points. The following result can then be established; see the Appendix for a proof.

Theorem 3.5 *The set of values (x, y, w_{xy}) such that (x, y) belongs to the rectangle $[l_x, u_x] \times [l_y, u_y]$ and (x, y, w_{xy}) satisfies conditions,*

$$w_{xy} = \sum_{i=1}^{p_x} \sum_{j=1}^{p_y} \lambda^{ij} x^i y^j, \quad x = \sum_{i=1}^{p_x} \sum_{j=1}^{p_y} \lambda^{ij} x^i, \quad y = \sum_{i=1}^{p_x} \sum_{j=1}^{p_y} \lambda^{ij} y^j, \tag{3.13a}$$

$$\sum_{i=1}^{p_x} \sum_{j=1}^{p_y} \lambda^{ij} = 1, \quad \lambda^{ij} \geq 0 \ \forall i = 1, \ldots, p_x, j = 1, \ldots, p_y, \tag{3.13b}$$

is equivalent to the set of points that satisfy the McCormick inequalities:

$$(McCormick) \begin{cases} w_{xy} \leq l_x y + u_y x - l_x u_y, \\ w_{xy} \leq u_x y + l_y x - u_x l_y, \\ w_{xy} \geq l_x y + l_y x - l_x l_y, \\ w_{xy} \geq u_x y + u_y x - u_x u_y. \end{cases} \tag{3.14}$$

Moreover, these two sets are equivalent to the convex hull of points (x, y, xy), with $l_x \leq x \leq u_x$ and $l_y \leq y \leq u_y$.

Despite the equivalence of our SOS-based outer approximation (3.13) and the McCormick one, we note that in some cases, the SOS approximations can be made tighter, if at least one argument of the bilinear product appears in another function of the problem. We illustrate this fact with the following NLP:

$$\begin{cases} \min & y^2 + xy + 5x + z, \\ \text{s.t.} & x^2 - z \leq 0, \\ & -x - z \leq -0.75, \\ & -x + y \leq -2, \\ & -1 \leq x \leq 2, \\ & -3 \leq y \leq 0, \\ & 0.25 \leq z \leq 4. \end{cases}$$

The solution of this problem is $(x^*, y^*, z^*) = (0.5, 1.5,\ 0.25)$. We construct an envelope problem (E) with three equally spaced breakpoints in each dimension: $\{x^i\}_{i=1}^3 = \{-1,\ 0.5,\ 2\}$ for x and $\{y^j\}_{j=1}^3 = \{-3,\ -1.5,\ 0\}$ for y, which gives nine variables $\lambda^{i,j}$. As explained in Section 2, the same SOS set λ can be employed to underestimate the square and the bilinear functions. At the solution of the resulting underestimation problem, the components x, y, and z are given by (x^*, y^*, z^*), and the value of the approximation of xy given by $w^*_{xy} = -0.75$ coincides with the exact value of $x^* y^*$. At the solution of the problem with the McCormick underestimators, we obtain the same solution (x^*, y^*, z^*). However, the underestimation of the bilinear product xy is not exact and is given by $w_{xy} = -3$.

Let us now consider the trigonometric functions. Caratzoulas and Floudas (2004) study the underestimators of such functions. They propose convex trigonometric underestimators. The underestimators of $\sin(x)$ and $\cos(x)$ on $[0, 2\pi]$ are respectively given by

$$U_{\sin}(x) = -15.72 \sin\left(\frac{1}{6}(x + 2\pi)\right) + 13.61 \tag{3.15}$$

and

$$U_{\cos}(x) = -16.99 \sin\left(\frac{1}{6}(x + 2\pi)\right) + 15.72. \tag{3.16}$$

Figure 5 compares these underestimators with the SOS-based underestimators for five equally spaced breakpoints. For the cosine function, the SOS-based underestimator dominates the under-

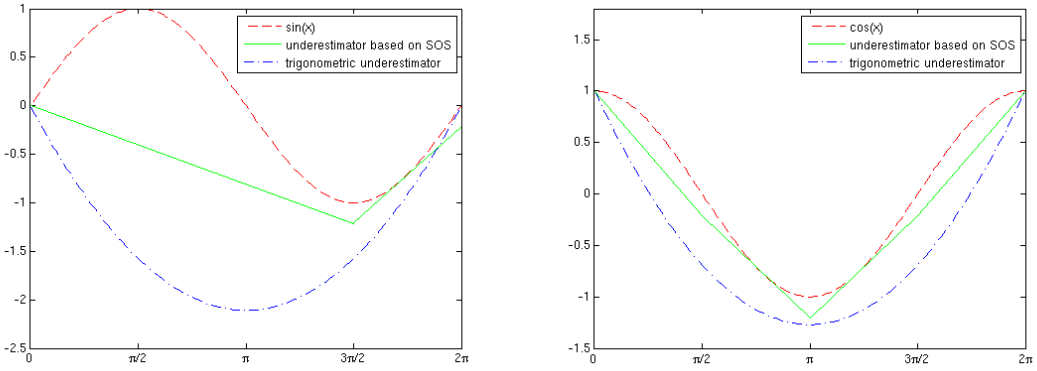

Figure 5: Comparison of SOS-based underestimator of $\sin(x)$ and $\cos(x)$ on $[0, 2\pi]$ and $U_{\sin}(x)$ and $U_{\cos}(x)$.

estimator in (3.16). The same is not true for the sine function near 2π. However, we can show that the area of underestimation is strictly smaller than for the underestimators in (Caratzoulas and Floudas, 2004) (see the Appendix for the proof of the following proposition).

Proposition 3.6 *Let $A^{\sin}_{SOS}$ (respectively $A^{\cos}_{SOS}$) be the area between the sine (respectively the cosine) function defined on $[0, 2\pi]$ and its SOS-based underestimator using five equally spaced breakpoints. Let $A^{\sin}_{trig}$ (respectively $A^{\cos}_{trig}$) be the area between the sine (respectively the cosine) function defined on $[0, 2\pi]$ and its convex trigonometric underestimator given by (3.15) (respectively (3.16)). Then, it follows that*

$$A^{\sin}_{SOS} = 0.451\ A^{\sin}_{trig} \quad \textit{and} \quad A^{\cos}_{SOS} = 0.313\ A^{\cos}_{trig}.$$

Moreover, Caratzoulas and Floudas have shown that the maximum underestimation error grows linearly with the size of the domain. This is not the case with the SOS-based underestimators because the periodicity of trigonometric functions can be exploited with our method. Note that, when the length of the approximation domain is smaller than π, the trigonometric function may be convex on this domain. In this case, the convex trigonometric underestimator is better than that based on SOS because it corresponds to the trigonometric function itself. Moreover, when

the trigonometric function is defined on a domain where it is concave, the underestimator used by Caratzoulas and Floudas is not a trigonometric function but the straight line joining the extreme points of the interval, exactly like the underestimation based on SOS.

3.4 The Piecewise Linear Envelope Problem

We are now able to formulate the piecewise linear envelope problem corresponding to (D) that forms the basis of our branch-and-refine method. We replace each component by its SOS-based envelope given in (3.1):

$$\left\{\begin{array}{lll}
\underset{x,y,w,\lambda}{\text{minimize}} & w_{0,s+t+K_0} & \\
\text{subject to} & w_{ij} = x_j & i = 0,\ldots,m,\ j = 1,\ldots,s \\
& w_{i,s+j} = y_j & i = 0,\ldots,m,\ j = 1,\ldots,t \\
& w_{i,s+t+j} \geq \displaystyle\sum_{k\in I_{ij}} \lambda_{ij}^k \left(g_{ij}(w_{i,j_1}^k\{, w_{i,j_2}^k\}) - L_{ij}^k \right) & j_1, j_2 < s+t+j,\ j = 1,\ldots,K_i, \\
& & i = 0,\ldots,m \\
& w_{i,s+t+j} \leq \displaystyle\sum_{k\in I_{ij}} \lambda_{ij}^k \left(g_{ij}(w_{i,j_1}^k\{, w_{i,j_2}^k\}) + U_{ij}^k \right) & j_1, j_2 < s+t+j,\ j = 1,\ldots,K_i, \\
& & i = 0,\ldots,m \\
& w_{i,s+t+K_i} = 0 & i = 1,\ldots,m \\
& x \in X,\ y \in Y \cap \mathbb{Z}^t,\ (x,y) \in P \text{ and } w \in W, &
\end{array}\right. \tag{E}$$

where we have defined

$$L_{ij}^k := \max(\epsilon_{L_{i,j,k-1}}, \epsilon_{L_{i,j,k}})$$

and

$$U_{ij}^k := \max(\epsilon_{U_{i,j,k-1}}, \epsilon_{U_{i,j,k}})$$

and $\epsilon_{L_{i,j,k-1}}, \epsilon_{L_{i,j,k}}, \epsilon_{U_{i,j,k-1}}$, and $\epsilon_{U_{i,j,k}}$ are the approximation errors that are valid on each piece. The constraint $w \in W$ collects all valid bounds on the variables w_{ij}. Note that when g_{ij} is linear, L_{ij}^k and U_{ij}^k are set to zero $\forall k \in I_{ij}$, and (E) can be simplified since the two associated constraints involving the SOS variables can be replaced by $w_{i,s+t+j} = g_{ij}(w_{i,j_1}\{, w_{i,j_2}\})$. In the case of inequality constraints, we can simplify the problem by removing one side of the corresponding inequalities.

4 A Branch-and-Refine Method

Another important difference from Martin et al. (2006) is that we do *not* branch to satisfy the SOS condition. Instead, we employ a classical branch-and-bound approach and branch on the original problem variables, refining the outer approximations. We believe that this branching choice is superior to SOS branching for three reasons. First, Theorem 3.4 shows that enforcing the SOS conditions invalidates the simple outer approximation property of the convex hull of the SOS set for bilinear functions. Second, refining the outer approximations allows us to adaptively generate tighter outer approximations as we dive down the tree. Third, in early experiments we have observed that branching on the problem variables is more efficient than SOS branching. The results of a careful numerical study are presented in a companion paper (Leyffer et al., 2008).

In our implementation, we branch on the original variables only. Our experience shows that branching on these variables improves the numerical efficiency than branching on the intermediate variables w_{ij}. Moreover, choosing the branching variable has a cost (see (Leyffer et al., 2008)), which is increasing with the number of possible candidates for branching. This branching methodology allows us to fix the number of SOS variables in our envelope problems, thereby simplifying the Jacobian storage. As a consequence, we adaptively refine the envelopes in places where the approximations are not adequate. We refer to our method as branch-and-refine. The key to its convergence is the fact that the envelopes are refined as we move down the tree.

4.1 Branching, Subproblems, and Fathoming Rules

To describe our branch-and-refine method, we define the subproblems that are solved during the treesearch. We start by solving the continuous linear programming (LP) relaxation of the envelope problem (E). If this problem is infeasible, then we conclude that the original problem is infeasible. Otherwise, we determine a branching variable and create two new problems that are placed on a stack. The algorithm continues to remove and solve problems on the stack, adding new problems by branching until the stack is empty.

In our method, we have the choice of branching on a continuous variable, x_i; tightening the piecewise envelope; or branching on a nonintegral integer variable, y_i, to improve integrality. We set $X_1 := X$ and $Y_1 := Y$ and branch on a nonintegral value y'_i by defining new variables ranges (we use the convention that left/down branches are indexed by even integers, and right/up branches by odd integers):

$$\begin{array}{l} Y_{2k} = \{y \in Y_k : y_i \leq \lfloor y'_i \rfloor\}, \text{ and } X_{2k} = X_k \quad \text{for the left branch,} \\ Y_{2k+1} = \{y \in Y_k : y_i \geq \lfloor y'_i + 1 \rfloor\}, \text{ and } X_{2k+1} = X_k \quad \text{for the right branch.} \end{array} \tag{4.1}$$

Similarly, we can branch on continuous variables x_i at a value x'_i:

$$\begin{array}{l} X_{2k} = \{x \in X_k : x_i \leq x'_i\}, \text{ and } Y_{2k} = Y_k \quad \text{for the left branch,} \\ X_{2k+1} = \{x \in X_k : x_i \geq x'_i\}, \text{ and } Y_{2k+1} = Y_k \quad \text{for the right branch.} \end{array} \tag{4.2}$$

Each time we branch, we also refine the envelopes. The constraint matrix is then updated. Thus, in practice, more than one bound shifts, and we update the bounds in W. Branching on the continuous variables rearranges the breakpoints, as indicated in Figure 6.

At every node of the branch-and-bound tree, we solve a continuous LP relaxation of the envelope problem (E), that is, the problem obtained by dropping the integrality conditions. This relaxation problem is denoted $\mathrm{LP}(X_k, Y_k)$, where X_k and Y_k are the current feasible subspaces. Its optimum value is referred to as z_{LP_k}, while $(\bar{x}^k, \bar{y}^k)$ denote its optimum solution.

We also define the NLP with fixed integer variables $y = \bar{y}^k$ and range X_k, as

$$\left\{ \begin{array}{lll} z_{\mathrm{NLP}_k} := \underset{x}{\text{minimize}} & g_0(x, \bar{y}^k) & \\ \text{subject to} & g_i(x, \bar{y}^k) = 0, \ i = 1, \ldots, m & \\ & x \in X_k, \ (x, \bar{y}^k) \in P. & \end{array} \right. \tag{NLP$(\bar{y}^k, X_k)$}$$

After solving $LP(X_k, Y_k)$ we can fathom nodes in the branch-and-bound tree if one of the following conditions holds.

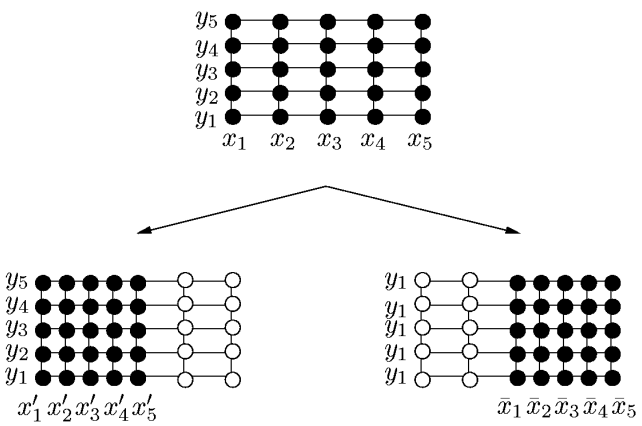

Figure 6: Update of breakpoints after branching on x at x_3 (2 dimensions). The white dots represent the discarded breakpoints.

F1 If the LP relaxation is infeasible, then we can conclude that there are no feasible points in $X_k \times Y_k$, and we can fathom this node.

F2 If the LP relaxation produces an integral solution $\bar{y}^k \in \mathbb{Z}^t$, *and* if the solution of (NLP($\bar{y}^k, X_k$)) satisfies $|z_{\text{NLP}_k} - z_{\text{LP}_k}| \leq \epsilon$, then we can fathom this node. In this case, either we have a new incumbent, or the node is dominated by a better solution.

F3 If the optimum value of the LP relaxation satisfies $|U - z_{\text{LP}_k}| \leq \epsilon$, where U is the optimal value of the incumbent, then no better solution can be found in $X_k \times Y_k$, and we can fathom this node.

We note that integrality of the integer variables alone is not sufficient to fathom a node: We also have to ensure that the envelope of the nonlinear functions is sufficiently close to the nonlinear functions before fathoming this node.

4.2 Branch-and-Refine Algorithm

We are now in a position to formally state in Algorithm 7 our branch-and-refine method. There are many details that we do not specify, such as the branching and node selection rules. See our companion paper (Leyffer et al., 2008) for details.

The algorithm differs in one important aspect from the usual MINLP branch-and-bound approach: Even if all integer variables are integral, we may need to branch in order to ensure that the envelopes are sufficiently close to the nonlinear expressions. There are different ways in which we can determine a branching variable after solving the LP relaxation (LP(X_k, Y_k)). We can solve the NLP relaxation of (P) for the subtree corresponding to $X_k \times Y_k$. This NLP solution provides an upper bound for the subtree, and we can find a nonlinear expression on which to branch by comparing the solutions of the NLP relaxation and (LP(X_k, Y_k)). A cheaper alternative is to simply evaluate the nonlinear expressions at the LP solution $(\bar{x}^k, \bar{y}^k)$ and to pick a nonlinear expression to branch on.

Initialize the upper bound to $U = \infty$, and choose a tolerance $\epsilon > 0$.
Set $k = 1$ and place the LP relaxation ($\mathrm{LP}(X_k, Y_k)$) on the stack.

while *stack is not empty* **do**
  Remove an LP relaxation, ($\mathrm{LP}(X_k, Y_k)$), from the stack and solve it.
  Let the solution be $(\bar{x}^k, \bar{y}^k)$.
  if *(LP(X_k, Y_k)) infeasible or* $z_{LP_k} \geq U - \epsilon$ **then**
    Fathom this node.

  else
    Set `branch = true`.
    if $\bar{y}^k$ *integer feasible* **then**
      Solve NLP ($\mathrm{NLP}(\bar{y}^k, X_k)$) and let the solution be $(\hat{x}^k, \bar{y}^k)$.
      if $z_{NLP_k} < U - \epsilon$ **then**
        Update the upper bound $U := z_{\mathrm{NLP}_k}$.
        New incumbent: $(x^*, y^*) := (\hat{x}^k, \bar{y}^k)$.
        if $|z_{NLP_k} - z_{LP_k}| \leq \epsilon$ **then**
          Fathom this node and every node i of the stack for which $z_{\mathrm{LP}_i} \geq U - \epsilon$, set `branch = false`.

    if `branch == true` **then**
      Find a branching variable, and add two new LP relaxations to the stack:

      1. Use (4.1) to branch on a non-integral integer, y_i.
      2. Use (4.2) to branch on a continuous variable, x_i, which intervenes in an envelope not sufficiently tight.

      Refine the problems by adding new breakpoints.
      Propagate the bounds through the expression tree to tighten bounds on all variables for both new problems.

  Set k:=k+1

Figure 7: Algorithm: branch-and-refine method

4.3 Convergence Proof

Theorem 4.1 *If the discrete variables and those appearing nonlinearly in the nonlinear problem (P) are bounded, Algorithm 7 converges to a global optimum of problem (P), within the accuracy ϵ, in a finite number of iterations.*

Proof. This theorem is shown in two steps. We start by proving that we cannot fathom the optimal solution without saving it, and then we show that the algorithm is finite.

Every relaxation (LP(X_k, Y_k)) of the envelope problem is an LP, and hence its solution is a global solution and provides a lower bound on the nonlinear problem (P) on $X_k \times Y_k$. Moreover, the optimum of (NLP$(\bar{y}^k, X_k)$) provides an upper bound on the solution of (P). Thus it follows that the three fathoming rules F1, F2, and F3 in Algorithm 7 are valid.

Consequently, a node is fathomed only when the associated domain is guaranteed not to contain a better optimum than the incumbent. It thus remains to prove that we cannot branch infinitely often. We observe that Y is bounded, which implies that the integer branches are finite. By branching on the continuous variables, we refine the envelope on that branch, because each LP retains the same number of SOS breakpoints. The differentiability of g_i implies that the approximation error of the envelope problem must become less than ϵ as we continue to branch. At this point, the envelope problems will be sufficiently close to the nonlinear problem such that one of the three conditions F1, F2, or F3 is satisfied.
□

4.4 Comparison with the Classical SOS-Branching

Figure 8 illustrates the advantages of refining the approximation compared to branching on the SOS condition. We consider the function x^2 on the interval $[-2, 2]$. The center plot shows the envelope before branching for three equally spaced breakpoints. We branch on $x = 0$ and obtain two symmetric problems. The left plot shows the refined problem on $[0, 2]$ after branching, and the right plot shows the convex hull of the SOS problem on the same domain after branching. We observe that refinement produces a tighter envelope.

The result of Figure 8 can be made more precise by comparing SOS branching and refinement about the same point. SOS branching on λ_j creates two problems, with

$$\sum_{i=1}^{j} \lambda^i = 1 \quad \text{and} \quad \sum_{i=j}^{p} \lambda^i = 1$$

for the left and right child node, respectively. The corresponding branching bounds for the variable x are given by

$$x \leq x^j \quad \text{and} \quad x \geq x^j.$$

The next proposition, whose proof is given in the Appendix, shows that refinement always creates a tighter outer approximation than does SOS branching.

Proposition 4.2 *Let Δ_x be the range of the variable x before branching and p be the number of equally spaced breakpoints. Then, the sum of the area of the envelopes for x^2 for the left and right*

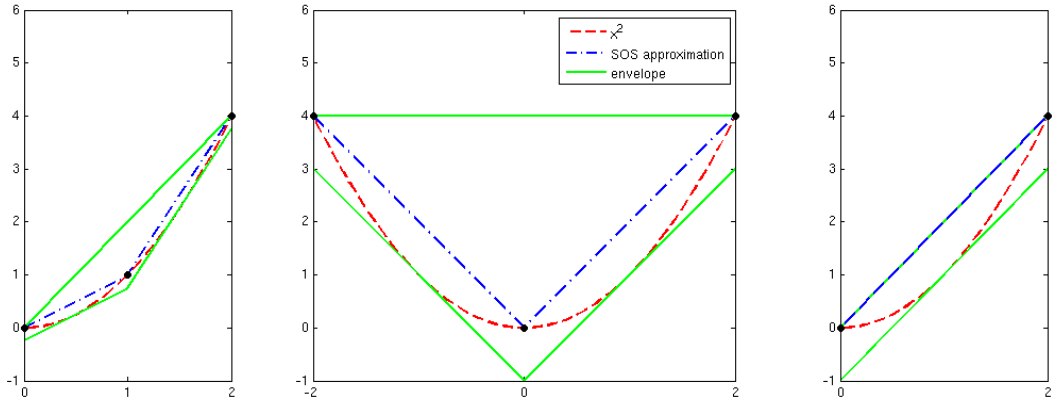

Figure 8: Refinement of the domain of the possible values (x, w_{x^2}) for the approximation of x^2 when $x \in [2, 2]$. Center: before any branching; left: after branching on x; right: after branching on λ.

subproblems obtained by branching on the variable λ^j without updating the breakpoints is given by

$$A_\lambda = \frac{\Delta_x^3}{12} \frac{1}{(p-1)^2} \left(1 + 2\left((p-1)^2 - 3(p-j)(j-1)\right)\right),$$

while that obtained by branching on the variable x at the jth breakpoint and by keeping a constant number of equally spaced breakpoints is given by

$$A_x = \frac{\Delta_x^3}{12} \frac{2p^2 - 4p + 3}{(p-1)^4} \left((p-1)^2 - 3(p-j)(j-1)\right).$$

The area associated to the branching on the variable λ^j is always larger, and the ratio between the two area is given by

$$\frac{A_\lambda}{A_x} = \frac{(p-1)^2\left(1 + 2\left((p-1)^2 - 3(p-j)(j-1)\right)\right)}{(2p^2 - 4p + 3)\left((p-1)^2 - 3(p-j)(j-1)\right)} > 1.$$

Moreover, the best choice to minimize the sum of the areas of the envelopes for the left and right subproblems consists in branching at the midpoint of the range of the variable x.

To see this proposition, assume that we use five equally spaced breakpoints and that we branch at the third one. Then it follows that

$$A_\lambda = \frac{3}{64}, \quad A_x = \frac{11}{256} \quad \text{and} \quad \frac{A_\lambda}{A_x} = \frac{12}{11}.$$

Next, we consider the bilinear function. There exist several ways to branch on the variables λ^{ij}: vertically, horizontally, or diagonally. Therefore, comparisons, which will be based on the *volume* of the envelopes, must be performed for these three branching techniques. Branching vertically and horizontally amounts to reduce the ranges of x and y respectively. However, the best results one can expect with such branching techniques is to isolate a rectangle of the grid instead of a triangle. In order to satisfy the SOS condition, a diagonal branching is used. When vertical or

horizontal branching is performed by branching and refining, or by branching on the SOS variables without modifying the breakpoints, we obtain exactly the same four extreme breakpoints and thus, by Theorem 3.4, the same envelopes; for more details on the proof, see (Wanufelle, 2007, Theorem 4.10).

Assume now that after having branched several times, it is no longer possible to branch vertically or horizontally on the SOS variables λ^{ij} and that the SOS condition is not fulfilled. A diagonal branching thus must be performed. Branching in this way amounts to approach the bilinear product by a plane, its SOS approximation, which violates the property of outer approximation (3.12). Practically, the errors in (3.1) become nonzero. To avoid this situation, when we branch on the original variables, we continue to branch vertically or horizontally.

Because branching on the variables λ^{ij} or on the original variables does not divide the approximation domain in the same way (see Figure 9), the resulting envelopes are different. It has been

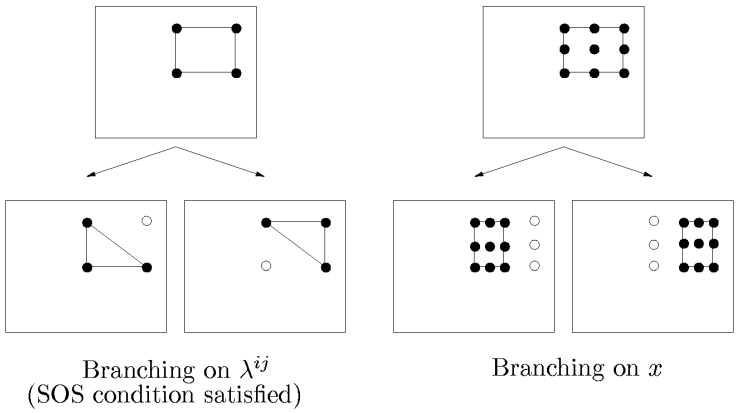

Figure 9: Breakpoints for a diagonal branching phase if we branch on λ^{ij} and for a vertical branching phase if we branch on x (three breakpoints used in each dimension for the first envelope problem).

shown in (Wanufelle, 2007, Section 4.2.2) that the volume of the envelope obtained by branching on λ^{ij} to satisfy the SOS condition is identical for the two subproblems and is three times larger than the one obtained by branching vertically or horizontally at the midpoint of the range of x or y. Branching on the original variables by updating breakpoints can thus lead to much tighter envelopes.

5 Conclusion

In this paper, we have presented a new global optimization method to solve a class of mixed-integer nonlinear and nonconvex problems arising in the management of electrical networks. The proposed method can be seen as an extension of the classical SOS approximation method. In contrast to the latter, our method builds a linear envelope problem and is able to refine it as much as necessary, which is the key for the convergence to a global optimum. In this context, branching is no longer performed on the SOS variables but on the original variables. Moreover, our method starts by decomposing each nonlinear function of the problem into unary and bilinear functions.

This process allows us to reduce the size of the problem and to develop a general framework that can be used to solve a larger class of problems. We have shown that our approach creates tighter relaxations than do standard underestimators.

Several open questions remain. On concerns the positioning of the breakpoints or the position of the branch in the range of the continuous variables. Adding cuts to the method is a way to possibly increase its speed of convergence. Moreover, other classes of envelopes such as the McCormick inequalities can be readily included in our framework.

A Proofs of Comparison to Other Envelopes

Proof of Theorem 3.5 To demonstrate this theorem, we show that both sets (x, y, w_{xy}) are equivalent to the convex hull of points (x, y, xy), with $(x, y) \in [l_x, u_x] \times [l_y, u_y]$. We first recall that the convex hull of a set of points corresponds to the set of convex combinations of these points; see (Hiriart-Urruty and Lemaréchal, 2001) for instance. We define C_{env} to be the set of points (x, y, w_{xy}) such that (x, y) belongs to $[l_x, u_x] \times [l_y, u_y]$ and (x, y, w_{xy}) satisfies conditions (3.13), and we prove that C_{env} is the set of convex combinations of points (x, y, xy), which we denote C_{hull}. $C_{env} \subseteq C_{hull}$, is easily checked. Indeed, by definition, each point of C_{env} is a convex combination of points $(x^i, y^j, x^i y^j)$ where (x^i, y^j) belongs to $[l_x, u_x] \times [l_y, u_y]$. Next, we prove that $C_{hull} \subseteq C_{env}$. By Theorem 3.4, any point (x, y, xy) such that $l_x \leq x \leq u_x$ and $l_y \leq y \leq u_y$ can be written as a convex combination of the four extreme points $(l_x, l_y, l_x l_y)$, $(l_x, u_y, l_x u_y)$, $(u_x, l_y, u_x l_y)$, and $(u_x, u_y, u_x u_y)$, where (l_x, l_y), (l_x, u_y), (u_x, l_y), and (u_x, u_y) are breakpoints. Accordingly, any point of C_{hull}, a convex combination of points (x, y, xy), can be expressed as a convex combination of the four extreme points given above and thus belongs to C_{env}.

Having shown that the points satisfying conditions (3.13) define the convex hull of points (x, y, xy), let us now consider the envelope generated by the McCormick inequalities (3.14). Al-Khayyal and Falk (1983) show that the convex and concave envelopes of xy defined on a rectangle $[l_x, u_x] \times [l_y, u_y]$ given by McCormick inequalities,

$$C_{vex}(x, y) = \max\{l_x y + l_y x - l_x l_y,\ u_x y + u_y x - u_x u_y\}, \tag{A.1}$$

$$C_{cave}(x, y) = \min\{l_x y + u_y x - l_x u_y,\ u_x y + l_y x - u_x l_y\}. \tag{A.2}$$

Using this result, one can show that the set of points (x, y, w_{xy}) satisfying McCormick inequalities corresponds to the convex hull of points (x, y, xy) and, as a consequence, to the set C_{env}. □

Proof of Proposition 3.6 We first consider the underestimators of $\sin(x)$. On $[0, 2\pi]$, the area $A^{\sin}_{trig}$ between the sine function and its convex trigonometric underestimator defined in (3.15) can be computed as follows:

$$\begin{aligned} A^{\sin}_{trig} &= \int_0^{2\pi} \left(\sin(x) + 15.72 \sin\left(\frac{1}{6}(x + 2\pi)\right) - 13.61 \right) dx \\ &= 6\ .\ 15.72\ \left[-\cos\left(\frac{1}{6}(x + 2\pi)\right) \right]_0^{2\pi} - 13.61\ .\ 2\pi \\ &= 94.32\big(-\cos(\tfrac{2\pi}{3}) + \cos(\tfrac{\pi}{3}) \big) - 27.22\pi \\ &\simeq 8.806. \end{aligned} \tag{A.3}$$

We now determine the area $A_{SOS}^{\sin}$ between the sine function and its SOS-based underestimator using five equally spaced breakpoints, which is represented on the right part of Figure 10. To build this underestimator, we start from the SOS approximation of $\sin(x)$. As shown on the left part of Figure 10, this approximation produces overestimation errors for the third and fourth pieces, that is, on $[\pi, \frac{3\pi}{2}]$ and $[\frac{3\pi}{2}, 2\pi]$. Applying Proposition 3.2 to these two pieces, we find that the overestimation errors are identical on both pieces and are equal to 0.210. To obtain an outer approximation, we remove this error from the SOS approximation as required by definition (3.1) and obtain the underestimator referred to as SOS underestimator on Figure 10. However, this representation implies that the SOS condition is satisfied, which is not necessarily true. As a consequence, this underestimator must be relaxed to allow any convex combination between points belonging to the SOS underestimator. We finally obtain the SOS-based underestimator shown on the right of Figure 10. This underestimator is thus defined on $[0, \frac{3\pi}{2}]$ by the straight line joining $(0, 0)$ to $(\frac{3\pi}{2}, -1.210)$, that is,

$$u(x) = -0.257x, \tag{A.4}$$

and on $[\frac{3\pi}{2}, 2\pi]$ by the straight line joining $(\frac{3\pi}{2}, -1.210)$ to $(2\pi, -0.210)$, that is,

$$u(x) = \frac{2}{\pi}x - 4.210. \tag{A.5}$$

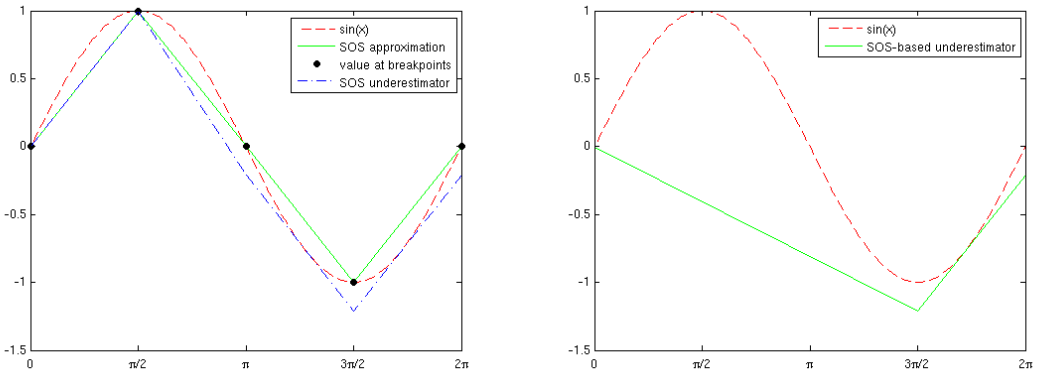

Figure 10: Construction of the SOS-based underestimator for $\sin(x)$ on $[0, 2\pi]$ using five equally spaced breakpoints.

The area $A_{SOS}^{\sin}$ is thus equal to

$$\begin{aligned} A_{SOS}^{\sin} &= \int_0^{2\pi} \sin(x)\, dx - \int_0^{\frac{3\pi}{2}} -0.257x\, dx - \int_{\frac{3\pi}{2}}^{2\pi} \frac{2}{\pi}(x - 4.210)\, dx \\ &= 0.257 \left[\frac{x^2}{2}\right]_0^{\frac{3\pi}{2}} - \left[\frac{2}{\pi}\frac{x^2}{2} - 4.210x\right]_{\frac{3\pi}{2}}^{2\pi} \\ &= 0.257\, \frac{9\pi^2}{8} - 4\pi + 2.105\pi + \frac{9\pi}{4} \\ &\simeq 3.969. \end{aligned} \tag{A.6}$$

The ratio between $A^{\sin}_{SOS}$ and $A^{\sin}_{trig}$ is thus given by

$$\frac{A^{\sin}_{SOS}}{A^{\sin}_{trig}} = 0.451.$$

Let us next consider the cosine function. On $[0, 2\pi]$, the area $A^{\cos}_{trig}$ between this function and its convex trigonometric underestimator defined in (3.16) is such that:

$$\begin{array}{rcl} A^{\cos}_{trig} & = & \displaystyle\int_0^{2\pi} \left(\cos(x) + 16.99 \sin\left(\frac{1}{6}(x+2\pi)\right) - 15.72\right) dx \\ & = & 6 \;.\; 16.99 \;\left[-\cos\left(\frac{1}{6}(x+2\pi)\right)\right]_0^{2\pi} - 15.72 \;.\; 2\pi \\ & = & 101.94\big(-\cos(\frac{2\pi}{3}) + \cos(\frac{\pi}{3})\big) - 31.44\pi \\ & \simeq & 3.168. \end{array} \tag{A.7}$$

To establish the expression of the SOS-based underestimator of $\cos(x)$ on $[0, 2\pi]$, we use the same scheme as for $\sin(x)$. We build its SOS approximation based on five equally breakpoints and then determine the pieces where an overestimation error arises, which are the second and third pieces; see the left of Figure 11. The overestimation errors on these pieces are again identical and equal to 0.210, by Proposition 3.3. Removing these errors from the SOS approximation, we obtain the SOS underestimator represented in Figure 11. Since this underestimator is convex, it corresponds to our SOS-based underestimator $u(x)$. It is thus defined by a different expression on each piece, that is,

$$u(x) = \begin{cases} \text{the straight line joining } (0,1) \text{ to } (\frac{\pi}{2}, -0.210) & \text{on } [0, \frac{\pi}{2}], \\ \text{the straight line joining } (\frac{\pi}{2}, -0.210) \text{ to } (\pi, -1.210) & \text{on } [\frac{\pi}{2}, \pi], \\ \text{the straight line joining } (\pi, -1.210) \text{ to } (\frac{3\pi}{2}, -0.210) & \text{on } [\pi, \frac{3\pi}{2}], \\ \text{the straight line joining } (\frac{3\pi}{2}, -0.210) \text{ to } (2\pi, 1) & \text{on } [\frac{3\pi}{2}, 2\pi]. \end{cases}$$

More precisely, we have

$$u(x) = \begin{cases} -\frac{2.421}{\pi}x + 1 & \text{on } [0, \frac{\pi}{2}], \\ -\frac{2}{\pi}x + 0.789 & \text{on } [\frac{\pi}{2}, \pi], \\ \frac{2}{\pi}x - 3.210 & \text{on } [\pi, \frac{3\pi}{2}], \\ \frac{2.421}{\pi}x - 3.842 & \text{on } [\frac{3\pi}{2}, 2\pi]. \end{cases} \tag{A.8}$$

Using (A.8), we compute the area $A^{\cos}_{SOS}$ as

$$\begin{array}{rcl} A^{\cos}_{SOS} & = & \displaystyle\int_0^{2\pi} \cos(x)\, dx - \int_0^{\frac{\pi}{2}} \left(-\frac{2.421}{\pi}x + 1\right) dx - \int_{\frac{\pi}{2}}^{\pi} \left(-\frac{2}{\pi}x + 0.789\right) dx \\ & & \displaystyle - \int_{\pi}^{\frac{3\pi}{2}} \left(\frac{2}{\pi}x - 3.210\right) dx - \int_{\frac{3\pi}{2}}^{2\pi} \left(\frac{2.421}{\pi}x - 3.842\right) dx \\ & = & \displaystyle\frac{2.421}{\pi}\left[\frac{x^2}{2}\right]_0^{\frac{\pi}{2}} + \left[\frac{x^2}{\pi}\right]_{\frac{\pi}{2}}^{\pi} + \left[\frac{-x^2}{\pi}\right]_{\pi}^{\frac{3\pi}{2}} + \frac{2.421}{\pi}\left[\frac{-x^2}{2}\right]_{\frac{3\pi}{2}}^{2\pi} \\ & & + \frac{\pi}{2}\,(-1 - 0.789 + 3.210 + 3.842) \\ & = & \displaystyle -\frac{1.205}{4}\frac{6\pi}{4} + 2\pi - \frac{10\pi}{4} + \frac{\pi}{2}5.263 \\ & \simeq & 0.992. \end{array} \tag{A.9}$$

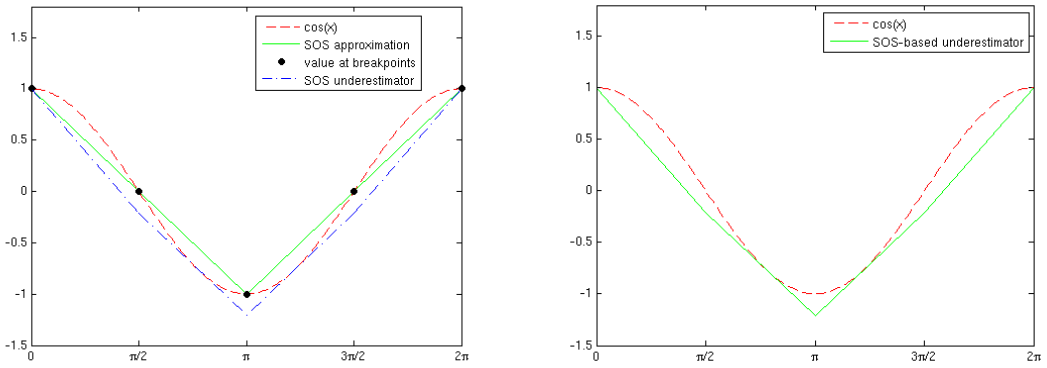

Figure 11: Construction of the SOS-based underestimator for $\cos(x)$ on $[0, 2\pi]$ using five equally spaced breakpoints.

We conclude this proof by computing the ratio between $A^{\cos}_{SOS}$ and $A^{\cos}_{trig}$, which is equal to

$$\frac{A^{\cos}_{SOS}}{A^{\cos}_{trig}} \simeq 0.313.$$

□

Proof of Proposition 4.2 We begin by computing the area A_x associated to the branching on the variable x at x^j. Figure 12 represents this kind of branching: on the left the envelope before branching and on the right after branching. On the right plot, the vertical line shows the branching place. The envelope on the left of this line thus corresponds to the left subproblem, while the one on the right is associated to the right subproblem.

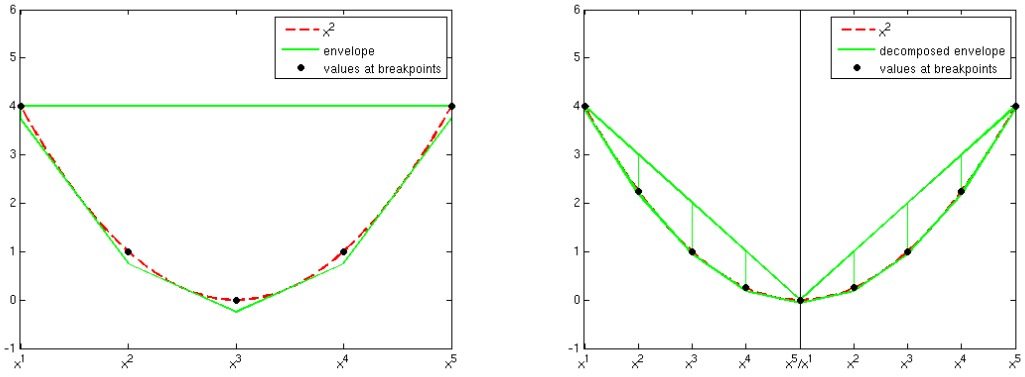

Figure 12: Left: envelope before branching; right: decomposition into 2 trapeziums of each envelope generated by branching on x^3 and refining (5 breakpoints used).

As represented in Figure 12, the feasible area for each subproblem generated by branching can be decomposed into $p-1$ trapeziums, where the number of trapeziums corresponds to the number

of pieces. The "height," H, of a trapezium is equal to the length of a piece, namely, $\frac{\Delta_{x'}}{p-1}$, where $\Delta_{x'}$ denotes the length of the approximation domain after branching. The length of the bases, the parallel sides defined in $x = x^i$, $1 \leq i \leq p$, and denoted B_i, is equal to the sum of the errors $\epsilon_{x^2,L}$ removed in (3.1) and $\epsilon_{x^2,O}(x^i)$, the maximum overestimation approximation error produced at x^i. For the sake of simplicity, we use another notation for $\epsilon_{x^2,L}$, namely, ϵ_{x^2,Δ_x} to highlight the length of the domain on which the square function is approximated. By using (3.3), it can be shown that the errors ϵ_{x^2,Δ_x} satisfy

$$\epsilon_{x^2,\Delta_{x'}} = \frac{\Delta_{x'}^2}{4(p-1)^2}. \tag{A.10}$$

The maximum overestimation approximation error that can be produced at x^i, $\epsilon_{x^2,O}(x^i)$, is equal to the difference at point x^i between the straight line joining $(x^1,(x^1)^2)$ to $(x^p,(x^p)^2)$ and the square function, that is,

$$\begin{aligned} \epsilon_{x^2,O}(x^i) &= (x^1+x^p)x^i - x^1x^p - (x^i)^2, \\ &= (x^p - x^i)(x^i - x^1). \end{aligned}$$

By using the fact that the breakpoints are equally spaced, we write $(x^p - x^i)$ and $(x^i - x^1)$ as functions of $\Delta_{x'}$. Then $\epsilon_{x^2,O}(x^i)$ becomes

$$\epsilon_{x^2,O}(x^i) = \frac{\Delta_{x'}^2}{(p-1)^2}\ (p-i)(i-1)\cdot \tag{A.11}$$

Because the area of an envelope obtained by branching on x, denoted A_{sub}, is equal to the sum of the areas of $p-1$ trapeziums, we have

$$A_{\text{sub}} = \sum_{i=1}^{p-1} \frac{1}{2} H(B_i + B_{i+1}) = \sum_{i=1}^{p-1} \frac{1}{2} \frac{\Delta_{x'}}{p-1}\ (2\epsilon_{x^2,\Delta_{x'}} + \epsilon_{x^2,O}(x^i) + \epsilon_{x^2,O}(x^{i+1})).$$

which simplifies to

$$A_{\text{sub}} = \frac{1}{2}\frac{\Delta_{x'}}{p-1}\ \left(2\sum_{i=1}^{p-1}\epsilon_{x^2,\Delta_{x'}} + 2\sum_{i=2}^{p-1}\epsilon_{x^2,O}(x^i)\right),$$

because$\epsilon_{x^2,O}(x^i)$ is equal to zero at points x^1 and x^p. Replacing $\epsilon_{x^2,\Delta_{x'}}$ and $\epsilon_{x^2,O}(x^i)$ by their values given in (A.10) and (A.11), respectively, we have

$$A_{\text{sub}} = \frac{\Delta_{x'}}{p-1}\left((p-1)\frac{\Delta_{x'}^2}{4(p-1)^2} + \frac{\Delta_{x'}^2}{(p-1)^2}\sum_{i=2}^{p-1}(p-i)(i-1)\right)\cdot$$

Next, we apply the following properties,

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \quad \text{and} \quad \sum_{i=1}^{n} i^2 = \frac{1}{6}\ n\ (n+1)(2n+1), \tag{A.12}$$

and obtain the desired result:

$$\begin{aligned}
A_{\text{sub}} &= \frac{\Delta^3_{x'}}{(p-1)^3}\left(\frac{1}{4}(p-1)+\sum_{i=2}^{p-1}\Big(-p+(p+1)i-i^2\Big)\right)\\
&= \frac{\Delta^3_{x'}}{(p-1)^3}\left(\frac{1}{4}(p-1)+(p-2)(-p)+(p+1)\left(\frac{(p-1)p}{2}-1\right)\right.\\
&\quad \left.-\frac{1}{6}\,(p-1)\;p\;(2(p-1)+1)+1\right)\\
&= \frac{\Delta^3_{x'}}{(p-1)^3}\left(\frac{-4p^2+9p-1}{4}+\frac{p^3+3p^2-10p}{6}\right)\\
&= \frac{\Delta^3_{x'}}{(p-1)^3}\;\frac{2p^3-6p^2+7p-3}{12}\\
&= \frac{\Delta^3_{x'}}{12}\;\frac{2p^3-2p^2-4p^2+4p+3p-3}{(p-1)^3}\\
&= \frac{\Delta^3_{x'}}{12}\;\frac{(p-1)(2p^2-4p+3)}{(p-1)^3}\\
&= \frac{\Delta^3_{x'}}{12}\;\frac{2p^2-4p+3}{(p-1)^2}.
\end{aligned}$$

Replacing $\Delta_{x'}$ by its value for the left and right subproblems, that is,

$$\Delta_{x^L} = \Delta_x\,\frac{j-1}{p-1} \;\text{ and }\; \Delta_{x^R} = \Delta_x\,\frac{p-j}{p-1},$$

we obtain the area of the envelopes for the left and right subproblems as

$$A_{x^L} = \frac{\Delta^3_x}{12}\;\frac{2p^2-4p+3}{(p-1)^5}\;(j-1)^3 \;\text{ and }\; A_{x^R} = \frac{\Delta^3_x}{12}\;\frac{2p^2-4p+3}{(p-1)^5}\;(p-j)^3.$$

Summing these two quantities, we obtain the area of the envelopes for both subproblems generated by branching at x^j:

$$\begin{aligned}
A_x &= \frac{\Delta^3_x}{12}\;\frac{2p^2-4p+3}{(p-1)^5}\;[(j-1)^3+(p-j)^3]\\
&= \frac{\Delta^3_x}{12}\;\frac{2p^2-4p+3}{(p-1)^5}\;[(j-1+p-j)\;\big((j-1)^2+(p-j)^2-(p-j)(j-1)\big)]\\
&= \frac{\Delta^3_x}{12}\;\frac{2p^2-4p+3}{(p-1)^4}\;\big((j-1)^2+(p-j)^2-(p-j)(j-1)\big)\\
&= \frac{\Delta^3_x}{12}\;\frac{2p^2-4p+3}{(p-1)^4}\;\big(((j-1)+(p-j))^2-3(p-j)(j-1)\big)\\
&= \frac{\Delta^3_x}{12}\;\frac{2p^2-4p+3}{(p-1)^4}\;\big((p-1)^2-3(p-j)(j-1)\big),
\end{aligned}$$

which is the desired result.

Let us now consider the area A_λ obtained after branching at λ^j, as shown in Figure 13. This area can be seen as the sum of the areas of $p-1$ trapeziums. The "height," H, of a trapezium is equal to the length of a piece, namely, $\frac{\Delta_x}{p-1}$. The length of the bases, the parallel sides at $x = x^i$, $1 \le i \le p$, and denoted B_i, is equal to the sum of ϵ_{x^2,Δ_x} defined in (A.10) and $\epsilon_{x^2,O}(x^i)$, the maximum overestimation approximation error that can be produced at x^i for the subproblems obtained after branching on λ^j. For the indices $i \le j$, the error $\epsilon_{x^2,O}(x^i)$ is equal to the difference

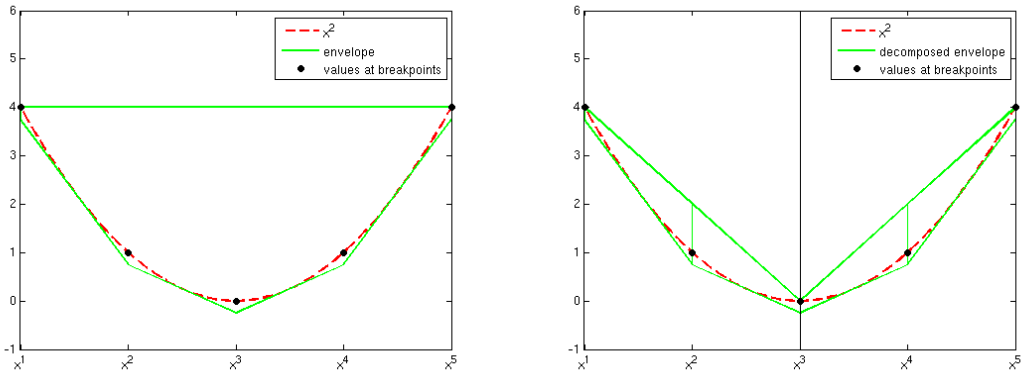

Figure 13: Left: envelope before branching; right: decomposition into 2 trapeziums of the envelopes for x^2 obtained by branching on λ^3 (5 breakpoints used).

at point x^i between the square function and the straight line joining $(x^1,(x^1)^2)$ to $(x^j,(x^j)^2)$, while, for the indices $i > j$, it corresponds to the difference with the straight line joining $(x^j,(x^j)^2)$ to $(x^p,(x^p)^2)$. Thus, by a similar argument to the one used prove (A.11), we obtain

$$\epsilon_{x^2,O}(x_i) = \begin{cases} \frac{\Delta_x^2}{(p-1)^2}\ (j-i)(i-1) & \text{if } i \leq j, \\ \frac{\Delta_x^2}{(p-1)^2}\ (p-i)(i-j) & \text{if } i > j. \end{cases} \tag{A.13}$$

Therefore, the area associated to the branching on λ^j can be computed as

$$A_\lambda \ = \ \sum_{i=1}^{p-1} \frac{1}{2} H(B_i + B_{i+1}) = \sum_{i=1}^{p-1} \frac{1}{2} \frac{\Delta_x}{p-1}\ (2\epsilon_{x^2,\Delta_x} + \epsilon_{x^2,O}(x^i) + \epsilon_{x^2,O}(x^{i+1})).$$

Because the overestimation error $\epsilon_{x^2,O}(x^i)$ is equal to zero at x^1 and x^p, A_λ becomes

$$A_\lambda = \frac{1}{2}\frac{\Delta_x}{p-1}\ \left(2\sum_{i=1}^{p-1} \epsilon_{x^2,\Delta_x} + 2\sum_{i=2}^{p-1} \epsilon_{x^2,O}(x^i)\right).$$

Since the expression of $\epsilon_{x^2,O}(x^i)$ depends on the value of i with regard to j, we obtain

$$A_\lambda = \frac{\Delta_x}{p-1}\ \left(\sum_{i=1}^{p-1} \epsilon_{x^2,\Delta_x} + \sum_{i=2}^{j} \epsilon_{x^2,O}(x^i) + \sum_{i=j+1}^{p-1} \epsilon_{x^2,O}(x^i)\right).$$

Using (A.10), (A.13) and properties (A.12), we obtain

$$\begin{aligned}
A_\lambda &= \frac{\Delta_x}{p-1}\left(\sum_{i=1}^{p-1}\frac{\Delta_x^2}{4(p-1)^2}+\frac{\Delta_x^2}{(p-1)^2}\sum_{i=2}^{j}(j-i)(i-1)+\frac{\Delta_x^2}{(p-1)^2}\sum_{i=j+1}^{p-1}(p-i)(i-j)\right)\\
&= \frac{\Delta_x^3}{(p-1)^3}\left(\frac{p-1}{4}+\sum_{i=2}^{j}(-i^2+(j+1)i-j)+\sum_{i=j+1}^{p-1}(-i^2+(p+j)i-pj)\right)\\
&= \frac{\Delta_x^3}{(p-1)^3}\left(\frac{p-1}{4}-\sum_{i=2}^{p-1}i^2+(j+1)\sum_{i=2}^{j}i-j(j-1)+(p+j)\sum_{i=j+1}^{p-1}i-pj(p-1-j)\right)\\
&= \frac{\Delta_x^3}{(p-1)^3}\left(\frac{p-1}{4}-\frac{1}{6}(p-1)p(2p-1)+1+\frac{j(j+1)}{2}(j+1-p-j)-j-1+j\right.\\
&\quad \left.+(p-1)j^2+\frac{p(p+j)(p-1)}{2}-pj(p-1)\right)\\
&= \frac{\Delta_x^3}{(p-1)^2}\left(\frac{1}{4}-\frac{1}{6}(2p^2-p)-\frac{(j^2+j)}{2}+j^2+\frac{(p^2+pj)}{2}-pj\right)\\
&= \frac{\Delta_x^3}{12(p-1)^2}\left(3+2p^2+6j^2-6j-6pj+2p\right)\\
&= \frac{\Delta_x^3}{12(p-1)^2}\left(1+2(1+p^2+p+3j^2-3j-3pj)\right)\\
&= \frac{\Delta_x^3}{12(p-1)^2}\left(1+2((p-1)^2+3p+3j^2-3j-3pj)\right)\\
&= \frac{\Delta_x^3}{12(p-1)^2}\left(1+2((p-1)^2-3(p-j)(j-1))\right).
\end{aligned}$$

Therefore, the ratio $\dfrac{A_\lambda}{A_x}$ can be derived as

$$\frac{A_\lambda}{A_x}=\frac{(p-1)^2\left(1+2\left((p-1)^2-3(p-j)(j-1)\right)\right)}{(2p^2-4p+3)\left((p-1)^2-3(p-j)(j-1)\right)}.$$

To show that A_λ is always larger than A_x, it suffices to prove that the ratio $\frac{A_\lambda}{A_x}$ is larger than 1, which is equivalent to

$$(p-1)^2(1+2\alpha)>(2p^2-4p+3)\alpha, \tag{A.14}$$

where we define α as

$$\alpha=(p-1)^2-3(p-j)(j-1).$$

In order to show (A.14), we compute

$$\begin{aligned}
(p-1)^2(1+2\alpha)-(2p^2-4p+3)\alpha &= (p-1)^2(1+2\alpha)-2(p-1)^2\alpha-\alpha,\\
&= (p-1)^2-\alpha,\\
&= (p-1)^2-(p-1)^2+3(p-j)(j-1),\\
&= 3(p-j)(j-1).
\end{aligned}$$

Equation (A.14) follows from the fact that $3(p-j)(j-1)$ is always positive, because $1<j<p$.

It remains to show that the best choice to minimize the sum of the areas of the envelopes for the left and right subproblems is to branch at the midpoint of the range of the variable x. Actually, we look for the point x^j that minimizes the quantity A_x, which can be seen as a function of j. At

this point, the derivative of $A_x(j)$ must be equal to zero. Therefore, we want to determine j^* for which

$$A'(j) = \frac{\Delta_x^3}{12} \frac{2p^2 - 4p + 3}{(p-1)^4} \big(- 3(-(j-1) + (p-j))\big) = 0,$$

that is

$$j^* = \frac{p+1}{2}.$$

Since it can be easily shown that the second derivative of $A_x(j)$ is positive, j^* minimizes $A_x(j)$. Because the breakpoints are equally spaced between x^1 and x^p, x^{j^*} corresponds to the point at the half of the range of the variable x, (which is not necessarily a breakpoint since j is not an integer if p is even). □

References

Abhishek, K., Leyffer, S., and Linderoth, J. (2006). FilMINT: An Outer-Approximation-Based Solver for Nonlinear Mixed Integer Programs. Technical Report ANL/MCS-P1374-0906, Mathematics and Computer Science Division, Argonne National Laboratory.

Adjiman, C., Dallwig, S., Floudas, C., and Neumaier, A. (1998). A global optimization method, αBB, for general twice-differentiable constrained NLPs - I. Theoretical advances. *Computers and Chemical Engineering*, 22:1137–1158.

Al-Khayyal, F. and Falk, J. (1983). Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8:273–286.

Beale, E. and Tomlin, J. (1970). Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables. In Lawrence, J., editor, *Proceedings of Fifth International Conference on Operation Research*, pages 447–454, London. Tavistock Publications.

Bonami, P., Biegler, L., Conn, A., Cornuejols, G., Grossmann, I., Laird, C., Lee, J., Lodi, A., Margot, F., Sawaya, N., and Wächter, A. (2005). An algorithmic framework for convex mixed integer nonlinear programs. Technical Report RC23771, IBM Research Report.

Bussieck, M. (2004). Private communication.

Caratzoulas, S. and Floudas, C. (2004). A trigonometric convex underestimator for the base functions in Fourier space. *Journal of Optimization Theory and Applications*, 124(2):339–362.

Duran, M. and Grossmann, I. (1986). An outer approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339.

Fletcher, R. and Leyffer, S. (1994). Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming*, 66:327–349.

Geoffrion, A. (1972). A generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260.

Griewank, A. (2000). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia.

Harjunkoski, I., Westerlund, T., Pörn, R., and Skrifvars, H. (1998). Different transformations for solving non-convex trim loss problems by MINLP. *Journal of Operational Research*, 105:594–603.

Hiriart-Urruty, J. and Lemaréchal, C. (2001). *Fundamentals of convex analysis*. Springer Verlag, Berlin.

Leyffer, S. (1993). *Deterministic Methods for Mixed Integer Nonlinear Programming*. PhD thesis, University of Dundee, United Kingdom.

Leyffer, S., Sartenaer, A., and Wanufelle, E. (2008). Numerical experiments with branch-and-refine for mixed integer nonconvex global optimization. Technical Report ANL/MCS-P1548-0908, Mathematics and Computer Science Division, Argonne National Laboratory.

Martin, A., Möller, M., and Moritz, S. (2006). Mixed integer models for the stationary case of gas network optimization. *Mathematical Programming*, 105:563–582.

McCormick, G. (1976). Computability of global solutions to factorable nonconvex programs: Part I - convex underestimating problems. *Mathematical Programming*, 10:147–175.

Möller, M. (2004). *Mixed Integer Models for the Optimisation of Gas Networks*. PhD thesis, Technische Universität Darmstadt.

Quesada, I. and Grossmann, I. (1992). An LP/NLP based branch-and-bound algorithm for convex MINLP optimization problems. *Computers and Chemical Engineering*, 16:937–947.

Quist, A., van Gemeert, R., Hoogenboom, J., Illes, T., Roos, C., and Terlaky, T. (1998). Application of nonlinear optimization to reactor core fuel reloading. *Annals of Nuclear Energy*, 26:423–448.

Tawarmalani, M. and Sahinidis, N. (2002). *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Tomlin, J. (1981). A suggested extension of special ordered sets to non-separable non-convex programming problems. *Annals of Discrete Mathematics*, 11:359–370.

Wanufelle, E. (2007). *A global optimization method for mixed integer nonlinear nonconvex problems related to power systems analysis*. PhD thesis, Department of Mathematics, FUNDP, Belgium.

Williams, H. (2005). *Model Building in Mathematical Programming*. John Wiley and Sons, 4th edition.