Abstract—There has been much recent interest in hierarchies of progressively stronger convexifications of polynomial optimisation problems (POP). These often converge to the global optimum of the POP, asymptotically, but prove challenging to solve beyond the first level in the hierarchy for modest instances. We present a finer-grained variant of the Lasserre hierarchy, together with first-order methods for solving the convexifications, which allow for efficient warm-starting with solutions from lower levels in the hierarchy.

Index Terms—Optimization, Optimization methods, Mathematical programming, Polynomials, Multivariable polynomials

I. INTRODUCTION

There has been much recent interest in efficient solvers for polynomial optimisation and semidefinite programming (SDP) relaxations therein. Much of this interest has been motivated by the work of Lavaei and Low [30] on relaxations of alternating-current optimal power flows (ACOPF), an important steady-state problem in power systems engineering [39]. For ACOPF, Ghaddar et al. [16] have shown that the relaxation of Lavaei and Low is the first level in a number of hierarchies of SDP relaxations, including that of Lasserre [28], whose optima converge to the global optimum of the polynomial optimisation problem (POP). The higher levels of the hierarchies present a considerable computational challenge both in ACOPF and other POP. This is due to the dimension of the relaxation, the super-cubic complexity of traditional interior-point methods for solving SDPs, as well as their limited warm-start capabilities. Although a number of hierarchies of second-order cone programming problems [27], [32], [36], [37] have been studied recently, for which the interior-point methods are better developed, the relatively weaker relaxations require yet larger instances to be solved to provide a strong bound, and have not been proven to dominate the SDP-based approaches in practice.

In this paper, we suggest that a finer-grained hierarchy of SDP relaxations may make sense, if it is accompanied by a method capable of “warm-starting,” i.e., the use of a solution for one relaxation in speeding up the solution of a stronger relaxation. Specifically, we:

- Present a variant of Lassere’s hierarchy of SDP relaxations of a POP, where localising matrices are added one by one.
- Design a first-order method for solving these relaxations, which allows for an efficient warm-starting, and employs a closed-form step in the coordinate descent.
- Study the conditions of asymptotic convergence of the approach to the global optimum of the POP.

We hope these contributions, alongside [31], [33], [34], could spur further interest in first-order methods for convex relaxations in polynomial optimisation.

II. NOTATION AND RELATED WORK

A. Notation

Let $\mathbb{R}$ and $\mathbb{Z}_+$ be the real numbers and non-negative integers, respectively. Denote by $j$ the imaginary unit of a complex number. The matrix transpose operator and trace operator are denoted by $^T$ and $\text{tr}$, respectively. $\text{Re}$ and $\text{Im}$ are the operators that return the real and imaginary parts of a complex number. Let $e_k$ be the $k$-th standard basis vector in $\mathbb{R}^n$, $N \in \mathbb{Z}_+$. Denote by $\mathbb{R}[x]$ the set of real-valued multivariate polynomials in $x_i$, $i = 1, \ldots, n$, where $n \in \mathbb{Z}_+$. A polynomial $f \in \mathbb{R}[x]$ is represented as $f(x) := \sum_{\alpha \in \mathcal{F}} c(\alpha)x^\alpha$, where $\mathcal{F} \subset \mathbb{Z}_+^n$, and $c(\alpha)$, $\alpha \in \mathcal{F}$ are the corresponding real coefficients, and $x^\alpha := x_1^{\alpha_1}x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, $\alpha \in \mathcal{F}$ are the corresponding monomials. The support of a polynomial $f \in \mathbb{R}[x]$ is defined by $\text{support}(f) := \{ \alpha \in \mathcal{F} | c(\alpha) \neq 0 \}$. The degree of a polynomial $f \in \mathbb{R}[x]$ is defined by degree$(f) := \max \{ \sum_{i=1}^n \alpha_i | \alpha \in \text{support}(f) \}$. For a non-empty finite set $\mathcal{G} \subset \mathbb{Z}^n_+$, $\mathbb{R}[x, \mathcal{G}] := \{ f \in \mathbb{R}[x] | \text{support}(f) \subset \mathcal{G} \}$. $\mathbb{R}[x, \mathcal{G}]^2$ is the set of the SOS polynomials in $\mathbb{R}[x, \mathcal{G}]$. $S(\mathcal{G})$ is the set of $|\mathcal{G}| \times |\mathcal{G}|$ symmetric matrices and $S_+(\mathcal{G})$ is the set of positive semidefinite matrices in $S(\mathcal{G})$ with coordinates $\alpha \in \mathcal{G}$, $u(x, \mathcal{G})$ is a $|\mathcal{G}|$-dimensional column vector consisting of element $x^\alpha$, $\alpha \in \mathcal{G}$. The set $\mathcal{A}^C_w$ is defined as $\mathcal{A}^C_w := \{ \alpha \in \mathbb{Z}^n_+ | \alpha_i = 0, i \notin C, \sum_{i \in C} \alpha_i \leq w \}$, for every $C \subset \{1, \ldots, n\}$, $w \in \mathbb{Z}_+$.

B. Polynomial Optimization Problem

Let us consider a polynomial optimisation problem:

$$\min_x f_0(x)$$
$$\text{s.t.} \quad f_k(x) \geq 0, \quad k = 1, \ldots, m,$$

(PP)
where $x \in \mathbb{R}^n$ is the decision variable, the objective and the constraints are defined in terms of are defined in terms of multi-variate polynomials $f_k$, for $k = 0, \ldots, m$, in $x \in \mathbb{R}^n$. A number of approaches have been proposed for solving polynomial optimization problems, including spatial branch-and-bound techniques \[3\] and branch-and-reduce \[45\], cutting plane methods \[7\], and moment and sum-of-squares methods \[28\]. Please see \[1\], \[2\], \[9\] for detailed surveys. At the same time, noticed that no method can be unconditionally finitely convergent, as per the solution to Hilbert’s tenth problem. 

### C. Moment-based Methods

Moment-based methods are a popular approach to solving POP \([PP]\), based on the work of Lasserre \[28\]. Given $S \in \mathbb{R}^n$, denote by $P_w(S)$ the cone of polynomials of degree at most $w$ that are non-negative over $S$. We use $\Sigma_w$ to denote the cone of polynomials of degree at most $d$ that are sum-of-square of polynomials. Using $G = \{f_k(x) : k = 1, \ldots, m\}$ and denoting $S_G = \{x \in \mathbb{R}^n : f(x) \geq 0, \forall f \in G\}$ the basic closed semi-algebraic set defined by $G$, we can rephrase POP \([PP]\) as:

$$
\max \varphi \quad \text{s.t.} \quad f(x) - \varphi \geq 0 \quad \forall x \in S_G,
$$

$$
= \max \varphi \quad \text{s.t.} \quad f(x) - \varphi \in P_w(S_G).
$$

Problem \([1]\) is referred to as \([PP-D]\). Although \([PP-D]\) is a conic problem, it is not known how to optimise over the cone $P_w(S_G)$ efficiently. Lasserre \[28\] introduced a hierarchy of SDP relaxations corresponding to liftings of polynomial problems into higher dimensions. In the hierarchy of SDP relaxations, one convexifies the problem, obtains progressively stronger relaxations, but the size of the SDP instances soon becomes computationally challenging. Under assumptions slightly stronger than compactness, the optimal values of these problems converge to the global optimal value of the original problem, \([PP]\).

The approximation of $P_w(S_G)$ used by Lasserre \[28\] is the cone $K_G$, where

$$
K_G = \Sigma_w + \sum_{k=1}^{m} f_k(x) \Sigma_{w-deg(f_k)},
$$

and $w \geq d$. The corresponding optimization problem over $S$ can be written as:

$$
\max \varphi \quad \text{s.t.} \quad f(x) - \varphi = \sigma_0(x) + \sum_{i=1}^{m} \sigma_i(x) f_k(x)
$$

$$
\sigma_0(x) \in \Sigma_w, \quad \sigma_i(x) \in \Sigma_{w-deg(f_k)}.
$$

Problem \([3]\) is referred to as \([PP-Hw]\). \([PP-Hw]\) can be reformulated as a semidefinite optimization problem. We denote the dual of \([PP-Hw]\) by \([PP-Hw]^*\). The computational cost of the problem clearly depends on both the degree of the polynomials, $w$, and the dimension of the problem. Both the number of constraints and their dimensions can be large when numerous variables of the POP are involved in high-degree polynomial expressions.

### D. Dynamic Generation of Relaxations

Recently, there has been a considerable effort \[10\]–\[12\], \[16\], \[17\], \[20\], \[23\]–\[25\], \[47\] focussed on the design and implementation of decomposition algorithms for solving relaxations \([PP-Hw]\) and algorithms for improving these relaxations. For the hierarchy \([PP-Hw]\), Ghaddar et al. \[16\]–\[18\] have proposed a method, which generated the most violated constraint in each iteration. When tested on examples in dimension up to 10 by Ghaddar et al. \[16\], this performs well, but due to the limitations of current SDP solvers, does not scale much further. A similar approach is being considered by Molzahn and Hiskens \[38\] in a power-system specific heuristic, and by Chen \[13\] for general polynomial optimisation problems. Kienati et al. \[23\]–\[25\] study a variant of Bender’s decomposition, where the moment variable is decomposed into blocks, such that the constraints particular to each block are considered in isolation in one sub-problem, and the constraints spanning multiple blocks are considered in a so called master problem. In \[25\] convergence to the optimum of the polynomial optimisation problem is studied, with details of run-time provided on examples in dimension up to 10. Wittek \[47\] describes a mixed-level relaxation, where monomials can be added arbitrarily, but without an algorithmic approach for their addition. In the context of relaxations of partial differential equations (PDEs), the so-called prolongation operators are used routinely. Campos \[10\]–\[12\] has translated this work to SDP relaxations of PDEs and beyond. Hall \[20\] describes a “Sum of Squares Basis Pursuit” using linear or second-order cone programming, but also shows \[20\] Proposition 3.5.5 the approach is not convergent. Our hope is to improve upon this state of the art.

### III. The Hierarchy

Let us consider the dual of Problem \([3]\), i.e., the semidefinite programming relaxation obtained by Lasserre in the method of moments. Following Chapter 6 in \[29\], we can write it as:

$$
\inf_y F(y) \quad \text{(PP-Hw)}
$$

$$
s.t. \quad M_w(y) \succeq 0
$$

$$
M_{w-v}(f_k y) \succeq 0 \quad \forall k = 1, \ldots, m
$$

$$
y_0 = 1
$$

where $F$ is a linear functional, $M_w(y)$ is called the moment matrix and $M_w(y) = \sum \alpha y_\alpha C_\alpha^0$ for some appropriate real matrix $C_\alpha^0$. $M_{w-v_j}(f_k y)$ is called a localising matrix and $M_{w-v_j}(f_k y) = \sum \alpha y_\alpha C_\alpha^0$ some appropriate real matrix $C_\alpha^0$ for each inequality $k = 1, \ldots, m$. 


It is clear we could replace constraints \(4\) and \(5\) with a constraint on a single block-diagonal matrix to be positive definite, where the blocks on the diagonal would be \(M_w(y)\) and \(M_{w-v_j}(f_k y), k = 1, \ldots, m\). Let us denote these blocks \(B_w\) and let us use the notation \(\prod_{b \in B_w} b\) for the formation of the block-diagonal matrix, with the blocks taken in arbitrary order. We can use a sub-set of blocks \(B_q \subseteq B_w\) to be considered in iteration \(q\). Let us also consider the complement \(\bar{B}_q\) of the block \(B_q\), i.e., \(B_q \cup \bar{B}_q = B_w\) for the \(w\) current at \(j\). Our hierarchy is simply based on the SDP relaxation [PP-H_w] parametrised by the choice \(B_q\) of the blocks at the \(q\)-th iteration within the \(w\)-th level of relaxation:

\[
\inf_y L(y) \quad \text{(R(w, B_q))}
\]

\[
\text{s.t. } \prod_{b \in B_q} b \geq 0
\]

\[
y_0 = 1.
\]

It is not clear what blocks \(B_q \subseteq B_w\) to consider, though. In determining those, we use two maps:

- \(F : B \to [m]\) maps blocks to constraints \(f_j\) of the polynomial optimization problem
- \(G : [m] \to N\) maps the constraints to the variables \(i \in N\) of the polynomial optimization problem.

The composite mapping \(G \circ F : B \to N\) hence maps the blocks to the variables of the polynomial optimization problem and describes the relationship between the POP and the SDP. Notice that \(F, G\) are known. One can construct \(F, G\) in the process of formulating the moment and localising matrices. Alternatively, one can use the simplistic procedure for obtaining \(F, G\), such as Algorithm [2]

### IV. AN ALGORITHM

Let us describe the complete algorithm for solving the polynomial optimisation problem, based on:

- The moment-based relaxations \((R(w, B_q))\) suggested above.

- A novel “all violated” block-addition rule, for picking suitable blocks to add to the relaxation, building upon the “most violated” [13] and “power mismatch” [38] block-addition rules.

- The augmented Lagrangian approach, an optimisation strategy studied since the 1950s, e.g., by Hestenes [21] and Powell [40].

- The block-wise additivity of the augmented Lagrangian in the moment-based relaxations, as outlined above and developed further below.

- A novel closed-form step for the augmented Lagrangian approach, as applied to the moment-based relaxations, as explained below.

The ingredients, which are truly novel, are the block-separable augmented Lagrangian and the closed-form step, but we argue that the overall algorithm design should also be of interest. Subsequently, we prove the convergence of the algorithm to the global optimum in the following section.

#### A. The Overall Algorithm

Algorithm [1] captures the key algorithm schema, with details elaborated in Algorithms [2, 3]. In Algorithm [1] we first initialise \(d\) to one half of the maximum degree involved in (PP), rounded up, and \(B\) is set to a list of blocks considered.
Algorithm 2 blocks
Input: level \( w \) in the hierarchy (PP-H\(_ w \))
1: \( \bar{B} \leftarrow \) all the blocks in relaxation (PP-H\(_ w \)),
2: for \( j = 1, \ldots, m \) do
3:   perturb coefficients of the constraint \( f_j \)
4:   for \( \alpha = 1, \ldots, |\mathcal{F}_w| \) do
5:     \( \bar{y}_\alpha \leftarrow \text{rand}(1, 1) \)
6:   end for
7:   \( \bar{M}(\bar{y})^{-} \leftarrow M(0) + \sum_{\alpha \in \mathcal{F}_w} \bar{M}(\alpha)\bar{y}_\alpha - (\bar{M}(0) + \sum_{\alpha \in \mathcal{F}_w} \bar{M}(\alpha)\bar{y}_\alpha) \)
8:   \( F(j) \leftarrow \) map the remaining elements in \( M(\bar{y})^{-} \) to the corresponding block
9:   \( G(j) \leftarrow \) all the buses involved in the constraint \( f_j \)
10: end for
11: return \( (\bar{B}, F, G) \)

in (PP-H\(_ d \)). The main loop of the algorithm has \( w \) as the counter, which denotes the order of the relaxation, from which we are adding blocks in that particular iteration. On line 7 \( \text{blocks}(w) \) constructs the mapping between the buses and the blocks in the \( w \)-th level of relaxation. \( \bar{B} \) is a list of blocks from level \( w \) of the hierarchy, which has not been part of the SDP relaxation yet. \( F(j) \) is a mapping between the \( j \)-th polynomial constraint and the corresponding block. \( G(j) \) is a mapping between the \( j \)-th constraint and the corresponding buses. On lines 10-27 we solve the SDP relaxation using the block-coordinate descent method, where the blocks correspond to the blocks from (PP-H\(_ {w-1} \)) and \( B \). On line 22 \( \text{kEigs}(X) \) is a procedure, which obtains the largest \( r \) eigenvalues with the associated eigenvectors of \( X \). This could be simply implemented using a spectral decomposition of \( X \), where all but the largest \( r \) eigenvalues and the associated eigenvectors are discarded. Alternatively, one could use the power method repeatedly with deflation, or the extensions of the power method extracting multiple eigenvalues in one pass. On line 28 \( \text{addBlocks} \) is a procedure, which computes the projection of \( X \) onto rank one matrices, i.e. the closest rank one matrix to \( X \) with respect to the Frobenius norm, and one-by-one verifies whether the constraints are satisfied up to the accuracy \( \epsilon = 10^{-5} \). Whenever a constraint at bus \( v \) is not satisfied, the blocks in \( \bar{B} \) corresponding to bus \( v \) are added. The number of blocks added to \( \bar{B} \) in this run of \( \text{addBlocks} \) is output into \( s \). The Cauchy convergence criteria are set to \( 10^{-3} \), on both lines 6 and 8. Finally, at the end of the outer loop on line 35 \( \text{lift}(X, Z, y) \) is a procedure, which lifts the two matrices \( X, Z \) and vector \( y \) to a higher dimension.

\section*{B. The Choice of Blocks}

Next, we suggest the procedure for the addition of blocks with constraints violated at the current relaxation to the relaxation. Alternatively, one can see that as the removal of the redundant blocks in the relaxation.

In Algorithm 2 on line 11 \( \bar{B} \) is a list which contains all the blocks in the \( w \)-th level of relaxation. On line 9 to 11 we construct the mapping \( F(j) \), for every \( j = 1, \ldots, m \). On line 9 we construct the mapping \( G \).

In Algorithm 3 we pick blocks to add to the current relaxation, which correspond to constraints violated by more than \( \epsilon \) by the current iterate. In order to obtain vector \( x \) of (PP) from the current iterate, on line 3 \( \text{kEigs}(X) \) computes the largest eigenvalue and the associated eigenvector of the matrix corresponding to second-order monomials. Subsequently, we obtain a vector \( x \) by Cholesky decomposition of the rank-1 projection on line 4. On line 5 we verify if the power constraints are satisfied up to the accuracy \( \epsilon = 10^{-5} \). On line 10 to 15 we remove the redundant blocks involved in the violated constraints. On line 13 we update the total number of blocks added and removed. On line 16 the blocks added to the list \( B \) are removed from the list \( \bar{B} \).

Algorithm 3 addBlocks
Input: \( X, B, F, G \)
1: \( s \leftarrow 0, \epsilon \leftarrow 10^{-5}, I \leftarrow \emptyset \)
2: \( M_2 \leftarrow \) submatrix of \( X \) corresponding to the second-order monomials
3: \( (E, V) \leftarrow \text{kEigs}(M_2, 1) \)
4: \( x \leftarrow \text{cho}(V EV^T) \)
5: for constraint \( k = 1, \ldots, m \) do
6:   if \( \min(f_k(x), 0) > \epsilon \) then
7:     \( I \leftarrow I + \{k\} \)
8:   end if
9: end for
10: for \( j = 1, \ldots, m \) do
11:   if \( \{G(j)\cap I\} \neq \emptyset \) then
12:     \( B \leftarrow B \cup \{F(j)\} \)
13:     \( s \leftarrow s + 1 \)
14:   end if
15: end for
16: \( \bar{B} \leftarrow B \setminus B \)
17: return \( (B, \bar{B}, s) \)

Algorithm 4 flatExtension
Input: constants \( D, d \), primal-dual pair \( X, y \) level \( w \in \mathbb{R} \)
1: \( M_2, M_3, \ldots, M_w \leftarrow \) sub-matrices of \( X \) with moment matrices of the order \( \leq w \)
2: if \( \text{rk}(M_w) = \text{rk}(M_{w-1}) \) or \( \text{rk}(M_w) = \text{rk}(M_{w-d}) \) then
3:   return True
4: end if
5: return False
In Algorithms 4 and 5, we suggest the usual flat extension test of convergence, and a comparison of objective-function values of the current iterate and the vector obtained by Cholesky decomposition of the closest rank-1 projection of the submatrix of $X$ corresponding to second-order monomials. The comparison of objective-function values needs to be considered alongside the satisfaction of all constraints of (PP), as suggested on Line 30 of the main algorithm.

**C. The Augmented Lagrangian**

The augmented Lagrangian $L_{\mu}$ of (PP-H) is defined by

$$L_{\mu}(Z, y, X) = c^T y + < Z, \tilde{M}(y) - X > + \frac{1}{2\mu} ||\tilde{M}(y) - X||^2_F,$$

where $Z \in \tilde{S}$ is the dual variable and the parameter $\mu$ is a positive real number. Typically, this is used in conjunction with the alternating direction method of multipliers (ADMM), which is used in iteration $k$ computes the updates:

$$y^{k+1} = \min_y L_{\mu}(Z^k, y, X^k),$$

$$X^{k+1} = \min_{X \geq 0} L_{\mu}(Z^k, y^{k+1}, X),$$

$$Z^{k+1} = Z^k + \frac{\tilde{M}(y^{k+1}) - X^{k+1}}{\mu}. \tag{12}$$

Notice that we effectively perform a two-block decomposition of the augmented Lagrangian, rather than the multi-block decomposition, which is known [14] to be divergent in some cases, esp. when [22] the functions involved are not strongly convex.

In our case, we have the following special structure:

**Proposition 1.** The augmented Lagrangian is additive with respect to the blocks.

**Proof:** The constraint $X \in \tilde{S}_+$ is sufficient to ensure that $\tilde{M}(y) - X$ has the same block structure of $\tilde{M}(y)$. To complete the proof, we use the fact that the trace of a block diagonal matrix with square blocks is equal to the sum of the traces of the blocks.

Given the block diagonal structure of $\tilde{M}(y)$, we are able to decompose the computation block-wise. The first-order optimality condition for (11) yields

$$- Z^k + \frac{1}{\mu}(X^{k+1} - \tilde{M}(y^{k+1})) = 0 \tag{13}$$

and

$$X^{k+1} = \tilde{M}(y^{k+1}) + \mu Z^k, \quad X^{k+1} \succeq 0. \tag{14}$$

In order to find the solution of (14), spectral decomposition is performed on the matrix

$$V^{k+1} := \tilde{M}(y^{k+1}) + \mu Z^k, \tag{15}$$

and the result is used to formulate $V_+ E_+ V_+^T$, where $E_+$ contains the nonnegative eigenvalues of the matrix $V^{k+1}$, and the columns of $V_+$ are the corresponding eigenvectors.

Substitute $\tilde{M}(y^{k+1}) + \mu Z^k$ by $V^{k+1}$ into (12), which yields

$$Z^{k+1} = V^{k+1} - X^{k+1}. \tag{16}$$

**D. Block-Coordinate Descent Method**

Further, we need a method to compute the update (10) efficiently. Considering [1] we suggest:

**Proposition 2.** For every $i \in \tilde{F}$, the first-order optimality conditions for the $i$-th coordinate of $y$ in (10) yield:

$$y^{k+1}_i = - \frac{a_2(i)}{2a_1(i)} \tag{17}$$

where

$$a_1(i) = \frac{1}{2\mu} \text{tr}(M_i M_i^T), \quad \tag{18}$$

$$a_2(i) = c(i) + \text{tr}(Z^k M_i^T) + \sum_{j \neq i} \frac{1}{\mu} \text{tr}(M_j y_j M_j^T)$$

$$- \frac{1}{\mu} \text{tr}(M(0) M_i^T) - \frac{1}{\mu} \text{tr}(M_i^T X^k). \tag{19}$$

**Proof:** Note that $L_{\mu}(Z^k, y^k, X^k)$ is a quadratic function of $y_i$ and algebraic manipulations lead to the result.

When we update the $\alpha$-th coordinate of $y$, the only blocks of $\tilde{M}(y)$ that are required are the ones containing $y_\alpha$. Each block of $\tilde{M}(y)$ only contains a portion of the variable $y$, which results in speeding up the implementation.

**V. AN ANALYSIS**

Algorithm 1 has been designed so as to be convergent. In particular:

- in the outer-most loop (Lines 3–23), for each $w$, we compute the optimum of a relaxation $[\text{OP}_2-H_w]^*$. In the limit of $w$, relaxations $[\text{OP}_2-H_w]^*$ converge to the global optimum of $[\text{OP}_2]$, as shown by Ghaddar et al. [16]. However, the relaxations are not formed explicitly, but dynamically, by considering the violated blocks.
- in the block-addition loop (Lines 6–21), specifically, we include the violated blocks. The finiteness of the loop
is given by the finiteness of the number of blocks in $[\text{OP}_2-H_w]^*$ for any finite $w$.

- in the inner-most loop (Lines 8–19), the dynamically constructed relaxation of $[\text{OP}_2-H_w]^*$ is solved by a first-order method. The convergence is based on a rich history of work on the convergence of first-order methods for semidefinite programming.

In the analysis, we start from the inner-most loop and proceed outwards, while formulating the assumptions before we use them. Throughout, we use the notion of the dual problem of $(\text{PP-H}_w)$, which is

$$
\begin{align*}
\max_Z & -\langle M(0), Z \rangle \\
\text{s.t.} & \langle M(i), Z \rangle = c_i, \ i = 1, \ldots, m, \\
& Z \succeq 0.
\end{align*}
$$

**Proposition 3.** If Assumptions 1–3 hold, the sequence $\{Z^k\}$ generated in the inner loop (Lines 8–19) converges to $Z^*$, where $Z^*$ is an optimal solution of $(\text{OP}_2-H_w)$ and the augmented Lagrangian $L_p(Z^k, y^k, X^k)$ converges to $p^*$ as $k$ goes to infinity, where $p^* = (\text{PP-H}_w) = (20)$.

**Proof:** The global convergence follows from Theorem 8 of [15], but could be derived also directly from Theorem 1 of [42] and Theorem 4 of [43]. For the stopping rule on Line 11 of Algorithm 1 one could envision each subsequent iteration of the loop on Lines 9–20 using an increased precision, but one can equally well prove convergence with limited precision.

**Proposition 4.** If Assumptions 7, 8 hold, for each $w \geq 3$, the sequence of optima of the semidefinite-programming relaxations $R(w, B_w)$ generated in block-addition loop (Lines 6–21) converges to $\text{PP-H}_w$ as $q$ goes to infinity.

**Proof:** The proof follows from Theorem 5 of [26]. The convergence is finite for any finite $w$, because the number of blocks is finite [26], for any finite $w$. Notice we have replaced the assumptions, in line with [16].

Overall, we have the convergence result as follows:

**Proposition 5.** If Assumptions 7, 8 hold, 
$$
\inf \{L^*_p(Z^k, y^k, X^k) \} \rightarrow \min \{(\text{PP})\} \text{ as } w \rightarrow \infty, \ k \rightarrow \infty.
$$

**Proof:** The proof combines Propositions 3 and 4 above. The complication is in the stopping rules for the coordinate-descent (Line 11) and overall solver of the SDP (Line 9), such that we can truly obtain the (possibly irrational) solution of the correct SDP, while adding blocks. For the purposes of the proof, we suggest a dynamic stopping rule for Line 9 of Algorithm 1, where in each subsequent SDP solved, i.e. each subsequent iteration of the loop on Lines 9–20, the precision required is doubled. A dynamic stopping rule for Line 11 of Algorithm 1 can be similarly doubled in each iteration. This would clearly lead to an arbitrary precision, eventually, and would be able to produce the possibly irrational numbers, asymptotically.

Notice that for many problems, including the ACOPF, Assumptions 7, 8 are satisfied for all realistic choices of parameters, as the feasible region is compact. Although the assumptions may be relaxed slightly, the work of Matiyasevich [35] suggests that unconditional finite convergence is impossible.

**VI. Conclusions**

The development of practical and globally convergent solvers for polynomial optimisation problems is a major challenge within mathematical optimisation. In turn, this poses challenges in convex optimisation, including first-order methods for semidefinite-programming, and numerical linear algebra, such as the incremental update of the (truncated) singular value decomposition [6], [8], [19]. Overall, we have made preliminary steps towards the development of a method, which is both convergent and efficient in practice, although there are still very distinct limitations.
