2013 European Control Conference (ECC)
July 17-19, 2013, Zürich, Switzerland.

ECOS: An SOCP Solver for Embedded Systems

Alexander Domahidi[1], Eric Chu[2] and Stephen Boyd[2]

Abstract— **In this paper, we describe the *embedded conic solver* (ECOS), an interior-point solver for second-order cone programming (SOCP) designed specifically for embedded applications. ECOS is written in low footprint, single-threaded, library-free ANSI-C and so runs on most embedded platforms. The main interior-point algorithm is a standard primal-dual Mehrotra predictor-corrector method with Nesterov-Todd scaling and self-dual embedding, with search directions found via a symmetric indefinite KKT system, chosen to allow stable factorization with a fixed pivoting order. The indefinite system is solved using Davis' SparseLDL package, which we modify by adding dynamic regularization and iterative refinement for stability and reliability, as is done in the CVXGEN code generation system, allowing us to avoid all numerical pivoting; the elimination ordering is found entirely symbolically. This keeps the solver simple, only 750 lines of code, with virtually no variation in run time. For small problems, ECOS is faster than most existing SOCP solvers; it is still competitive for medium-sized problems up to tens of thousands of variables.**

I. Introduction

Convex optimization [1] is used in fields as diverse as control and estimation [2], finance (see [1, §4.4] for numerous examples), signal processing [3] and image reconstruction [4]. Methods for solving linear, quadratic, second-order cone or semi-definite programs (LPs, QPs, SOCPs, SDPs) are well understood, and many solver implementations have been developed and released, both in the public domain [5], [6] and as commercial codes [7], [8].

Almost all existing codes are designed for desktop computers. In many applications, however, the convex solver is to be run on a computing platform that is embedded in a physical device, or that forms a small part of a larger software system. In these applications, the same problem is solved but with different data and often with a hard real-time constraint. This requires that the solver be split into two parts: (1) An initialization routine that symbolically analyzes the problem structure and carries out other setup tasks, and (2) a real-time routine that solves *instances* of the problem. In these applications, high reliability is required: When the solver does not attain the standard exit tolerance, it must at least return reasonable results, even when called with poor data (such as with rank deficient equality constraints). In such cases, currently available solvers cannot be used due to code size, memory requirements, dependencies on external libraries, availability of source code, and other complexities.

In this paper, we describe the implementation of a solver for second-order cone programming that targets embedded systems. We focus on SOCPs since many potential real-time applications can be transformed into this problem class, including LPs, QPs and quadratically constrained QPs (QCQPs). See [9] or [1, Chap. 6-8] for some potential applications of SOCPs; a recent example for real-time SOCP is minimum-fuel powered descent for landing spacecraft [10].

Our *embedded conic solver* (ECOS) is written in single-threaded ANSI-C with no library dependence (other than `sqrt` from the standard math library). It can thus be used on any platform for which a C compiler is available. The algorithm implemented is the same as in Vanderberghe's CVXOPT [11], [12]: a standard primal-dual Mehrotra predictor-corrector interior-point method with self-dual embedding, which allows for detection of infeasibility and unboundedness. CVXOPT obtains the search directions via a Cholesky factorization of the reduced system; in contrast, we use a sparse LDL solver on a variation of the standard KKT system. All standard sparse LDL solvers for indefinite linear systems use on-the-fly pivoting for numerical stability. This numerical pivoting step, however, is disadvantageous on embedded platforms due to code complexity and the need for dynamic memory allocation. Instead, we use a fixed elimination ordering, chosen once on the basis of the problem's sparsity structure. Consequently, no dynamic pivoting is performed. To ensure the factorization exists, and to enhance numerical stability, we use dynamic regularization and iterative refinement, as is done in CVXGEN [13]. Our current implementation uses a fill-in reducing ordering, based on the approximate minimum degree heuristic [14]. We also handle the diagonal plus rank one blocks appearing in the linear system by expanding them to larger but sparse blocks, improving solver efficiency when optimizing over large second-order cones (SOCs). The expansion is chosen in such a way that the lifted KKT matrix can be factored in a numerically stable way, despite a fixed pivoting sequence.

With these techniques, ECOS is a low footprint, high performance SOCP solver that is numerically reliable for accuracies beyond what is typically needed in embedded applications. Despite its simplicity, ECOS solves small problems more quickly than most existing SOCP solvers, and is competitive for medium-sized problems up to about 20 k variables. A Python interface, a native Matlab MEX interface and integration with CVX, a Matlab package for specifying and solving convex programs [15], is provided. ECOS is available from `github.com/ifa-ethz/ecos`.

A. Related work

Several solvers that target embedded systems have been made available recently. With exception of first order meth-

[1] Automatic Control Laboratory, ETH Zurich, Zurich 8092, Switzerland. Email: `domahidi@control.ee.ethz.ch`
[2] Department of Electrical Engineering, Stanford University, Stanford CA 94305, USA. Email: `echu508|boyd@stanford.edu`

ods (cf. [16], [17]), all existing implementations are limited to QPs, including efficient primal-barrier methods for model predictive control [18], primal-dual interior-point methods by C code-generation for small QPs with micro to millisecond solution times (CVXGEN, [13]) and active set strategies with good warm-starting capabilities (qpOASES, [19]). QPs, QCQPs as well as LPs with multistage property can be solved efficiently by FORCES [20], [21]. There are currently two implementations of C-code generators based on first order methods: μAO-MPC [22], [23] for linear MPC problems, and FiOrdOs [24] for general convex problems, which also supports SOCPs. First-order methods can be slow if the problem is not well conditioned or if it has a feasible set that does not allow for an efficient projection, while interior-point methods have a convergence rate that is independent of the problem data and the particular feasible set.

B. Outline

In §II, we define an SOCP. In §III, we outline an interior-point method (implemented in CVXOPT [11]). §IV describes modifications to the interior-point method in order to target embedded systems. §V presents a numerical example and run time comparison to existing solvers. §VI concludes the paper.

II. Second-Order Cone Problem

ECOS solves SOCPs in the standard form [11]:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & Gx + s = h, \quad s \in \mathcal{K}, \end{array} \tag{P}$$

where x are the primal variables, s denotes slack variables, $c \in \mathbf{R}^n$, $G \in \mathbf{R}^{M \times n}$, $h \in \mathbf{R}^M$, $A \in \mathbf{R}^{p \times n}$, $b \in \mathbf{R}^p$ are the problem data, and $\mathcal{K}$ is the cone

$$\mathcal{K} = \mathcal{Q}^{m_1} \times \mathcal{Q}^{m_2} \times \cdots \times \mathcal{Q}^{m_N}, \tag{1}$$

where

$$\mathcal{Q}^m \triangleq \left\{ (u_0, u_1) \in \mathbf{R} \times \mathbf{R}^{m-1} \mid u_0 \geq \|u_1\|_2 \right\}$$

for $m > 1$, and we define $\mathcal{Q}^1 \triangleq \mathbf{R}_+$. The *order* of the cone $\mathcal{K}$ is denoted by $M \triangleq \sum_{i=1}^N m_i$. Without loss of generality, we assume that the first $l \geq 0$ cones in (1) correspond to the positive orthant $\mathcal{Q}^1$. The associated dual problem to (P) is (see [1, Chap. 5])

$$\begin{array}{ll} \text{maximize} & -b^T y - h^T z \\ \text{subject to} & G^T z + A^T y + c = 0 \\ & z \in \mathcal{K}. \end{array} \tag{D}$$

III. Interior-Point Method

Interior point methods (IPMs) are widely used to solve convex optimization problems, see e.g. [1], [8], [25], [26]. Conceptually, a barrier function, e.g.

$$\Phi(u) \triangleq \sum_{i=1}^N \begin{cases} -\ln u_i & u_i \in \mathcal{Q}^1 \\ -\frac{1}{2} \ln \left(u_{i,0}^2 - u_{i,1}^T u_{i,1} \right) & u_i \in \mathcal{Q}^m \end{cases}, \tag{2}$$

is employed for $\mathcal{K}$ to replace the constrained optimization problems (P) and (D) by a series of smooth convex *unconstrained* problems, each approximately solved by Newton's method in one step. Thereby a sequence

$$\chi^{k+1} = \chi^k + \alpha^k \Delta\chi^k, \quad k = 0, 1, 2, \ldots \tag{3}$$

is generated, where $\chi^k \triangleq (x^k, y^k, s^k, z^k)$, $\alpha^k > 0$ is a step length found by line search and $\Delta\chi^k$ is a particular search direction found by solving one or more linear systems. In path-following algorithms, iterates (3) loosely track the *central path*, which ends in the solution set [25]. Details on the particular algorithm implemented by ECOS can be found in [27]; we give the most important aspects in the following.

A. Extended self-dual homogeneous embedding

To detect and handle infeasibility or unboundedness, we embed (P) and (D) into a single, self-dual problem that readily provides certificates of infeasibility, cf. [8], [11], [26]:

$$\begin{array}{ll} \text{minimize} & (M+1)\theta \\ \text{subject to} & 0 = A^T y + G^T z + c\tau + q_x \theta \\ & 0 = -Ax + b\tau + q_y \theta \\ & s = -Gx + h\tau + q_z \theta \\ & \kappa = -c^T x - b^T y - h^T z + q_\tau \theta \\ & 0 = -q_x^T x - q_y^T y - q_z^T z - q_\tau \tau + M + 1 \\ & (s, z) \in \mathcal{K}, \quad (\tau, \kappa) \geq 0, \end{array} \tag{ESD}$$

with additional variables τ, κ and θ, and with

$$(q_x, q_y, q_z, q_\tau) = \frac{M+1}{s^T z + \tau\kappa} (r_x, r_y, r_z, r_\tau),$$

where

$$\begin{aligned} r_x &\triangleq -A^T y - G^T z - c\tau, & r_y &\triangleq Ax - b\tau \\ r_\tau &\triangleq \kappa + c^T x + b^T y + h^T z, & r_z &\triangleq s + Gx - h\tau, \end{aligned} \tag{4}$$

denote residuals given $(x, y, z, s, \tau, \kappa)$. Using this self-dual embedding, the following certificates can be provided when $\theta = 0$:

1) $\tau > 0$, $\kappa = 0$: Optimality,
2) $\tau = 0$, $\kappa > 0$ and $h^T z + b^T y < 0$: Primal infeasibility,
3) $\tau = 0$, $\kappa > 0$ and $c^T x < 0$: Dual infeasibility,
4) $\tau = 0$, $\kappa = 0$: None.

B. Central path

The central path of problem (ESD) is defined by the set of points $(x, y, s, z, \tau, \kappa)$ satisfying

$$\begin{aligned} &\begin{bmatrix} 0 \\ 0 \\ s \\ \kappa \end{bmatrix} = \begin{bmatrix} 0 & A^T & G^T & c \\ -A & 0 & 0 & b \\ -G & 0 & 0 & h \\ -c^T & -b^T & -h^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \tau \end{bmatrix} + \mu \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_\tau \end{bmatrix} \\ &(s, z) \in \mathcal{K}, \quad (\kappa, \tau) \geq 0, \\ &\left(W^{-T} s\right) \circ (Wz) = \mu \mathbf{e}, \quad \tau\kappa = \mu, \end{aligned} \tag{CP}$$

where $\mu \triangleq (s^T z + \kappa\tau)/(M+1) = \theta$ and W is a (symmetric) *scaling* matrix (cf. §III-C). Search directions are generated by linearizing (CP) and choosing different μ, cf. §III-D. For $\mu \to 0$, (CP) is equivalent to the Karush-Kuhn-Tucker (KKT)

optimality conditions, which are necessary and sufficient conditions for optimality for convex problems.

In (CP), the operator $\circ$ denotes a cone product of two vectors $u, v \in \mathcal{Q}^m$, and $\mathbf{e}$ is the unit element such that $u \circ \mathbf{e} = u$ for all $u \in \mathcal{Q}^m$. For $m = 1$, this is the standard multiplication (and $\mathbf{e} = 1$), while for $m > 1$ we have

$$u \circ v = \begin{bmatrix} u^T v \\ u_0 v_1 + v_0 u_1 \end{bmatrix} \text{ and } \mathbf{e} = \begin{bmatrix} 1 & 0_{m-1}^T \end{bmatrix}^T. \quad (5)$$

The operator $\circ$ stems from the unifying treatment of symmetric cones using real Jordan algebra. The interested reader is referred to [28] and the references therein.

C. Symmetric scalings

Nesterov and Todd showed that the self-scaled property of the barrier function Φ (2) can be exploited to construct interior-point methods that tend to make long steps along the search direction [29]. A primal-dual scaling W is a linear transformation [11]

$$\tilde{s} = W^{-T} s, \qquad \tilde{z} = W z$$

that leaves the cone and the central path invariant, i.e.,

$$s \in \mathcal{K} \Leftrightarrow \tilde{s} \in \mathcal{K}, \quad z \in \mathcal{K} \Leftrightarrow \tilde{z} \in \mathcal{K},$$
$$s \circ z = \mu \mathbf{e} \Leftrightarrow \tilde{s} \circ \tilde{z} = \mu \mathbf{e}.$$

We use the symmetric *Nesterov-Todd (NT)* scaling that is defined by the scaling point $w \in \mathcal{K}$ such that

$$\nabla^2 \Phi(w) s = z,$$

i.e. w is the point where the Hessian of the barrier function maps s onto z. It can be shown that such w always exists and that it is unique [29]. The NT-scaling W is then obtained by a symmetric factorization of $\nabla^2 \Phi(w)^{-1} = W^T W$ [11]:

1) NT-scaling for LP-cone $\mathcal{Q}^1$: For $s, z \in \mathcal{Q}^1$,

$$W_+ = \sqrt{s/z}. \quad (6)$$

This is the standard scaling used in primal-dual interior point methods for LPs and QPs.

2) NT-scaling for SOC $\mathcal{Q}^m$, $m > 1$: For $s, z \in \mathcal{Q}^m$ define the normalized points

$$\bar{z} = \frac{z}{(z_0^2 - z_1^T z_1)^{1/2}}, \qquad \bar{s} = \frac{s}{(s_0^2 - s_1^T s_1)^{1/2}},$$
$$\gamma = \left(\frac{1 + \bar{z}^T \bar{s}}{2} \right)^{1/2}, \qquad \bar{w} = \frac{1}{2\gamma} \left(\bar{s} + \begin{bmatrix} \bar{z}_0 \\ -\bar{z}_1 \end{bmatrix} \right).$$

Then the NT-scaling is given by

$$W_{\text{SOC}} \triangleq \eta \begin{bmatrix} \bar{w}_0 & \bar{w}_1^T \\ \bar{w}_1 & I_{m-1} + (1 + \bar{w}_0) \bar{w}_1 \bar{w}_1^T \end{bmatrix}$$
$$\eta \triangleq \left((s_0^2 - s_1^T s_1) / (z_0^2 - z_1^T z_1) \right)^{1/4} \quad (7)$$

with I_{m-1} being the identity matrix of size $m - 1$.

3) Composition of scalings: The final scaling matrix W is a block diagonal matrix comprised of scalings for each cone $\mathcal{Q}^{m_i}$,

$$W = \text{blkdiag}\left(W_1, W_2, \ldots, W_N \right),$$

where the first l blocks $W_{1:l}$ are defined by (6) and the remaining $N - l$ ones by (7). Note that $W = W^T$ for the scalings considered in this paper.

D. Search directions

In predictor-corrector methods, the overall search direction is the sum of three directions [25]: The *affine search* direction aims at directly satisfying the KKT conditions, i.e. $\mu = 0$ in (CP), and allows one to make progress (in terms of distance to the optimal solution) at the current step. The *centering direction* does not make progress in the current step, but brings the current iterate closer to the central path, such that larger progress can be made at the next affine step. The amount of centering is determined by a parameter $\sigma \in [0, 1]$. Since the affine direction is based on linearization, the *corrector direction* aims at compensating the nonlinearities at the predicted affine step. The centering and the corrector direction can be pooled into a *combined* direction.

As suggested in [11], search directions can be computed as follows. Define the vectors

$$\Delta\xi_1 \triangleq \begin{bmatrix} \Delta x_1^T & \Delta y_1^T & \Delta z_1^T \end{bmatrix}^T,$$
$$\Delta\xi_2 \triangleq \begin{bmatrix} \Delta x_2^T & \Delta y_2^T & \Delta z_2^T \end{bmatrix}^T,$$
$$\beta_1 \triangleq \begin{bmatrix} -c^T & b^T & h^T \end{bmatrix}^T,$$
$$\beta_2 \triangleq \begin{bmatrix} d_x^T & d_y^T & d_z^T \end{bmatrix}^T,$$

and the matrix

$$K \triangleq \begin{bmatrix} 0 & A^T & G^T \\ A & 0 & 0 \\ G & 0 & -W^2 \end{bmatrix}, \quad (8)$$

which we refer to as the *KKT matrix* in this paper, as it follows from the coefficient matrix in the central path equation (CP) after simple algebraic manipulations. A search direction can now be computed by solving two linear systems,

$$K \Delta\xi_1 = \beta_1 \quad \text{and} \quad K \Delta\xi_2 = \beta_2, \quad (9)$$

and combining the results to

$$\Delta\tau = \frac{d_\tau - d_\kappa/\tau + \begin{bmatrix} c^T & b^T & h^T \end{bmatrix} \Delta\xi_1}{\kappa/\tau - \begin{bmatrix} c^T & b^T & h^T \end{bmatrix} \Delta\xi_2}, \quad (10a)$$

$$\begin{bmatrix} \Delta x^T & \Delta y^T & \Delta z^T \end{bmatrix}^T = \Delta\xi_1 + \Delta\tau \Delta\xi_2, \quad (10b)$$

$$\Delta s = -W \left(\lambda \setminus d_s + W \Delta z \right), \quad (10c)$$

$$\Delta\kappa = -\left(d_\kappa + \kappa \Delta\tau \right) / \tau, \quad (10d)$$

where $u \setminus w$ is the inverse operator to the conic product $\circ$, such that if $u \circ v = w$ then $u \setminus w = v$ (we give an efficient formula in (16)), and

$$\lambda \triangleq W^{-1} s = W z.$$

TABLE I

RHS FOR AFFINE AND COMBINED SEARCH DIRECTION. Δs_a AND Δz_a DENOTE AFFINE SEARCH DIRECTIONS.

RHS	affine	combined (centering & corrector)
d_x	r_x	$(1-\sigma)r_x$
d_y	r_y	$(1-\sigma)r_y$
d_z	$-r_z + s$	$-(1-\sigma)r_z + W(\lambda \setminus d_s)$
d_s	$\lambda \circ \lambda$	$\lambda \circ \lambda + \left(W^{-1}\Delta s_a\right) \circ (W \Delta z_a) - \sigma\mu\mathbf{e}$
d_τ	r_τ	$(1-\sigma)r_\tau$
d_κ	$\tau\kappa$	$\tau\kappa + \Delta s_a \Delta z_a - \sigma\mu$

The corresponding right hand sides for the affine and centering-corrector direction are summarized in Table I. For the former, (10c) simplifies to

$$\Delta s_a = -W\left(\lambda + W\Delta z_a\right).$$

Since β_1 is the same for both directions, only three linear systems have to be solved per iteration. This is the computational bottleneck of interior-point methods.

IV. IMPLEMENTATION FOR EMBEDDED SYSTEMS

In this section, we give implementation details which differ from CVXOPT. These are important for the solver to be run on embedded platforms. Some of the techniques described here have already been successfully employed in CVXGEN [13].

A. Efficient and stable sparse LDL factorization

Each iteration of our method solves the indefinite linear system (9) with different righthand sides. Typically, this is solved by factorizing K or an equivalent linear system. The computational cost thus depends on the number of nonzeros in this factorization.

We use the following strategies to minimize this overhead: (1) we work with the indefinite matrix K and avoid potential fill-in when performing a block reduction of K (as is done in CVXOPT), (2) we use a fill-in reducing ordering on K, (3) we perform *dynamic regularization* to ensure the existence of a factorization for *any* static ordering, and (4) we use *iterative refinement* to undo the effects of dynamic regularization. These strategies allow us to compute the permutation and symbolic factorization in an initialization stage. We modify Tim Davis' SparseLDL [30] to perform dynamic regularization and iterative refinement.

1) LDL factorization of indefinite K: We perform a (permuted) LDL factorization on K,

$$K = PLDL^TP^T,$$

where P is a permutation matrix of appropriate size, chosen to minimize the fill-in of L; L is a lower triangular matrix; and D is a diagonal matrix. Note that this differs from a general LDL factorization in which D may contain 2×2 blocks [31]. Since K is indefinite, this (simpler) factorization is not guaranteed to exist; hence, we introduce dynamic regularization to turn K into a quasi-definite matrix [32].

2) Permutations to reduce fill-in: The factor L has at least as many non-zero elements as the lower triangular part of K. We use the approximate minimum degree (AMD) code [14] to compute fill-in reducing orderings of K. AMD is a heuristic that performs well in practice and has low computational complexity.

3) Dynamic regularization: We turn K into a quasi-definite matrix by perturbing its diagonal entries. We do this only when needed during the numerical factorization, hence the term *dynamic* regularization. Specifically, we alter D_{ii} during the factorization whenever it becomes too small or has the wrong sign:

$$D_{ii} \leftarrow S_i\delta \quad \text{if} \quad S_iD_{ii} \leq \epsilon,$$

with parameters $\epsilon \approx 10^{-14}$, $\delta \approx 10^{-7}$. The signs S_i of the diagonal elements are known in advance,

$$S = \begin{bmatrix} +\mathbf{1}_n^T & -\mathbf{1}_p^T & -\mathbf{1}_l^T & \rho^T \end{bmatrix},$$

where ρ is the sign vector associated to SOCs. For K as defined in (8), $\rho = -\mathbf{1}_{M-l}^T$. For the sparse representation of K introduced in §IV-B.1 and used in the current implementation of ECOS,

$$\rho = \begin{bmatrix} \rho_{l+1}, \ldots, \rho_N \end{bmatrix} \quad \text{with} \quad \rho_i = \begin{bmatrix} -\mathbf{1}_{m_i}^T & -1 & 1 \end{bmatrix}$$

for each SOC.

4) Iterative refinement (IR): Due to regularization, we solve a slightly perturbed system $(K + \Delta K)\,\tilde{\xi}_i = \beta_i$, and thus make an error $\beta_i - K\tilde{\xi}_i$ with respect to (9). To compensate for this error, we solve

$$(K + \Delta K)\,\Delta\tilde{\xi}_i = \beta_i - K\tilde{\xi}_i$$

for a corrective term $\Delta\tilde{\xi}_i$, and set $\tilde{\xi}_i \leftarrow \tilde{\xi}_i + \Delta\tilde{\xi}_i$. This refinement requires only a forward- and a backward solve, since the factorization of $(K + \Delta K)$ has been computed already. Successive application of iterative refinement converges to the true solution of (9).

B. Structure exploitation of SOC scalings

The Nesterov-Todd scalings (7), which propagate to the (3,3) block in the KKT matrix K as $-W_{\text{soc}}^2$, are dense matrices involving a diagonal plus rank one term. We exploit this to obtain a sparse KKT system and to accelerate scaling operations. This is crucial for efficiently solving problems with cones of large dimension.

1) Sparse KKT system: We consider a single cone (for multiple cones, this construction can be carried out for each SOC separately). Based on the observation that W_{SOC}^2 can be rewritten as

$$W_{\text{soc}}^2 \equiv \eta^2\left(\mathcal{D} + uu^T - vv^T\right) \tag{11}$$

for some diagonal matrix $\mathcal{D}$ and vectors u, v, we introduce two scalar variables q and t to obtain the sparse representation of the last equation in (9):

$$\begin{bmatrix} G \\ 0 \\ 0 \end{bmatrix} \Delta x - \underbrace{\eta^2 \begin{bmatrix} \mathcal{D} & v & u \\ v^T & 1 & 0 \\ u^T & 0 & -1 \end{bmatrix}}_{\tilde{W}_{\text{SOC}}^2} \begin{bmatrix} \Delta z \\ q \\ t \end{bmatrix} = \begin{bmatrix} d_z \\ 0 \\ 0 \end{bmatrix}. \tag{12}$$

As a result, the KKT matrix is sparse (given that A and G are sparse), which is likely to yield sparse Cholesky factors and thereby significant computational savings if large cones are involved.

For any fixed permutation P, the condition

$$\mathcal{D} - vv^T \succ 0 \tag{13}$$

is sufficient for the factorization of the expanded KKT matrix to exist, because (13) implies that the coefficient matrix $\tilde{W}^2_{\text{SOC}}$ in (12) is quasi-definite. Consequently, the expanded KKT matrix is also quasi-definite (if regularized as described in §IV-A.3). It can be shown that the following choice of $\mathcal{D}$, u, and v satisfies both (11) and (13), and that it is always well defined:

$$\begin{aligned}
&\mathcal{D} \triangleq \begin{bmatrix} d & \\ & I_{m-1} \end{bmatrix}, \quad u \triangleq \begin{bmatrix} u_0 \\ u_1 \bar{w}_1 \end{bmatrix}, \quad v \triangleq \begin{bmatrix} 0 \\ v_1 \bar{w}_1 \end{bmatrix}, \\
&d \triangleq \frac{1}{2}\bar{w}_0^2 + \frac{1}{2}\bar{w}_1^T\bar{w}_1 \left(1 - \frac{\alpha^2}{1 + (\bar{w}_1^T\bar{w}_1)\beta}\right), \\
&u_0 \triangleq \sqrt{\bar{w}_0^2 + \bar{w}_1^T\bar{w}_1 - d}, \quad u_1 \triangleq \frac{\alpha}{u_0}, \quad v_1 \triangleq \sqrt{u_1^2 - \beta}, \\
&\alpha \triangleq 1 + \bar{w}_0 + (\bar{w}_1^T\bar{w}_1)/(1 + \bar{w}_0), \\
&\beta \triangleq 1 + 2/(1 + \bar{w}_0) + \bar{w}_1^T\bar{w}_1/(1 + \bar{w}_0)^2.
\end{aligned}$$

2) Fast scaling operations: With the definitions in (7), we can apply the NT-scaling W_{SOC} by

$$W_{\text{SOC}}\, z = \eta \begin{bmatrix} \bar{w}_0 z_0 + \zeta \\ z_1 + (z_0 + \zeta/(1 + \bar{w}_0))\, \bar{w}_1 \end{bmatrix}, \tag{14}$$

where $\zeta \triangleq \bar{w}_1^T z_1$. Similarly, we can apply the inverse scaling W^{-1}_{SOC} by

$$W^{-1}_{\text{SOC}}\, z = \frac{1}{\eta} \begin{bmatrix} \bar{w}_0 z_0 - \zeta \\ z_1 + (-z_0 + \zeta/(1 + \bar{w}_0))\, \bar{w}_1 \end{bmatrix}. \tag{15}$$

3) Fast inverse conic product: An efficient formulation of the inverse operator to $\circ$ (5) for second-order cones is given by:

$$\begin{aligned}
u \setminus w &\triangleq \frac{1}{\varrho} \begin{bmatrix} u_0 w_0 - \nu \\ (\nu/u_0 - w_0)u_1 + (\varrho/u_0)\, w_1 \end{bmatrix}, \\
\text{with } \varrho &= u_0^2 - u_1^T u_1 \ , \quad \nu = u_1^T w_1.
\end{aligned} \tag{16}$$

For $\mathcal{Q}^1$, $u \setminus w \triangleq w/u$, i.e. the usual scalar division.

By using (14), (15) and (16), a multiplication or left-division by W and the inverse conic product require $\mathcal{O}(m)$ instead of $\mathcal{O}(m^2)$ operations, as in the LP case. This is significant if the dimension of the cone is large.

C. Setup & solve: Dynamic and static memory

ECOS implements three functions: *Setup*, *Solve* and *Cleanup*. *Setup* allocates memory and creates the upper triangular part of the expanded sparse KKT matrix $\tilde{K}$ as well as the sign vector S needed for dynamic regularization. Then, a permutation of $\tilde{K}$ is computed using AMD, and $\tilde{K}$ and S are permuted accordingly. Last, a symbolic factorization of $P^T\tilde{K}P$ is carried out to determine the data structures needed for numerical factorization in *Solve*.

Our *Solve* routine implements the same interior-point algorithm as in CVXOPT [27, §7], but search directions are found as described in this Section above. We use the same initialization method and stopping criteria as CVXOPT [27, §7.3]. If the data structure for ECOS is no longer needed, the user can call the *Cleanup* routine to free memory that has been allocated during *Setup*.

Only *Setup* and *Cleanup* call `malloc` and `free`, while *Solve* uses only static memory allocation. Once the problem has been set up (possibly accomplished by a code generator), different problem *instances* can be solved without the need for dynamic memory allocation (and without the *Setup* overhead).

V. Example: Portfolio Optimization

We consider a simple long-only portfolio optimization problem [1, p. 185–186], where we choose relative weights of assets to maximize risk-adjusted return. The problem is

$$\begin{array}{ll} \text{maximize} & \mu^T x - \gamma(x^T \Sigma x) \\ \text{subject to} & \mathbf{1}^T x = 1, \quad x \geq 0, \end{array}$$

where the variable $x \in \mathbf{R}^n$ represents the portfolio, $\mu \in \mathbf{R}^n$ is the vector of expected returns, $\gamma > 0$ is the risk-aversion parameter, and $\Sigma \in \mathbf{R}^{n \times n}$ is the asset return covariance, given in factor model form,

$$\Sigma = FF^T + D.$$

Here $F \in \mathbf{R}^{n \times m}$ is the factor-loading matrix, and $D \in \mathbf{R}^{n \times n}$ is a diagonal matrix (of *idiosyncratic risk*). The number of factors in the risk model is m, which we assume is substantially smaller than n, the number of assets.

This problem can be converted in the standard way into an SOCP,

$$\begin{array}{ll} \text{minimize} & -\mu^T x + \gamma(t + s) \\ \text{subject to} & \mathbf{1}^T x = 1, \quad x \geq 0 \\ & \|D^{1/2}x\|_2 \leq u, \quad \|F^T x\|_2 \leq v \\ & \|(1 - t, 2u)\|_2 \leq 1 + t \\ & \|(1 - s, 2v)\|_2 \leq 1 + s \end{array} \tag{17}$$

with variables $x \in \mathbf{R}^n$ and $(t, s, u, v) \in \mathbf{R}^4$.

A run time comparison of existing SOCP solvers and ECOS for obtaining solutions to random instances of (17) with accuracy 10^{-6} is given in Fig. 1. The solvers MOSEK and Gurobi were run in single-threaded mode (the problems were solved more quickly than with multiple threads), while the current versions of SDPT3 and SeDuMi do not have the option to change the number of threads. For these solvers, we observed a CPU usage of more than 100%, which indicates that SDPT3 and SeDuMi make use of multiple threads. We switched off printing and used standard values for all other solver options. The number of iterative refinement steps for ECOS was limited to one.

As Fig. 1 shows, ECOS outperforms most established solvers for small problems, and is overall competitive for medium-sized problems up to several thousands of variables.

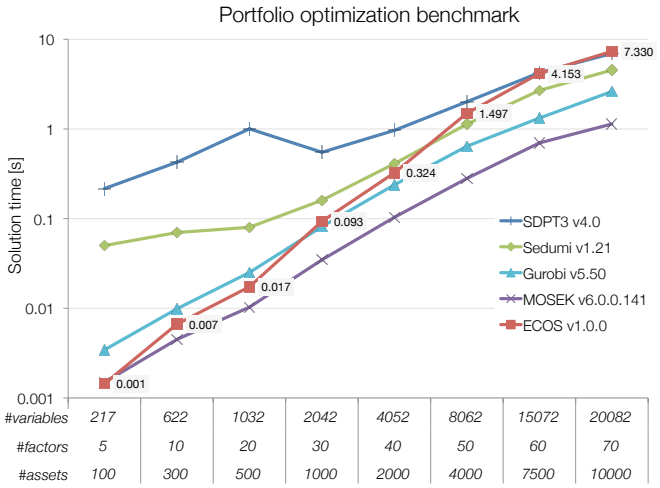

Fig. 1. Timing results for portfolio problem (17) on a MacBook Pro with Intel Core i7 2.6 GHz CPU.

VI. Conclusion

We have presented ECOS, an efficient implementation of an interior-point solver for SOCPs targeting embedded systems. Using a particularly chosen and (dynamically) regularized symmetric indefinite KKT system, search directions can be found stably by sparse LDL factorization with fixed pivoting. This avoids dynamic memory allocation during solves and results in only 750 lines of source code for the solver, including all linear algebra functionality. This makes ECOS very attractive for use on embedded systems; in fact, it could be certified for code safety (such as for no divisions by zero) with reasonable effort. Despite this simplicity, ECOS is competitive to established desktop solvers for small and medium-sized problems.

Acknowledgment

We thank M. N. Zeilinger and P. J. Goulart for helpful discussions, M. Morari for bringing the authors together, and L. Vandenberghe for making CVXOPT publicly available. The research leading to these results was supported by the EU in FP7 via EMBOCON (ICT-248940), by DARPA via XDATA FA8750-12-2-0306 and by NASA through grant NNX07AEIIA.

References

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

[2] J. A. d. D. Graham Goodwin, María M. Seron, *Constrained Control and Estimation: An Optimisation Approach*. Springer London, 2004.

[3] Z.-Q. Luo and W. Yu, "An introduction to convex optimization for communications and signal processing," *IEEE Jnl. on Selected Areas in Comm.*, vol. 24, no. 8, pp. 1426–1438, Aug. 2006.

[4] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans. on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[5] J. F. Sturm, "Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 625–653, 1999.

[6] K. C. Toh, M. J. Todd, and R. H. Tütüncü, "SDPT3 — A Matlab software package for semidefinite programming, Version 1.3," *Opt. Methods and Software*, vol. 11, no. 1-4, pp. 545–581, 1999.

[7] *Gurobi optimizer reference manual version 5.0*, Gurobi Optimization, Inc., Houston, Texas, July 2012.

[8] E. Andersen and K. Andersen, "The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm," *High Performance Optimization*, vol. 33, pp. 197–232, 2000.

[9] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of second-order cone programming," *Linear Algebra and its Applications*, vol. 284, no. 1-3, pp. 193–228, 1998.

[10] B. Acikmese and S. R. Ploen, "Convex programming approach to powered descent guidance for Mars landing," *Journal of Guidance, Control and Dynamics*, vol. 30, no. 5, pp. 1353–1366, 2007.

[11] M. S. Andersen, J. Dahl, and L. Vandenberghe, "CVXOPT: A python package for convex optimization, version 1.1.5," abel.ee.ucla.edu/cvxopt, 2012.

[12] M. Andersen, J. Dahl, Z. Liu, and L. Vandenberghe, "Interior-point methods for large-scale cone programming," in *Optimization for Machine Learning*. MIT Press, 2011.

[13] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, pp. 1–27, 2012.

[14] P. Amestoy, T. Davis, and I. Duff, "Algorithm 837: AMD, an approximate minimum degree ordering algorithm," *ACM Trans. on Math. Software*, vol. 30, no. 3, pp. 381–388, 2004.

[15] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming," http://cvxr.com/cvx, Apr. 2011.

[16] S. Richter, "Computational complexity certification of gradient methods for real-time model predictive control," Ph.D. dissertation, ETH Zurich, Switzerland, 2012.

[17] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for linear model predictive control," in *Conference on Decision and Control (CDC), Maui, HI, USA*, December 2012, pp. 662–667.

[18] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, March 2010.

[19] H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *Int. Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.

[20] A. Domahidi, A. Zgraggen, M. Zeilinger, M. Morari, and C. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *Conference on Decision and Control (CDC)*, Maui, HI, USA, Dec. 2012, pp. 668–674.

[21] A. Domahidi, "FORCES: Fast optimization for real-time control on embedded systems," http://forces.ethz.ch, Oct. 2012.

[22] P. Zometa and R. Findeisen, "μAO-MPC: Microcontroller applications online model predictive control," http://www.et.uni-magdeburg.de/syst/research/muAO-MPC/, 2012.

[23] ——, "Code generation of linear model predictive control for real-time embedded applications," in *Proceedings of the American Control Conference*, Washington, D.C., USA, 2013.

[24] F. Ullmann and S. Richter, "FiOrdOs: Code generation for first-order methods," http://fiordos.ethz.ch, May 2012.

[25] S. J. Wright, *Primal-dual Interior Point Methods*. SIAM, 1997.

[26] Y. Ye, M. J. Todd, and S. Mizuno, "An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm," *Mathematics of Operations Research*, vol. 19, no. 1, pp. 53–67, Feb. 1994.

[27] L. Vandenberghe, "The CVXOPT linear and quadratic cone program solvers." Online: http://cvxopt.org/documentation/coneprog.pdf, March 2010.

[28] G. Gu, M. Zangiabadi, and C. Roos, "Full Nesterov-Todd step infeasible interior-point method for symmetric optimization," *Eur. Jnl. of Operational Research*, vol. 214, no. 3, pp. 473–484, 2011.

[29] Y. E. Nesterov and M. J. Todd, "Self-scaled barriers and interior-point methods for convex programming," *Mathematics of Operations Research*, vol. 22, no. 1, pp. 1–42, 1997.

[30] T. Davis, "Algorithm 849: A concise sparse Cholesky factorization package," *ACM Transactions on Mathematical Software (TOMS)*, vol. 31, no. 4, pp. 587–591, 2005.

[31] J. Bunch and B. Parlett, "Direct methods for solving symmetric indefinite systems of linear equations," *SIAM Journal on Numerical Analysis*, vol. 8, no. 4, pp. 639–655, 1971.

[32] R. Vanderbei, "Symmetric quasidefinite matrices," *SIAM Journal on Optimization*, vol. 5, no. 1, pp. 100–113, 1995.