Notes on a possible CCSA–LOBPCG hybrid for nonlinear trace-optimization problems

Steven G. Johnson

Created February 2021; updated February 13, 2021.

1 Overview

For inverse-design problems involving random emission, random scattering, or other forms of incoherent waves, the problems often take the form

$$\begin{aligned} &\min_{p \in \mathbb{R}^P} \operatorname{tr}\left[A(p)\right] \\ &f_i(p) \le 0,\ i = 1 \dots M \end{aligned} \tag{1}$$

where $A(p)$ is a real-symmetric (or Hermitian) $N \times N$ matrix-valued function and the $f_i(p)$ are m nonlinear constraints, both f_i and A being smooth functions of the design variables p, with a non-empty feasible set. (Often the physics is formulated as trace *maximization*, but of course this is equivalent to trace minimization if we flip the sign of A.) In many cases, $A(p)$ will be negative-semidefinite in addition to being Hermitian.

Our goal is to devise an efficient algorithm to solve problems of this form (1), combining specialized Krylov eigenvalue methods from linear algebra with more general algorithms for nonlinear optimization, in such a way as to *simultaneously* iterate design parameters p and the trace estimate. In particular, we will consider a possible new algorithm based on a hybrid of the CCSA nonlinear-optimization algorithm (popular in topology optimization for its extreme flexibility and the fact that it yields *feasible* iterates p at intermediate steps) and the LOBPCG eigensolver algorithm (which is already employed in density functional theory for nonlinear optimization problems that only asymptotically approach linear eigenproblems).

1.1 Trace estimation during optimization

In particular, we are interested in situations where $A(p)$ is a large matrix that *cannot be computed explicitly*: typically it involves a matrix inverse or some other matrix function, and the only operation we are allowed is multiplying A by $\ll N$ vectors. In consequence, we must **estimate** the trace.

There are many trace-estimation procedures in the literature, often involving Monte-Carlo methods for $\operatorname{tr}[A] = E[x^* A x]$, where the expectation value

E is computed over a suitable distribution of vectors x (e.g. with independent random components ± 1, the "Hutchinson" trace estimator). One approach to combining this with optimization (1) over p would be to use a **stochastic optimization** algorithm (such as the *Adam* algorithm combined with an augmented-Lagrangian formulation of the constraints).

$$\begin{min_{p\in\mathbb{R}^P} E\left[x^* A(p) x\right] \\ f_i(p) \le 0,\ i = 1 \dots M}$$

where on each optimization step we might estimate the expected value by sampling a finite number of x vectors.

In our particular situation, however, a key fact is that we don't necessarily need a good trace estimate for an *arbitrary* matrix A. Instead, we only need our trace estimate to be accurate for the *optimum* matrix $A(p_*)$ at the optimal parameters p_* from our optimization problem (1). It turns out that, in our physical systems, this optimum matrix is very special: **the optimized trace is typically dominated by a handful of (large and negative) eigenvalues** of A, the strong **resonances** of our wave system that arise from optimization. So, we can formulate the trace estimation in terms of methods for computing the $K \ll N$ (algebraically) smallest eigenvalues of A.

In particular, suppose we order the eigenvalues of A as $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_N$. Then

$$\mathrm{tr}[A] = \sum_{j=1}^{N} \lambda_j \approx \sum_{j=1}^{K} \lambda_j = \min_{X \in \mathbb{R}^{N\times K}} \mathrm{tr}\left[X^T A X (X^T X)^{-1}\right],$$

using a block version of the min–max theorem (the Rayleigh-quotient bound on the minimum eigenvalues; technically this is an infimum over full-rank X's). Morover, if A is negative-definite (which is often the case for our problems), then this $\sum_{j=1}^{K} \lambda_j$ is actually an **upper bound** for our trace. Furthermore, we can *combine* the minimization over the subspace basis X with the minimization over the design parameters p into a *single* optimization problem:

$$\boxed{\begin{min_{p\in\mathbb{R}^P, X\in\mathbb{R}^{N\times K}} \mathrm{tr}\left[X^T A(p) X (X^T X)^{-1}\right] \\ f_i(p) \le 0,\ i = 1 \dots M}}. \tag{2}$$

2 Subspace-trace optimization

The "**subspace-trace optimization**" problem (2) has the advantage that it can then be attacked with any constrained nonlinear optimization problem. The objective function

$$f_0(p, X) = \mathrm{tr}\left[X^T A(p) X (X^T X)^{-1}\right] \tag{3}$$

only requires $K \ll N$ matrix–vector multiplications with $A(p)$ on each step, and minimizing f_0 over p and X **simultaneously** means that we don't waste time

computing tr[A] to high accuracy for each intermediate p step—we gradually improve our estimate of the trace at the same time as we improve our design parameters p.

Of course, the best optimization algorithms typically require us to supply gradients of our objective and constraints f_i ($i = 0, \ldots, M$). The gradients $\nabla_p f_i$ can normally be computed efficiently with the help of standard adjoint methods. The gradient $\nabla_X f_0$ has a simple analytical form:

$$\nabla_X f_0 = (I - XMX^T)AXM \tag{4}$$

where $M = (X^T X)^{-1}$.

However, it is important to realize that, for a fixed p, minimizing $f_0(p, X)$ over X is equivalent to solving for the K algebraically smallest eigenvalues and the corresponding eigenvectors. If we use a completely "generic" nonlinear optimization algorithm for X, then we are potentially discarding decades of progress in efficient "Krylov" iterative eigensolver algorithms. It might therefore be desirable to devise a *new* optimization algorithm specifically for (2) that is a *hybrid* of an existing Krylov eigensolver algorithm with a generic optimization algorithm. In these notes, we specifically propose a hybrid of the LOBPCG eigensolver algorithm with the CCSA nonlinear optimization algorithm.

One property of (3) to keep in mind is that $f_0(p, X) = f_0(p, XS)$ for any invertible $K \times K$ matrix S; this can easily be verified by explicit substitution in (3), but intuitively corresponds to the fact that such an S is simply a change of basis, and a trace is invariant under a change of basis. This will be useful to keep in mind in our final algorithm: we can arbitrarily change our X basis during optimization without changing anything important or introducing any troublesome discontinuity.

3 Review of CCSA optimization

First, let us briefly review a simplified form of the CCSA algorithm (Svanberg, 2002). We seek a *local* optimum of the problem

$$\begin{aligned} &\min_{p \in \mathbb{R}^P} f_0(p) \\ &f_i(p) \le 0, \; i = 1 \ldots M \end{aligned} \tag{5}$$

given a strictly feasible starting point p^0. In particular, CCSA yields a sequence p^k of *feasible* "outer" iterates with monotonically improving objective values $f_0(p^{k+1}) \le f_0(p^k)$ that converges to a local optimum p_*, assuming the f_i ($i = 0, \ldots, M$) are twice differentiable with bounded second derivatives (bounded Hessian norm). (The CCSA algorithm also has improved variants for simple bound constraints $a_j \le p_j \le b_j$ and a few other special cases, which are omitted here for simplicity.)

Each outer iterate is computed by solving an *approximate* **convex** optimization problem

$$\begin{gathered} \min_{p \in \mathbb{R}^P} g_0(p) \\ g_i(p) \le 0, \; i = 1 \dots M \\ p_j^k - \sigma_j \le p_j \le p_j^k + \sigma_j, \; j = 1 \dots P \end{gathered} \tag{6}$$

where the $\sigma_j > 0$ values define a kind of "trust region" and the g_i are *local* approximations for f_i near p^k defined below. It turns out that this "inner" convex problem (6) can be solved extremely efficiently because of the special form of g_i, yielding a "candidate" iterate p'. This candidate is *accepted* if it satisfies the condition $g_i(p') \ge f_i(p')$ for $i = 0 \dots M$, i.e. if the g_i functions are "conservative" approximations for f_i at p', in which case we set $p^{k+1} = p'$ and perform a new outer iteration. If $g_i(p') < f_i(p')$ for any i, however, we increase a quadratic "penalty" term in g_i to make it more conservative and re-solve the g optimization problem—this is an "inner" iteration. It follows that the outer iterates p^k are all feasible with monotonically improving objective f_0.

A key component of the CCSA algorithm is the choice of approximations g_i. It turns out that there are several choices with similar convergence properties, but all of them are *convex* functions that are a sum of *separable* functions of the design variables p_j, with $g_i(p^k) = f_i(p^k)$ and $\nabla_p g|_{p^k} = \nabla_p f|_{p^k}$. The simplest choice (which seems to perform about as well as the other choices) is the quadratic function:

$$g_i(p) = f_i(p^k) + \nabla_p f|_{p^k} \cdot (p - p^k) + \frac{\rho_i}{2} \sum_{j=1}^{P} \frac{(p_j - p_j^k)^2}{\sigma_j^2}, \tag{7}$$

where $\rho_i > 0$ is a penalty strength: larger ρ_i values make g_i more "conservative". On each **inner** iteration, when $g_i(p') < f_i(p')$ for a candidate solution p' of (6), we **increase** ρ_i geometrically (e.g. by a factor of 10). This guarantees that (6) will yield an acceptable "conservative" candidate p' in a small number of inner iterations (proportional to the log of the second derivative of f_i).

On each outer iteration p^{k+1}, CCSA also includes an update rule for the trust-region diameters σ_j and a decrease of the penalty factors ρ_i (so that subsequent optimization steps can be larger if the second derivative decreases). There is a straightforward proof that CCSA converges to a local optimum of (5), assuming a feasible starting point p^0 and bounded second derivatives of f_i.

CCSA has several nice properties that have made it a popular choice for PDE-constrained topology optimization, where the f_i can be extremely expensive (PDE solves) and the number P of design variables is usually $> 10^3$ and is often $> 10^6$. First, it has linear $\Theta(MP)$ scaling in time and memory, not including the application-dependent computation of $f_i(p)$. (In fact, if the derivatives $\nabla_p f_i$ are sparse, then CCSA can be implemented with cost proportional only to $\Theta(P)$ plus the number of nonzero entries.) Second, its convergence appears to be relatively robust and tolerant to small errors in $\nabla_p f_i$ (e.g. due to discretization or roundoff effects), although I'm not sure whether this has been rigorously

proven. Third, because it produces strictly feasible iterates p^k, it is possible to halt the optimization process early while still satisfying the constraints—in practical engineering-design problems, a solution within a few percent of optimum is typically satisfactory, but feasibility $f_i \leq 0$ may be absolutely required to satisfy physical laws or manufacturing limitations.

For our purposes, the relative simplicity of CCSA means that it may be easier to combine with Krylov eigensolvers, while retaining local-optimum convergence guarantees, than more sophisticated (and delicate) nonlinear optimization algorithms (e.g. BFGS quasi-Newton methods). Conversely, the fact that CCSA is essentially a first-order optimization algorithm makes a Krylov enhancement especially useful for accelerating eigensolver convergence for (2).

4 Review of LOBPCG

LOBPCG is a Krylov algorithm that finds the K algebraically smallest (or largest) eigenvalues of a real-symmetric (or Hermitian) matrix A, and can be viewed as a form of nonlinear conjugate-gradient algorithm to minimize the block Rayleigh quotient $f_0 = \operatorname{tr}\left[X^T A X (X^T X)^{-1}\right]$. (We omit the extension to the generalized eigenproblem.) The basic idea is that, on a given iteration X^k, the Rayleigh quotient can be minimized over a subspace spanned by X^k and the gradient $\nabla_X f_0$ using a Rayleigh–Ritz approach, which solves a small $2K \times 2K$ eigenproblem and takes the K smallest eigenvalues. Furthermore, by also including the *previous* iterate X^{k-1} in the search subspace, solving a $3K \times 3K$ eigenproblem, one obtains not only a strictly better iterate X^{k+1} but one also asymptotically (near the minimum, where f_0 becomes quadratic) obtains the "conjugacy" condition of conjugate-gradient algorithms and hence Krylov-like acceleration of convergence.

That is, on each iteration X^k one computes the gradient $R^k = \nabla_X f_0|_{X^k}$ from (4), and then (with some tricks outlined below) one essentially forms the matrix $Y^k = \left(X^k X^{k-1} R^k\right)$ and the $3K \times 3K$ Rayleigh–Ritz eigenproblem $(Y^k)^T A Y^k z = \mu (Y^k)^T Y^k z$. (This only requires the matrix–vector product AX^k.) Solving this (Hermitian) eigenproblem explicitly, one takes the smallest K Ritz values $\mu_1 \ldots \mu_K$ and the next iterate is given by the corresponding Ritz vectors: $X^{k+1} = (Y^k z_1, \cdots, Y^k z_K)$.

Note that, since the X^k are Ritz vectors from a previous iteration, they satisfy an orthogonality relationship $(X^k)^T X^k = I$, and in consequence the gradient $\nabla_X f_0|_{X^k}$ from (4) turns out to precisely consist of the residual vectors $AY^k z_\ell - \mu_\ell Y^k z_\ell$.

Although this is the essential idea, the full LOBPCG algorithm has a number of refinements. The "P" in LOBPCG stands for "preconditioned," and in general one may replace R^k with $W^k = TR^k$ where T is some symmetric-definite preconditioning matrix (an approximate inverse of A, shifted to be definite if necessary), with the simple case above corresponding to $T = I$. A key difficulty with the simple Rayleigh–Ritz iteration outlined above is that the basis Y^k may become poorly conditioned if X^k is close to X^{k+1}, so instead

one uses $Y^k = \left(X^k P^k W^k\right)$ where P^k is related to the difference between X^k and X^{k+1}; more precisely, the columns of P^{k+1} are given by $Y^k Z z_j$ where Z is a projection onto the last $2K$ components. One can also perform explicit re-orthogonalizations of the basis vectors as needed, "deflation" to project out already-computed eigenvectors, and other variations.

An important property of LOBPCG is that, by construction, it is no worse than a steepest-descent algorithm (and is expected to be substantially better as the solution is approached and conjugate gradient "kicks in"). In particularly, it is commonly applied in the context of density functional theory, where one minimizes a "nonlinear" version of the Rayleigh quotient in which the matrix $A(X)$ is a function of the current iterate X^k. Far from the converged minimum X_*, the matrix is changing rapidly on each iteration, and LOBPCG effectively degenerates to a steepest-descent algorithm, but as it converges and $A(X^k)$ approaches $A(X_*)$, the Krylov/conjugate-gradient properties begin to take effect and convergence accelerates. In our trace-estimation context (2), we hope to benefit from a similar property: as we approach the optimum p_*, our matrix $A(p)$ will approach $A(p_*)$ and the Krylov benefits of LOPBCG will appear, while far from the optimum it should be no worse than steepest descent. (More generally, the Krylov benefits should appear whenever p convergence stagnates.)

5 A possible CCSA–LOBPCG hybrid

My basic proposal is to combine CCSA iterations for p with LOBPCG iterations for X, by **interleaving** the two iterations.

1. Before each outer iteration of CCSA, we first perform an "ordinary" LOBPCG iteration with $A(p^k)$ to obtain X^{k+1} from the previous iteration's X^k and P^k (initialized to $P^0 = 0$). This is guaranteed to decrease f_0 (while leaving the f_i constraints unchanged).

2. Then, we form the CCSA approximations g_i exactly as in (7), and use them to perform inner iterations by solving (6) just as for ordinary CCSA, with **one crucial difference**: g_0 is formed from $f_0(p^k, X^{k+1})$ and the corresponding $\nabla_p f_0$, rather than using X^k, and the "conservatism" check $g_0(p') \geq f_0(p')$ is also performed using X^{k+1}.

The implementation of this is very straightforward, but is it guaranteed to converge? To answer that, one should go through the convergence proof from the original CCSA paper and see if it can be adapted to this hybrid algorithm for (2). (It looks like the convergence proof can be straightforwardly modified to hold for the new algorithm, but I haven't gone through it in detail.)