Dimension-reduced Interior Point Method

Discussion 4. Part 1

August 25, 2022

Solving LPs

Simplex constrained QP formulation

$$\begin{aligned} \min_{\mathbf{y},\mathbf{u}=(\mathbf{x},\mathbf{s},\kappa,\tau)} \quad & f(\mathbf{u}) := \frac{1}{2}\|\hat{\mathbf{A}}(\mathbf{y};\mathbf{u})\|^2 \\ \text{subject to} \quad & \mathbf{e}^\top \mathbf{u} = 1 \\ & \mathbf{u} \geq \mathbf{0}. \end{aligned}$$

Using the potential function

$$\varphi(\mathbf{u}) := \rho \log\left(f(\mathbf{u})\right) - B(\mathbf{x}) - B(\mathbf{s}) - \log\kappa - \log\tau$$

- $\mathbf{y}$ is treated unconstrained

$$\begin{aligned} \min_{\mathbf{d},\alpha^g,\alpha^m} \quad & \frac{1}{2}\mathbf{d}^\top \mathbf{H}\mathbf{d} + \mathbf{h}^\top \mathbf{d} \\ \text{subject to} \quad & \|\mathbf{U}^{-1}\mathbf{u}\|^2 \leq \beta \leq 1 \\ & \mathbf{d} = \alpha^g \mathbf{g}^k + \alpha^m \mathbf{m}^k. \end{aligned}$$

Preliminary results – Gradient + Momentum

The new method

- is able to solving NETLIB LPs to $10^{-3} \sim 10^{-4}$ relative accuracy (20000 iterations)
- slows down in reducing potential half way

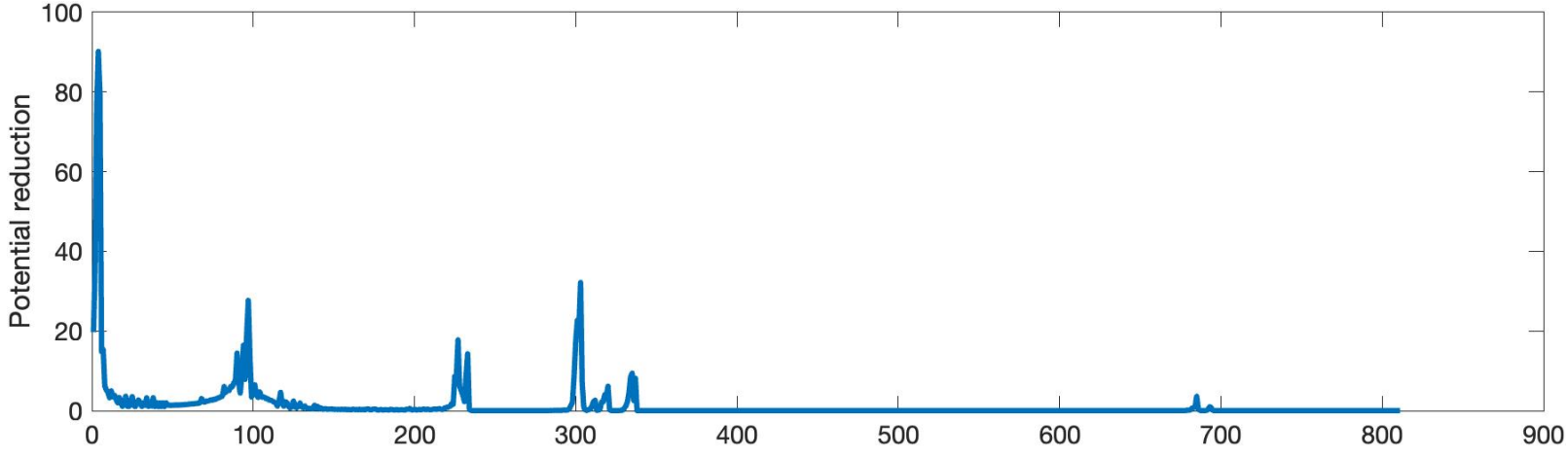


Figure 1. Reduction in potential

Main observation: steps may be trapped at local solutions.

So we use one more direction to escape it.

Adding Negative Curvatire

Note that

- $\mathbf{v}_{\min}$ that corresponds to $\lambda_{\min}(\nabla_{\mathbf{xx}}\varphi)$ is an ideal direction to escape a local solution.

We only need to consider $\begin{array}{ll}\min_{\mathbf{v}\neq\mathbf{0}} & \frac{\langle \mathbf{v}, \nabla_{\mathbf{xx}}\varphi\mathbf{v}\rangle}{\|\mathbf{v}\|^2} \\ \text{subject to} & \mathbf{e}^\top\mathbf{v}=0,\end{array}$ or simply

$$\min_{\mathbf{v}\neq\mathbf{0}} \frac{\langle \mathbf{v}, \mathbf{P}_\Delta\nabla_{\mathbf{xx}}\varphi\mathbf{P}_\Delta\mathbf{v}\rangle}{\|\mathbf{v}\|^2}$$

- Finding the miminum (negative) eigenvalue of $\mathbf{P}_\Delta\nabla_{\mathbf{xx}}\varphi\mathbf{P}_\Delta$
- Efficiently solvable using Lanczos exploiting

$$\nabla_{\mathbf{xx}}\varphi(\mathbf{x}) = \rho\left(-\frac{\nabla f(\mathbf{x})\nabla f(\mathbf{x})^\top}{f(\mathbf{x})^2} + \frac{\mathbf{A}^\top\mathbf{A}}{f(\mathbf{x})}\right) + \mathbf{X}^{-2}$$

and structure of $\mathbf{A}$.

In practice we can add curvature

- if $\varphi(\mathbf{x}^k) - \varphi(\mathbf{x}^{k+1}) \leq \varepsilon$
- every K iterations

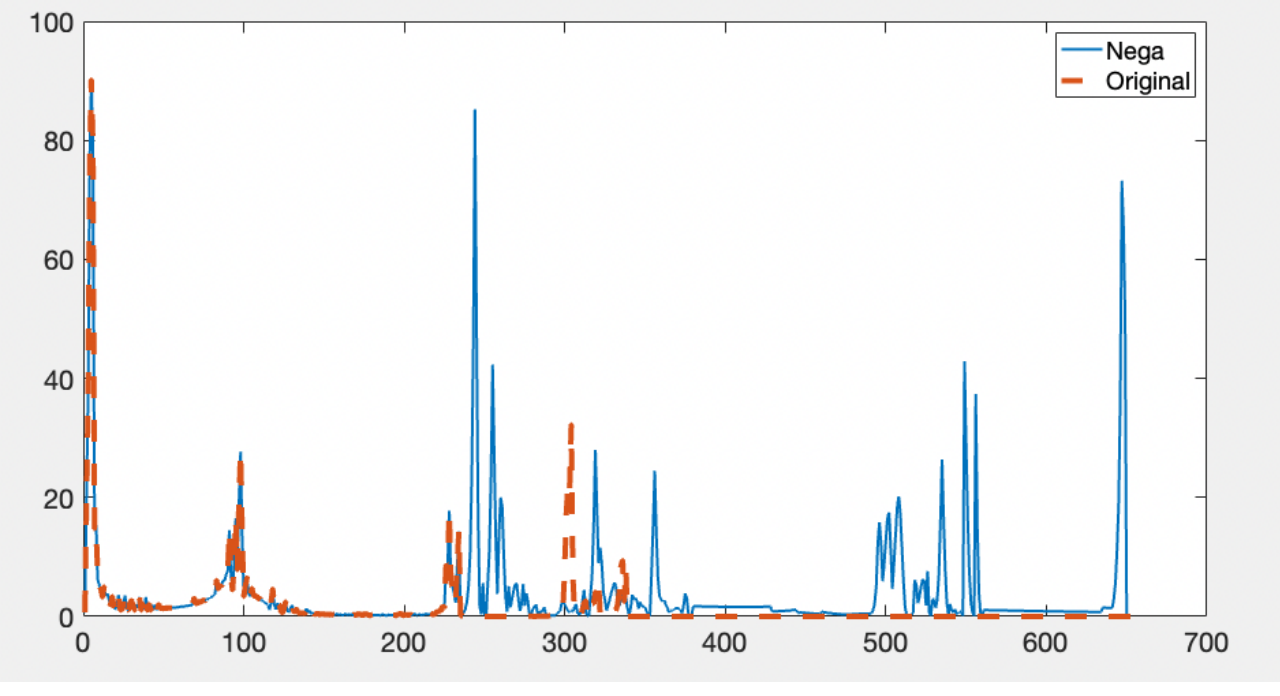


Figure 2. Adding negative curvature

Efficiency of Negative Curvature

Original: solving NETLIB LPs to $10^{-3} \sim 10^{-4}$ relative accuracy in 20000 iterations

Now: solving NETLIB LPs to $10^{-5} \sim 10^{-8}$ relative accuracy in 1000 iterations

Problem	PInfeas	DInfeas.	Compl.	Problem	PInfeas	DInfeas.	Compl.
DLITTLE	1.347e-10	2.308e-10	2.960e-09	KB2	5.455e-11	6.417e-10	7.562e-11
AFIRO	7.641e-11	7.375e-11	3.130e-10	LOTFI	2.164e-09	4.155e-09	8.663e-08
AGG2	3.374e-08	4.859e-08	6.286e-07	MODSZK1	1.527e-06	5.415e-05	2.597e-04
AGG3	2.248e-05	1.151e-06	1.518e-05	RECIPELP	5.868e-08	6.300e-08	1.285e-07
BANDM	2.444e-09	4.886e-09	3.769e-08	SC105	7.315e-11	5.970e-11	2.435e-10
BEACONFD	5.765e-12	9.853e-12	1.022e-10	SC205	6.392e-11	5.710e-11	2.650e-10
BLEND	2.018e-10	3.729e-10	1.179e-09	SC50A	1.078e-05	6.098e-06	4.279e-05
BOEING2	1.144e-07	1.110e-08	2.307e-07	SC50B	4.647e-11	3.269e-11	1.747e-10
BORE3D	2.389e-08	5.013e-08	1.165e-07	SCAGR25	1.048e-07	5.298e-08	1.289e-06
BRANDY	2.702e-05	7.818e-06	1.849e-05	SCAGR7	1.087e-07	1.173e-08	2.601e-07
CAPRI	7.575e-05	4.488e-05	4.880e-05	SCFXM1	4.323e-06	5.244e-06	8.681e-06
E226	2.656e-06	4.742e-06	2.512e-05	SCORPION	1.674e-09	1.892e-09	1.737e-08
FINNIS	8.577e-07	8.367e-07	1.001e-05	SCTAP1	5.567e-07	8.430e-07	5.081e-06
FORPLAN	5.874e-07	2.084e-07	4.979e-06	SEBA	2.919e-11	5.729e-11	1.448e-10
GFRD-PNC	4.558e-05	1.052e-05	4.363e-05	SHARE1B	3.367e-07	1.339e-06	3.578e-06
GROW7	1.276e-04	4.906e-06	1.024e-04	SHARE2B	2.142e-04	2.014e-05	6.146e-05
ISRAEL	1.422e-06	1.336e-06	1.404e-05	STAIR	5.549e-04	8.566e-06	2.861e-05
STANDATA	5.645e-08	2.735e-07	5.130e-06	STANDGUB	2.934e-08	1.467e-07	2.753e-06
STOCFOR1	6.633e-09	9.701e-09	4.811e-08	VTP-BASE	1.349e-10	5.098e-11	2.342e-10

Table 1. Solving NETLIB LPs in **1000** iterations

One observation:

Pre-conditioning by $\mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}\mathbf{A} \in \mathbb{R}^{n\times n}$ sometimes unfriendly to HSD since $\tau \to 0$.

Following aspects may accelerate the algorithm

- Adjust ρ adaptively
- More careful matrix scaling to enhance conditioning
- Faster eigen-routine (may be randomized)

  Solving the eigen-problem below

$$\lambda_{\min}\left\{\nabla_{\mathbf{xx}}\varphi(\mathbf{x}) = \rho\left(-\frac{\nabla f(\mathbf{x})\nabla f(\mathbf{x})^\top}{f(\mathbf{x})^2} + \frac{\mathbf{A}^\top\mathbf{A}}{f(\mathbf{x})}\right) + \mathbf{X}^{-2}\right\}$$

  with randomized technique for negative curvature
- More directions

Efficient Eigen-decomposition for Sparse Matrices

Discussion 4. Part 2

August 25, 2022

Efficient Eigen-decomposition

The eigen routine in `HDSDP` is now re-written and presented as a stand-alone library `SPEIGS`

It targets the problem

$$\mathbf{AV} = \Lambda \mathbf{V}$$

for **extremely** sparse $\mathbf{A} \in \mathbb{S}^{n \times n}$ and needs the full spectrum.

Recall that in dual-scaling

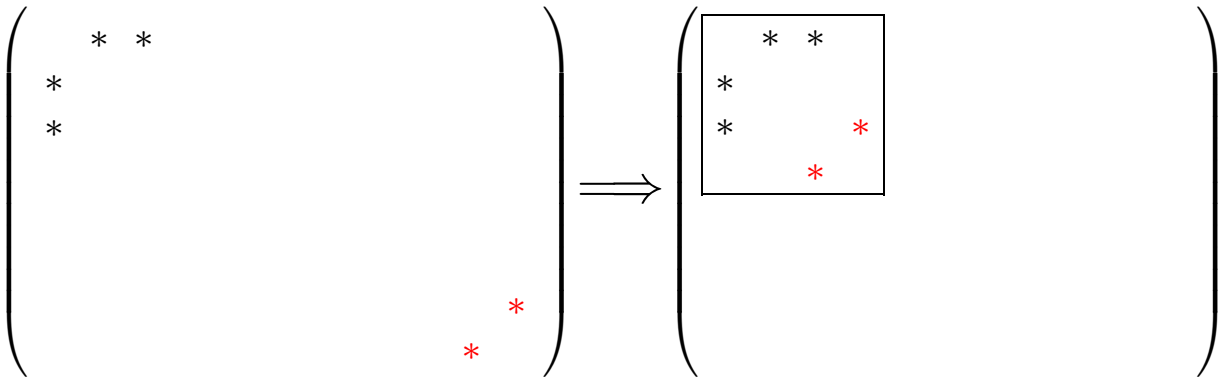
$$\langle \mathbf{C}, \mathbf{S}^{-1}\mathbf{A}\mathbf{S}^{-1} \rangle = \sum_{i=1}^{r(\mathbf{A})} \lambda_i \langle \mathbf{a}_i, \mathbf{C}\mathbf{a}_i \rangle.$$

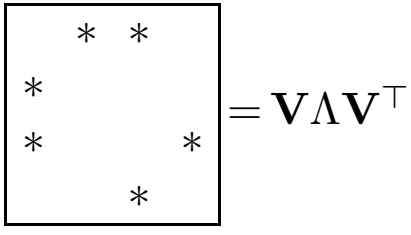
- Efficient eigen-routine is critical as a pre-processing step
- Generally hard to exploit sparsity in full-eigen decomposition problem
- Benson implements a method that targets decomposition of SDPs

Efficient Eigen-decomposition

Many SDP coefficient matrices are so sparse that very few entries exist

Benson's idea: permute the useful entries to a smaller dense block





$= \mathbf{V}\Lambda\mathbf{V}^\top$

Then Lapack is invoked on the smaller system.

1000x faster than running direct factorization. Simple but useful.

SPEIGS

SPEIGS now implments a standard-alone library for factorizing extremely sparse matrices.

It implements

- Submatrix detection and permutation
- Diagonal detection
- Givens' rotation
- Rank-one fast detection

for both sparse and dense matrices and works efficiently for SDP matrices

(and probably real-life matrices that are sparse)

Available in Matlab and C.