Acceleration Methods for Scientific and Data Science Applications

Mitchell Scott[1], Ziyuan Tang[2], Tianshi Xu[1], Yousef Saad[2], Yuanzhe Xi[1]

[1]Department of Mathematics, Emory University
[2]Department of Computer Science, University of Minnesota

Introduction

- **Fixed point iterations** are a common approach to solve (complex) physical simulations leading to a sequence of a physical quantities like charge densities, potentials, pressures, etc..
$$x_{j+1} = g(x_j)$$
- **Acceleration methods** combine $g(x_j)$ with past iterates, x_{j-m}'s, leading to faster convergence. These methods try to solve $f(x) = 0$, where
$$f(x) = x - \beta g(x).$$
- **Anderson Acceleration**[1] is a famous example. Given $\mathbf{x}_i, f_i \equiv f(\mathbf{x}_i)$, for $i = j-m, \cdots, j$, we construct
$$\Delta x_i = \mathbf{x}_{i+1} - \mathbf{x}_i, \quad \Delta f_i = f_{i+1} - f_i, \quad \forall i.$$
Constructing
$$\mathbf{P}_j = [\Delta x_{j-m} \cdots \Delta x_{j-1}], \quad \mathbf{V}_j = [\Delta f_{j-m} \cdots \Delta f_{j-1}].$$
Computing $\bar{\mathbf{x}}_j = \mathbf{x}_j - \mathbf{P}_j \boldsymbol{\theta}^{(j)}, \bar{f}_j = f_j - \mathbf{V}_j \boldsymbol{\theta}^{(j)}, \mathbf{x}_{j+1} = \bar{\mathbf{x}}_j + \beta \bar{f}_j$, where
$$\boldsymbol{\theta}^{(j)} = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^m} \|f_j - \mathbf{V}_j \boldsymbol{\theta}\|$$
- **Krylov subspace** methods construct a subspace of the problem space in which to find a solution.
$$\mathcal{K}_\ell = \operatorname{span}\{\mathbf{v}, \mathbf{J}\mathbf{v}, \cdots, \mathbf{J}^{\ell-1}\mathbf{v}\}, \quad \text{where } \mathbf{v} \equiv -f(x_j)$$
- **Aim** to extend linear accelerators into the nonlinear case for both scientific/ data science applications with **Nonlinear Truncated Generalized Conjugate Residual (nlTGCR)**

nlTGCR

- **Generalized Conjugate Residual (GCR)**[2] solves $\mathbf{A}\mathbf{x} = \mathbf{b}$, by building a sequence of search directions $\mathbf{p}_i, i = 1:j$ so $\{\mathbf{A}\mathbf{p}_i\}_{i=1:j}$ is orthogonal.
- GCR(k), a variant which restarts every k steps, is equivalent to GMRES(k).
- Going to the nonlinear case[3], we replace the orthogonal $\{\mathbf{A}\mathbf{p}_j\}$ with Jacobian for PDE problems and Fisher information matrix for neural network problems.
$$\mathbf{P}_j = [p_{j_m}, p_{j_m+1}, \cdots, p_j], \quad \mathbf{V}_j = [\mathbf{J}(x_{j_m})v_{j_m}, \cdots, \mathbf{J}(x_j)v_j]$$

ALGORITHM: nlTGCR(m,k)

Input: $f(x)$, initial x_0
Set $r_0 = -f(x_0)$
Compute $v = \mathbf{J}(x_0) r_0$ ▷ Use Frechet
$v_0 = v/\|f\|, p_0 = r_0/\|v\|$
for $j = 0, 1, 2, \ldots$ **do**
$y_j = \mathbf{V}_j^\top r_j$
$x_{j+1} = x_j + \mathbf{P}_j y_j$ ▷ Scalar α_j becomes vector y_j
$r_{j+1} = -f(x_{j+1})$ ▷ Replaces linear update: $r_{j+1} = r_j - \mathbf{V}_j y_j$
Set: $p := r_{j+1}$;
Compute $v = \mathbf{J}p$ ▷ Use Frechet
for $i = j_m$ to j **do**
$\beta_{ij} = \langle v, v_i \rangle$
$p = p - \beta_{ij} p_i, v = v - \beta_{ij} v_i$
end for
$p_{j+1} = p/\|v\|, v_{j+1} = v/\|v\|$
If mod $j, k == 0$, restart
end for

Fisher Information Matrix and Approximations

Generalized Gauss Newton:

- Goal to train NN $f(\mathbf{x}, \boldsymbol{\theta})$ with data $\mathbf{x}$ and parameter $\boldsymbol{\theta}$.
- The objective function is
$$h(\boldsymbol{\theta}) = \mathbb{E}_Q[L(y, f(\mathbf{x}, \boldsymbol{\theta}))],$$
where Q is the dataset distribution.
- Then, Generalized Gauss-Newton is
$$\mathbf{G} \approx \frac{1}{m} \sum_{i=1}^{m} \mathbf{J}_i^\top \mathbf{J}_i,$$
where $\mathbf{J}_i$ is the Jacobian of $f(x_i, \boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$.

Fisher Information Matrix:

- Using $\ell(\mathbf{y}, f(\mathbf{x}, \boldsymbol{\theta})) = -\log p(y|f(x, \theta))$, and conditional density like Gaussian, Poisson, and Bernoulli, $\mathbf{G} = \mathbf{F}$.
$$\mathbf{F} = \mathbb{E}\left[\frac{d \log p(y|x,\theta)}{d\theta} \frac{d \log p(y|x,\theta)}{d\theta}^\top\right] = \mathbb{E}[\mathcal{D}\boldsymbol{\theta}\mathcal{D}\boldsymbol{\theta}^\top]$$

Approximating Fisher Information:

- Only consider DNN with multiple linear layers.
- In each layer, if we assume $\mathbf{A}, \mathbf{G}$ are statistically independent:
$$\mathbf{F} = \mathbb{E}\left[\operatorname{vec}(\mathbf{g}\mathbf{a}^\top) \operatorname{vec}(\mathbf{g}\mathbf{a}^\top)^\top\right] \approx \mathbf{A} \otimes \mathbf{G},$$
where $\mathbf{A}$ is the gradient of the input $m \times m$, $\mathbf{G}$ is the gradient of the output $n \times n$, $\mathbf{F}$ is of size $mn \times mn$, where the weight matrix $\mathbf{W}$ is $m \times n$.
- Using Kronecker products[4], the inverse of $\mathbf{F}$ is:
$$\mathbf{F}^{-1} \approx \mathbf{A}^{-1} \otimes \mathbf{G}^{-1}.$$
- Then this can be done efficiently as
$$(\mathbf{A}^{-1} \otimes \mathbf{G}^{-1})\operatorname{vec}(\mathbf{X}) = \operatorname{vec}(\mathbf{G}^{-1}\mathbf{X}\mathbf{A}^{-\top}).$$
- This approximation to the Fisher information matrix can be used as a preconditioner for training the neural network[5].

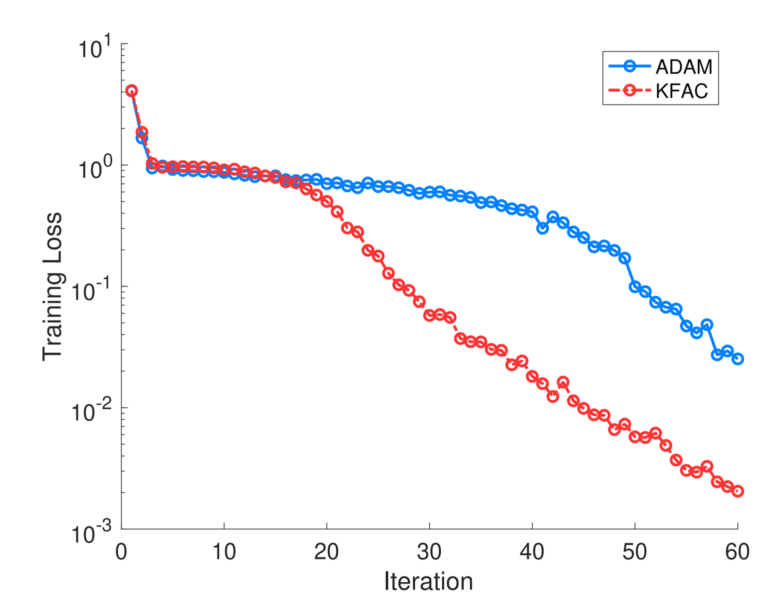
Neural Network Problems

2D Poisson Problem: *Set-up*:
$$\begin{cases} -\Delta u = 1 & u \in \mathcal{B}_1(0) \\ u = 0 & u \in \partial\mathcal{B}_1(0) \end{cases}$$

Network Parameters:

- 5 hidden layers
- 30 neurons on each layer
- 2,000 iterations

Method	Accuracy
Fisher Approach:	1.79125e-05
ADAM Approach:	8.27138e-04



References

[1] Donald G. Anderson. "Iterative Procedures for Nonlinear Integral Equations". In: *J. ACM* 12.4 (Oct. 1, 1965), pp. 547–560. ISSN: 0004-5411. DOI: 10.1145/321296.321305. URL: https://dl.acm.org/doi/10.1145/321296.321305 (visited on 10/24/2024).

[2] Stanley C. Eisenstat, Howard C. Elman, and Martin H. Schultz. "Variational Iterative Methods for Nonsymmetric Systems of Linear Equations". In: *SIAM Journal on Numerical Analysis* 20.2 (1983). Publisher: Society for Industrial and Applied Mathematics, pp. 345–357. ISSN: 0036-1429. URL: https://www.jstor.org/stable/2157222 (visited on 10/24/2024).

[3] Huan He et al. *NLTGCR: A class of Nonlinear Acceleration Procedures based on Conjugate Residuals*. Mar. 30, 2024. DOI: 10.48550/arXiv.2306.00325. arXiv: 2306.00325. URL: http://arxiv.org/abs/2306.00325 (visited on 10/24/2024).

[4] Charles F. Van Loan. "The ubiquitous Kronecker product". In: *Journal of Computational and Applied Mathematics*. Numerical Analysis 2000. Vol. III: Linear Algebra 123.1 (Nov. 1, 2000), pp. 85–100. ISSN: 0377-0427. DOI: 10.1016/S0377-0427(00)00393-9. URL: https://www.sciencedirect.com/science/article/pii/S0377042700003939 (visited on 10/24/2024).

[5] Mitchell Scott et al. *Fisher Information Matrix based Preconditioner for Deep Neural Networks*. 2024.

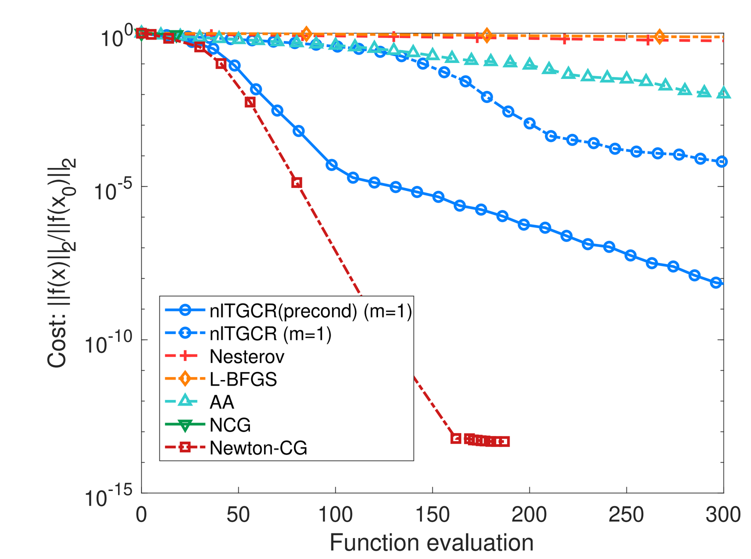
Nonlinear Eigenvalue Problems

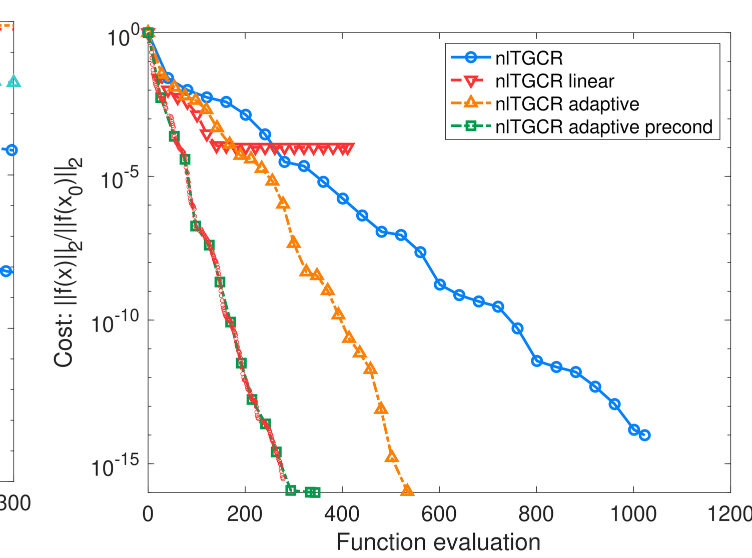
Bratu Problem:
Set-up:
$$\begin{cases} -\Delta u - \lambda e^u = 0 & \text{in } \Omega = (0,1)^2 \\ u(x,y) = 0 & \text{for } (x,y) \in \partial\Omega \end{cases}$$

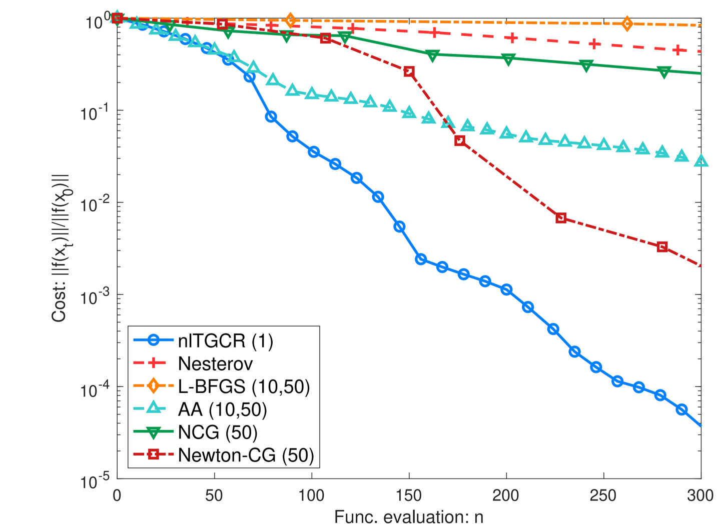
- $\lambda = 0.5$
- Using centered FD on a grid of $100 \times 100 \to n = 10,000$
- Hessian is $\mathbf{A} - \lambda \operatorname{diag}(e^u)$, where $\mathbf{A} = [-1, 2, -1]$ tridiagonal.

Adaptive nlTGCR

- Bratu problem is almost linear, especially near convergence.
- Exploit linearized form by
$$r_{j+1}^{nl} = -f(x_{j+1}), \quad r_{j+1}^{lin} = r_{j+1}^{nl} - \mathbf{V}_j \mathbf{y}_j$$
- Turn on linear updates when $d_j <$ threshold τ,
$$d_j = 1 - \frac{(r_j^{nl})^\top r_j^{lin}}{\|r_j^{nl}\| \|r_j^{lin}\|}$$
- Precondition the linearized problem







Take-aways:

- nlTGCR beats most other methods for the Bratu problem.
- By exploiting symmetry of Hessian, nlTGCR is the clear victor.
- Adapting, and preconditioning when possible, speeds up convergence while removing function evaluations.

Conclusions

- Extends linear Krylov accelerator TGCR to the nonlinear setting
- Exploits the short-term recurrence for symmetric problems
- Implements global convergence strategies
- Adaptable to stochastic gradient-type methods
- Extendable to develop short-term AA algorithms

Future Directions:

- Test Fisher method on larger DNN problems.
- Prove convergence bounds on stochastic problems.
- Compare to more preconditioned SGD algorithms.

Acknowledgements

This material is based upon work supported by the NSF under Award Number DMS-2208412.