arXiv preprint. Under review.

Randomized Iterative Solver as Iterative Refinement: A Simple Fix Towards Backward Stability

Ruihan Xu[1] **and Yiping Lu**[2]

[1] *1Department of Mathematics, University of Chicago*
[2] *2 Industrial Engineering & Mangenement Science, Northwestern University*

This manuscript was compile on Oct 1, 2024

Abstract

Iterative sketching and sketch-and-precondition are well-established randomized algorithms for solving large-scale over-determined linear least-squares problems. In this paper, we introduce a new perspective that interpreting Iterative Sketching and Sketching-and-Precondition as forms of Iterative Refinement. We also examine the numerical stability of two distinct refinement strategies: iterative refinement and recursive refinement, which progressively improve the accuracy of a sketched linear solver. Building on this insight, we propose a novel algorithm, Sketched Iterative and Recursive Refinement (SIRR), which combines both refinement methods. SIRR demonstrates a *four order of magnitude improvement* in backward error compared to iterative sketching, achieved simply by reorganizing the computational order, ensuring that the computed solution exactly solves a modified least-squares system where the coefficient matrix deviates only slightly from the original matrix. To the best of our knowledge, *SIRR is the first asymptotically fast, single-stage randomized least-squares solver that achieves both forward and backward stability*.

Keywords: *Numerical Stability, Sketching, Numerical Linear Algebra, Iterative Refinement*

Corresponding author: Yiping Lu *E-mail address:* yiping.lu@northwestern.edu

1. Introduction

Randomized numerical linear algebra (RNLA) [9], [11]–[13], [17], [18], [29] is a rapidly evolving branch of matrix computations, driving significant progress in low-rank approximations, iterative methods, and projections. This field has demonstrated that randomized algorithms are highly effective tools for developing approximate matrix factorizations. These methods are remarkable for their simplicity and efficiency, often producing surprisingly accurate results.

In this paper, we consider randomized algorithms to solve the overdetermined linear least-squares problem

$$x = \arg\min_{y \in \mathbb{R}^n} \|b - Ay\| \quad (A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m) \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm. This is one of the core problems in computational sceience [19], [20], [23], [27], statistics [25], [36] and accelerating machine learning [22], [28], [31]. In the past two decades, researchers in the field of *randomized numerical linear algebra* [13], [17], [29] have developed least-squares solvers that are faster than Householder QR factorization [15], the textbook algorithm for least square, which runs in $O(mn^2)$ operations. Randomized algorithms first sketch A to a smaller matrix SA with a random sketch matrix $S \in \mathbb{R}^{[cn] \times m}$ for some constant $c > 1$. The random embedding $v \to Sv$ satisfies $\|Sv\| \approx \|v\|$ for all vectors $v \in \text{range}([Ab])$ and matrix–vector products $v \to Sv$ can be computed efficiently [13], [29].

There are two main approaches to using the sketched matrix SA for a fast randomized least squares solver: *the sketch-and-precondition* [9] method and *iterative Hessian sketching* [21], [24], [26]. Most of the solvers (e.g. Blendenpik [12]) have a complexity of $O(mn \log m)$ operations. This is significantly better than the $O(mn^2)$ complexity. Consequently, for large least-squares (LS) problems, randomized solvers can be substantially faster than the LS solver implemented in LAPACK [12]. However, recent research [30], [37] surprisingly finds that sketch-and-precondition [9], [12] and iteratively Hessian sketch [21], [24], [26] are numerically unstable in their standard form, both stagnate in terms of residual and backward error, potentially before optimal levels are reached. [37] further propose sketch-and-apply, which is a provable method that attains backward stable solutions under modest conditions. Unfortunately, sketch-and-apply requires $O(mn^2)$ operations, the same as Householder QR-based direct solvers. In this paper, we provide a definitive answer to the open question posed by [30], [37]:

> *Is there a randomized least-squares algorithm that is both (asymptotically) faster than Householder QR and numerically stable?*

We constructed a solver called **S**ketched **I**terative and **R**ecursive **D**ebiasing, which enjoys both forward and backward stability while requires only $O(mn + n^3)$ computation. Our approach is based on a novel, unified perspective on sketch-and-precondition methods and iterative Hessian sketching. Although these two techniques may seem different, we demonstrate that they can be interpreted as iterative refinement processes. Iterative refinement (IR) is a well-known method for solving linear systems by progressively improving the accuracy of an initial approximation. We show that employing iterative refinement, a sketch-and-solve solver is equivalent to using Jacobi iteration in a sketch-and-precondition framework. We investigated the conditions that a single-step approximate solver needs to satisfy in order for iterative refinement to potentially achieve backward stability. To construct the single-step approximate solver, we studied another way for iterative refinement called **S**ketched **R**ecursive **R**efinement. Note that we find, both theoretically and numerically, that only in certain cases where data noise is relatively large, SRR alone can achieve a backward stable solution. Only using SRR as the meta-algorithm of iterative refinement, *i.e.* **S**ketched **I**terative and **R**ecursive **D**ebiasing, can provide a backward stable algorithm.

We would like to highlight a concurrent work [35], which also developed a backward stable solver with a computational complexity of $O(mn + n^3)$. However, the FOSSILS solver proposed in their work follows a two-stage approach, where each stage involves an iterative process. In contrast, our algorithm is a single-stage solver that offers the flexibility to stop at any point during the computation, making it more adaptable for scenarios where early termination is necessary or beneficial.

Notation Through out this paper, $A \in \mathbb{R}^{m \times n}$, $S \in \mathbb{R}^{s \times m}$, $b \in \mathbb{R}^m$. $\|\cdot\|$ denotes vector ℓ_2 norm for vectors and operator ℓ_2 norm for matrices. We use $|\cdot|$ denotes l_1 norm. $\kappa = \|A\|\|A^\dagger\|$ is the condition number of A and $\sigma_{\max}(\cdot)$,$\sigma_{\min}(\cdot)$ denotes the largest and smallest singular value. u denotes the machine epsilon which is used to measure the level of roundoff error in the floating-point number system. For IEEE standard double precision, u is around 2×10^{-16}. $a \lesssim b$ denotes $a \leq cb$ for some small constant c, which is independent of m, n, s, κ, u. $a \asymp b$ indicates that $a \lesssim b$ and $b \lesssim a$. γ_m is defined as $\gamma_m = \frac{mu}{1-mu}$. In numerical analysis, we assume that $u\kappa n^{\frac{3}{2}} < 1$ and $un^{\frac{3}{2}}\|x*\| \lesssim \|b - Ax^*\|$. We also assume that m has the same order with n for computational simplicity, which will be restated in the following sections. Note that $n^{\frac{3}{2}} < 1$ is a guarantee for a nonsingular $\hat{R}$ computed in QR factorization according to [7, Theorem 19.3]. Without loss of generality, $\|A\| = \|b\| = 1$ is assumed in analysis, except in forward stability analysis where we keep $\|A\|$ and $\|b\|$ unknown to align with Wedin's perturbation theorem. Computed quantities wear a hat, e.g. $\hat{x}$ denotes the computed approximation of x.

1.1. Contribution

We offer a unified understanding of existing randomized least squares solvers, such as iterative sketching and sketch-and-precondition, by interpreting them as forms of iterative refinement. This new perspective enables the development of novel techniques for analyzing the numerical stability of randomized algorithms by exploring and comparing the stability of iterative and recursive refinement strategies for progressively improving the accuracy of sketched linear solvers. Based on the analysis, we propose Sketched Iterative and Recursive Refinement (SIRR), which combines iterative and recursive refinement techniques and achieves the first single stage provably backward stable and computationally efficient, with asymptotic complexity $O(mn + n^3)$, faster than traditional direct solvers.

2. Preliminary

Sketch-and-Precondition There are lots of randomized methods that obtain a right preconditioner from SA for further iterative LS method, which is known as sketch-and-precondition [9], [12], [16]. The core insight of Sketch-and-Precondition is that sketching matrices can be used to precondition (i.e., reduce the condition number) the original matrix $A \in \mathbb{R}^{m\times n}$. To be specific, for a matrix $A \in \mathbb{R}^{m\times n}$ and sketching matrix $S \in \mathbb{R}^{s\times m}$ with distortion $0 < \eta < 1$ (*i.e.* $(1-\eta)\|Ay\| \leq \|SAy\| \leq (1+\eta)\|Ay\|$ holds for all $y \in \mathbb{R}^n$), the preconditioner R can be obtained from QR factorization of matrix $SA = QR$ with Q orthonormal and R square. The preconditioner R satisfies

$$\frac{1}{1+\eta} \leq \sigma_{\min}(AR^{-1}) \leq \sigma_{\max}(AR^{-1}) \leq \frac{1}{1-\eta}.$$

To be specific, one can always construct a random sparse embedding matrix S that satisfies the following Lemma.

Lemma 1 ([7], [30], [35], [37]). *For matrix $A \in \mathbb{R}^{m\times n}$, there exists sketching matrix $S \in \mathbb{R}^{s\times m}$. Suppose that $\hat{R}\hat{Q} = SA$ is the QR decomposition of matrix SA, then the following inequalities holds:*

- $\|\hat{R}\| \lesssim \|A\|$, $\|\hat{R}^{-1}\| \lesssim \frac{\kappa}{\|A\|}$
- $1 - u\kappa n^{\frac{5}{2}} \lesssim \sigma_{min}(A\hat{R}^{-1}) \lesssim \sigma_{max}(A\hat{R}^{-1}) \lesssim 1 + u\kappa n^{\frac{5}{2}}$

One of the most prominent sketch-and-precondition techniques is using R as the preconditoner for LSQR [5] which is known as Blendenpik [12]. In exact arithmetic, Blendenpik has a complexity of $\mathcal{O}(mn\log m)$ operations, which is better than the $\mathcal{O}(mn^2)$ QR-based direct solver. Consequently, for large LS problems, Blendenpik can be substantially faster than the LS solver implemented in LAPACK, a widely used software library for numerical linear algebra.

Iterative Hessian Sketching [21], [26] Iterative Sketching start from an initial solution $x_0 \in \mathbb{R}^n$ generate iterates $x_1, x_2, \cdots$ by solving a sequence of the sketched least-squares problems

$$x_{i+1} = x_i + \operatorname{argmin}_{y\in\mathbb{R}^n} \frac{1}{2}\|(SA)y\|^2 - y^\top A^\top(b - Ax_i), \tag{2}$$

for $i = 0, 1, 2, \cdots$. As with the classical least-squares sketch, the quadratic form is defined by the matrix $SA \in \mathbb{R}^{m\times d}$, which leads to computational savings. The closed form solution of (2) is given via $x_{i+1} = x_i + (A^\top S^\top SA)^\dagger A^\top(b - Ax_i)$ which encounter with the iterative refine a sketch-and-apply solver which shown in Algorithm 3. In Section 4.1.0.4, we also show that Iterative Hessian Sketching/Iterative Refinement is equivalent to Sketch-and-Precondition using a Jacobi Iteration Solver.

Backward-Stability Backward stability refers to the property of a numerical algorithm where the computed solution is the exact solution to a slightly perturbed version of the original problem. Specifically, a solver is said to be backward stable if the solver satisfies the following property:

Definition (Backward error). In floating point arithmetic, it produces a numerical solution $\hat{x}$ that is the exact solution to a slightly modified problem:

$$\hat{x} = \arg\min_{y\in\mathbb{R}^n} \|(b + \Delta b) - (A + \Delta A)y\| \tag{3}$$

where the (relative) size of the perturbations is at most

$$\|\Delta A\| \leq c\|A\|, \quad \|\Delta b\| \leq c\|b\| \quad \text{provided } c < 1. \tag{4}$$

[35] show that a backward stable solver can achieve accurate estimation of each component of the solution and can enforce residual orthogonality, *i.e.* the KKT condition of the least square problem that $A^\top(Ax - b) = 0$. The classic Householder QR least-squares method is backward stable [7, Ch. 20]. However, recent works [30], [37] showed that randomized sketching solver is not backward stable.

To prove a solver is backward stable, we follow [14], [35] which utilize the Karlson-Waldén estimate $\widehat{\mathrm{BE}}_\theta(\hat{\boldsymbol{x}}) := \frac{\theta}{\sqrt{1+\theta^2\|\hat{\boldsymbol{x}}\|^2}}\left\|\left(\boldsymbol{A}^\top\boldsymbol{A} + \frac{\theta^2\|\boldsymbol{b}-\boldsymbol{A}\hat{\boldsymbol{x}}\|^2}{1+\theta^2\|\hat{\boldsymbol{x}}\|^2}\mathbf{I}\right)^{-1/2}\boldsymbol{A}^\top(\boldsymbol{b} - \boldsymbol{A}\hat{\boldsymbol{x}})\right\|$ which can estimate the backward error up to a constant, *i.e.* $\widehat{\mathrm{BE}}_\theta(\hat{\boldsymbol{x}}) \leq \mathrm{BE}_\theta(\hat{\boldsymbol{x}}) \leq \sqrt{2}\widehat{\mathrm{BE}}_\theta(\hat{\boldsymbol{x}})$ [14]. Given singular value decomposition $A = \sum_{i=1}^n \sigma_i u_i v_i^\top$, the Karlson-Waldén estimation indicates that a least square $\hat{x}$ is backward stable is equivalent to satisfying a component-wise error bound $\left|\mathbf{v}_i^\top(\hat{\mathbf{x}} - \mathbf{x})\right| \lesssim \sigma_i^{-1}\cdot(1 + \|\hat{\mathbf{x}}\|)u + \sigma_i^{-2}\cdot\|\mathbf{b} - A\hat{\mathbf{x}}\|u \quad \text{for } i = 1, \ldots, n.$.

Definition ($\alpha - \beta$ Accuracy). We define $\hat{x}$ is $\alpha - \beta$ accurate if there exists $e_1, e_2 \in \mathbb{R}^n$ such that $\|e_1\|, \|e_2\| \leq 1$ and

$$\hat{x} - x^* = \alpha(1 + \|\hat{x}\|)\hat{R}^{-1}e_1 + \beta\|b - A\hat{x}\|(A^\top A)^{-1}e_2,$$

where $\hat{R}$ is a preconditioner of A such that for any singular value of $A\hat{R}^{-1}$ satisfies $\sigma(A\hat{R}^{-1}) \asymp 1$.

Lemma 2. *The computed solution $\hat{x}$ of problem $Ax = b$ has backward error $be(\hat{x}) \lesssim \sqrt{n}\epsilon$ if*

$$\hat{x} - x^* = \epsilon(1 + \|\hat{x}\|)\hat{R}^{-1}e_1 + \epsilon\|b - A\hat{x}\|(A^\top A)^{-1}e_2, \tag{5}$$

where $e_i \in \mathbb{R}^n$ satisfies $\|e_i\| \lesssim 1 (i = 1, 2)$.

Numerical Stability We provide several basic facts about numerical errors generated in floating-point arithmetic, most of which can be found in [7]. For error analysis, we denote the numerical error of an expression computed in floating-point arithmetic as $\mathrm{err}(\cdot)$. Specifically, for a real number x, let $\mathrm{fl}(x)$ denote its floating-point approximation. The numerical error in x is then defined as $\text{err}(x) = |x - \text{fl}(x)|$. Recall that u denotes the unit roundoff, which is the maximum relative error in representing a real number in floating-point arithmetic. That is, for any real number x, we have $|\text{fl}(x) - x| \leq u|x|$. We also define γ_n for a positive integer n as $\gamma_n = \frac{nu}{1-nu}$, assuming $nu \ll 1$, so that $\gamma_n \approx nu$.

Fact. For vector $x, y \in \mathbb{R}^n$, matrix $A \in \mathbb{R}^{m\times n}$, upper triangular matrix $R \in \mathbb{R}^{n\times n}$, we have

- $\|\text{err}(x \pm y)\| \leq u\sqrt{n}\|x \pm y\|$.
- $\|\text{err}(Ax)\| \leq \sqrt{n}\gamma_n\|A\|\|x\|$.
- higham2002accuracyFor problem $Rx = y$, the solution by Gaussian-elimination satisfies $(R + E)(R^{-1}y + error(R^{-1}y)) = y$ where $|E| \lesssim \gamma_n|R|$. This result further leads to $err(R^{-1}y) = \sqrt{n}\gamma_n\|R^{-1}y\|R^{-1}e$, where $\|e\| \lesssim 1$.
- For problem $Ax = b$, the solution by QR factorization satisfies $(A + \delta A)(A^\dagger b + \text{err}(A^\dagger b)) = b + \delta b$, where $\|\delta A\| \lesssim \gamma_{n^2}\|A\|, \|\delta b\| \lesssim \gamma_{n^2}\|b\|$ [7, Theorem 19.5].

3. Randomized Solver As Iterative Refinement

In this section, we present a novel approach for constructing a fast and stable randomized least squares solver by iteratively refining an approximate solver which we call a meta-algorithm, *e.g.* sketch-and-apply or early stopped iterative randomized solver. We introduce two ways to do the refinement: iterative refinement and recursive refinement. Both refinement process starts from a meta-algorithm and improve the previous solution by correcting it based on the residual error. The key difference between iterative and recursive refinement processes is that iterative refinement improves the solution by applying the meta-algorithm at each step to correct the residual, while recursive refinement refines the solution by repeatedly applying the same current solver to the residual error.

SIR: Sketched Iterative Refinement
Input :1
Output:2
if $N = 0$ **then**
  Return $\text{SIR}_0^{\text{meta}}(b)$ **Via** a meta-algorithm $\text{SIR}_0^{\text{meta}}(b) = \text{ALG}^{\text{meta}}(A^\top b)$;
  ; /* Initialization via Meta-Algorithm */
end
for $i \leftarrow 1$ **to** N **by** 1 **do**
  $\text{SIR}_i^{\text{meta}}(b) := \text{SIR}_{i-1}^{\text{meta}}(b) + \text{ALG}^{\text{meta}}(A^\top(b - A \cdot \text{SIR}_{i-1}^{\text{meta}}(b)))$; /* Iterative Refinement via Meta-Algorithm */
end
Return $\text{SIR}_N^{\text{meta}}(b)$

Algorithm 1: Sketched **I**terative **R**efinement

3.1. Iterative and Recursive refinement

Iterative Refinement Iterative refinement [3], [32], [33] is the classical approach to improving the quality of a computed solution in numerical linear algebra. The idea of iterative refinement is simple, to improve the quality of an approximate solution x_i, solve for the error $\delta x_i = x - x_i$ via approximately solving $\delta x_i := \arg\min_{\delta x_i} \|b - Ax_i - A\delta x_i\|$. Classically, the inexact solve used in the refinement step is a classical direct solver such as QR factorization computed in lower numerical precision (i.e., single precision), and all the other steps are performed in higher precision (e.g., double precision) [2], [34]. In our paper, we design an iterative algorithm, where each step incorporates the concept of iterative refinement, using a fast randomized linear solver to approximately solve the system. The algorithm is detailed in Algorithm 3.

Recursive Refinement We also introduce a novel way to implement an iterative refinement process which we call it (sketched) recursive refinement approach. Sketched Recursive Refinement process also iteratively refines the solution by incorporating corrections from previous iterations. Different from iterative refinement which updates the current solution by applying a fixed procedure to adjust the solution, recursive refinement refers back to itself to perform the next step and solve the problem in a nested fashion. The algorithm is detailed in Algorithm 4. Later, we demonstrate that recursive refinement is simply a reorganization of the computational steps in iterative refinement but the two types of refinement enjoy very different numerical stability behavior.

SRR: Sketched Recursive Refinement
Input :1
Output:2
if $N = 0$ **then**
  Return $\text{SRR}_0(b)$ **Via** meta-algorithm $\text{ALG}^{meta}(A^\top b)$;
end
for $i \leftarrow 1$ **to** N **by** 1 **do**
  $\text{SRR}_i(b) := \text{SRR}_{i-1}(b) + \text{SRR}_{i-1}(A^\top b - A^\top A \cdot \text{SRR}_{i-1}(b))$;
  /* Recursive Refinement */
end
Return $\text{SRR}_N(b)$

Algorithm 2: Sketched **R**ecursive **R**efinement.

Recursive Refinement as Reorganizing Computation We would like to point out that Recursive refinement and Iterative refinement perform the same if one uses exact arithmetic. With a linear meta-algorithm, *i.e.* $ALG^{meta}(A^\top b)$ can be represented as $TA^\top b + q$ for some matrix T which includes most useful randomized solver such as Sketch-and-Apply, the results of $\text{SIR}_N(b)$ and $\text{SRR}_{\log_2 N}(b)$ are the same and both can be presented in the same form as geometric series as $x = \sum_{i=0}^{N}(I - TA)^i Tb$ with same amount of compute $O(Nmn)$. This means that Recursive Refinement is just a reorganization of computation order in the Iterative Refinement procedure and would generate the same computational result if one use exact arithmetic. However, in the following discussion, we show that Recursive Refinement and Iterative Refinement behave very differently when using a floating point arithmetic.

Equivalence between Iterative Refinement and Sketch-and-Precondition Iterative Refinement (Iterative Hessian Sketching) and the Sketch-and-Precondition approach are commonly regarded as two distinct methodologies for designing iterative randomized least squares solvers. In this remark, we demonstrate the surprising equivalence between sketched iterative refinement and the sketch-and-precondition method. This insight provides a unified perspective on modern randomized linear solvers and suggests new possibilities for designing iterative least squares solvers as iterative refinement. Specifically, sketched iterative refinement (or Iterative Hessian Sketching) can be interpreted as a preconditioned Jacobi iteration using the sketched matrix. Assuming the meta-algorithm has a linear form $ALG^{meta}(A^\top b) = TA^\top b + q$, the sketched iterative refinement performs iteration $x_{i+1} = (I - T^{-1}A^\top A)x_i + T^{-1}A^\top b$, which is equivalent to Jacobi iteration with pre-conditer T. This indicates that the iterative refinement process implicitly acts as a preconditioning mechanism, enjoying the same convergence guarantees as described in [9]. Moreover, this new understanding of iterative refinement allows for a more detailed analysis of numerical stability of the solver shown in Section 5.2.

Convergence of Iterative and Recursive refinement In this section we demonstrate the convergence of $\|x - x^*\|$.

Theorem 3 (Convergence of Iterative/Recursive Refinement)**.** *Suppose that the meta-algorithm has a linear form $ALG^{meta}(A^\top b) = TA^\top b + q$, then SIR and SRR are convergent if and only if $\rho(I - TA) < 1$, with*

- $\|SIR_t^{meta}(A^\top b) - x^*\| \le \|SIR_0^{meta}(A^\top b) - x^*\| e^{-\alpha t}$,
- $\|SRR_t^{meta}(A^\top b) - x^*\| \le \|SRR_0^{meta}(A^\top b) - x^*\| e^{-\alpha 2^t}$

where $\alpha = -ln(\rho(I - TA))$ and x^* is the true solution which satisfies $x^* = \arg\min_x \|Ax - b\|$.

Remark 1 (Selection of Meta-Algorithm). If one use the standard sketch-and-solve algorithm as the meta-algorithm, t-th iteration of SIR algorithm convergence at speed $(\frac{1}{(1-\eta)^2} - 1)^t$ for a sketching matrix with distortion η where $\eta \in (0, 1)$. This means necessary sketching dimension depends on the intrinsic complexity of the problem. The algorithm would diverge if the "sufficient sketching dimension" condition is violated [21], [24]. To remove such condition, we consider a 2-step Krylov-based sketch-and-solve solver as the meta-algorithm, now the t-th iteration of SIR algorithm convergence at speed $\min\{\eta^k, \frac{1}{\eta^k}\}$ which removes the requirement that $\eta < 1$ (detailed proof shown in Appendix 8.4). We use the 2-step Krylov solver both for the stability analysis in Section 5.2 and the implementation in Section 6.

4. Randomized Solver As Iterative Refinement

In this section, we present a novel approach for constructing a fast and stable randomized least squares solver by iteratively refining an approximate solver which we call a meta-algorithm, *e.g.* sketch-and-apply or early stopped iterative randomized solver. We introduce two ways to do the refinement: iterative refinement and recursive refinement. Both refinement process starts from a meta-algorithm and improve the previous solution by correcting it based on the residual error. The key difference between iterative and recursive refinement processes is that iterative refinement improves the solution by applying the meta-algorithm at each step to correct the residual, while recursive refinement refines the solution by repeatedly applying the same current solver to the residual error.

SIR: Sketched Iterative Refinement
Input :1
Output :2
if $N = 0$ **then**
  Return $\text{SIR}_0^{\text{meta}}(b)$ **Via** a meta-algorithm $\text{SIR}_0^{\text{meta}}(b) = \text{ALG}^{\text{meta}}(A^\top b)$;
  ; `/* Initialization via Meta-Algorithm */`
end
for $i \leftarrow 1$ **to** N **by** 1 **do**
  $\text{SIR}_i^{\text{meta}}(b) := \text{SIR}_{i-1}^{\text{meta}}(b) + \text{ALG}^{\text{meta}}(A^\top(b - A \cdot \text{SIR}_{i-1}^{\text{meta}}(b)))$; `/* Iterative Refinement via Meta-Algorithm */`
end
Return $\text{SIR}_N^{\text{meta}}(b)$

Algorithm 3: Sketched **I**terative **R**efinement

4.1. Iterative and Recursive refinement

Iterative Refinement Iterative refinement [3], [32], [33] is the classical approach to improving the quality of a computed solution in numerical linear algebra. The idea of iterative refinement is simple, to improve the quality of an approximate solution x_i, solve for the error $\delta x_i = x - x_i$ via approximately solving $\delta x_i := \arg\min_{\delta x_i} \|b - Ax_i - A\delta x_i\|$. Classically, the inexact solve used in the refinement step is a classical direct solver such as QR factorization computed in lower numerical precision (i.e., single precision), and all the other steps are performed in higher precision (e.g., double precision) [2], [34]. In our paper, we design an iterative algorithm, where each step incorporates the concept of iterative refinement, using a fast randomized linear solver to approximately solve the system. The algorithm is detailed in Algorithm 3.

Recursive Refinement We also introduce a novel way to implement an iterative refinement process which we call it (sketched) recursive refinement approach. Sketched Recursive Refinement process also iteratively refines the solution by incorporating corrections from previous iterations. Different from iterative refinement which updates the current solution by applying a fixed procedure to adjust the solution, recursive refinement refers back to itself to perform the next step and solve the problem in a nested fashion. The algorithm is detailed in Algorithm 4. Later, we demonstrate that recursive refinement is simply a reorganization of the computational steps in iterative refinement but the two types of refinement enjoy very different numerical stability behavior.

SRR: Sketched Recursive Refinement
Input :1
Output :2
if $N = 0$ **then**
  Return $\text{SRR}_0(b)$ **Via** meta-algorithm $\text{ALG}^{meta}(A^\top b)$;
end
for $i \leftarrow 1$ **to** N **by** 1 **do**
  $\text{SRR}_i(b) := \text{SRR}_{i-1}(b) + \text{SRR}_{i-1}(A^\top b - A^\top A \cdot \text{SRR}_{i-1}(b))$;
  `/* Recursive Refinement */`
end
Return $\text{SRR}_N(b)$

Algorithm 4: Sketched **R**ecursive **R**efinement.

Recursive Refinement as Reorganizing Computation We would like to point out that Recursive refinement and Iterative refinement perform the same if one uses exact arithmetic. With a linear meta-algorithm, *i.e.* $ALG^{meta}(A^\top b)$ can be represented as $TA^\top b + q$ for some matrix T which includes most useful randomized solver such as Sketch-and-Apply, the results of $\text{SIR}_N(b)$ and $\text{SRR}_{\log_2 N}(b)$ are the same and both can be presented in the same form as geometric series as $x = \sum_{i=0}^{N}(I - TA)^i Tb$ with same amount of compute $O(Nmn)$. This means that Recursive Refinement is just a reorganization of computation order in the Iterative Refinement procedure and would generate the same computational result if one use exact arithmetic. However, in the following discussion, we show that Recursive Refinement and Iterative Refinement behave very differently when using a floating point arithmetic.

Equivalence between Iterative Refinement and Sketch-and-Precondition Iterative Refinement (Iterative Hessian Sketching) and the Sketch-and-Precondition approach are commonly regarded as two distinct methodologies for designing iterative randomized least squares solvers. In this remark, we demonstrate the surprising equivalence between sketched iterative refinement and the sketch-and-precondition method. This insight provides a unified perspective on modern randomized linear solvers and suggests new possibilities for designing iterative least squares solvers as iterative refinement. Specifically, sketched iterative refinement (or Iterative Hessian Sketching) can be interpreted as a preconditioned Jacobi iteration using the sketched matrix. Assuming the meta-algorithm has a linear form $ALG^{meta}(A^\top b) = TA^\top b + q$, the sketched iterative refinement performs iteration $x_{i+1} = (I - T^{-1}A^\top A)x_i + T^{-1}A^\top b$, which is equivalent to Jacobi iteration with pre-conditer T. This indicates that the iterative refinement process implicitly acts as a preconditioning mechanism, enjoying the same convergence guarantees as described in [9]. Moreover, this new understanding of iterative refinement allows for a more detailed analysis of numerical stability of the solver shown in Section 5.2.

Convergence of Iterative and Recursive refinement In this section we demonstrate the convergence of $\|x - x^*\|$.

Theorem 4 (Convergence of Iterative/Recursive Refinement). *Suppose that the meta-algorithm has a linear form $ALG^{meta}(A^\top b) = TA^\top b + q$, then SIR and SRR are convergent if and only if $\rho(I - TA) < 1$, with*

- $\|SIR_t^{meta}(A^\top b) - x^*\| \le \|SIR_0^{meta}(A^\top b) - x^*\| e^{-\alpha t}$,
- $\|SRR_t^{meta}(A^\top b) - x^*\| \le \|SRR_0^{meta}(A^\top b) - x^*\| e^{-\alpha 2^t}$

where $\alpha = -ln(\rho(I - TA))$ and x^* is the true solution which satisfies $x^* = \arg\min_x \|Ax - b\|$.

Remark 2 (Selection of Meta-Algorithm). If one use the standard sketch-and-solve algorithm as the meta-algorithm, t-th iteration of SIR algorithm convergence at speed $(\frac{1}{(1-\eta)^2} - 1)^t$ for a sketching matrix with distortion η where $\eta \in (0, 1)$. This means necessary sketching dimension depends on the intrinsic complexity of the problem. The algorithm would diverge if the "sufficient sketching dimension" condition is violated [21], [24]. To remove such condition, we consider a 2-step Krylov-based sketch-and-solve solver as the meta-algorithm, now the t-th iteration of SIR algorithm convergence at speed $\min\{\eta^k, \frac{1}{\eta^k}\}$ which removes the requirement that $\eta < 1$ (detailed proof shown in Appendix 8.4). We use the 2-step Krylov solver both for the stability analysis in Section 5.2 and the implementation in Section 6.

5. Fast and Stable Solver via Iterative and Recursive refinement

To construct a fast and stable randomized solver, we use Sketched Recursive Refinement as the meta-algorithm for a Sketched Iterative Refinement process. We call our algorithm Sketched Iterative and Recursive Refinement (SIRR) which is shown as algorithm 6 in the appendix. We also theoretically show that both iterative and recursive refinement are essential to achieve backward stability. The theoretical finding is also verified numerically in Section 6.

5.1. Sketched Iterative and Recursive Refinement

SIRR is Fast In this section, we first show that SIRR converges fast with a computational complexity at $O(n^3 + mn)$. Note that SIRR is a composite of meta-algorithm, so we examine the computational complexity and average convergence rate of meta-algorithm to show the whole computational complexity of SIRR.

Suppose that $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^{m\times 1}$ and upper triangular matrix $R \in \mathbb{R}^{n\times n}$, the computation of computational complexity follows:

- matrix and vector multiplication $A^\top b$: $O(mn)$
- solving triangular system $R^{-1}z$ for $z \in \mathbb{R}^{n\times 1}$: $O(n^2)$
- conducting QR factorization of SA: $O(sn^2) = O(n^3 log(n))$

For sketch-and-solve meta solver, the computational complexity is $O(n^2 + mn)$ and the convergence rate is $\frac{1}{(1-\eta)^2} - 1$. To reach machine precision, the iteration step is at most $O(log(\frac{1}{u}))$, thus the total computational complexity of SIR, SRR and SIRR are all $O(n^3 log(n) + log(\frac{1}{u})(n^2 + mn))$ matches the fast randomzied least square solvers such as Blendenpik [12] and FOSSILS [35].

5.2. SIRR in Floating Point Arithmetic

As shown in Section 4.1, SIRR is the same as SIR in exact arithmetic. In this section, we study the stability results for SIR, SRR and SIRR when one implements them in floating point arithmetic.

5.2.1. SIRR is forward Stable

In this section, we first prove that SIRR solver is forward stable, *i.e.* both the forward error $\|\hat{x} - x^*\|$ and the residual error $\|A\hat{x} - Ax^*\|$ converge geometrically for SIRR implemented in floating point arithmetic.

Definition (Forward Stability). A least-squares solver is forward stable if the computed solution $\hat{x}$ satisfies

$$\|\hat{x} - x^*\| \le \epsilon(\kappa\|x^*\| + \frac{\kappa^2}{\|A\|}\|r^*\|),$$

and is strongly forward stable if $\hat{x}$ satisfies

$$\|A(\hat{x} - x^*)\| \le \epsilon(\kappa\|r^*\| + \|A\|\|x^*\|),$$

where x^* is the exact solution, $r^* = b - Ax^*$ and $\epsilon \lesssim n^{\frac{3}{2}}$

Remark 3. This is the best error one can expect to achieve due to Wedin's theorem [4], where one always solving a perturbed problem $\text{argmin}_{y\in\mathbb{R}^n} \|(b+\delta b) - (A+\delta A)y\|$ in floating point arithmetic, where $\|\delta A\| \le \epsilon\|A\|$, $\|\delta b\| \le \epsilon\|b\|$.

Theorem 5. *For SIRR with meta-algorithm $ALG^{meta}(\cdot)$, which solves problem $x = \arg\min_y \|(A^\top A)y - r_A\|$, satisfying $ALG^{meta}(r_A) = (A^\top A)^{-1}r_A + c\|\hat{R}^{-\top}r_A\|\hat{R}^{-1}e$ where $\|e\| \lesssim 1$ and $c < 1$, the result $\hat{x}$ of SIRR is strongly forward stable, which satisfies*

$$\|\hat{x} - x^*\| \lesssim n^{\frac{3}{2}}(u\kappa\|x^*\| + \frac{u\kappa^2}{\|A\|}\|r^*\|),$$

$$\|A(\hat{x} - x^*)\| \lesssim n^{\frac{3}{2}}(u\kappa\|r^*\| + u\|A\|\|x^*\|),$$

With strongly forward stable, we can expect a non-pathological rounding error $\|\hat{x}\| \ge \|x^*\| + n^{\frac{3}{2}}\frac{u\kappa^2}{\|A\|}\|r^*\|$.

5.2.2. SIRR is Backward Stable

In this section, we provide the theoretical analysis showing that the Sketched Iterative and Recursive Refinement (SIRR) is provable backward stable when implemented in floating point arithmetic. To do this, we first find the requirement that the meta solver of the sketched iterative refinement needs to satisfy that can make SIR solver backward stable. Then we prove that Sketched Recursive Refinement can provably meet these requirements.

Theorem 6. *For simplicity, denote $\max\{u\kappa, \frac{1}{\kappa n^{\frac{3}{2}}}\}$ as $\tilde{\kappa}^{-1}$. Suppose that $un^{\frac{3}{2}}\|x^*\| \le \|b - Ax^*\|$ and the single step meta-solver $ALG(z)$ is $n^{\frac{3}{2}}\tilde{\kappa}^{-1}(\|Ax_z^*\| + u\kappa\|z\|) - un^{\frac{3}{2}}(\|Ax_z^*\| + \|z\|)$ accurate where x_z^* is the true solution of the least square problem,* i.e. $\text{argmin}_x \|Ax - z\|$. *Then SIR solver $x_{i+1} = ALG(b - Ax_i)$ will converge to a $(un^3\kappa\tilde{\kappa}^{-1}\|b - Ax_b^*\| + un^{\frac{3}{2}}\|x^*\|) - (un^{\frac{3}{2}}\|b - Ax_b^*\| + u^2n^3\|x^*\|)$ accurate solution which indicate a **backward stable** result by Lemma 2.*

Remark 4. Since SIR/SIRR solver enjoys non-pathological rounding error assumption $\|x^*\| + n^{\frac{3}{2}}\kappa^2 u\|b - Ax^*\| \lesssim \|\hat{x}\|$ (Theorem 5), we have $\overbrace{(n^{\frac{3}{2}}u + n^3\kappa^2u^2)}^{un^3\kappa\tilde{\kappa}^{-1}}\|b - Ax^*\| + un^{\frac{3}{2}}\|x^*\| \lesssim un^{\frac{3}{2}}\|b - Ax^*\| + un^{\frac{3}{2}}\|\hat{x}\| \lesssim un^{\frac{3}{2}}\|b - A\hat{x}\| + un^{\frac{3}{2}}\|\hat{x}\| \lesssim un^{\frac{3}{2}} + un^{\frac{3}{2}}\|\hat{x}\|$ and $un^{\frac{3}{2}}\|b - Ax^*\| + u^2n^3\|x^*\| \lesssim un^{\frac{3}{2}}\|b - Ax^*\| \lesssim un^{\frac{3}{2}}\|b - A\hat{x}\|$ based on the assumption that $un^{\frac{3}{2}}\|x^*\| \lesssim \|b - Ax^*\|$. By lemma 2, the solution has backward error $Be(\hat{x}) \lesssim n^2 u$ which indicates a backward stability result and *aligns the backward error estimation for QR-based solver [7, Theorem 19.5] which also dependency of matrix size at n^2.*

Then we study the stability result of SRR implemented in floating point arithmetic. We show that SRR can be backward stable only when $\frac{\|b-Ax^*\|}{\|Ax^*\|} = O(1)$ and is not backward stable when the residual $\|b - Ax^*\|$ is small. However, SRR provides an approximate solver that satisfies the assumption we require for the meta-algorithm in the backward stable result in Theorem 6.

Theorem 7. *For meta-algorithm $SRR_0(\cdot)$, which solves problem $x = \arg\min_y \|(A^\top A)y - r_A\|$, satisfying $SRR_0(r_A) = x^* + (a_1\|x^*\| + a_2\|Ax^*\|)\hat{R}^{-1}e_1 + (b_1\|x^*\| + b_2\|Ax^*\|)(A^\top A)^{-1}e_2$, where $x^* = (A^\top A)^{-1}r_A$ and*

$$\kappa a_1 + a_2 \asymp c, \kappa^2 b_1 + \kappa b_2 \asymp c, \|e_{1,2}\| \lesssim 1,$$

and $N = O(\log_2(\frac{\log(\tilde{\kappa}^{-1}n^{\frac{3}{2}})}{\log(c)}))$, the solution of corresponding SRR_N is $O(n^{\frac{3}{2}}(\tilde{\kappa}^{-1}\|Ax^*\| + u\|x^*\| + u\kappa\tilde{\kappa}^{-1}\|b\|), n^{\frac{3}{2}}(u\|Ax^*\| + u^2 n^{\frac{3}{2}}\|x^*\| + u\|b\|))$-accurate. As $N \to \infty$, SRR_N converges to a (a, b)–accurate solution $SRR_\infty(b)$ with

$$\hat{a} \lesssim (u^2\kappa^2 n^3\|Ax^*\| + un^{\frac{3}{2}}\|x^*\| + u^2\kappa^2 n^3\|b\|),$$
$$\hat{b} \lesssim (un^{\frac{3}{2}}\|Ax^*\| + u^2 n^3\|x^*\| + un^{\frac{3}{2}}\|b\|).$$

Remark 5. Theorem 7 indicates that SRR has the same backward error level as SIRR when $\frac{\|b-Ax^*\|}{\|Ax^*\|} = O(1)$. We verified numerically that SRR solver only is not backward stable when $\|b - Ax^*\|$ is large. The result is presented in Figure 2. This illustrates that our theoretical result for SRR is tight.

Although $SRR_N(\cdot)$ is not backward stable on its own, it satisfies the requirements (for $u\|x^*\| \lesssim \tilde{\kappa}^{-1}\|Ax^*\|$ and $un^{\frac{3}{2}}\|x^*\| \lesssim \|Ax^*\|$) of the meta-algorithm in Theorem 6 to achieve a backward-stable SIR solver, as demonstrated in Theorem 7. This implies that by using SRR as the meta-algorithm for the SIR solver—*i.e.*, the SIRR solver—it can be proven to be backward-stable, provided the meta-algorithm satisfies the conditions outlined in Theorem 7. Finally, we show that the two-step Krylov-based meta-algorithm, described in Remark 2, meets the meta-algorithm criteria specified in Theorem 7.

Lemma 8. *The result of 2-step Krylov-based meta-algorithm (Appendix Algorithm 5) for solving $x = \arg\min_y \|(A^\top A)y - r_A\|$ satisfies*

$$\hat{x} = x^* + u\kappa n^{\frac{3}{2}}\|Ax^*\|\hat{R}^{-1}e_1 + un^{\frac{3}{2}}\|Ax^*\|(A^\top A)^{-1}e_2,$$

where $x^* = (A^\top A)^{-1}r_A$ and $\|e_{1,2}\| \lesssim 1$. As a result, SIRR with Krylov-based meta-algorithm is backward stable.

6. Numerical Experiments

In this section, we compare SIR, SRR and SIRR solver to verify our theoretical findings. We also compare it with QR-based direct solver (*mldivide* (MATLAB)) and FOSSILS in concurrent work [35] to show that SIRR solver can beat the state-of-the-art randomized/direct solvers in realistic applications.

Error metrics Following [30], [35], we test three useful error metrics for all randomized least square solvers:

1. **Forward error.** The forward error quantifies how close the computed solution $\hat{x}$ is to the true solution x, *i.e.* $\mathrm{FE}(\hat{x}) := \frac{\|x-\hat{x}\|}{\|x\|}$.
2. **Residual error.** The (relative) residual error measures the *suboptimality* of $\hat{x}$ as a solution to the least-squares minimization problem, *i.e.* $\mathrm{RE}(\hat{x}) := \frac{\|r(x)-r(\hat{x})\|}{\|r(x)\|}$.
3. **Backward error.** The (relative) backward error [7, Section 20.7] is $\mathrm{BE}_b(\hat{x}) := \min_v \frac{\|\Delta A\|_F}{\|A\|_F}$ where $\hat{x} = \arg\min_v \|b - (A + \Delta A)v\|$.

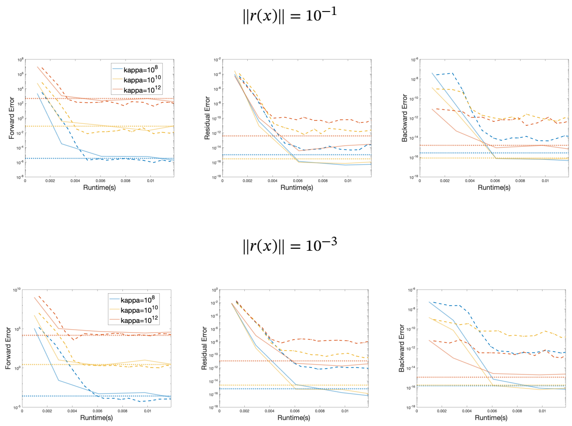

Figure 1. Results of SIRR with sketch and solve Initialization are shown as solid curve lines, with reference accuracy for MATLAB function $A\backslash b$ shown as dotted constant lines and IHS-Krylov shown as dotted curve lines

Experiment Setup We adopt a similar setup to [30], [37] in most of experiments. We set $A \in R^{m\times n}$, sketching matrix $S \in R^{s\times m}$, and choose parameters $\kappa \geq 1$ for the condition number of A and $\beta \geq 0$ for the residual norm $\|r(x)\|$. To generate A, x, and b, do the following:

- Choose Haar random orthogonal matrices $U = [U_1\ U_2]$ in $\mathbb{R}^{m\times m}$ and V in $\mathbb{R}^{n\times n}$, and partition U so that $U_1 \in \mathbb{R}^{m\times n}$.
- Set $A := U_1\Sigma V^T$ where Σ is a diagonal matrix with logarithmically equispaced entries between 1 and $\frac{1}{\kappa}$.
- Form vectors w in $\mathbb{R}^n$, z in $\mathbb{R}^{m-n}$ with independent standard Gaussian entries.
- Define the solution $x := \frac{w}{\|w\|}$, residual $r(x) = \beta \cdot U_2 z/\|U_2 z\|$, and right-hand side $b := Ax + r(x)$.

We also experiment on kernel regression task, where we consider least-squares problems for fitting the SUSY dataset using a linear combination of kernel functions. Similar to [30], [35], we generate real-valued least-squares problems of dimension $m = 10^6$ and $n \in [10^1, 10^3]$.

Both Iterative and Recursive Refinement is Essential In this section, we conduct numerical experiments to demonstrate that both iterative and recursive refinement are essential for constructing a backward-stable solver. To illustrate this, we compare SIR, SRR, and SIRR, each using a two-step Krylov solver as algorithm 5 in appendix. as the meta-solver, under varying levels of condition numbers and $\frac{\|b-Ax^*\|}{\|Ax^*\|}$ to validate theorem 7. Figure 1 shows that the SIR solver is not backward stable, while the SIRR solver achieves near machine-precision backward error. In a second experiment, we compare SIRR and SRR across different levels of residual size. Our theoretical results in theorem 7 indicate that when the magnitude of the residual $\|b - Ax^*\|$ exceeds the signal $\|Ax^*\|$, SRR achieves the same backward stability as SIRR. However, SRR cannot achieve the same level of backward stability as SIRR when the residual $\|b - Ax^*\|$ is small. Figure 2 confirms this result, showing that the backward error of SRR converges to that of SIRR in the white region and reaches the same level as SIRR in the grey region. In all experiments, we set $m = 2000, n = 50, s = 200$.

SIRR VS FOSSILS We also compare our SIRR solver with FOSSILS in concurrent work [35][1] by two experiments. In the first experiments, we adopt the same setting as [35], where a family of problems is

[1]We use the code from https://github.com/eepperly/Stable-Randomized-Least-Squares for the FOSSILS algorithm.

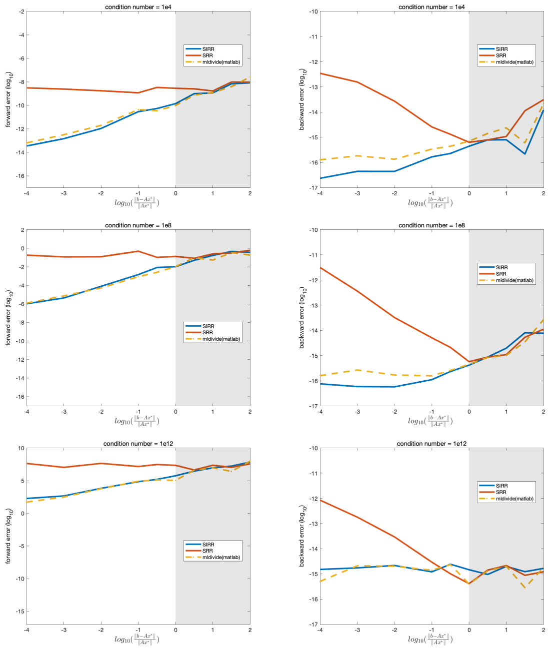

Figure 2. Forward error (left) and backward error (right) under different $\|b - Ax^*\|/\|Ax^*\|$. SRR is not backward stable when $\|b - Ax^*\|$ is small while SIRR can achieve backward stable estimates for all cases. We also plotted the result for *mldivide*(MATLAB) solver here for reference.

generated of increasing difficulty, with condition number κ and error size $\|b - Ax^*\|$ satisfying

$$\text{difficulty} = \kappa = \frac{\|b - Ax^*\|}{u} \in [10^0, 10^{16}].$$

We set $m = 5000, n = 200, s = 600$ for problem size.Figure 3 shows the forward error and backward error of SIRR and FOSSILS in problems of different difficulties, where both sketching algorithms have a similar forward stability while SIRR exhibits a better backward error performance.

In the second experiment we give further insight into the difference between SIRR and FOSSILS. First we test on kernel regression task to see the runtime of sketching solver and MATLAB solver (*mldivide*) in different sizes of n. then we test the dependence of the stability of different solvers on sketching dimension by changing sketching dimension and counting the times that algorithm fails to converge in 100 runs. The left of Figure 6 shows that SIRR and FOSSILS need comparable time to reach the same accuracy, faster than MATLAB solver when $log(n) \geq 2.4$. The right figure illustrates the fail rate of SIRR and FOSSILS, which is the ratio of times failing to converge to a backward stable result in 100 runs. The fail rate of FOSSILS linearly decreases with the growth of sketching dimension, while SIRR achieves great stability when sketch dimension $d \geq 1.75n$.

Error scale with n In this experiment we show that for sketching solver and MATLAB direct solver, it is inevitable that the error is in scale with n. We fix $m = 10000, \kappa = 10^8, \|b - Ax^*\| = 10^{-3}$ and change $n \in [100, 1600]$ with sketching dimension $s = 4 * n$. Figure 4 shows the dependence of forward error and backward error on n of different solvers. Note that three solvers actually have comparable forward error around 10^{-6} where MATLAB solver has slight edge. The dependence on n is significant for backward error, where SIRR and FOSSILS appear to have a lower order of dependence.

Remark 6. Empirically, the growth of backward error as the matrix size increases is slower than the theoretical prediction in Remark 4 as n^2. One possible reason is that the test matrix is random, and its randomness may not behave adversarially, leading to better performance.

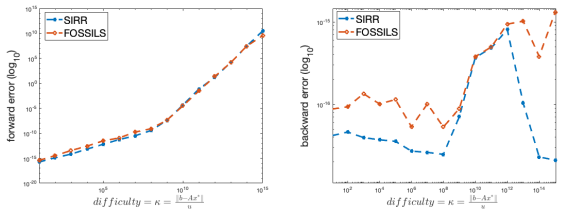

Figure 3. Comparing the Forward error (left) and backward error (right) of SIRR and FOSSILS on problems with different difficulties. SIRR has better backward stability in most situations and similar forward stability compared to FOSSILS.

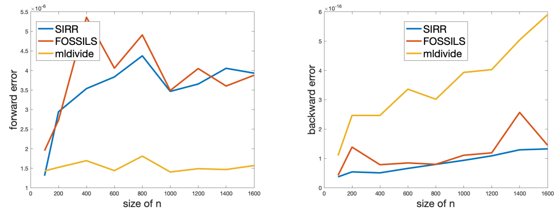

Figure 4. Forward error (left) and backward error (right) of different sizes of n.

6.1. Comparison with FOSSILS

We would like to highlight a concurrent work [35], which also developed a backward stable solver with a computational complexity of $O(mn + n^3)$. However, the FOSSILS solver proposed in their work follows a two-stage approach, where each stage involves an iterative process. In contrast, our algorithm is a single-stage solver that offers the flexibility to stop at any point during the computation, making it more adaptable for scenarios where early termination is necessary or beneficial. In this section we also compare our SIRR solver with the FOSSILS solver in both synthetic matrices (Figure 3) and realistic kernel regression datasets (Figure 4). We demonstrate that the SIRR solver consistently achieves better backward stability than the FOSSILS solver across various difficulty levels, while requiring a similar amount of computing time. Notably, when the sketch dimension is small, SIRR is less prone to failure compared to FOSSILS.

SIRR and FOSSILS with Different Embedding Quality We have proved that SIRR solver with 2-step Krylov-based meta-algorithm has a good convergence even in cases where sketching quality is bad and the distortion η of sketching matrix is high. In this section, we give experiment results of the convergence performance of two solvers, SIRR and FOSSILS, in different embedding quality, which depends on the relative sketch dimensions $\frac{s}{n}$. In the result, the fail rate means the ratio of times that the solver fails to converge in 100 parallel experiments. In different experiments, $m = 2000$ and $n = 100$, $\kappa \in \{10^4, 10^8, 10^{12}\}$, $\|b - Ax^*\| \in \{10^{-1}, 10^{-3}\}$. The results are presented in Figure 5.

References

[1] S. Kaczmarz, “Angenaherte auflosung von systemen linearer gleichungen. bull. internat. acad. polon. sci. lett”, 1937.

[2] G. H. Golub and J. H. Wilkinson, “Note on the iterative refinement of least squares solution”, *Numerische Mathematik*, vol. 9, no. 2, pp. 139–148, 1966.

[3] C. B. Moler, “Iterative refinement in floating point”, *Journal of the ACM (JACM)*, vol. 14, no. 2, pp. 316–321, 1967.

[4] P.-Å. Wedin, “Perturbation theory for pseudo-inverses”, *BIT Numerical Mathematics*, vol. 13, pp. 217–232, 1973.

[5] C. C. Paige and M. A. Saunders, “Lsqr: An algorithm for sparse linear equations and sparse least squares”, *ACM Transactions on Mathematical Software (TOMS)*, vol. 8, no. 1, pp. 43–71, 1982.

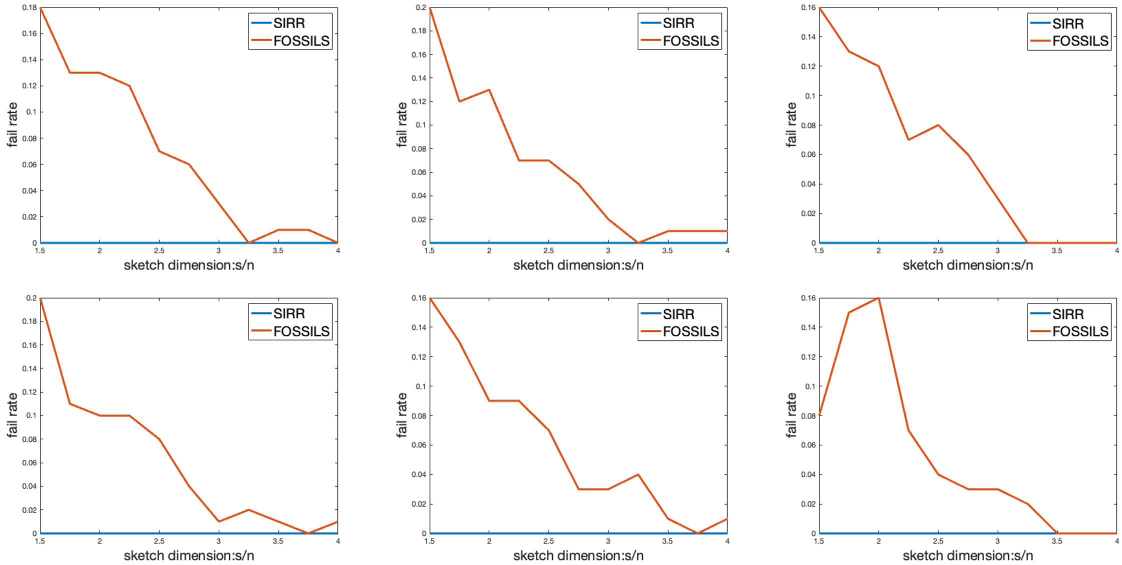

Figure 5. In first row $||b - Ax^*|| = 10^{-1}$ and in second row $||b - Ax^*|| = 10^{-3}$ with $\kappa = 10^4, 10^8, 10^{12}$ from left to right.

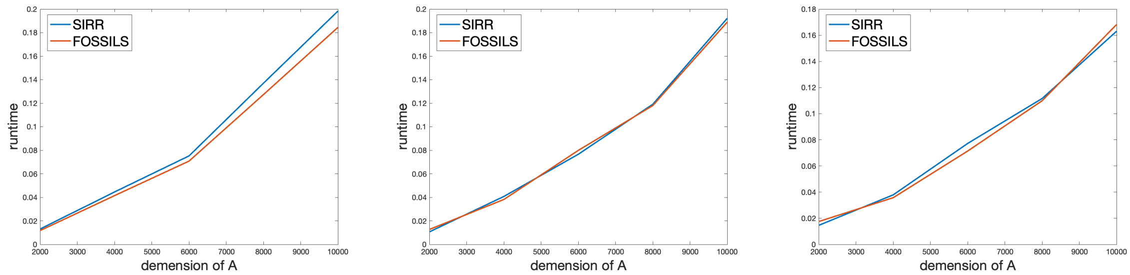

Figure 6. In 3 figures, $\kappa = 10^4, 10^8, 10^{12}$ from left to right. SIRR has roughly the same computational cost as FOSSILS as randomized solver.

[6] A. Greenbaum, *Iterative methods for solving linear systems*. SIAM, 1997.

[7] N. J. Higham, *Accuracy and stability of numerical algorithms*. SIAM, 2002.

[8] J. Liesen and P. Tichỳ, "Convergence analysis of krylov subspace methods", *GAMM-Mitteilungen*, vol. 27, no. 2, pp. 153–173, 2004.

[9] V. Rokhlin and M. Tygert, "A fast randomized algorithm for overdetermined linear least-squares regression", *Proceedings of the National Academy of Sciences*, vol. 105, no. 36, pp. 13 212–13 217, 2008.

[10] N. Ailon and B. Chazelle, "The fast johnson–lindenstrauss transform and approximate nearest neighbors", *SIAM Journal on computing*, vol. 39, no. 1, pp. 302–322, 2009.

[11] M. Tygert, "A fast algorithm for computing minimal-norm solutions to underdetermined systems of linear equations", *arXiv preprint arXiv:0905.4745*, 2009.

[12] H. Avron, P. Maymounkov, and S. Toledo, "Blendenpik: Supercharging lapack's least-squares solver", *SIAM Journal on Scientific Computing*, vol. 32, no. 3, pp. 1217–1236, 2010.

[13] M. W. Mahoney *et al.*, "Randomized algorithms for matrices and data", *Foundations and Trends® in Machine Learning*, vol. 3, no. 2, pp. 123–224, 2011.

[14] S. Gratton, P. Jiránek, and D. Titley-Peloquin, "On the accuracy of the karlson–walden estimate of the backward error for linear least squares problems", *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 3, pp. 822–836, 2012.

[15] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013.

[16] X. Meng, M. A. Saunders, and M. W. Mahoney, "Lsrn: A parallel iterative solver for strongly over-or underdetermined systems", *SIAM Journal on Scientific Computing*, vol. 36, no. 2, pp. C95–C118, 2014.

[17] D. P. Woodruff *et al.*, "Sketching as a tool for numerical linear algebra", *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 1–2, pp. 1–157, 2014.

[18] P. Drineas and M. W. Mahoney, "Randnla: Randomized numerical linear algebra", *Communications of the ACM*, vol. 59, no. 6, pp. 80–90, 2016.

[19] J. K. Fitzsimons, M. A. Osborne, S. J. Roberts, and J. F. Fitzsimons, "Improved stochastic trace estimation using mutually unbiased bases", *arXiv preprint arXiv:1608.00117*, 2016.

[20] L. Lin, Y. Saad, and C. Yang, "Approximating spectral densities of large matrices", *SIAM review*, vol. 58, no. 1, pp. 34–65, 2016.

[21] M. Pilanci and M. J. Wainwright, "Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares", *Journal of Machine Learning Research*, vol. 17, no. 53, pp. 1–38, 2016.

[22] H. Avron, K. L. Clarkson, and D. P. Woodruff, "Faster kernel ridge regression using sketching and preconditioning", *SIAM Journal on Matrix Analysis and Applications*, vol. 38, no. 4, pp. 1116–1138, 2017.

[23] L.-H. Lim and J. Weare, "Fast randomized iteration: Diffusion monte carlo through the lens of numerical linear algebra", *SIAM Review*, vol. 59, no. 3, pp. 547–587, 2017.

[24] J. Wang, J. Lee, M. Mahdavi, M. Kolar, and N. Srebro, "Sketching meets random projection in the dual: A provable recovery algorithm for big and high-dimensional data", in *Artificial Intelligence and Statistics*, PMLR, 2017, pp. 1150–1158.

[25] Y. Yang, M. Pilanci, and M. J. Wainwright, "Randomized sketches for kernels: Fast and optimal nonparametric regression", 2017.

[26] I. K. Ozaslan, M. Pilanci, and O. Arikan, "Iterative hessian sketch with momentum", in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 7470–7474.

[27] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, "Streaming low-rank matrix approximation with an application to scientific simulation", *SIAM Journal on Scientific Computing*, vol. 41, no. 4, A2430–A2463, 2019.

[28] T. D. Ahle, M. Kapralov, J. B. Knudsen, *et al.*, "Oblivious sketching of high-degree polynomial kernels", in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, 2020, pp. 141–160.

[29] P.-G. Martinsson and J. A. Tropp, "Randomized numerical linear algebra: Foundations and algorithms", *Acta Numerica*, vol. 29, pp. 403–572, 2020.

[30] E. N. Epperly, "Fast and forward stable randomized algorithms for linear least-squares problems", *arXiv preprint arXiv:2311.04362*, 2023.

[31] Y. Hur, J. G. Hoskins, M. Lindsey, E. M. Stoudenmire, and Y. Khoo, "Generative modeling via tensor train sketching", *Applied and Computational Harmonic Analysis*, vol. 67, p. 101 575, 2023.

[32] J. H. Wilkinson, *Rounding errors in algebraic processes*. SIAM, 2023.

[33] C. W. Wu, M. S. Squillante, V. Kalantzis, and L. Horesh, "Stable iterative refinement algorithms for solving linear systems", *arXiv preprint arXiv:2309.07865*, 2023.

[34] E. Carson and I. Daužickaitė, "A comparison of mixed precision iterative refinement approaches for least-squares problems", *arXiv preprint arXiv:2405.18363*, 2024.

[35] E. N. Epperly, M. Meier, and Y. Nakatsukasa, "Fast randomized least-squares solvers can be just as accurate and stable as classical direct solvers", *arXiv preprint arXiv:2406.03468*, 2024.

[36] Y. Khoo, Y. Peng, and D. Wang, "Nonparametric estimation via variance-reduced sketching", *arXiv preprint arXiv:2401.11646*, 2024.

[37] M. Meier, Y. Nakatsukasa, A. Townsend, and M. Webb, "Are sketch-and-precondition least squares solvers numerically stable?", *SIAM Journal on Matrix Analysis and Applications*, vol. 45, no. 2, pp. 905–929, 2024.

Orgization of the Appendix

The appendix is structured as follows. In section 7.2, we present affiliate algorithms in this paper which are employed in practice. Then we give theoretical analysis of our method which guarantees the stability and convergence of our method in section 8.

7. Details of Algorithms in Practice

7.1. random matrix

In many applications, it is crucial to construct a subspace embedding without prior knowledge of the target subspace. Such embeddings are known as oblivious subspace embeddings. Typically, the singular values of specific random matrices are bounded with high probability, which makes them well-suited for subspace embedding. Various designs of random matrices exist that exhibit both strong computational and mathematical properties:

- **Gaussian embedding**: $S = \mathbb{R}^{s\times m}$ with $i.i.d$ $N(0, \frac{1}{s})$ entries. The normalization $\frac{1}{s}$ ensures that S preserves the 2-norm in expectation, e.g. $E\|Sx\|_2^2 = \|x\|_2^2$.
- **Subsampled randomized trigonometric transform (SRTT)**[10]: $S = \sqrt{\frac{m}{s}}RDF \in \mathbb{R}^{s\times m}$, where $R \in \mathbb{R}^{s\times m}$ is an uniformly random set of s rows drawn from the identity matrix I_m, and $D \in \mathbb{R}^{m\times m}$ is a random diagonal matrix with $uniform(\pm 1)$ entries, and $F \in \mathbb{R}^{m\times m}$ is a DCT_2 matrix. SRTT requires less time in matrix/vector multiplication with cost $O(m\log(m))$ and has the same embedding property when $s \approx n\log(n)$.
- **Sparse random matrices**[1]: $S = [s_1, s_2, \cdots, s_m] \in \mathbb{R}^{s\times m}$, where s_i are sparse vectors, which means for each i, s_i has exactly ζ nonzero entries, equally likely to be $\pm\sqrt{\frac{1}{\zeta}}$. The cost of matrix/vector multiplication is $O(\zeta m)$, and it's embedding has distortion η when $s \approx \frac{n\log(n)}{\eta^2}$ and $\zeta \approx \frac{\log(n)}{\eta}$.

We use Sparse random matrices in our experiment which requires less operation and storage in computation.

7.2. Krylov-based meta-algorithm

A Krylov-based meta-algorithm is employed in our experiment, for it has a better convergence rate and is indifferent to the quality of embedding, making our solver more stable and faster even in worst cases. We present Krylov-based meta-algorithm in algorithm 5.

```
Krylov-based meta-algorithm
Input  :1
Output :2
y_0 = (R^T R)^{-1} A^T b;
Y = y_0 ;                                   /* Initialization via sketch-and-solve */
for i ← 1 to K by 1 do
    y_{i+1} = (R^T R)^{-1} A^T (b − A y_i);
    Y = [Y, y + i + 1]
end
a = (AY)^† b
Return ALG^meta(b) = Y a
```

Algorithm 5: Krylov-based meta-algorithm

7.3. Sketched Iterative and Recursive Refinement

Sketched Iterative and Recursive Refinement (SIRR) is provably fast and stable and is designed based on Sketched Iterative Refinement (SIR) and Sketched Recursive Refinement (SRR). We present SIRR in algorithm 6.

```
SIRR: Sketched Iterative Recursive Refinement
Input  :1
Output :2
if N = 0 then
    Initialize SIRR_0(b) = (SA)^† Sb ;
    ;                                        /* Initialization via sketch-and-solve */
end
for i ← 1 to N by 1 do
    SIRR_i(b) := SIRR_{i−1}(b) + SRR_T(A^T(b − A · SIRR_{i−1}(b)));   /* Iterative Refinement via Sketched Recursive
     Refinement (SRR) */
end
Return SIRR_N(b)
```

Algorithm 6: **S**ketched **I**terative **R**ecursive **R**efinement.

8. PROOF OF MAIN RESULTS

In this section, we first establish some fundamental numerical results, which serve as the foundation for the subsequent numerical analysis. Then we examine the convergence of the iterative algorithm to show our method is theoretically fast. Finally, we give a rigorous numerical analysis of the algorithm to support that it is stable in both forward and backward sense.

8.1. proof of lemma 1

In this section, we prove some practical bounds for computed QR factorization $SA = \bar{Q}\hat{R}$. The computation process of QR factorization can be decomposed as

$$\widehat{SA} = SA + E_1, \quad |E_1| \lesssim \gamma_n |S||A|,$$
$$\widehat{SA} + E_2 = \bar{Q}\hat{R}, \quad \|E_2\|_F \lesssim \gamma_{mn} \|\widehat{SA}\|_F,$$

Thus we have

$$\|\hat{R}\| = \|\bar{Q}\hat{R}\| = \|SA + E_1 + E_2\| \le \frac{1}{1-\eta}\|A\| + 2n\gamma_n\|A\| + 2\sqrt{n}\gamma_{mn}\|A\| \lesssim \|A\|,$$
$$\|\hat{R}^{-1}\| = \sigma_{min}(\bar{Q}\hat{R}) = \sigma_{min}(SA + E_1 + E_2) \ge (1-\eta)\sigma_{min}(A) - (2\sqrt{n}\gamma_n\|A\| + 2\sqrt{n}\gamma_{mn}\|A\|) \gtrsim \frac{\|A\|}{\kappa}.$$

With similar analysis we have

$$\|A\hat{R}^{-1}\| \le \frac{1}{1-\eta}\|SA\hat{R}\| \le 2\|\bar{Q} - E_1\hat{R}^{-1} - E_2\hat{R}^{-1}\| \lesssim 1 + u\kappa n^{\frac{5}{2}},$$
$$\sigma_{min}(A\hat{R}^{-1}) \ge (1-\eta)\sigma_{min}(SA\hat{R}) \ge \frac{1}{2}\sigma_{min}(\bar{Q} - E_1\hat{R}^{-1} - E_2\hat{R}^{-1}) \gtrsim 1 - u\kappa n^{\frac{5}{2}}.$$

8.2. proof of lemma 2

In this section, we use straightforward computation to verify the relationship between $\alpha-\beta$ accuracy and backward error. In Karlson–Waldén estimate the key evaluating matrix can be expressed as

$$\left(A^\top A + \frac{\|b - A\hat{x}\|^2}{1 + \|\hat{x}\|^2} I\right)^{-1/2} = \sum_{i=1}^{n} \left(\sigma_i^2 + \frac{\|b - A\hat{x}\|^2}{1 + \|x\|^2}\right)^{-1/2} v_i v_i^\top,$$

where $A = \sum_{i=1}^{n} \sigma_i u_i v_i^\top$ is SVD decomposition of matrix A. A further calculation shows that $\widehat{\mathrm{BE}}_1(\hat{x})$ can be expressed as

$$\begin{aligned}
\widehat{\mathrm{BE}}_1(\hat{x})^2 &= \frac{1}{\sqrt{1 + \|\hat{x}\|^2}} \left\| \left(A^\top A + \frac{\|b - A\hat{x}\|^2}{1 + \|\hat{x}\|^2} I\right)^{-1/2} A^\top (b - A\hat{x}) \right\| \\
&= \left\| \sum_{i=1}^{n} \left(\sigma_i^2 + \frac{\|b - A\hat{x}\|^2}{1 + \|x\|^2}\right)^{-1/2} v_i v_i^\top A^\top A (x - \hat{x}) \right\| \\
&= \sum_{i=1}^{n} \frac{\sigma_i^4}{\left(1 + \|\hat{x}\|^2\right)\sigma_i^2 + \|b - A\hat{x}\|^2} \left(v_i^\top (x - \hat{x})\right)^2.
\end{aligned}$$

Left multiplying (5) by $v_i^\top$ yields

$$\left(v_i^\top (x - \hat{x})\right)^2 \le ((1 + \|\hat{x}\|)\sigma_i^{-1} + \|b - A\hat{x}\|\sigma_i^{-2})^2.$$

Combining two lines gives $\widehat{\mathrm{BE}}_1(\hat{x})^2 \le n$.

8.3. the equivalence between SIR and SRR

In this section, we show that sketched iterative refinement (SIR) and sketched recursive refinement (SIR) have the same form of results when they have the same linear meta-algorithm, which gives theoretical support to the statement that Recursive Refinement is just a reorganization of computation order in the Iterative Refinement procedure.

Set target linear system $Ax = b$ with $b \in range(A)$. Suppose that $ALG^{meta}(b) = Tb + q$ for some full rank matrix T, then we have

$$x_{i+1} = x_i + ALG^{meta}(b - Ax) = (I - TA)x + (Tb + q).$$

Note that the iteration is invariant when we initialize with true solution x^* and $Ax^* = b$, thus $q = 0$. A direct calculation shows that with a zero initial $x_0 = 0$, we have

$$\mathrm{SIR}_N(b) = x_N = \sum_{i=0}^{N-1} (I - TA)^i Tb.$$

For $x_i = \mathrm{SRR}_i(b)$ and $x_0 = 0$, we claim that

$$\mathrm{SRR}_N(b) = x_N = \sum_{i=0}^{2^N-1} (I - TA)^i Tb,$$

and we will prove it by induction.

It is easy to check that the statement holds for $N = 1$ where

$$\begin{aligned}\mathrm{SRR}_0(b) &= ALG^{meta}(b) = Tb,\\ \mathrm{SRR}_1(b) &= ALG^{meta}(b) + ALG^{meta}(b - A \cdot ALG^{meta}(b))\\ &= Tb + T(b - ATb)\\ &= (I - TA)Tb + Tb.\end{aligned}$$

To apply induction, suppose that the statement holds for N and we compute $\mathrm{SRR}_{N+1}(b)$ as

$$\begin{aligned}\mathrm{SRR}_{N+1}(b) &= \mathrm{SRR}_N(b) + ALG^{meta}(b - A \cdot \mathrm{SRR}_N(b))\\ &= \sum_{i=0}^{2^N-1} (I - TA)^i Tb + \sum_{i=0}^{2^N-1} (I - TA)^i T(b - A \sum_{i=0}^{2^N-1} (I - TA)^i Tb).\end{aligned}$$

Note that $I - A\sum_{i=0}^{2^N-1}(I - TA)^i T = I - AT\sum_{i=0}^{2^N-1}(I - AT)^i = (I - AT)^{2^N}$, thus

$$\begin{aligned}\mathrm{SRR}_{N+1}(b) &= \sum_{i=0}^{2^N-1} (I - TA)^i Tb + \sum_{i=0}^{2^N-1} (I - TA)^i T(b - A \sum_{i=0}^{2^N-1} (I - TA)^i Tb)\\ &= \sum_{i=0}^{2^N-1} (I - TA)^i Tb + \sum_{i=0}^{2^N-1} (I - TA)^i T(I - AT)^{2^N} b\\ &= \sum_{i=0}^{2^N-1} (I - TA)^i Tb + \sum_{i=2^N}^{2^{N+1}-1} (I - TA)^i Tb\\ &= \sum_{i=0}^{2^{N+1}-1} (I - TA)^i Tb.\end{aligned}$$

Thus the statement holds for all N.

8.4. proof of theorem 4

With a geometric series form of result given in the previous paper, one can easily examine the convergence of the iterative algorithm. Recall that for solving a well-defined linear system $Ax = b$, the solution of SIR and SRR has the form

$$x_N = \sum_{i=0}^{N-1} (I - TA)^i Tb,$$

and the true solution x^* satisfies $x^* = A^{-1}b$. Thus the error $||x_N - x^*||$ satisfies

$$\begin{aligned}||x_N - x^*|| &= ||A^{-1}b - \sum_{i=0}^{N-1} (I - TA)^i Tb||\\ &= ||(I - \sum_{i=0}^{N-1} (I - TA)^i TA)A^{-1}b||\\ &= ||(I - TA)^N A^{-1}b||\\ &\le ||(I - TA)||^N ||A^{-1}b||.\end{aligned}$$

It implies that SIR has a linear convergence with a convergence rate $||I - TA||$ and SRR has a quadratic convergence. The solver is convergent if and only if $||I - TA|| < 1$.

Then we compute the exact convergence rate for randomized solvers in solving $A^\top Ax = A^\top b$ instead of $Ax = b$, since in general cases $||b - Ax^*|| \neq 0$. For iterative refinement with sketch-and-solve method, we have $T = (A^\top S^\top SA)^{-1}$ and thus $x_N = \sum_{i=0}^{N-1}(I - (A^\top S^\top SA)^{-1}A^\top A)^i(A^\top S^\top SA)^{-1}A^\top b$.

Note that

$$\begin{aligned}||(I - (A^\top S^\top SA)^{-1}A^\top A)^i (A^\top S^\top SA)^{-1}|| &= ||A^\dagger (I - A(A^\top S^\top SA)^{-1}A^\top)^i A(A^\top S^\top SA)^{-1}||\\ &\le ||A^\dagger|| ||(I - A(A^\top S^\top SA)^{-1}A^\top)^i|| A(A^\top S^\top SA)^{-1}||\\ &\lesssim (\frac{\kappa}{||A||})^2 (\frac{1}{(1-\eta)^2} - 1)^i.\end{aligned}$$

The third inequality comes from the fact that $\|A(A^\top S^\top SA)^{-1}\| \asymp \|A^\dagger\| = \frac{\kappa}{\|A\|}$ and $\|(I - A(A^\top S^\top SA)^{-1}A^\top)\| \le \max\{\frac{1}{(1-\eta)^2} - 1, 1 - \frac{1}{(1+\eta)^2}\} = \frac{1}{(1-\eta)^2} - 1$. It implies

$$\|x_N - x^*\| \le constant \cdot (\frac{1}{(1-\eta)^2} - 1)^i.$$

Note that to guarantee the convergence of SIR, the embedding distortion η should be bounded in $(0,1)$. However, η is usually bad in some difficult least-squares problems due to numerical error and small sketch dimensions. Fortunately, the Krylov subspace method is free from the restriction of η. We then verify the convergence of k-step Krylov-based iterative refinement. Note that

$$\begin{aligned} Ay_{i+1} &= A(I - (A^\top S^\top SA)^{-1}A^\top A)y_i + A(A^\top S^\top SA)^{-1}A^\top b \\ &= (I - A(A^\top S^\top SA)^{-1}A^\top)Ay_i + A(A^\top S^\top SA)^{-1}A^\top b \\ y_0 &= x_i \\ x_{i+1} &= \text{argmin}_{x\in span\{y_1,y_2,\cdots,y_k\}} \|Ax - b\|. \end{aligned}$$

Denote $A(A^\top S^\top SA)^{-1}A^\top$ as T. Since $Ax_{i+1} \in Ax_i + \mathcal{K}_k(A(A^\top S^\top SA)^{-1}A^\top, b - Ax_i)$, $A(x_{i+1} - x_i)$ can be expressed as

$$A(x_{i+1} - x^*) = p_k(T)A(x_i - x^*),$$

where p_k is a polynomial with order no more than k.

Since $A(A^\top S^\top SA)^{-1}A^\top$ is normal matrix and can be decomposed as

$$A(A^\top S^\top SA)^{-1}A^\top = V\Lambda V^\top, \quad V^\top V = I, \Lambda = diag(\lambda_1, \lambda_2, \cdots, \lambda_n),$$

then

$$A(x_{i+1} - x^*) = Vp_k(\Lambda)V^\top A(x_i - x^*),$$

where $\|x_{i+1} - x^*\|_A$ is bounded by $|p_k(\lambda)|\|x_i - x^*\|$. We follow the practical but the worst case upper bound [6], [8] for min-max problem

$$\min_p \max_i p_k(\lambda_i)$$

by choosing $p_k(\cdot)$ as the k-order Chebyshev polynomial. It leads to

$$\|x_{i+1} - x^*\|_A \le (\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa + 1}})^k \|x_i - x^*\|_A, \quad \kappa = \frac{\lambda_{max}(T)}{\lambda_{min}(T)},$$

which further leads to

$$\|x_{i+1} - x^*\|_A \le \min\{\eta^k, \frac{1}{\eta^k}\}\|x_i - x^*\|_A$$

since $\kappa(T) = \kappa(A(A^\top S^\top SA)^{-1}A^\top) \le \frac{(1+\eta)^2}{(1-\eta)^2}$. The result indicates that Krylov-based sketching method works even if the quality of subspace embedding is bad, requiring fewer sketching dimensions, which makes the algorithm faster.

8.5. proof of theorem 5

In this section, we give a detailed analysis of the forward stability of SIRR, which also serves as a foundation for further discussion about backward stability.

We first show the converged result of SRR can be decomposed into the form

$$\text{SRR}_N(r_A) \to x^* + u\sqrt{n}\|x^*\|e_1 + u\kappa n^{\frac{3}{2}}\|Ax^*\|\hat{R}^{-1}e_2,$$

where $\|e_{1,2}\| \lesssim 1$.

Consider the expression of $\text{SRR}_k(r_A)$ in real computation according to section 8.3, we claim that

$$\hat{x}_k = \text{SRR}_k(r_A) = (A^\top A)^{-1}r_A + a_k(\hat{R}^{-\top}r_A)e_k^1 + b_k(\hat{R}^{-\top}r_A)R^{-1}e_k^2,$$

where $a_k(\hat{R}^{-\top}r_A), b_k(\hat{R}^{-\top}r_A)$ are numerical errors, which are supposed to be functions of $\|\hat{R}^{-\top}r_A\|$, and $\|e_i^j\| \lesssim 1$. Then

$$\begin{aligned} \text{SRR}_{k+1}(r_A) &= \hat{x}_k + (A^\top A)^{-1}\hat{r}_k^A + a_k(R^{-\top}\hat{r}_k^A)e_k^5 + b_k(R^{-\top}\hat{r}_k^A)R^{-1}e_k^6 + \underbrace{\sqrt{n}\|\hat{x}_{k+1}\|}_{adding\ error} \\ &= (A^\top A)^{-1}r_A + a_{k+1}(\hat{R}^{-\top}r_A)e_{k+1}^1 + b_{k+1}(\hat{R}^{-\top}r_A)R^{-1}e_{k+1}^2, \end{aligned}$$

where

$$\hat{r}_k^A = r_A - A^\top A\hat{x}_k + un^{\frac{3}{2}}\|A\|\|\hat{x}_k\|A^\top e_k^3 + un^{\frac{3}{2}}\|A\|\|A\hat{x}_k\|e_k^4.$$

The first equation is a direct computation of SRR_{k+1}. The iteration of a_k, b_k can thus be presented as

$$\begin{aligned} b_{k+1}(\hat{R}^{-\top}r_A) &\lesssim b_k(\hat{R}^{-\top}\hat{r}_k^A) + un^{\frac{3}{2}}\|A\|\|\hat{x}_k\| + un^{\frac{3}{2}}\kappa\|A\hat{x}_k\| \\ a_{k+1}(\hat{R}^{-\top}r_A) &\lesssim a_k(\hat{R}^{-\top}\hat{r}_k^A) + u\sqrt{n}(\|\hat{x} - x^*\| + \|x^*\|) \end{aligned}$$

Note that

$$\begin{aligned} R^{-\top}\hat{r}_k^A &= -a_k\|\hat{R}^{-\top}r_A\|R^{-\top}(A^\top A)e_k^1 - b_k\|\hat{R}^{-\top}r_A\|R^{-\top}(A^\top A)R^{-1}e_k^2 \\ &\quad + R^{-\top}(un^{\frac{3}{2}}\|A\|\|\hat{x}_k\|A^\top e_k^3 + un^{\frac{3}{2}}\|A\|\|A\hat{x}_k\|e_k^4), \\ \|R^{-\top}\hat{r}_k^A\| &\lesssim a_k\|A\|\|\hat{R}^{-\top}r_A\| + b_k\|\hat{R}^{-\top}r_A\| + u\kappa n^{\frac{3}{2}}\|\hat{R}^{-\top}r_A\|. \end{aligned}$$

where the last inequality comes from the fact that for $A^\top Ax_r^* = r$,

$$\frac{\|A\|}{\kappa}\|x_r^*\| \lesssim \|Ax_r^*\| \asymp \|\hat{R}^{-\top}r\|.$$

For $k = 0$, the meta-algorithm is assumed to be

$$\mathrm{ALG}^{meta}(r_A) = (A^\top A)^{-1}r_A + c\|\hat{R}^{-\top}r_A\|\hat{R}^{-1}e$$

thus $a_0 = 0$ and $b_0 = c\|\hat{R}^{-1}r_A\|$. It's a natural idea to bound $a_k(\hat{R}^{-1}r_A)$ and $b_k(\hat{R}^{-1}r_A)$ by a linear function with respect to $\|\hat{R}^{-1}r_A\|$, since we can transform terms like $\|\hat{x}\|$, $\|A\hat{x}\|$ into $\|\hat{R}^{-1}r_A\|$ multiplied by some constant. First we convert terms $\|\hat{x}\|$ into $\|x^*\| + \|x^* - \hat{x}\|$ and convert terms $\|x^*\|$ and $\|Ax^*\|$ into $\|\hat{R}^{-\top}r_A\|$, and then compute $\hat{x}_k - x^*$ by leveraging the fact that $\hat{x}_k - x^* = a_k(\hat{R}^{-\top}r_A)e_k^1 + b_k(\hat{R}^{-\top}r_A)\hat{R}^{-1}e_k^2$. After assuming $a_k(\hat{R}^{-\top}r_A) \lesssim \alpha_k\|\hat{R}^{-\top}r_A\|$ and $b_k(\hat{R}^{-\top}r_A) \lesssim \beta_k\|\hat{R}^{-\top}r_A\|$, one gets the iteration of α_k, β_k

$$\begin{aligned} \beta_{k+1}\|\hat{R}^{-\top}r_A\| &\lesssim \beta_k(\alpha_k\|A\|\|\hat{R}^{-\top}r_A\| + \beta_k\|\hat{R}^{-\top}r_A\| + u\kappa n^{\frac{3}{2}}\|\hat{R}^{-\top}r_A\|) + un^{\frac{3}{2}}\|A\|\|\hat{x}_k\| + un^{\frac{3}{2}}\kappa\|A\hat{x}_k\| \\ &\lesssim \alpha_k\beta_k\|A\|\|\hat{R}^{-\top}r_A\| + \beta_k^2\|\hat{R}^{-\top}r_A\| + u\kappa n^{\frac{3}{2}}\beta_k\|\hat{R}^{-\top}r_A\| + un^{\frac{3}{2}}\kappa\|\hat{R}^{-\top}r_A\|, \\ a_{k+1}\|\hat{R}^{-\top}r_A\| &\lesssim \alpha_k(\alpha_k\|A\|\|\hat{R}^{-\top}r_A\| + \beta_k\|\hat{R}^{-\top}r_A\| + u\kappa n^{\frac{3}{2}}\|\hat{R}^{-\top}r_A\|) + u\sqrt{n}(\|\hat{x} - x^*\| + \|x^*\|) \\ &\lesssim \alpha_k^2\|A\|\|\hat{R}^{-\top}r_A\| + \alpha_k\beta_k\|\hat{R}^{-\top}r_A\| + u\kappa n^{\frac{3}{2}}\alpha_k\|\hat{R}^{-\top}r_A\|, \end{aligned}$$

which leads to

$$\|A\|\alpha_{k+1} + \beta_{k+1} \lesssim (\|A\|\alpha_k + \beta_k)^2 + un^{\frac{3}{2}}\kappa(\|A\|\alpha_k + \beta_k) + un^{\frac{3}{2}}\kappa.$$

Since $\|A\|\alpha_0 + \beta_0 = \kappa n^{\frac{3}{2}}u < 1$, $\|A\|\alpha_k + \beta_k$ converges to $u\kappa n^{\frac{3}{2}}$ and thus α_k and β_k converge to $u\kappa n^{\frac{3}{2}}$. Combined with (**??**) one gets

$$\begin{aligned} \alpha_k &\lesssim (u\kappa n^{\frac{3}{2}})^2\|\hat{R}^{-\top}r_A\| + \sqrt{n}u\|x^*\| \lesssim \sqrt{n}u\|x^*\|, \\ \beta_k &\lesssim un^{\frac{3}{2}}\|A\|\|x^*\| + u\kappa n^{\frac{3}{2}}\|Ax^*\| \lesssim u\kappa n^{\frac{3}{2}}\|Ax^*\|. \end{aligned}$$

Thus we have

$$\mathrm{SRR}(r_A) \to x^* + u\sqrt{n}\|x^*\|e_1 + u\kappa n^{\frac{3}{2}}\|Ax^*\|\hat{R}^{-1}e_2.$$

Now we iterate SRR with SIR to prove that SIRR is strongly forward stable. The real computation of SRR can be expressed as

$$\begin{aligned} \hat{r}_i &= \underbrace{b - A\hat{x}_i}_{r_i} + u\sqrt{n}\|r_i\|e_{i,1} + un^{\frac{3}{2}}\|A\|\|\hat{x}_i\|e_{i,2}, \\ \hat{r}_i^A &= A^\top\hat{r}_i + un^{\frac{3}{2}}\|A\|\|\hat{r}_i\|e_{i,3}, \\ \hat{x}_{i+1} &= \hat{x}_i + (A^\top A)^{-1}(\hat{r}_i^A) + u\sqrt{n}\|x_r^*\|e_{i,4} + u\kappa n^{\frac{3}{2}}\|Ax_r^*\|\hat{R}^{-1}e_{i,5}, \\ &= (A^\top A)^{-1}A^\top b + (A^\top A)^{-1}(un^{\frac{3}{2}}\|A\|\|\hat{r}_i\|e_{i,3} + u\sqrt{n}\|r_i\|A^\top e_{i,1} + un^{\frac{3}{2}}\|A\|\|\hat{x}_i\|A^\top e_{i,2}), \\ &\quad + u\sqrt{n}\|x_r^*\|e_{i,4} + u\kappa n^{\frac{3}{2}}\|Ax_r^*\|\hat{R}^{-1}e_{i,5}, \\ x_r^* &= (A^\top A)^{-1}\hat{r}_i^A, \\ &= (A^\top A)^{-1}(A^\top r_i + un^{\frac{3}{2}}\|A\|\|\hat{r}_i\|e_{i,3} + u\sqrt{n}\|r_i\|A^\top e_{i,1} + un^{\frac{3}{2}}\|A\|\|\hat{x}_i\|A^\top e_{i,2}), \end{aligned}$$

Denote $\|\hat{x}_i - x^*\|$ as err_i, $\|A\hat{x}_i - Ax^*\|$ as err_i^r. With decomposition $\|\hat{x}_i\| \le \|x^*\| + err_i$, $\|A\hat{x}_i\| \le \|Ax^*\| + err_i^r$, one gets

$$\begin{aligned}
\|x_r^*\| &\le \frac{\kappa}{\|A\|} err_i^r + \frac{u\kappa^2 n^{\frac{3}{2}}}{\|A\|}\|r^*\| + u\kappa n^{\frac{3}{2}} \cdot err_i + u\kappa n^{\frac{3}{2}}\|x^*\|, \\
\|Ax^*\| &\le err_i^r + u\kappa n^{\frac{3}{2}}\|r^*\| + un^{\frac{3}{2}}\|A\| err_i + un^{\frac{3}{2}}\|A\|\|x^*\|.
\end{aligned}$$

We can then present the iteration of err_i and err_i^r as

$$\begin{aligned}
err_{i+1} \le\ & \frac{u\kappa^2}{\|A\|} n^{\frac{3}{2}}(err_i^r + \|r^*\|) + u\kappa n^{\frac{3}{2}}(err_i + \|x^*\|) \\
& + u\sqrt{n}(\frac{\kappa}{\|A\|} err_i^r + \frac{u\kappa^2}{\|A\|}\|r^*\| + u\kappa \cdot err_i + u\kappa\|x^*\|) \\
& + \frac{u\kappa^2 n^{\frac{3}{2}}}{\|A\|}(err_i^r + u\kappa\|r^*\| + u\|A\| err_i + u\|A\|\|x^*\|) \\
& \le u\kappa n^{\frac{3}{2}} \cdot err_i + \frac{u\kappa^2 n^{\frac{3}{2}}}{\|A\|} err_i^r + \frac{u\kappa^2 n^{\frac{3}{2}}}{\|A\|}\|r^*\| + u\kappa n^{\frac{3}{2}}\|x^*\| \\
err_{i+1}^r \le\ & u\kappa n^{\frac{3}{2}}(\|r^*\| + err_i^r) + un^{\frac{3}{2}}\|A\|(err_i + \|x^*\|) \\
& + u\sqrt{n}\|A\|(\frac{\kappa}{\|A\|} err_i^r + \frac{u\kappa^2}{\|A\|}\|r^*\| + u\kappa \cdot err_i + u\kappa\|x^*\|) \\
& + u\kappa n^{\frac{3}{2}}(err_i^r + u\kappa\|r^*\| + u\|A\| err_i + u\|A\|\|x^*\|) \\
& = un^{\frac{3}{2}}\|A\| err_i + u\kappa n^{\frac{3}{2}} \cdot err_i^r + u\kappa n^{\frac{3}{2}}\|r^*\| + un^{\frac{3}{2}}\|A\|\|x^*\|
\end{aligned}$$

The iteration can be transformed into:

$$\begin{pmatrix} err_{i+1} \\ err_{i+1}^r \\ 1 \end{pmatrix} \lesssim n^{\frac{3}{2}} \begin{pmatrix} u\kappa & \frac{u\kappa^2}{\|A\|} & \frac{u\kappa^2}{\|A\|}\|r^*\| + u\kappa\|x^*\| \\ u\|A\| & u\kappa & u\|A\|\|x^*\| + u\kappa\|r^*\| \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} err_i \\ err_i^r \\ 1 \end{pmatrix}$$

Since the transition matrix has the largest eigenvalue 1, the vector series $(err_{i+1}, err_{i+1}^r, 1)^\top$ will converge to the eigenvector of 1, which leads to

$$\lim_{i\to\infty} \begin{pmatrix} err_i \\ err_i^r \\ 1 \end{pmatrix} \lesssim n^{\frac{3}{2}} \begin{pmatrix} u\kappa\|x^*\| + \frac{u\kappa^2}{\|A\|}\|r^*\| \\ u\kappa\|r^*\| + u\|A\|\|x^*\| \\ 1 \end{pmatrix}$$

Thus the result of SIRR is forward stable.

8.6. proof of theorem 6

In this section, we propose the requirements for single step meta-solver to ensure that the SIR algorithm based on this meta-algorithm is backward stable. Suppose that in ith iteration the solution x_i has a_i, b_i-accuracy, which can be expressed as

$$x_i = (A^\top A)^{-1}A^\top b + a_i\hat{R}^{-1}e_i^1 + b_i(A^\top A)^{-1}e_i^2$$

for some unit random vector e_i^1 and e_i^2. We aim to get the iteration of a_i, b_i. Recall that $\tilde{\kappa}^{-1} = \max\{u\kappa, \frac{1}{\kappa n^{\frac{3}{2}}}\}$. Following the computation of SIR we have

$$\begin{aligned}
r_i &= b - Ax_i + f_i, \quad \text{(computed residual in each step)} \\
x_{i+1} &= x_i + (A^\top A)^{-1}(A^\top r_i) \\
&\quad + n^{\frac{3}{2}}(\tilde{\kappa}^{-1}\|Ax_r^*\| + u\kappa\tilde{\kappa}^{-1}\|r_i\|)\hat{R}^{-1}e_1 \\
&\quad + n^{\frac{3}{2}}(u\|Ax_r^*\| + u\|r_i\|)(A^\top A)^{-1}e_2 \quad \text{(assumption of single step meta-solver)} \\
&= \underbrace{(A^\top A)^{-1}A^\top b}_{x^*} + a_{i+1}\hat{R}^{-1}e_{i+1}^1 + b_{i+1}(A^\top A)^{-1}e_{i+1}^2,
\end{aligned}$$

Here

$$\begin{aligned}
f_i &\lesssim u(n^{\frac{3}{2}}\|x_i\| + \sqrt{n}\|b - Ax_i\|)e_{f_i}, \quad \text{(error in computed residual)} \\
x_r^* &= (A^\top A)^{-1}(A^\top r_i) \\
&= -a_i(A^\top A)^{-1}A^\top(A\hat{R}^{-1})e_i^1 - b_i(A^\top A)^{-1}e_i^2 + (A^\top A)^{-1}A^\top f_i,
\end{aligned}$$

$$b - Ax_i = \underbrace{b - A(A^\top A)^{-1}A^\top b}_{r^*} - a_i A\hat{R}^{-1}e_i^1 - b_i A(A^\top A)^{-1}e_i^2,$$

We omit the adding error when computing $x_{i+1} = x_i + d_i$ where d_i is the refinement term, since it is already machine-precision. Then we have

$$\begin{aligned} a_{i+1}\hat{R}^{-1}e_{i+1}^1 &= n^{\frac{3}{2}}(\tilde{\kappa}^{-1}\|Ax_r^*\| + u\kappa\tilde{\kappa}^{-1}\|r_i\|)\hat{R}^{-1}e_1 \\ &+ \underbrace{u(n^{\frac{3}{2}}\|x_i\| + \sqrt{n}\|b - Ax_i\|)\hat{R}^{-1}(\hat{R}(A^\top A)^{-1}A^\top e_{f_i}}_{(A^\top A)^{-1}A^\top f_i}), \\ b_{i+1}(A^\top A)^{-1}e_{i+1}^2 &= n^{\frac{3}{2}}(u\|Ax_r^*\| + u\|r_i\|)(A^\top A)^{-1}e_2, \end{aligned}$$

which yields

$$\begin{aligned} a_{i+1} &\lesssim n^{\frac{3}{2}}(\tilde{\kappa}^{-1}\|Ax_r^*\| + u\kappa\tilde{\kappa}^{-1}\|r_i\|) + \|f_i\| \\ &\lesssim n^{\frac{3}{2}}\tilde{\kappa}^{-1}(a_i + \kappa b_i + \|f_i\|) \\ &+ n^{\frac{3}{2}}u\kappa\tilde{\kappa}^{-1}(\|r^*\| + a_i + \kappa b_i + \|f_i\|) \\ &+ \underbrace{u(n^{\frac{3}{2}}\|x_i\| + \sqrt{n}(\|r^*\| + a_i + \kappa b_i))}_{\|f_i\|} \\ &\asymp n^{\frac{3}{2}}\tilde{\kappa}^{-1}(a_i + \kappa b_i) + un^{\frac{3}{2}}\|x^*\| + u\kappa n^{\frac{3}{2}}\tilde{\kappa}^{-1}\|r^*\|, \\ b_{i+1} &\lesssim un^{\frac{3}{2}}\|Ax_r^*\| + un^{\frac{3}{2}}\|r_i\| \\ &\lesssim un^{\frac{3}{2}}(a_i + \kappa b_i + \|f_i\|) \\ &+ un^{\frac{3}{2}}(\|r^*\| + a_i + \kappa b_i + \|f_i\|) \\ &\asymp un^{\frac{3}{2}}(a_i + \kappa b_i) + u^2n^3\|x^*\| + un^{\frac{3}{2}}\|r^*\|. \end{aligned}$$

The iteration of a_i, b_i can be written in the form

$$\begin{pmatrix} a_{i+1} \\ b_{i+1} \\ 1 \end{pmatrix} \lesssim \begin{pmatrix} n^{\frac{3}{2}}\tilde{\kappa}^{-1} & n^{\frac{3}{2}}\kappa\tilde{\kappa}^{-1} & un^{\frac{3}{2}}\kappa\tilde{\kappa}^{-1}\|r^*\| + un^{\frac{3}{2}}\|x^*\| \\ un^{\frac{3}{2}} & un^{\frac{3}{2}}\kappa & un^{\frac{3}{2}}\|r^*\| + u^2n^3\|x^*\| \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_i \\ b_i \\ 1 \end{pmatrix}$$

Since the transition matrix has the largest eigenvalue 1, the vector series $(a_i, b_i, 1)^\top$ will converge to the eigenvector of 1, which leads to

$$\lim_{i\to\infty} \begin{pmatrix} a_i \\ b_i \\ 1 \end{pmatrix} \lesssim \begin{pmatrix} un^3\kappa\tilde{\kappa}^{-1}\|r^*\| + un^{\frac{3}{2}}\|x^*\| \\ un^{\frac{3}{2}}\|r^*\| + u^2n^3\|x^*\| \\ 1 \end{pmatrix},$$

where we use the fact that $\tilde{\kappa}^{-1}n^{\frac{3}{2}} < 1$.

With $u\kappa\tilde{\kappa}^{-1} = u\kappa\max\{u\kappa, \kappa^{-1}\} < u^2\kappa^2 + u$, the result $\lim_{i\to\infty} x_i$ is backward stable.

8.7. proof of theorem 7

In this section, we show that SRR generates a good single step meta-solver for SIR, in other words, SIRR is backward stable. In SRR, the meta-algorithm $\mathrm{SRR}_0(\cdot)$ solves a full rank linear system $(A^\top A)x = r_A$, and from iteration process we find the error of solution only depends on either $\|R^{-\top}r_A\|$, $\|x^*\|$ or $\|Ax^*\|$, where $x^* = (A^\top A)^{-1}r_A$ and $\|R^{top}r_A\| \asymp \|Ax^*\|$. Thus we can assume that

$$\mathrm{SRR}_i(r_A) = (A^\top A)^{-1}r_A + (a_i^1\|x^*\| + a_i^2\|Ax^*\|)R^{-1}e_1^i + (b_i^1\|x^*\| + b_i^2\|Ax^*\|)(A^\top A)^{-1}e_2^i$$

, one can get the iteration of a_i^j, b_i^j w.r.t i.

The iteration of SRR_i then can be written as

$$\begin{aligned} \mathrm{SRR}_{i+1}(r_A) &= \mathrm{SRR}_i(r_A) + \mathrm{SRR}_i(\underbrace{r_A - A^\top Ax_i + f_i}_{\hat{r}_i}) \\ &= x_i + (A^\top A)^{-1}(r_A - A^\top Ax_i) \\ &+ (A^\top A)^{-1}(\underbrace{(u\sqrt{n}\|r_A - A^\top Ax_i\| + un^{\frac{3}{2}}\|Ax_i\|)e_{f_1} + un^{\frac{3}{2}}\|x_i\|A^\top e_{f_2}}_{f_i}) \end{aligned}$$

$$
\begin{aligned}
&+ (a_i^1\|x_r^*\| + a_i^2\|Ax_r^*\|)\hat{R}^{-1}e_1^i + (b_i^1\|x_r^*\| + b_i^2\|Ax_r^*\|)(A^\top A)^{-1}e_2^i \\
&= (A^\top A)^{-1}r_A + (a_{i+1}^1\|x_r^*\| + a_{i+1}^2\|Ax_r^*\|)\hat{R}^{-1}e_1^{i+1} \\
&+ (b_{i+1}^1\|x_r^*\| + b_{i+1}^2\|Ax_r^*\|)(A^\top A)^{-1}e_2^{i+1},
\end{aligned}
$$

where

$$
\begin{aligned}
x_r^* &= (A^\top A)^{-1}\hat{r}_i \\
&= (A^\top A)^{-1}(r_A - A^\top A x_i + f_i) \\
&= -(a_i^1\|x^*\| + a_i^2\|Ax^*\|)\hat{R}^{-1}e_1^i - (b_i^1\|x^*\| + b_i^2\|Ax^*\|)(A^\top A)^{-1}e_2^i + (A^\top A)^{-1}f_i, \\
f_i &= u\sqrt{n}\|r_A - A^\top A x_i\| + un^{\frac{3}{2}}\|Ax_i\|)e_{f_1} + un^{\frac{3}{2}}\|x_i\|A^\top e_{f_2}.
\end{aligned}
$$

Denote $1 + u\kappa n^{\frac{3}{2}}$ as $\hat{1}$ for convenience, then the expansion of $\|x_r^*\|$ and $\|Ax_r^*\|$ yields

$$
\begin{aligned}
\|x_r^*\| &\lesssim (\kappa a_i^1 + \kappa^2 b_i^1)\|x^*\| + (\kappa a_i^2 + \kappa^2 b_i^2)\|Ax^*\| \\
&+ \underbrace{u\kappa^2\sqrt{n}((a_i^1 + b_i^1)\|x^*\| + (a_i^2 + b_i^2)\|Ax^*\|)}_{f_i\ term1} \\
&+ \underbrace{u\kappa^2 n^{\frac{3}{2}}(\|Ax^*\| + (a_i^1 + \kappa b_i^1)\|x^*\| + (a_i^2 + \kappa b_i^2)\|Ax^*\|)}_{f_i\ term2} \\
&+ \underbrace{u\kappa n^{\frac{3}{2}}(\|x^*\| + \kappa(a_i^1 + \kappa b_i^1)\|x^*\| + \kappa(a_i^2 + \kappa b_i^2)\|Ax^*\|)}_{f_i\ term3} \\
&\asymp (\hat{1}\kappa a_i^1 + \hat{1}\kappa^2 b_i^1 + u\kappa n^{\frac{3}{2}})\|x^*\| + (\hat{1}\kappa a_i^2 + \hat{1}\kappa^2 b_i^2 + u\kappa^2 n^{\frac{3}{2}})\|Ax^*\|,
\end{aligned}
$$

$$
\begin{aligned}
\|Ax_r^*\| &\lesssim (a_i^1 + \kappa b_i^1)\|x^*\| + (a_i^2 + \kappa b_i^2)\|Ax^*\| \\
&+ \underbrace{u\kappa\sqrt{n}((a_i^1 + b_i^1)\|x^*\| + (a_i^2 + b_i^2)\|Ax^*\|)}_{f_i\ term1} \\
&+ \underbrace{u\kappa n^{\frac{3}{2}}(\|Ax^*\| + (a_i^1 + \kappa b_i^1)\|x^*\| + (a_i^2 + \kappa b_i^2)\|Ax^*\|)}_{f_i\ term2} \\
&+ \underbrace{un^{\frac{3}{2}}(\|x^*\| + \kappa(a_i^1 + \kappa b_i^1)\|x^*\| + \kappa(a_i^2 + \kappa b_i^2)\|Ax^*\|)}_{f_i\ term3} \\
&\asymp (\hat{1}a_i^1 + \hat{1}\kappa b_i^1 + un^{\frac{3}{2}})\|x^*\| + (\hat{1}a_i^2 + \hat{1}\kappa b_i^2 + u\kappa n^{\frac{3}{2}})\|Ax^*\|,
\end{aligned}
$$

With assumption $u\kappa n^{\frac{3}{2}} < 1$, $\hat{1} \lesssim 1$, the iteration of a_i^j, b_i^j has the form

$$
\begin{aligned}
\begin{pmatrix} a_{i+1}^1 \\ a_{i+1}^2 \end{pmatrix} &\lesssim \begin{pmatrix} \kappa(a_i^1 + \kappa b_i^1 + un^{\frac{3}{2}}) & (a_i^1 + \kappa b_i^1 + un^{\frac{3}{2}}) \\ \kappa(a_i^2 + \kappa b_i^2 + u\kappa n^{\frac{3}{2}}) & (a_i^2 + \kappa b_i^2 + u\kappa n^{\frac{3}{2}}) \end{pmatrix} \begin{pmatrix} a_i^1 \\ a_i^2 \end{pmatrix} + n^{\frac{3}{2}} \begin{pmatrix} u + u\kappa a_i^1 + u\kappa^2 b_i^1 \\ u\kappa a_i^2 + u\kappa^2 b_i^2 \end{pmatrix}, \\
\begin{pmatrix} b_{i+1}^1 \\ b_{i+1}^2 \end{pmatrix} &\lesssim \begin{pmatrix} \kappa(a_i^1 + \kappa b_i^1 + un^{\frac{3}{2}}) & (a_i^1 + \kappa b_i^1 + un^{\frac{3}{2}}) \\ \kappa(a_i^2 + \kappa b_i^2 + u\kappa n^{\frac{3}{2}}) & (a_i^2 + \kappa b_i^2 + u\kappa n^{\frac{3}{2}}) \end{pmatrix} \begin{pmatrix} b_i^1 \\ b_i^2 \end{pmatrix} + n^{\frac{3}{2}} \begin{pmatrix} u(a_i^1 + \kappa b_i^1) \\ u(a_i^2 + \kappa b_i^2 + 1) \end{pmatrix}.
\end{aligned}
$$

Let $c_i^j = a_i^j + \kappa b_i^j$, then

$$
\begin{aligned}
c_{i+1}^1 &\lesssim \kappa(c_i^1)^2 + c_i^1 c_i^2 + u\kappa n^{\frac{3}{2}} c_i^1 + un^{\frac{3}{2}} c_i^2 + un^{\frac{3}{2}}, \\
c_{i+1}^2 &\lesssim \kappa c_i^1 c_i^2 + (c_i^2)^2 + u\kappa^2 n^{\frac{3}{2}} c_i^1 + u\kappa n^{\frac{3}{2}} c_i^2 + u\kappa n^{\frac{3}{2}}, \\
\kappa c_{i+1}^1 + c_{i+1}^2 &\lesssim (\kappa c_i^1 + c_i^2)^2 + u\kappa n^{\frac{3}{2}}(\kappa c_i^1 + c_i^2) + u\kappa n^{\frac{3}{2}}.
\end{aligned}
$$

For general cases, consider the initialization of

$$\kappa a_0^1 + a_0^2 \asymp c, \kappa(\kappa b_0^1 + b_0^2) \asymp c, c_0 \asymp \kappa c_0^1 + c_0^2 \asymp \kappa(a_0^1 + \kappa b_0^1) + a_0^2 + \kappa b_0^2 \asymp c.$$

Then $\kappa c_0^1 + c_0^2 = c \gtrsim u\kappa n^{\frac{3}{2}}$, thus

$$\kappa c_i^1 + c_i^2 \lesssim max\{(\underbrace{\kappa c_0^1 + c_0^2}_{c})^{2^i}, u\kappa n^{\frac{3}{2}}\}.$$

Similar result can be derived for c_i^1, c_i^2 where

$$c_{i+1}^1 \lesssim (\kappa c_i^1 + c_i^2)c_i^1 + un^{\frac{3}{2}}(\kappa c_i^1 + c_i^2) + un^{\frac{3}{2}} \lesssim max\{(\kappa c_i^1 + c_i^2)c_i^1, un^{\frac{3}{2}}\},$$
$$c_{i+1}^2 \lesssim (\kappa c_i^1 + c_i^2)c_i^2 + u\kappa n^{\frac{3}{2}}(\kappa c_i^1 + c_i^2) + u\kappa n^{\frac{3}{2}} \lesssim max\{(\kappa c_i^1 + c_i^2)c_i^2, u\kappa n^{\frac{3}{2}}\}$$

which leads to

$$c_i^1 \lesssim max\{c_0^1(c_0)^{2^i}, un^{\frac{3}{2}}\},$$
$$c_i^2 \lesssim max\{c_0^2(c_0)^{2^i}, u\kappa n^{\frac{3}{2}}\}.$$

For a_i^j, we can first calculate the iteration of $\kappa a_i^1 + a_i^2$by

$$\begin{aligned}
\kappa a_{i+1}^1 + a_{i+1}^2 &\lesssim (\underbrace{\kappa c_i^1 + c_i^2}_{\lesssim max\{(\kappa c_0^1 + c_0^2)(c)^{2^i}, u\kappa n^{\frac{3}{2}}\}})(\kappa a_i^1 + a_i^2) + u\kappa n^{\frac{3}{2}}(\kappa a_i^1 + a_i^2) \\
&+ u\kappa n^{\frac{3}{2}} + u\kappa n^{\frac{3}{2}}(\kappa c_i^1 + c_i^2) \quad \text{(from transition matrix)} \\
&\lesssim (\kappa c_i^1 + c_i^2)(\kappa a_i^1 + a_i^2) + u\kappa n^{\frac{3}{2}} \quad (\sup_i(\kappa c_i^1 + c_i^2) > u\kappa n^{\frac{3}{2}}, \sup_i \kappa a_i^1 + a_i^2 > u\kappa n^{\frac{3}{2}}) \\
&\lesssim max\{(c)^{2^{i+1}}(\kappa a_0^1 + a_0^2), u\kappa n^{\frac{3}{2}}\},
\end{aligned}$$

thus

$$\begin{aligned}
a_{i+1}^1 &\lesssim (\kappa a_i^1 + a_i^2)(c_i^1 + un^{\frac{3}{2}}) + un^{\frac{3}{2}} + u\kappa n^{\frac{3}{2}} c_i^1 \\
&\lesssim max\{(c)^{2^{i+1}}(\kappa a_0^1 + a_0^2)c_0^1, un^{\frac{3}{2}}\}.
\end{aligned}$$

Similarly

$$a_i^2 \lesssim max\{(c)^{2^i}(\kappa a_0^1 + a_0^2)c_0^2, u^2\kappa^2 n^3\},$$
$$b_i^1 \lesssim max\{(c)^{2^i}(\kappa b_0^1 + b_0^2)c_0^1, u^2 n^3\},$$
$$b_i^2 \lesssim max\{(c)^{2^i}(\kappa b_0^1 + b_0^2)c_0^2, un^{\frac{3}{2}}\},$$

which leads to the bound of $\mathrm{SRR}_\infty(r_A)$:

$$\mathrm{SRR}_\infty(r_A) = \underbrace{(A^\top A)^{-1} r_A}_{x^*} + \hat{a}\hat{R}^{-1}e_1 + \hat{b}(A^\top A)^{-1}e_2$$

where

$$\hat{a} = un^{\frac{3}{2}}\|x^*\| + u^2\kappa^2 n^3\|Ax^*\|, \hat{b} = u^2 n^3\|x^*\| + un^{\frac{3}{2}}\|Ax^*\|, \quad \|e_{1,2}\| \lesssim 1.$$

In iteration algorithm, we need to compute $A^\top b$ as r_A with $error(r_A) = un^{\frac{3}{2}}\|b\|e$, so $r_A = A^\top b + un^{\frac{3}{2}}\|b\|e$. Then SRR_∞ becomes

$$\mathrm{SRR}_\infty(b) = (A^\top A)^{-1}A^\top b + a\hat{R}^{-1}e_1 + b(A^\top A)^{-1}e_2,$$

where

$$\begin{aligned}
a &\asymp un^{\frac{3}{2}}\|\hat{x}^*\| + u^2\kappa^2 n^3\|A\hat{x}^*\| \\
&= un^{\frac{3}{2}}\|x^* + un^{\frac{3}{2}}\|b\|(A^\top A)^{-1}e\| + u^2\kappa^2 n^3\|Ax^* + un^{\frac{3}{2}}\|b\|A(A^\top A)^{-1}e\| \\
&= un^{\frac{3}{2}}\|x^*\| + u^2\kappa^2 n^3\|Ax^*\| + u^2\kappa^2 n^3\|b\|, \\
b &\asymp u^2 n^3\|\hat{x}^*\| + un^{\frac{3}{2}}\|A\hat{x}^*\| + un^{\frac{3}{2}}\|b\|
\end{aligned}$$

$$= u^2n^3\|x^*\| + un^{\frac{3}{2}}\|Ax^*\| + un^{\frac{3}{2}}\|b\|.$$

However, for practical use, we can stop the iteration as soon as the algorithm achieves the accuracy needed in theorem 6.For SRR_N with Two-step Krylov-based meta-algorithm, $a_0^1 = b_0^1 = 0, a_0^2 \asymp \kappa b_0^2 \asymp c$, the iteration only refines a_i^2 and b_i^2. With $(\kappa a_0^1 + a_0^2) \asymp 1, (\kappa b_0^1 + b_0^2)\kappa \asymp 1$, the steps we actually need in practice is $N = \max\{log_2(\frac{log(\tilde{\kappa}^{-1}n^{\frac{3}{2}})}{log(c)}), log_2(\frac{log(u\kappa n^{\frac{3}{2}})}{log(c)})\} = log_2(\frac{log(\tilde{\kappa}^{-1}n^{\frac{3}{2}})}{log(c)})$, with

$$\mathrm{SRR}_N(b) = (A^\top A)^{-1}A^\top b + n^{\frac{3}{2}}(u\|x^*\| + \tilde{\kappa}^{-1}\|Ax^*\|)\hat{R}^{-1}e_1 + (u^2n^3\|x^*\| + un^{\frac{3}{2}}\|Ax^*\|)(A^\top A)^{-1}e_2.$$

A similar discussion further leads to the result of considering the first multiplication $r_A = A^\top b$

$$\begin{aligned}\mathrm{SRR}_N(b) &= (A^\top A)^{-1}A^\top b\\ &\quad + n^{\frac{3}{2}}(\tilde{\kappa}^{-1}\|Ax^*\| + u\|x^*\| + u\kappa\tilde{\kappa}^{-1}\|b\|)\hat{R}^{-1}e_1\\ &\quad + n^{\frac{3}{2}}(u\|Ax^*\| + u^2n^{\frac{3}{2}}\|x^*\| + u\|b\|)(A^\top A)^{-1}e_2\end{aligned}$$

8.8. Proof of Lemma 8

In this section, we verify that the Krylov-based meta solver satisfies the condition of theorem 7, which finally proves that a k-step Krylov-based SIRR solver is backward stable. In Krylov subspace method, with $y_0, y_1, \cdots, y_k$ given by using iterative sketching, we solve the least squares problem in the space spanned by $\{y_i\}_{i=1}^k$

$$\mathrm{argmin}_{x\in span\{y_0,y_1,\cdots,y_k\}}\|r_A - (A^\top A)x\|.$$

Let $Y := [y_0, y_1, \cdots, y_k]$, then the solution is $x = Y(A^\top AY)^{-1}r_A$.

Consider the numerical process of computing $Y(A^\top AY)^{-1}r_A$, which is

$$\begin{aligned}\widehat{AY} &= AY + E_1, \quad \|E_1\| \lesssim un^{\frac{3}{2}}\|A\|\|Y\|,\\ (A^\top\widehat{AY} + E_2)\hat{a} &= r_A + h_1, \quad \|E_2\| \le un^{\frac{3}{2}}\sqrt{k}\|A\|\|AY\| + uk^{\frac{5}{2}}\|A^\top\widehat{AY}\|, \|h_1\| \le uk^2\|r_A\|,\\ \hat{x} &= Y\hat{a} + h_2, \quad \|h_2\| \lesssim uk^{\frac{3}{2}}\|Y\|\|\hat{a}\|.\end{aligned}$$

Note that y_i can be expressed as $y_i = (A^\top A)^{-1}r_A + c\|\hat{R}^{-\top}r_A\|\hat{R}^{-1}e$ with $c \asymp 1$ and $\|e\| \lesssim 1$, and we can assume $\|\hat{a}\| \asymp 1$ since y_i are good approximation of $(A^\top A)^{-1}r_A$, then

$$\begin{aligned}\hat{x} &= Y\hat{a} + h_2\\ &= (A^\top A)^{-1}(r_A + h_1 - (A^\top E_1 + E_2)\hat{a}) + h_2\\ &= (A^\top A)^{-1}r_A + (A^\top A)^{-1}(h_1 - E_2\hat{a}) + \hat{R}^{-1}(\hat{R}h_2 + \hat{R}(A^\top A)^{-1}A^\top E_1\hat{a}).\end{aligned}$$

With $\|r_A\| = \|A^\top Ax^*\| \le \|Ax^*\| \asymp \|\hat{R}^{-\top}r_A\|$, $\|Y\| \le \sqrt{k}\max_i\|y_i\| \lesssim \sqrt{k}(\|x^*\| + c\kappa\|\hat{R}^{-\top}r_A\|) \lesssim \sqrt{k}\kappa\|Ax^*\|$ and $\|AY\| \le \sqrt{k}(\|Ax^*\| + c\|\hat{R}^{-\top}r_A\|)$ we have following bounds

$$\begin{aligned}\|h_1\| &\lesssim uk^2\|Ax^*\|,\\ \|E_2\hat{a}\| &\le un^{\frac{3}{2}}k\|Ax^*\|,\\ \|h_2\| &\lesssim u\kappa k^2\|Ax^*\|,\\ \|\hat{R}(A^\top A)^{-1}A^\top E_1\hat{a}\| &\lesssim \|E_1\| \lesssim un^{\frac{3}{2}}\kappa\sqrt{k}\|Ax^*\|.\end{aligned}$$

Since k is small, the result has the form

$$x = (A^\top A)^{-1}r_A + u\kappa n^{\frac{3}{2}}\|Ax^*\|\hat{R}^{-1}e_1 + un^{\frac{3}{2}}\|Ax^*\|(A^\top A)^{-1}e_2, \quad \|e_{1,2}\| \lesssim 1.$$

The result consequently satisfies the condition of theorem 7 as $u\kappa n^{\frac{3}{2}} \lesssim c$, thus the k-step Krylov-based SIRR solver is backward stable.