# Estimating the Trace of the Matrix Inverse by Interpolating from the Diagonal of an Approximate Inverse

Lingfei Wu[a,*], Andreas Stathopoulos[a,*], Jesse Laeuchli[a], Vassilis Kalantzis[b], Efstratios Gallopoulos[c]

[a]*Department of Computer Science, College of William and Mary, Williamsburg, VA 23187, United States*
[b]*Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, United States*
[c]*Department of Computer Engineering and Informatics, University of Patras, Patras, Greece*

## Abstract

Determining the trace of a matrix that is implicitly available through a function is a computationally challenging task that arises in a number of applications. For the common function of the inverse of a large, sparse matrix, the standard approach is based on a Monte Carlo method which converges slowly. We present a different approach by exploiting the pattern correlation between the diagonal of the inverse of the matrix and the diagonal of some approximate inverse that can be computed inexpensively. We leverage various sampling and fitting techniques to fit the diagonal of the approximation to the diagonal of the inverse. Based on a dynamic evaluation of the variance, the proposed method can be used as a variance reduction method for Monte Carlo in some cases. Furthermore, the presented method may serve as a standalone kernel for providing a fast trace estimate with a small number of samples. An extensive set of experiments with various technique combinations demonstrates the effectiveness of our method in some real applications.

*Keywords:* Matrix trace, Monte Carlo method, variance reduction, preconditioner, fitting, interpolation

## 1. Introduction

Computing the trace of an explicit matrix $A$ is a straightforward operation. However, for numerous applications we need to compute the trace of an implicit matrix, a function of a matrix $f(A)$ in which the matrix $A$ can only be accessed through matrix-vector products. Examples include estimating parameters in image restoration using the generalized cross-validation approach [1], exploring the inverse covariance matrix in uncertainty quantification [2, 3], computing observables in lattice quantum chromodynamics (LQCD) [4], or counting triangles in large graphs [5]. Explicitly computing $f(A)$ for a large and sparse matrix is a challenging task, so the Monte Carlo (MC) approach has become the standard method [6, 7]. The main purpose of this paper is to develop practical numerical techniques to address the computation of the trace of the inverse of a large, sparse matrix. But our technique can also be adapted to other functions such as the trace of the logarithm (yielding the determinant) or the trace of the matrix exponential.

For small size problems, computing $A^{-1}$ through a dense or sparse LDU decomposition is the most efficient and accurate approach [8]. This works well for discretizations of differential operators in low dimensions but becomes intractable in high dimensional discretizations. For larger size problems, domain decomposition and divide and conquer strategies are more tractable but still expensive [9]. In many cases, however, a low accuracy approximation is sufficient. Numerous methods have been presented to address this need for estimating the trace of the inverse of symmetric positive definite matrices through Gaussian quadratures [7, 10], modified moments [11, 12], and MC techniques [7, 10, 13, 14, 11, 6].

---

*Corresponding author
  *Email addresses:* `lfwu@cs.wm.edu` (Lingfei Wu), `andreas@cs.wm.edu` (Andreas Stathopoulos)

Variants of MC estimators are mainly analyzed and compared based on the variance of one sample [6, 15], which depends on the quality of the selected random vectors. Choosing random vectors having each element $\pm 1$ with equal probability is known to minimize variance over all other choices of random vectors [1, 6] and therefore has been widely used in many applications. In [6], Avron and Toledo analyze the quality of trace estimators through three different metrics such as trace variance, $(\varepsilon, \delta)$-approximation of the trace, and the number of random bits for different choices of random vectors. In [15], they improve the bounds of $(\varepsilon, \delta)$-approximation for the Hutchinson, Gaussian and unit vector estimators. However, the structure of the matrix can also be exploited to accelerate the convergence of the MC methods.

There has been a number of efforts to combine MC with well-designed vectors based on the structure of the matrix [13, 14, 16, 4]. In [14], they use columns of the Hadamard matrix, rather than random vectors, to systematically capture certain diagonals of the matrix. The MC iteration achieves the required accuracy by continuously annihilating more diagonals with more Hadamard vectors. However, the location of the nonzeros, or of the large elements of $A^{-1}$, often does not coincide with the diagonals annihilated by the Hadamard vectors. In [16], graph coloring and probing vectors are used to identify and exploit special structures, such as bandedness or decaying properties in the elements of $A^{-1}$, to annihilate the error contribution from the largest elements. However, if the error for the chosen number of colors is large, all work has to be discarded and the probing procedure repeated until the accuracy is satisfied. In [4], we introduced hierarchical probing to avoid the previous problems and achieve the required accuracy in an incremental way. For all these approaches, the approximation error comes from non-zero, off-diagonal elements that have not been annihilated yet. Instead, this paper looks only at the main diagonal of $A^{-1}$.

Our motivation for focusing only on the main diagonal is that the trace of $A^{-1}$ is simply a summation of a discrete, 1-D signal of either the eigenvalues or the diagonal elements of $A^{-1}$. Although we cannot compute all the diagonal elements, we may have an approximation to the whole signal from the diagonal of an approximation of $A^{-1}$ (e.g., of a preconditioner). If the two diagonals have sufficiently correlated patterns, fitting methods can be used to refine the approximation both for the diagonal and the trace. Therefore, the proposed method may serve as a standalone kernel for providing a good trace estimate with a small number of samples. But it can also be viewed as a preprocessing method for stochastic variance reduction for MC in cases where the variance reduces sufficiently. This can be monitored dynamically by our method.

We present several techniques that improve the robustness of our method and implement dynamic error monitoring capabilities. Our extensive experiments show that we typically obtain trace estimates with much better accuracy than other competing methods, and in some cases the variance is sufficiently reduced to allow for further improvements through an MC.

## 2. Preliminaries

We denote by $\|.\|$ the 2-norm of a vector or a matrix, by $N$ the order of $A$, by $Z$ the approximation of $A$, by $D$ the diagonal elements of $A^{-1}$, by $M$ the diagonal elements of $Z^{-1}$, by $Tr(f(A))$ the trace of the matrix $f(A)$, and by extension, $Tr(D)$ the sum of the elements of the vector $D$, by $T_{e_i}(f(A))$ the MC trace estimator of $f(A)$ using unit (orthocanonical) vectors, by $T_{Z_2}(f(A))$ the MC trace estimator of $f(A)$ using Rademacher vectors, by $diag(.)$ the diagonal operator of a matrix, and by $Var(.)$ the variance operator of a random variable or a vector.

### 2.1. Hutchinson trace estimator and unit vector estimator

The standard MC method to estimate the trace of the matrix inverse is due to Hutchinson [1]. It estimates the $Tr(A^{-1})$ by averaging $s$ quadratures with random vectors $z_j \in Z_2^N = \{z(i) = \pm 1 \text{ with probability } 0.5\}$,

$$T_{Z_2}(A^{-1}) = \frac{1}{s} \sum_{j=1}^{s} z_j^T A^{-1} z_j. \tag{1}$$

2

The variance of this method is given by

$$Var(T_{Z_2}(A^{-1})) = \frac{2}{s}\|A^{-1}\|_F^2 - \frac{2}{s}\sum_{i=1}^{N}\|D\|^2. \tag{2}$$

The variance of the Hutchinson trace estimator is proven to be minimum over other noise vectors [1]. The confidence interval of a MC method reduces as $O(\sqrt{Var(T_{Z_2}(A^{-1}))})$ for the given matrix.

The unit vector estimator uniformly samples $s$ vectors from the orthocanonical basis $\{e_1,\ldots,e_N\}$ [6],

$$T_{e_i}(A^{-1}) = \frac{N}{s}\sum_{j=1}^{s} e_{i_j}^T A^{-1} e_{i_j}, \tag{3}$$

where $i_j$ are the random indices. The variance of the unit vector estimator is given by

$$Var(T_{e_i}(A^{-1})) = \frac{N^2}{s}Var(D). \tag{4}$$

The variance of the Hutchinson method depends on the magnitude of the off-diagonal elements. It converges in one step for diagonal matrices, and very fast for highly diagonally dominant matrices. On the other hand, the variance of the unit vector estimator depends only on the variance of the diagonal elements. The unit vector estimator converges in one step if the diagonal elements are all the same, and very fast if the diagonal elements are similar. Thus, the method of choice depends on the particular matrix.

## 2.2. Reducing stochastic variance through matrix approximations

Given an approximation $Z \approx A$, for which $Z^{-1}$ and $Tr(Z^{-1})$ are easily computable, we can decompose

$$Tr(A^{-1}) = Tr(Z^{-1}) + Tr(E), \tag{5}$$

where $E = A^{-1} - Z^{-1}$. We hope that by applying the MC methods on $E$ instead on $A^{-1}$, the variance of the underlying trace estimator, in (2) or (4), can be reduced, thereby accelerating the convergence of MC. Although it not guaranteed in general, in our experiments we observed that such variance reduction often happened for the two estimators. Among many ways to obtain a $Z$, we focus on the following two.

The first approach is when $Z^{-1} = (LU)^{-1}$, where the $L$, $U$ matrices stem from an incomplete LU (ILU) factorization of $A$. If the ILU is sufficiently accurate, then $M = diag(Z^{-1})$ might be a good approximation to $D$. The vector $M$ can be obtained by computing only those entries $Z_{ij}^{-1}$ for which $L_{ij}$ or $U_{ij} \neq 0$ [17].

The second approach is a low rank approximation $Z^{-1} = V\Lambda^{-1}U^T$, where $\Lambda$ is a diagonal matrix with a few smallest magnitude eigenvalues (or singular values) of $A$, and $U$ and $V$ are the corresponding left and right eigenvectors (or singular vectors). In this paper, we only consider computing singular triplets. This subspace can be obtained directly by an iterative eigensolver [18], or approximated by methods such as eigCG [19] or eigBiCG [20] while solving linear systems of equations. The latter approach is computationally beneficial because moderate accuracy such as $10^{-6}$ is sufficient for our problem, and also because this low rank approximation can be used to deflate and thus accelerate subsequent linear systems. Clearly, the quality of the approximation $Z^{-1}$ depends on the separation of the computed singular space.

In some cases, $Tr(M)$ can be close to $Tr(A^{-1})$ even if $M$ does not approximate $D$ accurately due to cancellation of component-wise errors. In general, however, we note the opposite trend as shown for some example matrices in Figures 1 and 2 for ILU and SVD respectively. The $Tr(M)$ may not be accurate, but there is a clear correlation between the patterns of $M$ and $D$. Also note that the pattern correlation as computed by the SVD is better than that by ILU in the first example but worse in the second.

A computationally inexpensive, albeit less accurate approach for computing an approximation $M$ is based on variational bounds on the entries of $A^{-1}$ [7, 21]. Upper and lower bounds on the $i$-th diagonal entry

(a) *M* not close to *D* but captures pattern  (b) *M* quite close to *D* and captures pattern
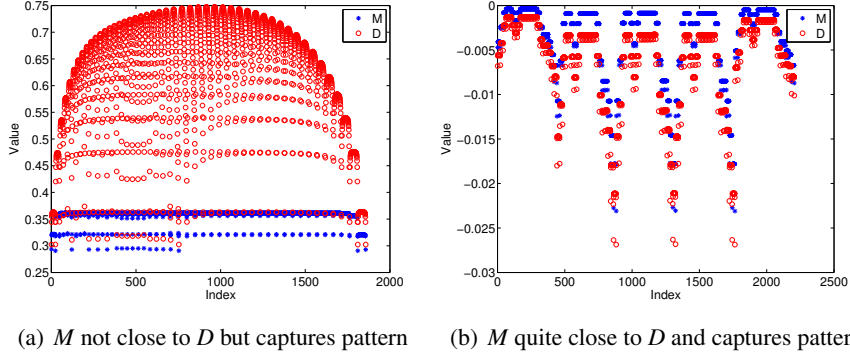
Figure 1: The pattern correlation between the diagonals of $A^{-1}$ and its approximation $Z^{-1}$ computed by ILU(0) on matrices (a) delsq50 and (b) orsreg2205. delsq50 is created in MATLAB by `delsq(numgrid('S',50))`.



(a) *M* not close to *D* but captures pattern  (b) *M* not close to *D* but captures pattern
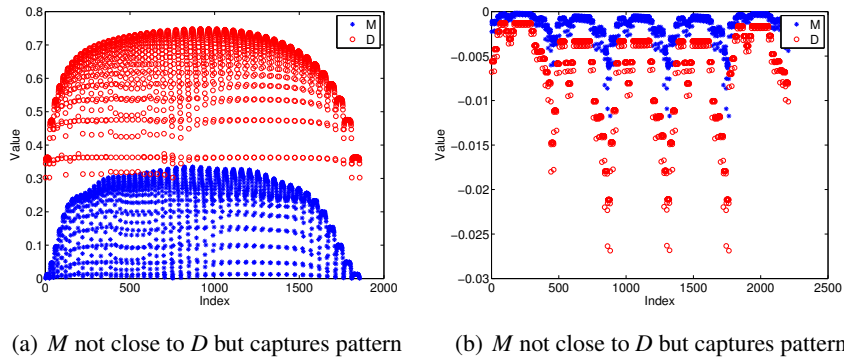
Figure 2: The pattern correlation between the diagonals of $A^{-1}$ and its SVD approximation $Z^{-1}$ computed from the 20 smallest singular triplets of *A* on matrices (a) delsq50 and (b) orsreg2205. delsq50 is created in MATLAB by `delsq(numgrid('S',50))`.

$A_{ii}^{-1}$ are derived inexpensively since they only depend on estimates of the smallest and largest algebraic eigenvalues, $\lambda_1, \lambda_N$, and the entries of *A*. The bounds apply to both symmetric and unsymmetric matrices. For the case of a real symmetric *A*, we have [21],

$$\frac{1}{\lambda_N} + \frac{(\lambda_N - A_{ii})^2}{\lambda_N(\lambda_N A_{ii} - s_{ii})} \le (A^{-1})_{ii} \le \frac{1}{\lambda_1} - \frac{(A_{ii} - \lambda_1)^2}{\lambda_1(s_{ii} - \lambda_1 A_{ii})}, \tag{6}$$

where $s_{ij} = \sum_{k=1}^{n} A_{ik} A_{kj}$. However the bounds in (6) will not be sharp especially the upper bound [11] and the error in the approximation can be large.

### 2.3. Comparison of different MC methods and discussion on importance sampling

Based on (2–4), we express the variance of the trace estimators $T_{Z_2}(E)$ and $T_{e_i}(E)$ as follows:

$$Var(T_{Z_2}(E)) = \frac{2}{s} \|E\|_F^2 - \frac{2}{s} \sum_{i=1}^{N} \|diag(E)\|^2, \tag{7}$$

$$Var(T_{e_i}(E)) = \frac{N^2}{s} Var(diag(E)). \tag{8}$$

Figures 1 and 2 show there is potential for the variances of $T_{Z_2}(E)$ or $T_{e_i}(E)$ to be smaller than those of $T_{Z_2}(A^{-1})$ and $T_{e_i}(A^{-1})$. However, for a given matrix, we must gauge which MC method would be better, and whether the variances need further improvement.

4

The estimator $T_{e_i}(E)$ has the interesting property that if $M = D + c$, where $c$ is a constant, then its variance in (8) is zero and we obtain the correct trace in one step. Although we cannot expect this in practice, it means that the shift observed between $M$ and $D$ in Figure 2(a) should not affect the effectiveness of $T_{e_i}(E)$.

On the other hand, $T_{e_i}(E)$ fails to identify correlations of the form $M = cD$. For such cases, importance sampling is preferred, where $M$ plays the role of a new distribution simulating the distribution of $D$. Assume that both $D$ and $M$ have been shifted by the same shift so that there are non negative, and that $M_i > 0$ if $D_i > 0$. To transform $M$ into a probability mass function, let $G = \frac{1}{Tr(M)} M$. To obtain an estimator of the trace of $D$ with importance sampling, we replace the uniform sampling of $D_i$ values with sampling with probability $G_i$ [22]. Then, instead of (3), the importance sampling estimator estimator is:

$$T_{IS}(D) = \frac{N}{s} \sum_{j=1}^{s} D_{i_j} \frac{\frac{1}{N}}{G_{i_j}} = \frac{Tr(M)}{s} \sum_{j=1}^{s} \frac{D_{i_j}}{M_{i_j}}. \tag{9}$$

When $M = cD$, the variance of $T_{IS}(D)$ is zero and it finds the trace in one step. However, it completely fails to identify shift correlations. In general $D$ and $M$ have a more complex relationship that neither $T_{IS}(D)$ or $T_{e_i}(E)$ can capture. This motivates our idea to explore general fitting models to approximate $D$.

## 3. Approximating the trace of a matrix inverse

We seek to construct a function $f$, such that $D \approx f(M)$. Then we can decompose

$$Tr(A^{-1}) = Tr(D - f(M)) + Tr(f(M)). \tag{10}$$

$Tr(f(M))$ is trivially computed for a given $f$. A key difference between the approaches in (10) and (5) is that a fitting model can improve a strong pattern correlation between $M$ and $D$ which is not easy or necessary for the whole matrix $Z^{-1}$. If $Tr(f(M))$ is a good approximation to $Tr(A^{-1})$ and its accuracy can be evaluated easily, then we can directly use this quantity; Otherwise, we can apply the unit vector MC estimator to compute $Tr(D - f(M))$ if the variance of MC on the vector $E_{fit} = D - f(M)$ given by

$$Var(T_{e_i}(E_{fit})) = \frac{N^2}{s} Var(E_{fit}) \tag{11}$$

is smaller than the variances in (2), (3), (7) and (8).

---

**Algorithm 1** Basic algorithm for approximating $Tr(A^{-1})$.

---

    **Input** : $A \in \mathbb{R}^{n \times n}$, $k \in \mathbb{Z}$
    **Output** : $Tr(A^{-1})$ estimation and $Z \in \mathbb{R}^{n \times n}$
    % $Z$ and $Z^{-1}$ are given in their implicit forms
    % $Z^{-1}$ also acts as a preconditioner when solving linear systems
  1: Compute an approximation $Z^{-1}$ and $M = diag(Z^{-1})$
  2: Compute fitting sample $S_{fit}$, a set of $k$ indices.
  3: Solve linear systems $D_i = e_i^T A^{-1} e_i$, $\forall i \in S_{fit}$.
  4: Obtain a fitting model $f(M) \approx D$ by fitting $f(M(S_{fit}))$ to $D(S_{fit})$.
  5: Compute refined trace approximation $T_{e_i}(E_{fit})$.
  6: Estimate and monitor the relative trace error and, if needed, the variances for different MC methods

---

The basic description of the proposed estimator is outlined in Algorithm 1. First, our method computes an approximation $M$ of $D$ by using the methods discussed in the previous section. Second, it finds a fitting sample $S_{fit}$, a set of indices that should capture the important distribution characteristics of $D$. Since we

have no information about $D$, we discuss in the following section how to tackle this task by considering the distribution of $M$. Third, it computes the values of $D(S_{fit})$ by solving the corresponding linear systems. Since this is the computational bottleneck, the goal is to obtain good accuracy with far fewer fitting points than the number of vectors needed in MC. Fourth, it computes a fitting model that has sufficient predictive power to improve the diagonal approximation. This critical task is discussed in Section 3.2. Finally, since there are no a-posteriori bounds on the relative error of the trace, we use a combination of statistical approaches and heuristics to estimate it incrementally at every step. If a MC method is needed afterwards, we also estimate the variances of different MC estimators during the fitting process so that we choose the MC method with the smallest variance. The following sections address the challenges arising from these tasks.

*3.1. Point Identification Algorithm*

It is helpful to view the computation of $Tr(D)$ as a one dimensional integral that we need to approximate with only a few points. Because it is one dimensional, it may be surprising that Monte Carlo is the standard approach and not numerical integration. However, $D$ as a one dimensional function of its index may have no smoothness, i.e., $D_i$ and $D_{i+1}$ may be arbitrarily different. Even for matrices that model physically smooth phenomena (e.g., in PDEs), the matrix may be given in an ordering that does not preserve physical locality. Without smoothness numerical integration cannot work better than MC.

Consider now a sorted permutation of the diagonal, $\hat{D} = sort(D)$. Obviously $Tr(D) = Tr(\hat{D})$, but $\hat{D}$ is monotonic and maximally smooth among all permutation of $D$. Monotonic implies that, in the absence of any additional information about the data, a simple trapezoidal rule minimizes the worst case integration error [23]. If in addition we are allowed to choose the integration points sequentially, based on the points computed thus far, much better average case error can be obtained [24, 25]. On the other hand, if bounds are known on the smoothness of $\hat{D}$, better worst case error bounds can be established. Since $D$, however, is not available, we turn to its approximation $M$.

A close pattern correlation between $M$ and $D$ means that the elements of $M$ should have a similar distribution as those of $D$, or that $\hat{M} = sort(M)$ should be similar to $\hat{D}$. Thus, we work on the surrogate model $\hat{M}$ for which we can afford to identify the best quadrature points that yield the smallest error in $Tr(\hat{M})$. Then, these will be the ideal points for finding the fitting $f$. Specifically, we need to select indices that capture the important distribution changes in $\hat{M}$. For example, identifying minimum and maximum elements of $M$ sets the range of approximation for $f$ and avoids extrapolation. We also look for entries in $\hat{M}$ that deviate highly from their neighbors (where the derivative of $\hat{M}$ and hopefully of $\hat{D}$ is high). The advantages are twofold. First, the integral $Tr(\hat{M})$ should be captured well by the trapezoidal rule between such indices. Second, and more important, we can obtain a more accurate fitting function $f$ in a piecewise manner in intervals where $\hat{D}$ has similar behavior. An example is shown in Figure 3(a), where sampling on $M$ requires a large number of samples to capture the pattern of $M$ and its relation to $D$. In contrast, far fewer points are needed for $\hat{M}$, and these points are easily mapped to a pattern of $\hat{D}$, as shown in Figure 3(b).

The proposed index selection method is shown in Algorithm 2. Initially the set of sampled indices, $\hat{S}_{fit}$, includes $1, N$ of the extrema of $\hat{M}$. Then, for every interval $(i, j)$, with $i, j$ consecutive indices in $\hat{S}_{fit}$, we find the index $t$ from $i+1$ to $j-1$ that minimizes the trapezoidal rule error for computing $Tr(\hat{M}(i:j))$:

$$\underset{t \in (i\ j), t \in \mathbb{Z}}{\arg\min} \ (|(\hat{M}(i) - \hat{M}(j)) * (i - j) - (\hat{M}(i) - \hat{M}(t)) * (i - t) - (\hat{M}(t) - \hat{M}(j)) * (t - j)|). \quad (12)$$

The process continues until it reaches a maximum number of sampling points or until the maximum error over all intervals decreases by a value, say 0.001. This threshold depends not only on how well $\hat{M}$ approximates $\hat{D}$, but also on how well the orderings from $M$ and $D$ are matched. For example, if $M$ is a random permutation of $D$, there is no good fitting, even though $\hat{M} = \hat{D}$. We discuss this in Section 4.2. Therefore, going below this threshold helps the accuracy of $Tr(M)$ but not necessarily of $Tr(D)$. Instead, we continue by simply bisecting the longest intervals until *maxPts* is reached. Line 11 forces a single bisection also every

five points. On exit, we compute $S_{fit}$ which maps $\hat{S}_{fit}$ to the original unsorted ordering. A typical sampling result produced by our method is shown in Figure 3.

---

**Algorithm 2** Point identification algorithm based on the trapezoidal rule

---

1: $[\hat{M}, J] = sort(M)$
2: $numSamples = 1$, $initErr = tempErr = |Tr(\hat{M}) - (\hat{M}(1) + \hat{M}(2))\frac{N-1}{2}|$
3: Add $1, N$ into $\hat{S}_{fit}$ and push interval $(1, N)$ and its $tempErr$ in $Q$.
4: **while** $numSamples < maxPts$ and $tempErr > 0.001 * initErr$ **do**
5:     Pop interval $(L, R)$ with largest error from $Q$
6:     **for** $k = L + 1 : R - 1$ **do**
7:         Use (12) to find a bisecting index $t$
8:     **end for**
9:     Add $t$ into $\hat{S}_{fit}$, $numSamples = numSamples + 1$
10:     Push intervals $(L, t)$ and $(t, R)$ each with their corresponding $tempErr$ in $Q$
11:     **if** $numSamples$ is a multiple of 5 **then**
12:         Insert one midpoint index into largest interval in $\hat{S}_{fit}$, $numSamples = numSamples + 1$
13:         Insert its corresponding left and right intervals in $Q$
14:     **end if**
15: **end while**
16: **while** w **do**hile $numSamples < maxPts$,
17:     Insert middle index of the largest interval into $S$, $numSamples = numSamples + 1$
18: **end while**
19: Return $S_{fit}$ indices in original ordering, such that $\hat{S}_{fit} = J(S_{fit})$

---



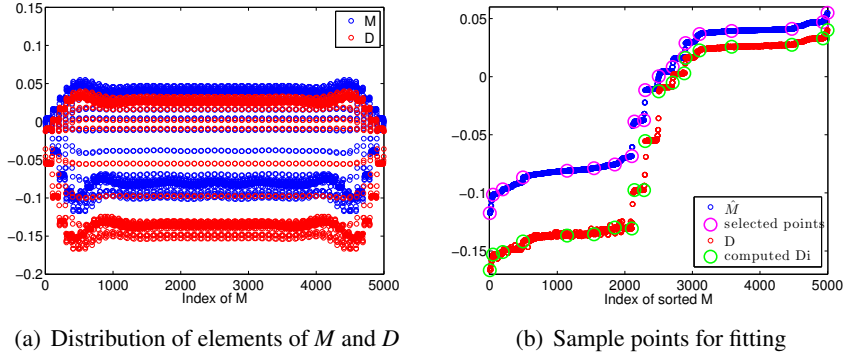(a) Distribution of elements of $M$ and $D$      (b) Sample points for fitting

Figure 3: A typical example to show our sampling strategy based on the pattern correlation of $M$ and $D$. $M$ is computed by 20 singular vectors of matrix RDB5000 with accuracy $1e - 6$. In the right figure, the pink and green circles denote the sample points associated with the sampling indices $S_{fit}$ in $\hat{M}$ and $D$.

### 3.2. Two Fitting Models

After the sample index set $S_{fit}$ is obtained, the associated elements $D(S_{fit})$ can be computed. Next we construct a fitting model that minimizes $\|f(M(S_{fit})) - D(S_{fit})\|$. The fitting model must have sufficient predictive power with only a small number of points and it should avoid oscillating behavior.

The MC methods in (5) and (10) can resolve the trace when $D = M + c$ while importance sampling may resolve the trace when $D = cM$. To combine these, we first use a linear model, $y = bM + c$. We determine the parameters $b, c$ by a least squares fitting, $argmin_{b,c \in R} \|D(S_{fit}) - (bM(S_{fit}) + c)\|_2$. The linear model may be

simple but avoids the large oscillations of higher degree polynomials, and in many cases it is quite effective in improving the accuracy of the trace estimation and reducing the variance of the diagonal elements of $E_{fit}$. The linear fitting algorithm is described in Algorithm 3. We show an example of the fitting result in the original order of $D$ in Figure 4(a).

---

**Algorithm 3** Linear least squares fitting model for approximating $Tr(A^{-1})$.

---

1: Compute $M$ using ILU or Eigendecompostion or SVD on $A$
2: Call Algorithm 2 to compute sample set $S_{fit}$.
3: Find $[b,c] = \text{argmin}\|D(S_{fit}) - (bM(S_{fit}) + c)\|_2$.
4: Compute trace approximation $T_f = \sum_{i=1}^{N}(b * M + c)$

---

Although the linear model preserves the shape of $M$, it relies too much on the quality of $M$. To take advantage of our premise that the fitting points approximate the distribution $\hat{M}$, our next fitting model is the Piecewise Cubic Hermite Spline Interpolation (PCHIP). It was proposed in [26] to construct a visually pleasing monotone piecewise cubic interpolant to monotone data. The PCHIP interpolant is only affected locally by changes in the data and, most importantly, it preserves the shape of the data and respects monotonicity. Therefore, we work on $\hat{M}$ and the indices $\hat{S}_{fit} = [1 = s_1, s_2, \ldots, s_{k-1}, s_k = N]$ which are given in order such that $\alpha = \hat{M}(s_1) \leq \hat{M}(s_2) \leq \cdots \leq \hat{M}(s_k) = \beta$ is a partition of the interval $[\alpha, \beta]$. An index $s_i$ corresponds to the index $J^{-1}(s_i)$ in the original ordering of $M$, where $J$ is from Algorithm 2. Thus, for each $s_i$ we compute $D(J^{-1}(s_i)), i = 1, \ldots, k$, and we use PCHIP to construct a piecewise cubic function such that,

$$p(\hat{M}(s_i)) = D(J^{-1}(s_i)), \quad i = 1, 2, \ldots, k. \tag{13}$$

Notice that $p(x)$ will be monotone in the subintervals where the fitting points $D(J^{-1}(s_i))$ are also monotone. Therefore, as long as $M$ is close to $D$, integration of $p(x)$ will be very accurate.

The PCHIP model is given in Algorithm 4. The first two steps are the same as in Algorithm 3. In step 3, we apply the function **unique** to remove the duplicate elements of $\hat{M}(\hat{S}_{fit}))$ to produce a sequence of unique values as required by PCHIP. This yields a subset of the indices, $\hat{S}'_{fit}$, which is mapped to original indices as $I = J^{-1}(\hat{S}'_{fit})$ to be used in PCHIP. We show an example of the fitting results in Figure 4(b).

---

**Algorithm 4** PCHIP fitting model for approximating $Tr(A^{-1})$.

---

3: Remove duplicates: $\hat{S}'_{fit} = \textbf{unique}(\hat{M}(\hat{S}_{fit})), I = J^{-1}(\hat{S}'_{fit})$
4: Apply PCHIP to fit $p(M(I)) = D(I)$ and obtain a polynomial $p(M) \approx D$
5: Compute trace approximation $T_f = \sum_{i=1}^{N} p(M)$

---

Table 1 compares the relative trace error and variances on $D$, $D - M$ and $D - p(M)$ using linear LS and PCHIP models for two matrices. In both cases, the two models can provide a trace estimate of surprising relative accuracy $O(1e-2)$ with only 20 fitting points. In addition, for matrix OLM5000, the standard deviation of MC on $D - p(M)$ is reduced by a factor of 10 compared to that of MC on $D$ (a speedup of 100 in terms of samples). However, for matrix KUU, the standard deviation of MC on $D - p(M)$ does not improve much over MC on $D$. Therefore, our methods could serve as a standalone kernel for giving a fast trace estimate or may stand as a preprocessing stage for accelerating MC when the variance reduction is sufficient.

## 4. Dynamic Evaluation of Variance and Relative Trace Error

Since there are no a-posteriori bounds for the accuracy of our results, we develop methods that use the information from the solution of the linear systems to incrementally estimate the trace error and the

(a) Linear LS model in original order      (b) PCHIP model in sorted order
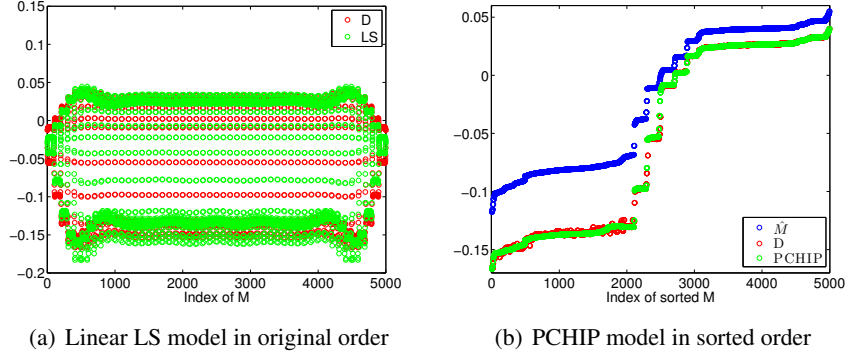
Figure 4: Fitting results of the matrix RDB5000 in original order and sorted order with linear LS model and PCHIP model.

Table 1: Comparing trace estimation, relative trace error and variances in (4), (8) and (11) on matrices OLM5000 and KUU between linear LS model and PCHIP model with 20 fitting points.

| Matrix | OLM5000 | | KUU | |
|---|---|---|---|---|
| Model | LS | PCHIP | LS | PCHIP |
| $Trace$ | -5.0848e+02 | -5.0848e+02 | 3.6187e+03 | 3.6187e+03 |
| $TraceEst$ | -4.9713e+02 | -5.0122e+02 | 3.5745e+03 | 3.5663e+03 |
| $RelErr$ | 2.2320e-02 | 1.4288e-02 | 1.2207e-02 | 1.4469e-02 |
| $Std(T_{e_i}(A^{-1}))$ | 1.1425e+02 | 1.1425e+02 | 2.8731e+02 | 2.8731e+02 |
| $Std(T_{e_i}(D-M))$ | 1.1007e+02 | 1.1007e+02 | 1.4999e+02 | 1.4999e+02 |
| $Std(T_{e_i}(D-p(M)))$ | 2.0332e+01 | 1.7252e+01 | 1.6137e+02 | 1.6289e+02 |

variances of the resulting approximations. This approach is also useful when $M$ is updated with more left and right eigenvectors or singular vectors obtained from the solution of additional linear systems.

*4.1. Dynamic Variance Evaluation*

To decide which MC method we should use after the fitting stage or even whether it is beneficial to use the fitting process for variance reduction, we monitor incrementally the following variances, $Var(T_{e_i}(A^{-1}))$, $Var(T_{e_i}(E_{fit}))$, $Var(T_{Z_2}(A^{-1}))$, and $Var(T_{Z_2}(E))$, with the aid of the cross-validation technique [27].

Our training set is the fitting sample set $D(S_{fit})$, while our test set $D(S_{mc})$ is a small random set which is independent from the fitting sample set. If we want to combine our method with MC, eventually more samples need to be computed, so we can pre-compute a certain number of them as the test set, $D(S_{mc})$. We have used the holdout method [28], a single train-and-test experiment for some data splitting strategy since the fitting sample set is fixed.

To compute $D(S_{fit})$ or $D(S_{mc})$, a linear system with multiple right hand sides is solved as follows:

$$A_{ii}^{-1} = e_i^T x_i, \quad Ax_i = e_i, \ \forall i \in S_{fit} \cup S_{mc}, \tag{14}$$

The computed column vectors $x_i$ can be used to estimate the Frobenius norm of both $A^{-1}$ and $E = A^{-1} - Z^{-1}$ [29, 30]. Then $Var(T_{Z_2}(A^{-1}))$ and $Var(T_{Z_2}(E))$ can be estimated as follows:

$$Var(T_{Z_2}(A^{-1})) \approx \frac{2N}{s^2} \sum (\|x_i\|^2 - |D_i|^2), \ \forall i \in S_{fit} \cup S_{mc}, \tag{15}$$

$$Var(T_{Z_2}(E)) \approx \frac{2N}{s^2} \sum (\|E(:,i)\|^2 - |E(i,i)|^2), \ \forall i \in S_{fit} \cup S_{mc}, \tag{16}$$

9

where $E(:,i) = Ee_i$. Simultaneously, based on the sampled diagonal elements $A_{ii}^{-1}$, we can also update the evaluation of $Var(T_{e_i}(E_{fit}))$, $Var(T_{e_i}(E))$ and $Var(T_{e_i}(A^{-1}))$. Here we only show the computation of the unbiased variance estimation for $Var(T_{e_i}(E_{fit}))$ by:

$$Var(T_{e_i}(E_{fit})) \approx \frac{N^2}{s-1} Var(E_{fit}(S_{mc})). \tag{17}$$

Note that $S_{fit}$ should not be used for estimating the variance of unit vector MC estimator since these sample points are exact roots of the PCHIP function.

---

**Algorithm 5** Dynamic variance evaluation algorithm for estimating variances of different MC methods

1: Initialize $maxPts$, $S_{fit}$, $S_{mc}$
2: **if** $M$ is computed using ILU on A **then**
3:     Compute the approximation $M$
4: **end if**
5: Generate random index set $S_{mc}$ without replacement and compute MC samples $D(S_{mc})$
6: **for** $i = 5 : 1 : maxPts$ **do**
7:     **if** $M$ is computed using Eigendecompostion or SVD on A **then**
8:         Update $M$ with $2*i$ number of left and right eigenpairs or singular triplets
9:     **end if**
10:     Call Algorithm 2 to find more indices so that $S_{fit}$ has $i$ fitting points
11:     Call Algorithms 3 or 4 to update approximation of $Tr(A^{-1})$
12:     Estimate variances of different MC methods based on (15), (16) and (17)
13: **end for**

---

We implement the dynamic variance evaluation scheme in Algorithm 5. In lines 2-4 and 7-9, the approximation $M$ can be computed by using ILU at the beginning or updated with increasing number of singular triplets or eigenpairs. Note that if $M$ is obtained by eigendecomposition or SVD, the updated $M$ is different in two consecutive steps, so Algorithm 2 would return a slightly different index set $S_{fit}$ which may not be incremental. This may not provide a consistent improvement of the relative trace error during the fitting progress. In line 10 of the algorithm, we force the points to be incremental between steps $i$ and $i+1$ as follows; we generate the entire set $S_{fit}^{(i+1)}$ and remove its indices that are closest to the previous index set $S_{fit}^{(i)}$. The remaining index is incorporated into $S_{fit}^{(i)}$. This simple scheme works quite well experimentally.

Figure 5 shows how the actual and the estimated variances for three MC methods match well for an example matrix. In addition, the relative difference between different MC methods becomes clear after only a few points which facilitates not only the proper choice of MC method, but also an early decision to stop if further fitting is not beneficial. In the experiments section we show that these results are typical for matrices from a wide variety of applications.

*4.2. Monitoring Relative Trace Error*

Table 1 showed that even though the variance of MC on the vector $E_{fit}$ may not be reduced, our deterministic fitting method can still provide a good trace estimation with only a small number of samples. We investigate the reason by comparing the elements of $D$ and $p(M)$ with different orderings. Figure 6(a) shows the elements of $p(M(J))$ and $D(J)$, i.e., with respect to the order of $\hat{M}$. It illustrates that although $p(M)$ captures the pattern of $D$, the order of its elements does not correspond exactly to that of $D$; hence the small reduction in $Var(T_{e_i}(E_{fit}))$. However, Figure 6(b) reveals that the distributions of the sorted $p(M)$ and the sorted $D$ almost coincide; hence, the two integrals $Tr(p(M))$ and $Tr(D)$ are very close.

The above observations suggest that the method could be used also as a standalone kernel for estimating the trace. The obstacle is that there is no known way to measure or bound the relative trace error. Resorting
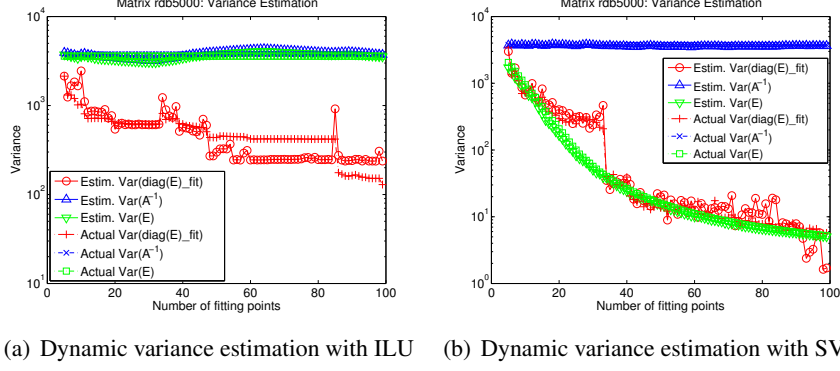
(a) Dynamic variance estimation with ILU     (b) Dynamic variance estimation with SVD

Figure 5: Comparing estimated variances and actual variances of unit vector on $E_{fit}$ and Rademacher vector on $A^{-1}$ and $E$ of the matrix RDB5000 with ILU and SVD respectively.
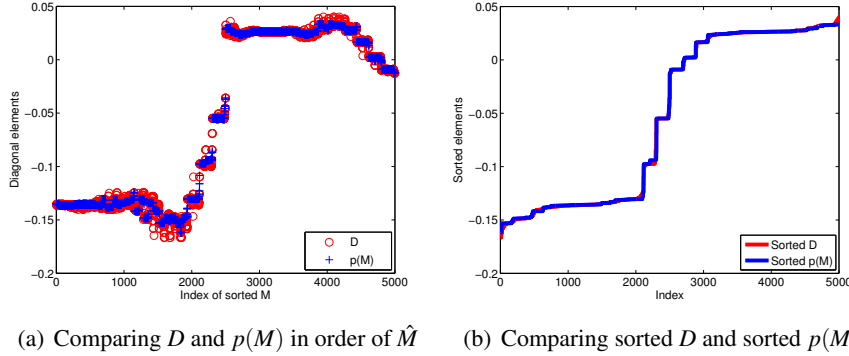


(a) Comparing $D$ and $p(M)$ in order of $\hat{M}$     (b) Comparing sorted $D$ and sorted $p(M)$

Figure 6: Compare $D$ and $p(M)$ of the matrix RDB5000 in different order where $M$ is computed by ILU.

to the confidence interval computed by the variance of a MC estimator is pessimistic as our results show (see also [6, 15]). We note the similarity to the much smaller error obtained in the average case of integrating monotonic functions with adaptive quadratures versus the worst case known bounds [23, 24].

At each step of the fitting process in Algorithm 5, we collect a sequence of trace estimates $T_i, i \in [1, maxPts]$. Consider the trace estimations in two successive steps,

$$\frac{|T_i - T_{i+1}|}{|T_{i+1}|} = \frac{|(T_i - Tr(D)) - (T_{i+1} - Tr(D))|}{|T_{i+1}|} = \frac{|E_i - E_{i+1}|}{|T_{i+1}|} \leq \frac{2max(|E_i|, |E_{i+1}|)}{|T_{i+1}|} \approx \frac{2max(|E_i|, |E_{i+1}|)}{|Tr(D)|}. \tag{18}$$

As long as $T_i$ converges with more fitting points, the relative difference of two successive trace estimations can serve as an approximation to the relative error. However, when the global pattern provided by $M$ and $p(M)$ is fully matched to that of $D$, convergence of $T_i$ stagnates until enough points have been added to resolve the various local patterns. To determine whether the current relative trace error estimation can be trusted, we present our second heuristic by considering the error bound of our fitting models.

When approximating $f(\hat{M})$ with $p(\hat{M})$, the PCHIP Hermite cubic splines with $n$ points on the interval $[\alpha, \beta]$, the bound on the error $E(\hat{M}) = f(\hat{M}) - p(\hat{M})$ is given by [31],

$$|E(\hat{M})| \leq \frac{1}{384} h^4 \|f^{(4)}\|_{\infty, [\alpha, \beta]} , \tag{19}$$

where $h = \frac{\beta - \alpha}{n}$, and $\|f^{(4)}\|_{\infty, [\alpha, \beta]}$ denotes the maximum value of the fourth derivative of $f$ in the entire

interval $[\alpha, \beta]$. Since $\|f^{(4)}\|_{\infty, [\alpha, \beta]}/384$ is a constant, in two successive fitting steps we have,

$$\frac{|E_i(\hat{M})|}{|E_{i+1}(\hat{M})|} \approx \frac{h_i^4}{h_{i+1}^4} = \frac{((\beta-\alpha)/n_i)^4}{((\beta-\alpha)/n_{i+1})^4} = \left(\frac{n_{i+1}}{n_i}\right)^4, \tag{20}$$

where $n_i$ and $n_{i+1}$ are the number of fitting points in two consecutive steps. We can use (20) to estimate the maximum possible improvement between two consecutive trace errors. If the $i+1$ trace error estimate reduces over the $i$-th estimate by a factor of more than $(n_i/n_{i+1})^4$, we do not trust it.

---

**Algorithm 6** Dynamic relative trace error evaluation algorithm during the fitting process

---

    % $TraceErr(i)$ is defined only for $i > 5$ and we assume TraceErr(6) is well defined
1: **if** $i == 6$ **then**
2:    $TraceErr(6) = |TraceFit(6) - TraceFit(5)|/|TraceFit(5)|$
3: **end if**
4: **if** $i > 6$ **then**
5:    $TempTraceErr = |TraceFit(i) - TraceFit(i-1)|/|TraceFit(i)|$
6:    **if** $(TempTraceErr/TraceErr(i-1) \geq ((i-1)/i)^4)$ **then**
7:      $TraceErr(i) = TempTraceErr$
8:    **else**
9:      $TraceErr(i) = TraceErr(i-1) * ((i-1)/i)^{9/4}$
10:    **end if**
11: **end if**

---

One caveat is that (19) may not be tight since the same bound holds for each subinterval $[\hat{M}(s_j), \hat{M}(s_{j+1})]$. This means that a high derivative in one subinterval might dominate the bound in (19) but should not affect the error in other intervals. Therefore, convergence might not be fully dictated by (20). In practice, we found that the improvement ratio is between $O(n_i/n_{i+1})$ and $O((n_i/n_{i+1})^4)$. Therefore, if the current relative trace error estimate is determined not to be trusted, we may instead use $|E_{i+1}(\hat{M})| = (n_i/n_{i+1})^k |E_1(\hat{M})|$, $k \in [1, 4]$. The choice of $k$ depends on the quality of $M$. In our experiments, we use the geometric mean of the four rates, yielding $k = 9/4$. Recall that the corresponding ratio in MC is $O(\sqrt{n_i/n_{i+1}})$, which is much slower than the proposed trace estimation method.

Algorithm 6 combines the two heuristics in (18) and (20) to dynamically monitor the relative trace error. It is called after step 12 of Algorithm 5. Figure 7 shows two examples of how our dynamic method provides reasonable estimates of the relative trace error.
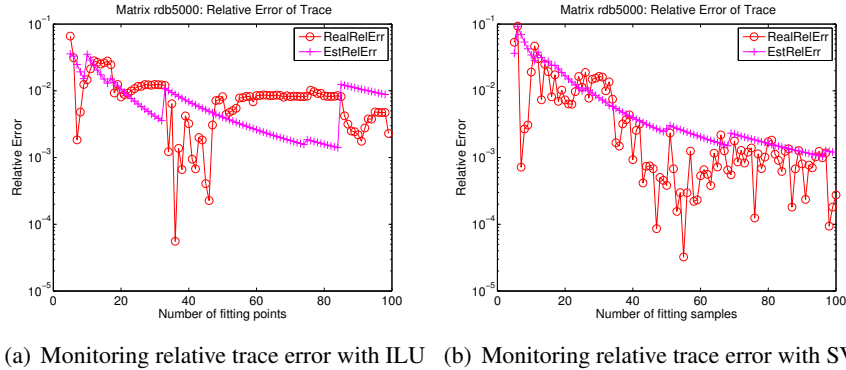


(a) Monitoring relative trace error with ILU    (b) Monitoring relative trace error with SVD

Figure 7: Two examples of monitoring relative trace error of the matrix RDB5000 with ILU and SVD respectively.

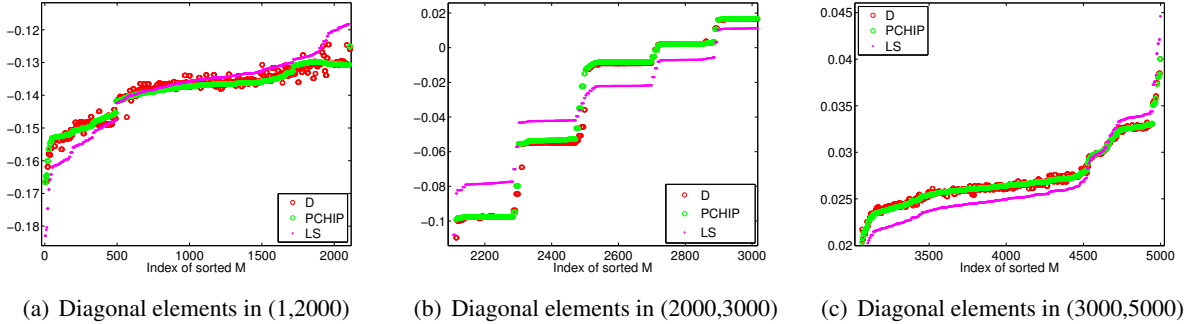| (a) Diagonal elements in (1,2000) | (b) Diagonal elements in (2000,3000) | (c) Diagonal elements in (3000,5000) |

Figure 8: Comparing the linear LS model with the PCHIP model on RDB5000 matrix with $M$ from SVD.

## 5. Numerical Experiments

We run experiments on matrices that are sufficiently large to avoid problems of sampling from small spaces but can still be inverted to obtain the exact trace error. We select matrices RDB5000 and cfd1 from the University of Florida sparse matrix collection [32] and generate three test matrices from applications that appear in [7]. The Heatflow160 matrix is from the discretization of the linear heat flow problem using the simplest implicit finite difference method. The matrix Poisson150 is from 5-point central difference discretization of the 2D Poisson's equation on a square mesh. The VFH6 matrix is from the transverse vibration of a Vicsek fractal that is constructed self-similarly. We also use matrix matb5 which is a discretization of the Wilson Dirac operator on $8^4$ lattices with 12 degrees of freedom at each link, using a mass near to critical. Table 2 lists these matrices along with some of their basic properties. All experiments are conducted using MATLAB 2013a. The number of fitting points increases as $s = 5 : 100$. The approximation $M$ is computed by ILU with parameters `type = ilutp` and `droptol = 1E-2`, or as a low rank approximation of $2s$ smallest singular vectors (twice the number of fitting points at each step), or by the bounds on the diagonal.

Table 2: Basic information of the test matrices

| Matrix | Order | nnz(A) | $\kappa(A)$ | Application |
|--------|-------|--------|-------------|-------------|
| RDB5000 | 5000 | 29600 | 1.7E3 | computational fluid |
| cfd1 | 70656 | 1825580 | 1.8E7 | computational fluid |
| Heatflow160 | 25600 | 127360 | 2.6E0 | linear heat flow |
| Poisson150 | 22500 | 111900 | 1.3E4 | computational fluid |
| VFH6 | 15625 | 46873 | 7.2E1 | vicsek fractal |
| matb5 | 49152 | 2359296 | 8.2E4 | lattice QCD |

### 5.1. Effectiveness of the fitting models

In Figure 8, we divide the diagonal elements of the matrix RDB5000 into three contiguous sets and zoom in the details. We see that despite a good $M$, the linear LS model cannot scale the entire $M$ onto $D$. The more flexible piecewise approach of PCHIP results in a much better fit.

In Figure 9, we look at three matrices with $M$ generated using the SVD. The PCHIP model typically has smaller relative trace error than the LS model. We also see that as more fitting points are sampled, the relative trace error of both models decreases significantly at early stages and slowly after a certain point. This relates to the quality of $M$, not of the model. Typically $M$ will approximate the global pattern of $D$ and the two can be matched well with only a few fitting points. But if the local patterns of $M$ and $D$ differ, a large
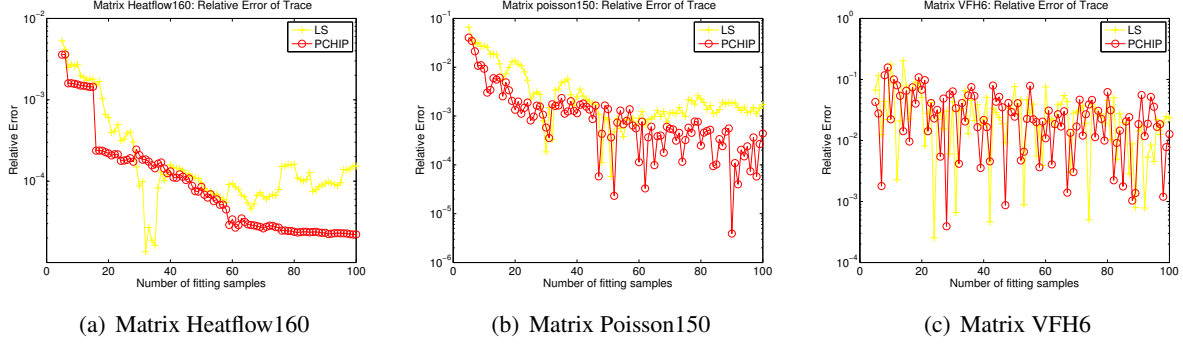
13

(a) Matrix Heatflow160　　　　　　(b) Matrix Poisson150　　　　　　(c) Matrix VFH6

Figure 9: Comparing relative trace error between the LS model and the PCHIP model in three typical cases with SVD.



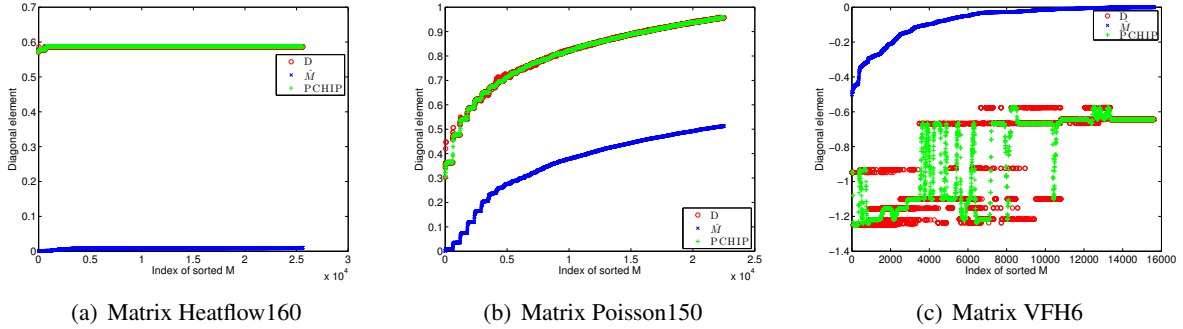(a) Matrix Heatflow160　　　　　　(b) Matrix Poisson150　　　　　　(c) Matrix VFH6

Figure 10: Fitting results of three typical cases with the PCHIP model and 100 fitting points using SVD.

number of fitting points will be required. This can be seen in Figure 10. Using the permutation of $\hat{M}$ for each case, we plot $\hat{M}$, $D$, and $p(\hat{M})$ using 100 fitting points. For the Heatflow160 matrix, $p(M)$ approximates $D$ well everywhere except for the small leftmost part of the plot, which allows the relative error to reach below $10^{-4}$ before convergence slows down (Figure 9). The behavior is similar for the Poisson150. The issue is more pronounced on matrix VFH6, where $M$ and $p(M)$ capture the average location of $D$ but completely miss the local pattern, which is reflected by a very slowly improving error in Figure 9.

We mention that the irregularity of the relative trace errors in Figure 9 relates to the variability of successive updates of $M$ and of the sampling indices, especially when $M$ is of lower quality.

Figure 11 demonstrates that the PCHIP model has smaller actual variance for MC on $E_{fit}$ than the LS model. Therefore, we only consider the PCHIP model in the rest of experiments.

### 5.2. Comparison between the fitting model and different MC methods

We address the question of whether the number of matrix inversions we spend on computing the fitting could have been used more efficiently in an MC method, specifically the Hutchinson method on $A^{-1}$ and the Hutchinson method on $E$. In Table 3 we compare the relative trace error of the PCHIP model with 20 fitting points against the relative errors of the two MC methods as computed explicitly from their respective standard deviations in (2) and (7), with $s = 20$, divided by the actual trace of $D$.

When $M$ approximates $D$ sufficiently well, the trace from the fitted diagonal is better, and for the ILU approximations far better, than if we just use the Hutchinson method on $A^{-1}$ (the first column of results). Although MC on $E$ exploits the ILU or SVD approximation of the entire matrix (not just the diagonal that our method uses), we see that it does not always improve on MC on $A^{-1}$, and in some cases (cfd1 with ILU) it is far worse. In contrast, our diagonal fitting typically improves on MC on $E$. The last column shows that even with an inexpensive diagonal approximation we obtain a similar or better error than MC on $A^{-1}$. The

only exception is the matrix VFH6 where, as we saw earlier, *M* cannot capture the pattern of *D*. Even then, its error is close to the errors from the MC methods and, as we show next, the best method can be identified dynamically with only a small number of samples.

Table 3: Relative trace error from our PCHIP model and from the MC method on $A^{-1}$ and on $E$ (computed explicitly as the standard deviation with $s = 20$ from (2) and (7) divided by the actual trace).

| | | ILU | | SVD | | Bounds |
|---|---|---|---|---|---|---|
| Matrix | $T_{Z_2}(A^{-1})$ | PCHIP | $T_{Z_2}(E)$ | PCHIP | $T_{Z_2}(E)$ | PCHIP |
| RDB5000 | 5.2E-2 | 8.1E-3 | 4.8E-2 | 4.1E-3 | 1.2E-2 | 5.3E-2 |
| cfd1 | 1.3E-1 | 2.8E-2 | 8.2E+2 | 8.8E-3 | 1.8E-2 | 2.6E-2 |
| Heatflow160 | 4.9E-4 | 1.6E-7 | 4.0E-5 | 2.0E-4 | 4.9E-4 | 3.5E-4 |
| Poisson150 | 2.6E-2 | 2.3E-3 | 2.5E-2 | 1.4E-3 | 4.3E-3 | 8.3E-3 |
| VFH6 | 3.2E-3 | 6.8E-5 | 3.1E-5 | 1.0E-2 | 3.2E-3 | 6.0E-2 |

If the user requires better trace accuracy than our fitting technique provides, we explore the performance of the diagonal fitting as a variance reduction for MC with unit vectors. We compute the actual values of $Var(T_{e_i}(E_{fit}))$, $Var(T_{e_i}(A^{-1}))$, $Var(T_{e_i}(E))$, $Var(T_{Z_2}(A^{-1}))$ and $Var(T_{Z_2}(E))$ for every step $s = 5 : 100$ and show results for three matrices in Figure 11. Note that the low rank approximation uses $2s$ singular vectors. As before, for Heatflow160, MC on $E_{fit}$ performs much better than other MC methods, achieving about two orders reduction in variance. For Poisson150, MC on $E_{fit}$ is slightly better compared to the Hutchinson method on $E$. In contrast, for VFH6, MC with unit vectors do not perform well regardless of the diagonal.
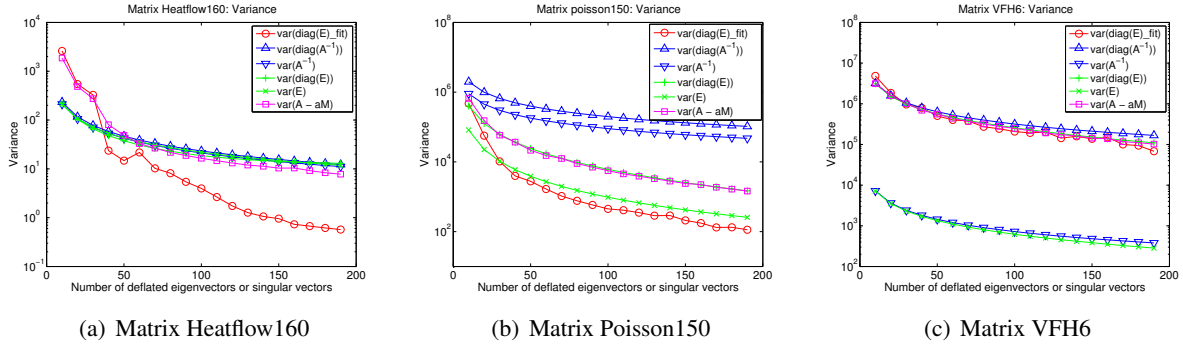


(a) Matrix Heatflow160      (b) Matrix Poisson150      (c) Matrix VFH6

Figure 11: Comparing actual variances of different MC methods in three typical cases with SVD.

### 5.3. Dynamic evaluation of variance and relative trace error

The above results emphasize the importance of being able to assess quickly and accurately the relative differences between the variances of different methods as well as the trace error, so that we can decide whether to continue with fitting or which MC method to switch to. First we show the effectiveness of the dynamic variance evaluation algorithm for our fitting MC method on $(D - p(M))$ with unit vectors, and on $A^{-1}$ and $E$ with Rademacher vectors. Then, we evaluate our algorithm for estimating the relative trace error during the fitting process.

Figure 12 compares the estimated variances with the actual variances of the three MC methods when increasing the number of fitting points from 5 to 100. The approximation $M$ is computed by using ILU. We can see that the estimated values of $Var(T_{Z_2}(A^{-1}))$ and $Var(T_{Z_2}(E))$ converge to the actual variances after only a few sample points. The estimated value of $Var(T_{e_i}(E_{fit}))$ gets close to and captures the trend
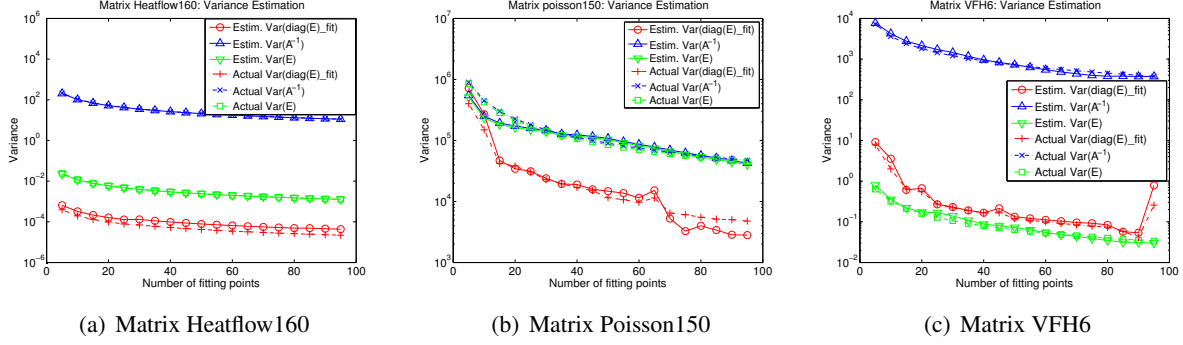
(a) Matrix Heatflow160     (b) Matrix Poisson150     (c) Matrix VFH6

Figure 12: Comparing estimated variances and actual variances of three MC methods with ILU.



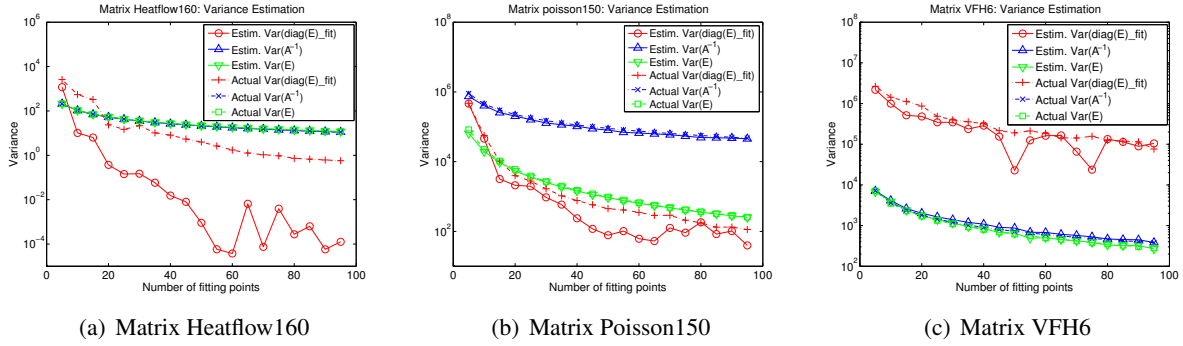(a) Matrix Heatflow160     (b) Matrix Poisson150     (c) Matrix VFH6

Figure 13: Comparing estimated variances and actual variances of three MC methods with SVD.

of the the actual variance as the fitting samples increase. Nevertheless, the relative differences between the variances of the various MC methods are apparent almost immediately.

Figure 13 shows the same experiments when the approximation $M$ is computed by SVD. Since $M$ is updated each step, $Var(T_{Z_2}(E))$ and $Var(T_{e_i}(E_{fit}))$ change accordingly. As with ILU, $Var(T_{Z_2}(A^{-1}))$ and $Var(T_{Z_2}(E)$ can be estimated very well in a few steps. $Var(T_{e_i}(E_{fit}))$ could be underestimated but the relative variance difference between these MC methods becomes clear when the fitting points increase beyond 20. Thus we are able to determine whether the fitting process is beneficial as a variance reduction preprocessing and which is the best MC method to proceed with for the trace estimation.

Figure 14 compares the estimated relative trace error with the actual one in the cases of Figure 13. We observe that the estimation is accurate as the fitting samples increase, even for cases such as VFH6 where
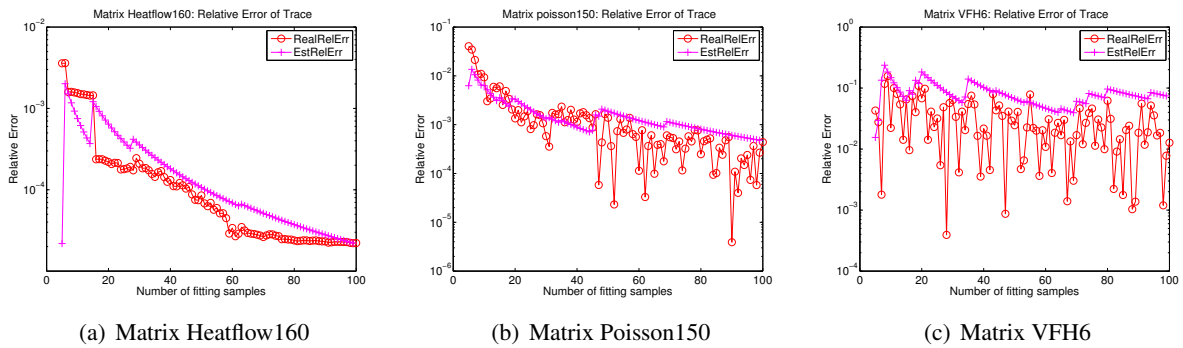


(a) Matrix Heatflow160     (b) Matrix Poisson150     (c) Matrix VFH6

Figure 14: Comparing estimated relative trace error with actual relative trace error with SVD.

16

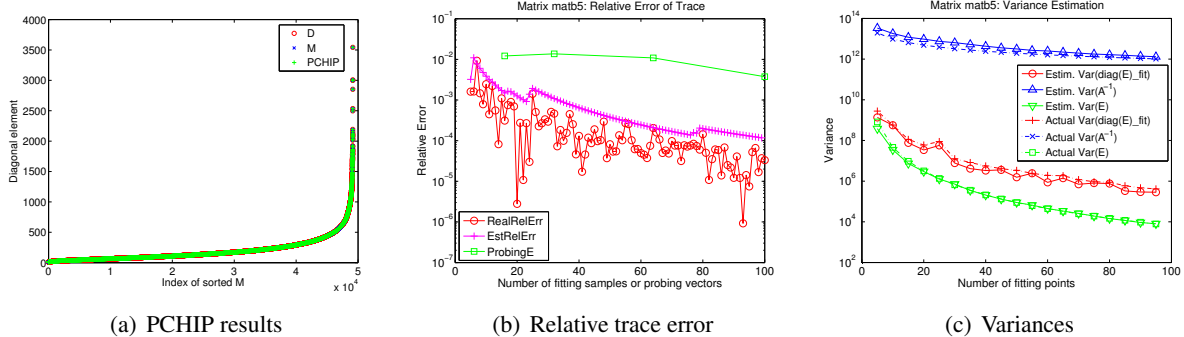(a) PCHIP results       (b) Relative trace error       (c) Variances

Figure 15: Fitting results, dynamic evaluation of relative trace error and variances with SVD on a large QCD matrix. In Figure 15(b), the green square denotes the relative trace error by applying hierarchical probing technique on deflated matrix $E$ in [4].

the fitting process is not as successful. Moreover, because our algorithm is based on upper bounds on the error of a piecewise cubic polynomial, the actual relative trace error could be lower than predicted.

### 5.4. A large QCD problem

The trace estimator presented in this paper has the potential of improving a number of LQCD calculations, where the trace of the Dirac matrix is related to an important property of QCD called spontaneous chiral symmetry breaking [19]. In our previous work [4], we presented the method of hierarchical probing that achieves almost optimal variance reduction incrementally and inexpensively.

As shown in Figure 15(a), a low rank approximation with 200 singular vectors yields a good approximation $M$ and an excellent fit $p(M)$. In Figure 15(b), we see that the actual relative trace error decreases very fast to $O(10^{-4})$ with increasing number of fitting points and singular vectors, and can be monitored well by our dynamic trace estimation algorithm. These singular vectors can be approximated while solving the linear systems with eigCG. Interestingly, we can improve the relative trace error of hierarchical probing by two orders of magnitude. In addition, the variances of different MC methods can be estimated dynamically to allow us to continue with the best estimator if needed (Figure 15(c)).

## 6. Conclusion and future work

A novel method has been presented to estimate the trace of the matrix inverse by exploiting the pattern correlation between the diagonal of the inverse of the matrix and some approximation. The key idea is to construct a good approximation $M \approx D$ through eigenvectors or some preconditioner, sample important patterns of $D$ by using the distribution of the elements of $M$, and use fitting techniques to obtain a better approximation $p(M) \approx D$ from where we obtain a trace estimate. The proposed method can provide a fast trace estimate with 2-3 digits relative accuracy given only a few samples while may or may not improve the variance of MC. When the variance is reduced sufficiently, our method can be also used as a diagonal estimator. We also propose an effective dynamic variance evaluation algorithm to determine the MC method with the smallest variance and a dynamic relative trace error estimation algorithm without any additional costs. We demonstrated the effectiveness of these methods through a set of experiments in some real applications.

# References

[1] M. F. Hutchinson, A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines, Communications in Statistics-Simulation and Computation 19 (2) (1990) 433–450.

[2] C. Bekas, A. Curioni, I. Fedulova, Low cost high performance uncertainty quantification, in: Proceedings of the 2nd Workshop on High Performance Computational Finance, ACM, 2009, p. 8.

[3] V. Kalantzis, C. Bekas, A. Curioni, E. Gallopoulos, Accelerating data uncertainty quantification by solving linear systems with multiple right-hand sides, Numerical Algorithms 62 (4) (2013) 637–653.

[4] A. Stathopoulos, J. Laeuchli, K. Orginos, Hierarchical probing for estimating the trace of the matrix inverse on toroidal lattices, SIAM Journal on Scientific Computing 35 (5) (2013) S299–S322.

[5] H. Avron, Counting triangles in large graphs using randomized matrix trace estimation, in: Workshop on Large-scale Data Mining: Theory and Applications, 2010.

[6] H. Avron, S. Toledo, Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix, Journal of the ACM (JACM) 58 (2) (2011) 8.

[7] Z. Bai, G. Fahey, G. Golub, Some large-scale matrix computation problems, Journal of Computational and Applied Mathematics 74 (1) (1996) 71–89.

[8] I. S. Duff, A. M. Erisman, J. K. Reid, Direct methods for sparse matrices, Clarendon press Oxford, 1986.

[9] J. M. Tang, Y. Saad, Domain-decomposition-type methods for computing the diagonal of a matrix inverse, SIAM Journal on Scientific Computing 33 (5) (2011) 2823–2847.

[10] H. Guo, Computing traces of functions of matrices, A Journal of Chinese Universities (English series) 2 (2000) 204–215.

[11] G. Meurant, Estimates of the trace of the inverse of a symmetric matrix using the modified chebyshev algorithm, Numerical Algorithms 51 (3) (2009) 309–318.

[12] C. Brezinski, P. Fika, M. Mitrouli, Moments of a linear operator, with applications to the trace of the inverse of matrices and the solution of equations, Numerical Linear Algebra with Applications 19 (6) (2012) 937–953.

[13] M. N. Wong, F. J. Hickernell, K. I. Liu, Computing the trace of a function of a sparse matrix via hadamard-like sampling, Tech. rep., Department of Mathematics, Hong Kong Baptist University (2004).

[14] C. Bekas, E. Kokiopoulou, Y. Saad, An estimator for the diagonal of a matrix, Applied numerical mathematics 57 (11) (2007) 1214–1229.

[15] F. Roosta-Khorasani, U. Ascher, Improved bounds on sample size for implicit matrix trace estimators, Foundations of Computational Mathematics (2014) 1–26.

[16] J. M. Tang, Y. Saad, A probing method for computing the diagonal of a matrix inverse, Numerical Linear Algebra with Applications 19 (3) (2012) 485–501.

[17] A. Erisman, W. Tinney, On computing certain elements of the inverse of a sparse matrix, Communications of the ACM 18 (3) (1975) 177–179.

[18] L. Wu, A. Stathopoulos, Primme_svds: A preconditioned svd solver for computing accurately singular triplets of large matrices based on the primme eigensolver, Tech. rep., Department of Computer Science, College of William and Mary (2014).

[19] A. Stathopoulos, K. Orginos, Computing and deflating eigenvalues while solving multiple right-hand side linear systems with an application to quantum chromodynamics, SIAM Journal on Scientific Computing 32 (1) (2010) 439–462.

[20] A. M. Abdel-Rehim, A. Stathopoulos, K. Orginos, Extending the eigcg algorithm to nonsymmetric lanczos for linear systems with multiple right-hand sides, Numerical Linear Algebra with Applications 21 (4) (2014) 473–493.

[21] P. D. Robinson, A. J. Wathen, Variational bounds on the entries of the inverse of a matrix, IMA journal of numerical analysis 12 (4) (1992) 463–486.

[22] D. J. MacKay, Introduction to monte carlo methods, in: Learning in graphical models, Springer, 1998, pp. 175–204.

[23] J. Kiefer, Optimum sequential search and approximation methods under minimum regularity assumptions, J. Soc. Indust. Appl. Math. 5 (1957) 105–136.

[24] E. Novak, Quadrature formulas for monotone functions, Proceedings of the American Mathematical Society 115 (1) (1992) 59–68.

[25] A. G. Sukharev, The concept of sequential optimality for problems in numerical analysis, Journal of Complexity 3 (3) (1987) 347–357. doi:http://dx.doi.org/10.1016/0885-064X(87)90020-3.

[26] F. N. Fritsch, R. E. Carlson, Monotone piecewise cubic interpolation, SIAM Journal on Numerical Analysis 17 (2) (1980) 238–246.

[27] F. E. Harrell, Regression modeling strategies, Springer Science & Business Media, 2001.

[28] S. Arlot, A. Celisse, et al., A survey of cross-validation procedures for model selection, Statistics surveys 4 (2010) 40–79.

[29] T. Gudmundsson, C. S. Kenney, A. J. Laub, Small-sample statistical estimates for matrix norms, SIAM journal on matrix analysis and applications 16 (3) (1995) 776–792.

[30] C. Kenney, A. J. Laub, M. Reese, Statistical condition estimation for linear systems, SIAM Journal on Scientific Computing 19 (2) (1998) 566–583.

[31] T. R. Lucas, Error bounds for interpolating cubic splines under various end conditions, SIAM Journal on Numerical Analysis 11 (3) (1974) 569–584.

[32] T. A. Davis, Y. Hu, The university of florida sparse matrix collection, ACM Transactions on Mathematical Software (TOMS) 38 (1) (2011) 1.