DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Interdisciplinary Project in Informatics

Generalization and Parallelization of Sherman-Morrison System Matrix Updates for Sparse Grid Density Estimation

Dmitrij Boschko

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Interdisciplinary Project in Informatics

Generalization and Parallelization of Sherman-Morrison System Matrix Updates for Sparse Grid Density Estimation

Author:	Dmitrij Boschko
Supervisor:	Prof. Dr. Hans-Joachim Bungartz
Advisors:	Kilian Röhner, Michael Obersteiner
Submission Date:	04.11.2019

I confirm that this interdisciplinary project in informatics is my own work and I have documented all sources and material used.

Munich, 04.11.2019 Dmitrij Boschko

Abstract

Sherman-Morrison rank-one updates have been used successfully for adaptive sparse grid density estimation. This allowed for regularization and adaptivity, but until now, this has only been possible in the offline/online splitting context using an orthogonal decomposition, such as tridiagonal. The new approach studied in this paper generalizes the old version of the Sherman-Morrison formula based sparse grid density estimation by using the Sherman-Morrison-Woodbury (SMW) formula. This allows for arbitrary decompositions in the offline phase and better supports parallelization by allowing for simultaneous adaptivity operations (rank-k updates), while keeping all features, such as regularization in between off/on phases. A parallelized/distributed version of the new SMW approach has been implemented and evaluated against the old version, and shows enhancements in terms of speed, while keeping the accuracy. It scales well for reasonably sized rank-k updates.

Contents

1 Introduction and Motivation

Sparse-grid based machine learning approaches have been shown to deliver good results in terms of processing high-dimensional data. While a full grid of N data points, with d features each, contains N^d values to be processed, a sparse grid consists only of a certain subset. When discretizing $N = 2^n$ data points with discretization level n, sparse grids allow a complexity reduction of processed data from $\mathcal{O}(2^{nd})$ to $\mathcal{O}(2^n n^{(d-1)})$.
In the context of online learning, sparse grid density estimation (SGDE) techniques have been developed which make use of splitting computation in offline and online phases, which allows to relocate complex computations into the offline phase.
In [Bos17], a sparse-grid density estimation algorithm was introduced, which uses an orthogonal decomposition in the offline phase, and performs refinement and coarsening in the online phase by making use of the Sherman-Morrison formula. This algorithm is able to

- compute a probability density function while maintaining sparse grid structure,
- perform refinement and coarsening,
- split the procedure in an offline phase, and a fast online phase,
- perform efficient λ-regularization right after the offline phase, and
- efficiently solve of the resulting system of linear equations.

While analyzing the run-time and function-call distribution, it became apparent that matrix-vector multiplication is a bottleneck. In order to tackle this problem, parallelizing and distributing the algorithm on a cluster is considered.
To make full use of the advantages of parallelization and distribution, a generalized version of the Sherman-Morrison formula, the Sherman-Morrison-Woodbury (SMW) formula, is used for the online phase. This allows for simultaneous refining and/or coarsening of more than one point at a time, while mainly processing matrix-matrix multiplications, which are more suited for parallelization than matrix-vector multiplications.
This paper summarizes the necessary mathematical background (Chapter 2) of sparse grids in order to dive into the theory (Chapter 3) and implementation (Chapter 4) of new SMW based approach. Additionally, the implementation is evaluated (Chapter 5).

2 Theoretical Background

This chapter briefly summarizes necessary mathematical background in order to understand the SMW algorithm. A introduction to sparse grids is not given, but can be found in [Bos17]. For a more elaborate description of the sections, refer to [Bos17] and the references therein.
Unless explicitly stated, vectors will be lower-case letters with an arrow $\vec{b}$ and their components will be indexed lower-case letters b_{i}. Matrices will be big-letters A, while its entries are indexed big-letters A_{ij}, i and j being the number of the row and column, respectively.

2.1 Sparse Grid Density Estimation

For given data $\mathcal{S} = \{\vec{x}_1, \dots, \vec{x}_N\} \subset \mathbb{R}^d$, where each $vecx_i$ is a vector of features, the goal of density estimation is to find a probability density function $\hat{p}$ in a function space $\mathcal{F}$, such that it approximates the real underlying probability density f of the data points. This can be done with a spline fitting approach

$$\hat{p} = \arg\min_{f \in \mathcal{F}} \int_{\Omega} (f(\vec{x}) - p_\epsilon(\vec{x}))^2 \mathrm{d}\vec{x} + \lambda \|\Lambda f\|_{L^2}^2,$$

where p_ϵ is an overfitted initial guess, $\lambda \|\Lambda f\|_{L^2}^2$ a regularization term, and $\|\cdot\|_{L^2}^2$ the norm in the function space $\mathcal{F}$. The task of finding such a $\hat{p}$ can be transformed into the task of solving a system of linear equations. Let $n \in \mathbb{N}$ be the number of basis functions φ_{i} needed to uniquely represent the underlying probability density f using sparse grids. With $R_{\mathrm{ij}} = \langle \varphi_{\mathrm{i}}, \varphi_{\mathrm{j}} \rangle_{L^2}$ and $b_{\mathrm{i}} = \frac{1}{N} \sum_{\mathrm{j}=1}^{N} \varphi_{\mathrm{i}}(x_{\mathrm{j}})$, the above problem is also solved by finding $\vec{\alpha}$ via solving

$$(R + \lambda I)\vec{\alpha} = \vec{b}\ .$$

Using that $\vec{\alpha}$, the approximated probability function $\hat{p}$ can be computed by

$$\hat{p}(\vec{x}) = \sum_{\mathrm{i}=1}^{N} \alpha_{\mathrm{i}} \varphi_{\mathrm{i}}(\vec{x}).$$

For a more elaborate derivation of the system of linear equations and more theoretical background on sparse grids, refer to [Peh13] and the references therein.

2.2 Offline/Online Splitting

With the goal of solving $(R + \lambda I)\vec{\alpha} = \vec{b}$ whenever needed, the right hand side can completely be constructed given the sparse grid and its basis functions. This is used in order to prepare the left hand side for quick solving whenever new data is available and results in changes to $\vec{b}$. The offline phase consists of decomposing $(R + \lambda I)$. In principle, any matrix decomposition technique can be used here, e.g. Cholesky decomposition, but depending on the one chosen, refining and/or coarsening may need special handling during online phase. In [Sie16] a Cholesky based SGDE approach is studied, which has dedicated offline and online phase implementations in order to handle the Cholesky decomposition $(R + \lambda I) = LL^T$. Utilizing tridiagonal decomposition $(R + \lambda I) = QTQ^T$ for the offline phase and Sherman-Morrison formula for the online phase, in [Bos17], an approach is introduced which expands the feature set of the Cholesky approach by additionally allowing for efficient λ-regularization right after the offline phase.

2.3 Sherman-Morrison-Woodbury (SMW) Formula

In [Hag89], a formula for rank-k updates is discussed, here referred to as Sherman-Morrison-Woodbury (SMW) formula. For $n, k \in \mathbb{N}\backslash\{0\}$, let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix, $U, V \in \mathbb{R}^{n\times k}$ matrices of rank k, then

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1}U)^{-1}V^T A^{-1}.$$

It is important to notice, that A has to be invertible for the formula to hold and both U and V have to be of rank k in order for the term $(I + V^T A^{-1}U)$ to be invertible.
Numerically, the inverse of A has to be known, and the inverse of $(I + V^T A^{-1}U)$ has to be computed, which can be computationally expensive. In the context of SGDE refining/coarsening, the initial sparse grid often is small compared to the resulting refined end grid. Also, for every batch of refined/coarsened grid points, the resulting matrix $(I + V^T A^{-1}U)$, which has to be inverted, is of the same size as the batch. Thus, by controlling the amount of grid points to refine/coarsen, the matrix can be inverted in reasonable time.
Let U_1, $V_1 \in \mathbb{R}^{n\times k}$ and U_2, $V_2 \in \mathbb{R}^{n\times k}$ be the matrices of two simultaneous rank-k-updates,

$$(A + U_1V_1^T + U_2V_2^T).$$

Without loss of generality, the update corresponding to U_1 and V_1 shall be considered first,

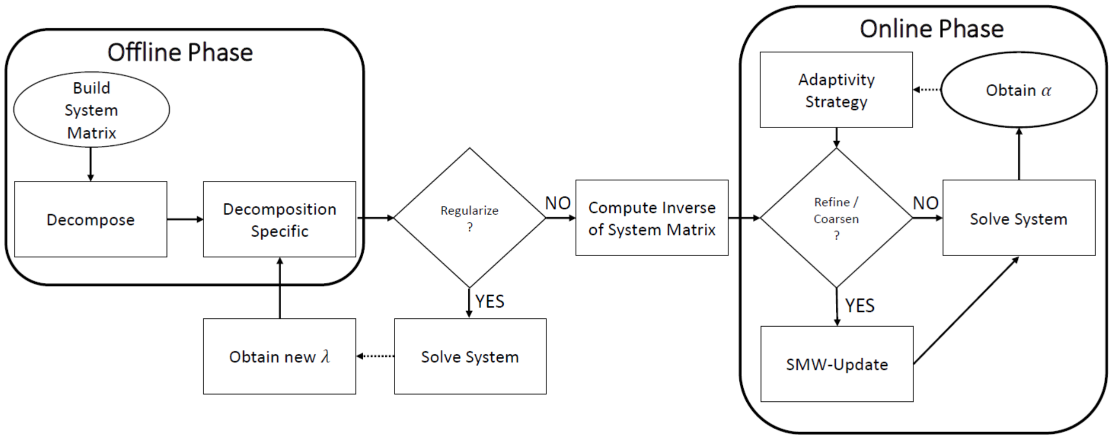
$$(A + U_1 V_1^T)^{-1} = A^{-1} - A^{-1} U_1 (I + V_1^T A^{-1} U_1)^{-1} V_1^T A^{-1}.$$

Let $\tilde{B}$ denote the computed inverse, i.e. $\tilde{B} := (A + U_1 V_1^T)^{-1}$, then, again w.l.o.g. the second update is

$$B := ((A + U_1 V_1^T) + U_2 V_2^T)^{-1} = \tilde{B} - \tilde{B} U_2 (I + V_2^T \tilde{B} U_2)^{-1} V_2^T \tilde{B}.$$

Therefore, it is sufficient to keep a matrix B, initialized as the zero matrix, overwrite it after every rank-k-update, and use it as the inverse of the base matrix for the next SMW update.

3 Algorithm



In this chapter, the procedure of the SMW based algorithm for offline/online sparse grid density estimation is explained. This includes the step-by-step procedure, runtime analysis of each step, as well as explanations why it is working.
Provided the level and index of the sparse grid of size $n \in \mathbb{N}\backslash\{0\}$, the algorithm's rough steps are:

1. Compute offline matrix R of initial grid R_{ij}, $\mathrm{i}, \mathrm{j} \in \{1, 2, \ldots, n\}$
2. Decompose offline matrix
3. Optional: Perform λ-regularization
4. Compute inverse of regularized offline matrix $(R + \lambda I)^{-1}$
5. Build refinement/coarsening matrices X, E
6. Perform SMW updates $B = ((R + \lambda I) + XE^T + EX^T)^{-1}$
7. Solve resulting system of linear equations $\vec{\alpha} = B\vec{b}$

3.1 Offline Phase

3.1.1 Initialization

With the data fixing dimension of the sparse grid, after a level is chosen, the total size of the initial grid is fixed and will be denoted with $n \in \mathbb{N}_{>0}$. With a fixed sparse grid, each basis function φ_i, i$\in \{1, 2, \ldots, n\}$ can be computed. With that, the representing $n \times n$ system matrix can be built,

$$R = \left(\langle \varphi_\mathrm{i}, \varphi_\mathrm{j} \rangle_{L^2}\right)_{\mathrm{i,j} \in \{1,2,\ldots,n\}}.$$

Due to the properties of the L^2-inner-product, this matrix is symmetric and positive definite.

3.1.2 Decomposition

The new SMW-based approach is compatible with any decomposition. In fact, it is even possible to perform no decomposition at all, as only the explicitly calculated inverse offline matrix is needed, regardless of how it is obtained.
Nevertheless, in the context of offline/online sparse grid density estimation, it is desirable to decompose the offline matrix, so it supports features, such as regularization. As the current standard decomposition for the SMW-based approach, tridiagonal decomposition is chosen, as it supports fast regularization and is invertible in reasonable time. The tridiagonal decomposition of a symmetric positive-definite real matrix R consists of two matrices Q and T, such that

$$R = QTQ^T,$$

with Q being orthogonal, and T being a tridiagonal matrix. Since R is symmetric, T is guaranteed to be symmetric also.
The run-time of tridiagonal decomposition is known to be $\mathcal{O}(n^3)$.

3.2 In-Between Phase

3.2.1 Regularization

Using tridiagonal decomposition, it is possible to perform λ-regularization. Given R and a right-side vector $\vec{b}$, find $\lambda \in \mathbb{R}$, such that the solution $\vec{\alpha}$ to the linear system of equations $(R + \lambda I)\vec{\alpha} = \vec{b}$ maximizes a chosen performance measure.
After R is decomposed, an arbitrary λ can be evaluated by solving

$$Q(T + \lambda I)Q^T\vec{\alpha} = \vec{b}.$$

This is equivalent to the initial system matrix $R + \lambda I$, due to

$$\begin{aligned}
&Q(T + \lambda I)Q^T && \text{(matrix distributive property)}\\
&= QTQ^T + Q(\lambda I)Q^T && \text{(mult. with scalar commutate)}\\
&= QTQ^T + \lambda IQQ^T && \text{(Q orthogonal)}\\
&= R + \lambda I.
\end{aligned}$$

Testing another λ does not require to decompose anew, but only solving the system again using T, Q and $\vec{b}$.
While solving, matrix-vector multiplications with Q have to be performed, which can be done in $\mathcal{O}(n^2)$. Additionally, solving a tridiagonal system is also $\mathcal{O}(n^2)$, and thus, the complete system can be solved in $\mathcal{O}(n^2)$. Finding an optimal λ therefore is in $\mathcal{O}(l \cdot n^2)$, with l denoting the amount of λ evaluated.

3.2.2 Inversion

After a suited λ is chosen, it is advantageous for SMW to explicitly compute the inverse of the resulting system matrix $(R + \lambda I)$. Using the already computed tridiagonal decomposition, this can be done fast by using the shown identity $R + \lambda I = Q(T + \lambda I)Q^T$ and the orthogonality of Q.

$$\begin{aligned}
(R + \lambda I)^{-1} &= (Q(T + \lambda I)Q^T)^{-1}\\
&= (Q^T)^{-1}(T + \lambda I)^{-1}Q^{-1}\\
&= Q(T + \lambda I)^{-1}Q^T
\end{aligned}$$

Therefore, it is only necessary to invert $(T + \lambda I)$, which is of tridiagonal shape, and thus can be inverted in $\mathcal{O}(n^2)$. Note, that the inverse is a full matrix in general, even though it originated from a tridiagonal shape.
As stated previously, any decomposition can be used, as long as $(R + \lambda I)^{-1}$ can be computed explicitly.

3.3 Online Phase

3.3.1 Building Refinement/Coarsening Matrix

The idea of SMW-based sparse grid density estimation is using matrix rank-updates, so that the grid matrix R is augmented by rows/columns, which correspond to the added grid-points. For example, adding an $n+1$-th grid-point to a n-sized grid results in

$$R = \begin{pmatrix} \langle \varphi_1, \varphi_1 \rangle & \cdots & \langle \varphi_1, \varphi_n \rangle \\ \vdots & \ddots & \vdots \\ \langle \varphi_n, \varphi_1 \rangle & \cdots & \langle \varphi_n, \varphi_n \rangle \end{pmatrix} \overset{\text{refine}}{\longrightarrow} \begin{pmatrix} & & & \langle \varphi_1, \varphi_{n+1} \rangle \\ & R & & \vdots \\ & & & \langle \varphi_n, \varphi_{n+1} \rangle \\ \langle \varphi_{n+1}, \varphi_1 \rangle & \cdots & \langle \varphi_{n+1}, \varphi_n \rangle & \langle \varphi_{n+1}, \varphi_{n+1} \rangle \end{pmatrix}.$$

Similarly, removing grid-points results in removing the corresponding rows/columns from the matrix. Let $k \in \mathbb{N}_{>0}$ be the amount of effectively added grid-points, then $(R + \lambda I)$ has to be resized to a $(n+k)$-sized matrix. Because resizing and filling the new entries with zeroes would result in a singular matrix, the new diagonal part shall be filled with ones instead. This way, the inverse of the resized matrix exists, and can be built from the inverse of the old one.
From now on, let A denote the $(n+k) \times (n+k)$ shaped matrix, which contains the old system matrix $(R + \lambda I)$ as an $n \times n$ sub-matrix, filled with ones as the new diagonal part, and filled with zeroes else, i.e.

$$A = \begin{pmatrix} \langle \varphi_1, \varphi_1 \rangle + \lambda & \cdots & \langle \varphi_1, \varphi_n \rangle & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \langle \varphi_n, \varphi_1 \rangle & \cdots & \langle \varphi_n, \varphi_n \rangle + \lambda & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 \end{pmatrix}.$$

One rank-k-update augments the new system matrix A by either added rows or added columns, depending on the choice of U and V. Thus, two rank-k-updates have to be performed, to add both, rows *and* columns, to the initial matrix, i.e.

$$(A + U_1 V_1^T + U_2 V_2^T),$$

with $U_1, V_1, U_2, V_2 \in \mathbb{R}^{(n+k) \times k}$.
To preserve symmetry, the added row should be equal to the transposed of the added column, and vice versa. Therefore, the U and V should be chosen accordingly. W.l.o.g., let these specific choices be denoted with $U_1 = X$ and $V_1 = E$. For the symmetry to consist, it has to hold that $U_2 = E$ and $V_2 = X$, and therefore the updates correspond to

$$(A + XE^T + EX^T).$$

When refining, the matrices X and E are chosen in a way, so that they augment the rows/columns of A by L^2-inner-products of basis functions. The elements of X corresponding to diagonal elements of A have to be adapted in order to account for λ, the previously added one on the diagonal of A, and the fact, that it is added two times (by previously adding the row *and* column respectively). Therefore, such an entry x has to be substituted by $0.5 \cdot (x - 1 + \lambda)$, resulting in the matrix

$$X = \begin{pmatrix} \langle \varphi_1, \varphi_{\text{n+1}} \rangle & \cdots & \cdots & \langle \varphi_1, \varphi_{\text{n+k}} \rangle \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ 0.5 \cdot (\langle \varphi_{\text{n+1}}, \varphi_{\text{n+1}} \rangle - 1 + \lambda) & & & \vdots \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & 0.5 \cdot (\langle \varphi_{\text{n+k}}, \varphi_{\text{n+k}} \rangle - 1 + \lambda) \end{pmatrix} \in \mathbb{R}^{(n+k)\times k}.$$

When coarsening, the matrix X differs only from the refining case, as in that it is identical in rows/columns of indices of grid-points, which are to be coarsened. The rest of the rows/columns just carry a one at the diagonal entry, and zeroes anywhere else. The matrix E simply corresponds to the operation of choosing according rows/columns, therefore has only ones at the same positions, where entries of X have been adapted previously, i.e.

$$E = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ 1 & & & \vdots \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{pmatrix} \in \mathbb{R}^{(n+k)\times k}.$$

To conclude, for both X and E, the irregular elements are those on the index-pairs (i, j) $\in \{$(n+1, 1), $\ldots$, (n+k, k)$\}$. To choose between refining and coarsening, E can be manipulated by multiplying it with $(+1)$ for refining and (-1) for coarsening, however, X has to be built accordingly when coarsening.

3.3.2 Performing SMW Updates

After adapting A and obtaining X and E, the SMW formula can be directly computed. If k grid-points are to be refined/coarsened, the complete SMW-update consists of two rank-k updates, i.e.

$$(A + XE^T + EX^T).$$

As shown in section 2.3, the rank-k-updates can be performed consecutively. W.l.o.g., the first update shall be $A + XE^T$, i.e.

$$\tilde{B} := (A + XE^T)^{-1} = A^{-1} - A^{-1}X(I + E^T A^{-1} X)^{-1} E^T A^{-1}.$$

After $\tilde{B}$ is computed, it can be used as the explicitly computed inverse needed for the SMW-formula of the second update, i.e.

$$B := ((A + XE^T) + EX^T)^{-1} = \tilde{B} - \tilde{B}E(I + X^T \tilde{B} E)^{-1} X^T \tilde{B}.$$

As B effectively is the inverse of a symmetric matrix, it is symmetric itself. In contrast, $\tilde{B}$ is not symmetric.
Since $\tilde{B}$ and B are not dependent of each other, both can be saved inside the matrix B. If additional refining/coarsening is performed, B always serves as the explicitly computed inverse needed in the SMW-formula. Therefore, arbitrary many SMW-updates can be performed.
Let $m \in \mathbb{N}_{>0}$ be the size of the current matrix B, which has been refined previously, i.e. $m \geq n$. The total worst-case run-time then is in $\mathcal{O}(m^2 k + k^3)$, as inverting the middle-term is $\mathcal{O}(k^3)$, and some of the matrix-matrix multiplications are $\mathcal{O}((m+k)^2 k)$. However, if k is kept reasonably small, the dominating complexity is $\mathcal{O}(m^2 k)$.

3.3.3 Solving the System

After refining/coarsening, the computed inverse $B \in \mathbb{R}^{m \times m}$ can be directly used to solve the original system

$$(A + XE^T + EX^T)\vec{\alpha} = \vec{b},$$

by computing

$$\vec{\alpha} = (A + XE^T + EX^T)^{-1}\vec{b} = B\vec{b}.$$

As this process consist of a single matrix-matrix multiplication, the run-time complexity is in $\mathcal{O}(m^2)$, m being the final grid-size.

4 Implementation

One of the main parts of the interdisciplinary project was to implement the new ideas into **SG++**[1]. This section includes a list of the functionalities and structures implemented. Besides the interaction between the single classes, each functionality is also integrated into the new datadriven pipeline, which is discussed in [Fuc18]. These integrations include changes in the factory classes, in the sgde pipeline classes (such as ModelFittingDensityEstimationOnOff), and in the new classes' parents.

- New decomposition types **SMW_ortho** and **SMW_chol**
  put offline/online sparse grid density estimation into effect. This is achieved by using tridiagonal and cholesky decomposition as offline phase, respectively, and then using the new SMW approach for the online class, together with a solver. The new decomposition types are integrated into the pipeline and can be chosen inside the configuration .json-files, under densityEstimationConfig for matrixDecompositionType.

- New class **DBMatOnlineDE_SMW**
  implements the main functionalities of the new SMW-based approach. These include refinement and coarsening, as well as building the necessary matrices for it. Given an explicitly computed inverse from the offline object, DBMatOnlineDE_SMW is able to execute SMW-updates independent of the offline's decomposition type. Other types than tridiagonal or cholesky decomposition can easily be added, by providing a new constant for it, analogous to SMW_ortho or SMW_chol. The following is a list of implemented functions and a broad description of each.

  – Compute_L2_refine_matrix
    computes the matrix $X \in \mathbb{R}^{(n+k)\times k}$ discussed in section 3.3.1 by explicitly calculating the L^2-products for the corresponding rows/columns. The built vectors are saved in order to be able to later coarsen corresponding grid-points without recalculating the L^2-products.

[1]**SG++** is a project developed at the chair for scientific computing at the Technical University of Munich and at the chair for simulation of large systems at the University of Stuttgart, sgpp.sparsegrids.org.

- Compute_L2_coarsen_matrix
  builds the matrix $X \in \mathbb{R}^{(n+k\times k)}$ out of vectors of L^2-product saved from previous refinements, see above and section 3.3.1.
- Smw_adapt
  implements the SMW formula for refinements/coarsenings, making use of GSL[2] operations and structures. One call of smw_adapt executes two rank-k-updates, i.e.

  $$B := B - BX(I + E^T BX)^{-1} E^T B,$$

  and

  $$B := B - BE(I + X^T BE)^{-1} X^T B.$$

  After coarsening, the matrix is reduced to eliminate appeared empty rows/-columns.
- Smw_adapt_parallel
  is the parallel/distributed version of smw_adapt. It executes the same two rank-k-updates, but uses ScaLAPACK[3]-based distributed matrix and vector operations implemented in [Sch19]. Furthermore, some direct calls to ScaLAPACK-routines are made, e.g. to invert the $k \times k$ matrix inside the SMW formula.
- Other functions
  include updateSystemMatrixDecomposition, solveSLE, their parallel versions, as well as other less complicated functions in order to implement inheritance.

• New class **DBMatDMS_SMW**
is a dedicated solver for the SMW-based approach, which simply performs a matrix-vector multiplication in order solve the underlying system of linear equations,

$$\vec{\alpha} = B \cdot \vec{b}.$$

• *Additional, non-SMW-related functionality*
includes the support for modlinear basis functions of the sparse grid, which can be configured in the .json configuration file. Furthermore, offline matrix decompositions (tridiagonal and cholesky) have been parallelized with the help of routines from ScaLAPACK.

[2]https://www.gnu.org/software/gsl
[3]https://www.netlib.org/scalapack/

5 Evaluation

In offline/online sparse grid density estimation algorithms, run-times for offline and online phase often are dependent of the decomposition type. Using the Sherman-Morrison-Woodbury formula for the online phase, however, not only allows for arbitrary decomposition types, but is also independent in terms of run-time in the online phase.
Refining and coarsening with the new SMW-based approach share the same mathematical operations, thus have the same run-time. Although the accuracy of sparse grid density estimation depends on data in general, the run-time itself only depends on the amount of grid-points, or equivalently, the size of the resulting system of linear equations. Therefore, run-time evaluation can be performed with no data, but only with arbitrary matrices of varying size, which suffice the conditions of the base matrix of the sparse grid, i.e. symmetric and positive definite. This is done by constructing a real quadratic matrix A of size $n \in \mathbb{N}_{>0}$, filling it with random values $A_{\text{ij}} \in [0, 1]$, getting a matrix B by multiplying A with its transpose,

$$B := AA^T.$$

In general, AA^T is only semi-definite. By using tridiagonal decomposition, however, this can be detected, A discarded, and a new A generated. This can be done repeatedly, until AA^T is positive definite. The symmetry of AA^T is guaranteed, as

$$B^T = (AA^T)^T = (A^T)^T A^T = AA^T = B.$$

All following run-time measurements are performed using an Intel Core i7-4710HQ CPU with 2.5GHz frequency, 4-cores, and hyper-threading support.

5.1 Offline Phase

The offline phase run-time of the SMW-based approach is entirely dependent on the chosen decomposition type. It consists of building the initial grid's system matrix $R \in \mathbb{R}^{n \times n}$, as well as decomposing it. Using tridiagonal decomposition for example, the decomposition $R = QTQ^T$ is part of the offline phase. This is no novelty of the SMW-based approach, and a run-time analysis can be found in [Bos17].

5.2 In-Between Phase

For the λ-regularization, the right hand side $\vec{b}$ of the system of linear equations is needed, which depends on data. Again, the computation entirely depends on the decomposition type chosen. For example, with tridiagonal decomposition, the system of linear equations

$$(R + \lambda I)\vec{\alpha} = Q(T + \lambda I)Q^T\vec{\alpha} = \vec{b}$$

has to be solved for every λ to be tested. The run-time complexity is in $\mathcal{O}(l \cdot n^2)$, where l is the amount of tested λ and n is the systems size. For an analysis of this, see [Bos17]. When a λ is chosen, the inverse of the system matrix has to be computed explicitly for the SMW-approach. For tridiagonal decomposition type, this is done by computing

$$(R + \lambda I)^{-1} = Q(T + \lambda I)^{-1}Q^T.$$

This whole operation consists of inverting $(T + \lambda I)$, which is $\mathcal{O}(n^2)$, and two $n \times n$ matrix-matrix multiplications. It is possible to implement matrix-matrix multiplications with orthogonal matrices in $\mathcal{O}(n^2)$, which makes this whole procedure of explicitly inverting a tridiagonal decomposed matrix $\mathcal{O}(n^2)$.

5.3 Online Phase

With a fixed λ and an explicitly computed inverse $Q(T + \lambda I)^{-1}Q^T$, refinement and coarsening are prepared for. In the SMW-based approach, both refining as well as coarsening use the same operations and subroutines, therefore evaluation is only done on refining. Since the complexity of one rank-k-update is in $\mathcal{O}(m^2k + k^3)$, with $m = n + k$ as seen in section 3.3.2, the parameters to be analyzed are the initial grid-size n, and the amount k of grid-points to be refined.
Figure 5.1 shows the dependency of run-time of SMW-updates and the initial grid-size for different refinement batches of size $k = 100$, which means the simultaneous refinement of 100 grid-points. It can be seen, that the new SMW-based approach outperforms k-times refining one grid-point with the old Sherman-Morrison refinements. Parallelizing the SMW-based approach gives additional boost, although not optimally due to not optimized synchronization of matrices in the current implementation.

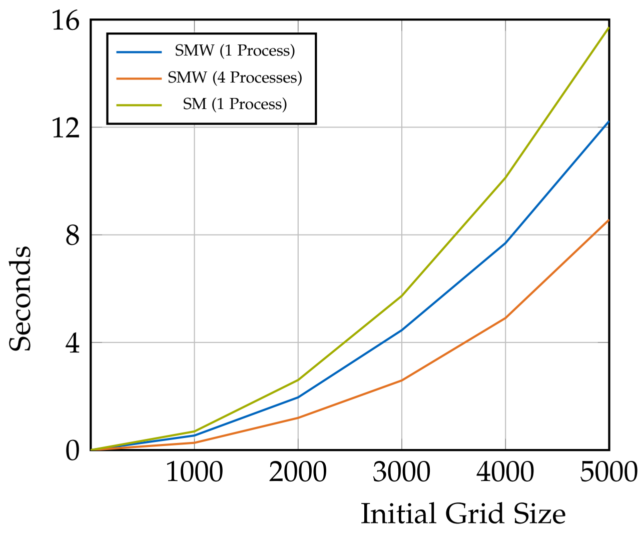


Figure 5.1: Runtimes for Sherman-Morrison (SM) refinement and the new Sherman-Morrison-Woodbury (SMW) refinement with 1 and 4 processes for varying initial grid size. The amount of refined grid-points is fixed to $k = 100$.

In Figure 5.2 the parameters are flipped, which means the initial grid size is fixed and the run-time is observed respective to different batch-sizes of grid-points to be refined. For smaller batch-sizes, the old approach is faster than the new SMW-based one, because in SMW refinements a $k \times k$ matrix has to be inverted, which penalizes run-time for bigger refinement batches. However, the parallelized version of SMW refinements with four processes significantly increases the run-time of this inversion, as the $k \times k$ matrix is directly given to a ScaLAPACK function-call, which is highly optimized. The kink for $k = 400$ probably results from a sub-optimal matrix shape for the block-cyclic ScaLAPACK distribution.

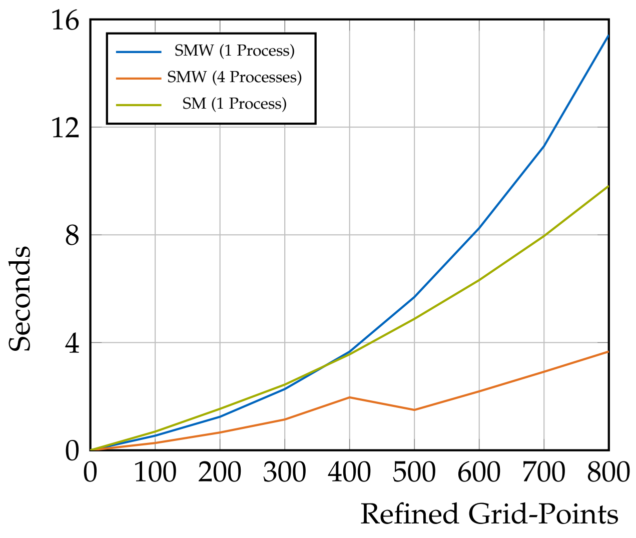


Figure 5.2: Runtimes for Sherman-Morrison (SM) refinement and the new Sherman-Morrison-Woodbury (SMW) refinement with 1 and 4 processes for varying batch size of refined grid-points. The initial grid size is fixed to $n = 1000$.

After refining and/or coarsening, the resulting system of linear equations

$$\vec{\alpha} = B\vec{b}$$

can be solved. Due to the unique approach of the SMW-based online phase, solving this system is a matrix-vector multiplication of size $(n + k)$, and as such is in $\mathcal{O}((n + k)^2)$. After $\vec{\alpha}$ is obtained, the approximated density function $\hat{p}$ can be computed via

$$\hat{p}(\vec{x}) = \sum_{i=1}^{N} \alpha_i \varphi_i(\vec{x}).$$

6 Conclusion

The new approach for performing refinements and/or coarsening in an offline/online split sparse grid density estimation setting brings a boost to run-time in refining/-coarsening, is compatible with any offline decomposition type, and makes use of the advantages of matrix-matrix parallelization. Due to its compatibility with any decomposition type, it profits from introduced features, e.g. fast λ-regularization with tridiagonal decomposition.

Drawbacks include the necessity to explicitly compute the inverse of the offline matrix, as this is expensive in general. Further, if large batches of new grid-points need to be refined simultaneously, the new SMW-based approach suffers from bad scaling, due to necessarily having to invert a resulting $k \times k$ matrix during its refinement operation.

Using Sherman-Morrison-Woodbury updates is a valid method for sparse grid density estimation in an offline/online split setting. It increases the workload in the offline phase, but greatly profits from it in the online phase. The fact that it is compatible with any offline decomposition type lets it inherit desirable features.

7 Appendix

initial grid size	SM(1 process)	SMW(1 process)	SMW(4 processes)
1000	0.693	0.541	0.270
2000	2.604	1.959	1.196
3000	5.732	4.457	2.588
4000	10.125	7.694	4.908
5000	15.720	12.230	8.563

Table 7.1: Runtimes (Seconds) for Sherman-Morrison (SM) refinement and the new Sherman-Morrison-Woodbury (SMW) refinement with 1 and 4 processes for varying initial grid size. The amount of refined grid-points is fixed to $k = 100$.

initial grid size	SM(1 process)	SMW(1 process)	SMW(4 processes)
100	0.693	0.541	0.270
200	1.539	1.244	0.660
300	2.439	2.273	1.143
400	2.562	3.662	1.964
500	4.880	5.691	1.498
600	6.320	8.251	2.188
700	7.956	11.294	2.916
800	9.819	15.423	3.667

Table 7.2: Runtimes (Seconds) for Sherman-Morrison (SM) refinement and the new Sherman-Morrison-Woodbury (SMW) refinement with 1 and 4 processes for varying batch size of refined grid-points. The initial grid size is fixed to $n = 1000$.

Bibliography

[Bos17] D. Boschko. “Orthogonal Matrix Decomposition for Adaptive Sparse Grid Density Estimation Methods.” Bachelor’s Thesis. 2017.

[Fuc18] D. Fuchsgruber. “Integration of SGDE-based Classification into the SG++ Datamining Pipeline.” Bachelor’s Thesis. 2018.

[Hag89] W. W. Hager. “Updating the Inverse of a Matrix.” In: (1989). Vol. 31, No. 2, pp. 221-239.

[Peh13] B. Peherstorfer. “Model Order Reduction of Parametrized Systems with Sparse Grid Learning Techniques.” PhD thesis. 2013.

[Sch19] J. Schopohl. “Domain Parallelization of SGDE based Classification.” Bachelor’s Thesis. 2019.

[Sie16] A. Sieler. “Refinement and Coarsening of Online-Offline Data Mining Methods with Sparse Grids.” Bachelor’s Thesis. 2016.