

Combinative preconditioning based on Relaxed Nested Factorization and Tangential Filtering preconditioner

Pawan Kumar, Laura Grigori, Frédéric Nataf, Qiang Niu

▶ **To cite this version:**

Pawan Kumar, Laura Grigori, Frédéric Nataf, Qiang Niu. Combinative preconditioning based on Relaxed Nested Factorization and Tangential Filtering preconditioner. [Research Report] RR-6955, INRIA. 2009. inria-00392881

HAL Id: inria-00392881
https://inria.hal.science/inria-00392881

Submitted on 9 Jun 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Combinative preconditioning based on Relaxed Nested Factorization and Tangential Filtering preconditioner

Pawan Kumar — Laura Grigori — Frederic Nataf — Qiang Niu

N° 6955

Juin 2009

Thème NUM



ISSN 0249-6399 ISRN INRIA/RR--6955--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY - ÎLE-DE-FRANCE

Combinative preconditioning based on Relaxed Nested Factorization and Tangential Filtering preconditioner

Pawan Kumar* , Laura Grigori† , Frederic Nataf‡ , Qiang Niu§

Thème NUM — Systèmes numériques
Équipe-Projet Grand-large

Rapport de recherche n° 6955 — Juin 2009 — 24 pages

Abstract: The problem of solving block tridiagonal linear systems arising from the discretization of PDE is considered. The nested factorization preconditioner introduced by [J. R. Appleyard and I. M. Cheshire, *Nested Factorization*, SPE 12264, presented at the Seventh SPE Symposium on Reservoir Simulation, San Francisco, 1983] is an effective preconditioner for certain class of problems and a similar method is implemented in Schlumerger's Eclipse oil reservoir simulator.

In this paper, a relaxed version of Nested Factorization preconditioner is proposed as a replacement to ILU(0). Indeed, the proposed preconditioner is SPD and leads to a stable splitting if the input matrix is S.P.D.. For ILU(0), equivalent properties hold if the input matrix is a M-matrix. Moreover it has no storage cost. Effective multiplicative/additive preconditioning is achieved in combination with Tangential filtering preconditioner with the filter vector chosen as vector of *ones*. Numerical tests are carried out with both additive and multiplicative combinations. With this setup the new preconditioner is as robust as the combination of ILU(0) with tangential filtering preconditioner.

Key-words: preconditioner, linear system, frequency filtering decomposition, GMRES, nested factorization, incomplete LU, eigenvalues

* INRIA Saclay - Ile de France, Laboratoire de Recherche en Informatique, Universite Paris-Sud 11, France (Email:pawan.kumar@lri.fr)

† INRIA Saclay - Ile de France, Laboratoire de Recherche en Informatique, Universite Paris-Sud 11, France (Email:laura.grigori@inria.fr)

‡ Laboratoire J. L. Lions, CNRS UMR7598, Universite Paris 6, France; Email: nataf@ann.jussieu.fr

§ School of Mathematical Sciences, Xiamen University, Xiamen, 361005, P.R. China; The work of this author was performed during his visit to INRIA, funded by China Scholarship Council; Email:kangniu@gmail.com

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université
4, rue Jacques Monod, 91893 ORSAY Cedex
Téléphone : +33 1 72 92 59 00

Préconditionnement basé sur factorisation emboîtée et filtrage tangentiel

Résumé : Dans ce papier nous présentons une version de factorisation emboîté avec relaxation. Cette factorisation est proposée comme remplacement de ILU(0). Le préconditionneur proposé est SPD si la matrice originale est SPD. Pour ILU(0)des propriété équivalentes existent si la matrice originale est une M-matrice. Un préconditionnement efficace est obtenu en combinaison avec le filtrage tangentiel avec le vecteur de filtrage de *ones*. Des tests numériques sont présentés pour des combinaisons additifs et multiplicatifs. Ils montrent que le préconditionneur est aussi robuste que la combinaison obtenue par ILU(0)avec le filtrage tangentiel.

Mots-clés : préconditionnement, factorisation emboîtée, factorisation LU incomplète

1 INTRODUCTION

Several applications are modelled by non-linear partial differential equations, examples include oil reservoir simulations and fluid flow through porous media. These equations are usually solved by the Newton's method or fixed point method, where at each step a preconditioned iterative method is generally used for solving a large sparse linear system

$$Ax = b. \tag{1}$$

The linear systems change at each step, and solving them constitute the most computationally intensive part of the simulation. Therefore, an efficient preconditioner with fast setup time plays an important role in the overall simulation process. Also, since the systems are very large, the preconditioner should not be too demanding in terms of memory requirements. This problem has already been extensively studied, see [7, 16, 18] for the description of algorithms and their comparisons. In particular, one class of methods which have proved particularly successful are the algebraic multigrid methods, see for example, [17, 19, 20, 24]. However, for certain problems the algebraic multigrid methods involve relatively high setup cost, which leads us to look into some other alternatives in this work.

With fast setup time and very modest storage requirement, the Nested Factorization (NF) preconditioner introduced in [1] is a powerful preconditioner; it has been found [1, 15] that for certain class of problems it performs better than the widely used ILU(0) or Modified ILU [18]. NF takes as input a matrix which has a nested block tridiagonal structure. Matrices arising from the discretization of P.D.E can be permuted to this form. The method of NF differs from ILU(0) or Modified ILU(0) in that the preconditioning matrix in NF is not formed strictly from upper and lower triangular factors. Instead, block lower and upper factors are constructed using a procedure which adds one dimension at a time to the preconditioning matrix having the diagonal matrix on the lowest level.

The NF preconditioner has some important properties. If B_{NF} is the NF preconditioner, then $colsum(B_{NF} - A) = 0$ (also known as zero colsum property), as a consequence the sum of the residuals in successive Krylov iterations remain zero, provided a suitable initial solution is used [1]. This property can provide a very useful check on the correctness of the implementation. Further, quoting [1], "the factorization procedure conserves material exactly for each phase at each linear iteration, and accommodates non-neighbour connections (arising from the treatment of the faults, completing the circle in three-dimensional coning studies, numerical aquifers, dual porosity/permeability systems etc.) in a natural way." Moreover, for fluid flow problems it is proved in [2] that the lower and the upper triangular factors of NF are nonsingular. Due to these desirable qualities, the NF preconditioner is of particular interest in the oil reservoir industry; a method similar to NF is implemented in Schlumerger's Eclipse oil reservoir simulator [12]. However, as we will see later in the tables, the NF preconditioner can fail on some problems.

The main result of this paper is the introduction of a new preconditioner which corresponds to a relaxed version of NF, namely RNF(0,0). Compared to ILU(0), it has several important properties:

- If A is S.P.D. then,
  RNF(0,0) is S.P.D.
  RNF(0,0) leads to a stable splitting
- no setup cost
- no storage requirement

The new preconditioner is particularly useful when multiplicate/additive preconditioning is achieved in combination with tangential filtering preconditioner [4]. With this setup the new preconditioner is as robust as the combination of ILU(0)with tangential filtering preconditioner. Notably both multiplicative and additive combination are tried. Good results are obtained with the additive combination which moreover has an advantage over multiplicative combination on parallel systems; each of the preconditioner solves involved with the additive case can be done simultaneously on two different processors. In order to make a comprehensive study, we consider a general relaxed version of NF, namely RNF(α,β).

This article is organized as follows. Notations play an important role in explaining the method of NF; we carefully introduce them in the next section. In section 3, NF is explained and later in the section RNF is introduced. The method of LFTFD is discussed briefly in section 4. In section 5, we present convergence results for the combination involving RNF(0,0) and LFTFD which shows that the combination is convergent. In section 6 we will comment on numerical results. And finally, section 7 concludes the paper.

2 NOTATIONS

Throughout the paper, matrix A is considered to be arising from the finite difference or finite volume discretization of two or three dimensional partial differential equations. For example, using 7-point formula for three dimensional problems for an $nx \times ny \times nz$ grid, where nx denotes the number of points on the line, ny denotes the number of lines on each plane, and nz denotes the number of planes, we obtain an $(nx \times ny \times ny) \times (nx \times ny \times nz)$ matrix. Let nxy denote $nx \times ny$, the number of unknowns on each plane, nyz denote $ny \times nz$, the total number of lines on all planes, and $nxyz$ denote $nx \times ny \times nz$, the total number of unknowns. Then the resulting matrix has a nested tridiagonal structure as follows:

$$A = \begin{pmatrix} \widehat{D}_1 & \widehat{U}_3^1 & & \\ \widehat{L}_3^1 & \widehat{D}_2 & \ddots & \\ & \ddots & \ddots & \widehat{U}_3^{nz-1} \\ & & \widehat{L}_3^{nz-1} & \widehat{D}_{nz} \end{pmatrix}. \tag{2}$$

Here the diagonal blocks $\widehat{D}_i s$ are the blocks corresponding to the unknowns of the i^{th} plane. The blocks $\widehat{L}_3^i$ and $\widehat{U}_3^i$ are diagonal matrices of size nxy, and they correspond to the connections between the i^{th} and $(i+1)^{th}$ plane. Further, the diagonal blocks $\widehat{D}_i s$ are themselves block tridiagonal, i.e. they have

the following structure,

$$\widehat{D}_i = \begin{pmatrix} \overline{D}_{(i-1)*ny+1} & \overline{U}_2^{(i-1)*ny+1} & & \\ \overline{L}_2^{(i-1)*ny+1} & \overline{D}_{(i-1)*ny+2} & \ddots & \\ & \ddots & \ddots & \overline{U}_2^{i*ny-1} \\ & & \overline{L}_2^{i*ny-1} & \overline{D}_{i*ny} \end{pmatrix}$$

where the blocks $\overline{L}_2^i$ and $\overline{U}_2^i$ are diagonal matrices of size nx each and they correspond to the connections between the lines. The diagonal blocks $\overline{D}_i$ are themselves tridiagonal matrices,

$$\overline{D}_i = \begin{pmatrix} \tilde{D}_{(i-1)*nx+1} & \tilde{U}_1^{(i-1)*nx+1} & & \\ \tilde{L}_1^{(i-1)*nx+1} & \tilde{D}_{(i-1)*nx+2} & \ddots & \\ & \ddots & \ddots & \tilde{U}_{i*nx-1} \\ & & \tilde{L}_1^{i*nx-1} & \tilde{D}_{i*nx} \end{pmatrix}$$

with the scalars $\tilde{L}_1^{(i-1)*nx+j}$ and $\tilde{U}_1^{(i-1)*nx+j}$ being the corresponding connections between the cells. We observe that this matrix has nested tridiagonal structure. For any matrix K, $Diag(K)$ refers to the strict diagonal of K; for any vector v, $Diag(v)$ refers to the diagonal matrix formed from the vector v. For any diagonal matrix M of size $nxyz$, the i^{th} entry of M is denoted by $\tilde{M}_i$, the i^{th} line block of M is denoted by $\overline{M}_i$, and the i^{th} plane block of M is denoted by $\widehat{M}_i$. Below we list the structures of the matrices L_1, U_1, L_2, U_2, L_3, and U_3 to be referred to later.

$$Diag(A) = \begin{pmatrix} \tilde{D}_1 & & & \\ & \tilde{D}_2 & & \\ & & \ddots & \\ & & & \tilde{D}_{nxyz} \end{pmatrix}, \quad M = \begin{pmatrix} \tilde{M}_1 & & & \\ & \tilde{M}_2 & & \\ & & \ddots & \\ & & & \tilde{M}_{nxyz} \end{pmatrix}$$

$$L_1 = \begin{pmatrix} 0 & & & \\ \tilde{L}_1^1 & 0 & & \\ & \ddots & \ddots & \\ & & \tilde{L}_1^{nxyz-1} & 0 \end{pmatrix}, \quad U_1 = \begin{pmatrix} 0 & \tilde{U_1}^1 & & \\ & 0 & \ddots & \\ & & \ddots & \tilde{U_1}^{nxyz-1} \\ & & & 0 \end{pmatrix}$$

$$L_2 = \begin{pmatrix} 0 & & & & \\ \overline{L}_2^1 & \ddots & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & \overline{L}_2^{nyz-1} & 0 \end{pmatrix}, \quad U_2 = \begin{pmatrix} 0 & \overline{U}_2^1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & \ddots & \overline{U}_2^{nyz-1} \\ & & & & 0 \end{pmatrix}$$

$$L_3 = \begin{pmatrix} 0 & & & & \\ \widehat{L}_3^1 & \ddots & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & \widehat{L}_3^{nz-1} & 0 \end{pmatrix}, \quad U_3 = \begin{pmatrix} 0 & \widehat{U}_3^1 & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & \ddots & \widehat{U}_3^{nz-1} \\ & & & & 0 \end{pmatrix}$$

3 RELAXED NESTED FACTORIZATION

In this section, we introduce RNF for three-dimensional case, the two-dimensional case will be considered a special case of three-dimensional case with just one plane. We first introduce a special version of the preconditioner for which we prove that it is S.P.D when the input matrix A is S.P.D . Later, we introduce a more general version which will prove significant for the two-dimensional problems.

3.1 A special case : RNF(0,0)

We define the RNF(0,0) preconditioner as follows:

$$\begin{aligned} B_{RNF(0,0)} &= (P + L_3)(I + P^{-1}U_3) \\ P &= (T + L_2)(I + T^{-1}U_2) \\ T &= (M + L_1)(I + M^{-1}U_1) \\ M &= diag(A)\,. \end{aligned}$$

3.2 The general case : RNF(α,β)

The general version of relaxed nested factorization preconditioner denoted by $B_{RNF(\alpha,\beta)}$ is defined hierarchically as follows:

$$\begin{aligned} B_{RNF(\alpha,\beta)} &= (P + L_3)(I + P^{-1}U_3), \\ P &= (T + L_2)(I + T^{-1}U_2), \\ T &= (M + L_1)(I + M^{-1}U_1), \\ M &= diag(A) - \alpha\ L_1M^{-1}U_1 \\ &- \beta\ colsum(L_2T^{-1}U_2) \\ &- \beta\ colsum(L_3P^{-1}U_3), \end{aligned}$$

where $colsum(K) = Diag(\mathbf{1}K)$ for any square matrix K, and $\mathbf{1}$ denotes the vector $[1, 1, ..., 1]$. Note that with $\alpha = 1$ and $\beta = 1$ we get the classical Nested Factorization preconditioner as described in [1].

For RNF(1,1) we observe that

$$B_{RNF(1,1)} - A = L_2T^{-1}U_2 - colsum(L_2T^{-1}U_2) + L_3P^{-1}U_3 - colsum(L_3P^{-1}U_3)$$

so that the preconditioner satisfies

$$\mathbf{1}B_{RNF(1,1)} = \mathbf{1}A, \tag{3}$$

i.e., a column sum constraint as in MILU [18].

The matrix M is computed before the iteration begins. Finding M requires writing down the above expression for M. We notice that M is a diagonal matrix, since $Diag(A)$, $L_1M^{-1}U_1$, $colsum(L_2T^{-1}U_2)$, and $colsum(L_3P^{-1}U_3)$ are diagonal matrices. The matrices T and P are block diagonal matrices with square diagonal blocks of sizes nx and nxy respectively. We have

$$L_1M^{-1}U_1 = \begin{pmatrix} 0 & & & \\ & \tilde{L}_1^1\tilde{M}_1^{-1}\tilde{U}_1^1 & & \\ & & \ddots & \\ & & & \tilde{L}_1^{nxyz-1}\tilde{M}_{nxyz-1}^{-1}\tilde{U}_1^{nxyz-1} \end{pmatrix},$$

$$T = \begin{pmatrix} T_1 & & & \\ & T_2 & & \\ & & \ddots & \\ & & & T_{nyz} \end{pmatrix},$$

$$colsum(L_2T^{-1}U_2) = \begin{pmatrix} 0 & & & \\ & colsum(\overline{L}_2^1T_1^{-1}\overline{U}_2^1) & & \\ & & \ddots & \\ & & & colsum(\overline{L}_2^{nyz-1}T_{nyz-1}^{-1}\overline{U}_2^{nyz-1}) \end{pmatrix},$$

$$P = \begin{pmatrix} P_1 & & \\ & \ddots & \\ & & P_{nz} \end{pmatrix}, and$$

$$colsum(L_3P^{-1}U_3) = \begin{pmatrix} 0 & & & \\ & colsum(\widehat{L}_3^1P_1^{-1}\widehat{U}_3^1) & & \\ & & \ddots & \\ & & & colsum(\widehat{L}_3^{nz-1}P_{nz-1}^{-1}\widehat{U}_3^{nz-1}) \end{pmatrix}.$$

3.3 Construction of RNF(α, β) preconditioner

A detailed pseudocode for the construction of $RNF(\alpha, \beta)$ preconditioner is provided in Algorithm (1), we briefly outline the construction in this section. The entries of M are calculated in a single sweep through the grid. Reading the entries row by row for the expression of M, we can determine M.

The first row gives $\tilde{M}_1 = \tilde{D}_1$. For the second cell onwards up to the last cell of first line there are contributions from the term $L_1 M^{-1} U_1$, and we have

$$\tilde{M}_i = \tilde{D}_i - \tilde{L}_1^{i-1}\tilde{U}_1^{i-1}/\tilde{M}_{i-1},\ i = 2,\ \ldots,\ nx.$$

When the update from the first line is finished, each element of M for the second line also depends on the previous line through the term $colsum(L_2 T^{-1} U_2)$, and each element of M for second plane onwards depends on the previous plane through the term $colsum(L_3 P^{-1} U_3)$. The $(j+1)^{th}$ line block of the block diagonal matrix $colsum(L_2 T^{-1} U_2)$, can be computed as follows:

$$colsum(\overline{L}_2^j T_j^{-1} \overline{U}_2^j) = Diag(\mathbf{1}\overline{L}_2^j T_j^{-1} \overline{U}_2^j) = Diag((\overline{U}_2^j)^T T_j^{-T} (\overline{L}_2^j)^T \mathbf{1}).$$

A similar calculation can be performed for the computation of the plane blocks of $colsum(L_3 P^{-1} U_3)$.

3.4 Solution procedure

In this section we will see how the nested expression for B is used to solve the equation

$$B_{RNF(\alpha,\beta)} u = v$$

required in a preconditioned Krylov iterative method. At the outermost level, we solve

$$(P + L_3)(I + P^{-1} U_3) u = v$$

doing the forward sweeps of the form $q = P_i^{-1}(v - \widehat{L}_3^{i-1} q)$, and the backward sweeps of the form $u = u - P_i^{-1} \widehat{U}_3^i q$. These equations are solved for one plane at a time. The solution involves solving equations of the form $P_i w = x$ for each plane. To solve this equation we use the fact that each plane block of the preconditioner has an incomplete block factorization represented by

$$P = (T + \overline{L}_2)(I + T^{-1}\overline{U}_2),$$

and we solve $P_i w = x$ using the forward sweeps of the form $r = T_i^{-1}(x - \overline{L}_2^{i-1} r)$, and the backward sweeps of the form $w = w - T_i^{-1}\overline{U}_2^i r$. We notice that we need to solve equations of the form $T_i y = z$ corresponding to each line block of the preconditioner, and for this we use the following factored form of T

$$T = (M + L_1)(I + M^{-1} U_1),$$

and we solve $T_i y = z$ using the forward sweeps of the form $s = \tilde{M}_i^{-1}(z - \tilde{L}_1^{i-1} s)$, and the backward sweeps of the form $y = y - \tilde{M}_i^{-1}\tilde{U}_1^i s$.

Algorithm 1 PSEUDOCODE TO FIND M FOR RNF(α, β)

INPUT: $Diag(A)$, L_1, L_2, L_3, U_1, U_2, U_3
OUTPUT: M
for $i = 1$ to nz (Number of planes) **do**
  if $i \neq 1$ and $\beta \neq 0$ (Not the first plane) **then**
    Solve $P_{i-1}^T \widehat{x} = (\widehat{L_3}^{i-1})^T.1$
    $\widehat{M_i} = \widehat{M_i}$ - $\beta(\widehat{U_3}^{i-1})^T \widehat{x}$
  end if
  UPDATE-FROM-LINES($\widehat{M_i}$)
end for

UPDATE-FROM-LINES($\widehat{M_i}$)
for $j = 1$ to ny (Number of lines in current plane) **do**
  if $j \neq 1$ and $\beta \neq 0$(Not the first line) **then**
    Solve $T_{j-1}^T \overline{x} = (\overline{L_2}^{j-1})^T.1$
    $\overline{M}_{(i-1)*ny+j} = \overline{M}_{(i-1)*ny+j}$ - $\beta(\overline{U}_2^{j-1})^T \overline{x}$
  end if
  UPDATE-FROM-CELLS($\overline{M}_{(i-1)*ny+j}$)
end for

UPDATE-FROM-CELLS($\overline{M}_i$)
$\tilde{M}_{(i-1)*nx+1} = \tilde{D}_{(i-1)*nx+1}$ (Update from first cell of current line)
for $k = 2$ to nx (Update from rest of the cells) **do**
  $\tilde{M}_{(i-1)*nx+k} = \tilde{M}_{(i-1)*nx+k}$ - $\alpha\ \tilde{L}_1^{(i-1)*nx+k-1} \tilde{M}^{-1}_{(i-1)*nx+k-1} \tilde{U}_1^{(i-1)*nx+k-1}$
end for

4 LOW FREQUENCY TANGENTIAL FILTERING DECOMPOSITION

In [4], a Low Frequency Tangential Filtering Decomposition(LFTFD) preconditioner was developed. If t is any vector and B_{LF} is the LFTFD preconditioner, then the preconditioner satisfies the right filtering property

$$At = B_{LF}t\,. \tag{4}$$

We briefly describe the decomposition procedure. The preconditioner B_{LF} is defined as follows

$$B_{LF} = (Q + L_3)(I + Q^{-1}U_3),$$

where Q is a block diagonal matrix with diagonal blocks Q_i of size nxy each, the blocks $Q_i s$ are defined as follows

$$Q_i \;=\; \begin{cases} \widehat{D}_1, i \;=\; 1, \\ \widehat{D}_i - \widehat{L}_3^{i-1}(2\beta_{i-1} - \beta_{i-1}Q_{i-1}\beta_{i-1})\widehat{U}_3^{i-1},\ i = 2\ to\ nz \end{cases}$$

where β_i is given by

$$\beta_i = Diag((Q_i^{-1}\widehat{U}_i t_{i+1})./(\widehat{U}_i t_{i+1})) \tag{5}$$

Here ./ is a pointwise vector division, $Diag(v)$ is the diagonal matrix constructed from the vector v and $t = [t_1, ..., t_{nz}]^T$ is the filter vector. Division by zero being undefined, it is assumed that no component of t is zero.

This preconditioner seems to be robust enough when combined with ILU(0) using multiplicative preconditioner similar to (2) for a wide class problems for which it was tested. For more on this we refer the reader to [4]. For other preconditioners satisfying the filtering property (4), the reader is referred to [9, 10, 21, 22, 23].

5 ANALYSIS OF THE MULTIPLICATIVE PRECONDITIONERS INVOLVING RNF(0,0), RNF(1,0), AND LFTFD

For any matrix K, let $K \succ 0$ denote that the matrix K is symmetric positive definite; let $K > 0$ denote that each entry of K is positive, and let $\lambda_i(K)$ denote an arbitrary eigenvalue of K.

For convenience, we denote the preconditioners RNF(0,0) by B_{RNF0}, RNF(1,0) by B_{RNF1}, and LFTFD by B_{LF}. When the results apply to both RNF(0,0) and RNF(1,0), we denote the preconditioner as B_{RNF}; further, let B_c denote

$$B_c^{-1} = B_{RNF}^{-1} + B_{LF}^{-1} - B_{LF}^{-1}AB_{RNF}^{-1}. \tag{6}$$

The Following theorem shows that the multiplicative preconditioner obtained using formula (6) satisfies the right filtering property (4).

Lemma 5.1 *If B_{LF} satisfies the right filtering property (4) on vector t, then the preconditioner B_{c_1} obtained by combining B_{LF} and $B_{RNF(\alpha,\beta)}$ using formula (6) satisfies the right filtering property (or constraint) on the same vector, i.e.*

$$B_{c1}t = At$$

.

Proof: From Eqn. (6) we have

$$I - B_{c1}^{-1}A = (I - B_{RNF(\alpha,\beta)}^{-1}A)(I - B_{LF}^{-1}A)$$

From this identity we have,

$$(I - B_{c1}^{-1}A)t \quad = \quad 0 \ (\ since\ (I - B_{LF}^{-1}A)t = 0)$$

Hence the lemma.

In Lemma 5.2, we prove that the fixed point iteration involving RNF(0,0) preconditioner is convergent. Lemma 5.3 can be found in [4]. In our last result Theorem 5.6, we prove that the fixed point iteration involving B_{c1} preconditioner is convergent.

Lemma 5.2 *If $A \succ 0$, then $B_{RNF0} \succ 0$, $B_{RNF1} \succ 0$, $\lambda_i(B_{RNF0}^{-1}A) \in (0,1]$, and $\lambda_i(B_{RNF1}^{-1}A) \in (0,1]$.*

Proof: The RNF(0,0) preconditioner is defined as follows:

$$\begin{aligned}
B_{RNF0} &= (P_{RNF0} + L_3)P_{RNF0}^{-1}(P_{RNF0} + L_3^T),\\
P_{RNF0} &= (T_{RNF0} + L_2)T_{RNF0}^{-1}(T_{RNF0} + L_2^T),\\
T_{RNF0} &= (M_{RNF0} + L_1)M_{RNF0}^{-1}(M_{RNF0} + L_1^T),\\
M_{RNF0} &= Diag(A).
\end{aligned}$$

It is easy to see that since $A \succ 0$, $M_{RNF0} = Diag(A) > 0$, so $T_{RNF0} \succ 0$, $P_{RNF0} \succ 0$, and hence $B_{RNF0} \succ 0$. Since $B_{RNF0}^{-1}A$ is similar to a symmetric matrix $B_{RNF0}^{-\frac{1}{2}}AB_{RNF0}^{-\frac{1}{2}}$, all the eigenvalues of $B_{RNF0}^{-1}A$ are real. Also, since B_{RNF0}, T_{RNF0}, and P_{RNF0} are positive definite and

$$B_{RNF0} = A + L_1M_{RNF0}^{-1}L_1^T + L_2T_{RNF0}^{-1}L_2^T + L_3P_{RNF0}^{-1}L_3^T,$$

we have

$$\begin{aligned}
(B_{RNF0}x, x) &= (Ax, x) + (M_{RNF0}^{-1}L_1^Tx, L_1^Tx) + (T_{RNF0}^{-1}L_2^Tx, L_2^Tx) + (P_{RNF0}^{-1}L_3^Tx, L_3^Tx),\\
&\geq (Ax, x),\\
&> 0,\ \forall x \neq 0,
\end{aligned}$$

and from this we have $\lambda_i(B_{RNF0}^{-1}A) \in (0,1]$. The B_{RNF1} preconditioner has similar hierarchical representation as B_{RNF0}, it is defined as follows:

$$\begin{aligned}
B_{RNF1} &= (P_{RNF1} + L_3)P_{RNF1}^{-1}(P_{RNF1} + L_3^T),\\
P_{RNF1} &= (T_{RNF1} + L_2)T_{RNF1}^{-1}(T_{RNF1} + L_2^T),\\
T_{RNF1} &= (M_{RNF1} + L_1)M_{RNF1}^{-1}(M_{RNF1} + L_1^T),\\
M_{RNF1} &= D - L_1M_{RNF1}^{-1}L_1^T.
\end{aligned}$$

Since $A \succ 0$, we have $T_{RNF1} = Diag(A) + L_1 + L_1^T \succ 0$, $P_{RNF1} \succ 0$, and hence $B_{RNF1} \succ 0$. Also

$$\begin{aligned} B_{RNF1} &= A + L_2 T_{RNF1}^{-1} L_2^T + L_3 P_{RNF1}^{-1} L_3^T, \\ (B_{RNF1}x, x) &\geq (Ax, x), \\ &> 0, \ \forall x \neq 0, \end{aligned}$$

and from this we have $\lambda_i(B_{RNF1}^{-1}A) \in (0, 1]$. Hence the proof.

Lemma 5.3 *Let $A \succ 0$, then $B_{LF} \succ 0$, $B_{LF} - A = N \succ 0$, and $\lambda_i(B_{LF}^{-1}A) \in (0, 1]$.*

Proof: For the proof of the fact that $B_{LF} \succ 0$ and $B_{LF} - A = N \succ 0$ see Lemma 2.2 in [4]. Now by the similar argument as in the previous lemma, we have

$$(B_{LF}x, x) = (Ax, x) + (Nx, x) \geq (Ax, x) > 0, \ \forall x \neq 0,$$

and from this it follows that $\lambda_i(B_{LF}^{-1}A) \in (0, 1]$.

If $A \succ 0$, then the inner product $(\ ,\ )_A$ defined by $(u, v)_A = u^T Av$ is a well defined inner product, and it induces the energy norm $\|\ \|_A$ defined by $\|v\|_A = (v, v)_A^{\frac{1}{2}}$ for any vector v. A matrix K is called A-selfadjoint if

$$(Ku, v)_A = (u, Kv)_A.$$

or equivalently if,

$$A^{-1}K^T A = K \tag{7}$$

Lemma 5.4 *If A is symmetric, then the matrices $I - B_{RNF}^{-1}A$, $I - B_{LF}^{-1}A$, and $I - B_c^{-1}A$ are A-selfadjoint.*

Proof: Using criteria for self-adjointness defined by (7), it can be easily proved that $I - B_{RNF}^{-1}A$ and $I - B_{LF}^{-1}A$ are A-selfadjoint. To prove that $I - B_c^{-1}A$ is self-adjoint, we have

$$\begin{aligned} A^{-1}(I - B_c^{-1}A)^T A &= A^{-1}(I - B_{LF}^{-1}A)^T (I - B_{RNF}^{-1}A)^T A, \\ &= A^{-1}(I - AB_{RNF}^{-1} - AB_{LF}^{-1} + AB_{LF}^{-1}AB_{RNF}^{-1})A, \\ &= I - B_c^{-1}A. \end{aligned}$$

Hence the theorem.

For any matrix K, let $\rho(K) = max_i(|\lambda_i(K)|)$ denote the spectral radius of K.

Lemma 5.5 *[Ashby, Holst, Manteuffel, and Saylor] [5] If $A \succ 0$ and K is A-self adjoint, then $\|K\|_A = \rho(K)$.*

Theorem 5.6 *If $A \succ 0$, then the fixed point iteration with the corresponding error propagation matrix $(I - B_c^{-1}A)$ is convergent, i.e.,*

$$\rho(I - B_c^{-1}A) < 1.$$

Proof: Using Theorem (5.4) and Lemma (5.5), we have

$$\begin{aligned}
\rho(I - B_c^{-1}A) &= \|I - B_c^{-1}A\|_A, \\
&= \|(I - B_{RNF}^{-1}A)(I - B_{LF}^{-1}A)\|_A, \\
&\leq \|(I - B_{RNF}^{-1}A)\|_A \|(I - B_{LF}^{-1}A)\|_A, \\
&= \rho(I - B_{RNF}^{-1}A)\rho(I - B_{LF}^{-1}A), \\
&< 1 \ (\textit{Using Lemma } 5.2 \textit{ and Lemma } 5.3).
\end{aligned}$$

Hence the proof.

6 NUMERICAL EXPERIMENTS

In this section we present numerical results for NF, RNF, and its combinations with LFTFD. We shall compare each of these preconditioners, and outline the advantages and disadvantages of each of them. First, we will compare the convergence of RNF(α,β) with other classical preconditioners, and later we shall observe the spectrum behavior. Secondly, we shall compare several combinative preconditionings involving RNF(α,β). Thirdly, we shall compare the results between additive and multiplicative preconditionings, and then finally we shall discuss the results of combinative preconditionings with the filter vector chosen as Ritz vector.

6.1 Source of matrices

We consider the boundary value problem as in [4]

$$\begin{aligned}
div(\mathbf{a}(x)u) - div(\kappa(x)\nabla u) = f \ \ in \ \ \Omega, \\
u = 0 \ \ on \ \ \partial\Omega_D, \\
\tfrac{\partial u}{\partial n} = 0 \ \ on \ \ \partial\Omega_N,
\end{aligned} \tag{8}$$

where $\Omega = [0,1]^n$ ($n = 2$, or 3), $\partial\Omega_N = \partial\Omega \setminus \partial\Omega_D$. The vector field $\mathbf{a}$ and the tensor κ are the given coefficients of the partial differential operator. In 2D case, we have $\partial\Omega_D = [0,1] \times \{0,1\}$, and in 3D case, we have $\partial\Omega_D = [0,1] \times \{0,1\} \times [0,1]$.

The following five cases are considered:

`Case 4.1:` *Advection-diffusion problem with a rotating velocity in two dimensions:*
The tensor κ is identity, and the velocity is $\mathbf{a} = (2\pi(x_2 - 0.5), 2\pi(x_1 - 0.5))^T$. We test problems for the two-dimensional case.

`Case 4.2:` *Non-homogeneous problems with large jumps in the coefficients in two dimensions:*
The coefficient $\mathbf{a}$ is zero. The tensor κ is isotropic and discontinuous. It jumps from the constant value 10^3 in the ring $\frac{1}{2\sqrt{2}} \leq |x - c| \leq \frac{1}{2}$, $c = (\frac{1}{2}, \frac{1}{2})^T$, to 1 outside. We test problems for the two-dimensional case.

`Case 4.3:` *Skyscraper problems:*
The tensor κ is isotropic and discontinuous. The domain contains many zones of high permeability which are isolated from each other. Let $[x]$ denote the integer value of x. For two-dimensional case, we define $\kappa(x)$ as follows:

$$\kappa(x) = \begin{cases} 10^3 * ([10 * x_2] + 1), \ if \ \ [10 * x_i] = 0 \ \ mod(2), \ i = 1, 2, \\ 1, \ otherwise. \end{cases}$$

and for three-dimensional case $\kappa(x)$ is defined as follows:

$$\kappa(x) = \begin{cases} 10^3 * ([10 * x_2] + 1), & if \ \ [10 * x_i] = 0 \ mod(2) \ , \ i = 1, 2, 3, \\ 1, & otherwise. \end{cases}$$

`Case 4.4:` *Convective skyscraper problems:*
The same with the Skyscraper problems except that the velocity field is changed to be $\mathbf{a} = (1000, 1000, 1000)^T$. Both two and three dimensional case are considered.

`Case 4.5:` *Anisotropic layers:*
The domain is made of 10 anisotropic layers with jumps of up to four orders of magnitude, and an anisotropy ratio of up to 10^3 in each layer. For three-dimensional case, the cube is divided into 10 layers parallel to $z = 0$, of size 0.1, in which the coefficients are constant. The coefficient κ_x in the ith layer is given by $v(i)$, the latter being the ith component of the vector $v = [\alpha, \beta, \alpha, \beta, \alpha, \beta, \gamma, \alpha, \alpha]$, where $\alpha = 1$, $\beta = 10^2$ and $\gamma = 10^4$. We have $\kappa_y = 10\kappa_x$ and $\kappa_z = 1000\kappa_x$. The velocity field is zero.

We consider problems on uniform grid with $n \times n$ nodes for the two-dimensional case, where we choose $n = 100, 200, 300$, and 400; whereas, for the three-dimensional case we consider the uniform grid with $n \times n \times n$ nodes, and we choose $n = 20, 30$, and 40.

6.2 Comments on the numerical tests

All our test results were obtained with GMRES(20) in double precision arithmetic with initial solution vector as vector of all zeros. The known solution is a random vector. The stopping criteria is the decrease of the relative residual below 10^{-12}. The maximum number of iterations allowed is 200. The method of NF is implemented in Fortran 90, whereas, the method of LFTFD is implemented in MATLAB, for this reason instead of CPU time, we provide the flop count (see Table 1) for a $nx \times nx \times nx$ uniform grid for all the methods considered.

In table 1, we introduce short notations for the methods and the test cases considered.

To compare RNF(1,1) and RNF(0,0) with other classical methods, we present results in Table (3). We notice that RNF(1,1) takes smaller number of steps to converge as compared to the classical preconditioners like ILU and MILU. In most of the test cases, we observe that whenever MILU converged within 200 steps, RNF(1,1) converged as well. For matrices with discontinuous coefficients like skyscrapper and convective skyscrapper problems, RNF(1,1), ILU(0), and MILU fail to converge within the maximum number of iterations. The conditioning behavior of RNF(0,0) seems to be similar to ILU(0) in terms of the number of steps required for convergence; the problems for which ILU(0) converges, RNF(0,0) converges as well, and for those for which ILU(0) does not converge, RNF(0,0) does not converge either. These observations suggest that RNF(0,0) alone can replace ILU(0) for certain problems; the advantage of doing this is that for RNF(0,0), no cost of construction is required, and there are no storage requirements (see Table 2). On the other hand, comparing RNF(0,0) and RNF(1,0) (see table 3 and table 4), we find that RNF(1,0) converges faster when compared to RNF(0,0). So, RNF(1,0) can also replace ILU(0) for the test cases considered.

Table 1: Short notations for the methods and the test cases considered.

METHODS	**STANDS FOR**
ILU	Incomplete LU with zero fill-in
MILU	Modified ILU with colsum constraint
TFILU	Combinative preconditioning with LFTFD and ILU
RNF(α,β)	Relaxed nested factorization with parameters α and β.
TFRNF(α,β)	Combinative preconditioning with RNF(α,β) and LFTFD
TF(r,l)RNF(1,1)	Combinative preconditioning with RNF(1,1) and LFTFD with filter vector as the largest Ritz vector
TF(r,s)RNF(0,0)	Combinative preconditioning with RNF(0,0) and LFTFD with filter vector as the smallest Ritz vector
MATRICES	**STANDS FOR**
2DNHm	2-dimensional non-homogeneous problem of size m by m
2DADm	2-dimensional advection diffusion problem of size m by m.
2DSKYm	2-dimensional sky scrapper problem of size m by m.
2DCONSKYm	2-dimensional convective skyscrapper of size m by m.
3DSKYm	3-dimensional skyscrapper problem of size m by m by m.
3DCONSKYm	3-dimensional convective skyscrapper of size m by m by m.
3DANIm	3-dimensional anisotropic problem of size m by m by m.

Table 2: Cost of construction, cost of applying, and the storage requirements for several methods (nnz(K) = no. of non-zeros in matrix K).

Method	Construction	To apply	Storage
RNF(α,β)	$\approx 720\,nx^7$	$\approx 18\,nx^3$	nx^3
RNF(1,0)	$\approx 4nx^3$	$\approx 18\,nx^3$	nx^3
RNF(0,0)	0	$\approx 18\,nx^3$	Null
LFTFD	$\approx \frac{2}{3}nx^7$	$\approx \frac{2}{3}nx^7$	nnz(A)-2nnz(L_3)
ILU(0)	$\approx 7nx^3$	$\approx 14nx^3$	nnz(A)

To study the spectrum of RNF(α,α) preconditioned matrix for different values of the parameter α, we display the spectrum for 30×30 skyscrapper problem in the Figure (1). The spectrum of TFRNF(0,0) preconditioned matrix is displayed in the Figure (2). As we observe in the figure, the parameter has influence on the spectrum; with $\alpha = 0$, the eigenvalues of RNF(0,0) lies between zero and one, which is favorable for the multiplicative preconditioning as seen in section 5.

For a fair comparison, in tables 3 and 4, the number of iterations corresponding to multiplicative combination are doubled. On the other hand, the number of iterations for the additive case are the actual number of iterations, keeping in mind that each of the preconditioner solves can be performed on two different processors on parallel architectures.

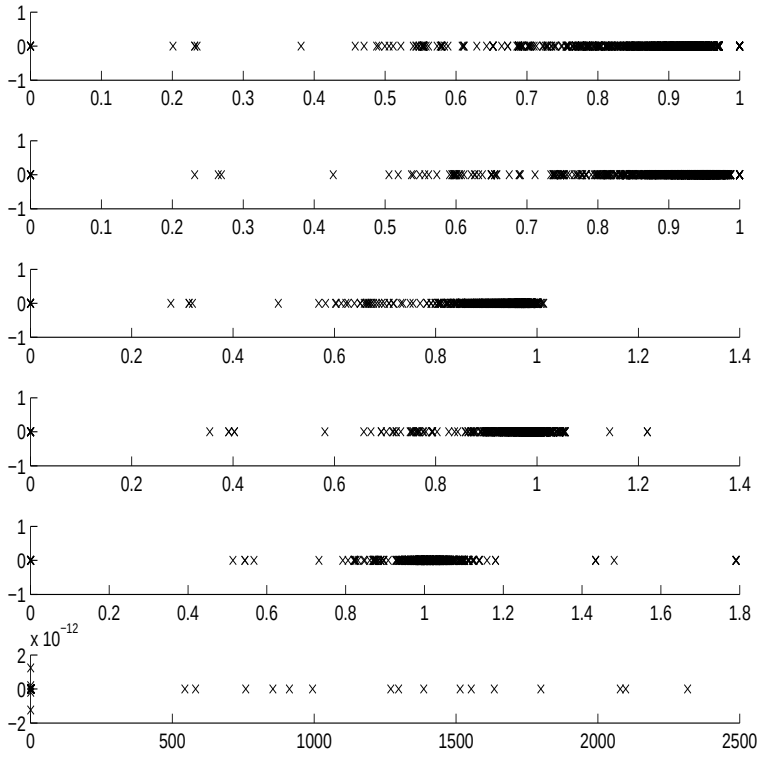


Figure 1: Spectrum plot for RNF(α, α) preconditioned 2D 30×30 skyscrapper problem: (top to bottom) $\alpha = 0$, 0.2, 0.4, 0.6, 0.8, and 1

Comparing the combinative preconditioning TFILU and TFRNF(1,α), we find that the combination TFRNF(1,α) performs much better compared to the combination TFILU or other classical methods (see table 3 and 4) when parameter is chosen close to 1. We notice that the significant number of steps can be saved for two-dimensional problems, but for three-dimensional problems dependence on the parameter is almost negligible. This dependence of iteration number on the parameter for each type of matrices are plotted in Figure (4) for two-dimensional case and in Figure (5) for three-dimensional case

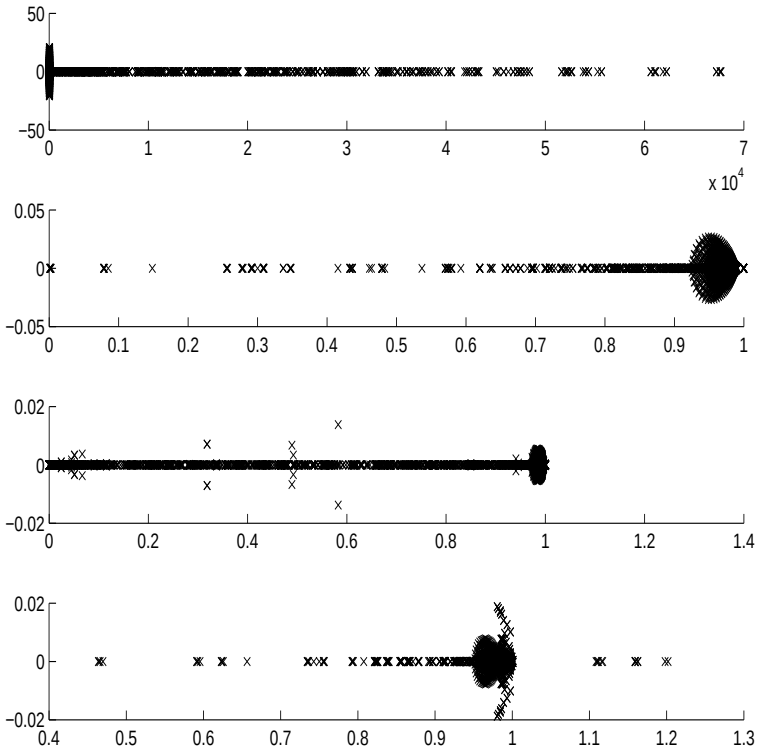


Figure 2: Spectrum plot for 2D 60 × 60 skyscrapper problem: (top to bottom) Spectrum of A followed by spectrum of RNF(0,0), LFTFD, and RNF mult.LFTFD preconditioned matrix

for RNF(1,α), RNF(α,α), for both additive and multiplicative combinations of TFRNF(1,α), and for additive and multiplicative combination of TFRNF(α,α). In these plots, the actual number of iterations are presented. From these figures we observe that for three-dimensional problems, the combinations involving RNF(0,0) and RNF(1,0) remain good parameter independent choices as not much is gained with any value of parameters. Between the combinations involving RNF(0,0) and RNF(1,0), we observe that TFRNF(1,0) performs slightly better than TFRNF(0,0). On the other hand in TFRNF(0,0), RNF(0,0) has no cost of construction and has no storage requirement.

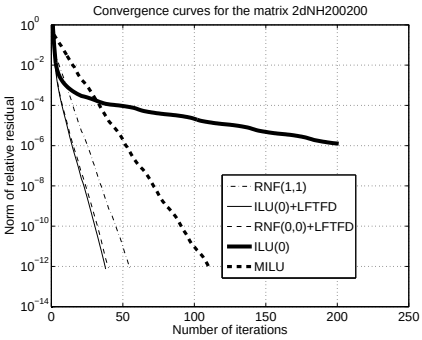
As mentioned before, there are two choices of combinative preconditioning, first is a multiplicative approach, and second is an additive approach. In terms of total number of iterations, the multiplicative approach performs better compared to additive approach. On parallel architecture, each preconditioner solve for the additive case can be done on two different processors, which is an advantage over the multiplicative case.

In Figure (3), we present convergence curves for RNF(1,1), TFILU, TFRNF(0,0), ILU(0), and MILU for one matrix each from the test cases. From these plots, we observe that the convergence behavior of the combination TFRNF(0,0) is very similar to the convergence behavior of TFILU; their curves seem to coin-

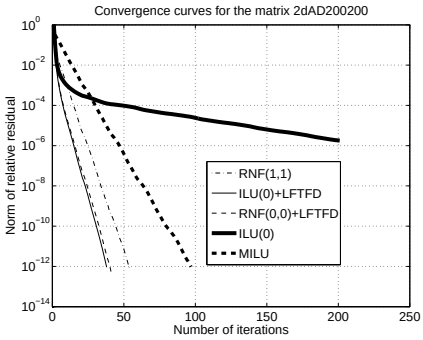
cide with each other; the reason for such similarity may be due to the fact that RNF(0,0) in some sense behave as ILU(0) for all the problems considered (see Table 3).

Other meaningful attempt involves combining RNF(1,1) with LFTFD where the filter vector chosen is the largest Ritz vector. In our test cases we observe that the reason for the failure of RNF(1,1) is the presence of some very large eigenvalues in the spectrum of the preconditioned matrix, see Figure (1). To deflate one of these largest eigenvalues, we chose the filter vector to be the largest Ritz vector obtained after 20 steps of Arnoldi iteration [6] with RNF(1,1) preconditioned matrix. We observe that in Table (4), where the total cost is shown, this combination is not robust enough, the reason being the presence of many other large eigenvalues in the RNF(1,1) preconditioned matrix which cause difficulties in the preconditioned GMRES method. On the other hand RNF(0,0) preconditioned matrix has few small eigenvalues, see Figure (2). For this reason we choose the filter vector corresponding to the smallest Ritz value in magnitude. This combination, i.e., TF(r,s)RNF(0,0) is not robust either, and the reason being the presence of other small eigenvalues in the spectrum of the preconditioned matrix.

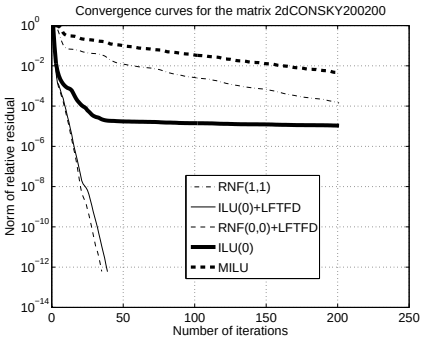
However, in our tests we found that for the combinative preconditioning, the choice of the filter vector as vector of all ones is quite robust. Moreover, it eliminates the cost of forming the filter vector.



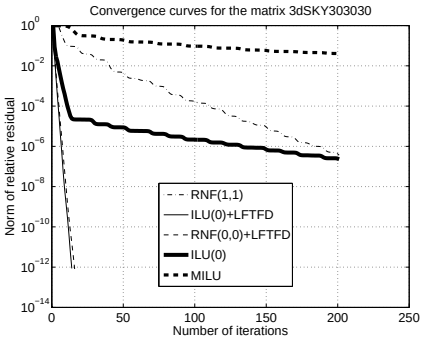
(a) 2D 200×200 non-homogeneous problem



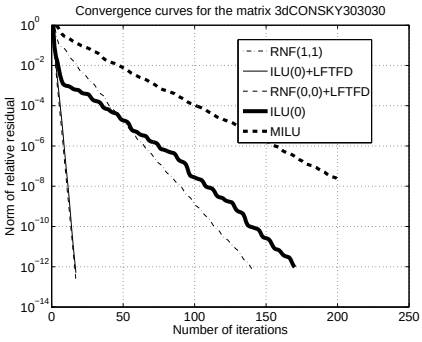
(b) 2D 200×200 advection diffusion problem



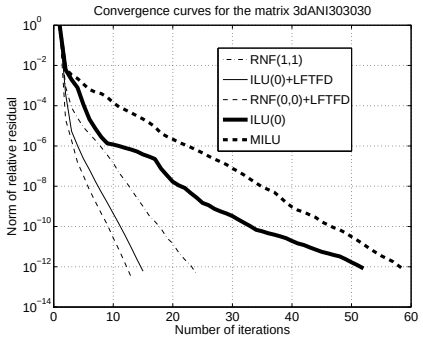
(c) 2D 200×200 convective skyscrapper problem



(d) 3D $30 \times 30 \times 30$ skyscrapper problem

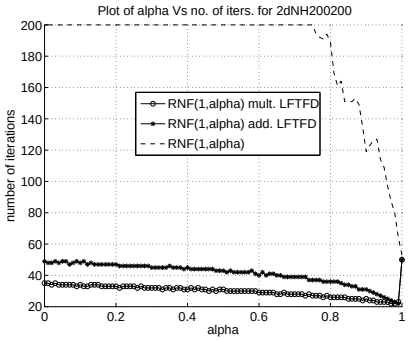


(e) 3D $30 \times 30 \times 30$ convective skyscrapper problem

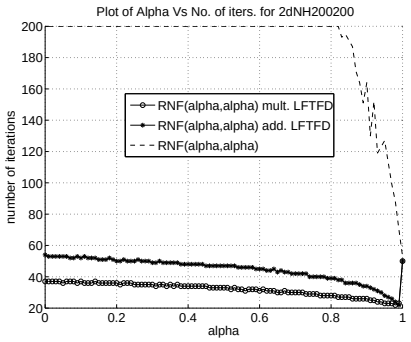


(f) 3D $30 \times 30 \times 30$ anisotropic problem

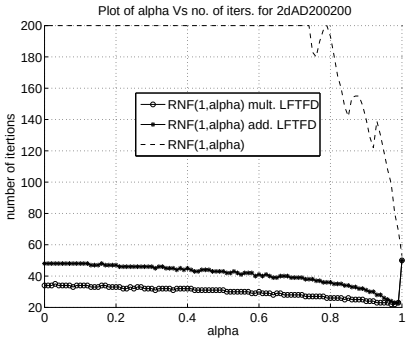
Figure 3: Convergence curves for each of the test problems.



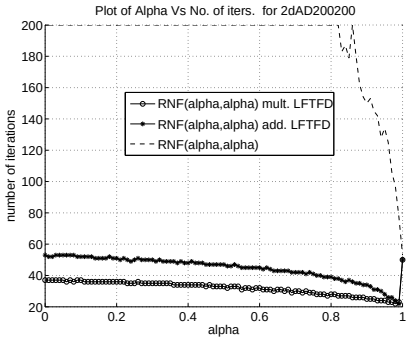
(a) 2D 200×200 non-homogeneous problem.



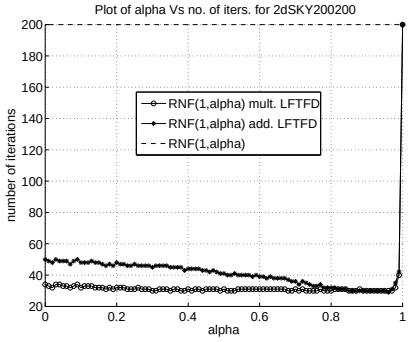
(b) 2D 200×200 non-homogeneous problem.



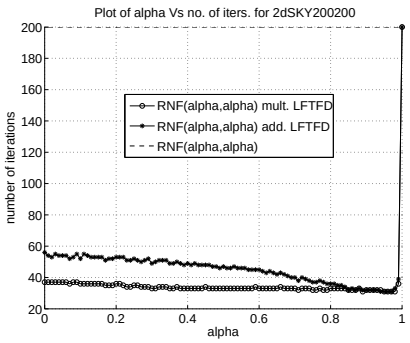
(c) 2D 200×200 advection-diffusion problem.



(d) 2D 200×200 advection-diffusion problem.

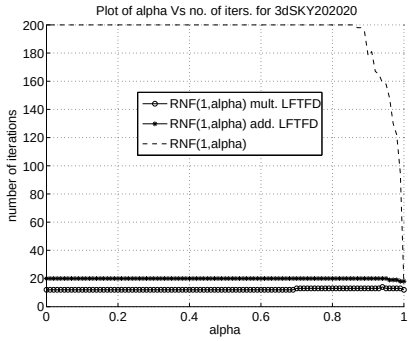


(e) 2D 200×200 skyscrapper problem.

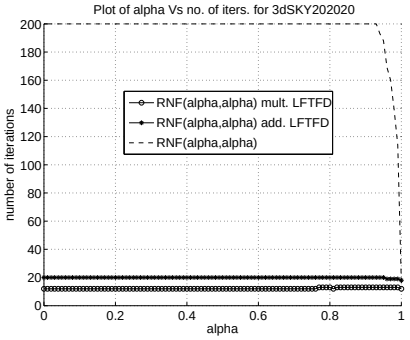


(f) 2D 200×200 skyscrapper problem.

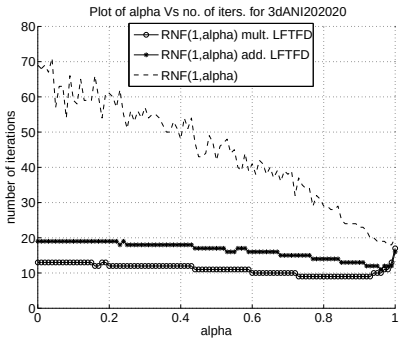
Figure 4: Alpha Vs number of iterations for 2D test cases.



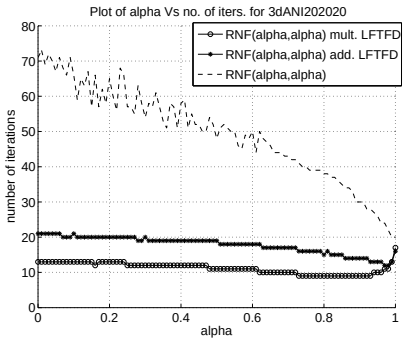
(a) 3D $20\times20\times20$ skyscrapper problem.



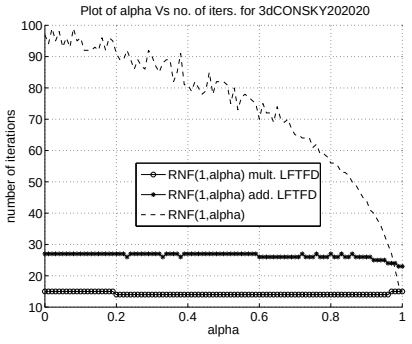
(b) 3D $20 \times 20 \times 20$ skyscrapper problem.



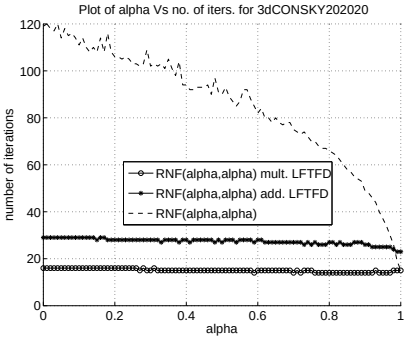
(c) 3D $20\times20\times20$ anisotropic problem.



(d) 3D $20 \times 20 \times 20$ anisotropic problem.



(e) 3D $20\times20\times20$ convective skyscrapper problem.



(f) 3D $20\times20\times20$ convective skyscrapper problem.

Figure 5: Alpha Vs number of iterations for 3D test cases.

Table 3: Test Results, with RNF(1,1), RNF(0,0), and combinations with LFTFD, tol. $= 10^{-12}$. We display the results as: iters(Error in sol.)

Meth./Mat.	ILU	MILU	TFILU		RNF(0,0)	TFRNF(0,0)		RNF(1,1)
			Mult.	Add.		Mult.	Add.	
2dNH100	-	72 (e-12)	56(e-11)	39(e-11)	-	52(e-11)	38(e-11)	37(e-11)
2dNH200	-	109(e-11)	78(e-10)	53(e-10)	-	72(e-11)	53(e-11)	54(e-11)
2dNH300	-	141(e-11)	96(e-10)	66(e-10)	-	88(e-10)	65(e-10)	67(e-10)
2dNH400	-	169(e-11)	110(e-10)	75(e-10)	-	104(e-10)	74(e-10)	78(e-10)
2dAD100	-	65(e-11)	56(e-11)	39(e-11)	-	52(e-11)	38(e-11)	36(e-11)
2dAD200	-	95(e-11)	80(e-10)	54(e-10)	-	74(e-10)	53(e-10)	53(e-11)
2dAD300	-	121(e-11)	96(e-10)	67(e-10)	-	88(e-10)	65(e-10)	67(e-11)
2dAD400	-	146(e-11)	110(e-10)	75(e-10)	-	104(e-10)	74(e-10)	79(e-11)
2dS100	-	-	56(e-08)	36(e-08)	-	50(e-07)	37(e-08)	-
2dS200	-	-	82(e-07)	54(e-07)	-	74(e-07)	55(e-07)	-
2dS300	-	-	100(e-06)	68(e-07)	-	96(e-06)	69(e-06)	110(e-09)
2dS400	-	-	124(e-06)	82(e-06)	-	116(e-06)	86(e-06)	-
2dCS100	-	-	50(e-09)	39(e-09)	-	50(e-09)	41(e-09)	-
2dCS200	-	-	66(e-08)	59(e-09)	-	76(e-08)	67(e-08)	-
2dCS300	-	92(e-10)	78(e-08)	73(e-08)	-	98(e-08)	80(e-08)	63(e-08)
2dCS400	-	-	120(e-08)	101(e-07)	-	138(e-08)	120(e-08)	-
3dCS15	113(e-11)	99(e-12)	22(e-11)	28(e-11)	92(e-10)	34(e-11)	30(e-10)	47(e-11)
3dCS20	88(e-10)	77(e-11)	20(e-10)	23(e-10)	122(e-10)	32(e-10)	29(e-10)	15(e-11)
3dCS30	169(e-09)	-	32(e-10)	27(e-10)	162(e-10)	32(e-10)	29(e-10)	138(e-10)
3dCS40	169(e-09)	-	28(e-10)	25(e-10)	191(e-09)	34(e-10)	30(e-10)	24(e-10)
3dS20	196(e-07)	-	24(e-09)	20(e-09)	-	24(e-09)	20(e-09)	16(e-09)
3dS30	-	-	30(e-08)	22(e-08)	-	26(e-08)	22(e-08)	-
3dS40	-	-	32(e-08)	23(e-08)	-	28(e-08)	23(e-08)	26(e-08)
3dANI20	29(e-08)	43(e-08)	22(e-09)	15(e-08)	72(e-08)	26(e-08)	21(e-08)	20(e-08)
3dANI30	51(e-07)	58(e-08)	24(e-08)	17(e-08)	104(e-07)	28(e-08)	23(e-08)	23(e-08)
3dANI40	55(e-07)	72(e-08)	26(e-08)	18(e-07)	132(e-07)	30(e-08)	23(e-08)	23(e-08)

Table 4: Test Results, with RNF(1,1), RNF(0,0), RNF(1,0), and combinations with LFTFD, tol. $= 10^{-12}$. We display the results as: iters(Error in sol.)

Meth./Mat.	RNF(1,0)	TFRNF(1,0)		TFRNF(.9,.9)		TFRNF(1,0.9)		TF(r,l)RNF(1,1)	TF(r,s)RNF(0,0)
		Mult.	Add.	Mult.	Add.	Mult.	Add.	Mult.	Mult.
2dNH100	168(e-10)	48(e-11)	35(e-11)	36(e-11)	24(e-11)	34(e-11)	21(e-10)	52(e-11)	52(e-11)
2dNH200	-	70(e-11)	48(e-10)	50(e-10)	34(e-11)	48(e-10)	29(e-10)	78(e-11)	-
2dNH300	-	84(e-10)	59(e-10)	62(e-10)	40(e-10)	58(e-10)	36(e-10)	104(e-10)	-
2dNH400	-	96(e-10)	69(e-10)	72(e-10)	46(e-10)	70(e-10)	42(e-10)	122(e-10)	-
2dAD100	168(e-10)	48(e-11)	35(e-11)	36(e-11)	24(e-11)	32(e-11)	21(e-10)	52(e-11)	52(e-11)
2dAD200	-	70(e-11)	49(e-10)	50(e-10)	33(e-10)	48(e-10)	30(e-10)	78(e-11)	394(e-09)
2dAD300	-	84(e-10)	59(e-10)	64(e-10)	40(e-10)	60(e-10)	37(e-10)	100(e-10)	-
2dAD400	-	96(e-10)	67(e-10)	70(e-10)	45(e-10)	70(e-10)	42(e-10)	128(e-10)	-
2dS100	-	44(e-07)	33(e-08)	42(e-07)	22(e-07)	42(e-07)	23(e-07)	-	-
2dS200	-	68(e-07)	51(e-07)	64(e-07)	32(e-07)	60(e-07)	30(e-07)	-	-
2dS300	-	84(e-06)	60(e-07)	58(e-06)	38(e-06)	56(e-07)	34(e-06)	124(e-09)	-
2dS400	-	102(e-06)	72(e-06)	94(e-06)	47(e-06)	92(e-06)	46(e-06)	-	-
2dCS100	-	46(e-09)	39(e-09)	38(e-09)	31(e-09)	38(e-09)	32(e-09)	-	390(e-09)
2dCS200	-	70(e-08)	57(e-08)	52(e-09)	39(e-08)	52(e-08)	39(e-09)	-	-
2dCS300	-	80(e-08)	73(e-08)	56(e-08)	43(e-08)	48(e-08)	39(e-08)	108(e-09)	-
2dCS400	-	124(e-07)	109(e-07)	74(e-08)	59(e-08)	72(e-08)	58(e-08)	-	-
3dCS15	92(e-10)	30(e-11)	28(e-10)	44(e-10)	36(e-11)	48(e-11)	37(e-11)	76(e-11)	154(e-10)
3dCS20	97(e-10)	30(e-11)	27(e-10)	28(e-10)	27(e-11)	28(e-10)	26(e-11)	30(e-11)	302(e-10)
3dCS30	168(e-09)	28(e-10)	26(e-10)	44(e-10)	29(e-10)	44(e-10)	29(e-10)	-	120(e-09)
3dCS40	-	28(e-10)	27(e-10)	26(e-10)	24(e-09)	24(e-09)	24(e-10)	78(e-10)	170(e-09)
3dS20	-	24(e-09)	20(e-09)	26(e-09)	20(e-09)	26(e-09)	20(e-09)	28(e-09)	278(e-08)
3dS30	-	26(e-08)	22(e-08)	50(e-09)	25(e-09)	52(e-09)	26(e-08)	-	188(e-08)
3dS40	-	28(e-08)	22(e-08)	30(e-08)	21(e-08)	32(e-09)	21(e-08)	46(e-09)	-
3dANI20	57(e-08)	26(e-09)	19(e-08)	18(e-08)	14(e-08)	18(e-08)	13(e-08)	28(e-09)	84(e-09)
3dANI30	75(e-07)	28(e-08)	21(e-08)	20(e-08)	15(e-08)	20(e-08)	14(e-08)	30(e-08)	72(e-06)
3dANI40	106(e-07)	28(e-08)	22(e-08)	20(e-07)	15(e-07)	20(e-08)	14(e-08)	32(e-08)	84(e-08)

7 CONCLUSION AND FUTURE DIRECTIONS

We have introduced a relaxed version of NF, and an effective combinative preconditioning is achieved in combination with tangential filtering preconditioners [4]. Compared to ILU(0), it has several important properties. If A is S.P.D. then RNF(0,0)is S.P.D. and leads to a stable splitting. There is no setup cost nor storage requirement. The new preconditioner is particularly useful when multiplicate/additive preconditioning is achieved in combination with tangential filtering preconditioner [4]. With this setup the new preconditioner is as robust as the combination of ILU(0)with tangential filtering preconditioner.

A future work consists in studying other modified version of NF, similar to one suggested by Gustaffson [13], [14] for MILU, where the diagonal is perturbed by an additive term of order $O(h^2)$, which seems to improve the conditioning of modified incomplete factorization methods for certain class of problems.

References

[1] J. R. Appleyard and I. M. Cheshire, *Nested Factorization*, SPE 12264, presented at the Seventh SPE Symposium on Reservoir Simulation, San Francisco, 1983.

[2] J. R. Appleyard, *Proof that simple Colsum modified ILU factorization of matrices arising in Fluid folw problems are not singular*, by John Appleyard, Polyhedron Software Ltd, May 2004. Available online : http://www.polyhedron.com/colsum-proof.

[3] J. R. Appleyard and I. M. Cheshire, *Strictly Triangular L.U formulation of Nested Factorization*, by John Appleyard, Polyhedron Software Ltd, May 2004. Available online : http://www.polyhedron.com/strictlu.

[4] Y. Achdou and F. Nataf, *Low frequency tangential filtering decomposition*, Numer. Linear Algebra Appl., 14, (2007), pp. 129-147.

[5] S. F. Ashby, M. J. Holst, T. A. Manteuffel, and P. E. Saylor *The role of inner product in stopping criteria for conjugate gradient iterations*, BIT., 41, (2001), pp. 26-52.

[6] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide* ,SIAM, Philadelphia, 2000.

[7] M. Benzi, A. M. Tuma, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30, (1999), pp. 305-340.

[8] D. Braess *Towards algebraic multigrid for elliptic problems of second order*, Computing, 55(4), (1995), pp. 379-393.

[9] A. Buzdin, *Tangential decomposition*, Computing., 61, (1998), pp. 257-276.

[10] A. Buzdin and G. Wittum, *Two-frequency decomposition,* Numer. Math., 97, (2004), pp. 269-295.

[11] C. Wagner and G. Wittum, *Adaptive filtering,* Numer. Math., 78, (2004), pp. 305-328.

[12] P. A. Crumpton, P. A. Fjerstad, and J. Berge, *Parallel computing Using ECLIPSE Parallel* Schlumberger, Eclipse, Technical Description 2003A, Chapter 38: Parallel Option. 2003.

[13] Gustafsson I, *A class of first order factorization methods,* BIT 18, pp. 142-56, 1978.

[14] Gustafsson I, *Modified incomplete cholesky(MIC) methods,* In D.Evans, editor, Preconditioning Methods-Theory and Applications, pp. 265-293. Gordon and Breach, New York, 1983.

[15] N. N. Kuznetsova, O. V. Diyankov, S. S. Kotegov, S. V. Koshelev, I. V. Krasnogorov, V. Y. Pravilnikov, and S. Y. Maliassov, *The family of nested factorizations*, Russian Journal of Numerical Analysis and Mathematical Modelling. 22(4) (2007), pp. 393âĂŞ412.

[16] G. Meurant *Computer Solution of Large Linear Systems* North-Holland Publishing Co. Amsterdam, 1999.

[17] J. W. Ruge, K. Stüben *Algebraic multigrid. Multigrid Methods*, Frontiers of Applied Mathematics, vol. 3. SIAM Philadelphia, PA, 1987; 73âĂŞ130.

[18] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS publishing company, Boston, MA, 1996.

[19] K. Stüben, *A review of algebraic multigrid*, J. Comput. Appl. Math. and applied mathematics, 128(1-2), (2001), pp. 281-309, Numerical analysis 2000, vol. VII, Partial differential equations.

[20] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press Inc. San Diego, CA, (2001). With contributions by Brandt A, Oswald P, and Stüben K.

[21] C. Wagner, *Tangential frequency filtering decompositions for symmetric matrices*, Numer. Math., 78, (1997), pp. 119-142.

[22] C. Wagner, *Tangential frequency filtering decompositions for unsymmetric matrices* Numer. Math., 78, (1997), pp. 143-163.

[23] C. Wagner and G. Wittum, *Adaptive filtering,* Numer. Math., 78, (1997), pp. 305-382.

[24] P. Vaněk, M. Brezina, and J. Mandel *Convergence of algebraic multigrid based on smoothed aggregation*, Numerische Mathematik, 88(3), (2001), pp. 559-579.

[25] L. Grigori, F. Nataf, and Q. Niu, *Two sides tangential filtering decomposition*, INRIA TR 6554.



Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 Orsay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399