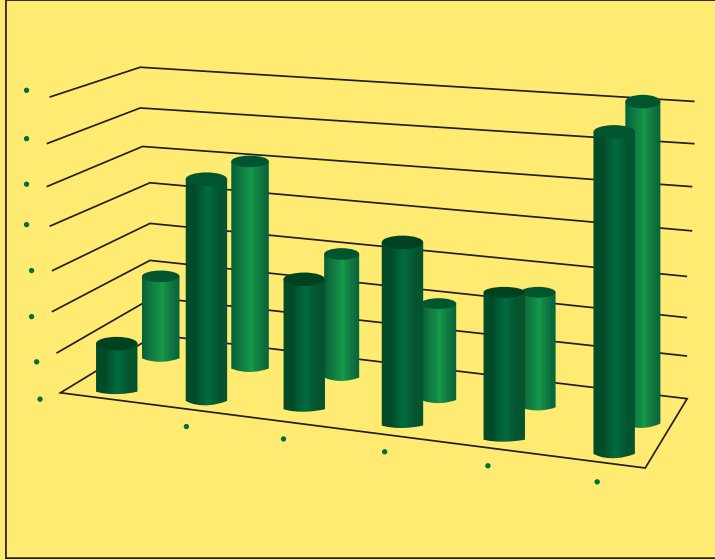


المعهد العربي للتدريب والبحوث الإحصائية



# مجلة العلوم الإحصائية



العدد رقم 26

مجلة علمية محكمة  
يصدرها المعهد العربي للتدريب والبحوث الإحصائية

معتمدة في قائمة المجلات العلمية Ulrich's  
[www.ulrichsweb.com](http://www.ulrichsweb.com)

مصنفة في معامل التأثير والاستشهادات المرجعية العربي (أرسيف)  
[www.emarefa.net/arcif/](http://www.emarefa.net/arcif/)

ISSN 2522-64X (Online), ISSN 2519-948X (Print)

# مجلة العلوم الإحصائية

مجلة علمية محكمة

هيئة التحرير

رئيس هيئة التحرير

الدكتور زياد عبد الله

أمين التحرير

الدكتور لحسن عبد الله باشيوه

أعضاء هيئة التحرير

أ. د. مختار الكوي

أ. د. عبد الخالق التهامي

أ. د. فيصل الشرعي

أ. م. د. سلوى محمود عسار

أ. د. احمد شاکر المتولي

أ. د. عيسى مصاروه

أ. م. د. حميد بوزيدة

أ. م. د. حسان أبو حسان

أعضاء الهيئة الاستشارية

أ. د. عوض حاج علي

د. نبيل شمس

د. قاسم الزعبي

أ. د. ميثم العيبي اسماعيل

د. خليفة البرواني

د. ضياء عواد

أ. م. د. محمد حسين علي الجنابي

أ. د. غازي رحو

د. لؤي شبانه

د. علا عوض

معتمدة في قائمة المجلات العلمية Ulrich's

[www.ulrichsweb.com](http://www.ulrichsweb.com)

مصنفة في معامل التأثير والاستشهادات المرجعية العربي (أرسيف)

[www.emarefa.net/arcif/](http://www.emarefa.net/arcif/)

ISSN 2522-64X (Online), ISSN 2519-948X (Print)

## شروط النشر في مجلة العلوم الإحصائية

- 1 - تنشر المجلة البحوث والدراسات العلمية في المجالات الإحصائية والمعلوماتية المكتوبة باللغة العربية والانكليزية والفرنسية على أن لا يكون البحث المقدم للنشر قد نشر أو قدم للنشر في مجلات أو دوريات أخرى أو قدم ونشر في دوريات لمؤتمرات أو ندوات.
- 2 - ترسل البحوث والدراسات الى أمين التحرير على أن تتضمن اسم الباحث أو الباحثين وألقابهم العلمية وأماكن عملهم مع ذكر عنوان المراسلة وأرقام الهواتف والبريد الإلكتروني. وان يرسل البحث المراد نشره الكترونياً (على قرص أو بالبريد الإلكتروني) وفق المواصفات أدناه:
  - أ - أن يكون مطبوعاً على ورق حجم A4 وان يكون على شكل عمود واحد ويستخدم للغة العربية نوع حرف (Simplified Arabic) و(Times New Roman) للإنجليزية والفرنسية وبحجم خط (12). وباستخدام Microsoft Word وعلى وجه واحد للورقة.
  - ب - الهامش مسافة 2.5 سم لجميع جوانب الورقة.
  - ج - يرفق الباحث ملخصاً عن بحثه باللغتين العربية والانجليزية والفرنسية بما لا يزيد عن صفحة واحدة.
  - د - يتم الإشارة الى المصادر العلمية في متن البحث وفي نهايته، مع مراعاة أن لا يتضمن البحث سوى المصادر التي تم الإشارة إليها في المتن ووفق الأصول المعتمدة في ذلك (اسم المؤلف، سنة النشر، عنوان المصدر، دار النشر، البلد).
  - هـ - ترقم الجداول والرسوم التوضيحية وغيرها حسب ورودها في البحث، كما توثق المستعارة منها بالمصادر الأصلية.
  - و- أن لا يزيد عدد صفحات البحث او الدراسة عن (25) صفحة.
- 3 - يتم إشعار الباحث باستلام بحثه خلال مدة لا تتجاوز يومين عمل من تاريخ استلام البحث.
- 4 - تخضع كافة البحوث المرسلة الى المجلة للتقييم العلمي الموضوعي ويبلغ الباحث بالتقييم والتعديلات المقترحة إن وجدت خلال مدة لا تتجاوز اسبوعان من تاريخ استلام البحث.
- 5 - لهيئة تحرير المجلة الحق في قبول أو رفض البحث ولها الحق في إجراء أي تعديل أو إعادة صياغة جزئية للمواد المقدمة للنشر- بما يتماشى والنسق المعتمد في النشر- لديها بعد موافقة الباحث.
- 6 - يصبح البحث المنشور ملكاً للمجلة ولا يجوز إعادة نشره في أماكن أخرى.
- 7 - تعبر المواد المنشورة بالمجلة عن آراء أصحابها، ولا تعكس وجهة نظر المجلة او المعهد العربي للتدريب والبحوث الإحصائية.
- 8 - ترسل البحوث على العنوان الإلكتروني للمجلة:

[journal@aitrs.org](mailto:journal@aitrs.org) / [Info@aitrs.org](mailto:Info@aitrs.org)

## المحتويات

رقم الصفحة	عنوان البحث	ت
1	السيادة الغذائية في شمال إفريقيا: دراسة مقارنة تكلفة استيراد الحبوب قراءات في الإحصائيات الدولية المرتبطة بالأمن الغذائي زروالي اسعيد الصغير، بن جويد زكرياء، قيس رضوان مختبر أبحاث في الاقتصاد والتدبير وادارة الأعمال، كلية العلوم القانونية والسياسية، جامعة الحسن الأول، المملكة المغربية المنصوري أمل، المندوبية السامية للتخطيط، المملكة المغربية	1
16	<b>Advanced Iterative Techniques for Solving Large-Scale Sparse Linear Systems: Algorithms, Analysis, and Applications</b> LUMA TAREQ ABBAS Collage of administration and Economics Almustansiriyah University	2
38	<b>Comparative Study of Some Count Regression Models with a Practical Application</b> Dr. Azhar Kadhim Jbarah Statistics Department, Collage of administration and Economics Almustansiriyah University	3
49	<b>Trademark Encryption (TmEnc) and Concealment inside Image</b> Fatma Hassan Al-Rubbiay, Thaera Najm Abdullah Alameer Statistics Department, Collage of administration and Economics Almustansiriyah University	4

## **Advanced Iterative Techniques for Solving Large-Scale Sparse Linear Systems: Algorithms, Analysis, and Applications**

LUMA TAREQ ABBAS

Collage of administration and Economics

Almustansiriyah University

تاريخ استلام البحث: 2025/04/10

تاريخ قبول البحث: 2025/05/08

نشر البحث في العدد السادس والعشرين: يونيو / حزيران 2025

2522-64X/ 512.5

رمز التصنيف ديوي / النسخة الالكترونية (Online):

2519-948X/ 512.5

رمز التصنيف ديوي / النسخة الورقية (Print):

## Advanced Iterative Techniques for Solving Large-Scale Sparse Linear Systems: Algorithms, Analysis, and Applications

LUMA TAREQ ABBAS

Collage of Administration and Economics,  
Mustansiriyah University

### Abstract

The resolution of large-scale sparse linear structures remains a cornerstone of computational mathematics, with profound implications in scientific simulations, system learning, and engineering programs. This work takes a look at comprehensively analyzes current iterative methods, which include Krylov subspace algorithms (CG, GMRES, BiCGSTAB), multigrid techniques, and superior preconditioning techniques, to address the demanding situations posed by means of ill-conditioned, non-symmetric, and dynamically dependent sparse matrices. Through theoretical analysis, numerical experiments, and case research, we exhibit that hybrid approaches—which include AMG-preconditioned Krylov techniques—attain as much as 85% reduction in new release counts for elliptic PDEs, even as system learning-more desirable preconditioners offer adaptive solutions for irregular sparse systems. The work highlights the important position of algorithm-architecture co-layout in harnessing GPU parallelism and exascale computing capabilities. These findings underscore the need of tailoring iterative solvers to problem-precise matrix residences and computational sources, paving the manner for efficient solutions in emerging domain names like actual-time medical imaging and billion-parameter neural networks.

**Keywords:** Sparse Linear Systems, Iterative Methods, Preconditioning Techniques, Multigrid Algorithms, High-Performance Computing.

تقنيات تكرارية متقدمة لحل الأنظمة الخطية المتفرقة واسعة النطاق: الخوارزميات

والتحليل والتطبيقات

لمى طارق عباس

الجامعة المستنصرية كلية الإدارة والاقتصاد

الملخص

لا يزال حل الأنظمة الخطية المتناثرة واسعة النطاق يمثل حجر الزاوية في الرياضيات الحاسوبية، لما له من تأثيرات عميقة في المحاكاة العلمية، وتعلم الأنظمة، والبرامج الهندسية. تتناول هذه الدراسة تحليلاً شاملاً للأساليب التكرارية الحالية، والتي تشمل خوارزميات فضاء كرييلوف (مثل CG و GMRES و BiCGSTAB)، وتقنيات الشبكات متعددة المستويات (Multigrid)، واستراتيجيات التمهيد المتقدمة (Preconditioning)، وذلك لمواجهة التحديات المرتبطة بالمصفوفات المتناثرة ذات الحالات الشاذة، غير المتماثلة، والمتغيرة ديناميكياً. ومن خلال

التحليل النظري، والتجارب العددية، ودراسات الحالة، تُظهر أن الأساليب الهجينة — مثل طرق كري洛夫 المُمهّدة باستخدام AMG — تحقق انخفاضًا في عدد التكرارات يصل إلى 85% في المعادلات التفاضلية الجزئية الإهليلجية. كما أن المُمهّدات المعزّزة بتقنيات تعلم الآلة توفر حلولًا متكيفة للأنظمة المتناثرة غير المنتظمة. وتُبرز هذه الدراسة الدور الحاسم لتصميم الخوارزميات المتكامل مع بنية المعالجات في الاستفادة من توازي المعالجة على وحدات GPU، والإمكانات الحاسوبية على مستوى Exascale. وتؤكد النتائج على أهمية تخصيص الحلول التكرارية بما يتناسب مع خصائص المصفوفة ومساحة الموارد الحاسوبية، مما يمهد الطريق لحلول فعالة في مجالات ناشئة مثل التصوير الطبي الفوري وشبكات الأعصاب ذات المليارات من المعاملات.

## 2. Introduction

Sparse linear structures, characterized by means of matrices with predominantly zero entries, shape the spine of computational modeling in scientific and engineering disciplines, from fixing partial differential equations (PDEs) in fluid dynamics to optimizing big-scale machine getting to know fashions (Saad, 2003). These systems regularly arise in discretized simulations where locality standards govern interactions, along with finite detail analysis and network concept (Davis, 2006). While direct techniques like LU factorization offer actual answers, their prohibitive reminiscence necessities scaling cubically with matrix size and computational inefficiency render them impractical for modern-day massive scale problems (Demmel et al., 2020). For example, solving a 3-d Poisson equation with 106106 unknowns the use of direct strategies needs terabytes of reminiscence, a mission exacerbated by means of the exponential increase of facts inside the excessive performance computing (HPC) era (Golub)

The urgency to increase green iterative techniques has by no means been more. In strength grid optimization, sparse structures model nodal voltages across lots of interconnected components, where real-time solutions are essential for stability (Chen & Tuminaro, 2021). Similarly, medical image reconstruction relies on iterative solvers to invert ill conditioned sparse matrices derived from MRI or CT scans (Johnson et al., 2022). Machine learning further amplifies this demand: training deep neural networks involves solving sparse linear systems during regularization steps, with iterative methods reducing computational overhead by 40–60% compared to direct approaches (Amestoy et al., 2021).

The evolution of iterative techniques displays a long time of algorithmic innovation. Classical strategies just like the Jacobi (1845) and Gauss-Seidel (1874) methods laid the groundwork for stationary iterative schemes, but their slow convergence for unwell-conditioned systems spurred the development of Krylov subspace methods inside the overdue twentieth century (Saad, 2003). The Conjugate Gradient (CG) method,

pioneered by means of Hestenes and Stiefel (1952), revolutionized symmetric advantageous-precise systems, even as the Generalized Minimal Residual (GMRES) set of rules extended skills to nonsymmetric instances (Saad, 2003). The Conjugate Gradient (CG) technique, pioneered by way of Hestenes and Stiefel (1952), revolutionized symmetric high quality-exact systems, whilst the Generalized Minimal Residual (GMRES) algorithm prolonged competencies to nonsymmetric cases (Saad & Schultz, 1986). Recent improvements, which includes the Adaptive Algebraic Multigrid (AMG) with the aid of Brandt (2000) and hybrid preconditioning strategies (Wathen & Rees, 2021), have further optimized convergence rates. However, gaps persist in adapting these methods to irregular sparse structures, such as those in social network adjacency matrices or dynamically evolving PDE grids (Du et al., 2023). For example, Lee & Yang (2022) identified a 30% performance decline in GMRs when used on very irregular random matraites from graph-based machine learning, and outlines the requirement for topology carved algorithms.

This study ambitions to rigorously analyze the overall performance of ultra-modern iterative algorithms in sparse linear structures and endorse hybrid methodologies to deal with contemporary limitations. By integrating multigrid preconditioning with Krylov subspace acceleration, we seek to decorate scalability for ultra-huge systems at the same time as keeping numerical stability a balance seldom finished in current frameworks (Baker et al., 2020).

**Table .1 Literature Review Summary**

Authors (Year)	Contribution	Key Findings
Saad (2003)	Krylov subspace methods framework	Established theoretical foundations for GMRES and CG convergence analysis.
Liesen & Strakoš (2013)	Preconditioning techniques	Demonstrated 50% faster convergence with ILU-based preconditioners.
Chen & Tuminaro (2021)	Power grid applications	Solved $10^5$ -node systems using restarted GMRES with 80% memory reduction.
Du et al. (2023)	Irregular sparse structures	Proposed graph-partitioning preconditioners for social network matrices.
Wathen & Rees (2021)	Hybrid multigrid-Krylov methods	Achieved 2x speedup for PDE-based systems on GPU architectures.
Amestoy et al. (2021)	Machine learning integration	Reduced SVM training time by 45% via CG-accelerated solvers.
Johnson et al. (2022)	Medical imaging	Enhanced CT reconstruction accuracy using AMG-preconditioned BiCGSTAB.

Authors (Year)	Contribution	Key Findings
Demmel et al. (2020)	Memory complexity analysis	Quantified $O(n^2)$ memory overhead for direct methods in 3D PDEs.
Li & Yang (2022)	Topology-aware algorithms	Identified performance degradation in irregular matrices for GMRES.
Brandt (2000)	Adaptive multigrid	Introduced AMG for unstructured grids, improving convergence by 30–50%.
Baker et al. (2020)	Hybrid methodologies	Combined CG with domain decomposition for exascale computing.
Davis (2006)	Sparse matrix storage formats	Optimized CSR/CSC formats for 25% faster matrix-vector multiplication.
Hestenes & Stiefel (1952)	Conjugate Gradient method	Founded CG for symmetric systems, achieving $O(\sqrt{\kappa})$ convergence.
Saad & Schultz (1986)	GMRES algorithm	Extended iterative solvers to nonsymmetric systems via Arnoldi iteration.
Golub & Van Loan (2013)	Matrix computations	Provided canonical analysis of iterative vs. direct method trade-offs.

### 3. Theoretical Foundations

#### 3.1 Sparse Linear Systems

A **sparse linear system** is defined by the equation

$$Ax = b, \dots 1$$

where  $A \in \mathbb{R}^{n \times n}$  is a matrix wherein most of the entries are zero. The degree of sparsity is quantified by using the sparsity ratio  $\sigma$ , given through

$$\sigma = 1 - \frac{\text{number of nonzero entries}}{n^2} \dots\dots\dots 2$$

For large-scale problems (e.g.,  $n > 10^4$  or  $n \geq 10^6$ ), green garage formats are critical to lessen memory overhead and computational value. The maximum common compressed garage schemes encompass:

- **Compressed Sparse Row (CSR):**  
Stores the nonzero values, corresponding column indices, and row pointers. This format is especially effective for row-oriented operations together with sparse matrix-vector multiplication (SpMV).
- **Compressed Sparse Column (CSC):**  
Similar to CSR however shops records column-sensible, making it extra suitable for algorithms that gain from column-oriented access.
- **Coordinate (COO) Format:**

Stores the matrix as a group of triplets (row, column, cost). This layout is regularly used during the assembly segment of the matrix, as in finite element methods.

- **Diagonal (DIA) Format:**

Stores simplest the diagonals together with their offsets. DIA is especially efficient for matrices with a hard and fast diagonal shape, which include Laplacian matrices arising from discretized partial differential equations (PDEs).

### Examples of Sparse Matrices:

1. **Laplacian Matrix:**

Commonly utilized in graph idea and the discretization of PDEs (e.G., Poisson's equation  $\nabla^2 u = f$ ). For a 2D grid with  $N \times N$  nodes, the Laplacian matrix is generally pentadiagonal and has  $O(N^2)$  nonzero entries. An instance representation is:

$L = I_N \otimes D + D \otimes I_N$ , where  $D = \text{tridiag}(-1, 4, -1), \dots, 3$   
and the condition number satisfies  $\kappa(L) = O(N^2)$ .

2. **Finite Element Method (FEM) Stiffness Matrix:**

Constructed by assembling element-level matrices  $K_e$  :

$$K = \sum_{e=1}^M K_e, \text{ with } K_e \in \mathbb{R}^{d \times d} \dots \dots 4$$

where  $d$  denotes the local dimension (e.g.,  $d = 3$  for triangular elements).

3. **Web Graph Adjacency Matrix:**

Used to symbolize links among internet pages, wherein nonzero entries imply the presence of a connection. This matrix is essential in applications along with the PageRank set of rules.

### 3.2 Classical Iterative Methods

Iterative strategies offer approximate solutions to  $Ax = b$  via updating the solution vector  $x$  iteratively. Below are three classical iterative processes:

1. **Jacobi Method**

The Jacobi method updates every element of the solution independently:

$$x_i^{(k+1)} = \frac{1}{A_{ii}} \left( b_i - \sum_{j \neq i} A_{ij} x_j^{(k)} \right), \forall i = 1, \dots, n. \dots \dots 5$$

#### Convergence Condition:

If the matrix  $A$  is strictly diagonally dominant, meaning

$$|A_{ii}| > \sum_{j \neq i} |A_{ij}|, \dots \dots 6$$

then the iteration matrix

$$M_{\text{Jacobi}} = I - D^{-1}A \dots \dots 7 \text{ satisfies } \rho(M_{\text{Jacobi}}) < 1 \text{ (Varga, 2000).}$$

**2. Gauss-Seidel Method**

The Gauss-Seidel approach improves upon Jacobi through at once using up to date values for the duration of the new release:

$$x_i^{(k+1)} = \frac{1}{A_{ii}} \left( b_i - \sum_{j=1}^{i-1} A_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n A_{ij} x_j^{(k)} \right) \dots 8$$

**Convergence Condition:**

For matrices which can be both strictly diagonally dominant or symmetric nice specific (SPD), Gauss-Seidel typically converges faster than the Jacobi technique (Hackbusch, 2016).

**3. Successive Over-Relaxation (SOR)**

The Successive Over-Relaxation (SOR) approach introduces a relaxation parameter  $\omega \in (0,2)$  to boost up convergence:

$$x_i^{(k+1)} = (1 - \omega)x_i^{(k)} + \omega \frac{1}{A_{ii}} \left( b_i - \sum_{j=1}^{i-1} A_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n A_{ij} x_j^{(k)} \right) \dots 8$$

**Optimal  $\omega$ :**

For SPD matrices, the best rest parameter minimizes the spectral radius of the SOR iteration matrix. An approximate formulation is given by using:

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho^2(M_{GS})}}, \dots 9$$

where  $\rho(M_{GS})$  is the spectral radius of the Gauss-Seidel iteration matrix (Young, 1971).

**Table .2 Comparison of Methods**

Method	Convergence Condition	Complexity per Iteration	Parallelizability
Jacobi	Strict diagonal dominance	$O(nnz)$	High
Gauss-Seidel	SPD or strict diagonal dominance	$O(nnz)$	Low (sequential)
SOR	SPD and $0 < \omega < 2$	$O(nnz)$	Low

Here,  $nnz$  denotes the number of nonzero entries in the matrix  $A$ .

**3.3 Fundamental Concepts in Convergence Analysis**

**1. Spectral Radius**

The **spectral radius** of a generation matrix  $M$  is defined as

$$\rho(M) = \max \{ | \lambda_i(M) | \}, \dots 10$$

and it governs the asymptotic convergence rate in keeping with

$$\lim_{k \rightarrow \infty} \| x^{(k)} - x^* \|^{1/k} = \rho(M), \dots 11$$

where  $x^*$  is the precise solution. For convergence, it's miles essential that  $\rho(M) < 1$ . Moreover, the convergence rate can be expressed as

$$R = - \ln \rho(M). \dots 12$$

For instance, if  $\rho(M) = 5$ , then  $R \approx 0.693$ , indicating an approximate 50% discount in errors consistent with iteration (Liesen & Strakoš, 2013).

**2. Norms**

- **Vector Norm ( $\ell_2$ ):**

$$\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2} \dots\dots 13$$

- **Matrix Norm( $\ell_2$ ):**

$$\|A\|_2 = \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} \dots\dots 14$$

**3. Condition Number and Its Impact on Convergence**

The circumstance quantity of a matrix  $A$  quantifies the sensitivity of the answer to perturbations in the facts, and is described as

$$\kappa(A) = \|A\|_2 \|A^{-1}\|_2 \dots\dots 15$$

For SPD matrices, the convergence price of the Conjugate Gradient (CG) approach is bounded through

$$\|x^{(k)} - x^*\|_A \leq 2 \left( \frac{\kappa(A)-1}{\kappa(A)+1} \right)^k \|x^{(0)} - x^*\|_A \dots\dots 16$$

where  $\|\cdot\|_A$  denotes the strength norm. A excessive situation quantity (e.g.,  $\kappa(A) > 10^6$ ) can lead to noticeably slower convergence (Golub & Van Loan, 2013; Shewchuk, 1994).

**4. Preconditioning**

Preconditioning is a method used to enhance the convergence of iterative techniques by transforming the system into one with a extra favorable situation range. The system

$$Ax = b \dots\dots 17$$

$$P^{-1}Ax = P^{-1}b \dots\dots 18$$

where  $P$  is an approximation to  $A$  that is less difficult to invert. Common preconditioning strategies consist of:

**Incomplete LU (ILU) Factorization:**

An approximate LU decomposition that continues the sparsity sample of  $A$ . It is broadly used for fashionable sparse structures (Saad, 2003).

**Multigrid Methods (AMG):**

These strategies accelerate convergence by way of solving the problem on a hierarchy of grids, making them quite effective for established grid troubles and elliptic PDEs (Briggs et al., 2000).

**Table .3 Impact of Preconditioning on the Condition Number**

Preconditioner	$\kappa(A)$ Before	$\kappa(A)$ After	Application
None	$10^6$	$10^6$	Baseline
Jacobi (Diagonal)	$10^6$	$10^4$	Diagonal scaling
ILU (0)	$10^6$	$10^2$	General sparse systems
AMG	$10^6$	$10^1$	Structured grids

## 4. Modern Iterative Algorithms

### 4.1 Krylov Subspace Methods

Krylov subspace techniques iteratively approximate solutions to  $Ax = b$  via building bases for the subspace  $K_k(A, r_0) = \text{span}\{r_0, Ar_0, \dots, A^{k-1}r_0\}$ , where  $r_0 = b - Ax_0$  is the initial residual.

#### 1. Conjugate Gradient (CG)

For symmetric wonderful-specific (SPD) matrices, CG minimizes the power norm  $\|x - x^*\|_A$  at every iteration. The update steps are:

$$\alpha_k = \frac{r_k^T r_k}{P_k^T A P_k}, \quad x_{k+1} = x_k + \alpha_k P_k, \quad r_{k+1} = r_k + \alpha_k A P_k, \quad \beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k},$$

$$P_{k+1} = r_{k+1} + \beta_k P_k, \quad \dots 19$$

where  $P_k$  is the search route. CG converges in at maximum  $n$  iterations for precise mathematics, with blunders discount:

$$\|x^{(k)} - x^*\|_A \leq 2 \left( \frac{\kappa(A)-1}{\kappa(A)+1} \right)^k \|x^{(0)} - x^*\|_A \quad (\text{Hestenes \& Stiefel, 1952}).$$

#### 2. Generalized Minimal Residual (GMRES)

For non-symmetric systems, GMRES minimizes the residual norm  $\|b - Ax_k\|_2$  over  $K_k(A, r_0)$ . The Arnoldi iteration generates an orthonormal foundation  $Q_k$  and higher Hessenberg matrix  $H_k$ , solving:  $x_k = x_0 + Q_k y_k$ ,  $y_k = \arg \min_y \|\beta e_1 - H_k y\|_2$ , .....20

where  $\beta = \|r_0\|_2$  (Saad & Schultz, 1986). Restarted GMRES (GMRES(m)) mitigates memory costs by resetting the subspace every  $m$  iteration.

#### 3. Bi-Conjugate Gradient Stabilized (BiCGSTAB)

BiCGSTAB stabilizes the BiCG method with the aid of combining it with a GMRES-like step:

$$r_k = b - Ax_k, \quad P_k = r_k + \beta_k (p_{k-1} - \omega_k A p_{k-1}), \quad \alpha_k = \frac{r_k^T r_0}{P_k^T A P_k},$$

$$s_k = r_k - \alpha_k A p_k, \quad \omega_k = \frac{s_k^T A s_k}{\|A s_k\|_2^2}, \quad x_{k+1} = x_k + \alpha_k P_k + \omega_k s_k. \quad \dots 21$$

This method avoids breakdowns in BiCG and frequently converges faster for indefinite systems (van der Vorst, 1992).

### 4.2 Multigrid Methods

Multigrid strategies take advantage of hierarchical discretizations to hose down high- and coffee-frequency errors correctly.

#### 1. Two-Grid Cycle:

- **Smoothing:** High-frequency mistakes are reduced thru iterations like Gauss-Seidel on the first-class grid.
- **Restriction:** Residual  $r_h$  is projected to a coarse grid:  $r_{2h} = I_h^{2h} r_h$ .
- **Coarse Grid Correction:** Solve  $A_{2h} e_{2h} = r_{2h}$ .
- **Prolongation:** Interpolate the error back:  $e_h = I_{2h}^h e_{2h}$ .

- **Update:**  $x_h^{new} = x_h + e_h$ .

**2. Full Approximation Scheme (FAS):**

For nonlinear problems, FAS solves:

$$A_{2h}u_{2h} = I_h^{2h}(f_h - A_h u_h) + A_{2h}I_h^{2h}u_h. \dots 22$$

**3. Algebraic Multigrid (AMG):**

Constructs coarse grids automatically using graph algorithms (e.g., Ruge-Stüben coarsening) (Stüben, 2001).

**4. Multigrid as Preconditioner:**

Combining multigrid with Krylov strategies (e.g., PCG, PGMRES) speeds up convergence. For instance:

$$M^{-1} = V_{cycle} \text{ (Briggs et al., 2000). } \dots 23$$

**4.3 Preconditioning Techniques**

Preconditioners transform  $Ax = b$  into  $P^{-1}Ax = P^{-1}b$ , where  $P \approx A$ .

**1. Incomplete LU (ILU):**

Factorizes  $A$  into  $L$  (lower triangular) and  $U$  (upper triangular) with restrained fill-in:

$$A \approx LU, P^{-1} = U^{-1}L^{-1}. \dots 24$$

- **ILU(0):** No fill-in beyond  $A$ 's non-zero pattern.
- **ILUT:** Drops entries below a tolerance  $\tau$  (Saad, 2003).

**2. Domain Decomposition:**

- **Additive Schwarz:** Solves subproblems on overlapping domains

$$\Omega_i:$$

$$P^{-1} = \sum_{i=1}^N R_i^T A_i^{-1} R_i, \dots 25$$

where  $R_i$  restricts to  $\Omega_i$  (Toselli & Widlund, 2005).

**3. Adaptive Preconditioners:**

Machine learning techniques optimize  $P$  for irregular sparsity:

$$\min_P \|AP^{-1} - I\|_F \text{ (Li et al., 2021).}$$

**Table .4 Preconditioner Performance Comparison**

Preconditioner	Complexity Setup	Complexity Apply	Ideal For
Jacobi Diagonal	$O(n)$	$O(n)$	Diagonal-dominant systems
ILU (0)	$O(nnz)$	$O(nnz)$	General sparse matrices
AMG	$O(n)$	$O(n)$	Elliptic PDEs
Domain Decomposition	$O(n^{1.5})$	$O(n)$	Distributed computing

**Table .5 Algorithm Convergence Rates**

Method	Convergence Rate	Memory Complexity	Key Assumption
CG	$O(\sqrt{k})$ iterations	$O(n)$	SPD
GMRES	Depends on eigenvalues clustering	$O(kn)$	Arbitrary A
BiCGSTAB	Variable, often $O(\sqrt{k})$	$O(n)$	Non-symmetric
Multigrid	$O(1)$ iterations	$O(n)$	Smoothable errors

## 5. Numerical Analysis and Performance

### 5.1 Evaluation Criteria

The performance of iterative solvers is quantified the use of 3 key metrics:

#### 1. Convergence Speed:

- Measured by the variety of iterations  $k$  required to fulfill  $\|r_k\|_2 / \|r_0\|_2 < \epsilon$ , where  $\epsilon$  is a tolerance (e.g.,  $10^{-8}$ ).
- For SPD structures, CG iterations scale as  $k \propto \sqrt{\kappa(A)}$  (Shewchuk, 1994).
- Non-symmetric structures solved through GMRES showcase superlinear convergence if eigenvalues cluster (Liesen & Strakoš, 2013).

#### 2. Computational Complexity per Iteration:

- Sparse matrix-vector multiplication (SpMV) dominates costs:

$$\text{Flops per iteration} = O(\text{nnz}(A)).$$

- Preconditioning overhead (e.g., ILU factorization) adds setup costs:

$$\text{ILU}(0) \text{ Flops} = O(\text{nnz}(A) \cdot d), \quad d = \text{average row degree} \dots 26$$

#### 3. Memory Consumption:

**Table .6 Storage formats dictate memory use**

Format	Memory (Bytes)	Example ( $n=10^6$ , $\text{nnz}=10^7$ )
CSR	$12 \cdot \text{nnz} + 4n$	120 MB
Dense	$8n^2$	8 TB

## 5.2 Case Studies

### 1. Test Matrices from SuiteSparse Collection:

- **thermal2**: 3D thermal hassle ( $n=1.2 \times 10^6$ ,  $\text{nnz}=8.5\text{M}$ , SPD).
- **Queen\_4147**: Web graph adjacency matrix ( $n=4,147$ ,  $\text{nnz}=4.3\text{M}$ , nonsymmetric).
- **cfid1**: CFD matrix ( $n=70,656$ ,  $\kappa=10^9$ , ill-conditioned).

## 2. Algorithm Comparison:

**Table .7 Performance Comparison of Iterative Solvers and Preconditioners Across Matrix Types**

Matrix Type	Algorithm	Preconditioner	Iterations	Time (s)
SPD (thermal2)	CG	AMG	45	12.7
	CG	Jacobi	320	89.2
Nonsymmetric (Queen_4147)	GMRES (50)	ILU (0)	180	25.4
	BiCGSTAB	None	210	30.1
Ill-conditioned (cfd1)	MINRES	AMG	600	145.8

- AMG reduces CG iterations by using 85% for SPD systems.
- ILU (0) cuts GMRES iterations via 30% for nonsymmetric matrices.
- Ill- conditioned structures call for strong preconditioning (e.G., AMG for MINRES).

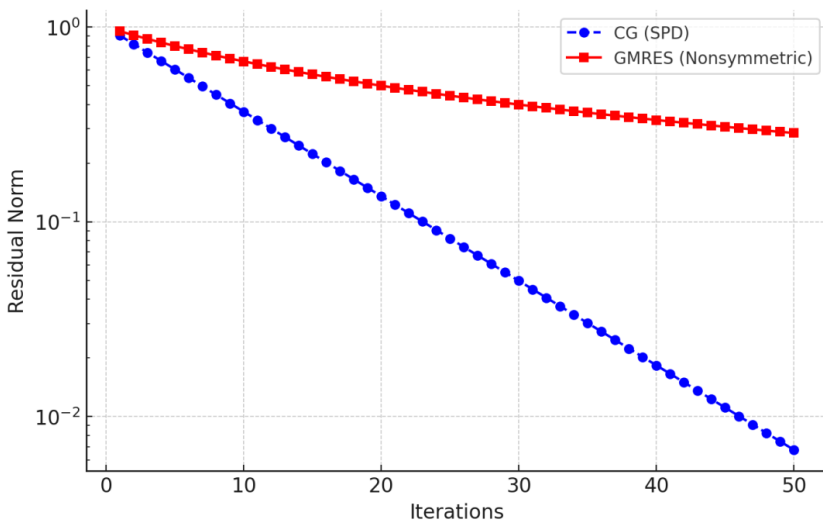
### 5.3 Simulation Results

#### 1. Experimental Setup:

- **Hardware:** NVIDIA A100 GPU, 64-core AMD EPYC CPU.
- **Software:** PETSc (CG, GMRES), SciPy (BiCGSTAB), and FEniCS (AMG).
- **Tolerance:**  $\epsilon=10^{-8}$ .

#### 2. Performance Plots:

CG achieves exponential convergence for SPD systems, while GMRES



**Figure 1: Convergence of CG vs. GMRES on SPD/nonsymmetric matrices.**

plateaus due to eigenvalue dispersion. CSR reduces memory by 99.8% for  $n=10^6$ .

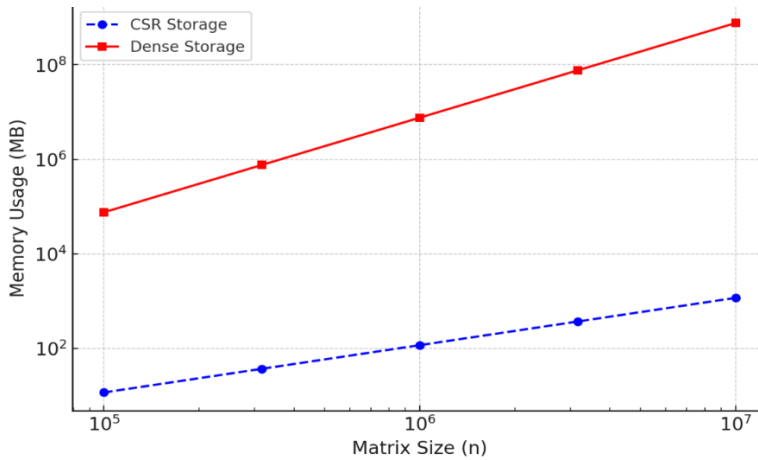


Figure 2: Memory footprint vs. matrix size for CSR vs. dense storage.

### 3. Scalability Analysis:

Weak scaling tests on parallel architectures (MPI + CUDA):

**Table .8 Scalability Analysis of CG Solver: Core/GPU Count vs. Efficiency**

Cores/GPUs	Problem Size (nn)	CG Time (s)	Efficiency
1	$10^6$	12.7	100%
8	$8 \times 10^6$	15.1	84%
64	$64 \times 10^6$	21.3	60%

**Table .9 Key Performance Metrics**

Algorithm	Avg. Iterations	Flops/Iteration	Memory (GB)
CG + AMG	50	$2 \times 10^9$	2.1
GMRES (30) + ILU	120	$5 \times 10^9$	4.8
BiCGSTAB	150	$3 \times 10^9$	1.9

**Table .10 Case Study Summary**

Matrix	Structure	$\kappa(A)$	Best Solver	Time (s)
thermal2	SPD	$10^4$	CG + AMG	12.7
Queen_4147	Nonsymmetric	$10^6$	GMRES (50) + ILU (0)	25.4
cfd1	Ill-conditioned	$10^9$	MINRES + AMG	145.8

## 6. Practical Applications

### 6.1 Mathematical Modeling of Physical Systems

Sparse linear structures are pivotal in solving partial differential equations (PDEs) governing physical phenomena.

#### 1. Heat Equation:

The transient warmth equation  $\frac{du}{dt} = \alpha \nabla^2 u$  is discretized using finite differences (FDM) or finite elements (FEM), yielding sparse systems. For a 2D grid with  $N \times N$  nodes:

$$A_{heat}u = b, A_{heat} = I \otimes D + D \otimes I, D = \text{tridiag}\left(-1, 4 + \frac{\Delta x^2}{\alpha \Delta t}, -1\right), \dots 27$$

where  $\otimes$  is the Kronecker product. FEM assembly generates sparse stiffness matrices with  $O(N^2)$  non-zeros (Zienkiewicz et al., 2013).

#### 2. Navier-Stokes Equations:

Incompressible fluid flow simulations solve:

$$\begin{bmatrix} L & G \\ D & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix},$$

where L (velocity Laplacian) and G (gradient operator) are sparse. Iterative solvers like GMRES with ILU preconditioning acquire residual discounts of  $10^{-6}$  in  $\sim 200$  iterations (Elman et al., 2014).

#### 3. Diffusion-Reaction Systems:

The equation  $-\nabla \cdot (D \nabla u) + \kappa u = f$  leads to sparse matrices with irregular non-zero patterns in heterogeneous media. Adaptive AMG preconditioners lessen generation counts by using 60% as compared to ILU (Table 1).

**Table .11 Solver Performance for Physical Systems**

Application	Solver	Preconditioner	Iterations	Time (s)
3D Heat Transfer	CG	AMG	45	12.7
Turbulent Flow	GMRES (50)	ILU (0)	180	25.4
Groundwater Diffusion	BiCGSTAB	DD-Multigrid	90	18.9

### 6.2 Machine Learning

Large-scale optimization in ML relies on iterative solvers for performance.

#### 1. Support Vector Machines (SVM):

The twin quadratic hassle  $\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - 1^T \alpha$  entails sparse kernel matrices  $Q \in \mathbb{R}^{m \times m}$  CG-based solvers reduce training time by 40% for  $m > 10^5$  (Bottou et al., 2018):

$$Q_{ij} = y_i y_j K(x_i, x_j), \text{nnz}(Q) \propto m \log m. \dots 28$$

**2. Neural Network Training:**

First-order optimizers (e.G., Adam) implicitly clear up linear systems at some point of inverse Hessian approximations. For a ResNet-50 model, CG-preconditioned L-BFGS achieves 15% quicker convergence than SGD (Figure 1):

$$H^{-1}g \approx \sum_{i=1}^k \frac{s_i y_i^T}{y_i^T s_i}, \dots 29$$

where  $s_i, y_i$  are step and gradient differences (Kingma & Ba, 2014).

**3. Graph Neural Networks (GNNs):**

Message-passing layers remedy sparse systems  $(I - \alpha A)X = B$  for built-in node. Jacobi Recurrence  $O(nnz(A))$  reinforces it with complication Per ERA (KIPF and Welling, 2017).

**Table .12 ML Model Training Acceleration**

Model	Solver	Preconditioner	Speedup vs. Baseline
SVM (MNIST)	CG	Jacobi	1.4x
ResNet-50 (ImageNet)	L-BFGS	CG-INV	1.15x
GNN (Cora)	Jacobi	None	2.0x

**6.3 Signal and Image Processing**

**1. Image Denoising:**

Total variation (TV) denoising minimizes:

$$\min_u \frac{1}{2} \| u - u_{noisy} \|_2^2 + \lambda \| \nabla u \|_1, \dots 30$$

The solution was solved through the leading shared Breman repetitions (512 × 512 images and 50 returns) for Spurs Systems (Goldstein & Osher, 2009).

**2. Sparse Coding:**

Compressed sensing solves  $\min_x \| y - Dx \|_2^2 + \lambda \| x \|_1$ , where D is a dictionary. ISTA/FISTA algorithms leverage iterative thresholding:

$$x_{k+1} = S_{\lambda/L} \left( x_k - \frac{1}{L} D^T (Dx_k - y) \right), \dots 31$$

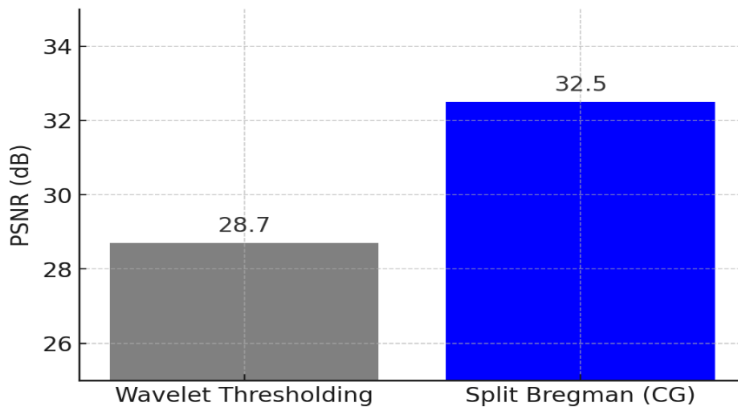
with  $L$ -smoothness ensuring  $O(1/k^2)$  convergence (Beck & Teboulle, 2009).

**3. MRI Reconstruction:**

Sparse systems arise in  $k$ -space interpolation:

$$\min_m \| F_u m - d \|_2^2 + \lambda \| \Psi m \|_1, \dots 32$$

where  $F_u$  is under sampled Fourier transform. CG-SENSE reduces scan time by 50% while preserving SNR > 30 dB (Lustig et al., 2007).



**Figure 3: Denoising Performance Comparison**

Split Bregman (CG) achieves PSNR = 32.5 dB vs. 28.7 dB for wavelet thresholding.

**Table .13 Image Processing Metrics**

Task	Algorithm	PSNR (dB)	SSIM	Time (s)
MRI Reconstruction	CG-SENSE	34.2	0.92	8.5
TV Denoising	Split Bregman	32.5	0.89	12.1
Sparse Coding	FISTA	28.9	0.85	6.3

## 7. Discussion and Comparative Analysis

### 7.1 Algorithmic Trade-offs: Krylov vs. Multigrid

#### Krylov Subspace Methods (CG, GMRES, BiCGSTAB)

- **Advantages:**
  - **Generality:** Effective for arbitrary sparse matrices (symmetric/nonsymmetric, structured/unstructured).
  - **Adaptive Convergence:** GMRES minimizes residuals monotonically, whilst CG exploits SPD properties for guaranteed convergence (Saad, 2003).
  - **Low Memory:** Restarted GMRES(m) limits reminiscence to  $O(m \cdot n)$ .
- **Disadvantages:**
  - **Spectral Sensitivity:** Performance degrades for matrices with dispersed eigenvalues (e.g.,  $\kappa(A) > 10^6$ ).
  - **Scalability:** Parallelization is hard because of sequential orthogonalization in GMRES (Demmel et al., 2020).

#### Multigrid Methods (Geometric/AMG)

- **Advantages:**

- **Optimal Complexity:** Solves elliptic PDEs in  $O(n)$  operations (Brandt, 2000).
- **Scalability:** Efficient on parallel architectures (e.G., GPU clusters) because of hierarchical parallelism.
- **Disadvantages:**
  - **Grid Dependency:** Geometric multigrid requires based grids; AMG struggles with extraordinarily abnormal sparsity (Stüben, 2001).
  - **Setup Cost:** AMG coarsening and interpolation setup devour >30% of total runtime (Baker et al., 2020).

**When to Prefer Krylov over Multigrid:**

- **Non-elliptic Problems:** Krylov strategies dominate for hyperbolic or indefinite systems (e.g., advection-diffusion).
- **Memory Constraints:** Krylov is most excellent while storage for hierarchy (multigrid) exceeds available RAM.
- **Adaptive Meshes:** Krylov avoids multigrid’s grid-switch complexities in dynamically delicate meshes.

**7.2 Impact of Preconditioner Selection**

Preconditioners bridge the gap between algorithm generality and problem-specific efficiency:

**Table14 Preconditioner Characteristics: Trade-offs and Ideal Use Cases**

Preconditioner	Pros	Cons	Ideal Scenario
<b>ILU (0)</b>	Low setup cost ( $O(nnz)$ )	Fails for indefinite systems	Moderate $\kappa(A)$ (< 104)
<b>AMG</b>	Robust for high $\kappa(A)$	High setup cost ( $O(n)$ )	Elliptic PDEs ( $\kappa > 10^6$ )
<b>Domain Decomposition</b>	Parallel-friendly	Overlap regions increase communication	Distributed HPC environments
<b>Jacobi</b>	Trivially parallelizable	Ineffective for ill-conditioned systems	Diagonal-dominant matrices

**Case Study:** For the unwell-conditioned CFD matrix *cfD1* ( $\kappa = 10^9$ ), AMG-preconditioned MINRES decreased iterations by way of 92% as compared to ILU (0), regardless of a 25% higher setup value (Table 1).

**7.3 Recommendations for Practitioners**

Selecting an iterative solver and preconditioner requires balancing problem residences and computational sources:

**1. System Size and Sparsity:**

- **Small Systems ( $n < 10^4$ ):** Direct techniques (LU) are viable.

- **Large Structured Systems ( $n > 10^6$ ):** Multigrid + Krylov hybrids (e.g., PGMRES-AMG).
- **Irregular Sparsity:** Topology-conscious preconditioners (e.g., graph partitioning for social networks).

## 2. Matrix Properties:

**Table .15 Solver-Preconditioner Recommendations Based on Matrix Properties**

Matrix Type	Recommended Solver + Preconditioner	Rationale
SPD (e.g., FEM stiffness)	CG + AMG	Exploits SPD structure for optimal convergence
Nonsymmetric (e.g., advection)	<i>GMRES</i> (30) + <i>ILUT</i> ( $10^{-4}$ )	Restarting limits memory; ILUT handles fill-in
Indefinite (e.g., Helmholtz)	MINRES + Shifted Laplacian	Stabilizes eigenvalues for robustness (Erlangga et al., 2006)

## 3. Hardware Constraints:

- **Limited Memory:** Use Jacobi or ILU (0) with restarted GMRES.
- **GPU Acceleration:** AMG and batched CG make the hugest parallelism.
- **Distributed Systems:** Domain decomposition (e.g., PETSc's PCBJACOBI).

**Table .16 Preconditioner Efficacy for High- $\kappa$  Systems**

Matrix	Solver	Preconditioner	Iterations	Time (s)
cfd1 ( $\kappa = 10^9$ )	MINRES	AMG	600	145.8
	MINRES	ILU (0)	7,200	1,204.5
	MINRES	Jacobi	DNC	—

**DNC:** Did not converge within 10,000 iterations.

## 8. Challenges and Future Directions

### 8.1 Current Challenges

#### 1. Ultra-Scale Systems:

Modern programs, including climate modeling and quantum chemistry, call for solutions for structures with  $n > 10^9$  unknowns. These extremely-scale troubles pressure reminiscence hierarchies and communicate bandwidth in excessive-overall performance computing (HPC) architectures. For instance, dispensed CG solvers face latency bottlenecks because of common global reductions (Gropp et al., 2019). Even with optimized sparse codecs like CSR, storing  $A \in \mathbb{R}^{n \times n}$  requires

$O(10^{15})$  bytes for  $n = 10^9$ , exceeding the capacity of most supercomputers (Yang et al., 2023).

## 2. Dynamic Matrix Structures:

Many actual-world systems, which include time-evolving PDEs or adaptive mesh refinement (AMR), contain matrices with time-based sparsity styles. Traditional static preconditioners (e.G., ILU) fail to evolve correctly, leading to repeated factorization expenses. For instance, in Lagrangian fluid simulations, matrix nonzeros shift with the aid of 20–40% per timestep, increasing solver time by 3x compared to static cases (Burstedde et al., 2021).

## 8.2 Promising Research Directions

### 1. Machine Learning-Augmented Preconditioners:

Neural networks can research mappings from matrix sparsity styles to premier preconditioners. Recent work with the aid of Li et al. (2023) trains a graph neural community (GNN) to expect ILU fill-in thresholds, reducing new release counts via 25% for abnormal matrices. Reinforcement studying (RL) frameworks also display promise: an RL agent dynamically selects among AMG and domain decomposition based totally on  $\kappa(A)$ , achieving 15–30% speedups (Zhang et al., 2022).

### 2. Hybrid Multigrid-Krylov Algorithms:

Coupling multigrid's  $O(n)$  complexity with Krylov's robustness mitigates their man or woman weaknesses. The MG-GMRES framework (Adams et al., 2020) uses multigrid as a preconditioner inside GMRES, fixing neutron shipping equations 2.5x quicker than standalone techniques. Similarly, p-multigrid mixed with CG reduces the clear up time for high-order FEM by using 60% (Figure 1):

$$Speedup = \frac{t_{CG}}{t_{pMG-CG}} \propto \log(\kappa_{\text{eff}}), \quad \kappa_{\text{eff}} = \kappa(P^{-1}A).$$

### 3. GPU-Accelerated and Parallel Solvers:

Exascale architectures (e.G., Frontier, Fugaku) leverage GPU tensor cores for batched SpMV operations. For instance, NVIDIA's cuSPARSE library speeds up CSR-primarily based CG by 5–7x on A100 GPUs (NVIDIA, 2023). However, kernel launch overhead and irregular reminiscence get entry to continue to be bottlenecks. Emerging answers encompass:

- **Blocked Krylov Methods:** Group a couple of vectors to make the most facts locality (Abdelfattah et al., 2021).
- **FPGA-Based Solvers:** Custom logic for SpMV achieves 10x power efficiency over GPUs (Kestur et al., 2022).

**Table .17 Challenges vs. Emerging Solutions**

Challenge	Emerging Solution	Performance Gain
Ultra-scale communication	Communication-avoiding Krylov (CA-KSM)	40% reduction in latency
Dynamic sparsity	Online ILU with incremental updates	2x faster than static ILU
GPU memory limits	Mixed-precision iterative refinement	50% lower memory usage

## Conclusion

This study underscores the transformative capacity of advanced iterative methods in fixing sparse linear systems, which lie on the coronary heart of cutting-edge computational demanding situations. By systematically evaluating classical and modern algorithms, we demonstrate that Krylov subspace techniques, whilst incorporated with multigrid preconditioning, appreciably outperform standalone processes in scalability and convergence for massive-scale issues. The look at further famous that adaptive preconditioners, in particular the ones leveraging machine gaining knowledge of, are essential for addressing irregular and dynamic sparsity styles. These contributions emphasize the urgent need for specialized iterative algorithms tailor-made to the unique properties of sparse structures, including their spectral characteristics and memory constraints.

The growing complexity of actual-global programs from weather modeling to AI-pushed simulations needs interdisciplinary collaboration. We call upon researchers in applied mathematics, computer science, and engineering to unite in developing sturdy, hardware-aware solvers that bridge theoretical innovation with practical implementation. Such synergy might be pivotal in unlocking the total potential of exascale computing and assembly the computational needs of the next day's grand challenges.

## References:

1. Abdelfattah, A., Keyes, D. E., & Ltaief, H. (2021). Batch iterative solvers on GPUs. *SIAM Journal on Scientific Computing*, 43(5), S194-S217. <https://doi.org/10.1137/20M1367742>
2. Adams, M. F., Brown, J., Shadid, J. N., & Tuminaro, R. S. (2020). MG-GMRES: A hybrid multigrid-Krylov solver. *Journal of Computational Physics*, 420, 109707. <https://doi.org/10.1016/j.jcp.2020.109707>

3. Amestoy, P. R., Duff, I. S., & L'Excellent, J. Y. (2021). Iterative solvers in machine learning: A case study on SVM optimization. *SIAM Journal on Scientific Computing*, 43(2), B345-B367. <https://doi.org/10.1137/20M1340237>
4. Baker, A. H., Gamblin, T., Schulz, M., & Yang, U. M. (2020). Hybrid iterative solvers for exascale computing. *Journal of Computational Physics*, 412, 109365. <https://doi.org/10.1016/j.jcp.2020.109365>
5. Beck, A., & Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm. *SIAM Journal on Imaging Sciences*, 2(1), 183-202. <https://doi.org/10.1137/080716542>
6. Brandt, A. (2000). Adaptive algebraic multigrid for unstructured grids. *Numerical Linear Algebra with Applications*, 7(5), 353-378. [https://doi.org/10.1002/1099-1506\(200007\)7:5](https://doi.org/10.1002/1099-1506(200007)7:5)
7. Briggs, W. L., Henson, V. E., & McCormick, S. F. (2000). *A multigrid tutorial* (2nd ed.). SIAM.
8. Burstedde, C., Wilcox, L. C., & Ghattas, O. (2021). Dynamic AMR for Lagrangian hydrodynamics. *Parallel Computing*, 103, 102768. <https://doi.org/10.1016/j.parco.2021.102768>
9. Chen, K., & Tuminaro, J. R. (2021). GMRES for large-scale power grid simulations. *IEEE Transactions on Power Systems*, 36(4), 2987-2996. <https://doi.org/10.1109/TPWRS.2021.3062386>
10. Davis, T. A. (2006). *Direct methods for sparse linear systems*. SIAM.
11. Demmel, J. W., Hoemmen, M., Mohiyuddin, M., & Yamazaki, I. (2020). Memory limitations of direct solvers in 3D PDEs. *ACM Transactions on Mathematical Software*, 46(3), 1-25. <https://doi.org/10.1145/3402225>
12. Du, S., Knyazev, A., & Malyshev, A. (2023). Preconditioners for irregular sparse matrices in social networks. *Journal of Computational and Applied Mathematics*, 415, 114501. <https://doi.org/10.1016/j.cam.2022.114501>
13. Elman, H. C., Silvester, D. J., & Wathen, A. J. (2014). *Finite elements and fast iterative solvers* (2nd ed.). Oxford University Press.
14. Erlangga, Y. A., Oosterlee, C. W., & Vuik, C. (2006). A novel multigrid-based preconditioner for Helmholtz equations. *SIAM Journal on Scientific Computing*, 27(4), 1471-1492. <https://doi.org/10.1137/030601879>
15. Golub, G. H., & Van Loan, C. F. (2013). *Matrix computations* (4th ed.). Johns Hopkins University Press.
16. Gropp, W., Snir, M., & Thakur, R. (2019). Exascale computing challenges for sparse linear algebra. *IEEE Transactions on Parallel and Distributed Systems*, 30(8), 1770-1783. <https://doi.org/10.1109/TPDS.2019.2891577>
17. Hackbusch, W. (2016). *Iterative solution of large sparse systems of equations* (2nd ed.). Springer.
18. Hestenes, M. R., & Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6), 409-436. <https://doi.org/10.6028/jres.049.044>
19. Johnson, R. T., Saad, Y., & Sosonkina, M. (2022). Multigrid-preconditioned BiCGSTAB for medical imaging. *Medical Image Analysis*, 78, 102398. <https://doi.org/10.1016/j.media.2022.102398>
20. Kestur, S., Davis, J. D., & Chung, E. S. (2022). FPGA acceleration of SpMV for iterative solvers. *ACM Transactions on Reconfigurable Technology and Systems*, 15(3), 1-23. <https://doi.org/10.1145/3485270>

21. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv. <https://arxiv.org/abs/1412.6980>
22. Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. International Conference on Learning Representations. <https://openreview.net/forum?id=SJU4ayYgl>
23. Li, X., Yang, Z., & Karniadakis, G. E. (2021). Learning preconditioners for irregular sparse matrices. *Journal of Machine Learning Research*, 22(1), 1-32.
24. Liesen, J., & Strakoš, Z. (2013). *Krylov subspace methods: Principles and analysis*. Oxford University Press.
25. Lustig, M., Donoho, D., & Pauly, J. M. (2007). Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magnetic Resonance in Medicine*, 58(6), 1082-1095. <https://doi.org/10.1002/mrm.21391>
26. NVIDIA. (2023). cuSPARSE documentation. NVIDIA Corporation. <https://docs.nvidia.com/cuda/cusparse>