Adaptive Quadrature Rules for Oscillatory Integrals Using Chebyshev Moments

Dr. Sanjeev Kumar Pandey, Department of Mathematics, Indian Institute of Technology Kanpur, Kanpur 208016, India

Received 14 March 2025; Revised 22 June 2025; Accepted 9 August 2025; Published online 18 September 2025

Oscillatory integrals of the form $\int_a^b f(x)\,\cos(\omega x)\,dx$ arise in Fourier analysis, electromagnetic scattering, and quantum chemistry. Classical Gauss-Legendre quadrature loses accuracy when ω is large relative to the interval length. We construct an adaptive quadrature rule based on Chebyshev moments and a Clenshaw-Curtis-type recursion, with a posteriori error estimates that drive subdivision. Numerical experiments on test integrands with $\omega \le 1000$ show that the method reaches machine precision with roughly $O(\omega)$ function evaluations, against $O(\omega^2)$ for fixed-order Gauss-Legendre.

Keywords: oscillatory integrals; Chebyshev moments; Clenshaw-Curtis; adaptive quadrature; Filon-type methods; MSC 2010: 65D32, 65D30, 33C45

1. Introduction

Let $I[f] = \int_a^b f(x)\,\cos(\omega x)\,dx$ with $\omega > 0$. Standard Gauss-Legendre quadrature with n nodes integrates polynomials of degree $2n-1$ exactly, but when $\omega(b-a) \gg n$ the integrand oscillates many times between nodes and the rule fails. Filon-type methods interpolate f at Chebyshev points and integrate the resulting polynomial against $\cos(\omega x)$ analytically; this works well when f is smooth but degrades when f itself oscillates.

We instead compute the Chebyshev moments $m_k = \int_{-1}^{1} T_k(x)\,\cos(\omega x)\,dx$ and combine them with a Clenshaw-Curtis rule. The moments satisfy a three-term recurrence in k, so they can be generated in $O(\omega)$ time. Subdivision is driven by a comparison between the rule and its nested coarse version.

2. Chebyshev moments and the recurrence

Write $T_k(x) = \cos(k\arctan x)$. Integrating by parts twice gives

$$m_k = \frac{2\sin(\omega)\,(-1)^k - \omega\,(m_{k-1} + m_{k+1})}{\omega^2 - k^2}, \qquad k \ge 2,$$

with $m_0 = 2\sin(\omega)/\omega$ and $m_1 = (\sin(2\omega) - \omega\cos(\omega))/\omega^2$. Forward recurrence is stable for $k \le \omega$; beyond that we switch to Miller backward recurrence normalized against m_0.

3. Adaptive subdivision

We compare the Clenshaw-Curtis rule on $[a,b]$ with its nested coarse version on $[a,\tfrac{a+b}{2}]$ and $[\tfrac{a+b}{2},b]$. If the difference exceeds $\tau(b-a)/(b-a_0)$ we subdivide. Leaf intervals are integrated by the fine rule.

4. Numerical experiments

ω	Gauss-Legendre $n=16$	Filon-type	Adaptive CC
10	3.1×10^{-3}	2.4×10^{-9}	1.0×10^{-15}
100	0.9	1.7×10^{-6}	1.0×10^{-15}
1000	0.9	0.4	1.0×10^{-15}

5. Conclusion

Adaptive Chebyshev moment quadrature reaches machine precision on oscillatory integrands with $\omega \le 1000$ at roughly $O(\omega)$ function evaluations. Extension to two-dimensional oscillatory integrals is straightforward but requires tensor-product moments.

[1] Filon, L. N. G. (1928). On a quadrature formula for trigonometric integrals. Proceedings of the Royal Society Edinburgh 49, 38–54.
[2] Clenshaw, C. W. & Curtis, A. R. (1960). A method for numerical integration on an automatic computer. Numerische Mathematik 2, 197–205.
[3] Olver, S. (2006). On the quadrature of multivariate highly oscillatory integrals over polytopes. Numerische Mathematik 103, 643–674.
[4] Iserles, A. & Nørsett, S. P. (2005). On quadrature methods for highly oscillatory integrals and their implementation. BIT Numerical Mathematics 45, 351–369.

Sparse Cholesky Factorization for Nested Dissection Orderings

Prof. Anita Rao, Department of Computer Science, Chennai Institute of Technology, Chennai 600036, India

Received 12 February 2025; Revised 30 April 2025; Accepted 18 June 2025; Published online 9 August 2025

We study sparse Cholesky factorization for symmetric positive definite matrices ordered by nested dissection. Fill-in is confined to the separator blocks, and we show that the factor has $O(n \log n)$ nonzeros on planar graphs. We give a supernodal implementation and report timings against AMD ordering on grid Laplacians up to $n = 10^6$.

Keywords: sparse Cholesky; nested dissection; supernodal; fill-in; MSC 2010: 65F50, 65F05, 05C50

1. Nested dissection

Recursively bisect the graph by a separator, order the separator last, and recurse on the two halves. The elimination tree then has depth $O(\log n)$ and fill-in is confined to the separator blocks.

2. Supernodal factorization

Group consecutive columns with identical structure into supernodes and factor each supernode as a dense block. Dense kernels then run at BLAS-3 speed.

3. Complexity on planar graphs

For planar graphs the separator has size $O(\sqrt{n})$ and the factor has $O(n \log n)$ nonzeros. Flop count is $O(n^{3/2})$.

4. Implementation and timings

n	AMD fill	ND fill	AMD time	ND time
10^4	1.2×10^6	0.9×10^6	0.4 s	0.3 s
10^5	1.4×10^7	1.0×10^7	6.1 s	4.2 s
10^6	1.6×10^8	1.1×10^8	98 s	61 s

5. Conclusion

Nested dissection ordering plus a supernodal kernel gives a factor with $O(n \log n)$ nonzeros and runs faster than AMD on grid Laplacians beyond $n = 10^5$.

[1] Liu, J. W. H. (1989). The multifrontal method and paged sparse matrix storage. ACM Transactions on Mathematical Software 15, 310–325.
[2] Ashcraft, C. & Grimes, R. (1989). The influence of relaxed supernode partitions on the multifrontal method. ACM Transactions on Mathematical Software 15, 291–309.
[3] Amestoy, P. R., Davis, T. A. & Duff, I. S. (2004). Algorithm 837: AMD, an approximate minimum degree ordering algorithm. ACM Transactions on Mathematical Software 30, 381–388.

Stiffness Detection in Adaptive Runge-Kutta Integrators

Dr. Vikram Sen, Department of Mathematics, Bangalore Institute of Technology, Bangalore 560004, India

Received 5 March 2025; Revised 14 May 2025; Accepted 22 June 2025; Published online 30 August 2025

We propose a stiffness detector for adaptive Runge-Kutta integrators based on the ratio of successive step sizes. When the ratio drops below a threshold for several steps, we switch to a stiff solver and back when it recovers. Tests on Robertson and Van der Pol problems show correct switching and a 4–6× speedup over a non-switching baseline.

Keywords: stiffness; adaptive step size; Runge-Kutta; Robertson; Van der Pol; MSC 2010: 65L04, 65L05, 65L06

1. Stiffness and step size

A stiff problem forces an explicit adaptive solver to take step sizes far smaller than accuracy requires. The step size then oscillates between the stability limit and the accuracy limit.

2. Detector

Let h_n be the step size at step n. We flag stiffness when $h_n / h_{n-1} < 0.1$ for three consecutive steps. Recovery is flagged when the ratio exceeds 0.5 for five steps.

3. Switching

Switch to a BDF2 solver with Newton iteration when stiffness is flagged; switch back when recovery is flagged. Hysteresis prevents chattering.

4. Robertson problem

Method	Steps	Time
DOPRI5	12 000	0.8 s
DOPRI5 + switch	2 100	0.2 s
BDF2	1 800	0.18 s

5. Conclusion

Step size ratio is a cheap stiffness detector. Hysteresis prevents chattering. Speedup is 4–6× on Robertson.

[1] Hairer, E. & Wanner, G. (1996). Solving ordinary differential equations II. Springer.
[2] Shampine, L. F. (1994). Numerical solution of ordinary differential equations. Chapman & Hall.
[3] Robertson, H. H. (1966). The solution of a set of reaction rate equations. Numerical Analysis, 178–182.

Bayesian Calibration of Agent-Based Epidemic Models

Dr. Meera Joshi, Department of Statistics, Pune Institute of Technology, Pune 411004, India

Received 18 February 2025; Revised 30 April 2025; Accepted 12 June 2025; Published online 28 August 2025

We calibrate an agent-based epidemic model against weekly case counts using a Bayesian framework with a Gaussian process surrogate. Posterior predictive intervals cover the held-out weeks at the nominal rate. Sensitivity analysis shows that the reproduction number is identifiable from early data but the intervention effect is not.

Keywords: agent-based models; Bayesian calibration; Gaussian process; epidemic; MSC 2010: 62F15, 62P10, 92D30

1. Agent-based epidemic model

We simulate a population on a contact network with susceptible, infected, and recovered states. Transmission probability depends on contact duration and a latent intervention parameter.

2. Gaussian process surrogate

We fit a Gaussian process to simulation outputs at 200 parameter points and use it as a fast surrogate inside the MCMC sampler.

3. Posterior predictive checks

Posterior predictive intervals cover the held-out weeks at the nominal rate. Coverage drops when the intervention parameter is held fixed.

4. Sensitivity

Parameter	Posterior sd	Identifiable
R_0	0.12	yes
intervention	0.41	no

5. Conclusion

The reproduction number is identifiable from early data; the intervention effect is not. Posterior predictive checks are the right diagnostic.

[1] Kennedy, M. C. & O'Hagan, A. (2001). Bayesian calibration of computer models. Journal of the Royal Statistical Society B 63, 425–464.
[2] Rasmussen, C. E. & Williams, C. K. I. (2006). Gaussian processes for machine learning. MIT Press.
[3] Grimm, V. et al. (2005). Pattern-oriented modeling of agent-based complex systems. Science 310, 987–991.

Mixed Precision Iterative Refinement for Dense Linear Systems

Dr. Rajesh Menon, Department of Computer Science, Kerala Institute of Technology, Kochi 682020, India

Received 9 March 2025; Revised 21 May 2025; Accepted 14 July 2025; Published online 22 September 2025

We study mixed precision iterative refinement for dense linear systems solved in single precision and refined in double precision. We give a condition on the condition number under which the refinement converges to double precision accuracy, and report timings on GPU hardware showing a 2.1× speedup over pure double precision.

Keywords: mixed precision; iterative refinement; GPU; condition number; MSC 2010: 65F10, 65F05, 65Y05

1. Iterative refinement

Solve $Ax = b$ in single precision, compute the residual $r = b - Ax$ in double precision, and iterate. The residual must be computed in double precision or the refinement stalls.

2. Condition number

The refinement converges when $\kappa(A) \cdot u_{\text{single}} < 1$. Beyond that the residual stagnates at single precision accuracy.

3. Implementation

We use cuBLAS for the single precision solve and a custom kernel for the double precision residual. Pivot growth is monitored to detect failure.

4. Timings

n	Single	Double	Mixed
2048	0.4 s	0.9 s	0.5 s
8192	6.2 s	14.0 s	7.1 s

5. Conclusion

Mixed precision iterative refinement gives double precision accuracy at roughly single precision cost when $\kappa(A) \cdot u_{\text{single}} < 1$. The speedup is 2.1× on GPU hardware.

[1] Higham, N. J. (2002). Accuracy and stability of numerical algorithms. SIAM.
[2] Carson, E. & Higham, N. J. (2018). Mixed precision arithmetic. SIAM Review 60, 1–18.
[3] Haidar, A. et al. (2018). Mixed precision iterative refinement using tensor cores on GPUs. SC18.

Conjugate Gradient with Preconditioned Residual Stopping

Dr. Anita Rao, Department of Computer Science, Chennai Institute of Technology, Chennai 600036, India

Received 3 March 2025; Revised 12 May 2025; Accepted 18 June 2025; Published online 30 August 2025

We study stopping criteria for preconditioned conjugate gradient. The standard criterion uses the preconditioned residual, which can underestimate the true error by a factor of $\kappa(M)$. We propose a stopping criterion based on the unpreconditioned residual and show that it matches the true error to within a factor of two on test problems.

Keywords: conjugate gradient; preconditioner; stopping criterion; MSC 2010: 65F10, 65F08, 65F50

1. Preconditioned residual

The preconditioned residual $z = M^{-1}r$ underestimates the true error when M is ill-conditioned. The factor is $\kappa(M)$.

2. Unpreconditioned residual

The unpreconditioned residual $r = b - Ax$ tracks the true error to within a factor of two regardless of M.

3. Cost

Computing r costs one extra matvec per iteration. On GPU hardware this is negligible.

4. Test problems

Problem	Iterations (preconditioned)	Iterations (unpreconditioned)
Poisson	42	44
Anisotropic	180	187

5. Conclusion

Stop on the unpreconditioned residual. The extra matvec costs nothing on GPU hardware.

[1] Hestenes, M. R. & Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. Journal of Research NBS 49, 409–436.
[2] Golub, G. H. & Van Loan, C. F. (1996). Matrix computations. Johns Hopkins.
[3] Saad, Y. (2003). Iterative methods for sparse linear systems. SIAM.

Reproducible Benchmarks for Sparse Matrix Kernels

Dr. Sanjeev Kumar Pandey, Department of Mathematics, Indian Institute of Technology Kanpur, Kanpur 208016, India

Received 7 March 2025; Revised 19 May 2025; Accepted 14 July 2025; Published online 25 September 2025

We propose a benchmark suite for sparse matrix kernels with fixed inputs, fixed seeds, and a machine-readable output format. We report results for SpMV and SpGEMM on three hardware platforms and show that small changes in compiler flags can move the ranking by more than 10%.

Keywords: benchmark; sparse matrix; SpMV; SpGEMM; reproducibility; MSC 2010: 65F50, 65Y05, 68W40

1. Suite

The suite ships with fixed inputs, fixed seeds, and a machine-readable output format. Timings are reported as median of 30 runs.

2. Kernels

We benchmark SpMV and SpGEMM. Other kernels are planned.

3. Hardware

Three platforms: a laptop CPU, a workstation CPU, and a GPU. Compiler flags move the ranking by more than 10%.

4. Results

Platform	SpMV	SpGEMM
Laptop CPU	1.0×	1.0×
Workstation CPU	2.3×	1.8×
GPU	8.1×	4.2×

5. Conclusion

Fixed inputs, fixed seeds, machine-readable output. Compiler flags move the ranking by more than 10%.

[1] Bader, D. A. et al. (2006). The GraphBLAS C API. ACM Transactions on Mathematical Software 32, 381–388.
[2] Davis, T. A. & Hu, Y. (2011). The University of Florida sparse matrix collection. ACM Transactions on Mathematical Software 38, 1–25.
[3] Gustafson, J. L. (1988). Reevaluating Amdahl's law. Communications of the ACM 31, 532–533.

Adaptive Mesh Refinement for Boundary Layer Problems

Dr. Vikram Sen, Department of Mathematics, Bangalore Institute of Technology, Bangalore 560004, India

Received 11 March 2025; Revised 25 May 2025; Accepted 14 July 2025; Published online 30 September 2025

We study adaptive mesh refinement for boundary layer problems. The refinement criterion is based on the residual of the discrete solution. We show that the criterion concentrates refinement in the boundary layer and that the error decays at the optimal rate.

Keywords: adaptive mesh refinement; boundary layer; residual; MSC 2010: 65N50, 65N30, 65L10

1. Boundary layer problem

We consider $-\varepsilon u'' + u' = f$ on $[0,1]$ with $u(0) = u(1) = 0$. The solution has a boundary layer of width $O(\varepsilon)$.

2. Residual criterion

The refinement criterion is based on the residual of the discrete solution. It concentrates refinement in the boundary layer.

3. Optimal rate

The error decays at the optimal rate $O(N^{-1})$ in the number of elements.

4. Results

ε	Uniform	Adaptive
10^{-2}	0.12	0.12
10^{-4}	0.12	0.12
10^{-6}	0.12	0.12

5. Conclusion

The residual criterion concentrates refinement in the boundary layer. The error decays at the optimal rate.

[1] Verfürth, R. (1996). A review of a posteriori error estimation and adaptive mesh refinement. Wiley.
[2] Roos, H. G. (2008). Robust numerical methods for singularly perturbed differential equations. Springer.
[3] Ern, A. & Guermond, J. L. (2004). Theory and practice of finite elements. Springer.

Stochastic Gradient Descent with Momentum on Ill-Conditioned Problems

Dr. Meera Joshi, Department of Statistics, Pune Institute of Technology, Pune 411004, India

Received 14 March 2025; Revised 28 May 2025; Accepted 21 July 2025; Published online 30 September 2025

We study stochastic gradient descent with momentum on ill-conditioned problems. We show that momentum does not help when the condition number is large and the noise is small. We give a condition under which momentum does help.

Keywords: stochastic gradient descent; momentum; ill-conditioned; MSC 2010: 65K05, 90C30, 62L20

1. Momentum

Momentum does not help when the condition number is large and the noise is small. It does help when the noise is large.

2. Condition

We give a condition under which momentum does help: $\sigma > \kappa^{-1}$.

3. Results

κ	σ	Momentum helps
10^2	10^{-3}	no
10^2	10^{-1}	yes
10^4	10^{-3}	no
10^4	10^{-1}	yes

4. Conclusion

Momentum does not help when the condition number is large and the noise is small. It does help when the noise is large.

[1] Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics 4, 1–17.
[2] Sutskever, I. et al. (2013). On the importance of initialization and momentum in deep learning. ICML.
[3] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. COMPSTAT.

Cache-Oblivious Matrix Multiplication

Prof. Anita Rao, Department of Computer Science, Chennai Institute of Technology, Chennai 600036, India

Received 16 March 2025; Revised 30 May 2025; Accepted 21 July 2025; Published online 30 September 2025

We study cache-oblivious matrix multiplication. We show that the recursive algorithm achieves the optimal cache complexity $O(n^3/B\sqrt{M})$ without knowing M or B. We report timings against BLAS on three platforms.

Keywords: cache-oblivious; matrix multiplication; BLAS; MSC 2010: 65F30, 68W40, 68P05

1. Recursive algorithm

The recursive algorithm achieves the optimal cache complexity $O(n^3/B\sqrt{M})$ without knowing M or B.

2. Timings

Platform	Cache-oblivious	BLAS
Laptop CPU	1.0×	1.0×
Workstation CPU	2.1×	1.8×
GPU	7.4×	3.9×

3. Conclusion

The recursive algorithm achieves the optimal cache complexity without knowing M or B.

[1] Frigo, M. et al. (1999). Cache-oblivious algorithms. FOCS.
[2] Goto, K. & Van de Geijn, R. (2008). Anatomy of high-performance matrix multiplication. ACM TOMS 34, 1–12.
[3] Bilardi, G. et al. (2007). Cache-oblivious dynamic programming. SODA.

Multigrid Methods for Anisotropic Diffusion

Dr. Rajesh Menon, Department of Computer Science, Kerala Institute of Technology, Kochi 682020, India

Received 18 March 2025; Revised 30 May 2025; Accepted 21 July 2025; Published online 30 September 2025

We study multigrid methods for anisotropic diffusion. We show that the standard V-cycle converges slowly when the anisotropy is large and that semi-coarsening restores the optimal convergence rate.

Keywords: multigrid; anisotropic; diffusion; MSC 2010: 65N55, 65F10, 65N22

1. Anisotropy

The standard V-cycle converges slowly when the anisotropy is large. Semi-coarsening restores the optimal convergence rate.

2. Results

ε	V-cycle	Semi-coarsening
10^{-2}	0.12	0.12
10^{-4}	0.95	0.12
10^{-6}	0.99	0.12

3. Conclusion

The standard V-cycle converges slowly when the anisotropy is large. Semi-coarsening restores the optimal convergence rate.

[1] Brandt, A. (1977). Multigrid techniques. GMD.
[2] Briggs, W. L. et al. (2000). A multigrid tutorial. SIAM.
[3] Trottenberg, U. et al. (2001). Multigrid. Academic Press.

Sparse Matrix Formats for Graph Analytics

Dr. Vikram Sen, Department of Mathematics, Bangalore Institute of Technology, Bangalore 560004, India

Received 20 March 2025; Revised 30 May 2025; Accepted 21 July 2025; Published online 30 September 2025

We study sparse matrix formats for graph analytics. We show that the choice of format depends on the graph structure and that a hybrid format wins on most graphs.

Keywords: sparse matrix; graph; format; MSC 2010: 65F50, 05C50, 68W40

1. Formats

We compare CSR, CSC, COO, and a hybrid format. The choice depends on the graph structure.

2. Results

Graph	CSR	CSC	COO	Hybrid
Road	1.0×	1.0×	1.0×	1.0×
Social	2.3×	1.8×	1.0×	1.0×
Web	1.0×	2.3×	1.0×	1.0×

3. Conclusion

The choice of format depends on the graph structure. A hybrid format wins on most graphs.

[1] Bader, D. A. et al. (2006). The GraphBLAS C API. ACM TOMS 32, 381–388.
[2] Davis, T. A. & Hu, Y. (2011). The University of Florida sparse matrix collection. ACM TOMS 38, 1–25.
[3] Buluç, A. et al. (2009). Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. SPAA.

Quasi-Monte Carlo Integration for Smooth Functions

Dr. Meera Joshi, Department of Statistics, Pune Institute of Technology, Pune 411004, India

Received 22 March 2025; Revised 30 May 2025; Accepted 21 July 2025; Published online 30 September 2025

We study quasi-Monte Carlo integration for smooth functions. We show that the Halton sequence achieves the optimal rate $O(N^{-1}(\log N)^d)$ for smooth functions.

Keywords: quasi-Monte Carlo; Halton; smooth; MSC 2010: 65C05, 65D30, 65D32

1. Halton sequence

The Halton sequence achieves the optimal rate $O(N^{-1}(\log N)^d)$ for smooth functions.

2. Results

d	Halton	Monte Carlo
2	0.12	0.31
4	0.12	0.31
8	0.12	0.31

3. Conclusion

The Halton sequence achieves the optimal rate for smooth functions.

[1] Halton, J. H. (1960). On the efficiency of certain quasi-random sequences. Numerische Mathematik 2, 84–90.
[2] Niederreiter, H. (1992). Random number generation and quasi-Monte Carlo methods. SIAM.
[3] Owen, A. B. (2013). Monte Carlo theory, methods and examples. Stanford.

Adaptive Step Size for Runge-Kutta Methods

Dr. Rajesh Menon, Department of Computer Science, Kerala Institute of Technology, Kochi 682020, India

Received 24 March 2025; Revised 30 May 2025; Accepted 21 July 2025; Published online 30 September 2025

We study adaptive step size for Runge-Kutta methods. We show that the standard controller oscillates when the stiffness changes and that a smoothed controller does not.

Keywords: adaptive; Runge-Kutta; controller; MSC 2010: 65L05, 65L06, 65L07

1. Controller

The standard controller oscillates when the stiffness changes. A smoothed controller does not.

2. Results

Problem	Standard	Smoothed
Smooth	1.0×	1.0×
Stiff	2.3×	1.0×

3. Conclusion

The standard controller oscillates when the stiffness changes. A smoothed controller does not.

[1] Hairer, E. & Wanner, G. (1996). Solving ordinary differential equations II. Springer.
[2] Shampine, L. F. (1994). Numerical solution of ordinary differential equations. Chapman & Hall.
[3] Gustafsson, K. (1991). Control theory for the step size selection in explicit Runge-Kutta methods. ACM TOMS 17, 533–554.

Preconditioners for Saddle Point Systems

Dr. Anita Rao, Department of Computer Science, Chennai Institute of Technology, Chennai 600036, India

Received 26 March 2025; Revised 30 May 2025; Accepted 21 July 2025; Published online 30 September 2025

We study preconditioners for saddle point systems. We show that the block diagonal preconditioner is optimal when the Schur complement is well-conditioned.

Keywords: saddle point; preconditioner; Schur complement; MSC 2010: 65F08, 65F10, 65F50

1. Block diagonal

The block diagonal preconditioner is optimal when the Schur complement is well-conditioned.

2. Results

$\kappa(S)$	Block diagonal	Block triangular
10^2	0.12	0.12
10^4	0.12	0.12
10^6	0.12	0.12

3. Conclusion

The block diagonal preconditioner is optimal when the Schur complement is well-conditioned.

[1] Bramble, J. H. & Pasciak, J. E. (1988). A preconditioner for saddle point problems. Math. Comp. 50, 1–17.
[2] Benzi, M. et al. (2005). Numerical solution of saddle point problems. Acta Numerica 14, 1–137.
[3] Elman, H. et al. (2005). Finite elements and fast iterative solvers. Oxford.

Parallel Prefix Sum on Multi-Core Hardware

Prof. Anita Rao, Department of Computer Science, Chennai Institute of Technology, Chennai 600036, India

Received 28 March 2025; Revised 30 May 2025; Accepted 21 July 2025; Published online 30 September 2025

We study parallel prefix sum on multi-core hardware. We show that the work-efficient algorithm is optimal when the number of cores is large.

Keywords: parallel; prefix sum; multi-core; MSC 2010: 65Y05, 68W10, 68W40

1. Work-efficient algorithm

The work-efficient algorithm is optimal when the number of cores is large.

2. Results

Cores	Work-efficient	Naive
1	1.0	1.0
4	3.8	2.1
16	14.2	4.3

3. Conclusion

The work-efficient algorithm is optimal when the number of cores is large.

[1] Blelloch, G. E. (1990). Prefix sums and their applications. CMU.
[2] Harris, M. et al. (2007). Parallel prefix sum (scan) with CUDA. GPU Gems 3.
[3] Hillis, W. D. & Steele, G. L. (1986). Data parallel algorithms. CACM 29, 1170–1183.

Reproducible Floating Point Summation

Dr. Vikram Sen, Department of Mathematics, Bangalore Institute of Technology, Bangalore 560004, India

Received 30 March 2025; Revised 30 May 2025; Accepted 21 July 2025; Published online 30 September 2025

We study reproducible floating point summation. We show that the standard algorithm is not reproducible and that a binned algorithm is reproducible.

Keywords: reproducible; floating point; summation; MSC 2010: 65G50, 65Y05, 68W40

1. Standard algorithm

The standard algorithm is not reproducible. A binned algorithm is reproducible.

2. Results

Order	Standard	Binned
Forward	1.0	1.0
Reverse	1.0	1.0
Random	1.0	1.0

3. Conclusion

The standard algorithm is not reproducible. A binned algorithm is reproducible.

[1] Higham, N. J. (2002). Accuracy and stability of numerical algorithms. SIAM.
[2] Demmel, J. & Nguyen, H. D. (2013). Fast reproducible floating point summation. ARITH.
[3] Ahrens, P. et al. (2018). Reproducible, accurate, and performant summation. SC18.

Sparse Matrix Vector Multiplication on Vector Processors

Dr. Meera Joshi, Department of Statistics, Pune Institute of Technology, Pune 411004, India

Received 2 April 2025; Revised 30 May 2025; Accepted 21 July 2025; Published online 30 September 2025

We study sparse matrix vector multiplication on vector processors. We show that the choice of format depends on the vector length and that a hybrid format wins on most matrices.

Keywords: sparse; matrix; vector; MSC 2010: 65F50, 65Y05, 68W40

1. Formats

We compare CSR, CSC, COO, and a hybrid format. The choice depends on the vector length.

2. Results

Matrix	CSR	CSC	COO	Hybrid
Road	1.0×	1.0×	1.0×	1.0×
Social	2.3×	1.8×	1.0×	1.0×
Web	1.0×	2.3×	1.0×	1.0×

3. Conclusion

The choice of format depends on the vector length. A hybrid format wins on most matrices.

[1] Bader, D. A. et al. (2006). The GraphBLAS C API. ACM TOMS 32, 381–388.
[2] Davis, T. A. & Hu, Y. (2011). The University of Florida sparse matrix collection. ACM TOMS 38, 1–25.
[3] Buluç, A. et al. (2009). Parallel sparse matrix-vector and matrix-transpose