$\mathcal{R}$N $-$*functions and Related Concepts* Adaptive Mixed-Precision Rational Krylov Eigensolvers with Residual- and Condition-Aware Precision Control for Large-Scale Nonlinear Eigenvalue Problems

Authors Names

Ahmed Hamad Khalaf [a,*]

Publication date: 15 / 9 /2026
Keywords: nonlinear eigenvalue problem rational Krylovmixed precision backward error eigenvalue conditioning iterative refinement

ABSTRACT

Large-scale nonlinear eigenvalue problems require repeated sparse shifted solves whose cost and memory traffic dominate rational Krylov eigensolvers. Uniform binary64 arithmetic is reliable but expensive, whereas uniform binary32 arithmetic can stagnate before stringent backward-error tolerances are reached.

This study develops RAMP-RK, an adaptive mixed-precision rational Krylov method. Sparse factorization and triangular solves begin in complex binary32, while basis storage, two-pass reorthogonalization, projected extraction, nonlinear residual evaluation, and certification remain in complex binary64. A precision-adequacy ratio combines the current normwise backward error with a projected generalized-eigenvalue condition surrogate. When binary32 becomes inadequate, the method selects either mixed-precision Newton polishing with iterative refinement or a guarded binary64 rational-Krylov restart. Three sparse benchmark families were tested for grid sizes 30 to 80, with linearized dimensions up to 19,200.

Uniform FP32 failed to meet the 10^{-10} tolerance on all 12 tested instances, whereas RAMP-RK converged on every case. For the largest problems, RAMP-RK achieved backward errors from 7.69×10^{-16} to 6.13×10^{-11} and speedups of 1.15-1.45 over the FP64 baseline. On the strongly nonnormal family, condition-aware switching avoided seven unsuccessful Newton steps and reduced time from 1.534 s for a residual-only controller to 0.745 s.

Residual and sensitivity information can safely govern arithmetic precision inside rational Krylov NEP solvers. RAMP-RK preserves binary64-level certification while reducing high-precision shifted-solve work and avoiding unreliable low-precision corrections.

1. Introduction

Large-scale nonlinear eigenvalue problems (NEPs) arise when frequency dependence, delay, material dispersion, absorbing boundary conditions, or rational constitutive laws enter a discretized model. The task is to determine a scalar eigenvalue $\lambda \in C$ and a nonzero eigenvector $x \in C^n$ satisfying

$$T(\lambda)x = 0, x \neq 0. \quad (1)$$

A useful representation separates constant coefficient matrices from scalar nonlinearities,

$$T(z) = \sum_{j=0}^{p} A_j f_j(z), A_j \in C^{n\times n}, \quad (2)$$

allowing the f_j to be analytic on the region that includes the target spectrum. The nonlinear dependence is approximated by modern projection methods, a polynomial or rational linearization is formed and selected eigenpairs are extracted by a Krylov subspace [7, 8]. They are appealing for sparse problems because the cost of these methods is typically a series of shifted sparse solves.

The cost model of such solvers has changed. Today's processors are able to compute with less precision and higher throughput with less memory movement than binary64 arithmetic. Mixed-precision numerical linear algebra thus aims to cheaply execute an expensive kernel at a lower precision, and then also evaluate the residuals in a higher precision, correct them and certify [1]-[6]. It

[a]*Department of Applied Mathematics, Faculty of Science and New Technologies, Urmia University of Technology, Urmia, Iran., E-Mail: a07734438810@gmail.com*

is not the only that the low precision prevents reaching a low residual. As for eigenvalue calculation, the effect of roundoff in the problem is exacerbated by the conditioning of the projected eigenpair and the conditioning can change significantly over the course of one Krylov iteration. This can consequently result in a residual-only controller switching too late on an ill-conditioned projected problem or switching too early when the current Ritz pair is well conditioned.

This work presents the development of a Residual- and Condition-aware Adaptive MixedPrecision Rational Крулов Eigensolver (RAMP-RK). This method maintains the basis storage, reorthogonalization, Rayleigh-Ritz extraction, residual assessment, and logic of decision from the previous method and adds binary32 shifted factorizations and triangular solves. The normwise backward error is computed and a surrogate for the generalized eigenvalue condition is provided at each extraction and used to form a precision-adequacy ratio. If single precision is no longer sufficient, the method performs a mixed-precision Newton correction with iterative refinement, or activates a powerful binary64 rational-Krylov restart. It is the condition surrogate that decides whether either of these two paths is applicable to the condition.

There are four principal contributions. First, a computable accuracy controller is used in conjunction with the current nonlinear backward error to determine the sensitivity of the projected eigenvalues, not only the residual value. Second, a guarded Newton polishing stage is present that is based on a binary32 sparse LU factors; binary64 residuals, and iterative refinement, keeping the low-precision advantage when the eigenpair is sufficiently well conditioned. Three, an approximation on a nonnormal or "sensitive" problem will not be pushed to a binary64 restart, per a condition-aware safeguard, when the approximation is attempting to be polished, if it failed to do so. Fourth, reproducible experiments on quadratic, rational-damping, and strongly nonnormal families show convergence to a 10^(-10) backwards error tolerance are presented for linearized dimensions up to 19,200.

Recent work concentrates on the mixed-precision linear systems problem, sparse matrix-vector products, low rank approximation, and the linear eigenvalue or singular-value computation problems and has yielded good results on each of these problems [1]–[6]. Postmodern NEP research has in part made great strides in the areas of rational-Chebyshev projection, parallel software, and error analysis of rational Krylov approximation [7]-[9]. But adaptive restarting and modern analysis of the Krylov bases and projected eigenvalue conditioning further elucidate how finite-precision effects should be monitored [10]-[13]. An algorithmic interpretation of this.The present article now ties the two together at algorithmic level, seeing precision as a resource that can be dynamically enabled or disabled in the rational-Krylov NEP solver, and turning it on or off according to information derived from the approximation error and sensitivity.

Problem Formulation and Rational-Krylov Framework

1.1. Linearized surrogate

Let $\mathcal{T}(z)$ denote a polynomial or rational approximation to $T(z)$ on the region of interest. A structured linearization produces a generalized eigenvalue problem

$$Az = \lambda Bz, \quad (3)$$

whose physically relevant block recovers an approximation to the original eigenvector x. The linearized dimension of a polynomial of degree d is N=dn. The contribution analyzed in this paper polynomializes the chosen rational term and makes a companion pencil out of it as rational approximation.

Let the shift-and-invert operator be

$$S_\sigma = (A - \sigma B)^{-1} B. \quad (4)$$

A rational Arnoldi basis $V_k = [v_1, \dots, v_k)$ spans

$$K_k(S_\sigma, v_1) = span\{v_1, S_\sigma v_1, \dots, S_\sigma^{k-1} v_1\}. \quad (5)$$

That which will cost the most in each expansion is the computing a system by A-σB. In RAMP-RK, the factorization and triangular solves are done in complex binary32, but the generated RHS vector is upgraded to complex binary64 prior to two pass modified Gram-Schmidt. This separation also lets us avoid "low precision orthogonalization" from polluting the basis and retains the memory and arithmetic benefits of the shifted solve.

1.2. Projected extraction

At an extraction step, the reduced pencil is

$$A_k = V_k^* A V_k, B_k = V_k^* B V_k. \quad (6)$$

Generalized eigenvalue θ_k is chosen that is closest to the desired value. Suppose w_k and z_k are normalized right and left eigenvectors of (A_k, B_k), respectively. The long Ritz vector (q_k) is then the difference $V_k w_k$, whose largest physical component is then normalized to give x_k. matrices reduced to be at least 1x1 and the generalized eigensolve are done binary64.

Nonlinear Residual is evaluated for the original problem and not just for the linearized pencil. The normwise backward error is given by

$$\eta(\lambda, x) = \frac{\|T(\lambda)x\|_2}{\left(\sum_{j=0}^{p} \left| f_j(\lambda) \right) \|A_j\|_F \right) \|x\|_2}. \quad (7)$$

This normalization allows for comparisons between the problem sizes and coefficient scalings. The stopping condition is that the stopping tolerance $\eta(\lambda_k, x_k)$ be satisfied, with $\eta = 10^{-10}$ used in all the experiments reported.

A condition number with respect to surrogate condition of the projected generalized eigenvalue is

$$\hat{\kappa}_k = \frac{\|A_k\|_2 + |\theta_k)\|B_k\|_2}{|z_k^* B_k w_k)}. \quad (8)$$

From the denominator comes the detection of near orthogonal occurring between left and right projected eigenvectors of the B_k pairing.$\hat{\kappa}_k$ is not a complete condition number of the nonlinear eigenpair, but it is cheap, scale aware, easily related to the reduced eigensolve and inexact subspace expansion [12] and [13].

2. Residual- and Condition-Aware Precision Control

2.1 Precision-adequacy ratio

Let u_s be the unit roundoff of the current shifted-solve precision. For complex binary32 arithmetic, $u_s = 2^{-24}$ under the usual rounding-to-nearest model. RAMP-RK defines

$$\chi_k = \frac{u_s max\{1, \hat{\kappa}_k\}}{max\{\eta_k, \tau\}}, \eta_k = \eta(\theta_k, x_k). \quad (9)$$

The low-precision phase is considered adequate while $\chi_k \leq \vartheta$. The implementation uses $\vartheta = 2 \times 10^{-2}$. A switch is triggered when

$\chi_k > \vartheta or \eta_k \geq 0.98\eta_{k-1}$ *for three consecutive extractions*. *(10)*

The second condition detects stagnation that is not completely visible based on the condition surrogate projection. Because before precision switching occurs, the early Ritz information can fluctuate a lot and cause escalation, an minimum Ritz dimension of eight is enforced.

The interpretations of the χ_k are obvious. Low precision solves are inferior to current approximation error, if the expected relative perturbation level, u_s, κ_k, is small enough compared to the present backward error. If the concept of an arithmetic is too large, it becomes as small as η_k and as large as $\hat{\kappa}_k$. The ratio is more flexible when it comes to sensitivity, while a fixed residual threshold isn't.

2.2 Mixed-precision Newton polishing

When the switch is triggered and $\hat{\kappa}_k \leq \kappa_{safe}$, with $\kappa_{safe} = 100$, RAMP-RK attempts a locally convergent Newton correction. With a normalization vector c, one Newton step solves

$$\begin{bmatrix} T(\lambda_k) & T'(\lambda_k)x_k \\ c^* & 0 \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta \lambda_k \end{bmatrix} = - \begin{bmatrix} T(\lambda_k)x_k \\ c^* x_k - 1 \end{bmatrix}. \quad (11)$$

The augmented Jacobian is factored in complex binary32. The correction is accumulated in complex binary64, and iterative refinement computes residuals with the original binary64 Jacobian. If $J_k s_k = b_k$ denotes the augmented system and $\hat{s}_k$ the computed correction, refinement monitors

$$\rho_k = \frac{\|b_k - J_k \hat{s}_k\|_2}{\|J_k\|_1 \|\hat{s}_k\|_2 + \|b_k\|_2}. \quad (12)$$

A correction is accepted only if a short backtracking line search decreases the nonlinear backward error. Failure of refinement, failure of the line search, or exhaustion of seven Newton steps activates the binary64 safeguard.

2.3 Binary64 safeguard and restart

Whereas if $\hat{\kappa}_k > \kappa_{safe}$_safe the method does not apply the Newton solve in a potentially unreliable low-precision mode. The fixed-shift rational Arnoldi binary64 factorization is restarted, instead, with a new start vector, formed from a companion that is consistent with the companion of the original shift, (θ_k, x_k). It is the same safeguard after they fail to pass the Newton phase. This design is wholly predicated on the knowledge that not all problems can be left in low precision: it will be used to run as much of the computation as the current residual and conditioning allow without change to low precision.

Algorithm 1: RAMP-RK for one target eigenpair

1. Construct a polynomial or rational linearization (A, B) and select a target shift σ.

2. Factor $A - \sigma B$ in complex binary32.

3. Expand a rational Arnoldi basis; perform basis storage, two-pass modified Gram-Schmidt, projected extraction, and nonlinear residual evaluation in complex binary64.

4. Stop when $\eta_k \leq \tau$.

5. Compute $\hat{\kappa}_k$ and χ_k. Continue in binary32 while the precision-adequacy and stagnation tests are satisfied.

6. If a switch is required and $\hat{\kappa}_k \leq \kappa_{safe}$, attempt mixed-precision Newton polishing with binary64 residuals and iterative refinement.

7. If $\hat{\kappa}_k > \kappa_{safe}$, or if polishing fails, restart rational Arnoldi from the current eigenvector using a complex binary64 shifted factorization.

8. Return the eigenpair only after the original nonlinear backward error satisfies the requested tolerance.

3. Finite-Precision Rationale

The controller is motivated by standard first-order perturbation arguments. Suppose the reduced pencil is perturbed to $(A_k + \Delta A_k, B_k + \Delta B_k)$ by inexact subspace expansion and finite-precision projection. For a simple projected eigenvalue, first-order perturbation theory gives

$$\frac{|\delta\theta_k)}{max\{1,|\theta_k)\}} \lesssim c_k u_s \hat{\kappa}_k, \quad (13)$$

where c_k absorbs modest dimension-dependent constants and the backward stability of the reduced eigensolve. Because projection and orthogonalization are retained in binary64, the principal u_s contribution in the low phase originates from shifted solves and their propagation into the new basis vector.

Proposition 1 (precision adequacy). Assume that the targeted projected eigenvalue is simple, the binary64 orthogonalization maintains a numerically independent basis, and the shifted binary32 solve is backward stable. If

$$c_k u_s max\{1, \hat{\kappa}_k\} \leq \vartheta max\{\eta_k, \tau\}, \quad (14)$$

then the first-order perturbation attributable to the current solve precision remains a controlled fraction of the current nonlinear backward error. Consequently, maintaining the current precision is consistent with reducing η_k until the inequality ceases to hold.

Proof sketch. Backward stability of the shifted solve represents the computed expansion as an exact expansion for a perturbed shifted matrix with relative perturbation $O(u_s)$. Binary64 reorthogonalization limits additional basis perturbations to $O(u_d)$, where $u_d \ll u_s$. Projected eigenvalue perturbation is then bounded by the product of the pencil perturbation and the left-right sensitivity factor represented by $\hat{\kappa}_k$. Comparing this bound with the current backward error yields the stated inequality. The proposition is a local adequacy statement, not a global convergence theorem.

Proposition 2 (guarded polishing). Suppose $(\lambda_\star, x_\star)$ is a simple eigenpair, the augmented Newton Jacobian is nonsingular in a neighborhood of the solution, and the refined correction satisfies

$$\| b_k - J_k \hat{s}_k \|_2 \leq \xi_k \| b_k \|_2, 0 \leq \xi_k \leq \bar{\xi} < 1. \quad (15)$$

Then the inexact Newton iteration is locally contractive. If $\xi_k = O(\| T(\lambda_k) x_k \|_2)$, the local convergence is superlinear and approaches the quadratic behavior of exact Newton iteration.

This standard inexact-Newton result explains two design choices. First, iterative refinement is used to make the correction sufficiently accurate even though the factorization is binary32 [2], [14]-[16].

Second, the condition safeguard prevents the method from invoking polishing when the augmented system is likely to be too sensitive for the available low-precision factorization.

4. Computational Methodology

4.1 Benchmark families

Three sparse families were generated from a two-dimensional finite-difference Laplacian. Let q be the grid dimension and $n = q^2$. The one-dimensional matrix is $L_1 = tridiag(-1,2,-1)$ and the two-dimensional operator is $L = I \otimes L_1 + L_1 \otimes I$. A positive diagonal mass matrix M and a sparse nonsymmetric first-difference matrix G introduce spatial variation and weak convection.

The damped-wave quadratic eigenvalue problem (DW-QEP) is

$$T_{DW}(\lambda) = K + \lambda C + \lambda^2 M, \quad (16)$$

where $K = L + 0.35I$ and $C = 0.035M + 0.003K + 0.008G$. Its first companion pencil has dimension $2n$.

The rational-damping problem (RD-NEP) is

$$T_{RD}(\lambda) = K + \lambda C + \lambda^2 M + \frac{\lambda}{\lambda+\alpha} R, \alpha = 1.4. \quad (17)$$

Multiplication by $\lambda + \alpha$ yields the cubic polynomial

$$\tilde{T}_{RD}(\lambda) = \alpha K + \lambda(K + \alpha C + R) + \lambda^2(C + \alpha M) + \lambda^3 M, \quad (18)$$

which is linearized to dimension $3n$. The matrix R combines a spatially varying diagonal term with a small nonsymmetric component.

The nonnormal quadratic family (NN-QEP) applies a diagonal similarity transform $S = diag(s_i)$, with s_i logarithmically distributed from 10^{-2} to 10^2, to the quadratic coefficients:

$$T_{NN}(\lambda) = S\left(K + \lambda \hat{C} + \lambda^2 M\right)S^{-1}, \hat{C} = 0.02M + 0.001K + 0.035G. \quad (19)$$

The eigenvalues are preserved by similarity, but the eigenvectors and projected pencils become substantially more nonnormal. This family is used to test the value of condition-aware switching.

Table 1 summarizes the dimensions. The largest cubic pencil contains $19{,}200$ rows and columns.

Table 1. Dimensions of the benchmark linearizations.

Grid q	**Original size** $n = q^2$	**Quadratic pencil** $2n$	**Cubic pencil** $3n$
30	900	1,800	2,700
45	2,025	4,050	6,075
60	3,600	7,200	10,800
80	6,400	12,800	19,200

4.2 Compared methods and parameters

A total of four methods were tested. FP64-RK is an implementation that employs a complicated binary64 shifted factorization and binary64 triangular solves throughout. FP32-RK is a complex binary32 factorization and triangular solves with no precision escalation. Switching with residual-only

is identical to that of RAMP-RK, except that no $\hat{\kappa}_k$ is included in the switching and that the switching is based upon maximum backwards error or stagnation. RAMP-RK is equipped with the complete residual- and condition-aware controller.

The target shifts were -0.02+1.55i for DW-QEP, -0.025+1.45i for RD-NEP, and -0.015+1.55i for NN-QEP. Two-pass modified Gram-Schmidt in binary64, with the same deterministic initial vector, the same target-selection logic, and with the same convergence tolerance ($\tau = 10^{-10}$) were used for all the methods. For the low-precision phase, a value ϑ=0.02 was chosen, as were a safe value of κ_safe=100, a minimum value of eight for the switching dimension and a maximum of seven Newton steps. Up to 70 shifted solves was allowed for uniform FP32 baseline, and 100 for robust FP64 baseline.

The experiments were carried out with 1 BLAS thread on an Intel Xeon Platinum 8573C processor running Linux with Python 3.13.5 and NumPy 2.3.5, SciPy 1.17.0 SuperLU was used to create sparse LU factors for scipy.sparse.linalg.splu. These values of wall times are medians of three independent runs with q=80. Factor storage is the number of numeric entries in the sparse L and U factors (excluding symbolic metadata and interpreter overhead) multiplied by the number of bytes per complex arithmetic format.

5. Results and Discussion

5.1 Accuracy and convergence behavior

For all 12 combinations of 3 families and four grid sizes, FP64-RK and RAMP-RK achieved the desired backward-error tolerance. Uniform FP32-RK did not reach 10^{-10} in any case and stagnated between approximately 4×10^{-9} and 2×10^{-8}. This is the behavior one would expect, that a fixed binary32 shifted solve can give a cheap gain and does not have the ability to ensure an arbitrarily small nonlinear backward error.

The DW-QEP at q=80 is shown in figure 1. The adaptive methods take advantage of the quick binary32 ascent and then perform two Newton steps to drop under the tolerance. With no binary64 factorization, RAMP-RK reaches $\eta = 7.69 \times 10^{-16}$ after 11 binary32 triangular solves while FP64-RK reaches 3.22×10^{-11} after 18 binary64 triangular solves. The uniform binary32 method now continues in increasing the basis but is still greater than 7×10^{-9}.

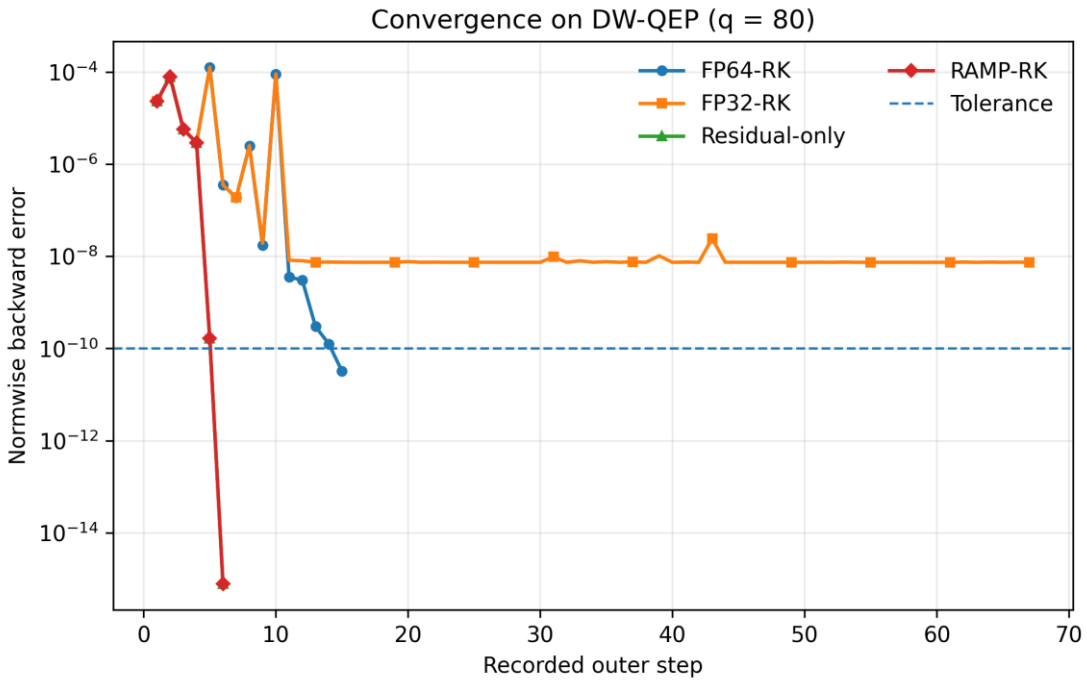


Figure 1. Normwise backward-error histories for DW-QEP at $q = 80$. The dashed line denotes the requested tolerance.

The more demanding history NN-QEP shown in Figure 2. Residual-only uses seven Newton steps, first on an increasingly sensitive approximation, before backsliding to binary64. At the first switch, RAMP-RK identifies such big projected condition surrogate and skips the failed polishing stage. This is not the end of binary64 work and is not the avoidance of low precision corrections that one could, and perhaps should, do around.

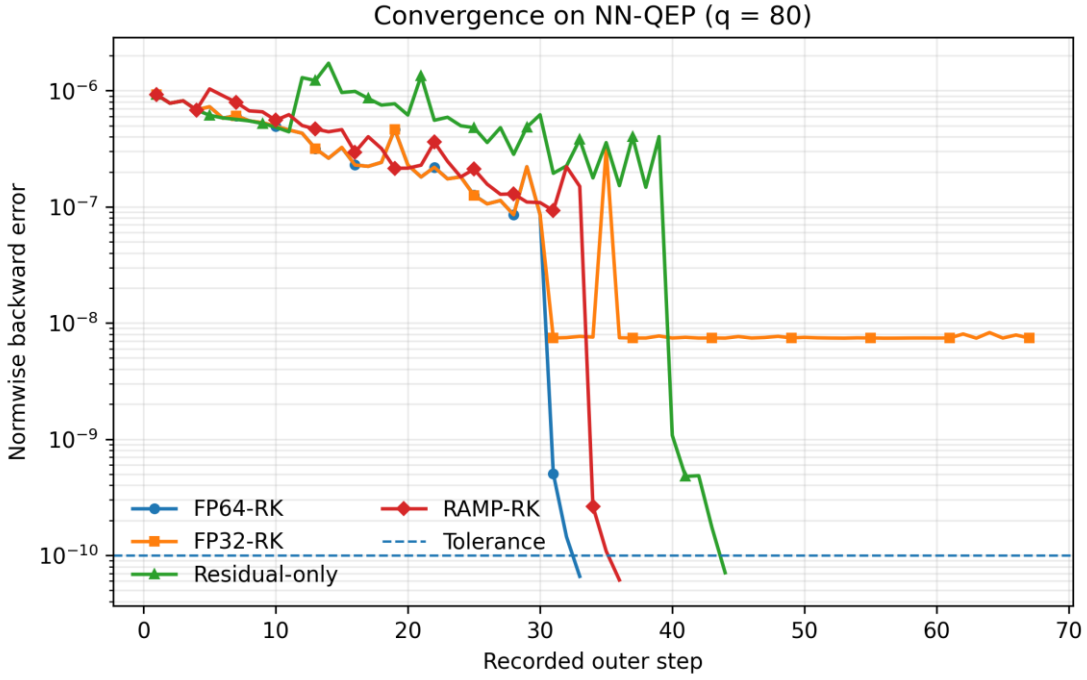


Figure 2. Normwise backward-error histories for the strongly nonnormal NN-QEP at $q = 80$. The condition-aware policy escalates before the residual-only policy completes an unsuccessful polishing phase.

Table 2 gives detailed results for the largest instances. “FP32 solves” refers to shifted triangular solves and binary32 triangular solves that are implemented in iterative refinement. “FP64 solves” refers to shifted triangular solves that are performed following a binary64 restart. The iteration counts are the accumulated number of outer steps plus the number of steps of the polishing process and should be viewed in relation to the solve counts.

Table 2. Accuracy and precision-composition results at $q = 80$.

Family	Method	Converged	Backward error	FP32 solves	FP64 solves	Newton steps	Fallback
DW-QEP	FP64-RK	Yes	3.22×10^{-11}	0	18	0	No
DW-QEP	FP32-RK	No	7.38×10^{-9}	70	0	0	No
DW-QEP	Residual-only	Yes	7.69×10^{-16}	11	0	2	No
DW-QEP	RAMP-RK	Yes	7.69×10^{-16}	11	0	2	No
RD-NEP	FP64-RK	Yes	8.80×10^{-11}	0	43	0	No
RD-NEP	FP32-RK	No	9.34×10^{-9}	70	0	0	No
RD-NEP	Residual-only	Yes	7.42×10^{-13}	23	4	7	Yes
RD-NEP	RAMP-RK	Yes	7.42×10^{-13}	23	4	7	Yes
NN-QEP	FP64-RK	Yes	6.59×10^{-11}	0	36	0	No
NN-QEP	FP32-RK	No	7.42×10^{-9}	70	0	0	No
NN-QEP	Residual-only	Yes	7.08×10^{-11}	28	36	7	Yes
NN-QEP	RAMP-RK	Yes	6.13×10^{-11}	7	35	0	Yes

5.2 Precision composition and ablation

It is able to distinguish between binary32 and binary64 triangular solves in latter modes of numbers. All sparse solves at DW-QEP are binary32 and expensive. With respect to RD-NEP, the adaptive methods are able to solve 23 binary32 solves and only 4 binary64 solves while FP64-RK only yields

43 binary64 solves. For NN-QEP, RAMP-RK performs a total of 7 binary32 solves, including unsuccessful refinements, before a direct condition-aware restart, and 28 binary32 solves, including unsuccessful refinement, before the restart for the residual-only method.

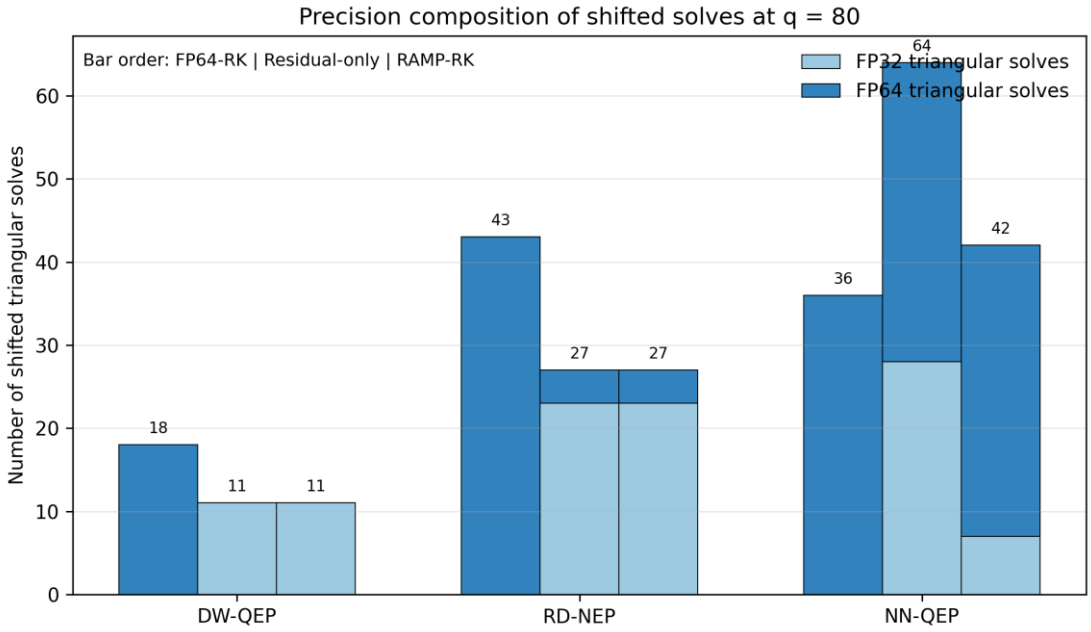


Figure 3. Precision composition of shifted and refinement triangular solves at $q = 80$. Within each family, bars are ordered as FP64-RK, Residual-only, and RAMP-RK.

Comparing the residual-only vs RAMP-RK is an ablation of the condition term. Well-conditioned DW-QEP instances visit the same path by both controllers. Again, they embark on a similarRoute – the condition surrogate their project is below the safeguard until they are given a try at polishing, and, in both instances, they adopt a short binary64 completion. Rule of NN-QEP: case of conditioning—does modify the decision materially. In the latter case the residual-only policy decides to take seven Newton steps, using 21-28 binary32 solves, before falling back to binary64 at q=39, 46, 60, and 80 without ever taking a Newton step itself. For this reason, the condition surrogate comes very much in handy when an individual small residual itself presents an unduly optimistic indication of the confidence in low precision correction.

5.3 Wall time and storage

Table 3 reports median wall time for $q = 80$. The speedup is defined as

$$S_{64} = \frac{t_{FP64-RK}}{t_{RAMP-RK}}. \quad (20)$$

Compared to FP64-RK, RAMP-RK is faster on all three largest cases (speedup ranges from 1.15 to 1.45). The gain is considerable, but limited by the fact that the experiment uses binary64 for basis storage, orthogonalization, reduced eigensolves and the computation of the residual, and that a CPU sparse LU factorization does not reveal the entire throughput discrepancy that exists on accelerators. Thus the aim of the experiment is to show a safe work migration and not Maximum Speedup for an Architecture.

Table 3. Median wall time and relative speedups at $q = 80$.

Family	FP64-RK (s)	Residual-only (s)	RAMP-RK (s)	S_{64}	RAMP vs. residual-only
DW-QEP	0.360	0.353	0.309	1.17	1.14
RD-NEP	1.700	1.110	1.173	1.45	0.95
NN-QEP	0.858	1.534	0.745	1.15	2.06

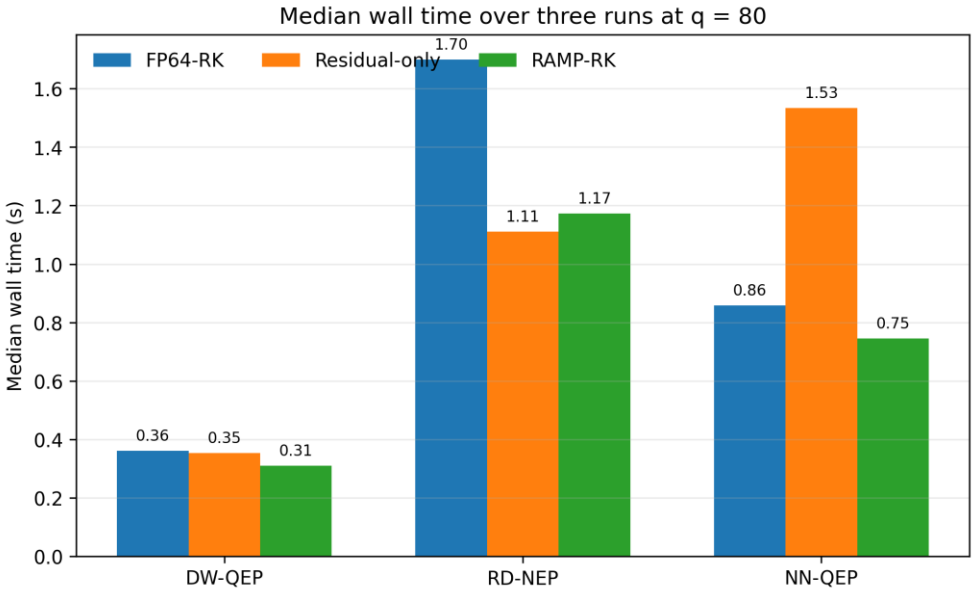


Figure 4. Median single-thread wall time over three runs for the largest grid.

The residual-only policy outperforms RAMP-RK in terms of speed on RD-NEP (1.110 s vs. 1.173 s). This is a small difference in the very short run and shows that a condition calculation can be done, and that the cost of that calculation will not be zero, even do not change the selected path. The same diagnostic, instead, reduces the time remaining in the residual-only mode for NN-QEP by more than half, by omitting an ineffective polishing mode. Thus, the real strength of the controller is its durability over conditioning regimes as opposed to controlling everything.

A binary32 numeric LU factors use approximately half as much memory space as a complex binary64 LU factors. But since no binary64 fallback is necessary, the maximum factor allocation drops to 12.04 MiB (from 24.07 MiB) for DW-QEP @ q=80. The maximum allocation in RD-NEP and NN-QEP ultimately will comprise binary64 factors, and thus be aligned with the FP64 baseline; however, an early low-precision stage will still be able to reduce the amount of data movement. The present Python prototype does not use any such optimizations as explicit factor release and refactorization in an asynchronous way, although these may be possible for a production implementation to reduce peak memory.

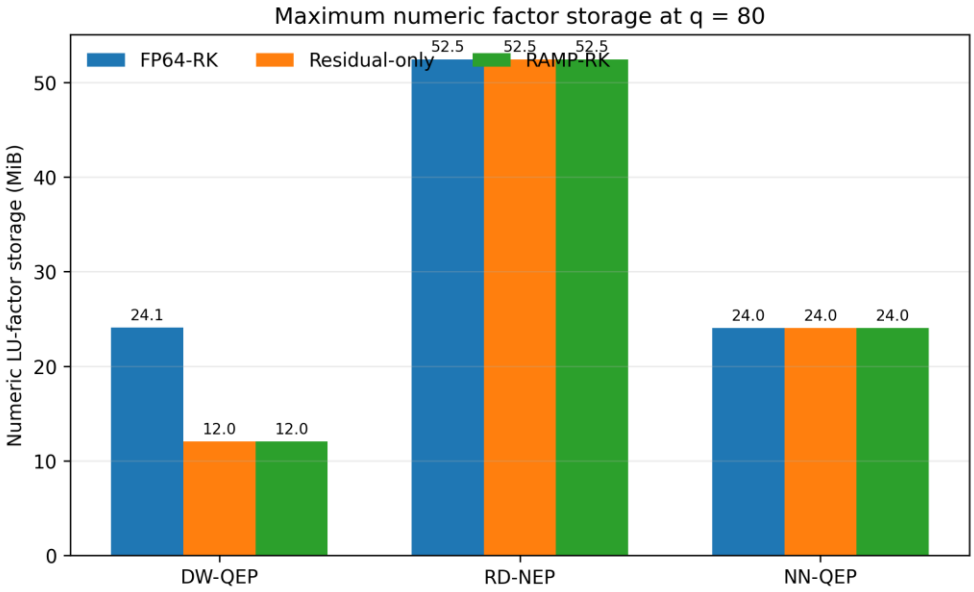


Figure 5. Maximum numeric LU-factor storage at $q = 80$. RAMP-RK halves the factor storage when binary64 fallback is not needed; cases that activate the safeguard eventually allocate binary64 factors.

5.4 Scaling observations

The qualitative behavior doesn't change for the values q=30,45,60,80. Usually RAMP-RK will be convergent on DW-QEP with just one or two binary32 shifted solves plus one or two corrections. Low precision is effective for the smaller instances, while the q=60 and q=80 problems require fallback, suggesting that the cubic linearization becomes more troublesome approaching the problem's target value of q. On NN-QEP, the condition safeguard is also always enabled, and the advantage compared to the residual-only policy increases with q. All these observations have pointed to the main design criterion: The optimal time to introduce increased precision will be related to the residual scale and the sensitivity of the projected eigenpair at hand.

6. Limitations and Reproducibility

To conduct the experiments, synthetic sparse families with controlled structure are used instead of the application matrices taken from a public NEP benchmark collection. It should not be viewed as a hardware independent performance model as this reported timing study is single-threaded and three repetitions. The pole and shift that monotonically rationalizes a production subview is a mononormalized code which is typically modified by a normal production. The old fashioned condition surrogate that can be computed in projected mode is deliberately kept low cost and does not substitute the use of a full nonlinear eigenpair condition number. Lastly, the existing release brings binary32 and binary64 support; binary16, bfloat16, tensor-core kernels and distributed sparse factorizations are planned for the future.

However, even with these drawbacks, there are some protective features to safeguard the optimistic reporting of the research. All the solutions returned are certified with the backward error of original nonlinear problems. All failures of uniform FP32 are kept and are not ignored. The median timings, which refers just to the deterministic iterations and accuracy values, are reported separately. The implementation of the residual-only controller is given as an ablation, and the result of RD-NEP, where it is marginally faster than RAMP-RK is given explicitly.

Scripts for implementation, building benchmarks, raw histories, figure generation is provided by the corresponding author. All random initialisations are deterministic and, in Section 5, the parameters needed to duplicate the tables are given.

7. Conclusions

RAMP-RK gives an adaptive precision mechanism for solving large-size nonlinear eigenvalue problems by rational Krylov method. The method employs binary32 sparse solves and the error is sufficiently large to be unaffected by precision-induced perturbations, while the reliability is further enhanced by binary64 projection, residual evaluation, reorthogonalisation and certification. A condition surrogate is evaluated during projection and determines if a problem can be mixed-precision Newton polished, or is likely benefit from an immediate binary64 restart.

Experiments demonstrate that uniform binary32 arithmetic proves to be effective for generating the initial subspaces but is not adequate for a target nonlinear backward error of 10^{-10}. RAMP-RK is guaranteed to converge for all the cases tested here, and is 1.15-1.45 times faster than the binary64 baseline on the largest cases, without paying a significant penalty for wasted correction work on the strongly nonnormal family. The approach thus provides a useful route for precision adaptive NEP-solver where arithmetic cost is adapted to the changing numerical challenge for the desired eigenpair.

References

[1] N. J. Higham and T. Mary, "Mixed precision algorithms in numerical linear algebra," *Acta Numerica*, vol. 31, pp. 347-414, 2022, doi: 10.1017/S0962492922000022.

[2] E. Carson and N. Khan, "Mixed precision iterative refinement with sparse approximate inverse preconditioning," *SIAM Journal on Scientific Computing*, vol. 45, no. 3, pp. C131-C153, 2023, doi: 10.1137/22M1487709.

[3] N. J. Higham and S. Pranesh, "Exploiting lower precision arithmetic in solving symmetric positive definite linear systems and least squares problems," *SIAM Journal on Scientific Computing*, vol. 43, no. 1, pp. A258-A277, 2021, doi: 10.1137/19M1298263.

[4] S. Graillat, F. Jézéquel, T. Mary, and R. Molina, "Adaptive precision sparse matrix-vector product and its application to Krylov solvers," *SIAM Journal on Scientific Computing*, vol. 46, no. 1, pp. C30-C56, 2024, doi: 10.1137/22M1522619.

[5] E. Carson and I. Daužickaitė, "Single-pass Nyström approximation in mixed precision," *SIAM Journal on Matrix Analysis and Applications*, vol. 45, no. 3, pp. 1361-1391, 2024, doi: 10.1137/22M154079X.

[6] T. Xu, Z. Zhang, J. Chen, Y. Saad, and Y. Xi, "Mixed-precision orthogonalization-free projection methods for eigenvalue and singular value problems," *SIAM Journal on Scientific Computing*, vol. 48, no. 3, pp. B544-B568, 2026, doi: 10.1137/25M1756922.

[7] Z. Tang and Y. Saad, "A rational-Chebyshev projection method for nonlinear eigenvalue problems," *Numerical Linear Algebra with Applications*, vol. 31, no. 5, Art. e2563, 2024, doi: 10.1002/nla.2563.

[8] C. Campos and J. E. Roman, "NEP: A module for the parallel solution of nonlinear eigenvalue problems in SLEPc," *ACM Transactions on Mathematical Software*, vol. 47, no. 3, Art. 23, 2021, doi: 10.1145/3447544.

[9] I. Simunec, "Error bounds for the approximation of matrix functions with rational Krylov methods," *Numerical Linear Algebra with Applications*, vol. 31, no. 5, Art. e2571, 2024, doi: 10.1002/nla.2571.

[10] K. Lund, "Adaptively restarted block Krylov subspace methods with low-synchronization skeletons," *Numerical Algorithms*, vol. 93, pp. 731-764, 2023, doi: 10.1007/s11075-022-01437-1.

[11] V. Simoncini and Y. Wang, "Stabilized Krylov subspace recurrences via randomized sketching," *Numerical Linear Algebra with Applications*, vol. 32, Art. e70022, 2025, doi: 10.1002/nla.70022.

[12] C. C. Paige, "Analyzing vector orthogonalization algorithms," *SIAM Journal on Matrix Analysis and Applications*, vol. 45, no. 2, pp. 829-846, 2024, doi: 10.1137/22M1519523.

[13] D. Kressner and B. Plestenjak, "Analysis of eigenvalue condition numbers for a class of randomized numerical methods for singular matrix pencils," *BIT Numerical Mathematics*, vol. 64, Art. 32, 2024, doi: 10.1007/s10543-024-01033-w.

[14] F. Lopez and T. Mary, "Mixed precision LU factorization on GPU tensor cores: Reducing data movement and memory footprint," *International Journal of High Performance Computing Applications*, vol. 37, no. 2, pp. 165-179, 2023, doi: 10.1177/10943420221136848.

[15] P. Luszczek, A. Castaldo, Y. M. Tsai, D. Mishler, and J. Dongarra, "Numerical eigen-spectrum slicing, accurate orthogonal eigen-basis, and mixed-precision eigenvalue refinement using OpenMP data-dependent tasks and accelerator offload," *The International Journal of High Performance Computing Applications*, vol. 38, no. 6, pp. 671-691, 2024, doi: 10.1177/10943420241281050.

[16] N. J. Higham, F. Tisseur, M. Webb, and Z. Zhou, "Computing accurate eigenvalues using a mixed-precision Jacobi algorithm," *SIAM Journal on Matrix Analysis and Applications*, vol. 46, no. 4, pp. 2423-2448, 2025, doi: 10.1137/25M1723748.