www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 17(5), 2026, 61-79. ISSN: 2007-1558.
https://doi.org/10.61467/2007.1558.2026.v17i5.1420

Cayley–Hamilton-Guided Krylov Regularization for Machine Learning:

Theory, Computational Architecture, and Empirical Evidence

Jaime Aguilar-Ortiz
Universidad Politécnica de Pachuca
E-mail: jao@upp.edu.mx

Abstract: This study develops and evaluates a Cayley–Hamilton-based framework for regularized machine learning. Its objective is to formulate a practical Krylov-subspace architecture for classification that replaces repeated dense inversion with stable operator iteration, drawing on the theorem's finite polynomial closure for square matrices. The analysis connects the characteristic-polynomial identity with regularized least-squares learning, derives full and truncated conjugate-gradient solvers, and evaluates the proposed method on four datasets. Full conjugate gradients reproduce the direct solution to machine precision, whereas truncated iteration preserves competitive predictive accuracy while reducing training cost and improving tolerance to perturbations in ill-conditioned data. These findings establish the Cayley–Hamilton theorem not merely as an algebraic identity, but as a design principle for finite-dimensional learning operators, efficient computation, and interpretable matrix-polynomial models.
Keywords: Cayley–Hamilton theorem, matrix polynomials, Krylov subspaces, regularized least-squares classification, conjugate gradient, machine learning operators.

Article Info
Received April 24, 2026
Accepted Aug 23, 2026

1. Introduction

1.1. Problem statement and research gap

Although supervised-learning pipelines may initially appear quite different, many ultimately depend on a limited family of linear operators, including normal matrices, Gram matrices, covariance operators, and regularized system matrices. In ridge-type estimation and regularized least-squares classification, the spectrum and conditioning of the associated linear system determine both the statistical behavior of the estimator and its computational cost (Hoerl & Kennard, 1970; Björck, 1996; Golub et al., 1999; Rifkin et al., 2003; Hastie et al., 2009). Krylov methods are particularly well suited to this setting because they approximate inverse actions through repeated matrix–vector products. They often eliminate the need for explicit dense inversion while retaining the approximation guarantees established in classical numerical linear algebra (Hestenes & Stiefel, 1952; Greenbaum, 1997; Saad, 2003).

The argument developed here begins with a familiar, yet underused, observation that provides the requisite organizing principle for these operators. The Cayley–Hamilton theorem states that every square matrix satisfies its own characteristic polynomial. Consequently, the associated operational closure is a finite polynomial closure: once the dimension is fixed, sufficiently high matrix powers can be expressed as combinations of lower powers, and inverse-like actions of nonsingular matrices remain within the finite algebra generated by the matrix itself (Horn & Johnson, 2012; Gohberg et al., 2009). This elementary result from matrix theory rarely occupies a central role in regularized learning models. Polynomial approximations and Krylov solvers are, of course, standard instruments in numerical linear algebra and inverse problems. Recent studies in predictive modeling have likewise connected partial least squares, steepest descent, and conjugate-gradient-type procedures with Krylov geometry and regularization. Nevertheless, the Cayley–Hamilton theorem is more often treated as background theory than as an architectural principle governing the learning rule itself (Gazzola et al., 2015; Gazzola & Sabaté Landman, 2020; Qin et al., 2023).

This omission is consequential at both the conceptual and computational levels. When the theorem is left implicit, Krylov and polynomial solvers can appear to be merely convenient numerical shortcuts. Once the Cayley–Hamilton framework is made explicit, however, recurrence structure, truncation depth, and early stopping acquire a more precise interpretation: each determines the degree and spectral selectivity of a finite matrix polynomial. This perspective accords with the regularization literature, in which truncated iteration and spectral damping are understood as controlled forms of bias rather than as incomplete optimization (Engl et al., 1996; Golub et al., 1999; Gazzola & Sabaté Landman, 2020).

Regularized least-squares classification provides the bridge developed in this paper. The learning rule is governed by the regularized normal matrix $A_\lambda = \tilde{X}^T\tilde{X} + \lambda I$, where $\tilde{X}$ is the augmented design matrix and $\lambda > 0$ is the regularization parameter. Matrices of this form are symmetric positive definite and therefore suitable for conjugate-gradient iteration and finite-dimensional polynomial analysis. By the Cayley–Hamilton theorem, the inverse appearing in the closed-form solution belongs to the polynomial space generated by A_λ. Krylov methods provide a stable means of realizing that polynomial action without explicitly forming the numerically delicate coefficients of the characteristic polynomial (Rifkin et al., 2003; Suykens & Vandewalle, 1999; Saad, 2003).

The motivation deliberately operates in two directions. On the one hand, the study recovers a classical result as an active computational principle for machine learning. On the other, it translates that principle into a reproducible architecture and mechanism rather than invoking it only as a metaphor. The argument does not assert that every algorithm should be recast in Cayley–Hamilton form, nor that a theorem-guided method will outperform optimized alternatives on every criterion. Its claim is narrower: for a learning problem formulated through regularized finite-dimensional operators, the theorem provides a principled basis for matrix-polynomial solvers, Krylov-subspace interpretation, and controlled trade-offs among accuracy, cost, and stability. The contribution is therefore specific and testable, and the theorem's value is assessed through exactness, stability, and interpretability rather than through an unqualified promise of superiority.

A meaningful contribution in this setting must do more than place a known theorem beside a known algorithm; it must show how the theorem changes the formulation, implementation, and empirical interpretation of the learning problem. This study addresses that requirement by deriving a theorem-guided regularized classifier, implementing it as a modular computational architecture, and evaluating it on benchmark datasets in terms of exactness, prediction, efficiency, robustness, and interpretability. Accordingly, the Cayley–Hamilton theorem is treated not as a historical footnote in matrix algebra, but as a design grammar for finite-dimensional learning operators. The analysis traces the relationship among the theorem, the actual role of the recurrence, and the empirical consequences of those choices.

The Krylov-based classifier offers greater algebraic transparency than support vector machines, random forests, and neural networks. SVMs often perform effectively in margin-based classification, but require careful selection of kernels and parameters. Random forests are robust and accessible, although their ensemble structure makes the final decision less mathematically explicit. Neural networks can represent complex nonlinear relationships, but generally demand more data, tuning, and computational resources. By contrast, the Krylov-based classifier is especially attractive when the objective is a stable, efficient, and interpretable regularized solution that avoids direct matrix inversion. Its principal advantages are numerical stability, controlled computational cost, and an explicit connection to learning through matrix polynomials.

1.2. Contributions and article organization

The study makes four interrelated contributions. First, it reformulates regularized least-squares classification as a finite matrix-polynomial inference problem whose admissible operator space is constrained by the Cayley–Hamilton theorem. Second, it develops a Cayley–Hamilton-guided Krylov procedure in which Full CG recovers the direct regularized solution, whereas TCG acts as a low-order polynomial regularizer. Third, it organizes the computation as a modular engine that separates data preparation, operator construction, solver execution, evaluation, and artifact export. Fourth, it provides empirical evidence that the full Krylov realization reproduces the direct solution to numerical precision, while truncation preserves much of its predictive utility at lower computational cost and, for selected ill-conditioned data, with greater tolerance to perturbations. These contributions should be considered jointly, because a theoretical reformulation becomes useful only when it also changes implementation and empirical diagnosis.

These contributions are situated within the classical literature on iterative solvers and inverse problems, where Krylov recurrences are routinely interpreted as structured polynomial processes and stopping rules as regularizing decisions (Hestenes & Stiefel, 1952; Engl et al., 1996; Greenbaum, 1997; Gazzola et al., 2015; Gazzola & Sabaté Landman, 2020). The present study extends this perspective to regularized least-squares classification by using Cayley–Hamilton closure to place the exact solution, truncated iteration, and their computational interpretation within a single finite-operator framework.

The article is organized as follows. Section 2 establishes the theoretical foundations by introducing the theorem, deriving the regularized-operator interpretation, and connecting it to Krylov subspaces. Section 3 presents the experimental design, benchmark datasets, evaluation metrics, and Cayley–Hamilton-guided conjugate-gradient protocol. Section 4 describes the

computational architecture required to reproduce the theoretical construction. Section 5 examines the tables and figures in relation to exactness, predictive performance, efficiency, robustness, and interpretability. Section 6 discusses the evidence and draws the principal conclusions. Section 7 outlines future work on larger-scale, kernelized, and graph-structured learning systems. This organization allows the reader to move from algebraic foundations to empirical analysis while retaining the same operator throughout the development.

This article contributes to regularized machine learning by developing and validating an algorithmic framework grounded in the Cayley–Hamilton theorem. Under this interpretation, the regularized least-squares solution is a finite matrix-polynomial construction that can be implemented efficiently with Krylov-subspace methods. The approach replaces matrix inversion with numerically stable iterative operators while preserving the accuracy of the exact solution when conjugate gradients are run to full convergence. Controlled truncation reduces computational cost while maintaining competitive predictive performance. The framework also exhibits greater tolerance to perturbations and numerical instability in ill-conditioned datasets. These results illustrate how an established theorem from linear algebra can provide a useful foundation for scalable, interpretable, and efficient machine-learning algorithms.

2. Theoretical foundations

2.1. Cayley–Hamilton Theorem and finite polynomial operator spaces

The analysis begins with the characteristic polynomial of a square matrix. For a matrix $A \in \mathbb{R}^{n\times n}$, this polynomial is written as

$$p_A(\lambda) = \det(\lambda I - A) = \lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_0 \quad (1)$$

Here, I denotes the $n \times n$ identity matrix, λ is a scalar indeterminate, and $c_0, c_1, \ldots, c_{n-1}$ are the characteristic coefficients determined by A. The Cayley–Hamilton theorem states that replacing the polynomial variable with the matrix itself causes the polynomial to annihilate that matrix.

$$p_A(A) = A^n + c_{n-1}A^{n-1} + \cdots + c_1A + c_0I = 0 \quad (2)$$

Equation (2) expresses more than a symbolic identity. It establishes the linear dependence of the matrix family generated by $I, A, A2, .., A^{n`}\}$, so every power of A of degree n or higher can be reduced to a combination of lower powers. In finite dimension, the algebra generated by A therefore closes after finitely many elements. A square matrix does not produce an indefinitely expanding family of operators through its powers alone; instead, those powers remain confined to a finite polynomial space. Matrix analysis and matrix-polynomial theory use this principle to characterize operator reductions, polynomial identities, and finite-dimensional representations of matrix functions (Horn and Johnson, 2012; Gohberg et al., 2009). The underlying point is elementary but consequential: finite dimension imposes an algebraic ceiling on the complexity of operator constructions based solely on matrix powers.

If A is nonsingular, Equation (2) also yields a polynomial expression for its inverse. Because nonsingularity implies $c_0 \neq 0$, multiplying the identity by $-c_0^{-1}$ and rearranging gives

$$A^{-1} = -\frac{1}{c_0}(A^{n-1} + c_{n-1}A^{n-2} + \cdots + c_2A + c_1I) \quad (3)$$

Equation (3) is directly relevant to machine learning because many learning rules apply an inverse operator to a data-dependent vector or matrix. The Cayley–Hamilton theorem shows that, in finite dimensions, such inverse-like linear actions remain within polynomial calculus: they belong to a finite family of matrix polynomials generated by the same operator. This property is important when explicit inversion is computationally costly, numerically unstable, or difficult to interpret (Horn & Johnson, 2012; Gohberg et al, 2009).

The spectral interpretation expresses the same principle differently. If A is diagonalizable as $A = Q\Lambda Q - 1$, with Q collecting the eigenvectors and Λ diagonal, then every polynomial in A acts through the same eigenvectors, with the polynomial evaluated at the corresponding eigenvalues. Therefore,

$$q(A) = Q\, q(\Lambda)\, Q^{-1} \quad (4)$$

for any scalar polynomial p, $p(\Lambda)$ is the diagonal matrix whose entries are $p(\lambda i)$ for the eigenvalues λi of A. Equation (4) therefore recasts polynomial-operator design as an eigenvalue-design problem. This reasoning also supports the approximation of matrix functions without requiring an eigendecomposition in the computation (Horn and Johnson, 2012; Gohberg et al., 2009; Saad, 2003).

The effective polynomial degree may also be smaller than the matrix dimension. Although Equation (2) is stated with degree n, the minimal polynomial of A may have degree m, in which case the operator space generated by A closes in m dimensions rather than n. For iterative solvers, convergence consequently depends more strongly on spectral clustering and the number of distinct eigenvalues than on ambient dimension alone. In learning problems, the expressive complexity of a polynomial operator is therefore governed by the matrix's intrinsic spectrum, not only by the raw number of features (Horn & Johnson, 2012).

The theorem thus provides a concrete rationale for machine learning: an operator acting on finite-dimensional data can be studied through a bounded polynomial basis. This perspective aligns naturally with Krylov techniques, which repeatedly apply the same operator to obtain approximate or exact inverse actions. Model complexity can consequently be described in algebraic terms rather than treated merely as an implementation detail (Greenbaum, 1997; Saad, 2003; Qin et al., 2023). The present innovation is to translate that reasoning into a compact classification architecture centered on regularized data operators. Such a formulation is particularly useful for regularized models, in which the tension between fit and stability is already encoded in the operator spectrum.

2.2. Regularized predictive modeling as matrix-polynomial inference

Regularized least-squares multiclass classification provides the bridge from operator theory to learning (Rifkin et al. 2003; Suykens & Vandewalle 1999). Let X denote the design matrix with m samples and d variables. Let $\tilde{X}$ be the augmented design matrix obtained by appending a bias column of ones. Let Y denote the one-hot response matrix for c classes, and let W be the weight matrix. The regularized empirical objective is:

$$\mathcal{J}(W) = \| Y - X_a W \|_F^2 + \lambda \| W \|_F^2 \tag{5}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and λ is the regularization parameter. The first term in (5) measures prediction error, whereas the second controls coefficient magnitude. The parameter λ therefore governs the trade-off between fitting the data and smoothing the operator. Statistically, the objective is a ridge- or Tikhonov-regularized quadratic loss; numerically, λ shifts the spectrum away from zero and stabilizes the normal matrix (Hoerl & Kennard, 1970; Engl et al., 1996; Hastie et al., 2009).

Differentiating Equation (5) with respect to W and setting the gradient equal to zero yields the normal equations:

$$(X_a^\top X_a + \lambda I) W = X_a^\top Y \tag{6}$$

where I is the identity matrix in $\mathbb{R}^{(d+1)\times(d+1)}$. Define the regularized operator

$$A_\lambda = X_a^\top X_a + \lambda I \tag{7}$$

and define the cross-covariance term

$$G = X_a^\top Y \tag{8}$$

Then the solution is

$$W_\lambda = A_\lambda^{-1} G \tag{9}$$

For a new input vector x, the prediction is given by

$$\hat{y} = \arg\max_{1 \le j \le C} ([x^\top \ 1] W_\lambda)_j \tag{10}$$

The resulting class scores are compared across categories. Equations (5)–(10) define the standard regularized least-squares classifier. The Cayley–Hamilton contribution lies in the interpretation of Equation (9): because the regularized operator is square and nonsingular when regularization is positive, Equation (3) shows that its inverse belongs to a finite polynomial space generated by that same operator. Thus,

$$A_\lambda^{-1} = q_{m-1}(A_\lambda) \tag{11}$$

for a polynomial q_{m-1} of degree at most $m-1$, where m is the degree of the minimal polynomial of A_λ. Equation (9) then yields:

$$W_\lambda = q_{m-1}(A_\lambda) G \tag{12}$$

Equation (12) constitutes the principal conceptual pivot of the article: it permits regularized learning to be viewed as matrix-polynomial inference. The learned operator is not merely the outcome of an inverse; rather, it is the action of a finite polynomial in the regularized data matrix on a data-derived right-hand side. This interpretation exposes the solution family, its spectral degrees of freedom, and the role of iterative approximation within a single algebraic description. It also allows exactness, truncation, and robustness to be examined in the subsequent experiments as properties of one coherent polynomial family.

Several consequences follow from this formulation. First, the solution lies within a bounded operator family. Equation (12) shows that, instead of treating the inverse as a "black box," the effective classifier can be represented through repeated applications of the same data operator. This representation provides a precise language for describing spectral bias, regularization strength, and iterative approximation. It also connects regularized prediction with the conventional interpretation of Krylov techniques as adaptive polynomial approximators of inverse actions. Moreover, it clarifies the meaning of early stopping: if the exact inverse is a finite polynomial, then a lower-degree approximation is not a fundamentally different object, but a reduced model within the same admissible family (Engl et al. 1996, Gazzola et al. 2015, Qin et al. 2023). The resulting model class therefore has a complexity that is explicit in its polynomial degree, spectrum, and stopping policy rather than concealed within implementation details.

The operator perspective also improves reproducibility. Once the regularized operator A_λ and the right-hand side B are fixed, the classifier follows from an explicit algebraic map rather than an opaque sequence of heuristic updates. Implementation then concerns only how the polynomial action is realized: directly or through an iterative recurrence.

This interpretation also explains truncation. If the inverse is a polynomial of degree m, any lower-degree approximation is a reduced operator model rather than a conceptually unrelated surrogate. Truncated conjugate gradients may therefore be understood as a controlled form of polynomial regularization. Although truncation restricts the degree and thus the available spectral resolution, the resulting approximation remains within the same theorem-guided operator family (Engl et al., 1996; Gazzola & Sabaté Landman, 2020).

2.3. Krylov-subspace interpretation of learning operators

The next step relates the finite-polynomial interpretation to a concrete computational mechanism. Let A_λ be the regularized operator defined by Equation (7), and let g denote one column of G. The Krylov subspace generated from g through repeated applications of A_λ is:

$$\mathcal{K}_k(A_\lambda, g) = \text{span}\{g, A_\lambda g, A_\lambda^2 g, \dots, A_\lambda^{k-1} g\} \tag{13}$$

Equation (13) makes the theorem computationally useful: a Krylov subspace consists of polynomial-operator actions on a given vector. Any element $z \in \mathcal{K}_k(A_\lambda, g)$ can be written as:

$$z = q_{k-1}(A_\lambda) g \tag{14}$$

for some polynomial q_{k-1} of degree at most $k-1$. Because Equation (12) expresses the exact solution as a finite polynomial action, Krylov methods are not external approximators in this setting. They are constructive procedures that traverse the same admissible operator family while selecting coefficients from the data and the solver recurrence.

Conjugate gradients are particularly appropriate here because A_λ is symmetric positive definite whenever $\lambda > 0$. For a single right-hand side b, the method produces a sequence x_k that minimizes the error in the A_λ-energy norm over the current Krylov subspace. Standard analyses emphasize both finite termination in exact arithmetic and the method's character as a polynomial approximation process (Greenbaum, 1997; Saad, 2003). The recurrence starts from an initial guess x_0, a residual r_0, and a search direction p_0. Its iterations proceed as follows.

$$\alpha_k = \frac{r_k^\top r_k}{p_k^\top A_\lambda p_k}, \qquad z_{k+1} = z_k + \alpha_k p_k, \qquad r_{k+1} = r_k - \alpha_k A_\lambda p_k \tag{15}$$

and

$$\beta_k = \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k}, \qquad p_{k+1} = r_{k+1} + \beta_k p_k \tag{16}$$

In Equations (15)–(16), αk denotes the step length, β_k updates the conjugate search direction, r_k is the residual at step k, and p_k is the direction vector. The iterate z_k lies in $\mathcal{K}_k(A_\lambda, g)$, and z_k denotes that same intermediate solution. Accordingly, every intermediate solution is already a polynomial model of the regularized operator. In exact arithmetic, full convergence yields

the precise regularized solution after at most as many steps as there are distinct eigenvalues. Truncated convergence instead permits the lower-degree polynomial model to be controlled through the iteration budget.
This interpretation accords with the inverse-problem literature, in which Krylov iterations are viewed as polynomial filters and truncated iterative procedures as regularization devices. Within that framework, both the subspace dimension and the spectral information incorporated at each iteration are explicitly controlled. Together, these controls determine the quality of the approximation and the extent of noise amplification (Engl et al., 1996; Gazzola et al., 2015; Gazzola & Sabaté Landman, 2020).

The Cayley-Hamilton framework adds two clarifications to the conventional account of Krylov methods. First, it clarifies why the subspace construction is algebraically natural: rather than merely forming an approximation, the solver traverses a space known to contain the exact finite-dimensional solution. Second, it gives early stopping a complete model-based interpretation. If the process terminates at step k, the operator complexity is at most $k-1$. Truncated conjugate gradients can therefore be understood as an explicit bias mechanism, not simply as incomplete optimization. This restriction may reduce sensitivity to high-frequency directions or to directions associated with small eigenvalues, while also producing a smoothing effect.

If W_k denotes the multi-class weight matrix after k column-wise Krylov iterations applied to G, then:

$$W_k = q_{k-1}(A_\lambda)G \tag{17}$$

where q_{k-1} is specified implicitly by the recurrence. When the solver is permitted to converge, the solution W_λ from (12) is complete. By contrast, the truncated solution W_k is a low-order approximation within the same operator space. This provides the basis for the computational architecture presented below.

In summary

The Krylov-based classifier establishes a direct connection among data preparation, model construction, and prediction. In the initial stage, the dataset is prepared by addressing missing values, converting categorical information into numerical form when necessary, and normalizing or standardizing the features. These operations prevent variables measured on different scales from exerting disproportionate influence on learning. The data are then divided into training and testing partitions, so that the model is fitted on one partition and evaluated on the other.

The objective is to minimize a function comprising a classification loss and a regularization term. This process learns a set of coefficients that relates the input features to the target classes without becoming excessively sensitive to noise or other random fluctuations in the data. At this stage, the regularization term helps improve numerical stability and the regularized model's capacity to generalize to unseen observations.

Rather than computing the solution through direct matrix inversion, which is often computationally expensive and unstable for large or ill-conditioned data sets, the proposed technique employs a Krylov-subspace approach. Beginning from an initial approximation, the algorithm progressively refines the approximate solution vector through repeated matrix–vector multiplication. The Cayley–Hamilton theorem provides the theoretical justification for this procedure by guaranteeing that the matrix action can be expressed as a finite polynomial. Accordingly, learning can be interpreted as the construction of progressively improved polynomial approximations to the desired solution.

The conjugate-gradient algorithm is used to estimate the classifier parameters. When the iterative process is allowed to reach convergence, its solution is numerically the exact regularized least-squares solution. If the iterations are stopped earlier, however, the resulting model is designated as truncated because it requires less computation than the full solution while retaining much of its predictive power. The principal practical advantage of the proposed framework is therefore its trade-off between accuracy and efficiency.

Once the model parameters have been obtained, the classifier is applied to unseen samples. After the feature vector for an observation has been identified, it is processed by the learned matrix polynomial operator and assigned a prediction score that reflects its correlation with each available class. The final prediction corresponds to the class with the highest score.

The final stage assesses the efficiency of the proposed method. Predictive performance is evaluated using established classification criteria, including accuracy, precision, recall, and F-Score, whereas computational efficiency is assessed through training time and convergence behavior. Additional robustness analyses can also be conducted by introducing perturbations or examining ill-conditioned datasets. These analyses provide a fuller understanding of the stability and practical reliability

of the Krylov-based classification framework. Overall, the workflow translates a classical theorem of linear algebra into a concrete and computationally efficient machine learning procedure.

3. Methodology

3.1. Experimental design, datasets, and evaluation metrics

The experimental design was guided by three considerations derived from the claims presented in Section 2. The first is algebraic exactness: a full Krylov realization should reproduce a direct regularized least-squares solution to numerical precision because both instantiate the same finite-dimensional operator. The second is predictive adequacy: even when truncated, the theorem-informed Krylov model should remain competitive on standard classification benchmarks. The third is computational usefulness: the truncated polynomial realization should reduce training time and may, under perturbation, offer greater stability in ill-conditioned settings.

To evaluate these claims, the method was tested on four benchmark datasets (iris, wine, breast cancer, and digits) available offline through scikit-learn – Pedregosa et al. (2011). The efficient implementation provided by the scientific Python ecosystem makes dataset loading, array computation, and solver execution reproducible through well-defined components, as described in (Pedregosa et al., 2011; Virtanen et al., 2020). The selected suite spans several dimensional and statistical regimes. Iris is a small, low-dimensional, three-class problem that is useful for observing short Krylov behavior. Wine contains a range of chemical variables in a relatively mild setting. Breast_cancer exhibits stronger feature correlations and a more challenging condition profile, both of which are important for regularization and perturbation analysis. Digits increases the feature dimension through an 8×8 pixel representation and tests whether the operator perspective remains useful for a richer multiclass structure. These four datasets are not intended to exhaust the potential application domain; rather, they support controlled and reproducible evaluation of the algebraic and computational claims.

For every training fold, the features were normalized using only the mean and standard deviation computed from that fold. The augmented matrix $\tilde{X}$ was formed by appending a bias column, and the multiclass labels were one-hot encoded as a response matrix Y. This design preserves a transparent connection between the data and the regularized operator while producing class-specific scores in matrix form. It is also consistent with simple one-versus-all decompositions for multiclass linear classifiers (Rifkin & Klautau, 2004).

The empirical protocol considered four methods. The 'direct RLS' method is the dense regularized least-squares solution obtained by directly solving equation (6). It serves as the algebraic reference because it is the exact dense realization of the regularized operator. The full CG algorithm solved the same systems by conjugate gradients, using a strict residual tolerance and an iteration budget sufficient for floating-point convergence; it therefore constitutes the full theorem-guided Krylov realization. Truncated CG (TCG) used the same recurrence with a restricted iteration budget, thereby enforcing the low-degree matrix-polynomial interpretation developed in Section 2. Logistic regression, a general linear probabilistic classifier, served as a contextual baseline and provided a familiar benchmark for accuracy and macro-F1 (Hastie et al., 2009). Although it is not algebraically equivalent to regularized least squares, it helps situate the theorem-guided methods among practical classifiers. This design separates the algebraic question from the broader question of which linear classifier is most suitable for a given dataset.

An inner validation procedure was used within each training fold to select the hyperparameters. The regularization grid for Direct RLS and Full CG was $\lambda = 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10$. For TCG, $k \in 1,2,3,5,8,13,21$ was used to tune both the regularization parameter and the iteration budget. Logistic regression used the inverse-regularization grid C, from which $\in \{0.01, 0.1, 1, 10, 100\}$ was selected. Macro-F1 served as the inner selection objective because it gives class-wise performance more equal consideration than accuracy in multiclass settings.

Repeated stratified cross-validation, with five folds and two repetitions, was used for the outer assessment. This protocol offers a practical balance among reproducibility, statistical stability, and computational affordability, making the experiments straightforward to repeat in research or teaching settings. It follows a standard model-assessment methodology that balances bias, variance, and cost (Hastie et al., 2009). For each outer split, the appropriate hyperparameters were fitted on the training partition and evaluated on the test partition, and the metrics described below were recorded. The resulting performance estimates therefore do not depend on a single favorable split, while the experiment remains sufficiently small to inspect and reproduce.

Classification accuracy was calculated as

$$\text{Accuracy} = \frac{1}{N_t} \sum_{i=1}^{N_t} \mathbf{1}\left(\hat{y}_i = y_i\right) \tag{18}$$

In expression (18), N_t denotes the number of test examples, $\hat{y}_i$ is the predicted label for sample i, y_i is the true label, and 1 denotes the indicator function. Macro-F1 was computed as:

$$\text{Macro-F1} = \frac{1}{C}\sum_{j=1}^{C}\frac{2P_jR_j}{P_j+R_j} \tag{19}$$

where P_k and R_k denote, respectively, the precision and recall for class k. Macro averaging assigns equal weight to every class, thereby preventing dominant classes from biasing a classifier under aggregate accuracy. This choice follows standard recommendations for multiclass evaluation when class balance is important (Sokolova & Lapalme, 2009).

Training time was measured for every fitted model. For the Krylov-based methods, average iteration counts were also retained so that polynomial complexity remained visible in the empirical record. A separate exactness experiment was conducted outside the cross-validation pipeline. The complete standardized version of each dataset was used to construct the regularized operator A_λ with $\lambda = 1$. Random-valued right-hand sides were then solved both directly and by conjugate gradients. Relative solution error, final residual, condition number, and convergence iteration count were recorded. Under the conditions of the numerical experiment, these measurements test the fundamental theoretical statement.

The final robustness experiment perturbed both the full and truncated operators. A train-test split with fixed stratification was used for every dataset. Gaussian noise was introduced in the standardized feature space and subsequently mapped back to the raw input scale, ensuring that the perturbation magnitude remained comparable across features measured in different units. The drop measure was then used to compare clean and noisy accuracies.

$$\Delta_{noise} = \text{Acc}_{clean} - \text{Acc}_{noisy} \tag{20}$$

where Δ_{noise} quantifies sensitivity to corruption in the inputs. This measure is analyzed because Section 2 predicts that low-degree polynomial truncation will provide a smoothing bias, particularly in ill-conditioned problems.

The procedure is therefore consistent with numerical linear algebra, inverse-problem theory, and reproducible scientific computing. The solver is not treated as a secondary numerical detail; instead, precision, spectral behavior, and computational cost are regarded as first-order properties of the learning system (Gazzola et al., 2015; Gazzola & Sabaté Landman, 2020; Virtanen et al., 2020). This choice is important because a theorem-guided method must be evaluated both as a predictor and as a numerical realization of an operator identity.

3.2. Cayley–Hamilton-guided conjugate-gradient learning protocol

The computational protocol implements Section 2 through an explicit sequence of operations. Given the training data (X, y), preprocessing standardizes the feature matrix, appends a bias column, and converts the labels into a one-hot matrix Y. Operator construction then forms the regularized system matrix A_λ and the cross-covariance matrix G in accordance with Equations (7) and (8). The solution stage varies with the selected realization. The direct version solves Equation (6) with a dense linear solver. The full theorem-guided version uses conjugate gradients to solve the same system column by column until the residual is negligible. The truncated version stops that recurrence after the selected budget k.

For every class column g_j in G, the column-wise solution is

$$A_\lambda w_j = g_j \tag{21}$$

where $w_j(k)$ denotes the coefficient vector for class j computed by full CG within the Krylov family $\mathcal{K}_k(A_\lambda, g_j)$, using the same regularized operator.

$$W^{(k)} = \left[w_1^{(k)}\ w_2^{(k)}\ \cdots\ w_C^{(k)}\right] \tag{22}$$

and, following the argument in Section 2,

$$W^{(k)} = q_{k-1}(A_\lambda)G \tag{23}$$

for a polynomial q_{k-1} defined implicitly. Equation (23) represents the complete training routine as a polynomial action on a matrix of right-hand sides. Without loss of generality, the mail may be assumed to arrive at time zero, and the model remains essentially unchanged if the time constant is increased.

The stopping policy determines how the model acquires operational meaning. In the complete setting, the algorithm seeks practical equivalence to the direct solve by using a strict tolerance and, when useful, floating-point refinement beyond the nominal feature dimension. This is reasonable because exact arithmetic would terminate after no more iterations than the number of distinct eigenvalues, whereas ill-conditioned floating-point systems may require additional refinement. In the truncated setting, the algorithm stops at the low iteration count selected through validation. The resulting model is a lower-degree matrix polynomial that exchanges some accuracy for lower cost and smoother spectral behavior. In practice, the iteration count is an interpretable modeling choice rather than an obscure solver decision embedded in the code.

This trade-off is reflected in computational complexity. Let $d_a = d + 1$ denote the augmented feature dimension and C the number of classes. A dense direct solution has the following approximate leading cost:

$$\mathcal{C}_{direct} = O(d_a^3 + d_a^2 C) \tag{24}$$

Because factorization dominates, the multiple right-hand sides are solved thereafter. The approximate cost of a Krylov realization using k iterations is:

$$\mathcal{C}_{CG} = O(k d_a^2 C) \tag{25}$$

when the matrix is treated as dense. For sparse or highly structured problems, the cost of a matrix-vector product can be substantially lower, making the Krylov approach considerably more useful in practice. Equations (24) and (25) show why the theorem-guided perspective is computationally meaningful: the same finite-dimensional operator family can be realized through dense inversion or iterative polynomial construction, with the latter exposing the degree parameter k (Greenbaum, 1997; Saad, 2003).

The algorithm employed in this study is formally summarized below.

1. Standardize the training features and subsequently append the bias column.
2. Convert the labels to one-hot encoding.
3. Form $A_\lambda = X_a^\top X_a + \lambda I$ and $G = X_a^\top Y$.
4. For each class column of G, perform a direct solve, full CG, or truncated CG.
5. Assemble the class-wise coefficient vectors into W.
6. Score the test samples using Equation (10).
7. Record the prediction, timing, iteration, residual, and robustness statistics.
8. Save the tables and figures in appropriately named subsection folders so that the empirical narrative can be transferred to the manuscript without compromising traceability.

Despite its compact form, this protocol expresses the central methodological claim: the Cayley–Hamilton theorem becomes computationally active when learning is understood as finite-dimensional matrix–polynomial construction. The implementation deliberately avoids estimating characteristic-polynomial coefficients, which are numerically unstable in moderate and high dimension. Instead, conjugate gradients serve as a stable polynomial generator, consistent with standard practice in numerical linear algebra (Björck, 1996; Greenbaum, 1997; Saad, 2003). The implementation thereby preserves the spirit of the theorem without relying on the numerically unsafe construction of explicit characteristic polynomials.

The experimental outputs are also stored in directories aligned with the article's structure. Exactness and convergence artifacts are placed in subsection 5.1, predictive and efficiency artifacts in subsection 5.2, and robustness and interpretability artifacts in subsection 5.3. This arrangement is not merely decorative. The computational architecture mirrors the paper's explanatory structure, allowing every analytical claim to be traced to its numerical evidence.

4. Computational architecture

4.1. Modular architecture of the CH-Krylov learning engine

The article's theoretical framework has been operationalized and may contribute to advancing the understanding or experience of art. Rather than relying on a single aggregated script in which algebra, optimization, evaluation, and reporting are intermingled, the design separates five layers: Data ingestion and preprocessing, operator construction, solver realization, evaluation and diagnostics, and artifact packaging. This separation preserves a clear relationship among theorem, algorithm, and evidence. Reproducible implementation is further supported because transformations, solvers, and diagnostics remain independently auditable computational components (Pedregosa et al., 2011; Virtanen et al., 2020). A modular design also

reduces the likelihood that empirical conclusions arise from an entangled script, since every stage can be inspected independently.

The first layer, data ingestion and preprocessing, loads the benchmark datasets, creates the cross-validation (i.e., robustness) splits, standardizes each training partition, and appends the bias coordinate. Consequently, the matrix supplied to the learning rule has a numerically stable scale and a consistent affine representation. Standardization is particularly important because the condition numbers of the regularized operator are highly relevant to solver behavior and the interpretation of truncation.

The second layer, operator construction, transforms the preprocessed data into the relevant linear-algebraic statistics used by the theorem-conforming model. Specifically, it constructs A_λ and G. From this point onward, the machine learning problem becomes an operator problem. Given A_λ and G, a sample-independent routine can compute the classifier coefficients. This separation is essential because it shows that training is mediated by a compact finite-dimensional operator rather than by repeated sample-wise optimization.

The third layer, solver realization, comprises three interchangeable engines. The direct engine performs dense linear solves and serves as the reference implementation of the exact regularized operator. The full CG engine carries out theorem-guided Krylov construction until convergence; its polynomial recurrence produces the same operator action. The truncated CG engine limits the number of iterations, thereby producing a lower-degree polynomial approximation to the same finite set of operators. This is the computational stage at which the Cayley–Hamilton interpretation becomes active: the three engines address the same operator problem through different realizations of finite polynomial closure.

The fourth layer, evaluation and diagnostics, derives accuracy, macro-F1, learning time, iteration counts, condition numbers, relative solution errors, residual trajectories, perturbation sensitivity, and coefficient-based interpretability summaries. These outputs are considered equally important. A proof-driven paper must not assess the method solely by task accuracy, because the theorem also makes claims about operator structure and computational behavior. Accordingly, the diagnostic layer records information for verifying the accuracy of its outputs, identifying their depth in polynomial time, and determining how model truncation changes them. This complete diagnostic record makes it possible to distinguish among algebraic equivalence, predictive similarity, and computational convenience.

At the fifth layer, artifact packaging exports every table and figure to folders aligned with the manuscript. Folders 5.1, 5.2 and 5.3 correspond, respectively, to the article's analytical divisions: algebraic exactness, predictive efficiency, and robustness-interpretability evidence. This arrangement strengthens reproducibility by allowing readers and reviewers to connect each subsection's claim directly to its machine-generated artifact, without having to reconstruct the entire workflow.

The learning engine's architectural level is represented by the following transformation chain:

$$(X, y) \to (X_a, Y) \to (A_\lambda, G) \to W \to \hat{y} \to \text{diagnostics} \tag{26}$$

Each arrow denotes a module, while the central transition $(A_\lambda, G) \to W$ is governed by the theorem-driven solver choice. Under the implemented logic, raw records are first formulated as a regularized finite-dimensional operator system and are only subsequently interpreted as predictions and evidence (26).

The software design is deliberately simple, yet conceptually transparent. This balance serves the argument by keeping its theoretical basis visible rather than obscuring it within opaque engineering detail. It also supports extensibility. Larger sparse systems, kernelized variants, and preconditioned Krylov methods can be accommodated by replacing or augmenting the operator-construction and solver layers while preserving the same theorem-guided design. Accordingly, the computational architecture functions not only as an effective platform for the present experiments, but also as a reusable template for finite-operator learning workflows consistent with reproducible scientific computing (Virtanen et al., 2020). The architecture has therefore been kept sufficiently compact for auditing while remaining general enough to support future numerical enhancements.

5. Results and analysis

5.1. Algebraic exactness and convergence behavior

Table 1 reports the exactness probe conducted on the regularized operators induced by the four benchmark datasets. The result is straightforward: in every case, the full conjugate-gradient realization reproduced the direct solution with extremely small relative errors. The average relative solution error was on the order of 2.3×10^{-14} for iris, 2.13×10^{-15} for wine, 7.93×10^{-14} for breast_cancer, and 8.09×10^{-15} for digits. These values are negligible for prediction and demonstrate that, in the full setting, the theorem-guided Krylov realization is not merely an approximation; for practical purposes, it implements the same operator as the dense direct solve.

Table 1. Exactness and convergence diagnostics.

Dataset	Augmented dimension	Distinct eigenvalues	Condition number	Mean CG iterations	Mean relative solution error	Mean final residual
iris	5	5	106.83	5.0	2.30e-14	2.60e-13
wine	14	14	43.23	16.0	2.13e-15	8.08e-15
breast_cancer	31	31	7026.32	63.0	7.93e-14	2.91e-13
digits	65	63	13192.22	85.8	8.09e-15	6.43e-13

The convergence statistics further support this interpretation. In the regularized probe, the iris system, with five eigenvalues, reached closure in an average of 5.0 iterations. The wine operator required 16.0 iterations on average for fourteen distinct eigenvalues, representing only a modest floating-point refinement beyond the exact-arithmetic ideal. The breast cancer problem and the digits system required 63.0 and 85.8 iterations, respectively. These higher counts reinforce the theorem-guided interpretation by revealing how finite precision and conditioning interact within the same finite polynomial space. The exact solution remains within the closure permitted by the theorem; numerical refinement merely requires additional recurrence steps.

Figure 1 displays the convergence pattern. Residuals for iris and wine fall rapidly to machine precision, whereas breast_cancer and digits exhibit a longer, yet still definitive, semilogarithmic decline. The curve shapes accord with the condition numbers in Table 1: iris (106.83), wine (43.23), breast_cancer (7026.32), and digits (13192.22). The convergence difficulty is therefore attributable to spectral conditioning, not to any failure of the theorem-guided formulation.

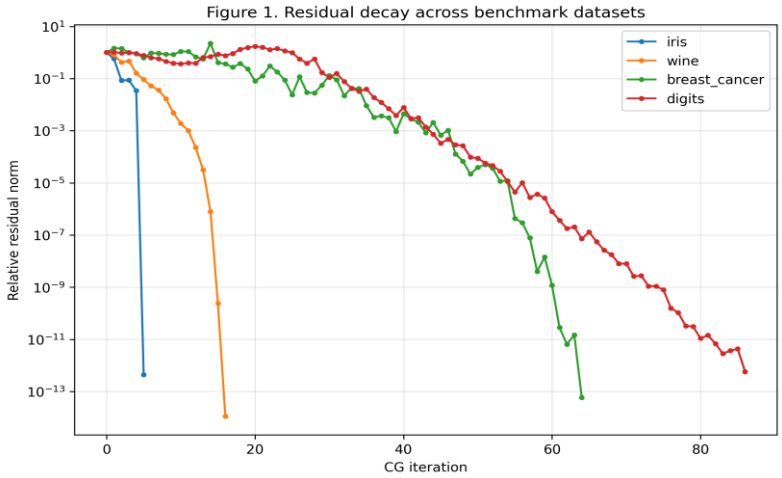

Figure 1. Residual decay across benchmark datasets.

Figure 1 shows that Krylov iteration can fully access the operator family generated by each dataset, although the number of required steps depends on spectral geometry. The abrupt declines for iris and wine indicate that low-dimensional or well-conditioned systems attain finite polynomial closure quickly. A more coherent and less complex spectrum would require a lower-degree polynomial construction than a richer spectrum. Accordingly, a theorem-guided computational architecture must expose iteration depth as a modeling parameter rather than conceal it as an implementation detail.

Figure 2 focuses on breast_cancer, the most informative ill-conditioned case. It traces both the relative residual and the relative solution error as the iteration budget increases. The two curves decline together and eventually level off at machine precision. This behavior confirms that the CG recurrence does more than reduce residuals formally: it converges to the corresponding direct solution. Because breast cancer is subsequently used to assess truncation, establishing the exactness of the full-solver baseline is a prerequisite for evaluating any claims concerning the truncated model.

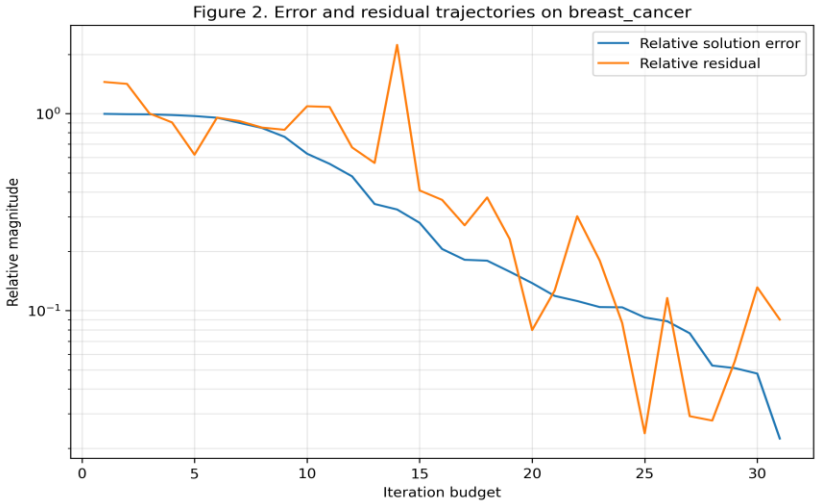

Figure 2. Error and residual trajectories on the breast_cancer operator.

Table 1 and Figures 1 and 2 substantiate the study's first objective. The Cayley-Hamilton-directed Krylov realization is algebraically reliable: when allowed to converge, it recovers the direct regularized operator to machine precision, and its convergence can be understood through the same spectral properties that motivate the theorem-directed perspective.

5.2. Predictive performance and computational efficiency

Table 2 reports the mean cross-validated accuracy and macro-F1 for the four methods, while Table 3 and Figure 3 extend the comparison to computational efficiency and visual synthesis. The reported results motivate the ensuing analyses. Full CG and direct RLS achieved identical mean predictive performance across all datasets. This finding follows directly from the preceding section: once the full Krylov recurrence converges, it realizes the same regularized operator as the dense solve. The truncated version was likewise highly competitive, delivering comparable performance with shorter training times; this suggests that low-order, theorem-guided polynomial models retain much of the useful predictive signal.

Table 2. Predictive performance across methods.

Dataset	Method	Accuracy	Macro-F1
breast_cancer	cg	0.954 ± 0.019	0.950 ± 0.022
breast_cancer	direct	0.954 ± 0.019	0.950 ± 0.022
breast_cancer	logreg	0.974 ± 0.014	0.972 ± 0.015
breast_cancer	tcg	0.950 ± 0.020	0.945 ± 0.022
digits	cg	0.935 ± 0.009	0.934 ± 0.009
digits	direct	0.935 ± 0.009	0.934 ± 0.009
digits	logreg	0.967 ± 0.008	0.967 ± 0.008
digits	tcg	0.932 ± 0.006	0.931 ± 0.006
iris	cg	0.823 ± 0.058	0.818 ± 0.059
iris	direct	0.823 ± 0.058	0.818 ± 0.059
iris	logreg	0.923 ± 0.073	0.923 ± 0.073
iris	tcg	0.807 ± 0.065	0.801 ± 0.065
wine	cg	0.986 ± 0.014	0.986 ± 0.014
wine	direct	0.986 ± 0.014	0.986 ± 0.014
wine	logreg	0.977 ± 0.021	0.978 ± 0.020
wine	tcg	0.977 ± 0.025	0.978 ± 0.024

For breast_cancer, Direct RLS and Full CG each attained a mean accuracy of 0.954 and a mean macro-F1 of 0.950; TCG remained close, at 0.950 and 0.945. On that dataset, logistic regression produced the most promising results, reaching 0.974 accuracy and 0.972 macro-F1. On digits, Direct RLS and Full CG again matched, with 0.935 accuracy and 0.934 macro-F1,

while TCG followed at 0.932 and 0.931. Logistic regression led with approximately 0.967 on both metrics. On iris, the theorem-guided methods were less competitive: Direct RLS and Full CG recorded 0.823 accuracy and 0.818 macro-F1, compared with 0.923 for logistic regression. By contrast, on wine, Full CG and Direct RLS achieved the highest mean accuracy and macro-F1, both at 0.986, slightly exceeding the logistic baseline.

These results delimit the contribution carefully. The theorem-guided architecture is not presented as a classifier that surpasses every baseline, and the evidence does not support such a claim. Instead, the findings indicate that the direct and full Krylov realizations are operator-equivalent and that truncated polynomial models may attain performance in the same range at substantially lower computational cost. Wine appears to provide the strongest predictive case. The breast_cancer and digits examples are primarily valuable because they yield an interpretable, flexible regularized operator whose exact and truncated forms can be compared on algebraic grounds. This distinction is essential when interpreting the results: the article proposes a structured operator alternative, not a wholesale replacement for existing classifiers.

Table 3. Computational efficiency summary.

Dataset	Method	Fit time (s)	Mean iterations
breast_cancer	cg	0.0097 ± 0.0252	61.6 ± 1.9
breast_cancer	direct	0.0086 ± 0.0241	-
breast_cancer	logreg	0.0285 ± 0.0396	-
breast_cancer	tcg	0.0006 ± 0.0001	8.7 ± 5.3
digits	cg	0.0051 ± 0.0007	60.7 ± 0.7
digits	direct	0.0014 ± 0.0001	-
digits	logreg	0.0790 ± 0.0350	-
digits	tcg	0.0018 ± 0.0003	8.2 ± 5.6
iris	cg	0.0005 ± 0.0001	5.0 ± 0.0
iris	direct	0.0004 ± 0.0000	-
iris	logreg	0.0114 ± 0.0264	-
iris	tcg	0.0004 ± 0.0000	2.1 ± 0.7
wine	cg	0.0007 ± 0.0001	15.7 ± 0.2
wine	direct	0.0088 ± 0.0253	-
wine	logreg	0.0023 ± 0.0002	-
wine	tcg	0.0005 ± 0.0001	2.6 ± 3.5

Table 3 shows that TCG is the fastest theorem-guided method. Averaged across all folds and datasets, its training time was approximately 0.0008 seconds, compared with 0.0040 seconds for Full CG, 0.0048 seconds for Direct RLS, and 0.0303 seconds for logistic regression. The contrast is most pronounced for the more demanding datasets. On breast_cancer, TCG required an average of 0.0006 seconds, versus 0.0097 for Full CG. On digits, TCG required 0.0018 seconds, whereas Full CG required 0.0051 and logistic regression 0.0790. These differences reflect the truncated solver's lower polynomial depth rather than merely incidental implementation effects.

The iteration statistics in Table 3 reinforce the polynomial interpretation. Full CG required approximately 61.6 iterations on breast_cancer and 60.7 on digits, whereas TCG used averages of 8.7 and 8.2 iterations after validation. The TCG depths for iris and wine were lower still. Thus, the truncated model is not simply a cost-reduction device; it is a validated, data-selected low-degree operator, and its lower cost can be interpreted as reduced polynomial complexity.

Figure 3 presents the accuracy comparison. The bars for Direct RLS and Full CG are indistinguishable across datasets, graphically corroborating the exactness results. The TCG bars are generally slightly lower, although they remain close, particularly for wine and breast cancer. Logistic regression performs best on three of the four datasets, which is consistent with the study's stated scope. The purpose is not to replace every linear classifier, but to show through benchmark comparisons that the theorem-guided architecture is competitive, exact when fully converged, and flexible when truncated.

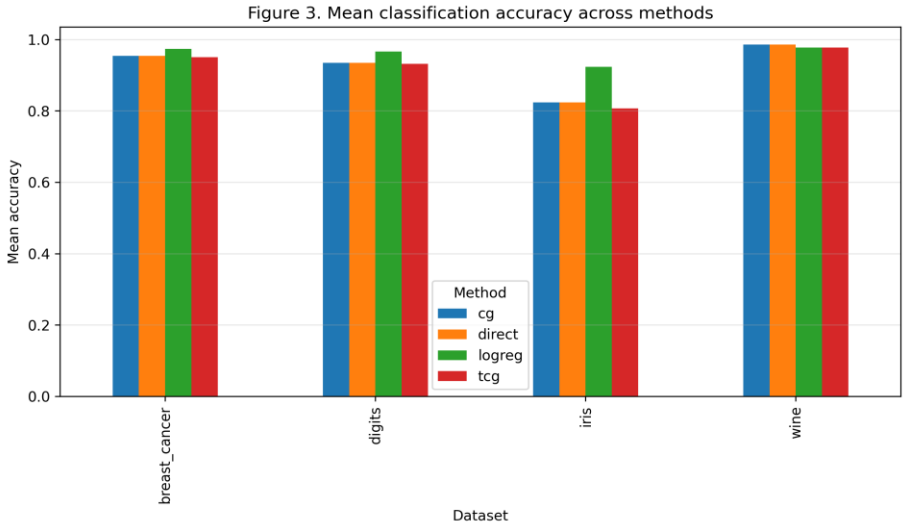

Figure 3. Mean classification accuracy across methods.

Section 5.2 therefore confirms the article's second objective. The complete theorem-based Krylov model is predictively identical to the direct regularized model, whereas the truncated version retains most of that predictive benefit at a very small fraction of the computational cost. This is precisely the behavior anticipated by the finite polynomial operator perspective.

5.3. Stability, robustness, and interpretability

The robustness experiment reported in Table 4 is particularly informative because it reveals behavior that clean-set accuracy alone cannot capture. For iris, all theorem-guided variants were invariant under the selected standardized perturbation. In the CV wine results, Direct RLS and Full CG each lost 0.019 accuracy, whereas TCG showed no decrease. On digits, Direct RLS and Full CG each declined by 0.011, while TCG declined by 0.015. The clearest case was breast_cancer: Full CG and Direct RLS each fell by 0.310, whereas TCG decreased by only 0.018.

Table 4. Robustness under additive Gaussian noise.

Dataset	Method	Clean accuracy	Noisy accuracy	Absolute drop
iris	direct	0.822	0.822	0.000
iris	cg	0.822	0.822	0.000
iris	tcg	0.844	0.844	0.000
wine	direct	0.981	0.963	0.019
wine	cg	0.981	0.963	0.019
wine	tcg	0.981	0.981	0.000
breast_cancer	direct	0.953	0.643	0.310
breast_cancer	cg	0.953	0.643	0.310
breast_cancer	tcg	0.947	0.930	0.018
digits	direct	0.941	0.930	0.011
digits	cg	0.941	0.930	0.011
digits	tcg	0.944	0.930	0.015

This finding accords with theoretical expectations. Because Full CG and Direct RLS realize the same regularized operator, their robustness profiles coincide. TCG instead represents a lower-degree polynomial operator. For breast cancer, this lower-degree restriction appears to function as a beneficial spectral smoother. Relative to the full global solution, clean accuracy declines only slightly, while noisy accuracy improves by a substantial percentage. The truncated theorem-guided model therefore preserves most clean performance while shedding considerable sensitivity to perturbation. This robustness gain should be understood as a consequence of controlled polynomial restriction, rather than as an independent heuristic introduced after training.

Figure 4 illustrates how TCG performance on breast_cancer varies with the iteration budget. Accuracy is approximately 0.971 and macro-F1 approximately 0.968 at iteration 4. After this peak, the curve settles near the full-solution level of about 0.953 as the budget increases. Early Krylov polynomials rapidly capture the most stable discriminative directions, whereas later polynomials add detail associated with less important spectral points that contribute less to prediction and robustness. A lower degree is therefore not merely cheaper; in this case, it is statistically preferable.

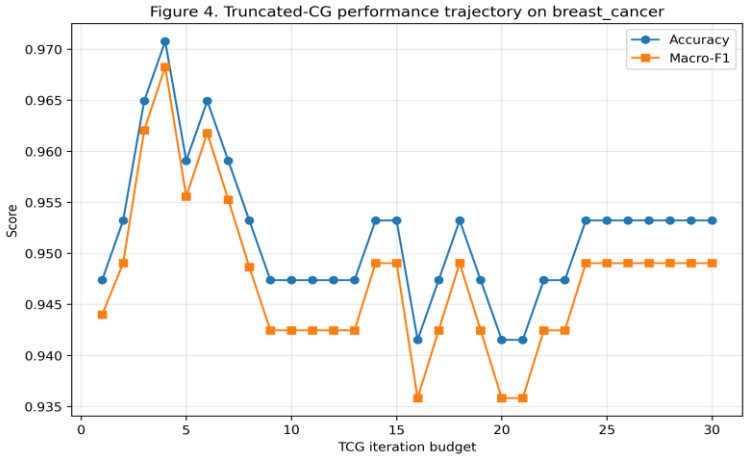

Figure 4. Truncated-CG performance trajectory on breast_cancer.

The interpretability analysis in Figure 5 provides further evidence. In the breast cancer model (direct theorem-guided), the largest mean absolute coefficients corresponded to mean radius, worst radius, mean perimeter, worst area, and mean area. Compactness and concavity descriptors followed these variables in importance. This ranking is coherent because it concentrates importance on a related group of geometric and morphological measures rather than distributing it uniformly across all features.

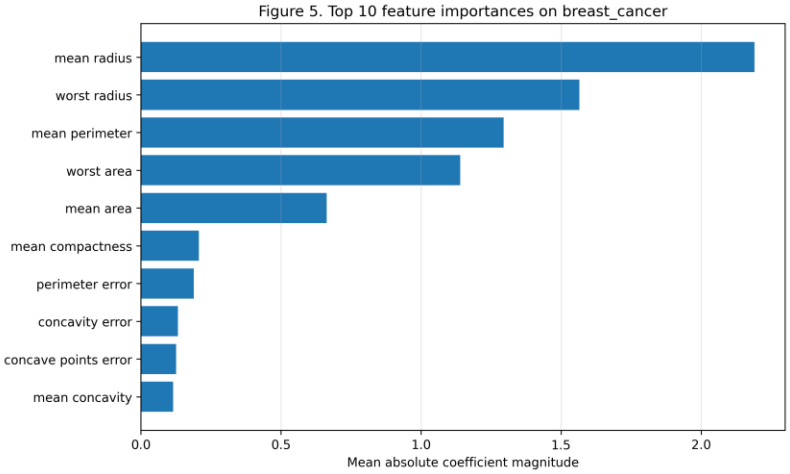

Figure 5. Top coefficient-based feature importances on breast_cancer.

This finding is important because it implies that the theorem-based model remains inspectable at the coefficient level. The learned transformation can be examined directly through the same operator perspective that supports the analyses of exactness and truncation. Compared with more opaque pipelines, this permits the analyst to ask not only whether the model is accurate, but also which feature directions the finite polynomial operator emphasizes.

Section 5.3 closes the empirical argument with a qualified yet important conclusion. Truncation is not solely a means of accelerating computation; it can also constrain operator complexity in ways that improve robustness, particularly in ill-conditioned problems. At the same time, the architecture retains an interpretable coefficient structure grounded in the theorem. Together, these properties allow the practical significance of the Cayley–Hamilton perspective to extend beyond algebraic exactness alone. For practical applications, TCG is most useful on breast_cancer and wine because it combines lower cost with clear robustness gains. It offers no tangible advantage on iris or digits, where predictive gains are absent and robustness is comparable to, or slightly weaker than, that of the full solution.

6. Discussion and conclusions

6.1. Discussion

Sections 5.1–5.3 provide evidence supporting the article's aim: to recast the Cayley-Hamilton theorem from a classical matrix identity as a constructive design principle for machine-learning operators. The central empirical question is whether this transformation produces a useful computational architecture rather than merely an elegant reformulation. The stated objectives have been met. Accordingly, the discussion treats the experiments as evidence for a design principle, rather than as a simple compilation of benchmark results.

The principal finding is the exact agreement between the full Krylov realization and the direct regularized solution. Rather than being assumed, this equivalence was demonstrated through relative solution errors at machine precision and matching predictive metrics across the datasets. The result confirms the central claim: when the solver converges, the inverse in regularized least-squares classification can be realized by a finite polynomial action without loss of fidelity. The theorem-based approach thereby advances from an algebraic possibility to an established computational fact.

A second major finding concerns truncation. The truncated method does not merely produce an inferior but faster version of the full solution; instead, it establishes a consequential trade-off among polynomial depth, running time, predictive quality, and robustness. Across the benchmark suite, TCG remained close to the full operator in terms of clean accuracy while requiring more economical training. Most importantly, the perturbation analysis on breast_cancer showed that a low-degree polynomial realization can suppress detrimental sensitivity retained by the exact solution. The iteration budget should

therefore be understood as a model-complexity control parameter with operator-theoretic significance, rather than treated solely as a computational setting.

The article also clarifies the architecture's limitations. Across the four datasets, logistic regression outperformed the theorem-guided methods in three cases. This result strengthens the thesis rather than undermining it. The contribution is not a universal guarantee of superior benchmark accuracy, but a mathematically principled and computationally interpretable alternative for regularized operator learning. A theorem-guided architecture is most powerful when exact operator equivalence, controllable polynomial truncation, and diagnostic transparency are jointly valued. Within such settings, moving coherently from exact to approximate solutions within a single operator family offers a substantial advantage. Recognizing this limitation prevents the theorem-guided proposal from being overstated and identifies more precisely where its merits reside.

These findings also have methodological implications for machine-learning research. Many algorithmic comparisons leave the underlying linear-algebraic structure implicit and focus only on task-level performance; the present study suggests that this perspective may be overly broad. When finite-dimensional operators govern a problem, theorem-level structure can offer explanations for a method's behavior, the changes introduced by truncation, and the origins of robustness gains. Algebraic structure should therefore be treated as substantive explanatory content of the model, not merely as an implementation detail. In this respect, the article advocates a closer integration of numerical linear algebra with empirical machine-learning evaluation.

The computational architecture also served as a useful object of study. By separating operator construction from solver realization, evaluation, and artifact export, the implementation aligns code outputs with the manuscript's analytical claims. This constitutes a practical contribution: when the architecture reflects the structure of the argument, reproducibility improves, interpretation becomes clearer, and the numerical evidence is easier to audit.

6.2. Conclusions

This article sought to demonstrate that the Cayley–Hamilton theorem can serve as a constructive foundation for machine learning rather than remain in the background. It did so by developing a theorem-guided formulation of regularized least-squares classification, implementing that formulation within a modular Krylov architecture, and validating it through evidence concerning exactness, prediction, efficiency, robustness, and interpretability. The strongest support arises when all three dimensions—algebraic identity, numerical realization, and empirical behavior—converge on the same conclusion.

The conclusion is direct. First, the theorem-guided full conjugate-gradient realization is algebraically exact in practical computation, reproducing the direct regularized solution to machine precision. Second, the truncated Krylov realization yields a low-degree polynomial model that combines low computational cost with competitive predictive performance. Third, under ill-conditioned conditions, truncation can provide a stability benefit absent from the exact solution, as demonstrated by breast_cancer robustness. Fourth, the operator perspective reinforces the conclusion that the resulting model remains interpretable through its coefficients.

In conclusion, the Cayley-Hamilton theorem can be viewed as a useful framework for organizing machine-learning design simultaneously at three levels: theory, by characterizing the finite-dimensional closure of operators; computation, by motivating the construction of Krylov polynomials; and empirical analysis, by integrating various established criteria of empirical design. The article therefore answers its central question affirmatively: the theorem is a viable and useful guiding principle for structured machine-learning operators.

6.3. Limitations of the study

Validation based on only four small benchmark datasets constitutes a limitation of the present study. Although these datasets are sufficiently specific to examine the numerical behavior, predictive performance, and computational efficiency of the proposed Cayley-Hamilton-guided framework, they cannot adequately represent the scale, complexity, heterogeneity, and noise levels of real machine learning problems. The study consequently shows that Krylov-based matrix-polynomial learning achieves accurate and stable classification while reducing computational cost. Nevertheless, these findings require validation on larger and more diverse datasets drawn from several application domains. Future work therefore needs to test the framework in high-dimensional settings, on large data sets, and under more demanding real-world traffic situations to assess scalability, robustness, and general applicability more fully. The results derive solely from four small benchmark datasets. The study must therefore be regarded as proof-of-concept evidence until the framework has been validated on larger, heterogeneous datasets involving real-world datasets.

7. Future work

7.1. Research extensions and translational directions

These findings suggest several directions for future research. The theorem-guided architecture should be evaluated on larger and sparser datasets, for which matrix-vector products are even more attractive than dense factorization. A second direction is pre-conditioning. Because conditioning exerted a strong influence on iteration depth in the accuracy tests, theorem-guided

preconditioners could yield a faster and more stable Krylov realization while preserving the polynomial interpretation. They would also clarify the circumstances in which matrix-free computation becomes the principal advantage of the Krylov realization.

Kernelization represents the third direction. A dual architecture would permit the same finite-operator perspective to operate on kernel Gram matrices, thereby providing a route to nonlinear decision boundaries while remaining within the theorem-guided framework. A final direction concerns adaptive stopping and hybrid regularization. Polynomial degree could be selected through spectral diagnostics, generalized cross-validation, or stability criteria rather than exclusively through external validation. The purpose of these extensions is to test whether the same finite-polynomial reasoning is useful when the representation space is implicit or data-adaptive.

Ultimately, domain-specific applications in inverse problems, scientific computing, and operator-learning settings with strong structural priors would provide an appropriate testbed. In such contexts, the theorem-driven architecture's interpretability and operator transparency may become practically decisive.

8 Data availability and code

The complete code can be accessed through the following repository URL:

https://github.com/JAIME6609/CAYLEY-HAMILTON/blob/main/CODE-ART-CAYLEY-HAMILTON.py

References

Björck, Å. (1996). *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611971484

Engl, H. W., Hanke, M., & Neubauer, A. (1996). *Regularization of inverse problems*. Kluwer Academic Publishers. https://doi.org/10.1007/978-94-009-1740-8

Gazzola, S., & Sabaté Landman, M. (2020). Krylov methods for inverse problems: Surveying classical, and introducing new, algorithmic approaches. *GAMM-Mitteilungen, 43*(4), e202000017. https://doi.org/10.1002/gamm.202000017

Gazzola, S., Novati, P., & Russo, M. R. (2015). On Krylov projection methods and Tikhonov regularization. *Electronic Transactions on Numerical Analysis, 44*, 83–123. ETNA full text

Gohberg, I., Lancaster, P., & Rodman, L. (2009). *Matrix polynomials*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898719024

Golub, G. H., Hansen, P. C., & O'Leary, D. P. (1999). Tikhonov regularization and total least squares. *SIAM Journal on Matrix Analysis and Applications, 21*(1), 185–194. https://doi.org/10.1137/S0895479897326432

Greenbaum, A. (1997). *Iterative methods for solving linear systems*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611970937

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer. https://doi.org/10.1007/978-0-387-84858-7

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

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics, 12*(1), 55–67. https://doi.org/10.1080/00401706.1970.10488634

Horn, R. A., & Johnson, C. R. (2012). *Matrix analysis* (2nd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9781139020411

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830. JMLR article

Qin, S. J., Liu, Y., & Tang, S. (2023). Partial least squares, steepest descent, and conjugate gradient for regularized predictive modeling. *AIChE Journal, 69*(4), e17992. https://doi.org/10.1002/aic.17992

Rifkin, R. M., & Klautau, A. (2004). In defense of one-vs-all classification. *Journal of Machine Learning Research, 5*, 101–141. JMLR article

Rifkin, R. M., Yeo, G. W., & Poggio, T. (2003). Regularized least-squares classification. In J. A. K. Suykens, G. Horvath, S. Basu, C. Micchelli, & J. Vandewalle (Eds.), *Advances in learning theory: Methods, models and applications* (pp. 131–154). IOS Press.

Saad, Y. (2003). *Iterative methods for sparse linear systems* (2nd ed.). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898718003

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Suykens, J. A. K., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters, 9*(3), 293–300. https://doi.org/10.1023/A:1018628609742

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., . . . van Mulbregt, P. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods, 17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2