Fast and scalable computation of reduced-order nonlinear solutions with application to evolutinal neural networks

William Anderson and Mohammad Farazmand*

Department of Mathematics, North Carolina State University, 2311 Stinson Drive, Raleigh, NC 27695-8205, USA

Abstract

We develop a fast and scalable method for computing Reduced-order Nonlinear Solutions (RONS). RONS is a recently proposed framework for reduced-order modeling of time-dependent partial differential equations (PDEs), where the reduced model depends nonlinearly on a set of time-varying parameters. RONS obtains an explicit set of ordinary differential equations (ODEs) for the parameters, which optimally evolve the shape of the approximate solution. However, a naive construction of these ODEs requires the evaluation of $\mathcal{O}(n^2)$ integrals, where $n$ is the number of model parameters. For high-dimensional models, the resulting computational cost becomes prohibitive. Here, exploiting the structure of the RONS equations and using symbolic computing, we develop an efficient computational method which requires only $\mathcal{O}(K^2)$ integral evaluations, where $K \ll n$ is an integer independent of $n$. Our method drastically reduces the computational cost and allows for the development of highly accurate spectral methods where the modes evolve to adapt to the solution of the PDE, in contrast to existing spectral methods where the modes are static in time.

1 Introduction

Reduced-order models are routinely used to study the behavior of partial differential equations (PDEs) [3, 7]. Obtaining numerical solutions of PDEs is often computationally expensive, and therefore reduced-order models provide a cheaper alternative to obtain approximate solutions. Anderson and Farazmand [1] recently proposed reduced-order nonlinear solutions (RONS) as a framework for deriving reduced-order models for time-dependent PDEs.

RONS considers shape-morphing approximate solutions $\hat{u}(x, q(t))$ to the PDE which depend nonlinearly on time-varying parameters $q(t) \in \mathbb{R}^n$. Therefore, it significantly expands the scope of reduced models which often comprise a linear combination of time-independent |

*Corresponding author’s email address: farazmand@ncsu.edu
modes [3]. RONS produces a set of ordinary differential equations (ODEs) for the evolution of the parameters $q(t)$ by minimizing the instantaneous error between dynamics of the reduced-order solution $\hat{u}(x, q(t))$ and the true dynamics of the PDE. Furthermore, RONS ensures that the reduced-order solution preserves conserved quantities of the governing PDE.

The main computational cost of RONS comes from evaluating the integrals, or inner products, which are required to form the reduced-order equations (see Section 2 below). More specifically, in order to form the RONS reduced-order equations, $O(n^2)$ inner product computations are required, where $n$ is the number of parameters. Making matters worse, the inner products need to be recomputed at each time step. To compute these inner products, one can use quadrature, Monte Carlo integration, or symbolic computing. As the number of parameters $n$ grows, all these methods become quickly prohibitive.

In this paper, we exploit the hidden structure of RONS equations to reduce the number of required inner product computations to $O(K^2)$ where $K \ll n$ is an integer independent of $n$. Furthermore, since our method uses symbolic computation, the inner products do not need to be recomputed during time stepping. As a result, the computational cost remains low even when the number of parameters $n$ is very large. This scalability allows us to go beyond reduced-order modeling and use RONS as a spectral method where the modes (or basis functions) evolve over time through their nonlinear dependence on time-dependent parameters.

1.1 Related work

The closest work to RONS is the evolutional deep neural networks (EDNNs) developed by Du and Zaki [5]. EDNNs approximate solutions of PDEs by evolving weights and biases of a neural network over time. Since the activation functions are nonlinear, an EDNN depends nonlinearly on its parameters, i.e., weights and biases. As such, EDNNs are a special case of reduced-order nonlinear solutions. Therefore, it is not surprising that the EDNN equations are identical to RONS. We note that Du and Zaki [5] developed EDNN simultaneously and independently of RONS [1].

Although the reduced equations for RONS and EDNNs are identical, there are some notable differences between the two methods. Namely, RONS ensures that the reduced-order model respects the conserved quantities of the PDE. EDNN does not guarantee these conservation laws, although they can be easily enforced following the methodology introduced in Ref. [1]. On the other hand, Du and Zaki [5] show how various boundary conditions of the PDE can be embedded into the EDNN framework, an important contribution which was not considered in the development of RONS.

EDNN employs Monte Carlo integration with uniform sampling to approximate the required inner products. Bruna et al. [4] proposed an adaptive sampling method to estimate the integrals. These adaptive samples are drawn from a distribution which depends on the reduced-order solution at any given time. They show that for PDEs whose solutions are localized in space, adaptive sampling results in more accurate solutions than uniform sampling.
1.2 Outline

This paper is organized as follows. In section 2 we briefly review RONS. Section 3 contains our main results where we develop a fast and scalable method for constructing the RONS reduced-order equations. Our concluding remarks are presented in Section 4.

2 Set-up and preliminaries

In this section, we present a brief review of RONS and refer to Ref. [1] for a more detailed explanation. RONS builds reduced-order models for PDEs of the general form

\[ \frac{\partial u}{\partial t} = F(u), \quad u(x,0) = u_0(x), \]  

where \( u : D \times \mathbb{R}^+ \to \mathbb{R} \) is the solution of the PDE, \( D \subseteq \mathbb{R}^d \) is the spatial domain, \( F \) is a nonlinear differential operator, and \( u_0 \) is the initial condition. We assume the solution \( u(\cdot,t) \) belongs to some Hilbert space \( H \) with the inner product \( \langle \cdot, \cdot \rangle_H \) and the induced norm \( \| \cdot \|_H \).

For a given shape-morphing approximate solution \( \hat{u}(x,q(t)) \), RONS returns a set of ODEs to evolve the parameters \( q(t) \) such that the instantaneous error between dynamics of the reduced-order solution and dynamics of the true PDE is minimized. The instantaneous error is defined by

\[ J(q, \dot{q}) = \frac{1}{2} \| \hat{u}_t - F(\hat{u}) \|_H^2, \]  

which measures the difference between the rate of change of the ansatz \( \hat{u}_t \) and the rate of change \( F(\hat{u}) \) dictated by the PDE. If the PDE has no conserved quantity, the reduced-order equations are obtained by minimizing (2). However, let’s consider the more general case where the PDE has \( m \) conserved quantities, \( I_k : H \to \mathbb{R} \) with \( k \in \{1, 2, \ldots, m\} \). If these quantities are conserved they must satisfy \( I_k(u(\cdot,t)) = I_k(u_0) \) for all \( t \geq 0 \). Preservation of these conserved quantities by the reduced-order model is crucial to ensure that the model does not exhibit unphysical properties [6].

To obtain an evolution equation for the parameters \( q(t) \), we solve the constraint optimization problem

\[ \min_{q \in \mathbb{R}^n} J(q, \dot{q}), \]  

subject to \( I_k(q(t)) = I_{k,0}, \quad k = 1, 2, \ldots, m, \quad \forall t \geq 0, \) \( (3) \)

where \( I_k(q(t)) \) is shorthand for \( I_k(\hat{u}(\cdot,q(t))) \). As shown in [1], the solution to this minimization problem is

\[ M(q)\dot{q} = f(q) - \sum_{k=1}^{m} \lambda_k \nabla I_k(q), \]  

which we call the RONS equation. Here \( M(q) \in \mathbb{R}^{n \times n} \) is the symmetric positive definite metric tensor defined by

\[ M_{ij} = \left\langle \frac{\partial \hat{u}}{\partial q_i}, \frac{\partial \hat{u}}{\partial q_j} \right\rangle_H, \quad i, j \in \{1, 2, \ldots, n\}. \]  

(5)
The entries of the right-hand side vector field $f : \mathbb{R}^n \to \mathbb{R}^n$ are given by

$$f_i = \left\langle \frac{\partial \hat{u}}{\partial q_i}, F(\hat{u}) \right\rangle_H, \quad i = 1, 2, \ldots, n.$$ (6)

The Lagrange multipliers $\lambda = (\lambda_1, \ldots, \lambda_m)^\top$ are determined the linear system

$$C(q) \lambda = b(q),$$ (7)

where $C(q) \in \mathbb{R}^{m\times m}$ is the symmetric positive definite constraint matrix defined by

$$C_{ij} = \langle \nabla I_j, M^{-1} \nabla I_i \rangle, \quad i, j \in \{1, 2, \ldots, m\},$$ (8)

and the vector $b = (b_1, b_2, \ldots, b_m)^\top \in \mathbb{R}^m$ is given by

$$b_i = \langle \nabla I_i, M^{-1} f \rangle, \quad i = 1, 2, \ldots, m.$$ (9)

Here the inner product $\langle \cdot, \cdot \rangle$ denotes the standard Euclidean inner product.

If no conserved quantities are enforced, then we must solve the optimization problem (3) without any constraints. The unique minimizer of the unconstrained problem is given by removing the sum from Eq. (4), i.e.,

$$M(q) \dot{q} = f(q).$$ (10)

A geometric depiction of RONS in the unconstrained case is shown in Figure 1. We view the shape-morphing solution $\hat{u}(\cdot, q)$ as a map from the parameters $q$ to the Hilbert space $H$ where solutions of the PDE lie. The approximate solution $\hat{u}$ then maps the space of all viable parameter values $\Omega \subseteq \mathbb{R}^n$ to an $n$-dimensional manifold $\mathcal{M} \subset H$. We consider any arbitrary but smooth evolution of parameters $q(t)$ that lies in $\Omega$, which will be mapped to a curve that lies on the manifold $\mathcal{M}$. The tangent vector $\dot{q}(t)$ to the parameter evolution is mapped to a vector $\hat{u}_t$ which lies on the tangent space of the manifold $T_{\hat{u}} \mathcal{M}$. In general, the manifold is not invariant under the dynamics of the governing PDE (1), and so $F(\hat{u})$ will not lie in tangent space. By minimizing (2) with respect to $\dot{q}(t)$, we find the orthogonal projection of $F(\hat{u})$ onto $T_{\hat{u}} \mathcal{M}$. In other words, we evolve the approximate solution $\hat{u}$ so that it most closely resembles the expected PDE dynamics.

If we enforce no conserved quantities and consider an approximate solution which depends linearly on the parameters

$$\hat{u}(x, q(t)) = \sum_{i=1}^r q_i(t) u_i(x),$$ (11)

where the modes $\{u_i(x)\}_{i=1}^r$ are orthonormal, the RONS equations coincide with standard Galerkin projection. In this special case, the manifold $\mathcal{M}$ is a linear subspace of the Hilbert space $H$. Galerkin projection finds the orthogonal projection of $F(\hat{u})$ onto the manifold and RONS finds the orthogonal projection onto the tangent space of the manifold, so for the case when $\mathcal{M}$ is a linear subspace the two methods coincide. We refer to [1] for more details.
Figure 1: Geometric illustration of RONS. The shape-morphing approximate solution $\hat{u}$ maps the parameter space $\Omega$ to the manifold $\mathcal{M}$. An evolution of the parameters $q(t)$ is mapped to a curve on $\mathcal{M}$, and the tangent vector to the parameters $\dot{q}(t)$ is mapped to a vector $\hat{u}_t$ on the tangent space of the manifold.

3 A fast and scalable computational method

3.1 Computational bottlenecks

For simplicity, we consider the unconstrained RONS equation (10). The method is similar for the constrained case (4) where conserved quantities are enforced. The computational cost of solving equation (10) comes from two sources:

1. Forming the metric tensor $M(q)$ and the right-hand side vector $f(q)$.

2. Inverting the metric tensor. By inverting the metric tensor we refer to any numerical method for solving the linear equation $M(q)\dot{q} = f(q)$ for $\dot{q}$.

There exist several fast linear numerical algebra methods for solving large linear equations which can be used for inverting the metric tensor. Therefore, we focus on item 1 above which constitutes the main computational bottleneck.

To form the metric tensor (5), we need to compute $n^2$ inner products. Since this matrix is symmetric, the number of independent inner products is in fact $n(n + 1)/2$. Additionally, to form the right-hand side vector (6), we need to compute $n$ inner products. Therefore, a total of $n(n + 3)/2$ integrals need to be computed. As the number of parameters $n$ grows, this becomes computationally prohibitive. Making matters worse, during time stepping, $q(t)$ changes and these quantities need to be recomputed at each time step.

3.2 Efficient construction of the RONS equation

In this section, we present a method for efficient construction of the metric tensor $M$ and right-hand side vector $f$. We show that only a small number of symbolic computations are required to build the RONS equation, provided that the shape-morphing approximate solution has a specific form and analytical expressions for the inner products in $M$ and $f$ can be obtained.
Specifically, we consider shape-morphing approximations of the form,
\[
\hat{u}(\mathbf{x}, \mathbf{q}(t)) = \sum_{i=1}^{r} \alpha_i(t) \phi(\mathbf{x}, \mathbf{\beta}_i(t)),
\]
where \( \phi : \mathbb{R}^d \rightarrow \mathbb{R} \) is a \( C^1 \) function. We refer to \( \phi(\mathbf{x}, \mathbf{\beta}_i(t)) \) as the \( i \)-th shape-morphing mode. The vector of shape parameters \( \mathbf{\beta}_i \in \mathbb{R}^{K-1} \) controls the shape of the \( i \)-th mode; it contains parameters such as scaling and center of the mode. The scalar \( \alpha_i(t) \) denotes the mode amplitude. Therefore, parameters of the shape-morphing solution \( \hat{u} \) are given by \( \mathbf{q} = (\alpha_1, \beta_1^\top, ..., \alpha_r, \beta_r^\top) \in \mathbb{R}^{rK} \), where \( n = rK \). Note that the number of shape parameters \( K - 1 \) is independent of the number of terms \( r \) in the sum. This independence plays an importance role in the proposed computational method.

One particular approximate solution with the form (12) is to assume the modes are Gaussian, i.e.,
\[
\hat{u}(x, \mathbf{q}(t)) = \sum_{i=1}^{r} A_i(t) \exp \left[ -\frac{(x - x_i(t))^2}{L_i(t)^2} \right],
\]
where \( A_i(t) = A_i(t) \) and \( \mathbf{\beta}_i(t) = (L_i(t), x_i(t))^\top \). Here \( A_i \) controls the amplitude of the \( i \)-th Gaussian, \( L_i \) is a length scale that determines the Gaussian’s width, and \( x_i \) determines the Gaussian’s center. We use this one-dimensional Gaussian mixture as an illustrative example throughout this section.

There are many other possible choices of modes \( \phi \) for the approximate solution. For example, we could take the modes to be activation functions typically used in neural networks, such as the rectified linear unit (ReLU) function or hyperbolic tangent. In this case, Eq. (12) represents a shallow neural network and the shape parameters \( \mathbf{\beta}_i \) are the weights and biases of the \( i \)-th node. Another choice could be wavelet functions, where the shape parameters are dilations and translations.

We now exploit the structure of the shape-morphing approximation (12) to efficiently calculate the inner products in \( M \) and \( f \) using symbolic computation. We first discuss how to efficiently build the metric tensor \( M \). Consider arbitrary indices \( i \) and \( j \) with \( i, j \in \{1, ..., r\} \). Using these general indices and the approximate solution (12), all entries of \( M \) will have the form of one of the following inner products,
\[
\mathcal{I}_{\alpha_i \alpha_j} := \left\langle \frac{\partial \hat{u}}{\partial \alpha_i}, \frac{\partial \hat{u}}{\partial \alpha_j} \right\rangle_H,
\]
\[
\mathcal{I}_{\alpha_i \beta_{jk}} := \left\langle \frac{\partial \hat{u}}{\partial \alpha_i}, \frac{\partial \hat{u}}{\partial \beta_{jk}} \right\rangle_H, \quad k \in \{1, ..., K - 1\}
\]
\[
\mathcal{I}_{\beta_{ik} \beta_{jl}} := \left\langle \frac{\partial \hat{u}}{\partial \beta_{ik}}, \frac{\partial \hat{u}}{\partial \beta_{jl}} \right\rangle_H, \quad k, l \in \{1, ..., K - 1\},
\]
where \( \beta_{ik} \) denotes the \( k \)-th component of \( \mathbf{\beta}_i = (\beta_{i1}, \beta_{i2}, ..., \beta_{i(K-1)})^\top \). The advantage of using symbolic computation for the expressions in equation (14) is that after obtaining closed-form expressions for the inner products, we can build the entire metric tensor through substitution of the appropriate indices \( i \) and \( j \). For example, rather than computing \( \mathcal{I}_{\alpha_1 \alpha_2} \), we simply need to substitute the values of \( \alpha_1 \) and \( \alpha_2 \) into the already obtained symbolic expression for \( \mathcal{I}_{\alpha_1 \alpha_2} \).
This same principle holds regardless of which indices we choose as $i$ and $j$. Thus, after obtaining closed-form expressions the inner products in equation (14), we know what all the entries of the metric tensor $M$ will look like regardless of how many modes we choose to use in the approximate solution. Additionally, we only need to perform the symbolic computations at the initial time and can then substitute the updated parameter values as we march the approximate solution forward in time. Note that, by symmetry of the inner product, we only need to calculate $K(K - 1)/2$ inner products for equation (14c). Therefore, there are in total $K(K + 1)/2$ terms to be calculated in equation (14). We emphasize that this number is independent of the number of modes $r$ in the shape-morphing approximation (12); it only depends on the number of shape parameters $K$.

Perhaps this concept can be better explained by examining the structure of the metric tensor. The matrix $M$ is composed of blocks $M^{(i,j)} \in \mathbb{R}^{K \times K}$ such that

$$M = \begin{bmatrix}
M^{(1,1)} & M^{(1,2)} & \cdots & M^{(1,r)} \\
M^{(2,1)} & M^{(2,2)} & \cdots & M^{(2,r)} \\
\vdots & \vdots & \ddots & \vdots \\
M^{(r,1)} & M^{(r,2)} & \cdots & M^{(r,r)}
\end{bmatrix},$$

(15)

where $M^{(i,j)} = M^{(j,i)}$ since $M$ is symmetric. Each block can be expressed in terms of the inner products (14),

$$M^{(i,j)} = \begin{bmatrix}
I_{\alpha_i,\beta_j} & I_{\alpha_i,\beta_{j+1}} & \cdots & I_{\alpha_i,\beta_{(K-1)}} \\
I_{\beta_{j+1},\alpha_i} & I_{\beta_{j+1},\beta_j} & \cdots & I_{\beta_{j+1},\beta_{(K-1)}} \\
\vdots & \vdots & \ddots & \vdots \\
I_{\beta_{(K-1)}},\alpha_i & I_{\beta_{(K-1)},\beta_{j+1}} & \cdots & I_{\beta_{(K-1)},\beta_{(K-1)}}
\end{bmatrix}.$$  

(16)

With this labeling, the block $M^{(i,j)}$ represents all of the inner products involving derivatives of the approximate solution with respect to the shape parameters $\beta_{jk}$ and $\beta_{jk}$ of the $i$th and $j$th modes and their respective amplitudes, $\alpha_i$ and $\alpha_j$. Although there are $K^2$ entries in $M^{(i,j)}$, we only need to perform symbolic calculations for the $K(K + 1)/2$ entries in the lower triangular part of the block. The other entries of the matrix, colored in blue, are then determined through substitution. This is due to the fact that if we have a closed-form expression for an entry $I_{q_i,q_j}$ in the lower triangular part of $M^{(i,j)}$, then we obtain the corresponding entry $I_{q_i,q_j}$ in the upper triangular part of the block by simply switching the values of $q_i$ and $q_j$ in the symbolic expression.

As an example, consider the Gaussian mixture (13). We must symbolically compute six inner products to form the block,

$$M^{(i,j)} = \begin{bmatrix}
\langle \frac{\partial u}{\partial A_i}, \frac{\partial u}{\partial A_j} \rangle_H & \langle \frac{\partial u}{\partial A_i}, \frac{\partial u}{\partial L_j} \rangle_H & \langle \frac{\partial u}{\partial A_i}, \frac{\partial u}{\partial x_j} \rangle_H \\
\langle \frac{\partial u}{\partial L_i}, \frac{\partial u}{\partial A_j} \rangle_H & \langle \frac{\partial u}{\partial L_i}, \frac{\partial u}{\partial L_j} \rangle_H & \langle \frac{\partial u}{\partial L_i}, \frac{\partial u}{\partial x_j} \rangle_H \\
\langle \frac{\partial u}{\partial x_i}, \frac{\partial u}{\partial A_j} \rangle_H & \langle \frac{\partial u}{\partial x_i}, \frac{\partial u}{\partial L_j} \rangle_H & \langle \frac{\partial u}{\partial x_i}, \frac{\partial u}{\partial x_j} \rangle_H
\end{bmatrix}.$$ 

(17)
Note that the terms in blue can be evaluated using the lower triangular part of the matrix. For instance, \( \langle \partial A_i \hat{u}, \partial L_j \hat{u} \rangle \) is evaluated using the symbolic expression for \( \langle \partial L_i \hat{u}, \partial A_j \hat{u} \rangle \) by substituting the values of \( A_i \) and \( L_j \) instead of \( A_j \) and \( L_i \), respectively. Therefore, only 6 symbolic computations are required to form the matrix block (17) and consequently the entire metric tensor \( M \). In comparison, computing the metric tensor by a brute force method, such as quadrature or Monte Carlo methods, would require evaluating \( n(n+1)/2 = 3r(3r+1)/2 \) integrals, which becomes prohibitive as the number of terms \( r \) increases.

The idea for building \( \mathbf{f} \) is similar to that of the metric tensor. We again consider a general index \( i \in \{1, \ldots, r\} \) and note all entries of \( \mathbf{f} \) will have the form of one of the following inner products:

\[
\begin{align*}
&\left\langle \frac{\partial \hat{u}}{\partial \alpha_i}, F(\hat{u}) \right\rangle_H, & &\left\langle \frac{\partial \hat{u}}{\partial \beta_k}, F(\hat{u}) \right\rangle_H, & &k = 1, \ldots, K - 1.
\end{align*}
\]

(18)

After using symbolic computation to obtain closed-form expressions for the \( K \) inner products in equation (18), we can then build \( \mathbf{f} \) through substitution rather than individually calculating each of the \( n = rK \) inner products in \( \mathbf{f} \).

The vector field \( \mathbf{f} \) also has a block structure. We may consider \( \mathbf{f} \) as \( r \) vectors \( \mathbf{f}^{(i)} \in \mathbb{R}^K \) stacked on top of each other so that

\[
\mathbf{f} = \begin{bmatrix}
\mathbf{f}^{(1)} \\
\vdots \\
\mathbf{f}^{(r)}
\end{bmatrix},
\]

(19)

where each vector \( \mathbf{f}^{(i)} \) is defined by

\[
\mathbf{f}^{(i)} = \left[ \left\langle \frac{\partial \hat{u}}{\partial \alpha_i}, F(\hat{u}) \right\rangle_H, \left\langle \frac{\partial \hat{u}}{\partial \beta_{i1}}, F(\hat{u}) \right\rangle_H, \ldots, \left\langle \frac{\partial \hat{u}}{\partial \beta_{i(K-1)}}, F(\hat{u}) \right\rangle_H \right]^\top.
\]

(20)

Again using the Gaussian mixture as an example, we have

\[
\mathbf{f}^{(i)} = \left[ \left\langle \frac{\partial \hat{u}}{\partial A_i}, F(\hat{u}) \right\rangle_H, \left\langle \frac{\partial \hat{u}}{\partial L_i}, F(\hat{u}) \right\rangle_H, \left\langle \frac{\partial \hat{u}}{\partial x_i}, F(\hat{u}) \right\rangle_H \right]^\top.
\]

(21)

Using a general index \( i \) we need symbolic expressions for only \( n = 3 \) integrals to build the vector \( \mathbf{f} \) through substitution rather than naively calculating \( 3r \) integrals.

In summary, building the metric tensor \( M \) requires \( K(K+1)/2 \) symbolic integrations and building the right-hand side vector \( \mathbf{f} \) requires \( K \) symbolic computations, resulting in only \( K(K+3)/2 \) symbolic computations to evaluate \( n(n+1) \) terms appearing in the RONS equation (10). The above discussion leads to the following theorem.

**Theorem 1.** For a shape-morphing solution of the form (12), forming the metric tensor \( M \) and the right-hand side vector \( \mathbf{f} \) in the RONS equation (4) requires symbolic calculation of \( K(K+3)/2 \) inner products, regardless of the number of modes \( r \) used in the approximate solution.
4 Conclusions

We developed a computationally efficient and scalable method to approximate the solution of a PDE with the linear superposition of $r$ shape-morphing modes as in Eq. (12). Each mode $\phi(\cdot, \beta_i(t))$ depends nonlinearly on time-dependent shape parameters $\beta_i \in \mathbb{R}^{K-1}$. RONS [1] is used to optimally evolve the amplitude of each mode and its shape parameters.

Brute force construction of the RONS equations requires the evaluation of $O(n^2)$ integrals, where $n = rK$. This approach becomes computationally prohibitive when a large number of modes $r$ are required to accurately approximate the solution. Making matters worse, the integrals need to be recomputed during time stepping as the shape parameters and amplitudes change.

Exploiting the structure of the RONS equations and using symbolic computing, we developed a method that drastically reduces this computational cost to $O(K^2)$ where $K \ll n$. The number of shape parameters $K$ is independent of the number of terms $r$ in the approximation. As a result, arbitrary large number of modes can be used at a negligible additional computational cost. Furthermore, since we use symbolic computation, the integrals do not need to be recomputed during time stepping; they are evaluated by direct substitution of the updated shape parameters and amplitudes.

Our method allows for development of highly accurate spectral methods where the modes evolve to adapt to the solution of the PDE, in contrast to existing spectral methods where the modes are static in time. In particular, this allows fast approximation of PDEs with shallow evolutionary neural networks as first proposed in [5]. In this case, the shape-morphing modes $\phi$ are the activation function of the network and the shape parameters are the weights and biases of each node.

References

[1] W. Anderson and M. Farazmand. Evolution of nonlinear reduced-order solutions for PDEs with conserved quantities. SIAM J. on Scientific Computing, 44:A176–A197, 2022.

[2] W. Anderson and M. Farazmand. Shape-morphing reduced-order models for nonlinear schrödinger equations. Nonlinear Dyn, 108:2889–2902, 2022.

[3] P. Benner, S. Gugercin, and K. Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. SIAM Review, 57(4):483–531, 2015.

[4] J. Bruna, B. Peherstorfer, and E. Vanden-Eijnden. Neural galerkin scheme with active learning for high-dimensional evolution equations, 2022.

[5] Y. Du and T.A. Zaki. Evolutional deep neural network. Phys. Rev. E, 104:045303, Oct 2021.

[6] A. J. Majda and Y. Yuan. Fundamental limitations of ad hoc linear and quadratic multi-level regression models for physical systems. Discrete & Continuous Dynamical Systems - B, 17(1531-3492_2012_4_1333):1333, 2012.
[7] C. W. Rowley and S. T. M. Dawson. Model reduction for flow analysis and control. *Annual Review of Fluid Mechanics*, 49(1):387–417, 2017.