

Optimal parameter selection in Weeks' method for numerical Laplace transform inversion based on machine learning

Journal of Algorithms &
Computational Technology
Volume 15: 1–22
© The Author(s) 2021
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/1748302621999621
journals.sagepub.com/home/act



Patrick O Kano¹ , Moysey Brio² and Jacob Bailey^{1,2}

Abstract

The Weeks method for the numerical inversion of the Laplace transform utilizes a Möbius transformation which is parameterized by two real quantities, σ and b . Proper selection of these parameters depends highly on the Laplace space function $F(s)$ and is generally a nontrivial task. In this paper, a convolutional neural network is trained to determine optimal values for these parameters for the specific case of the matrix exponential. The matrix exponential e^A is estimated by numerically inverting the corresponding resolvent matrix $(sI - A)^{-1}$ via the Weeks method at (σ, b) pairs provided by the network. For illustration, classes of square real matrices of size three to six are studied. For these small matrices, the Cayley-Hamilton theorem and rational approximations can be utilized to obtain values to compare with the results from the network derived estimates. The network learned by minimizing the error of the matrix exponentials from the Weeks method over a large data set spanning (σ, b) pairs. Network training using the Jacobi identity as a metric was found to yield a self-contained approach that does not require a truth matrix exponential for comparison.

Keywords

Numerical Laplace transform inversion, Weeks' method, machine learning, matrix exponential

Received 17 November 2020; revised 17 November 2020; accepted 12 February 2021

Introduction

In 1966, while working at IBM, William Weeks presented an algorithm to perform a numerical Laplace transform inversion by expanding in terms of Laguerre polynomials. In his approach, he introduced two real parameters σ and b . He then proceeded to provide a few heuristic rules for determining optimal (σ, b) pairs and illustrated the efficacy of his algorithm for test cases that were numerically tractable at that time. Since then, the Weeks method has become one of a handful of well-known algorithms for numerically inverting a Laplace space function $F(s)$ to the time domain $f(t)$. None-the-less, the problem of optimally selecting the (σ, b) parameters in his algorithm remains difficult for arbitrary functions.

Machine learning is currently quickly evolving into a powerful tool for solving a wide variety of pattern recognition and optimization problems. Convolutional neural networks in particular are now readily available and can be quickly constructed in high level languages

such as MATLAB. In previous works, the authors have explored the optimal selection of Weeks' method parameters by conventional search algorithms and also applied Weeks' method to the solution of practical engineering problems. Given these experiences and the growth of machine learning, training a neural network to assist with the optimal (σ, b) problem in the Weeks method has therefore seemed like a natural application with real utility.

In this paper, we demonstrate that this machine learning approach can in fact be effective in the Weeks method for numerical inversion of the Laplace transform. To illustrate, we apply the Weeks method to

¹Raytheon Missile Systems, Tucson, AZ, USA

²Department of Mathematics, University of Arizona, Tucson, AZ, USA

Corresponding author:

Patrick O Kano, Raytheon Missile Systems, 1151 E. Hermans, Tucson, AZ 85756, USA.

Email: Patrick_O_Kano@Raytheon.com



the computation of the matrix exponential. The use of Laplace transform methods for matrix exponentiation is method twelve of the nineteen approaches described in the often cited work by Moler and Van Loan.¹ This follows from the general definition of a complex function $f: \mathbb{M}_n(\mathbb{C}) \rightarrow \mathbb{M}_n(\mathbb{C})$ of a square complex matrix $A \in \mathbb{M}_n(\mathbb{C})$

$$f(A) \equiv \frac{1}{2\pi i} \int_{\Gamma} f(s)(sI - A)^{-1} ds \quad (1)$$

In the particular case that scalar integrand function $f(s) = e^{st}$, one obtains a definition for the matrix exponential

$$e^{At} \equiv \frac{1}{2\pi i} \int_{\Gamma} e^{st}(sI - A)^{-1} ds \quad (2)$$

One recognizes from this that the Laplace transform space function corresponding to the matrix exponential $f(t) = e^{At}$ is the resolvent matrix $F(s) = (sI - A)^{-1}$. Alternatively, from the perspective of differential equations, the matrix exponential and resolvent matrix are the Laplace transform pair which arise from a set of first order ordinary differential equations involving matrix A and $\vec{u} \in \mathbb{C}_n$.²

$$\frac{d\vec{u}}{dt} = A\vec{u} \quad (3)$$

$$F(s) \equiv \mathcal{L}(f(t)) = \int_0^{\infty} e^{-st} f(t) dt \quad (4)$$

$$f(t) \equiv \mathcal{L}^{-1}(F(s)) = \frac{1}{2\pi i} \int_{\Gamma} e^{st} F(s) ds \quad (5)$$

$$\hat{u} = (sI - A)^{-1} \vec{u}_0 \quad (6)$$

$$\vec{u}(t) = e^{At} \vec{u}_0 \quad (7)$$

The importance of the matrix exponential in the numerical solution of partial differential equations and in numerous practical engineering problems has also motivated its accurate calculation as a focus for this paper.

Our paper is organized into five parts. After this first introductory section, the second part of the paper reviews the Weeks method. Of particular important here are the role of the Möbius transformation and the introduction of the two parameters (σ, b) . In the third section, we describe the regression neural

networks we constructed for predicting optimal (σ, b) values for computing a matrix exponential based on the input matrix. In this section, we also describe how the network was trained on data sets for four classes of matrices and with three different metrics. Algorithms for computing "truth" values for the matrix exponential based on the Cayley-Hamilton theorem and by rational approximation is also discussed since they are critical to the definition of the three metrics. The fourth section presents the results from the network training and describes the validation process and its results for the four matrix classes. The 3×3 skew-symmetric rotation evolution, 4×4 quaternion evolution, 5×5 random, and 6×6 Dramadah matrices used for illustration are also discussed there. The concluding section summarizes the findings and suggests directions for future research.

The Weeks method

Weeks' method is one of the most well known algorithms for the numerical inversion of a Laplace space function.³⁻⁶ (The notation in this paper follows the conventions used by Weideman.⁷⁻⁹) A recent survey of numerical inversion algorithms lists the Weeks' method and three of its variants on the top of this list.¹⁰ It is in part due to its popularity when compared to other inversion algorithms that it has been chosen for this paper. One of the strengths of the Weeks method over other well known approaches, such as the Talbot method,¹¹⁻¹⁴ the Fourier series method¹⁵, or Post's formula¹⁶⁻¹⁸ is that it returns an explicit expression for the time domain function. In particular, Weeks' method assumes that a smooth function of bounded exponential growth $f(t) : [0, \infty) \rightarrow \mathbb{C}$, given by the inverse Laplace transform of a complex function $F(s) : \mathbb{C} \rightarrow \mathbb{C}$

$$f(t) = \frac{1}{2\pi i} \int_{\Gamma} e^{st} F(s) ds \quad (8)$$

where Γ is a contour in the complex plane, can be expressed as the limit of an expansion in scalar Laguerre polynomials

$$f(t) = e^{\sigma t} \sum_{n=0}^{\infty} a_n e^{-bt} L_n(2bt) \quad (9)$$

$$f_N(t) \approx e^{\sigma t} \sum_{n=0}^{N-1} a_n e^{-bt} L_n(2bt) \quad (10)$$

The polynomials $L_n(x)$ for $n \geq 0$ are defined by the equation

$$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n} (e^{-x} x^n) \quad (11)$$

on $x \in (0, \infty)$.

The coefficients a_n , which may be scalars, vectors, or matrices, contain the information particular to the Laplace space function $F(s)$ and may be complex if $f(t)$ is complex. More importantly, these coefficients are time independent so that $f(t)$ can be evaluated at multiple times from a single set of coefficients.

The two free scaling parameters σ and b in the expansion must be selected according to the constraints that $0 < b$ and $\sigma_0 \leq \sigma$, where σ_0 is the abscissa of convergence for the singularities of $F(s)$. The restriction of b to positive values ensures that the weighted Laguerre polynomials $e^{-bt} L_n(2bt)$ are well behaved for large t , as shown in Figure 1. This condition also implies that $|e^{-bt} L_n(2bt)| < 1$. The convergence of the series is uniform.

To compute the coefficients a_n for a general vector or matrix function $F(s)$ one must perform the contour integration in the complex plane of equation (8). If one chooses the Bromwich contour $\Gamma(s) = \sigma + iy$, with $\sigma_0 \leq \sigma$ and y a real number

$$f(t) = \frac{e^{\sigma t}}{2\pi} \int_{-\infty}^{\infty} e^{iyt} F(\sigma + iy) dy \quad (12)$$

and assumes the expansion

$$f(t) = e^{\sigma t} \sum_{n=0}^{\infty} a_n e^{-bt} L_n(2bt) \quad (13)$$



Figure 1. Laguerre polynomials $e^{-bt} L_n(2bt)$.

then equating the two expressions yields

$$\sum_{n=0}^{\infty} a_n e^{-bt} L_n(2bt) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{iyt} F(\sigma + iy) dy \quad (14)$$

Conveniently, the weighted Laguerre polynomials have the Fourier representation⁷

$$e^{-bt} L_n(2bt) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{iyt} \frac{(iy - b)^n}{(iy + b)^{n+1}} dy \quad (15)$$

Performing the appropriate substitution, assuming it is possible to interchange the sum and integral, and equating integrands thus leaves

$$\sum_{n=0}^{\infty} a_n \frac{(iy - b)^n}{(iy + b)^{n+1}} = F(\sigma + iy) \quad (16)$$

The functions $\frac{(iy - b)^n}{(iy + b)^{n+1}}$ form a complete, orthogonal basis in $L_2(\mathbb{R})$ ⁷

In principle, one could try to use directly the orthogonality of the basis to determine a_n . These functions are highly oscillatory but could possibly be accurately estimated with adaptive integration. Weeks' insight was however to apply a Möbius transformation.

The Möbius transformation is one of the most fundamental mappings in mathematics.^{19–21} Algebraically, it can be expressed as the conformal mapping from the complex plane s to complex plane w according to $w = \frac{As+B}{Cs+D}$, $(A, B, C, D) \in \mathbb{C}$. The mapping is unique only up to an arbitrary scaling parameter. It is common when utilizing the mapping analytically, but not necessary, to allow the determinant $AD - BC = 1$. One of the mapping's properties that is particularly relevant to the current discussion is that it maps circles in s to circles in w . If the Bromwich contour $s = \sigma + iy$ is viewed as a circle with infinite radius, then its mapping is also a circle. Another relevant property of this transformation is that the singularities of $F(s)$ in the half-plane $\sigma < \sigma_0$ are mapped to the exterior of the unit circle in the w plane. An exception to this occurs when $F(s)$ has a singularity at infinity which then maps onto the unit circle itself (5). In the following we will assume that the singularities of $F(s)$ occur in a finite region of the complex plane. The resolvent matrix $F(s) = (sI - A)^{-1}$ Laplace space function corresponding to the matrix exponential e^A clearly belongs to this class of functions.

In his method, Weeks' choose for the Möbius transformation $(A = 1, B = -\sigma - b, C = 1, D = -\sigma + b)$.

The corresponding determinant is then $AD - BC = 2b$ and

$$w = \frac{s - \sigma - b}{s - \sigma + b} \quad (17)$$

For the Bromwich contour $s = \sigma + iy$

$$w = \frac{iy - b}{iy + b} \quad (18)$$

Using $y = \frac{ib(w+1)}{w-1}$ along the Bromwich contour, equation (16) can be expressed as

$$\sum_{n=0}^{\infty} a_n w^n = (iy + b)F(\sigma + iy) \quad (19)$$

$$\sum_{n=0}^{\infty} a_n w^n = \frac{2b}{1-w} F\left(\sigma - b \frac{w+1}{w-1}\right) \quad (20)$$

The coefficients a_n of the original expansion (9) are the coefficients of a Maclaurin series. The radius of convergence R is strictly greater than unity due to our selection of functions $F(s)$ which do not have a singularity at infinity. Furthermore, within this radius, the power series converges uniformly.

Cauchy's integral theorem²² provides a method to compute a_n . Since the function is analytic inside the radius of convergence $R > 1$, the integration can be performed along the unit circle $w = e^{i\theta}$

$$a_n = \frac{1}{2\pi i} \int_{|w|=1} \frac{1}{w^{n+1}} \frac{2b}{1-w} F\left(\sigma - b \frac{w+1}{w-1}\right) dw \quad (21)$$

$$a_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-in\theta} \frac{2b}{1 - e^{i\theta}} F\left(\sigma - b \frac{e^{i\theta} + 1}{e^{i\theta} - 1}\right) d\theta \quad (22)$$

Numerically the evaluation of the integral can be computed very accurately using the midpoint rule; $\theta_m = \frac{m\pi}{M}$, where $n = 0, \dots, N-1$ and $m = -M, \dots, M-1$

$$a_n \approx \frac{e^{-in\pi/(2M)}}{2M} \sum_{m=-M}^{M-1} \frac{2be^{-in\theta_m}}{1 - e^{i\theta_{m+1/2}}} F\left(\sigma - b \frac{e^{i\theta_{m+1/2}} + 1}{e^{i\theta_{m+1/2}} - 1}\right) \quad (23)$$

For $N = M$, this sum can be efficiently computed using the fast Fourier transform⁷ in $O(N \log(N))$ operations. The coefficients corresponding to negative indices which result from the FFT algorithm are not needed for the summation and are neglected.

Once the coefficients have been computed and the parameters selected, it is necessary to perform the Laguerre expansion. A naive approach is to generate the Laguerre polynomials using the recurrence relation

$$(n+1)L_{n+1}(x) = (2n+1-x)L_n(x) - nL_{n-1}(x) \quad (24)$$

with starting values

$$L_0(x) = 1$$

$$L_1(x) = 1 - x$$

multiply by the coefficients, and compute the sum

$$f(t) = e^{(\sigma-b)t} \sum_{n=0}^{\infty} a_n L_n(2bt) \quad (25)$$

The Laguerre polynomials however can be large for increasing n and thus lead to an unstable summation. A stable method which does not require explicit evaluation of the Laguerre polynomials is the backward Clenshaw algorithm.²³

This brings us now to consider the effect of the (σ, b) choice on the numerical integration. This is illustrated in Figure 2 where the σ parameter is varied and b held fixed. The truncated contour plotted is $s = \sigma + iy, |y| \leq 10$. One clearly sees that varying σ , while holding $b = 1/2$, yields the exact same discrete mapped points on the unit circle in the w plane. Varying σ in the s plane moves the contour closer to or further away from poles but in the w plane this action translates to leaving the circular contour fixed and moving the mapped poles.

The multifaceted roles that the b parameter plays in the Weeks method is more complex. The parameter can

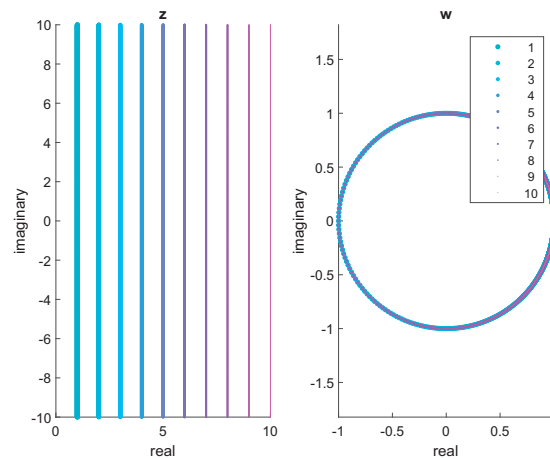


Figure 2. $z \rightarrow w$ mappings with σ varied and $b = 1/2$.

be seen to play at least three roles. One role is that b is a time scaling parameter in the Laguerre polynomial expansion. A second role is that it is $1/2$ the determinant in the Möbius transformation. As noted earlier, a Möbius transformation $w = \frac{Az+B}{Cz+D}$, $(A, B, C, D) \in \mathbb{C}$ is unique only up to an arbitrary scaling parameter. Given this ambiguity, it is common in analytic computations to define parameters so that the determinant is unity. In the standard Weeks formulation, the determinant is $2b$. Setting $b=1/2$ therefore appears natural and indeed in the original 1966 paper by William Weeks, he chooses this value as his starting point before later allowing it to vary in order to accommodate numerical errors given the limited precision available to him at the time.

In the third role, b defines the amount of contour truncation that is acceptable when evaluating the Bromwich integral numerically. Consider the w contours in Figure 3 where the truncated contour $s = \sigma + iy$, $|y| \leq 10$ is mapped with $\sigma=1$ and b varying. Typically, when choosing the values for the parameters in the Möbius transformation one picks three mapped pairs of points ($p \rightarrow \hat{p}, q \rightarrow \hat{q}, r \rightarrow \hat{r}$) in the z and w planes and then utilizes the invariance of the cross-ratio under Möbius transformation, equation (26), to arrive at parameters $\{A, B, C, D\}$.

$$\frac{(w - \hat{p})(\hat{q} - \hat{r})}{(w - \hat{r})(\hat{q} - \hat{p})} = \frac{(z - p)(q - r)}{(z - r)(q - p)} \quad (26)$$

In the case of the Weeks method, the three points pairs can be chosen as $z = \sigma \rightarrow w = -1$, $z \rightarrow +\infty \rightarrow w = 1$, and $z \rightarrow -\infty \rightarrow w = 1$. The last two limits can be deduced using L'Hospital's rule to handle the ratio of infinity values in the Möbius transformation. Clearly, in a numerical integration approach, one cannot map $z = \pm\infty$ and the contour

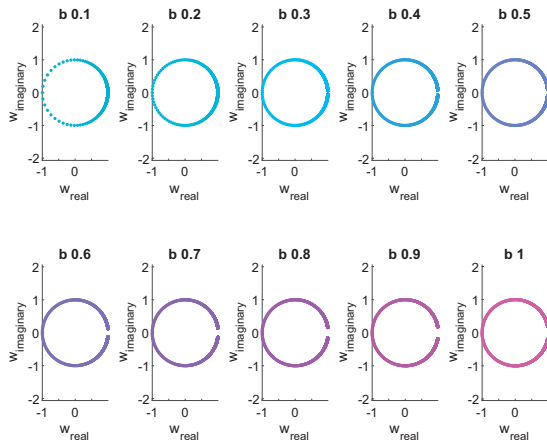


Figure 3. $z \rightarrow w$ mappings with b varied and $\sigma = 1$.

must be truncated. Varying b allows one to adjust the amount.

Machine learning

Neural network

Optimization problems are one of the most important applications of machine learning.^{24–27} A currently active of research regarding machine learning is its application to optimal numerical integration.²⁸ While certainly optimization algorithms have been used for parameter selection in numerical Laplace transform algorithms, to our knowledge neural network techniques have not been used previously for optimal parameter selection in the Weeks' method. In this paper, supervised learning was implemented to train a regression network where the input to the network are the values of the matrix itself and the corresponding continuous output are the (σ, b) pair which provides the most accurate estimate for the matrix exponential from the Weeks method.

To implement the network, we have leveraged the existing capability provided in the MATLAB machine learning toolbox.²⁹ With the basic components provided in the toolbox, we constructed simple convolution neural networks for four classes of matrices. The weights in a network were adjusted to accurately estimate a matrix exponential from the Weeks method with a particular (σ, b) pair.

To illustrate, we limited ourselves to square matrices of real values of size 3, 4, 5, and 6. A more detailed description of the selected matrices is provided below when describing network training. With respect to the network however, the actual convolutional neural networks used are quite simple. These small matrices allowed for only a few layers such as that shown in Figure 4. The contents on each of the layers is summarized in the Tables 1 to 4 for the four classes of matrices.

Detailed descriptions of each of the layers and their available options can be found in the MATLAB toolbox documentation.²⁹ For this specific implementation, the initial layer consists of a MATLAB image layer, where here the "image" is the matrix A . After this a single convolution layer is applied to the matrix. The results from the convolution are normalized and thresholding is applied. For the activation function, a standard rectified linear unit (ReLU) $f(x) = \max(0, x)$ was utilized. The results are then pooled and flattened down to provide weights for the two outputs (σ, b) . In the final regression layer, the weights are utilized to estimate a (σ, b) pair given the input matrix.

Certainly, a more sophisticated analysis could be performed to develop more accurate neural networks

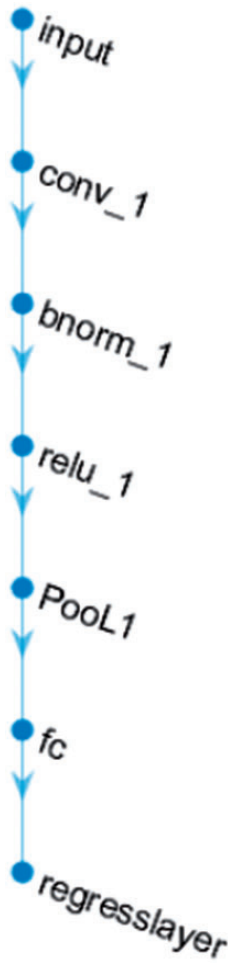


Figure 4. Convolution neural network example.

for this application. We choose simple networks in order to illustrate the approach and settled on those shown after some experimentation.

Regarding network training, a summary of the options used can be found in Table 5. The validation and training data are a 5% and 95% fraction of the total database created, respectively.

Training metrics

To gauge accuracy and to train the network, we considered three different metrics:

1. Jacobi Identity:
 $|\det(e^A) - e^{\text{trace}(A)}|$
2. Max Element Error:
 $\max_{m,n} |(e_{m,n}^A)_{Weeks} - (e_{m,n}^A)_{truth}|$
3. Total Elements Error:

$$\sum_{m,n} |(e_{m,n}^A)_{Weeks} - (e_{m,n}^A)_{truth}|$$

Table 1. 3×3 Direction cosines matrices evolution.

Neural network layers	Parameters
imageInputLayer	([3 3 1], 'Name', 'input')
convolution2dLayer	(2,2,'Name', 'conv_1')
batchNormalizationLayer	('Name','bnorm_1')
reluLayer	('Name', 'relu_1')
maxPooling2dLayer	(2,'Stride',2,'Name','Pool1')
fullyConnectedLayer	(2, 'Name', 'fc')
regressionLayer	('Name','regresslayer')

Table 2. 4×4 Quaternions evolution.

Neural network layers	Parameters
imageInputLayer	([4 4 1], 'Name', 'input')
convolution2dLayer	(2,2,'Name', 'conv_1')
batchNormalizationLayer	('Name','bnorm_1')
reluLayer	('Name', 'relu_1')
maxPooling2dLayer	(2,'Stride',2,'Name','Pool1')
fullyConnectedLayer	(2, 'Name', 'fc')
regressionLayer	('Name','regresslayer')

Table 3. 5×5 Random matrices.

Neural network layers	Parameters
imageInputLayer	([5 5 1], 'Name', 'input')
convolution2dLayer	(2,5,'Stride',1,'Name', 'conv_1')
batchNormalizationLayer	('Name','bnorm_1')
reluLayer	('Name', 'relu_1')
maxPooling2dLayer	-
fullyConnectedLayer	(2, 'Name', 'fc')
regressionLayer	('Name','regresslayer')

Table 4. 6×6 Dramadah matrices.

Neural network layers	Parameters
imageInputLayer	([6 6 1], 'Name', 'input')
convolution2dLayer	(5,8,'Stride',1,'Name', 'conv_1')
batchNormalizationLayer	('Name','bnorm_1')
reluLayer	('Name', 'relu_1')
maxPooling2dLayer	(3,'Stride',1,'Name','Pool1')
fullyConnectedLayer	(2, 'Name', 'fc')
regressionLayer	('Name','regresslayer')

The Jacobi identity metric is particularly interesting in that it provides a self-contained error estimate that depends only on the original matrix A and its approximation e^A . There is no need for another approximation method to utilize for validation.

For the other two metrics based on comparing the Weeks estimated matrix exponential with a truth value, it was necessary to obtain "truth". Clearly, utilizing another trusted approximation method and simply comparing approximations is straight forward and is

Table 5. Network training options.

Matrix	3 × 3	4 × 4	5 × 5	6 × 6
Algorithm	sgdm	sgdm	sgdm	sgdm
MiniBatchSize	128	128	16	16
GradientThreshold	5	5	5	5
InitialLearnRate	0.0005	0.0005	0.0005	0.0005
LearnRateSchedule	piecewise	piecewise	piecewise	piecewise
LearnRateDropPeriod	1	1	1	1
MaxEpochs	5	5	5	5
Verbose	true	true	true	true
Plots	training-progress	training-progress	training-progress	training-progress
Shuffle	every-epoch	every-epoch	every-epoch	every-epoch
ExecutionEnvironment	gpu	gpu	gpu	gpu
ValidationPatience	50	50	50	50
Momentum	0.5	0.5	0.5	0.5
ValidationData	X,Y	X,Y	X,Y	X,Y

in-fact the approach we have utilized in past publications.^{13,30,31} When dealing with two high accuracy methods however, this direct comparison of the two approximations leaves one questioning which of the two is actually the more accurate and thus leads to some ambiguity in the error metric. In this paper, to avoid that ambiguity, we have employed a 2-out-of-3 error estimate approach. For the max element and total elements error metrics, we have computed the matrix exponential via three different algorithms^{1,32-34}

1. Weeks Method
2. Padé Rational Approximation
3. Cayley-Hamilton Theorem

The max element and total elements errors are then defined as the minimum of the differences between the Weeks and the Padé approximations and the differences between the Weeks approximation and the Cayley-Hamilton expression.

$$Error(e_{Weeks}^A) = \quad (27)$$

$$\min(\delta(e_{Weeks}^A - e_{Pade}^A), \delta(e_{Weeks}^A - e_{Cayley-Hamilton}^A))$$

It is assumed that if two of the algorithms agree up to n decimal digits that those digits arose because the algorithms agree up to that accuracy and not due to round-off.

The Padé rational approximation is the algorithm implemented for the matrix exponential in MATLAB's *expm* function.³⁵ Briefly, the original matrix is scaled by a factor of 2^n to create a matrix whose eigenvalues are reduced so that the infinity norm is less than $1/2$. A rational approximation is

then computed from the reduced matrix. To regain the exponential of the original matrix, the corresponding reduced matrix exponential is squared n times. Written concisely with matrix polynomials P and Q

$$e^A = (e^{A/2^n})^{2^n} \approx \left(\frac{P_m(A/2^n)}{Q_m(A/2^n)} \right)^{2^n} \quad (28)$$

The Padé rational approximation approach has the advantages of possessing a well understood error estimate and widespread use due to its inclusion in MATLAB.

For a small square matrix it is not necessary to approximate the matrix exponential as it can be expressed analytically via the Cayley-Hamilton theorem.³⁴ The theorem states simply that *every matrix satisfies its own characteristic equation*. That is, given a square matrix A with dimension n and with a characteristic polynomial

$$\Delta(s) = |sI - A| = s^n + c_{n-1}s^{n-1} + \dots + c_0 \quad (29)$$

and defining a corresponding matrix polynomial, formed by substituting A for s

$$\Delta(A) = A^n + c_{n-1}A^{n-1} + \dots + c_0I \quad (30)$$

one has $\Delta(A) = [0]$.

A consequence of this theorem, is that *the analytic function of a matrix A of dimension n may be expressed as a polynomial of degree $(n-1)$ or less*.

$$f(A) = \sum_{k=0}^{n-1} \alpha_k A^k \quad (31)$$

The exponential function is analytic and thus the matrix exponential can be determined by finding expressions for the coefficients $\{\alpha_k\}$.

$$e^A = \sum_{k=0}^{n-1} \alpha_k A^k \quad (32)$$

To find those coefficients, it is sufficient to solve the corresponding set of equations given by the eigenvalues of A . For distinct eigenvalues, the eigenvalue equation is

$$e^{\lambda_i} = \sum_{k=0}^{n-1} \alpha_k \lambda_i^k \quad (33)$$

For any eigenvalue of multiplicity m , the first $(m-1)$ derivatives of $\Delta(s)$ all vanish at those eigenvalues. In that case, the derivatives up to the multiplicity of the eigenvalue are considered.

$$f(\lambda_i) = \sum_{k=0}^{n-1} \alpha_k \lambda_i^k = R(\lambda_i) \quad (34)$$

$$\frac{df}{d\lambda} \Big|_{\lambda=\lambda_i} = \frac{dR}{d\lambda} \Big|_{\lambda=\lambda_i} \quad (35)$$

$$\frac{d^{m-1}f}{d\lambda^{m-1}} \Big|_{\lambda=\lambda_i} = \frac{d^{m-1}R}{d\lambda^{m-1}} \Big|_{\lambda=\lambda_i} \quad (36)$$

The main disadvantages of the Cayley-Hamilton expressions are the complexity of the analytic expressions and their sensitivity to round-off error. Except for 2×2 , 3×3 , and 4×4 matrices, analytic expressions for eigenvalues are generally impossible to derive. For our training purposes, however, the formula is tractable and to solve the equations we have used Mathematica (2019).³⁶ Since the Cayley-Hamilton matrix exponential expressions in MATLAB are not easily obtained and they are central to our analysis, we have included for the reader the exact Mathematica script used for solving the equations and their conversion to MATLAB in the supplemental material.³⁷ We also present the specific case of the Cayley-Hamilton theorem applied to a 4×4 matrix.

Briefly, for a general square 4×4 matrix, there are five eigenvalue cases to consider

1. All distinct: $\lambda_1 \neq \lambda_2 \neq \lambda_3 \neq \lambda_4$
2. One pair, other 2 distinct: $\lambda_1 \neq \lambda_2 \neq \lambda_3, \lambda_3 = \lambda_4$
3. Two distinct pairs: $\lambda_1 = \lambda_2, \lambda_3 = \lambda_4, \lambda_1 \neq \lambda_3$
4. 3 identical, 1 unique: $\lambda_1 = \lambda_2 = \lambda_3, \lambda_1 \neq \lambda_4$
5. All identical: $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4$

If we allow $b_i = e^{\lambda_i}$, then the equations to solve in the five cases become:

1. All distinct

$$b_1 = a_0 + a_1 \lambda_1 + a_2 \lambda_1^2 + a_3 \lambda_1^3 \quad (37)$$

$$b_2 = a_0 + a_1 \lambda_2 + a_2 \lambda_2^2 + a_3 \lambda_2^3 \quad (38)$$

$$b_3 = a_0 + a_1 \lambda_3 + a_2 \lambda_3^2 + a_3 \lambda_3^3 \quad (39)$$

$$b_4 = a_0 + a_1 \lambda_4 + a_2 \lambda_4^2 + a_3 \lambda_4^3 \quad (40)$$

2. One pair, other 2 distinct

$$b_1 = a_0 + a_1 \lambda_1 + a_2 \lambda_1^2 + a_3 \lambda_1^3 \quad (41)$$

$$b_2 = a_0 + a_1 \lambda_2 + a_2 \lambda_2^2 + a_3 \lambda_2^3 \quad (42)$$

$$b_3 = a_0 + a_1 \lambda_3 + a_2 \lambda_3^2 + a_3 \lambda_3^3 \quad (43)$$

$$\lambda_3 b_3 = a_1 + 2 \cdot a_2 \lambda_3 + 3 \cdot a_3 \lambda_3^2 \quad (44)$$

3. Two distinct pairs

$$b_1 = a_0 + a_1 \lambda_1 + a_2 \lambda_1^2 + a_3 \lambda_1^3 \quad (45)$$

$$\lambda_1 b_1 = a_1 + 2 \cdot a_2 \lambda_1 + 3 \cdot a_3 \lambda_1^2 \quad (46)$$

$$b_3 = a_0 + a_1 \lambda_3 + a_2 \lambda_3^2 + a_3 \lambda_3^3 \quad (47)$$

$$\lambda_3 b_3 = a_1 + 2 \cdot a_2 \lambda_3 + 3 \cdot a_3 \lambda_3^2 \quad (48)$$

4. 3 identical, 1 unique

$$b_1 = a_0 + a_1 \lambda_1 + a_2 \lambda_1^2 + a_3 \lambda_1^3 \quad (49)$$

$$\lambda_1 b_1 = a_1 + 2 \cdot a_2 \lambda_1 + 3 \cdot a_3 \lambda_1^2 \quad (50)$$

$$\lambda_1^2 \cdot b_1 = 2 \cdot a_2 + 6 \cdot a_3 \lambda_1 \quad (51)$$

$$b_4 = a_0 + a_1 \lambda_4 + a_2 \lambda_4^2 + a_3 \lambda_4^3 \quad (52)$$

5. All identical

$$b = a_0 + a_1 \lambda + a_2 \lambda^2 + a_3 \lambda^3 \quad (53)$$

$$\lambda b = a_1 + 2 \cdot a_2 \lambda + 3 \cdot a_3 \lambda^2 \quad (54)$$

$$\lambda^2 \cdot b = 2 \cdot a_2 + 6 \cdot a_3 \lambda \quad (55)$$

$$\lambda^3 \cdot b = 6 \cdot a_3 \quad (56)$$

A simple check of these equations is to note for the case that all of the eigenvalues are identical, the coefficients are simply

$$a_3 = (1/6)b\lambda^3 \quad (57)$$

$$a_2 = (1/2)b\lambda^2 \cdot (\lambda^2 - 1) \quad (58)$$

$$a_1 = b\lambda(1 + \lambda^2 - (4/3)\lambda^4) \quad (59)$$

$$a_0 = b(1 + (2/3)\lambda^6 - (1/2)\lambda^4 - \lambda^2) \quad (60)$$

If the matrix is the 4×4 identity matrix, then $e^I = e \cdot I$, $\lambda = 1$, $b = e^I$, and the coefficient equations simplify to

$$a_3 = e/6 \quad (61)$$

$$a_2 = 0 \quad (62)$$

$$a_1 = (2/3)e \quad (63)$$

$$a_0 = (1/6)e \quad (64)$$

Now performing the sum (32), $e^I = (1/6)eI + (2/3)eI + 0 * \hat{I}^2 + (1/6) * \hat{I}^3$ or $e^I = e(1/6 + 2/3 + 1/6) I = e \cdot I$, which confirms our calculations.

Test case matrices

To illustrate the formalism outlined above, we have computed the matrix exponential of four square matrices

- 3×3 Skew Symmetric Direct Cosine Matrix Evolution Matrices
- 4×4 Quaternion Evolution Matrices
- 5×5 Random Correlation Matrices
- 6×6 Dramadah Matrices

The 3×3 and 4×4 skew symmetric and quaternion evolution matrices are those in the ordinary differential equations that describing rigid body rotation.^{38,39} For a 3×3 direction cosines matrix (DCM) M , its evolution in the absence of external forces is described by a cross product of the rotation rates vector with the DCM. This cross-product can be expressed as a skew symmetric 3×3 matrix A .

$$\frac{dM}{dt} = A \cdot M \quad (65)$$

$$A = \begin{pmatrix} 0 & -\omega_\psi & \omega_\theta \\ \omega_\psi & 0 & -\omega_\phi \\ -\omega_\theta & \omega_\phi & 0 \end{pmatrix} \quad (66)$$

Since the 3×3 matrix is real and skew-symmetric it has eigenvalues which are either zero or purely imaginary, specifically $\lambda_1 = 0, \lambda_{2,3} = \pm i\sqrt{\omega_\psi^2 + \omega_\theta^2 + \omega_\phi^2}$.

The case is similar for the 4-vector of quaternions \vec{q}

$$\vec{q} = \begin{bmatrix} \cos(\delta) \\ \cos(\alpha)\sin(\delta) \\ \cos(\beta)\sin(\delta) \\ \cos(\gamma)\sin(\delta) \end{bmatrix} \quad (67)$$

$\alpha, \beta, \gamma = \text{direction angles of the axis}$
 $\delta = \text{measure of rotation angle}$
 $\hat{u} = \{\cos(\alpha), \cos(\beta), \cos(\gamma)\}$
 $= \text{unit vector axis of rotation}$

One can define an ordinary differential equation for the evolution of \vec{q} by means of a real skew symmetric matrix

$$\frac{d\vec{q}}{dt} = A \cdot \vec{q} \quad (68)$$

$$A = \frac{1}{2} \begin{pmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{pmatrix} \quad (69)$$

This choice for \vec{q} and the matrix A is only one of a number of possible formulations.³⁹ The solution to this equation is however also the matrix exponential $\vec{q}(t) = e^{At}\vec{q}_0$ and the eigenvalues are either zero or purely imaginary. Specifically, $\lambda_{1,2} = \frac{i}{2}\sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2}$ and $\lambda_{3,4} = -\frac{i}{2}\sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2}$.

For 5×5 matrices, real square 5×5 random correlation matrices were selected. The matrices were chosen to stress the neural network approach studied here. We have also studied the matrix exponential of random matrices to some extent in our previous publication,¹³ which utilized a Dempster-Shafer evidential theory approach to parameter selection in Talbot's method for numerical inversion, and thus it seemed appropriate to do so again here. To create the database, we leveraged the MATLAB gallery matrix function *gallery* (*'randcorr',n*) with $n=5$. This generates a random square correlation matrix that is symmetric positive semidefinite with ones on the diagonal. The eigenvalues from these matrices are real, drawn from a uniform distribution, and thus are distributed fundamentally

differently in the complex plane than for the 3×3 and 4×4 rotation matrices.

The 6×6 Dramadah matrix is the largest matrix for which we computed the matrix exponential using the Weeks, Padé, and Cayley-Hamilton methods. This matrix was also constructed by leveraging the MATLAB gallery matrix `gallery('dramadah',6,3)`. This is a binary matrix (70) of zeros and ones.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \quad (70)$$

An interesting fact is that the determinant of the n th Dramadah matrix is the n th Fibonacci number, in this case 8. This can be verified directly from the product of the eigenvalues, which are approximately equal to:

$$\begin{pmatrix} 2.6523 \\ 0.1307 + 1.2512i \\ 0.1307 - 1.2512i \\ 1.3863 \\ 0.8500 + 0.8077i \\ 0.8500 - 0.8077i \end{pmatrix} \quad (71)$$

To vary the values of the Dramadah matrix, the matrix was multiplied by random values from $\gamma \in [-1, 1] \setminus 0$. The eigenvalues of the scaled Dramadah matrices are then clearly scalar multiples of the eigenvalues (71). More importantly, the corresponding inverse matrix becomes $(1/\gamma)A^{-1}$ where

$$A^{-1} = \begin{pmatrix} 5 & -3 & 2 & -1 & 1 & -1 \\ 3 & 3 & -2 & 1 & -1 & 1 \\ -3 & 5 & 2 & -1 & 1 & -1 \\ -2 & -2 & 4 & 2 & -2 & 2 \\ -1 & -1 & -2 & 5 & 3 & -3 \\ -1 & -1 & -2 & -3 & 3 & 5 \end{pmatrix} / 8 \quad (72)$$

Given that $|\gamma| < 1$, the corresponding inverse matrix may have large values. This could be a challenge for the Weeks method which requires matrix inversion along the integration contour and thus is an excellent stressing case.

Results

With the Weeks method and machine learning approaches outlined above, results from their application are reported here. Two separate sets of results are discussed. First are the results of the supervised

training to construct the twelve neural networks, four tests case with the three metrics. The main takeaway from this section is that the simple networks employed were able to capture the shape of the error surface as a function of (σ, b) . The total element and maximum element metrics, as expected, were found to be useful for this purpose. What was surprising and particularly useful to find is that the supervised learning by the neural network with the Jacobi identity led to a predictive network.

The second set of results focus on validation. In particular, we have focused on the neural networks created using the Jacobi identity. The networks based on the other two metrics work very well, but because they require a second approximation for training, we decided to focus on the metric which leads to a self-contained algorithm. Even for the simple neural networks, the validation results demonstrate that machine learning can be utilized with the Weeks method to accurately compute the matrix exponential.

Neural network training results

Table 6 contains a summary of the specific set parameters chosen to construct the databases of matrices used to train the networks. In all cases, sixteen Laguerre polynomials were used in the estimate of the matrix exponential. Clearly, accuracy will be dependent on the number of basis functions but for illustration purposes, this number of polynomials was found to be sufficient.

For the 3×3 DCM evolution and 4×4 quaternion evolution matrices, the matrices were constructed from the angles rates. The roll and yaw rates were taken from dividing the interval $[-1, 1]$ evenly into 81 intervals, while the pitch was sampled from 41 linearly space intervals from $[-1, 1]$. The elements of the random 5×5 matrices were those obtained from the MATLAB toolbox directly and drawn from $[-1, 1]$. Last, the 6×6 Dramadah matrices were formed from multiplying the matrix by a constant values from $[-1, 1]$ where values were linearly spaced by 5000 points, excluding 0.

To illustrate the results from the training phase, two types of plots have been constructed. The first is a set of surface plots for error of the Weeks method estimated matrix exponential as a function of (σ, b) as

Table 6. Training database resolution parameters.

Parameter	3×3	4×4	5×5	6×6
σ	[0,10]	[0,10]	[0,10]	[0,10]
b	(0,5]	(0,5]	(0,5]	(0,5]
N_σ	21	21	21	41
N_β	21	21	21	41

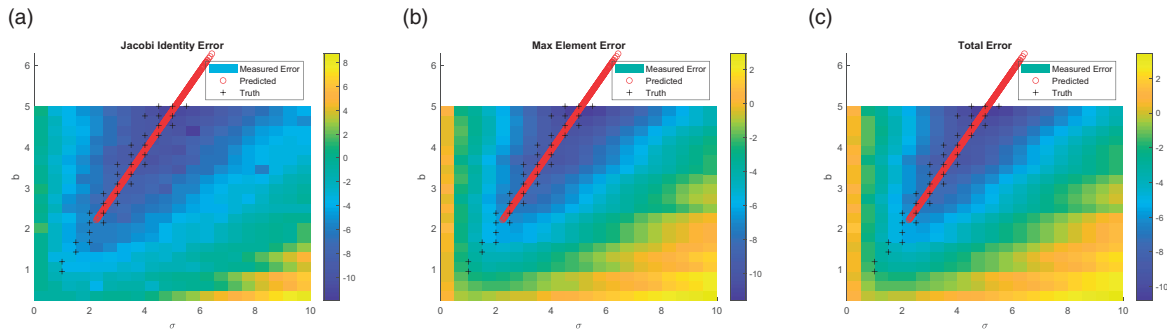


Figure 5. Error surfaces: 3×3 skew symmetric. (a) Jacobi. (b) Max element. (c) Total.

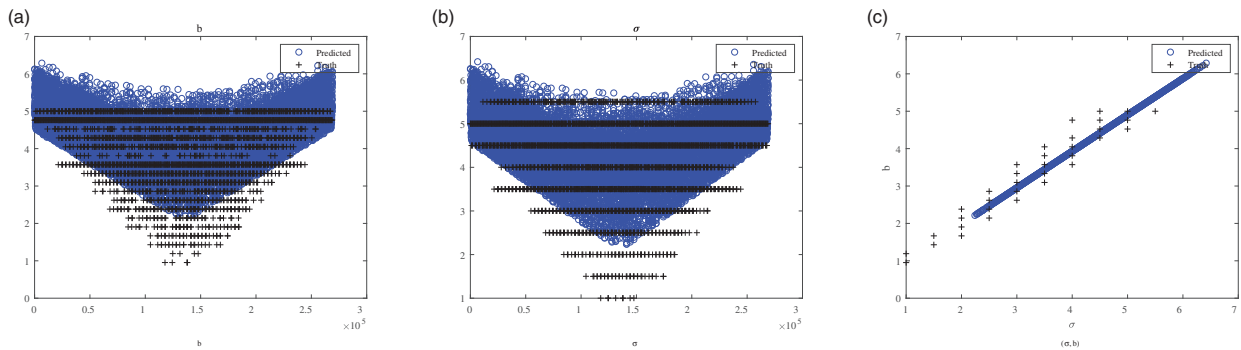


Figure 6. Truth vs. prediction: 3×3 skew symmetric.

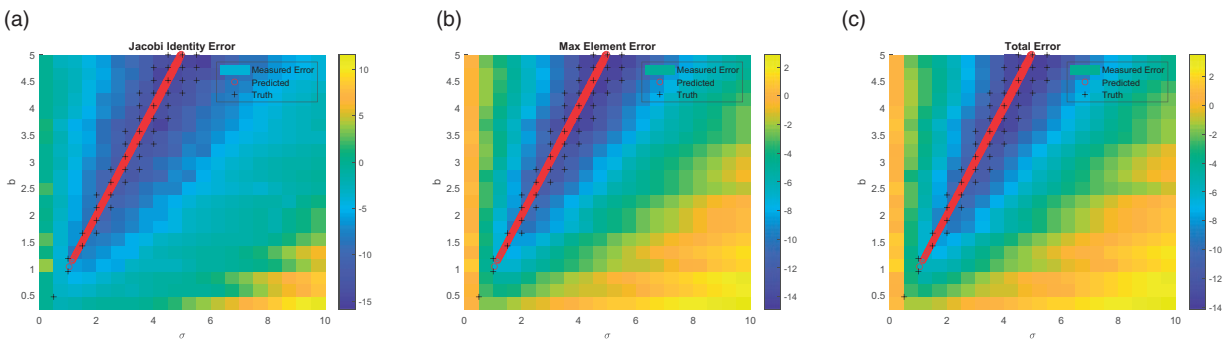


Figure 7. Error Surfaces: 4×4 quaternions. (a) Jacobi. (b) Max element. (c) Total.

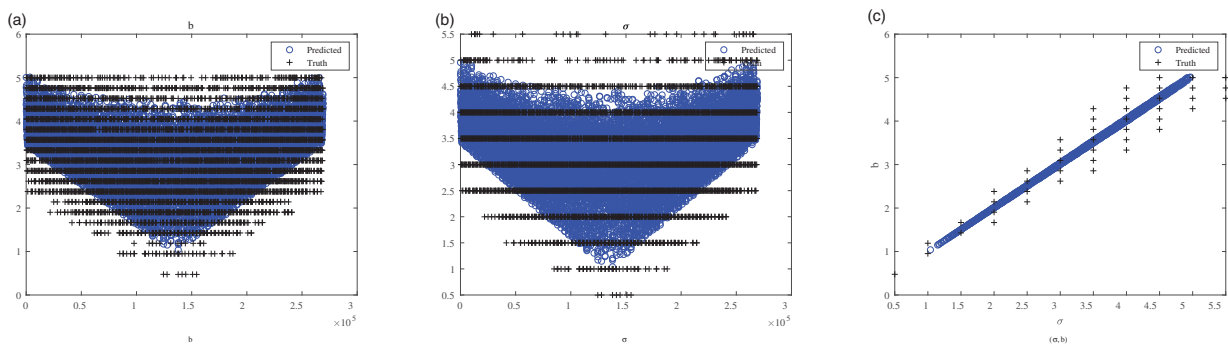


Figure 8. Truth vs. Prediction: 4×4 quaternions.

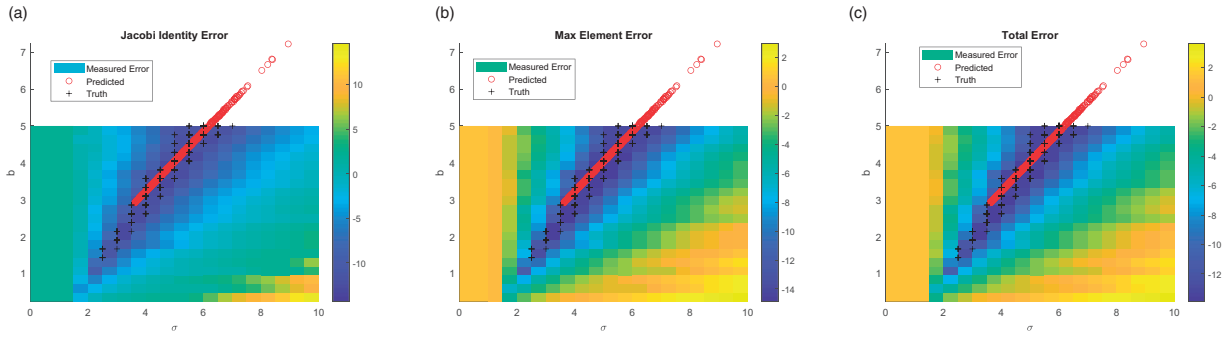


Figure 9. Error surfaces: 5×5 random. (a) Jacobi. (b) Max element. (c) Total.

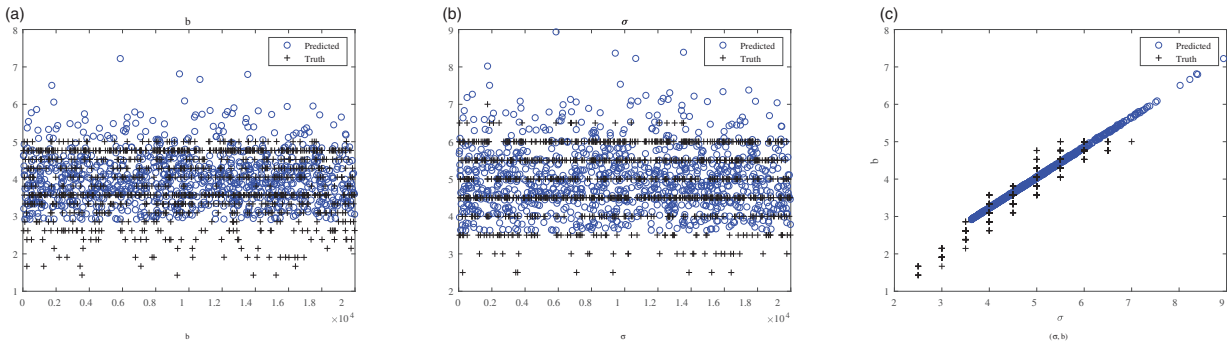


Figure 10. Truth vs. prediction: 5×5 random.

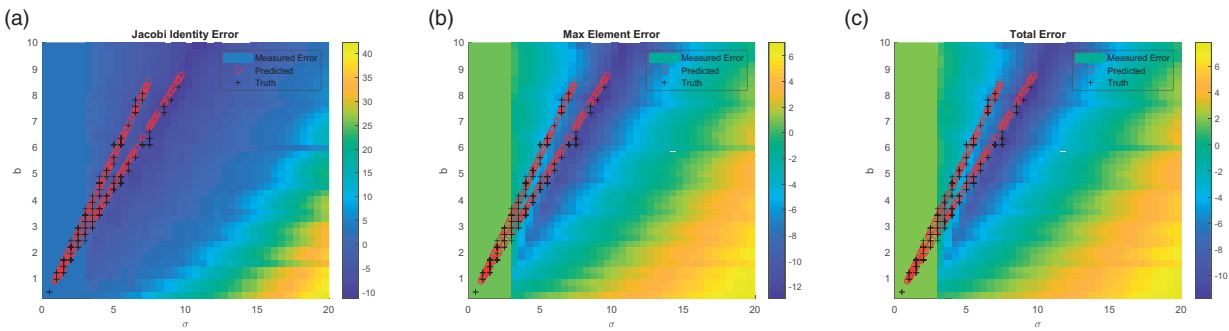


Figure 11. Error surfaces: 6×6 Dramadah. (a) Jacobi. (b) Max element. (c) Total.

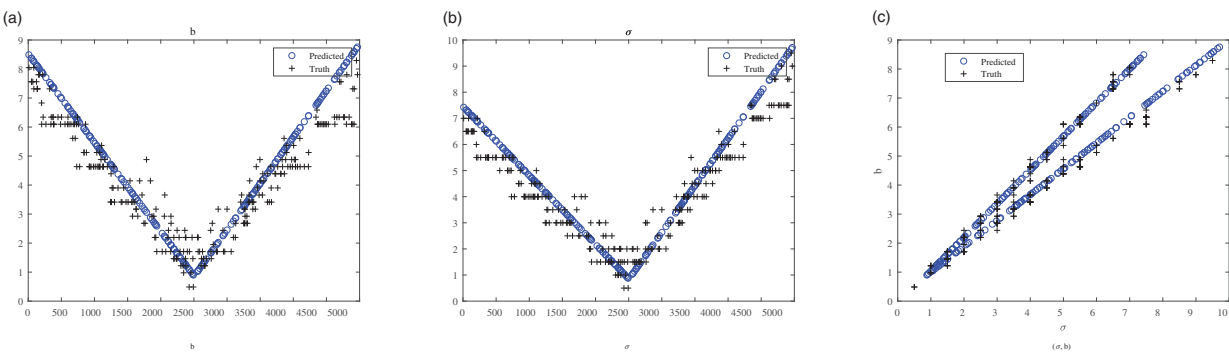


Figure 12. Truth vs. prediction: 6×6 Dramadah.

measured by the Jacobi identity metric, the maximum element error metric, and total matrix elements error metric. Recall from the previous discussion, that the truth when creating the element error surfaces is defined based on 2-out-of-3 rule where the Weeks estimate is compared with the Padé and Cayley-Hamilton

theorem derived matrix exponentials. The Jacobi identity surface is defined by directly comparing A and the Weeks method estimated matrix exponential. The surfaces plotted are for a typical matrix from each of the four families of matrices, the skew-symmetric (Figure 5), the quaternions (Figure 7), the random

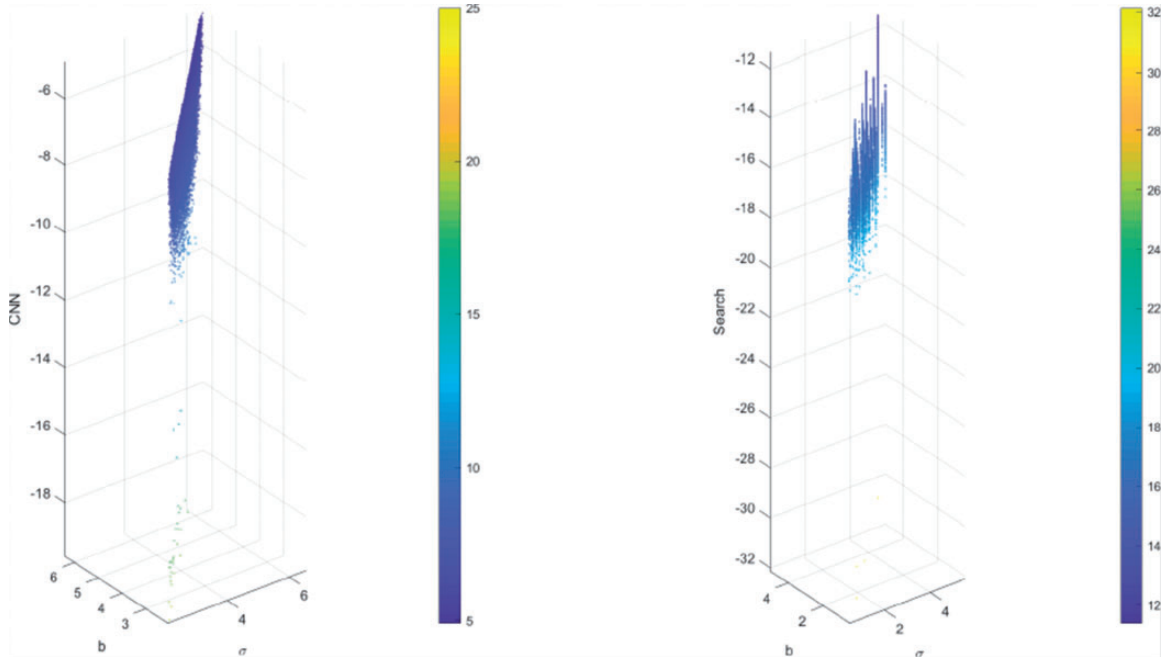


Figure 13. Jacobi error: 3×3 skew symmetric matrix.

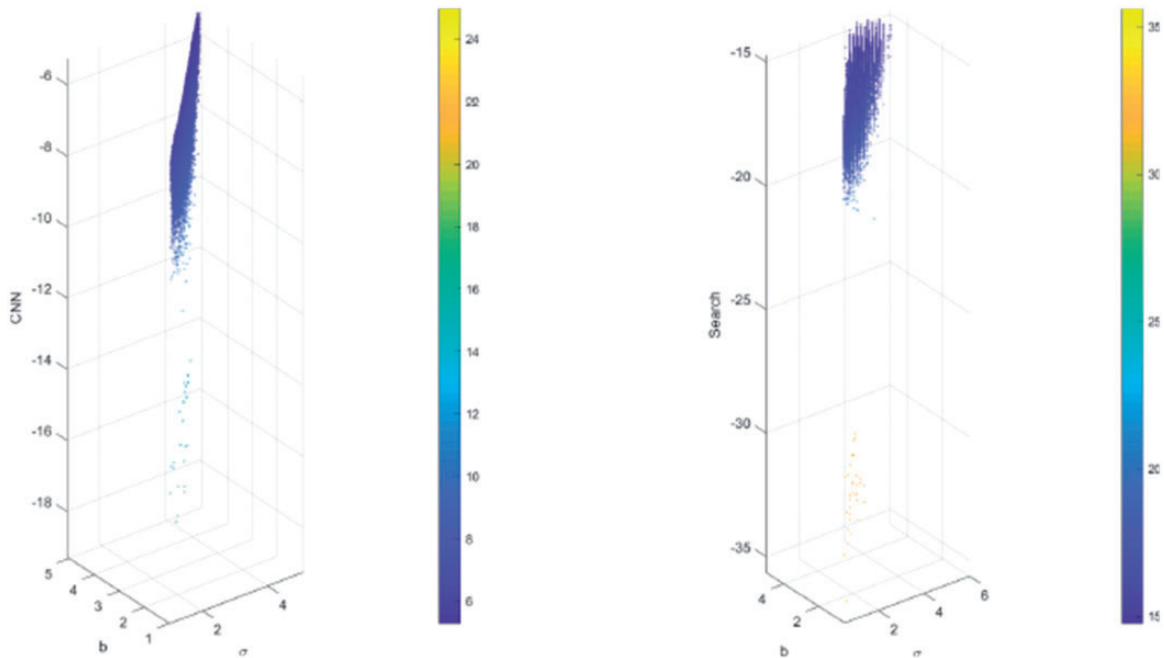


Figure 14. Jacobi error: 4×4 quaternions evolution matrix.

(Figure 9), and the Dramadah (Figure 11). For every matrix in each class, we have computed a corresponding surface.

What one finds from these surface plots is that there are general structures to the error surfaces for each of the three metrics. It is therefore possible to find a minimum that corresponds to an optimal (σ, b) for the construction of the matrix exponential for all three metrics.

Also plotted on these surfaces are cross and circle pairs from each of the 5% of validation matrices

reserved from the total set of matrices utilized in the process of training the neural network. The crosses mark the (σ, b) pairs that corresponds to the true minimum of the error surfaces for each tested matrix. The circles correspond to the neural network estimated (σ, b) pairs. The cross and circle pairs should be close but are rarely identical due to the finite sampling of the σ and b when creating the surfaces and the fact that the regression from the neural network returns a value on a continuous interval. To be clear, the shaded surface is for only one matrix from the class while the crosses and

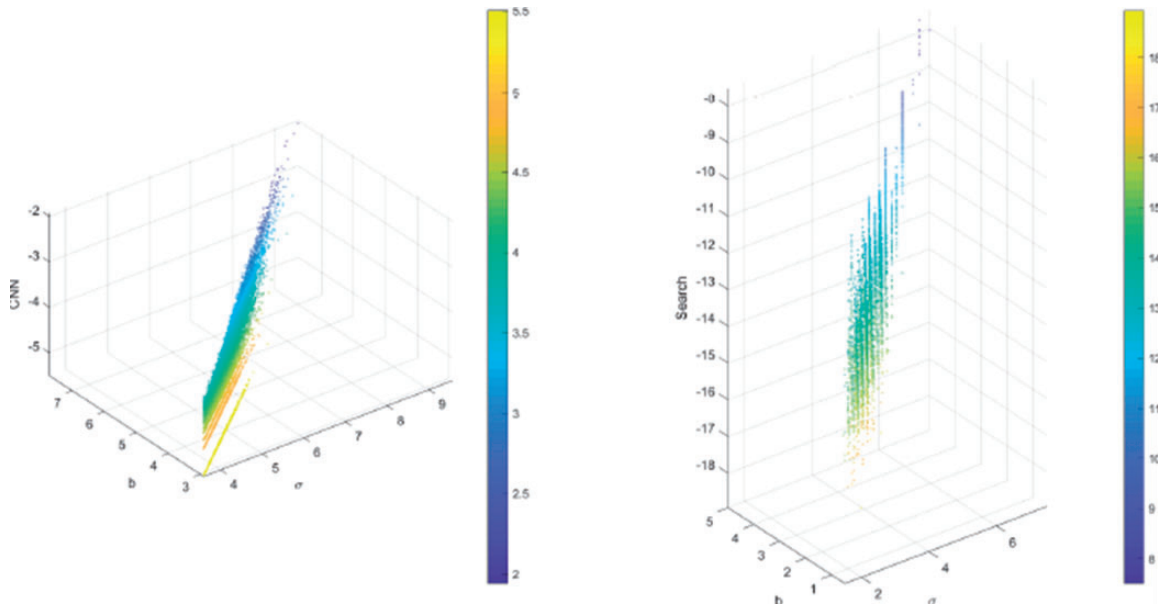


Figure 15. Jacobi error: 5×5 random matrix.

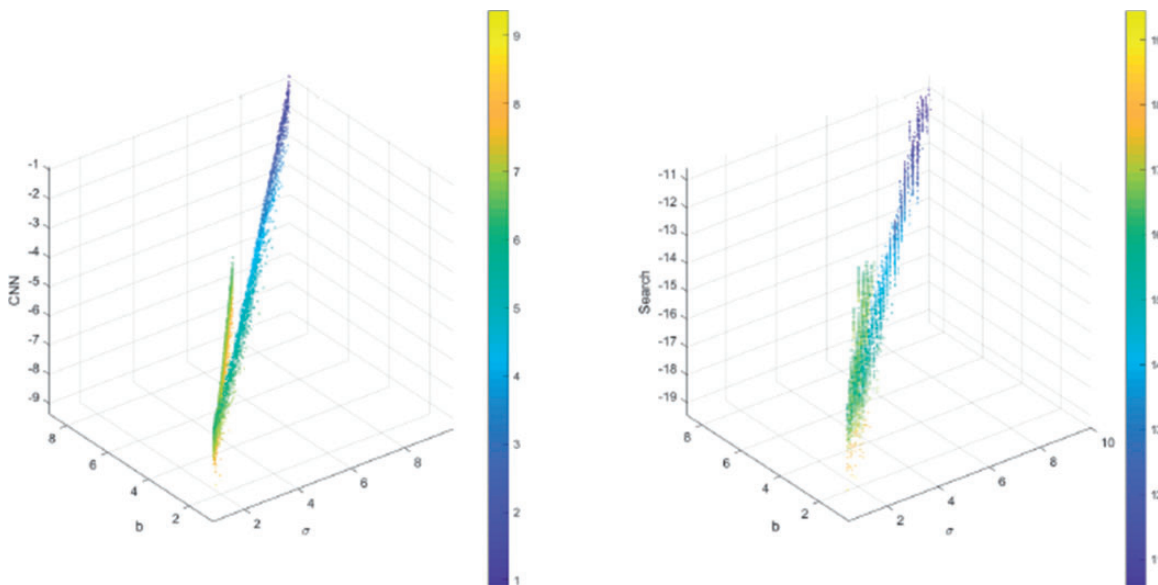


Figure 16. Jacobi error: 6×6 Dramadah.

circles are for all of the matrices in the 5% of the data set not used when training the network. For illustration purposes, all of the pairs are shown on the same plot. These figures never-the-less show that there is a pattern to the predictions of the neural network that overlaps with the underlying minimal error surfaces across matrices of a class.

The second set of plots further illustrates that there is a pattern to the predictions of the neural network that overlaps with the underlying minimal error surfaces across matrices of a class. Plotted in Figure 6 for the DCM evolution, Figure 8 for the quaternions, Figure 10 for the random, and Figure 12 for the Dramadah, are the truth and neural network estimated

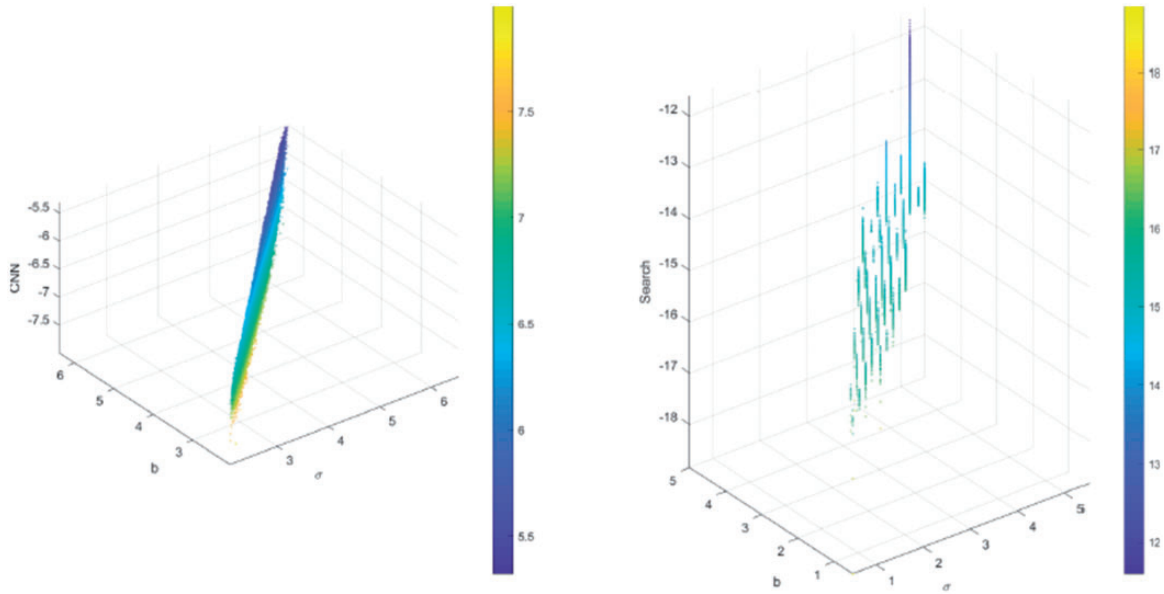


Figure 17. Max element error: 3×3 skew symmetric matrix.

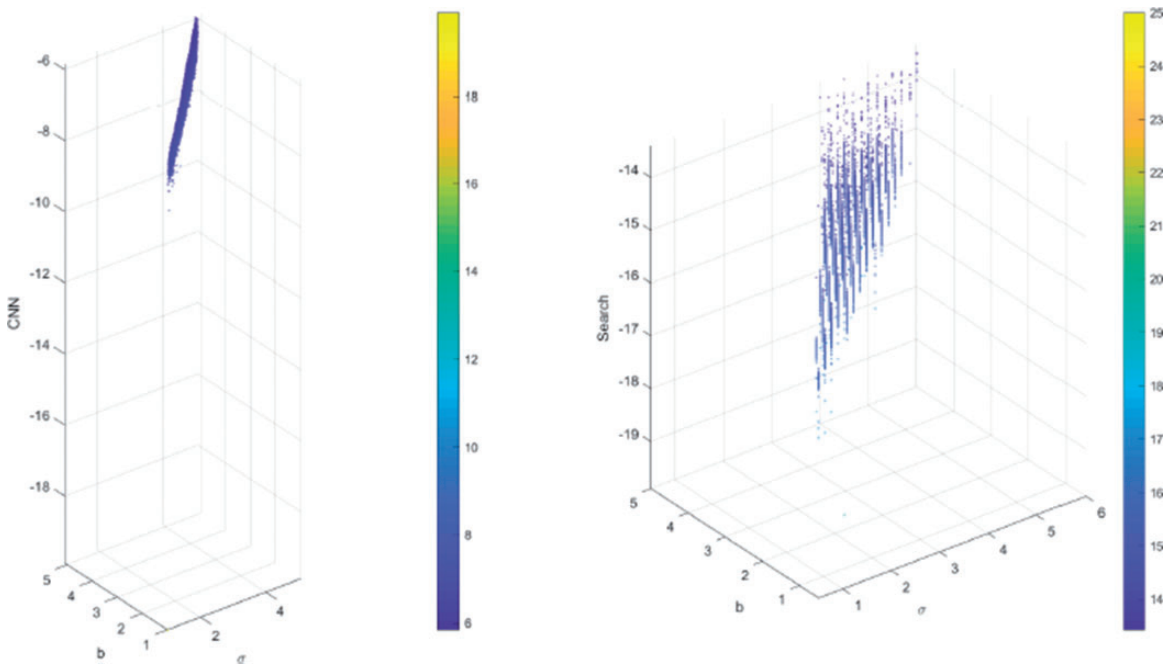


Figure 18. Max element error: 4×4 quaternions evolution matrix.

σ and b parameters as a function of the test case. These too show that the simple neural networks used for this analysis none-the-less capture the distribution of optimal (σ, b) values as the corresponding matrices are varied.

Validation

In the previous training results, the optimal (σ, b) predictions from the twelve simple neural networks were found to be reasonably accurate when compared to the true minimum on the (σ, b) grid defined in Table 6.

For a more extensive validation of the convolutional networks generated to predict Weeks' method (σ, b) , we here take the full database and both utilize the network to predict a (σ, b) pair and then also demonstrate that the matrix exponential with the Weeks method is reasonably accurate. That is, for every matrix of a class in the database, the corresponding network trained using the Jacobi identity was run to generate an estimated optimal (σ, b) and the error measured. Note that this is different from the training. In the training, all twelve networks were generated and compared on the limited data subset. Here we still use the three error

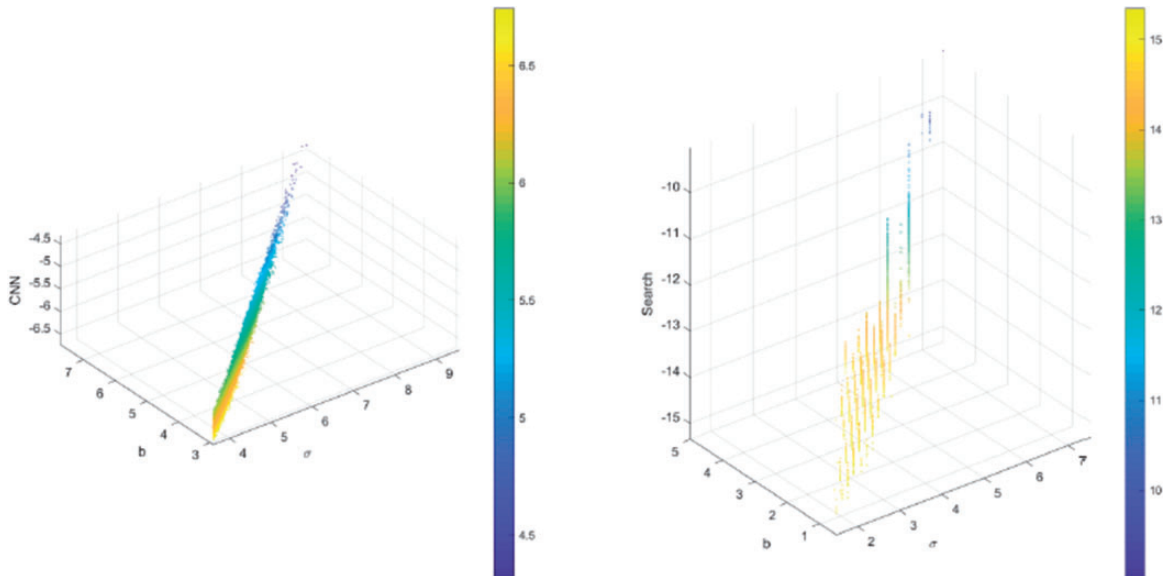


Figure 19. Max element error: 5×5 random matrix.

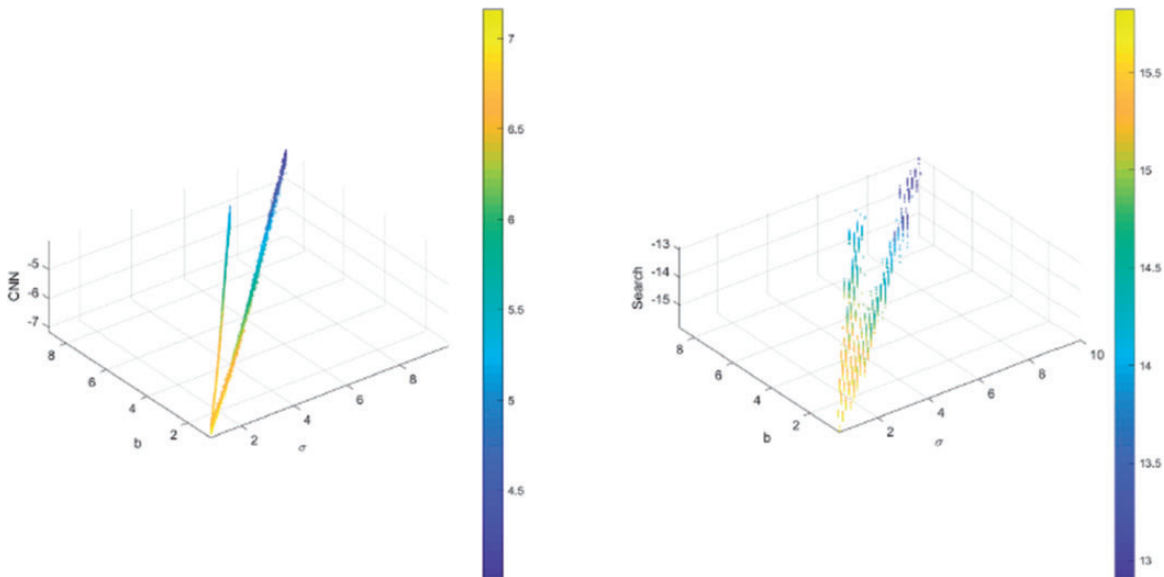


Figure 20. Max element error: 6×6 Dramadah.

metrics and define the maximum element and total element errors with the 2-out-of-3 rules against the Padé and Cayley-Hamilton estimated exponentials. However, the predictions are solely from the networks trained using the Jacobi identity. We have limited ourselves to only predictions based on the four networks for each class that were trained using the Jacobi identity for the main reason that networks trained using the

2-out-of-3 rule work well but are more difficult to use in practice. When solving differential equations involving much larger matrices, the 2-out-of-3 approach based on the Cayley-Hamilton theorem requires solving systems of equations which become too cumbersome to practically solve. Comparison's to Padé approximations are reasonable for large matrices, but then one returns to the problem of comparing only two approximations.

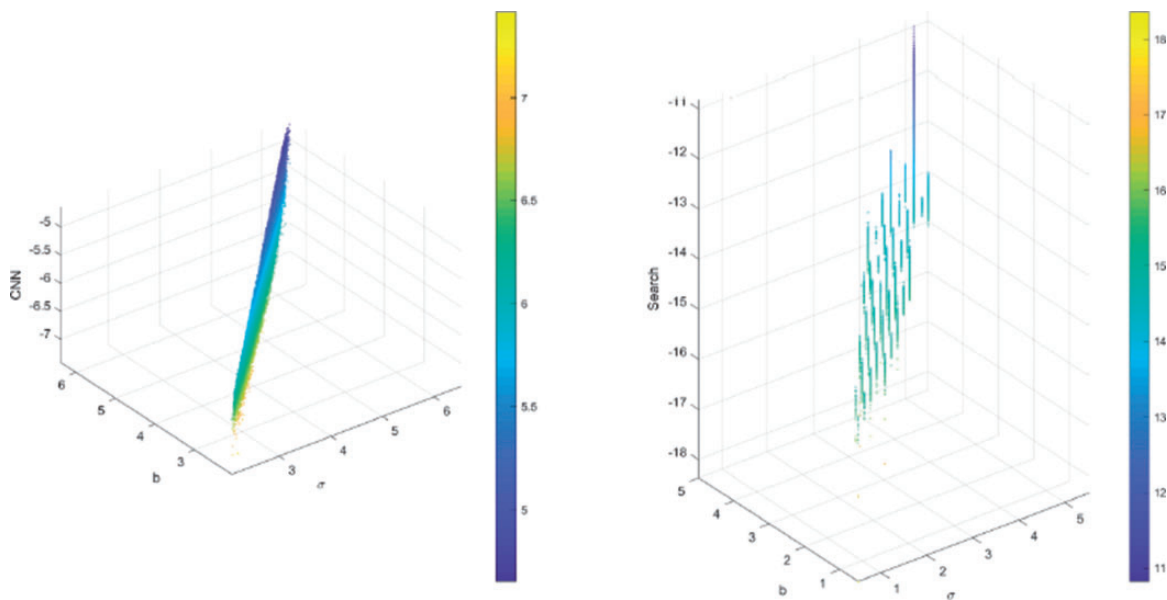


Figure 21. Total elements error: 3×3 skew symmetric matrix.

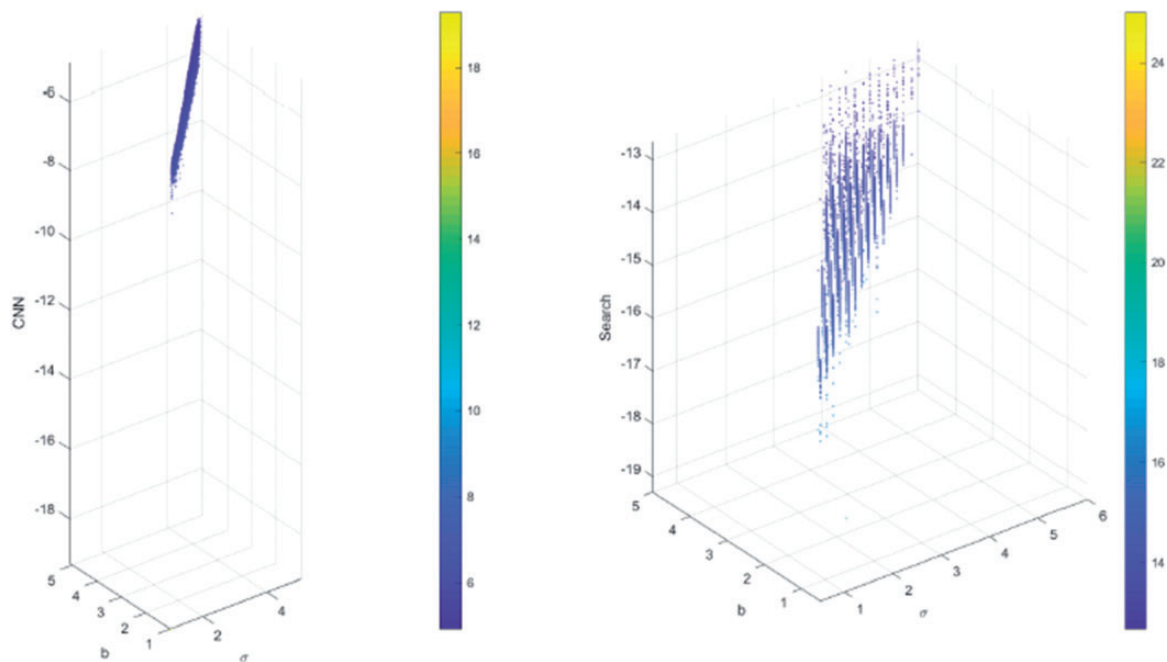


Figure 22. Total elements error: 4×4 quaternions evolution matrix.

In contrast, the self-contained Jacobi identity metric is practically feasible for very large matrices.

What one observes in the following figures is that while they disagree in the exact amount of error, all demonstrate that the approach outlined in this paper is feasible. Specifically, the error as measured with respect to the Jacobi identity can be found in Figures 13 to 16 for the 3,4,5,and 6 square matrices, respectively. The plots are in log base 10 with the left side being the error based on the network (σ, b) and the right side being the actual minimum error from any (σ, b) sampled when creating the database.

Each point in the figures corresponds to a matrix in the complete database. One finds from investigating this figures that the slow full scale minimization of the Jacobi error surface yields excellent results for the Weeks estimated matrix exponential. The neural network derived values, being much faster, are bounded 10^{-6} for the 3×3 and 4×4 matrices. For the 5×5 and 6×6 , the Jacobi errors of the Weeks method estimated exponentials with the simple networks are higher but still follow the correct distribution.

The maximum per element matrix error results are plotted in Figures 17 to 20 for the four matrix families.

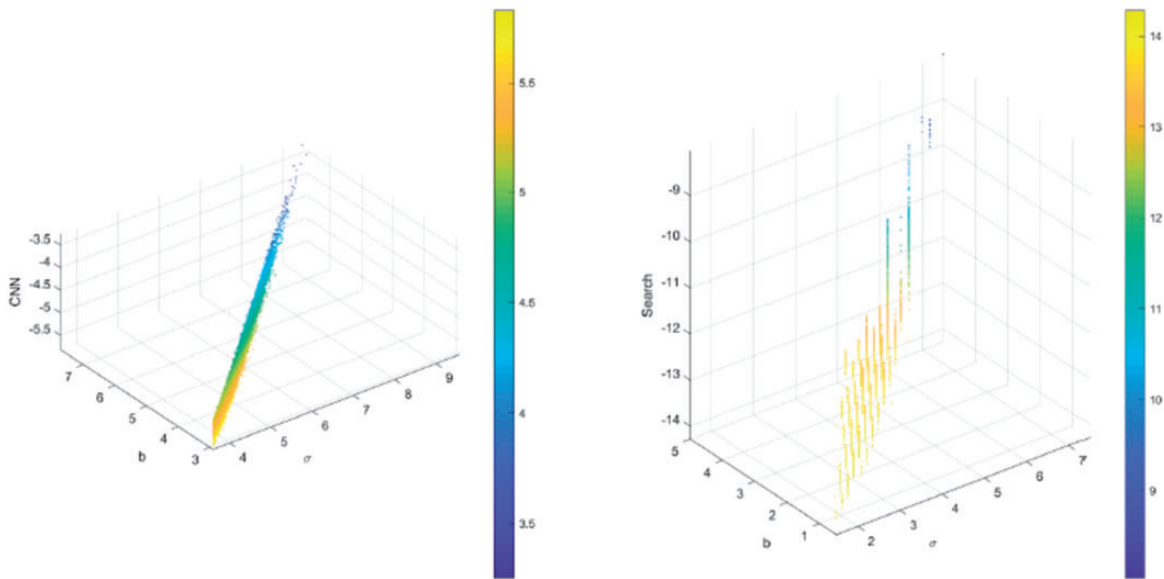


Figure 23. Total elements error: 5×5 random matrix.

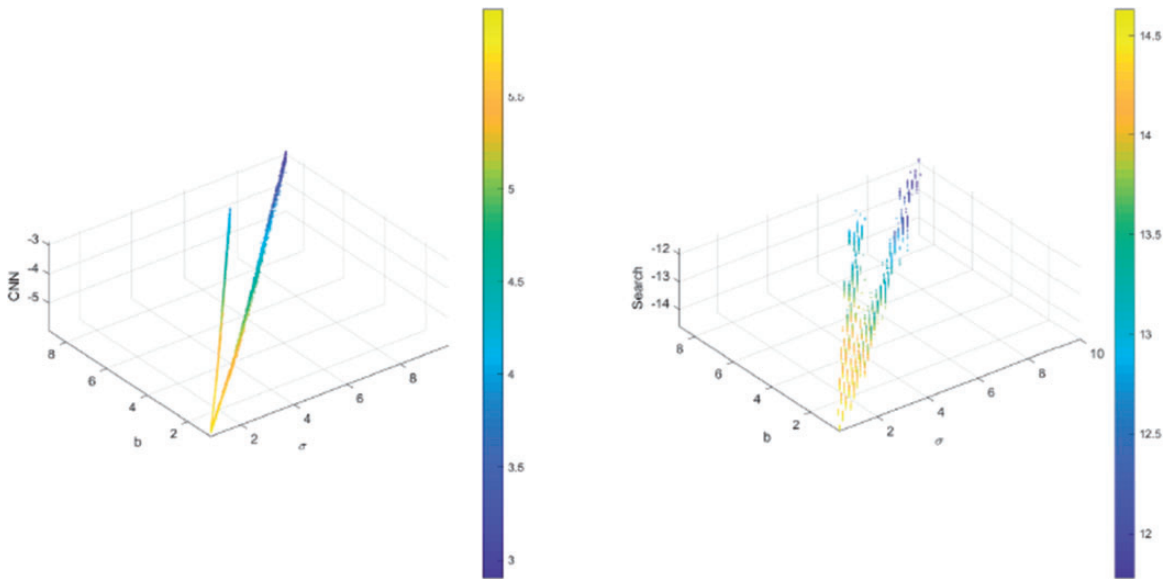


Figure 24. Total elements error: 6×6 Dramadah.

Finally, those based on the total element error are found in Figures 21 to 24. The points across the three metric plots are for the same matrix exponential estimates. The maximum element and total element error calculations also provide a clear picture that the simple neural networks have been able to reasonably capture the basic minimum error surface shape for the four classes.

As a final summary note on the validation results, in Figures 25 and 26 are the 95% confidence intervals for the error metrics recorded in the previous validation figures. That is, the means and confidence intervals are of the matrix exponential error from the neural network predicted parameters and the direct minimization. From these figures one sees that the neural network based approach does yield similar confidence

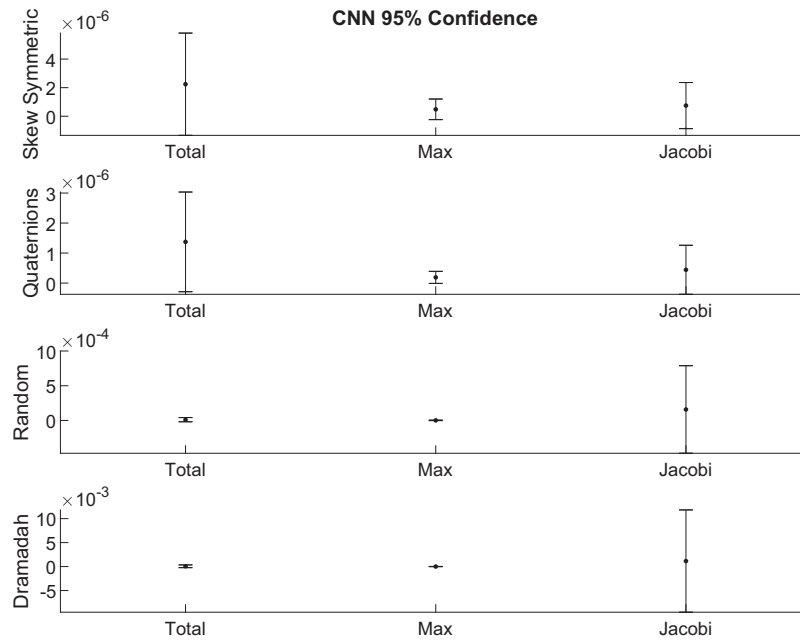


Figure 25. Confidence intervals for network derived errors.

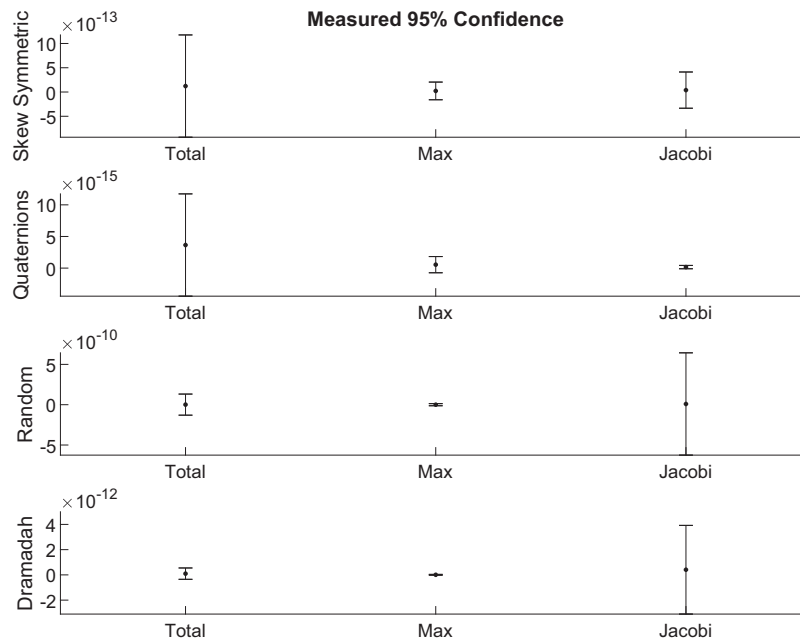


Figure 26. Confidence intervals for measured errors.

intervals relative to the mean error as observed from the full minimization. Another interesting result seen is that the Jacobi error metric has considerably wider confidence intervals for the larger matrices than for the smaller two matrices. Given that the Jacobi metric has potentially broader utility for larger matrices than the other two metrics, this fact about the confidence intervals may be of importance in practical applications or for training a network with larger matrices.

Conclusion

In this paper, we have introduced a machine learning based approach to the problem of selecting optimal parameters in Weeks method for the numerical inversion of the Laplace transform. Specifically, we have demonstrated that it is possible to train a convolutional neural regression network to estimate a (σ, b) pair of parameters which yields an accurate numerical inversion when utilized in Weeks' method. Both the mathematical framework for the approach and concrete results have been presented.

To illustrate, we have focused on the estimation of the matrix exponential e^A by numerically inverting the corresponding resolvent matrix $(sI - A)^{-1}$. Four classes of matrices were studied, the 3×3 skew-symmetric direction cosines evolution matrices, 4×4 quaternion evolution matrices, 5×5 random matrices, and 6×6 Dramadah matrices. For the training and to quantitatively describe the error of the Weeks method estimated matrix exponentials, we considered three metrics. These are the Jacobi identity, a comparison of the maximum per element error, and a comparison of total elements' error. For the last two metrics, it was necessary to define "truth" matrix exponentials based on the Padé rational approximation and from the Cayley-Hamilton theorem.

A particularly useful result which came out of this analysis is the ability to train the Weeks method for the matrix exponential directly from the Jacobi identity. Practically speaking, this allows one to potentially train a neural network with the Weeks method for the exponential of matrices of any size without the need for comparison with a truth value. The small matrices used in this study were chosen to illustrate the approach and because it is relatively straightforward to compute their exponentials via the Cayley-Hamilton formula. Moreover, with the rational Padé approximations and the Cayley-Hamilton expressions for these small matrices, it has been possible to accurately estimate the matrix exponential from the Weeks method as the Möbius transformation parameters (σ, b) were discretized. For many applications, particularly those involving partial differential equations, the

matrices are much larger and yet their exponentials could be computed with the approach presented in this paper.

For future mathematical research, three avenues are suggested. One is a more thorough investigation into different machine learning techniques for Weeks' method optimization. The simple convolutional neural networks presented here demonstrate the efficacy of the approach but the choice of network layers and layer options was not fully optimized. There is a rich diversity of neural network architectures which may be more effective than those utilized here.

A second avenue is to reduce the parameters to only σ and allow b to be fixed. This would simplify the neural network training considerably. To compensate for a fixed b parameter, it may be possible to utilize adaptive integration for the numerical quadrature along the Möbius transform mapped unit circle contour in w . With adaptive quadrature, one might be able to outperform the trapezoidal rule approach used in this paper.⁴⁰

Expanding this approach to other Laplace transform pairs is an obvious third avenue for future work. The accurate solution of the differential equations describing viscoelastic beams,⁴¹ the modeling of fluids,⁴² and the high accuracy propagation of electromagnetic waves¹⁸ are only a few specific examples of difficult problems which may benefit from the machine learning based approach to the Weeks method.

Acknowledgements

This paper would not have been possible without the support from a number of colleagues at Raytheon Missile Systems. The authors especially wish to thank Michael Stinely who has been encouraging from the earliest conceptual phases of this work. Also to thank for their support are Nitesh Shah, Chanon Stewart, and Ross Newton. Ultimately, preparing this paper has required time and understanding from family. P. Kano particularly wishes to thank his son Brennan who challenges him with interesting discussions. The idea to explore the application of the approach in this paper to quaternions arose from one such conversation.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

ORCID iD

Patrick O Kano  <https://orcid.org/0000-0002-3688-0519>

Supplemental material

Supplemental material for this article is available online.

References

- Moler C and Loan CV. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Rev* 2003; 45: 3–49.
- Brio M, Zakharian A and Webb G. *Numerical time-dependent partial differential equations for scientists and engineers*. Cambridge, MA: Elsevier, 2010.
- Kano P and Brio M. Weeks' method for numerical Laplace transform inversion with GPU acceleration. Available on the MATLAB file exchange, 2011. <https://www.mathworks.com/matlabcentral/fileexchange/30965-weeks-method-for-numerical-laplace-transform-inversion-with-gpu-acceleration>
- Abate J, Choudhury G and Whitt W. On the Laguerre method for numerically inverting Laplace transforms. *Informs J Comput* 1996; 8: 413–427.
- Davies B. *Integral transforms and their applications*. Vol. 3. New York: Springer, 2002.
- Kuhlman K. Review of inverse Laplace transform algorithms for Laplace-space numerical approaches. *Numer Algor* 2013; 63: 339–355.
- Weideman J. Algorithms for parameter selection in the weeks method for inverting the Laplace transform. *SIAM J Sci Comput* 1999; 21: 111–128.
- Weideman J. Optimizing Talbot's contours for the inversion of the Laplace transform. *SIAM J Numer Anal* 2006; 44: 2342–2362.
- Weideman J and Trefethen LN. Parabolic and hyperbolic contours for computing the Bromwich integral. *Math Comp* 2007; 76: 1341–1356.
- Weber H. Numerical inversion/computation of the Laplace transform, <https://www.cs.hs-rm.de/weber/lapinv/lapinv.htm> (2020, accessed 22 February 2021).
- Talbot A. The accurate numerical inversion of Laplace transforms. *IMA J Appl Math* 1979; 23: 97–120.
- Murli A and Rizzardi M. Algorithm 682. Talbot's method for the Laplace inversion problem. *ACM Trans Math Softw* 1990; 16: 158–168.
- Kano P, Brio M, Dostert P, et al. Dempster-Shafer evidential theory for the automated selection of parameters for Talbot's method contours and application to matrix exponentiation. *Comput Math Appl* 2012; 63: 1519–1535.
- Defreitas CL and Kane SJ. The noise handling properties of the Talbot algorithm for numerically inverting the Laplace transform. *J Algorithm Comput Technol* 2018; 13: 1–14.
- De Hoog FR, Knight JH and Stokes AN. An improved method for numerical inversion of Laplace transforms. *SIAM J Sci Stat Comput* 1982; 3: 357–366.
- Abate J and Valkó P. Multi-precision Laplace transform inversion. *Int J Numer Meth Eng* 2004; 60: 979–993.
- Valkó P and Abate J. *Numerical inversion of Laplace transform with multiple precision using the complex domain*. Mathematica Information Center: Mathsource, <http://library.wolfram.com/infocenter/MathSource/5026/> (2003, accessed 22 February 2021).
- Kano P and Brio M. Application of post's formula to optical pulse propagation in dispersive media. *Comput Math Appl* 2010; 59: 629–650.
- Arnold DN and Rogness J. Möbius transformations revealed. *Notices AMS* 2008; 55: 1226–1231.
- Olsen J. The geometry of moebius transformations, <http://johnno.dk/mathematics/moebius.pdf> (2010, accessed 22 February 2021).
- Jayasundera D, Kano P, Stinely M, et al. Conformal mappings of complex boundaries. Raytheon unpublished unclassified internal report, 2017.
- Arfken GB and Weber HJ. *Mathematical methods for physicists*. 5th ed. Cambridge, MA: Harcourt/Academic Press, 2001.
- Press WH, Vetterling WT, Teukolsky SA, et al. *Numerical recipes in C: The art of scientific computing*. 2nd ed. Cambridge: Cambridge University Press, 1992.
- Theodoridis S and Koutroumbas K. *Pattern recognition*. Orlando, FL: Academic Press, 2009.
- The Mathworks, Inc. *Mastering machine learning*. Electronic book with Matlab source code examples. 2018. <https://www.mathworks.com/campaigns/offers/machine-learning-with-matlab.html>
- Wood R. Cognitive systems: advances in processing capabilities will revolutionize system performance. *Technology Today* 2005; 3, www.raytheon.com/technology_today/archive/2005_Issue3.pdf (accessed 22 February 2021).
- US Department of Defense. Summary of the 2018 department of defense artificial intelligence strategy. Unclassified, publically available, report, 2018.
- Lloyd S, Irani R and Ahmadi M. Using neural networks for fast numerical integration and optimization. *IEEE Access* 2020; 8: 84519–84531.
- MATLAB. Matlab machine learning toolbox, www.mathworks.com/products/deep-learning.html (2019, accessed 22 February 2021).
- Kano P, Brio M and Moloney J. Application of weeks method for the numerical inversion of the Laplace transform to the matrix exponential. *Comm Math Sci* 2005; 3: 335–372.
- Kano P and Brio M. C++/CUDA implementation of the weeks method for numerical Laplace transform inversion, www.acunum.com (2011, accessed 22 February 2021).
- Davies P and Higham N. A Schur-Parlett algorithm for computing matrix functions. *SIAM J Matrix Anal Appl* 2003; 25: 464–485.
- Van Loan C. The sensitivity of the matrix exponential. *SIAM J Numer Anal* 1977; 14: 971–981.
- Rowell D. *The Cayley-Hamilton theorem and the matrix exponential*. 2004. Wolfram Research, Inc.
- Al-Mohy AH and Higham NJ. A new scaling and squaring algorithm for the matrix exponential. *SIAM J Matrix Anal Appl* 2010; 31: 970–989.
- Mathematica 11.3, www.mathematica.com (2019, accessed 22 February 2021).

37. Ojanen H. Mathematica expression in Matlab m-file converter, library.wolfram.com/infocenter/MathSource/577/ (1999, accessed 22 February 2021).
38. Yershova A, Jain S, LaValle S, et al. Generating uniform incremental grids on $so(3)$ using the HOPF fibration. *Int J Rob Res* 2010; 29: 801–812.
39. Glazier AT. Understanding transformations and rotations. Online lecture notes to Raytheon Course SYS1031A, 2010.
40. Swierczek S, Brio M and Kano P. 2018. *Adaptive integration in the weeks method for numerical Laplace transform inversion*. University of Arizona, Independent study final report.
41. Kano P. An accelerated weeks method for numerical Laplace transform inversion: application to viscoelastic beam modeling, S0415-Accelerated-Weeks-Method-for-Numerical-Laplace-Transform-Inversion.pdf (2012, accessed 22 February 2021).
42. Cousins W, Gremaud PA and Tartakovsky DM. A new physiological boundary condition for hemodynamics. *SIAM J Appl Math* 2013; 73: 1203–1223.