Theory of Connections Applied to Support Vector Machines to Solve Differential Equations

Carl Leake, Hunter Johnston, Lidia Smith, and Daniele Mortari, Fellow, IEEE

Abstract—Differential equations are used as numerical models to describe physical phenomena throughout the field of engineering and science, including heat and fluid flow, structural bending, and systems dynamics. Although there are many other techniques for finding approximate solutions to these equations, this paper looks to compare the application of the Theory of Connections (ToC) with one based on Support Vector Machines (SVM). The ToC method uses a constrained expression (an expression that always satisfies the differential equation constraints), which transforms the process of solving a differential equation into an unconstrained problem, and is ultimately solved via least-squares. In addition to individual analysis, the two methods are merged into a new methodology, called constrained SMVs (CSVM), by incorporating the SVM method into the ToC framework to solve unconstrained problems. Numerical tests are conducted on three sample problems: one first order linear ODEs, one first order non-linear ODE, and one second order linear ODE. Using the SVM method as a benchmark, a speed comparison is made for all the problems by timing the training period, and an accuracy comparison is made using the maximum error and mean-squared error on the training and test sets. In general, ToC is shown to be slightly faster (by an order of magnitude or less) and more accurate (by multiple orders of magnitude) over the SVM and CSVM approaches.

1 INTRODUCTION

Differential equations (DE) and their solutions remain an important topic in science and engineering. The solutions drive the design of predictive systems models and optimization tools. Currently, these equations are solved by a variety of existing approaches with the most popular based on the Runge-Kutta family [1]. Other methods include those which leverage low-order Taylor expansions, namely Gauss-Jackson [2] and Chebyshev-Picard iteration [3], [4], [5] which have proven to be highly effective. More recently developed techniques are based on spectral collocation methods [6]. This approach discretizes the domain about collocation points, and the solution of the DE is expressed by a sum of “basis” functions with unknown coefficients that are approximated in order to satisfy the DE as closely as possible. Yet, in order to incorporate boundary conditions, one or more equations must be replaced by equations to enforce the constraints.

The Theory of Connections (ToC) is a new technique that first analytically derives a constraint expression which satisfies the problem’s constraints exactly while maintaining a function that can be freely chosen [7]. This process transforms the DE into an unconstrained optimization problem where the free function is used to search for the solution of the DE. Prior studies [8], [9], [10], have defined this free function as a summation of basis functions; more specifically, Chebyshev polynomials. Motivated by recent results that solve ordinary DEs with support vector machines (SVMs) [11], this article explores the ability to solve DEs by leveraging the flexibility of the ToC framework and incorporating an SVM as the free function. The SVM model for function estimation [12] also uses a linear combination of functions that depend on the input data points. While in the first uses of SVMs the prediction for an output value was made based on a linear combination of the inputs $x_i$, a later technique uses a mapping of the inputs to feature space, and the model SVM becomes a linear combination of feature functions $\varphi(x)$. Further, with the kernel trick, the function to be evaluated is determined based on a linear combination of kernel functions, with Gaussian kernels being a popular choice.

This article compares this combined method, referred to hereafter as CSVM for constrained SVMs, to vanilla versions of ToC [8], [9] and SVM [11] over a variety of DEs. In all cases, the vanilla version of ToC outperforms both the SVM and the CSVM methods in terms of accuracy and speed. However, the CSVM method is easily scalable to higher dimensional problems, PDEs for example, whereas the vanilla ToC is not. In addition, the CSVM method satisfies the boundary conditions of the problem exactly, whereas the vanilla SVM method solves the boundary condition with the same accuracy as the remainder of the data points in the problem. Thus, the CSVM method combines the scalability of the SVM method with the exact boundary condition satisfaction of ToC. Of course, SVMs are not the only machine learning algorithm that can be substituted as the free function in ToC. Any machine learning algorithm that is differentiable up to the order of the DE can be seamlessly incorporated in ToC. Future work will leverage this benefit to analyze the ability to solve DEs via neural networks.
2 BACKGROUND ON THE Theory of Connections

The Theory of Connections (ToC), as established in [7], is a generalized interpolation method. This technique allows for the construction of all functions satisfying k linear constraints with the general form,

\[ y(t) = g(t) + \sum_{i=1}^{k} \eta_i p_i(t), \]

where \( g(t) \) represents a “freely chosen” function, \( \eta_i \) are the coefficients derived from the \( k \) linear constraints, and \( p_i(t) \) are user selected functions that must be linearly independent from \( g(t) \). Using this approach, least-squares solutions for initial-value (IVP), boundary-value (BVP), and multi-value (MVP) problems on both linear [8] and nonlinear DEs have been solved with machine error accuracy. Additionally, this method extends to relative, integral, and infinite constraints [10], establishing a unified framework to solve DEs. This method is best understood through example. Let us consider a general DE subject to initial value constraints on the function and its derivative such that,

\[ \mathcal{L}(\dot{y}, y, t) = 0 \quad \text{subject to:} \quad \begin{cases} y(t_0) = y_0 \\ \dot{y}(t_0) &= \dot{y}_0 \end{cases} \]

To solve this DE we start with Eq. (1) where \( k = 2 \) and \( p_i(t) \) is chosen to be \( p_1(t) = 1 \) and \( p_2(t) = t \). Using these parameters, Eq. (1) becomes,

\[ y(t) = g(t) + \eta_1 + \eta_2 \ t. \] (2)

To apply the constraints, this equation is evaluated at the constraint locations producing two equations:

\[ \begin{cases} y(t_0) = g(t_0) + \eta_1 + \eta_2 \ t_0 \\ \dot{y}(t_0) = \dot{g}(t_0) + \eta_2 \end{cases} \]

where the only unknowns are \( \eta_1 \) and \( \eta_2 \). With these equations a linear system can be constructed,

\[ \begin{pmatrix} y_0 - y_0 \\ y_0 - y_0 \end{pmatrix} = \begin{bmatrix} 1 & t_0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix}, \]

and the two unknowns are solved through matrix inversion,

\[ \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} = \begin{bmatrix} 1 & -t_0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_0 - y_0 \\ y_0 - y_0 \end{bmatrix} \]

leading to the \( \eta \) coefficients of

\[ \begin{cases} \eta_1 = (y_0 - y_0) - t_0 (\dot{y}_0 - \dot{y}_0) \\ \eta_2 = y_0 - y_0 \end{cases} \]

These two terms are now plugged into Eq. (2) to form the final expression for \( y(t) \) that always satisfies the constraints regardless of the function chosen for \( g(t) \),

\[ y(t) = g(t) + (y_0 - y_0) + (t - t_0)(\dot{y}_0 - \dot{y}_0), \]

known as the constrained expression. Analyzing this equation, it can be seen that if \( t = t_0 \) then the equation reduces to \( y(t_0) = y_0 \), and if the derivative is taken and also evaluated at \( t = t_0 \) then it becomes \( \dot{y}(t_0) = \dot{y}_0 \). Now, with the constraints fully incorporated in Eq. (3) and \( g(t) \) representing all possible functions passing through those constraints, the solution to the DE is found by “searching” for \( g(t) \). In prior work [3, 11], this was done by representing \( g(t) \) as a basis of Chebyshev orthogonal polynomials,

\[ y(t) = \xi^i h(x) \quad \text{where} \quad x = x(t), \]

where \( \xi \) is the unknown vector of coefficients to be solved and \( h(x) \) represents a vector of \( m \) basis functions. In general the independent variable \( t \in [t_0, t_f] \) does not coincide with the range of Chebyshev polynomials which are strictly defined on the interval \( x \in [-1, +1] \). Therefore, a change of variable is needed in order to correctly incorporate these basis functions. This can be done by linearly mapping \( x \) to \( t \),

\[ x = x_0 + \frac{x_f - x_0}{t_f - t_0} (t - t_0), \]

\[ t = t_0 + \frac{t_f - t_0}{x_f - x_0} (x - x_0). \]

It follows that the first derivative of \( g(t) \) becomes,

\[ \frac{dq}{dt} = \frac{d^2 \xi h(x)}{dx^2} \frac{dx}{dt}, \]

where it can be seen that \( \frac{dx}{dt} = \frac{x_f - x_0}{t_f - t_0} = c \) from the variable mapping. This extends to all subsequent derivatives by the equation,

\[ \frac{d^k q}{dt^k} = c^k \xi^\top \frac{d^k h(x)}{dx^k}. \]

Using this, the constrained expression defined in Eq. (5) and its derivatives become,

\[ \begin{cases} y(t) = \xi^\top h(x) + (y_0 - \xi^\top h_0) \\ \dot{y}(t) = c \xi^\top h'(x) + (\dot{y}_0 - c \xi^\top h'_0) \\ a(t) = c^2 \xi^\top h''(x) \end{cases} \]

which can be written in the form,

\[ \begin{cases} y(t) = a^\top x \xi + b(t) \\ \dot{y}(t) = a^\top \dot{x} \xi + \dot{b}(t) \\ \ddot{y}(t) = a^{\prime\prime\top} x \xi + \ddot{b}(t) \end{cases} \]

where \( a(x) \) is zero when evaluated at the boundary conditions and \( b(t) \) is equal to the constraints. The constrained expression and its derivatives can be plugged into the DE reducing the equation to an expression with the only unknown being the vector \( \xi \).

\[ \mathcal{L}(a^{\prime\prime\top} x + \ddot{b}, a^\top x + \dot{b}, a^\top \xi + b, t) = 0 \]

The resulting equation is discretized over a set of \( N \) values of \( x \) (and inherently \( t \)). In expressing the solution of a DE as a basis of Chebyshev orthogonal polynomials, the optimal point distribution is provided by collocation points [13], [14], defined as,

\[ x_i = -\cos \left( \frac{i \pi}{N} \right) \quad \text{for} \quad i = 0, 1, \ldots, N, \]

where \( N \) is the number of point in the interval. As compared to the uniform distribution, the collocation point distribution is the most efficient way to numerically represent Chebyshev polynomials. By discretizing the DE over the domain of \( x \) creates a system of equations that is linear with
where, $\xi$. Then, Eq. (6) can be rewritten in the form,

$$\forall \xi = b.$$  

(7)

In prior work [8], [9], a scaled QR decomposition approach was determined to be optimal, and thus, Eq. (7) can be solved by,

$$B = A S = Q R \rightarrow \xi = S R^{-1} Q^T b.$$ 

In the case of a nonlinear DE an iterative least-squares approach is needed, which is developed in Ref. [9]. The equation is solved by first using an initial guess for $\xi_0$, obtained by fitting an initial estimated solution of the DE, and then starting the nonlinear Newton iteration whose k-th iteration is,

$$L_k + \left[ \frac{\partial L}{\partial \xi} \right]_k (\xi_{k+1} - \xi_k) \approx 0,$$

where,

$$\frac{dL}{d\xi} = \sum_{i=0}^{n} \left( \frac{\partial^1 L}{\partial \xi_{(i)}} \cdot \frac{\partial y_{(i)}}{\partial \xi} \right) \quad \text{and} \quad \frac{\partial y_{(i)}}{\partial \xi} = a_{(i)},$$

where $L_k$ is the loss function defined by the residuals of the DE which are specified using $\xi_k$ at all values of the vector $x$. The convergence is obtained when the $L_2$ norm, $L_2[f(x, \xi_k)] < \varepsilon$, where $\varepsilon$ is a given tolerance. In the end, the final value of $\xi$ constitutes the solution of the DE. This $\xi$ vector is plugged back into the constrained expression Eq. (4) providing an analytical solution of the DE that always satisfies the constraint conditions. This feature allows for analytical differentiation and integration of the solution.

### 3 An Overview of SVMs

The technique of Support Vector Machines (SVMs) was originally introduced to solve classification problems (as pattern recognition [12]). A classification problem consists of determining if a given input, $x$, belongs to one of the two possible classes. The proposed solution was to find a decision boundary surface that separates the two classes. The equation of the separating boundary could depend only on a few input vectors called the support vectors.

The first algorithm that learned from examples to perform classification was Rosenblatt’s perceptron introduced in 1957. One of the ideas introduced with the perceptron was to use a mapping of the inputs in 1957. One of the ideas introduced with the perceptron form classification was Rosenblatt’s perceptron introduced on a few input vectors called the decision boundary surface that separates the two classes. The proposed solution was to find a pattern recognition [12]). A classification problem consists

The technique of Support Vector Machines (SVMs) was built using a linear combination of feature vectors with the weights $w$ to be determined such that the model

$$g(x) = \text{sign}\left\{ \sum_{k=1}^{n} w_k \varphi_k(x) \right\}$$

performs the correct classification for new inputs.

Later the ideas were extended to function estimation. A function that is linear in parameters is used as a model,

$$f(x) = \sum_{k=1}^{n} w_k \varphi_k(x) = w^T \varphi(x)$$

where $w^T = \{w_1, \ldots w_n\}$ and $\varphi(x) = \{\varphi_1, \ldots, \varphi_n\}$.

The perceptron algorithm updated the weights iteratively, and found a separating boundary, but not an optimal one. The initial SVM algorithm introduced in the 1960’s was looking for an optimal boundary, but the search for a separating boundary in the feature space was not used until the 1980s. Vapnik in [12] formulates the goal of the statistical learning models as finding the function that minimizes the empirical risk on the set of functions from the class chosen as the model,

$$R_{emp} = \frac{1}{n} \sum_{i=1}^{n} \left| y_i - f(x_i) \right|^2.$$ 

The empirical risk defined above is the Mean Squared Error (MSE) for the function $f$ that tries to estimate the output $y$ given input-output data pairs $(x_1, y_1), \ldots, (x_n, y_n)$ like in linear regression. Minimizing the empirical risk is an optimization problem, for which one of the popular solutions is to use a gradient descent algorithm, updating the parameters (weights) of the model iteratively.

In 1959, the LMS (least mean squares) algorithm was introduced as a simple way of training linear adaptive systems with mean square error minimization. LMS uses steepest-descent algorithm to update the weight vector,

$$w_{n+1} = w_n - h \nabla_x e^2_n,$$

where $w_n$ is the weight vector, $h$ is the step size, $x_n$ is the input vector and $e_n = y_n - f_n$.

Without mapping to feature space, working with the original input vectors $x$ multiplied by the weights $w$ the function to minimize is written as,

$$J_w = \sum_{i=1}^{n} \left| y_i - f(x_i) \right|^2.$$ 

In this context the function to be minimized is called a cost function and it gives rise to the usual least squares regression model which can be solved either by an iterative gradient descent algorithm or by using the normal equations, which require finding the inverse of the matrix $X^T X$ where $X$ is the matrix formed by using the input vectors as columns. The regression problem also has a probabilistic interpretation. Minimizing the empirical risk also corresponds to finding the Maximum Likelihood (ML) parameters when measurements are assumed to be corrupted by Gaussian noise

$$y_i = f(x_i) + \epsilon_i \quad \text{where} \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2).$$

Minimizing the expected square error is equivalent to minimizing the logarithm of the likelihood of the data, which is a product of the probabilities of observing the given outputs for the selected values of the parameters. For normally distributed errors, the probabilities are proportional to exponential functions with the negative of the squared error as the argument. The density of the normal distribution gives

$$p(\epsilon_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{\epsilon_i^2}{2\sigma^2} \right)$$

from which we can write the conditional probabilities

$$p(y_i|x_i) = \frac{1}{\sqrt{2\pi\sigma}} \exp \left( -\frac{(y_i - f(x_i))^2}{2\sigma^2} \right)$$
The notation, terms of the partial derivatives of the kernel. We will use Dot products involving derivatives can be expressed in the computation, and thus the dot product kernel can be used in \[\varphi\] rating hyperplanes, a regularization term is added to the least squares problem \([15]\). In the SVM model for classification an additional term, \(b\), called bias, is added to the linear combination of features \(\varphi(x)\) with weights \(w_i\) leading to the model,

\[\hat{y}(x) = \sum_{i=1}^{N} w_i \varphi_i(x) + b = w^T \varphi(t) + b. \tag{8}\]

Assuming that the training data is separable by a linear decision boundary, a separating hyperplane with equation \(w^T \varphi(x) + b = 0\) is sought. The parameters are rescaled such that closest training point, to the hyperplane, let’s say \((x_k, y_k)\), is on a parallel surface with equation \(w^T \varphi(x) + b = 1\). By using the formula for orthogonal projection, if \(x\) satisfies the equation of the hyperplane then the distance to the origin of the space has magnitude given by \(w^T x / w^T w\). It follows that the distance between the two hyperplanes, called the “separating margin” is \(1 / w^T w\). Thus to find the largest separating margin, one needs to minimize \(w^T w\).

Instead of just maximizing the margin between separating hyperplanes, a regularization term is added to the objective function,

\[\min \frac{1}{2} (w^T w + \gamma e^T e).\]

### 3.1 The kernel trick

In Eq. \([8]\) the function \(\varphi(t)\) always shows up in a dot product, and thus the dot product kernel can be used in the computation,

\[\varphi(u)^T \varphi(v) = K(u, v).\]

Dot products involving derivatives can be expressed in terms of the partial derivatives of the kernel. We will use the notation,

\[
\begin{align*}
\varphi'(u)^T \varphi(v) &= \frac{\partial}{\partial u} K(u, v) =: K_{10}(u, v) \\
\varphi(u)^T \varphi'(v) &= \frac{\partial}{\partial v} K(u, v) =: K_{01}(u, v) \\
\varphi'(u)^T \varphi'(v) &= \frac{\partial^2}{\partial u \partial v} K(u, v) =: K_{11}(u, v),
\end{align*}
\]

where \(K\) is a kernel, popularly defined as the RBF (radial basis function) kernel. This is also known as Gaussian kernel, and has the formula,

\[K(u, v) = \exp \left(-\frac{(u - v)^2}{\sigma^2}\right).
\]

### 3.2 Using SVMs to Solve Differential Equations

We follow the method of solving DEs using RBF kernels proposed in \([11]\). As an example, we take a first order linear initial value problem,

\[y' - p(t)y = r(t), \quad \text{subject to:} \quad y(t_0) = y_0,\]

to be solved on the interval \([t_0, t_f]\). The domain is partitioned into \(N\) sub-intervals using grid points \(t_0, t_1, \ldots, t_N = t_f\), which, from a machine learning perspective, represents the training points. The model given in Eq. \([6]\), is proposed for the solution \(y(t)\), where the functions \(\varphi(t)\) have the property that,

\[K(t_i, t) = \varphi(t_i)^T \varphi(t),\]

and the parameter \(\sigma\) is a value that is learned in the training, together with the coefficients \(w_i, i = 1, \ldots, N\). Note that the number of coefficients \(w_i\) equals the number of grid points \(t_i\), and thus the system of equations used to solve for the coefficient is a square matrix. Let \(e\) be the vectors of residuals obtained when we use the model solution \(\hat{y}(t)\) in the DE, that is, \(e_i\) is the amount by which \(\hat{y}(t_i)\) fails to satisfy the DE,

\[\hat{y}'(t_i) - p(t_i)\hat{y}(t_i) - r(t_i) = e_i,
\]

which gives,

\[w^T \varphi(t_i) = p(t_i)[w^T \varphi(t_i) + b] + r(t_i) + e_i,
\]

and for the initial condition, it is desired that,

\[w^T \varphi(t_0) + b = y_0,
\]

is satisfied exactly. In order to have the model close to the exact solution, the sum of the squares of the residuals, \(e^T e\), is to be minimized. This expression can be viewed as a regularization term added to the objective of maximizing the margin between separating hyperplanes. The problem is formulated as an optimization problem with constraints:

\[\min \frac{1}{2} (w^T w + \gamma e^T e) \quad \text{subject to:}
\]

\[
\begin{align*}
\begin{cases}
w^T \varphi'(t_i) - p(t_i) [w^T \varphi(t_i) + b] - r(t_i) - e_i = 0 \\
w^T \varphi(t_0) + b - y_0 = 0.
\end{cases}
\]

Using the method of Lagrange multipliers, a multiplier (factor) is introduced for each constraint, introducing a penalty term to me minimized, leading to Eq. \([9]\). The values where the gradient of \(L\) is zero give candidates for the minimum such that,

\[
\begin{align*}
\frac{\partial L}{\partial w} &= 0 \Rightarrow w = \sum_{i=1}^{N} \alpha_i [\varphi'(t_i) - p(t_i) \varphi(t_i)] + \beta \varphi(t_0) \\
\frac{\partial L}{\partial e_i} &= 0 \Rightarrow \gamma e_i = -\alpha_i \\
\frac{\partial L}{\partial b} &= 0 \Rightarrow -\frac{N}{\alpha_i q(t_i) - \beta} \\
\frac{\partial L}{\partial \alpha_i} &= 0 \Rightarrow w^T \varphi(t_i) - p(t_i) [w^T \varphi(t_i) + b] - g(t_i) - e_i = 0 \\
\frac{\partial L}{\partial \beta} &= 0 \Rightarrow w^T \varphi(t_0) + b - y_0.
\end{align*}
\]
\[ \mathcal{L}(w, e, \alpha, \beta, b) = \frac{1}{2} (w^T w + \gamma e^T e) - \sum_{i=1}^{N} \alpha_i [w^T \varphi'(t_i) - p(t_i) (w^T \varphi(t_i) + b) - r(t_i) - e_i] - \beta (w^T \varphi(t_0) + b - y_0) \] (9)

Note that the conditions found by differentiating \( \mathcal{L} \) with respect to \( \alpha_i \) and \( \beta \) are simply the constraint conditions, while the remaining conditions are the standard Lagrange multiplier conditions, that the gradient of the function to be minimized is a linear combination of the gradients of the constraints. Using,

\[ w = \sum_{j=1}^{N} \alpha_i [\varphi'(t_j) - p(t_j) \varphi(t_j)] + \beta \varphi(t_0), \]

we obtain a new formulation of the approximate solution given by Eq. (10), where the inner products can be expressed in terms of the kernel,

\[ K(t_i, t_j) = \varphi(t_i)^T \varphi(t_j) = \exp\left(-\frac{(t_i - t_j)^2}{\sigma^2}\right), \]

and its partial derivatives,

\[ K_1(t_i, t_j) = \varphi'(t_i)^T \varphi(t_j) = \exp\left(-\frac{(t_i - t_j)^2}{\sigma^2}\right) \]
\[ = -2(t_i - t_j) \frac{1}{\sigma^2} \exp\left(-\frac{(t_i - t_j)^2}{\sigma^2}\right) \]
\[ K_1(t_i, t_j) = \varphi(t_i)^T \varphi'(t_j) = \frac{2(t_i - t_j)}{\sigma^2} \exp\left(-\frac{(t_i - t_j)^2}{\sigma^2}\right) \]
\[ K_1(t_i, t_j) = \varphi'(t_i)^T \varphi'(t_j) = \frac{4(t_i - t_j)^2}{\sigma^4} \exp\left(-\frac{(t_i - t_j)^2}{\sigma^2}\right) \]
\[ K_1(t_i, t_j) = \varphi''(t_i)^T \varphi(t_j) = \frac{2}{\sigma^2} - \frac{4(t_i - t_j)^2}{\sigma^4} \exp\left(-\frac{(t_i - t_j)^2}{\sigma^2}\right) \]

To find the actual solution one needs to train and choose a good value for \( \sigma \) and solve for the coefficients \( \alpha_i, \beta, \) and \( b. \) Substituting the results from the equations on the gradient of \( \mathcal{L} \) with respect to \( w \) and \( e_i \) into the equation on the gradient of \( \mathcal{L} \) with respect to \( \alpha_i \) we obtain Eq. (11). The last condition, the constraint \( w^T \varphi(t_0) + b - y_0 = 0 \) becomes Eq. (12). Replacing all the inner products by the corresponding kernel values, we have a system with the same number of equations as unknowns. The third condition,

\[ \beta = \sum_{j=1}^{N} \alpha_j p(t_j), \]

could also be used to reduce the number of unknowns in the system.

### 3.3 Constrained SVM (CSVM) Technique

In the Theory of Connections (ToC) [7] the general constrained expression can be written for an initial value constraint by,

\[ y(t) = g(t) + (y_0 - y_0), \]

where \( g(t) \) is a “freely chosen” function. In prior studies [8], [9], this free function was defined by a set of orthogonal basis functions, but this function can also be defined using SVMs by,

\[ g(t) = \sum_{i=1}^{N} w_i \varphi_i(t) = w^T \varphi(t), \]

where \( y_0 \) becomes,

\[ g(t_0) = \sum_{i=1}^{N} w_i \varphi_i(t_0) = w^T \varphi(t_0). \]

This leads to the equation,

\[ y(t) = w^T [\varphi(t) - \varphi(t_0)] + y_0, \]

such that the initial value constraint is always satisfied regardless of the value of \( w \) and \( \varphi(t) \). Through this process, the constraints only remain on the residuals and the problem becomes,

\[ \min \frac{1}{2} (w^T w + \gamma e^T e) \quad \text{subject to:} \]
\[ w^T \varphi(t_i) - p(t_i) [w^T \varphi(t_i) - w^T \varphi(t_0) + y_0] - r(t_i) - e_i = 0. \]

Again, using the method of Lagrange multipliers, a term is introduced for each of the constraints, as shown in Eq. (13), and the values where the gradient of \( \mathcal{L} \) is zero give candidates for the minimum,

\[ \frac{\partial \mathcal{L}}{\partial w} = 0 \to w = \sum_{i=1}^{N} \alpha_i [\varphi'(t_i) - p(t_i) (\varphi(t_i) - \varphi(t_0))] \]
\[ \frac{\partial \mathcal{L}}{\partial e_i} = 0 \to e_i = -\frac{\alpha_i}{\gamma} \]
\[ \frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \to 0 = w^T \varphi'(t_i) - p(t_i) (\varphi(t_i) - \varphi(t_0)) + y_0 - r(t_i) - e_i. \]

Using,

\[ w = \sum_{j=1}^{N} \alpha_j [\varphi'(t_j) - p(t_j) (\varphi(t_j) - \varphi(t_0))], \]

we obtain a new formulation of the approximate solution,

\[ y(t) = w^T (\varphi(t) - \varphi(t_0)) + y_0, \]

that can be expressed in terms of the kernel \( K(u, v) = \varphi(u)^T \varphi(v) \) and its derivatives. Combining the three equations obtained when taking the gradients of \( \mathcal{L} \), we can obtain a linear system with unknowns \( \alpha_i \),

\[ \sum_{j=1}^{N} M_{ij} \alpha_j = r(t_i) + p(t_i)y_0. \]

The coefficient matrix is given by Eq. (14), where \( K(t_0, t_0) = 1 \), and we use the notation given in Eq. (15). Finally, in terms of the kernel matrix,

\[ y(t_i) = \sum_{j=1}^{N} \alpha_j K_p(i, j) + y_0. \]
\[ \dot{y}(t) = w^t \varphi(t) + b = \sum_{j=1}^{N} \alpha_j [\varphi'(t_j)^T \varphi(t) - p(t_j) \varphi(t_j)^T \varphi(t)] + \beta \varphi(t_0)^T \varphi(t) + b \]  

(10)

\[ \sum_{j=1}^{N} \alpha_j [\varphi'(t_j)^T \varphi'(t_i) - p(t_j) \varphi(t_j)^T \varphi'(t_i)] + \beta \varphi(t_0)^T \varphi'(t_i) - p(t_i) \sum_{j=1}^{N} \alpha_j [\varphi'(t_j)^T \varphi(t_j) - p(t_j) \varphi(t_j)^T \varphi(t_j)] - p(t_i) \beta \varphi(t_0)^T \varphi(t_i) + \alpha_i \gamma = p(t_i) b + r(t_i) \]

(11)

\[ \sum_{j=1}^{N} \alpha_j [\varphi'(t_j)^T \varphi(t_0) - p(t_j) \varphi(t_j)^T \varphi(t_0)] + \beta \varphi(t_0)^T \varphi(t_0) + b = y_0. \]

(12)

\[ \mathcal{L}(w, e, \alpha) = \frac{1}{2} (w^T w + \gamma e^T e) - \sum_{i=1}^{N} \alpha_i [w^T \varphi'(t_i) - p(t_i) (w^T \varphi(t_i) - w^T \varphi(t_0) + y_0) - r(t_i) - e_i], \]

(13)

\[ M_{ij} = K_{11}(t_i, t_j) - p(t_j) [K_{11}(t_i, t_j) - K_{11}(t_i, t_0)] - p(t_i) K_{11}(t_j, t_0) + \delta_{ij} / \gamma \]

(14)

\[ K_{4}(i, j) = K(t_i, t_j) - K(t_i, t_0) - K(t_j, t_0) + K(t_0, t_0) \]

\[ K_y(i, j) = K_1(t_i, t_j) - K_1(t_i, t_0) - p(t_j) [K_1(t_i, t_j) - K(t_i, t_j) - K(t_i, t_0) + K(t_i, t_0)] \]

(15)

### 3.4 Non-linear ODEs

Non-linear ODEs with initial value boundary conditions can be written generally using the form,

\[ y'(t) = f(t, y), \quad y(t_0) = y_0, \quad t \in [t_0, t_f]. \]

The solution form is again the one given in Eq. (8) and the domain discretized into \( N \), sub-intervals, \( t_0, t_1, \ldots, t_N \) (training points). Let \( e_i \) be the residuals for the solution \( \hat{y}(t_i) \),

\[ e_i = \varphi'(t_i) - f(t_i, \hat{y}(t_i)) \]

To minimize the error, the sum of the squares of the residuals is minimized, \( \min(e^T e) = \min \sum_{i=1}^{N} e_i^2 \). As in the linear case, the regularization term \( w^T w \) is added to the expression to be minimized. Now, the problem can be formulated as an optimization problem with constraints:

\[ \min \frac{1}{2} (w^T w + \gamma e^T e) \]

subject to:

\[ w^T \varphi(t_i) = f(t_i, y_i) + e_i \]

\[ w^T \varphi(t_0) + b = y_0 \]

\[ y_i = w^T \varphi(t_i) + b. \]

(16)

The variables \( y_i \) are introduced into the optimization problem to keep track of the nonlinear function \( f \) at the values corresponding to the grid points. The method of Lagrangian multiplies is used for this optimization problem, as shown in Eq. (16). The values where \( \mathcal{L} \) are zero give candidates for the minimum.

\[ \frac{\partial \mathcal{L}}{\partial w} = 0 \rightarrow w = \sum_{i=1}^{N} \alpha_i \varphi'(t_i) + \sum_{i=1}^{N} \eta_i \varphi(t_i) + \beta \varphi(t_0) \]

\[ \frac{\partial \mathcal{L}}{\partial e_i} = 0 \rightarrow \gamma e_i = -\alpha_i \]

\[ \frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \rightarrow w^T \varphi'(t_i) = f(t_i, y_i) + e_i \]

\[ \frac{\partial \mathcal{L}}{\partial \eta_i} = 0 \rightarrow y_i = w^T \varphi(t_i) + b \]

\[ \frac{\partial \mathcal{L}}{\partial \beta} = 0 \rightarrow w^T \varphi(t_0) + b = y_0 \]

\[ \frac{\partial \mathcal{L}}{\partial \eta_i} = 0 \rightarrow \alpha_i f_y(t_i, y_i) + \eta_i = 0 \]

A system of equations can be setup by substituting the results found by differentiating \( \mathcal{L} \) with respect to \( w \) and \( e_i \) into the remaining five equations found by taking the gradients of \( \mathcal{L} \). This will lead to a set of \( 3N + 2 \) equations and \( 3N + 2 \) unknowns which are \( \alpha_i, \eta_i, y_i, \beta, b \). This system of equations is given in Eq. (17). The system of equations given in Eq. (17) is solved using a multivariate Newton’s method. This leads to the same matrices in Eq. (20) of Ref. [11] with one exception: the regularization term, \( I/\gamma \), in the second row of the second column entry will be missing from this set of equations. The reason is, while running the experiments presented in this paper, that regularization term had an insignificant effect on the overall accuracy of the method. Moreover, as has been demonstrated here, it is not necessary in the setup of
The accuracy gain of ToC and CSVM compared to SVM methods is more accurate than the SVM method, and vice-versa for values less than one. Values greater than one indicate that the compared solution is better than the SVM solution, and the bottom plot shows the error ratio between the SVM and CSVM solutions. Values greater than one indicate that the compared method is more accurate than the SVM method, and vice versa for values less than one.

The accuracy gain of ToC and CSVM compared to SVM methods is more accurate than the SVM method, and vice-versa for values less than one. Values greater than one indicate that the compared solution is better than the SVM solution, and the bottom plot shows the error ratio between the SVM and CSVM solutions. Values greater than one indicate that the compared method is more accurate than the SVM method, and vice versa for values less than one.

The accuracy gain of ToC and CSVM compared to SVM methods is more accurate than the SVM method, and vice-versa for values less than one. Values greater than one indicate that the compared solution is better than the SVM solution, and the bottom plot shows the error ratio between the SVM and CSVM solutions. Values greater than one indicate that the compared method is more accurate than the SVM method, and vice versa for values less than one.

\[ \mathcal{L}(w, b, e, y, \alpha, \beta, \eta) = \frac{1}{2} (w^T w + \gamma e e^T) - \sum_{i=1}^{N} \alpha_i [w^T \varphi'(t_i) - f(t_i, y_i) - c_i] - \beta [w^T \varphi(t_0) + b - y_0] - \sum_{i=1}^{N} \eta_i [w^T \varphi(t_i) + b - y_i] \]

\[ \sum_{j=1}^{N} \alpha_j \varphi'(t_j)^T \varphi(t_i) + \sum_{j=1}^{N} \eta_j \varphi(t_j)^T \varphi(t_i) + \beta \varphi(t_0)^T \varphi(t_i) + \frac{\alpha_i}{\gamma} = f(t_i, y_i) \]

\[ \sum_{j=1}^{N} \alpha_j \varphi'(t_j)^T \varphi(t_i) + \sum_{j=1}^{N} \eta_j \varphi(t_j)^T \varphi(t_i) + \beta \varphi(t_0)^T \varphi(t_i) + b - y_i = 0 \]

\[ \sum_{j=1}^{N} \alpha_j \varphi'(t_j)^T \varphi(t_0) + \sum_{j=1}^{N} \eta_j \varphi(t_j)^T \varphi(t_0) + \beta \varphi(t_0)^T \varphi(t_0) + b = y_0 \]

\[ \beta + \sum_{i=1}^{N} \eta_i = 0 \]

\[ \alpha_i f_y(t_i, y_i) + \eta_i = 0 \]

The problem. Once the set of equations has been solved, the model solution is given in the dual form by,

\[ \hat{y}(t) = \sum_{i=1}^{N} \alpha_i \varphi'(t_i)^T \varphi(t) + \sum_{i=1}^{N} \eta_i \varphi(t_i)^T \varphi(t) + \beta \varphi(t_0)^T \varphi(t) + b. \]

As with the linear ODE case, the set of 3N + 2 equations that need to be solved and dual form of the model solution can be written in terms of the Kernel matrix and its derivatives.

4 Numerical Results

This section compares the methodologies described in the previous sections on a subset of the problems given in [11]. Problem #1 is a 1st order linear ODE. Problem #2 is a 1st order non-linear ODE, and problem #3 is a 2nd order linear ODE. All problems were solved in MATLAB on a Windows 10 operating system running on an Intel(R) Core(TM) i7-7700 CPU at 3.60GHz and 16.0 GB of RAM. The tabulated results from this comparison are included in the appendix at the end of the paper. A summary of those tabulated values is included in the subsections that follow, along with a short description of each problem.

4.1 Problem #1

This problem is originally from [11], and is called “Problem 5.4”. The problem is the ODE given in Eq. (18) for \( t \in [0, 1] \),

\[ \dot{y} + \left( t + \frac{1 + 3t^2}{1 + t + t^2} \right) y = t^3 + 2t + \frac{1 + 3t^2}{1 + t + t^2}, \]

subject to: \( y(0) = 1 \).

The accuracy gain of ToC and CSVM compared to SVM for problem #1 is shown in Fig. 1. The results were obtained using 100 training points. The top plot shows the error ratio between the SVM and ToC solutions, and the bottom plot shows the error ratio between the SVM and CSVM solutions. Values greater than one indicate that the compared method is more accurate than the SVM method, and vice versa for values less than one.

Figure 1 shows that ToC is the most accurate of the three methods followed by CSVM and finally SVM. The error reduction when using CSVM instead of SVM is typically an order of magnitude or less. However, the error reduction when using ToC instead of the other two methods is multiple orders of magnitude. The attentive reader will notice that the plot that includes ToC solution has less data points in Fig. 1 than the other methods. This is because the calculated points and the true solutions vary less than machine level accuracy and when the subtraction operation is used the resulting number becomes zero.

Tables 1 through 3 in the appendix compare the three methods for various numbers of training points when solv-
ing problem #1. Additionally, these tables show that solving the DE using ToC is both the quickest solution (i.e., it has the shortest training time) and the most accurate solution (i.e., it has the lowest maximum error and MSE on both the training set and test set) of the three. The CSVM results are the slowest, but they are more accurate than the SVM results. The accuracy gained when using CSVM compared to SVM is small, typically less than an order of magnitude. On the other hand, the accuracy gained when using ToC is multiple orders of magnitude. Moreover, the speed gained when using SVM compared to CSVM is typically less than an order of magnitude, whereas the speed gained when using ToC is approximately one order of magnitude. An accuracy versus speed comparison is shown graphically in Fig. 2, where the mean-squared error and training time are plotted for five specific cases: 8, 16, 32, 50, and 100 training points.

\[ \dot{y} = y^2 + t^2 \quad \text{subject to: } y(0) = 1. \quad (19) \]

The accuracy gain of ToC compared to SVM for problem #2 is shown in Fig. 3. The figure was created using 100 training points. The plot shows the error in the SVM solution divided by the ToC solution. Values greater than one indicate that ToC is more accurate than the SVM method, and values less than one indicate that ToC is less accurate than the SVM method. Figure 3 shows that ToC is the most accurate of the two methods. The ToC error is orders of magnitude lower than the SVM method.

Tables 4 through 5 in the appendix compare the two methods for various numbers of training points when solving problem #2. Additionally, these tables show that solving the DE using ToC is faster than using the SVM method for all cases except the second case. However, the speed gained using ToC is less than an order of magnitude. Furthermore, ToC is more accurate, by multiple orders of magnitude, than the SVM method for all of the test cases. In addition, ToC is able to continue reducing the MSE and maximum error on the test and training set as more training points are added, whereas the SVM method actually gets worse when adding more training points. An accuracy versus speed comparison is shown graphically in Fig. 4, where the mean-squared error and training time are plotted for five specific cases: 8, 16, 32, 50, and 100 training points.

\[ \ddot{y} + \frac{1}{5} \dot{y} + y = -\frac{1}{5} e^{-t/5} \cos t, \quad (20) \]

subject to \( y(0) = 1 \) and \( y'(0) = 1 \). The accuracy gain of ToC compared to SVM for problem #3 is shown in Fig. 5. The figure was created using 100 training points. The plot shows the error in the SVM solution divided by the ToC solution. Values greater than one indicate that ToC is more accurate than the SVM method, and values less than one indicate that ToC is less accurate than the SVM method. Figure 5 shows that ToC is the most accurate of the two methods. The ToC error is orders of magnitude lower than the SVM method.

Tables 6 through 7 in the appendix compare the two methods for various numbers of training points when solving problem #3. Additionally, these tables show that solving
the DE using ToC is faster than using the SVM method for all cases. The speed gained by using ToC is approximately an order of magnitude or less. Furthermore, ToC is more accurate than the SVM method for all of the test cases. One interesting note is that when moving from 16 to 32 training points the ToC actually loses a bit of accuracy, whereas the SVM method continues gain accuracy. Despite this, the ToC is still multiple orders of magnitude more accurate than the SVM method. An accuracy versus speed comparison is shown graphically in Fig. 6 where the mean-squared error and training time are plotted for five specific cases: 8, 16, 32, 50, and 100 training points.

5 Summary and Future Work

This article presents three methods to solve various types of ODEs: ToC, SVM method, and a method called CSVM that incorporates SVMs into the ToC framework. In addition, four problems are presented that include linear first order ODEs, non-linear first order ODEs, and linear second order ODEs. In summation, these comparisons show that, in general, ToC is faster, by approximately an order of magnitude or less, and more accurate, by multiple orders of magnitude, than the other two methods. The CSVM method has similar performance to the SVM method, but the CSVM method satisfies the boundary constraints exactly whereas the SVM method does not. Moreover, the performance of the CSVM method shows that machine learning algorithms can be easily incorporated into the ToC framework.

This will be exploited in future studies and specifically for partial DEs where the scalability of SVMs (or other machine learning algorithms such as neural networks) give a major advantage over the orthogonal basis functions used in the vanilla version of ToC. In addition, future work will focus on comparing the SVM method and ToC for higher order linear and non-linear ODEs as well as PDEs. Additionally, comparison problems should be looked at other than initial value problems. For example, problems could be used for comparison that have boundary value constraints, differential constraints, and integral constraints.
## APPENDIX

### TABLE 1
ToC Results for Problem #1

| Number of Training Points | Training Time (s) | Maximum Error on Training Set | MSE on Training Set | Maximum Error on Test Set | MSE on Test Set | \( m \) |
|---------------------------|-------------------|-------------------------------|---------------------|---------------------------|----------------|------|
| 8                         | 7.813e-05         | 6.033e-06                     | 1.057e-11           | 6.187e-06                 | 8.631e-12      | 7    |
| 16                        | 1.406e-04         | 2.012e-11                     | 1.257e-22           | 1.814e-11                 | 9.864e-23      | 17   |
| 32                        | 5.000e-04         | 2.220e-16                     | 3.331e-16           | 2.086e-32                 | 2.862e-32      | 25   |
| 50                        | 1.266e-03         | 4.441e-16                     | 2.220e-16           | 1.801e-32                 | 2.220e-16      | 26   |

### TABLE 2
SVM Results for Problem #1

| Number of Training Points | Training Time (s) | Maximum Error on Training Set | MSE on Training Set | Maximum Error on Test Set | MSE on Test Set | \( \gamma \) | \( \sigma \) |
|---------------------------|-------------------|-------------------------------|---------------------|---------------------------|----------------|----------|----------|
| 8                         | 1.719e-03         | 1.179e-03                     | 5.638e-11           | 1.439e-05                 | 7.251e-11      | 5.995e+17| 3.162e+00|
| 16                        | 1.719e-03         | 1.710e-06                     | 1.107e-12           | 1.849e-06                 | 1.161e-12      | 3.594e+11| 6.813e-01|
| 32                        | 2.188e-03         | 9.792e-08                     | 3.439e-15           | 9.525e-08                 | 2.356e-17      | 3.594e-15| 3.162e-01|
| 50                        | 4.375e-03         | 3.671e-09                     | 2.163e-19           | 3.537e-19                 | 2.154e-13      | 3.162e-01|           |

### TABLE 3
CSVM Results for Problem #1

| Number of Training Points | Training Time (s) | Maximum Error on Training Set | MSE on Training Set | Maximum Error on Test Set | MSE on Test Set | \( m \) |
|---------------------------|-------------------|-------------------------------|---------------------|---------------------------|----------------|------|
| 8                         | 3.125e-04         | 1.018e-05                     | 4.131e-11           | 5.547e-11                 | 2.154e-13      |      |
| 16                        | 1.406e-03         | 2.581e-07                     | 2.818e-07           | 2.468e-07                 | 5.995e-17      | 6.813e-01|
| 32                        | 5.313e-03         | 2.356e-17                     | 1.494e-16           | 3.594e-15                 | 3.162e-01      | 3.162e-01|
| 50                        | 3.281e-03         | 1.072e-08                     | 2.783e-17           | 7.734e-08                 | 3.162e-01      | 3.162e-01|
| 100                       | 1.078e-02         | 2.230e-09                     | 2.154e-19           | 5.354e-19                 | 1.468e-01      | 5.354e-19|

### TABLE 4
ToC Results for Problem #2

| Number of Training Points | Training Time (s) | Maximum Error on Training Set | MSE on Training Set | Maximum Error on Test Set | MSE on Test Set | \( m \) |
|---------------------------|-------------------|-------------------------------|---------------------|---------------------------|----------------|------|
| 8                         | 3.437e-04         | 8.994e-06                     | 2.242e-11           | 1.92e-05                  | 4.132e-11      |      |
| 16                        | 1.547e-03         | 4.586e-12                     | 6.514e-24           | 9.183e-12                 | 2.431e-23      | 16   |
| 32                        | 1.891e-03         | 3.109e-15                     | 9.291e-31           | 4.885e-15                 | 9.590e-31      | 32   |
| 50                        | 3.125e-03         | 1.110e-15                     | 2.665e-15           | 3.954e-31                 | 4.321e-31      | 32   |
| 100                       | 4.828e-03         | 1.776e-15                     | 2.665e-15           | 4.321e-31                 | 4.321e-31      | 32   |

### TABLE 5
SVM Results for Problem #2

| Number of Training Points | Training Time (s) | Maximum Error on Training Set | MSE on Training Set | Maximum Error on Test Set | MSE on Test Set | \( \gamma \) | \( \sigma \) |
|---------------------------|-------------------|-------------------------------|---------------------|---------------------------|----------------|----------|----------|
| 8                         | 7.813e-04         | 1.001e-03                     | 1.965e-07           | 7.904e-08                 | 1.000e+10      | 3.704e-01|          |
| 16                        | 1.250e-03         | 4.017e-03                     | 4.909e-06           | 4.514e-06                 | 1.000e+10      | 4.198e-01|          |
| 32                        | 6.875e-03         | 4.046e-03                     | 4.834e-06           | 4.575e-06                 | 1.000e+10      | 4.536e-01|          |
| 50                        | 1.203e-02         | 4.048e-03                     | 4.792e-06           | 4.580e-06                 | 1.000e+10      | 4.666e-01|          |
| 100                       | 3.156e-02         | 4.050e-03                     | 4.752e-06           | 4.582e-06                 | 1.000e+10      | 4.835e-01|          |
| Number of Training Points | Training Time (s) | Maximum Error on Training Set | MSE on Training Set | Maximum Error on Test Set | MSE on Test Set | $m$ |
|---------------------------|------------------|------------------------------|---------------------|---------------------------|----------------|-----|
| 8                         | 1.563e-03        | 1.313e-06                    | 5.184e-15           | 1.436e-06                 | 6.818e-15      | 8   |
| 16                        | 7.969e-04        | 5.551e-16                    | 6.122e-32           | 8.882e-16                 | 7.229e-32      | 15  |
| 32                        | 7.187e-04        | 1.221e-15                    | 2.372e-31           | 9.992e-16                 | 7.229e-31      | 15  |
| 50                        | 5.000e-04        | 7.772e-16                    | 3.991e-32           | 5.551e-16                 | 3.672e-32      | 15  |
| 100                       | 9.844e-04        | 7.772e-16                    | 5.525e-32           | 6.661e-16                 | 3.518e-32      | 15  |

**TABLE 6**
ToC Results for Problem #3

| Number of Training Points | Training Time (s) | Maximum Error on Training Set | MSE on Training Set | Maximum Error on Test Set | MSE on Test Set | $\gamma$ | $\sigma$ |
|---------------------------|------------------|------------------------------|---------------------|---------------------------|----------------|----------|---------|
| 8                         | 1.563e-03        | 1.420e-06                    | 8.300e-13           | 1.638e-06                 | 6.922e-13      | 5.995e+17 | 6.813e+00 |
| 16                        | 1.875e-03        | 1.811e-08                    | 1.015e-16           | 1.871e-08                 | 1.014e-16      | 3.594e+15 | 3.162e+00 |
| 32                        | 4.667e-03        | 5.455e-10                    | 1.025e-19           | 9.005e-10                 | 1.015e-19      | 5.995e+17 | 1.468e+00 |
| 50                        | 7.656e-03        | 8.563e-11                    | 3.771e-21           | 8.391e-11                 | 3.648e-21      | 2.154e+13 | 1.468e+00 |
| 100                       | 2.688e-02        | 6.441e-11                    | 1.500e-21           | 6.128e-11                 | 1.640e-21      | 2.154e+13 | 1.468e+00 |

**TABLE 7**
SVM Results for Problem #3
REFERENCES

[1] J. Dormand, P. Prince, A Family of Embedded Runge-Kutta Formulas, J. Comp. Appl. Math. 6 (1980) 19–26.
[2] M. M. Berry, L. M. Healy, Implementation of Gauss-Jackson integration for orbit propagation, The Journal of the Astronautical Sciences 52 (3) (2004) 351–357.
[3] X. Bai, J. L. Junkins, Modified Chebyshev-Picard Iteration Methods for Orbit Propagation, The Journal of the Astronautical Sciences 58 (4) (2011) 583–613.
[4] J. L. Junkins, A. B. Younes, R. Woollands, X. Bai, Picard Iteration, Chebyshev Polynomials, and Chebyshev Picard Methods: Application in Astrodynamics, The Journal of the Astronautical Sciences 60 (3) (2015) 623–653. doi:10.1007/s40295-015-0061-1
[5] J. Reed, A. B. Younes, B. Macomber, J. L. Junkins, D. J. Turner, State Transition Matrix for Perturbed Orbital Motion using Modified Chebyshev Picard Iteration, The Journal of the Astronautical Sciences 6 (2015) 148–167. doi: 10.1007/s40295-015-0051-3.
[6] T. A. Driscoll, N. Hale, Rectangular spectral collocation, IMA Journal of Numerical Analysis 36 (1) (2016) 108–132. doi: 10.1093/imanum/dru062. URL http://dx.doi.org/10.1093/imanum/dru062
[7] D. Mortari, The Theory of Connections: Connecting Points, MDPI Mathematics 5 (57) (2017) 1–18. doi:10.3390/math5040057 URL http://www.mdpi.com/2227-7390/5/4/57
[8] D. Mortari, Least-squares Solutions of Linear Differential Equations, MDPI Mathematics 5 (48) (2017) 1–18.
[9] D. Mortari, H. Johnston, L. Smith, Least-squares Solutions of Non-linear Differential Equations, in: 2018 AAS/AIAA Space Flight Mechanics Meeting Conference, Kissimmee, FL, January 8–12, 2018, AAS/AIAA, 2018, p. 0959.
[10] H. Johnston, D. Mortari, Linear Differential Equations Subject to Relative, Integral, and Infinite Constraints, in: 2018 AAS/AIAA Astrodynamics Specialist Conference Snowbird, UT, August 19–23, 2018, AAS/AIAA, 2018.
[11] T. F. S. Mehrkanoon, J. A. K. Suykens, Approximate Solutions to Ordinary Differential Equations using Least-squares Support Vector Machines, IEEE Transactions on Neural Networks and Learning Systems 23 (9) (2012) 1356–1367.
[12] V. N. Vapnik, Statistical Learning Theory, Wiley, 1998.
[13] C. Lanczos, Applied Analysis, Dover Publications, Inc., New York, 1957, Ch. 7, p. 504.
[14] K. Wright, Chebyshev Collocation Methods for Ordinary Differential Equations, The Computer Journal 6 (1) (1964) 358–365.
[15] A. Ng, CS229 Lecture notes: Part V Support Vector Machines, http://cs229.stanford.edu/notes/cs229-notes3.pdf online; accessed 13 December 2018.

Carl Leake is a doctoral student at Texas A&M University, conducting research in the fields of n-dimensional searching techniques, neural networks, Theory of Connections, and flexible multi-body dynamics. Carl earned his Bachelors degree in Aerospace Engineering from Embry-Riddle Aeronautical University in 2017. E-mail: leakec@tamu.edu

Hunter Johnston is a doctoral student at Texas A&M University, focusing on the development of the Theory of Connections and its applications problems in aerospace engineering. Hunter earned his Bachelors degree in Mechanical Engineering from the University of Dayton in 2017. E-mail: hunterjohnston@tamu.edu

Lidia Smith is a Professor in the Mathematics Department at Blinn College. She received her Ph.D. in Mathematics from Texas A&M University. Her graduate research work was on orbits of operators in Hilbert spaces. Current research interests include the analysis of mathematical models in sensor networks and studying the mathematics behind machine learning algorithms. E-mail: lidia.smith@blinn.edu

Daniele Mortari is Professor of Aerospace Engineering at Texas A&M University, working on the field of attitude and position estimation, satellite constellation design, and linear algebra. In addition, he has taught at the School of Aerospace Engineering of Rome’s University, and at Electronic Engineering of Perugia’s University. He received his Dottore degree in Nuclear Engineering from University of Rome “La Sapienza,” in 1981. He has published more than 300 papers and received the best paper Award from AAS/AIAA, two NASA’s Group Achievement Awards, 2007 IEEE Judith A. Resnik Award, and 2015 AAS Dirk Brouwer Award. He is an IEEE and AAS Fellow, AIAA Associate Fellow, and IAA Corresponding member, Honorary Member of IEEE-AESS Space System Technical Panel, and a former IEEE Distinguish Speaker. E-mail: mortari@tamu.edu