machine learning & knowledge extraction

MDPI

Article

Analytically Embedding Differential Equation Constraints into Least Squares Support Vector Machines Using the Theory of Functional Connections

Carl Leake [1,*], Hunter Johnston [1], Lidia Smith [2] and Daniele Mortari [1]

1 Department of Aerospace Engineering, Texas A&M University, College Station, TX 77843, USA; hunterjohnston@tamu.edu (H.J.); mortari@tamu.edu (D.M.)
2 Mathematics Department, Blinn College, Bryan, TX 77802, USA; lidia.smith@gmail.com
* Correspondence: leakec@tamu.edu

Received: 21 August 2019; Accepted: 5 October 2019; Published: 9 October 2019

Abstract: Differential equations (DEs) 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. While there are many other techniques for finding approximate solutions to these equations, this paper looks to compare the application of the *Theory of Functional Connections* (TFC) with one based on least-squares support vector machines (LS-SVM). The TFC method uses a constrained expression, an expression that always satisfies the DE constraints, which transforms the process of solving a DE into solving an unconstrained optimization problem that is ultimately solved via least-squares (LS). In addition to individual analysis, the two methods are merged into a new methodology, called constrained SVMs (CSVM), by incorporating the LS-SVM method into the TFC framework to solve unconstrained problems. Numerical tests are conducted on four sample problems: One first order linear ordinary differential equation (ODE), one first order nonlinear ODE, one second order linear ODE, and one two-dimensional linear partial differential equation (PDE). Using the LS-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, TFC is shown to be slightly faster (by an order of magnitude or less) and more accurate (by multiple orders of magnitude) than the LS-SVM and CSVM approaches.

Keywords: differential equation; numerical methods; support vector machines; function approximation; least-squares; theory of functional connections

1. Introduction

Differential equations (DE) and their solutions are important topics 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 [3–5], 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 added to enforce the constraints.

The *Theory of Functional Connections* (TFC) is a new technique that analytically derives a constrained expression which satisfies the problem's constraints exactly while maintaining a function that can be freely chosen [7]. This theory, initially called "Theory of Connections", has been renamed for two reasons. First, the "Theory of Connections" already identifies a specific theory in differential geometry, and second, what this theory is actually doing is "functional interpolation", as it provides all functions satisfying a set of constraints in terms of a function and any derivative in rectangular domains of n-dimensional spaces. 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 [8–11], have defined this free function as a summation of basis functions; more specifically, orthogonal polynomials.

This work was motivated by recent results that solve ordinary DEs using a least-squares support vector machine (LS-SVM) approach [12]. While this article focuses on the application of LS-SVMs to solve DEs, the study and use of LS-SVMs remains relevant in many areas. In reference [13] the authors use the support vector machines to predict the risk of mold growth on concrete tiles. The mold growth on roofs affects the dynamics of heat and moisture through buildings. The approach leads to reduced computational effort and simulation time. The work presented in reference [14] uses LS-SVMs to predict annual runoff in the context of water resource management. The modeling process starts with building a stationary set of runoff data based on mode functions which are used as input points in the prediction by the SVM technique when chaotic characteristics are present. Furthermore, reference [15] uses the technique of LS-SVMs as a less costly computational alternative that provides superior accuracy compared to other machine learning techniques in the civil engineering problem of predicting the stability of breakwaters. The LS-SVM framework was applied to tool fault diagnosis for ensuring manufacturing quality [16]. In this work, a fault diagnosis method was proposed based on stationary subspace analysis (SSA) used to generate input data used for training with LS-SVMs.

In this article, LS-SVMs are incorporated into the TFC framework as the free function, and the combination of these two methods is used to solve DEs. Hence, the contributions of this article are twofold: (1) This article demonstrates how boundary conditions can be analytically embedded, via TFC, into machine learning algorithms and (2) this article compares using a LS-SVM as the free function in TFC with the standard linear combination of CP. Like vanilla TFC, the SVM model for function estimation [17] 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 $\boldsymbol{\varphi}(x)$. Further, with the kernel trick, the function to be evaluated is determined based on a linear combination of kernel functions; Gaussian kernels are a popular choice, and are used in this article.

This article compares the combined method, referred to hereafter as CSVM for constrained LS-SVMs, to vanilla versions of TFC [8,9] and LS-SVM [12] over a variety of DEs. In all cases, the vanilla version of TFC outperforms both the LS-SVM and the CSVM methods in terms of accuracy and speed. The CSVM method does not provide much accuracy or speed benefit over LS-SVM, except in the PDE problem, and in some cases has a less accurate or slower solution. However, in every case the CSVM satisfies the boundary conditions of the problem exactly, whereas the vanilla LS-SVM method solves the boundary condition with the same accuracy as the remainder of the data points in the problem. Thus, this article provides support that in the application of solving DEs, CP are a better choice for the TFC free function than LS-SVMs.

While the CSVM method underperforms vanilla TFC when solving DEs, its implementation and numerical verification in this article still provides an important contribution to the scientific community. CSVM demonstrates that the TFC framework provides a robust way to analytically embed constraints into machine learning algorithms; an important problem in machine learning. This technique can be extended to any machine learning algorithm, for example deep neural networks. Previous techniques have enforced constraints in deep neural networks by creating parallel structures, such as radial basis networks [18], adding the constraints to the loss function to be minimized [19], or by modifying the optimization process to include the constraints [20]. However, all of these techniques significantly modify the deep neural network architecture or the training process. In contrast, embedding the constraints with TFC does not require this. Instead, TFC provides a way to analytically embed these constraints into the deep neural network. In fact, any machine learning algorithm that is differentiable up to the order of the DE can be seamlessly incorporated into TFC. Future work will leverage this benefit to analyze the ability to solve DEs using other machine learning algorithms.

2. Background on the *Theory of Functional Connections*

The *Theory of Functional Connections* (TFC) is a generalized interpolation method, which provides a mathematical framework to analytically embed constraints. The univariate approach [7] to derive the expression for all functions satisfying k linear constraints follows,

$$f(t) = g(t) + \sum_{i=1}^{k} \eta_i \, p_i(t), \tag{1}$$

where $g(t)$ represents a "freely chosen" function, η_i are the coefficients derived from the k linear constraints, and $p_i(t)$ are user selected functions that must be linearly independent from $g(t)$. Recent research has applied this technique to embedding DE constraints using Equation (1), allowing for least-squares (LS) solutions of initial-value (IVP), boundary-value (BVP), and multi-value (MVP) problems on both linear [8] and nonlinear [9] ordinary differential equations (ODEs). In general, this approach has developed a fast, accurate, and robust unified framework to solve DEs. The application of this theory can be explored for a second-order DE such that,

$$F(t, y, \dot{y}, \ddot{y}) = 0 \qquad \text{subject to:} \begin{cases} y(t_0) = y_0 \\ \dot{y}(t_0) = \dot{y}_0 \end{cases} \tag{2}$$

By using Equation (1) and selecting $p_1(t) = 1$ and $p_2(t) = t$, the constrained expression becomes,

$$y(t) = g(t) + \eta_1 + \eta_2 \, t. \tag{3}$$

By evaluating this function at the two constraint conditions a system of equations is formed in terms of η,

$$\begin{Bmatrix} y_0 \\ \dot{y}_0 \end{Bmatrix} = \begin{bmatrix} 1 & t_0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} \eta_1 \\ \eta_2 \end{Bmatrix}$$

which can be solved for by matrix inversion leading to,

$$\begin{aligned} \eta_1 &= (y_0 - g_0) - t_0 \, (\dot{y}_0 - \dot{g}_0) \\ \eta_2 &= \dot{y}_0 - \dot{g}_0. \end{aligned}$$

These terms can are substituted in Equation (3) and the final constrained expression is realized,

$$y(t) = g(t) + (y_0 - g_0) + (t - t_0)(\dot{y}_0 - \dot{g}_0),$$

By observation, it can be seen that the function for $y(t)$ always satisfies the initial value constraints regardless of the function $g(t)$. Substituting this function into our original DE specified by Equation (2) transforms the problem into a new DE with no constraints,

$$\tilde{F}(t, g, \dot{g}, \ddot{g}) = 0. \tag{4}$$

Aside from the independent variable t, this equation is only a function of the unknown function $g(t)$. By discretizing the domain and expressing $g(t)$ as some universal function approximator, the problem can be posed as an unconstrained optimization problem where the loss function is defined by the residuals of the $\tilde{F}$ function. Initial applications of the TFC method to solve DEs [8,9] expanded $g(t)$ as some basis (namely Chebyshev or Legendre orthogonal polynomials); however, the incorporation of a machine learning framework into this free function has yet to be explored. This will be discussed in following sections. The original formulation expressed $g(t)$ as,

$$g(t) = \boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{h}(x) \qquad \text{where} \qquad x = x(t),$$

where $\boldsymbol{\xi}$ is an unknown vector of m coefficients and $\boldsymbol{h}(x)$ is the vector of m basis functions. In general the independent variable is $t \in [t_0, t_f]$ while the orthogonal polynomials are defined in $x \in [-1, +1]$. This gives the linear mapping between x and t,

$$x = x_0 + \frac{x_f - x_0}{t_f - t_0}(t - t_0) \qquad \leftrightarrow \qquad t = t_0 + \frac{t_f - t_0}{x_f - x_0}(x - x_0).$$

Using this mapping, the derivative of the free function becomes,

$$\frac{\mathrm{d}g}{\mathrm{d}t} = \frac{\mathrm{d}\boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{h}(x)}{\mathrm{d}t} = \boldsymbol{\xi}^{\mathrm{T}}\frac{\mathrm{d}\boldsymbol{h}(x)}{\mathrm{d}x} \cdot \frac{\mathrm{d}x}{\mathrm{d}t},$$

where it can be seen that the term $\frac{\mathrm{d}x}{\mathrm{d}t}$ is a constant such that,

$$c := \frac{x_f - x_0}{t_f - t_0}.$$

Using this definition, it follows that all subsequent derivatives are,

$$\frac{\mathrm{d}^k g}{\mathrm{d}t^k} = c^k \boldsymbol{\xi}^{\mathrm{T}}\frac{\mathrm{d}^k\boldsymbol{h}(x)}{\mathrm{d}x^k}.$$

Lastly, the DE given by Equation (4) is discretized over a set of N values of t (and inherently x). When using orthogonal polynomials, the optimal point distribution (in terms of numerical efficiency) is provided by collocation points [21,22], defined as,

$$x_i = -\cos\left(\frac{i\pi}{N}\right) \qquad \text{for} \qquad i = 0, 1, \cdots, N,$$

By discretizing the domain, Equation (4) becomes a function solely of the unknown parameter $\boldsymbol{\xi}$,

$$\tilde{F}(\boldsymbol{\xi}) = 0, \tag{5}$$

which can be solved using a variety of optimization schemes. If the original DE is linear then the new DE defined by Equation (5) is also linear. In this case, Equation (5) is a linear system,

$$A\,\boldsymbol{\xi} = \boldsymbol{b},$$

which can be solved using LS [8]. If the DE is nonlinear, a nonlinear LS approach is needed, which requires an initial guess for $\boldsymbol{\xi}_0$. This initial guess can be obtained by a LS fitting of a lower order integrator solution, such as one provided by a simple improved Euler method. By defining the residuals of the DE as the loss function $\mathcal{L}_k := \tilde{F}(\boldsymbol{\xi}_k)$, the nonlinear Newton iteration is,

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k - (\mathcal{J}_k^{\mathrm{T}}\mathcal{J}_k)^{-1}\mathcal{J}_k^{\mathrm{T}}\mathcal{L}_k \quad \text{where} \quad \mathcal{J}_k := \left[\frac{\partial\mathcal{L}}{\partial\boldsymbol{\xi}}\right]_k$$

where k is the iteration. The convergence is obtained when the L_2-norm of $\mathcal{L}$ satisfies $L_2[\mathcal{L}_k] < \varepsilon$, where ε is a specified convergence tolerance. The final value of $\boldsymbol{\xi}$ is then used in the constrained expression to provide an approximated analytical solution that perfectly satisfies the constraints. Since the function is analytical, the solution can be then used for further manipulation (e.g., differentiation, integration, etc.). The process to solve PDEs follows a similar process with the major difference involving the derivative of the constrained expression. The TFC extension to n-dimensions and a detailed explanation of the derivation of these constrained expressions are provided in references [23,24]. Additionally, the free function also becomes multivariate, increasing the complexity when using CPs.

3. The Support Vector Machine Technique

3.1. An Overview of SVMs

Support vector machines (SVMs) were originally introduced to solve classification problems [17]. A classification problem consists of determining if a given input, x, belongs to one of two possible classes. The proposed solution was to find a decision boundary surface that separates the two classes. The equation of the separating boundary depended only on a few input vectors called the support vectors.

The training data is assumed to be separable by a linear decision boundary. Hence, a separating hyperplane, H, with equation $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(x) + b = 0$, is sought. The parameters are rescaled such that the closest training point to the hyperplane H, let's say (x_k, y_k), is on a parallel hyperplane H_1 with equation $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(x) + b = 1$. By using the formula for orthogonal projection, if $\boldsymbol{x}$ satisfies the equation of one of the hyperplanes, then the signed distance from the origin of the space to the corresponding hyperplane is given by $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(x)/\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$. Since $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(x)$ equals $-b$ for H, and $1-b$ for H_1, it follows that the distance between the two hyperplanes, called the "separating margin", is $1/\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$. Thus to find the largest separating margin, one needs to minimize $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$. The optimization problem becomes,

$$\min \frac{1}{2}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}\right) \quad \text{subject to: } y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(x_i) + b) \geq 1, \quad i = 1, \ldots, m.$$

If a separable hyperplane does not exist, the problem is reformulated by taking into account the classification errors, or slack variables, ξ_i , and a linear or quadratic expression is added to the cost function. The optimization problem in the non-separable case is,

$$\min \frac{1}{2}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}\right) + C\left(\sum \xi_i\right) \quad \text{subject to: } y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(x_i) + b) \geq 1 - \xi_i.$$

When solving the optimization problem by using Lagrange multipliers, the function $\boldsymbol{\varphi}(t)$ always shows up as a dot product with itself; thus, the kernel trick [25] can be applied. In this research, the kernel function chosen is the radial basis function (RBF) kernel proposed in [12]. Hence, the function $\boldsymbol{\varphi}(t)$ can be written using the kernel [25],

$$K(t_i, t) = \boldsymbol{\varphi}(t_i)^{\mathrm{T}}\boldsymbol{\varphi}(t) = \exp\left(-\frac{(t-t_i)^2}{\sigma^2}\right), \tag{6}$$

and its partial derivatives [12,26],

$$
\begin{aligned}
K(t_i, t_j) &= \boldsymbol{\varphi}(t_i)^{\mathsf{T}}\boldsymbol{\varphi}(t_j) = \exp\left(-\frac{(t_i - t_j)^2}{\sigma^2}\right) \\
K_1(t_i, t_j) &= \boldsymbol{\varphi}'(t_i)^{\mathsf{T}}\boldsymbol{\varphi}(t_j) = -\frac{2(t_i - t_j)}{\sigma^2}\exp\left(-\frac{(t_i - t_j)^2}{\sigma^2}\right) \\
K_1^{\mathsf{T}}(t_i, t_j) &= \boldsymbol{\varphi}(t_i)^{\mathsf{T}}\boldsymbol{\varphi}'(t_j) = \frac{2(t_i - t_j)}{\sigma^2}\exp\left(-\frac{(t_i - t_j)^2}{\sigma^2}\right) \\
K_{11}(t_i, t_j) &= \boldsymbol{\varphi}'(t_i)^{\mathsf{T}}\boldsymbol{\varphi}'(t_j) = \frac{2}{\sigma^2} - \frac{4(t_i - t_j)^2}{\sigma^4}\exp\left(-\frac{(t_i - t_j)^2}{\sigma^2}\right),
\end{aligned}
\tag{7}
$$

where the Kernel bandwidth, σ, is a tuning parameter that must be chosen by the user.

We follow the method of solving DEs using RBF kernels proposed in [12]. 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,

$$\hat{y}(x) = \sum_{i=1}^{N} w_i \varphi_i(x) + b = \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\varphi}(x) + b \tag{8}$$

is proposed for the solution $y(t)$. 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 $\boldsymbol{e}$ be the vector of residuals obtained when using 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,

$$\hat{y}'(t_i) - p(t_i)\hat{y}(t_i) - r(t_i) = e_i.$$

This results in,

$$\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\varphi}'(t_i) = p(t_i)[\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\varphi}(t_i) + b] + r(t_i) + e_i,$$

and for the initial condition, it is desired that,

$$\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\varphi}(t_0) + b = y_0,$$

is satisfied exactly. In order to have the model close to the exact solution, the sum of the squares of the residuals, $\boldsymbol{e}^{\mathsf{T}}\boldsymbol{e}$, 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,

$$\min \frac{1}{2}\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{w} + \gamma\,\boldsymbol{e}^{\mathsf{T}}\boldsymbol{e}\right) \quad \text{subject to:} \begin{cases} \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\varphi}'(t_i) - p(t_i)\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\varphi}(t_i) + b\right) - r(t_i) - e_i = 0 \\ \boldsymbol{w}^{\mathsf{T}}\boldsymbol{\varphi}(t_0) + b - y_0 = 0. \end{cases}$$

Using the method of Lagrange multipliers, a loss function, $\mathcal{L}$, is defined using the objective function from the optimization problem and appending the constraints with corresponding Lagrange multipliers α_i and β.

$$\mathcal{L} = \frac{1}{2}\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{w} + \gamma\,\boldsymbol{e}^{\mathsf{T}}\boldsymbol{e}\right) + \alpha_i\left[\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\varphi}'(t_i) - p(t_i)\left(\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\varphi}(t_i) + b\right) - r(t_i) - e_i\right] + \beta\left[\boldsymbol{w}^{\mathsf{T}}\boldsymbol{\varphi}(t_0) + b - y_0\right]$$

The values where the gradient of $\mathcal{L}$ is zero give candidates for the minimum.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0 \quad &\rightarrow \quad \boldsymbol{w} = \sum_{i=1}^{N} \alpha_i \left[\boldsymbol{\varphi}'(t_i) - p(t_i)\boldsymbol{\varphi}(t_i)\right] + \beta \boldsymbol{\varphi}(t_0) \\
\frac{\partial \mathcal{L}}{\partial e_i} = 0 \quad &\rightarrow \quad \gamma e_i = -\alpha_i \\
\frac{\partial \mathcal{L}}{\partial b} = 0 \quad &\rightarrow \quad 0 = \sum_{i=1}^{N} \alpha_i q(t_i) - \beta \\
\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \quad &\rightarrow \quad 0 = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\varphi}'(t_i) - p(t_i)\left(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{\varphi}(t_i) + b\right) - g(t_i) - e_i \\
\frac{\partial \mathcal{L}}{\partial \beta} = 0 \quad &\rightarrow \quad 0 = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\varphi}(t_0) + b - y_0
\end{aligned}$$

Note that the conditions found by differentiating $\mathcal{L}$ with respect to α_i and β 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,

$$\boldsymbol{w} = \sum_{j=1}^{N} \alpha_j \left[\boldsymbol{\varphi}'(t_j) - p(t_j)\boldsymbol{\varphi}(t_j)\right] + \beta \boldsymbol{\varphi}(t_0),$$

we obtain a new formulation of the approximate solution

$$\hat{y}(t) = \sum_{j=1}^{N} \alpha_j \left[\boldsymbol{\varphi}'(t_j) - p(t_j)\boldsymbol{\varphi}(t_j)\right]^{\mathsf{T}} \boldsymbol{\varphi}(t) + \beta \boldsymbol{\varphi}(t_0)^{\mathsf{T}} \boldsymbol{\varphi}(t) + b$$

where the inner products of $\boldsymbol{\varphi}(t)$ can be re-written using Equations (6) and (7), and the parameter σ in the kernal matrix is a value that is learned during the training period together with the coefficients $\boldsymbol{w}$. The remaining gradients of $\mathcal{L}$ can be used to form a linear system of equations where α_i, β, and b are the only unknowns. Note, that this system of equations can also be expressed using the kernal matrix and its partial derivatives rather than inner-products of $\boldsymbol{\varphi}$.

3.2. Constrained SVM (CSVM) Technique

In the TFC method [7], the general constrained expression can be written for an initial value constraint as,

$$y(t) = g(t) + (y_0 - g_0),$$

where $g(t)$ is a "freely chosen" function. In prior studies [8,9,11], this free function was defined by a set of orthogonal basis functions, but this function can also be defined using SVMs,

$$g(t) = \sum_{i=1}^{N} w_i \varphi_i(t) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\varphi}(t),$$

where g_0 becomes,

$$g(t_0) = \sum_{i=1}^{N} w_i \varphi_i(t_0) = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{\varphi}(t_0).$$

This leads to the equation,

$$y(t) = \boldsymbol{w}^{\mathsf{T}} \left[\boldsymbol{\varphi}(t) - \boldsymbol{\varphi}(t_0)\right] + y_0, \tag{9}$$

where the initial value constraint is always satisfied regardless of the values of $\boldsymbol{w}$ and $\boldsymbol{\varphi}(t)$. Through this process, the constraints only remain on the residuals and the problem becomes,

$$\min \frac{1}{2}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + \gamma \boldsymbol{e}^{\mathrm{T}}\boldsymbol{e}\right) \quad \text{subject to:} \quad \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}'(t_i) - p(t_i)\left[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(t_i) - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(t_0) + y_0\right] - r(t_i) - e_i = 0.$$

Again, using the method of Lagrange multipliers, a term is introduced for the constraint on the residuals, leading to the expression,

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

The values where the gradient of $\mathcal{L}$ is zero give candidates for the minimum,

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}} = 0 \quad &\rightarrow \quad \boldsymbol{w} = \sum_{i=1}^{N}\alpha_i\left[\boldsymbol{\varphi}'(t_i) - p(t_i)\left(\boldsymbol{\varphi}(t_i) - \varphi(t_0)\right)\right]\\
\frac{\partial\mathcal{L}}{\partial e_i} = 0 \quad &\rightarrow \quad e_i = -\frac{\alpha_i}{\gamma}\\
\frac{\partial\mathcal{L}}{\partial \alpha_i} = 0 \quad &\rightarrow \quad 0 = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}'(t_i) - p(t_i)\left(\boldsymbol{w}^{\mathrm{T}}\left(\boldsymbol{\varphi}(t_i) - \boldsymbol{\varphi}(t_0)\right) + y_0\right) - r(t_i) - e_i.
\end{aligned}$$

Using,

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

we obtain a new formulation of the approximate solution given by Equation (9), that can be expressed in terms of the kernel and its derivatives. Combining the three equations for the gradients of $\mathcal{L}$, we can obtain a linear system with unknowns α_j,

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

The coefficient matrix is given by,

$$M_{ij} = K_{11}(t_i,t_j) - p(t_j)\left[K_1(t_i,t_j) - K_1(t_i,t_0)\right] - p(t_i)K_y(i,j) + \delta_{ij}/\gamma,$$

where we use the notation,

$$\begin{aligned}
K_4(t_i,t_j) &= K(t_i,t_j) - K(t_j,t_0) - K(t_i,t_0) + 1\\
K_y(t_i,t_j) &= K_1(t_j,t_i) - K_1(t_j,t_0) - p(t_j)K_4(t_i,t_j).
\end{aligned}$$

Finally, in terms of the kernel matrix, the approximate solution at the grid points is given by,

$$y(t_i) = \sum_{j=1}^{N}\alpha_j K_y(t_i,t_j) + y_0,$$

and a formula for the approximate solution at an arbitrary point t is given by,

$$y(t) = \sum_{j=1}^{N}\alpha_j K_y(t,t_j) + y_0.$$

3.3. Nonlinear ODEs

The method for solving nonlinear, first-order ODEs with LS-SVM comes from reference [12]. Nonlinear, first-order 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 Equation (8) and the domain is again discretized into N sub-intervals, $t_0, t_1, \ldots, t_N$ (training points). Let e_i be the residuals for the solution $\hat{y}(t_i)$,

$$e_i = \hat{y}'(t_i) - f(t_i, \hat{y}(t_i)).$$

To minimize the error, the sum of the squares of the residuals is minimized. As in the linear case, the regularization term $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$ is added to the expression to be minimized. Now, the problem can be formulated as an optimization problem with constraints,

$$\min \frac{1}{2}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + \gamma\, \boldsymbol{e}^{\mathrm{T}}\boldsymbol{e}\right) \quad \text{subject to: } \begin{cases} \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}'(t_i) = f(t_i, y_i) + e_i \\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(t_0) + b = y_0 \\ y_i = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(t_i) + b. \end{cases}$$

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 Lagrange multipliers is used for this optimization problem just as in the linear case. This leads to a system of equations that can be solved using a multivariate Newton's method. As with the linear ODE case, the set of equations to be solved and the dual form of the model solution can be written in terms of the kernel matrix and its derivatives.

The solution for nonlinear ODEs when using the CSVM technique is found in a similar manner, but the primal form of the solution is based on the constraint function from TFC. Just as the linear ODE case changes to encompass this new primal form, so does the nonlinear case. A complete derivation for nonlinear ODEs using LS-SVM and CSVM is provided in Appendix B.

3.4. Linear PDEs

The steps for solving linear PDEs using LS-SVM are the same as when solving linear ODEs, and are shown in detail in reference [27]. The first step is to write out the optimization problem to be solved. The second is to solve that optimization problem using the Lagrange multipliers technique. The third is to write the resultant set of equations and dual-form of the solution in terms of the kernel matrix and its derivatives.

Solving linear PDEs using the CSVM technique follows the same solution steps except the primal form of the solution is derived from a TFC constrained expression. A complete derivation for the PDE shown in problem #4 of the numerical results section using CSVM is provided in Appendix C. The main difficulty in this derivation stems from the numerous amount of times the function $\boldsymbol{\varphi}$ shows up in the TFC constrained expression. As a result, the set of equations produced by taking gradients of $\mathcal{L}$ contain hundreds of kernel matrices and their derivatives. The only way to make this practical (in terms of the derivation and programming the result) was to write the constrained expression in tensor form. This was reasonable to perform for the simple linear PDE used in this paper, but would become prohibitively complicated for higher dimensional PDEs. Consequently, future work will investigate using other machine learning algorithms, such as neural networks, as the free function in the TFC framework.

4. Numerical Results

This section compares the methodologies described in the previous sections on four problems given in references [12] and [27]. Problem #1 is a first order linear ODE, problem #2 is a first order nonlinear ODE, problem #3 is a second order linear ODE, and problem #4 is a second order linear PDE. All problems were solved in `MATLAB R2018b` (MathWorks, Natick, MA, USA) on a Windows 10 operating system running on an Intel® Core™ i7-7700 CPU at 3.60GHz and 16.0 GB of RAM. Since all test problems have analytical solutions, absolute error and mean-squared error (MSE) were used to quantify the error of the methods. MSE is defined as,

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{10}$$

where n is the number of points, y_i is the true value of the solution, and $\hat{y}_i$ is the estimated value of the solution at the i-th point.

The tabulated results from this comparison are included in Appendix A. A graphical illustration and summary of those tabulated values is included in the subsections that follow, along with a short description of each problem. These tabulated results also include the tuning parameters for each of the methods. For TFC, the number of basis functions, m, was found using a grid search method, where the residual of the differential equation was used to choose the best value of m. For LS-SVM and CSVM, the kernel bandwidth, σ, and the parameter γ were found using a grid search method for problems #1, #3, and #4. For problem #2, the value of σ for the LS-SVM and CSVM methods was tuned using fminsearch while the value of γ was fixed at 10^{10} [12]. This method was used in problem #2 rather than grid search because it did a much better job choosing tuning parameters that reduced the error of the solution. For all problems, a validation set was used to choose the best value for σ and γ [12,26]. It should be noted that the tuning parameter choice affects the accuracy of the solution. Thus, it may be possible to achieve more accurate results if a different method is used to find the value of the tuning parameters. For example, an algorithm that is better suited to finding global optimums, such as a genetic algorithm, may find better tuning parameter values than the methods used here.

4.1. Problem #1

Problem #1 is the linear ODE,

$$\dot{y} + \left(t + \frac{1+3t^2}{1+t+t^3}\right)y = t^3 + 2t + t^2\frac{1+3t^2}{1+t+t^3}, \quad \text{subject to: } y(0) = 1, \quad t \in [0,1] \tag{11}$$

which has the analytic solution,

$$y(t) = \frac{e^{-t^2/2}}{1+t+t^3} + t^2$$

The accuracy gain of TFC and CSVM compared to LS-SVM for problem #1 is shown in Figure 1. The results were obtained using 100 training points. The top plot shows the error of the LS-SVM solution divided by the error in the TFC solution, and the bottom plot shows the error of the LS-SVM solution divided by the error of the CSVM solution. Values greater than one indicate that the compared method is more accurate than the LS-SVM method, and vice-versa for values less than one.

Figure 1 shows that TFC is the most accurate of the three methods followed by CSVM and finally LS-SVM. The error reduction when using CSVM instead of LS-SVM is typically an order of magnitude or less. However, the error reduction when using TFC instead of the other two methods is multiple orders of magnitude. The attentive reader will notice that the plot that includes TFC solution has less data points in Figure 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.

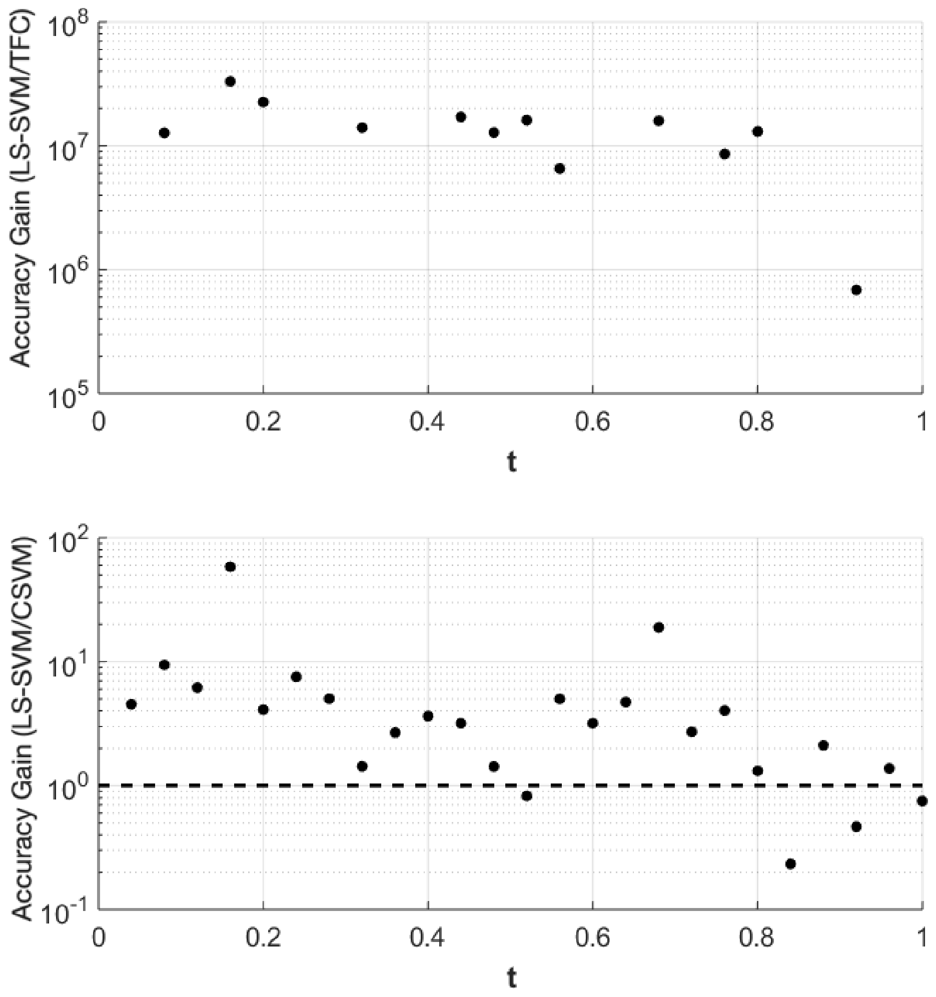


Figure 1. Accuracy gain for the Theory of Functional Connections (TFC) and constrained support vector machine (CSVM) methods over least-squares support vector machines (LS-SVMs) for problem #1 using 100 training points.

Tables A1–A3 in the appendix compare the three methods for various numbers of training points when solving problem #1. Additionally, these tables show that TFC provides the shortest training time and the lowest maximum error and mean square error (MSE) on both the training set and test set. The CSVM results are the slowest, but they are more accurate than the LS-SVM results. The accuracy gained when using CSVM compared to LS-SVM is typically less than an order of magnitude. On the other hand, the accuracy gained when using TFC is multiple orders of magnitude. Moreover, the speed gained when using LS-SVM compared to CSVM is typically less than an order of magnitude, whereas the speed gained when using TFC is approximately one order of magnitude. An accuracy versus speed comparison is shown graphically in Figure 2, where the MSE on the test set is plotted against training time for five specific cases: 8, 16, 32, 50, and 100 training points.

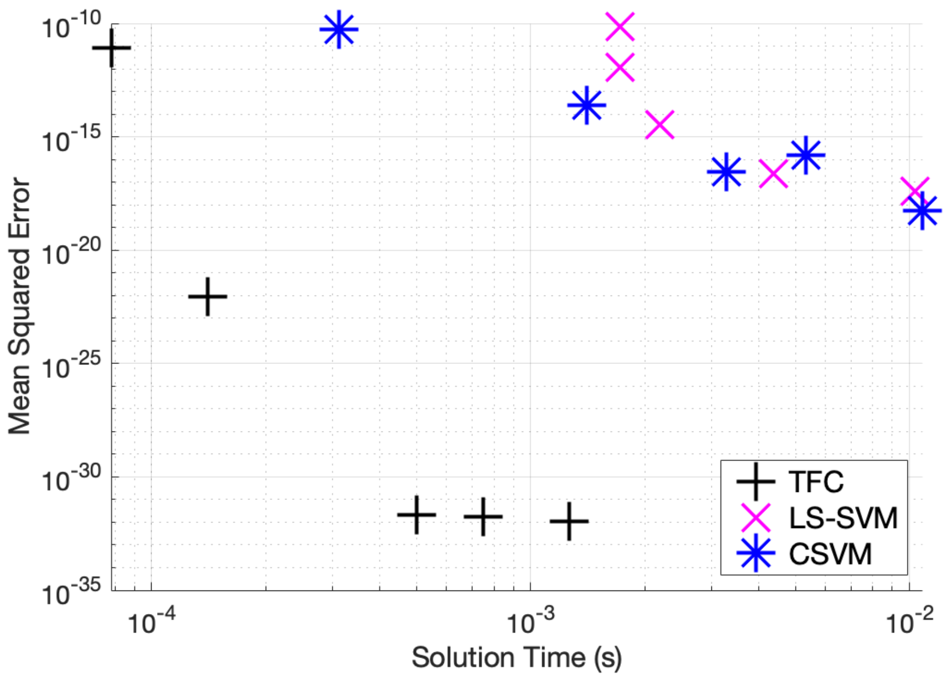


Figure 2. Mean squared error vs. solution time for problem #1.

4.2. Problem #2

Problem #2 is the nonlinear ODE given by,

$$\dot{y} = y^2 + t^2, \quad \text{subject to: } y(0) = 1, \quad t \in [0, 0.5], \tag{12}$$

which has the analytic solution,

$$y(t) = -\frac{t\left(\Gamma\left(\frac{1}{4}\right) J_{-\frac{3}{4}}\left(\frac{t^2}{2}\right) + 2\Gamma\left(\frac{3}{4}\right) J_{\frac{3}{4}}\left(\frac{t^2}{2}\right)\right)}{\Gamma\left(\frac{1}{4}\right) J_{\frac{1}{4}}\left(\frac{t^2}{2}\right) - 2\Gamma\left(\frac{3}{4}\right) J_{-\frac{1}{4}}\left(\frac{t^2}{2}\right)},$$

where Γ is the gamma function defined as,

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} \ \mathrm{d}x$$

and J is Bessel function of first kind defined as,

$$J_\nu(z) = \left(\frac{z}{2}\right)^\nu \sum_{k=0}^{\infty} \frac{\left(\frac{-z^2}{4}\right)^k}{k!\Gamma(\nu + k + 1)}$$

The accuracy gain of TFC and CSVM compared to LS-SVM for problem #2 is shown in Figure 3. This figure was created using 100 training points. The top plot shows the error in the LS-SVM solution divided by the TFC solution. The bottom plot provides the error in the LS-SVM solution divided by the error in the CSVM solution.

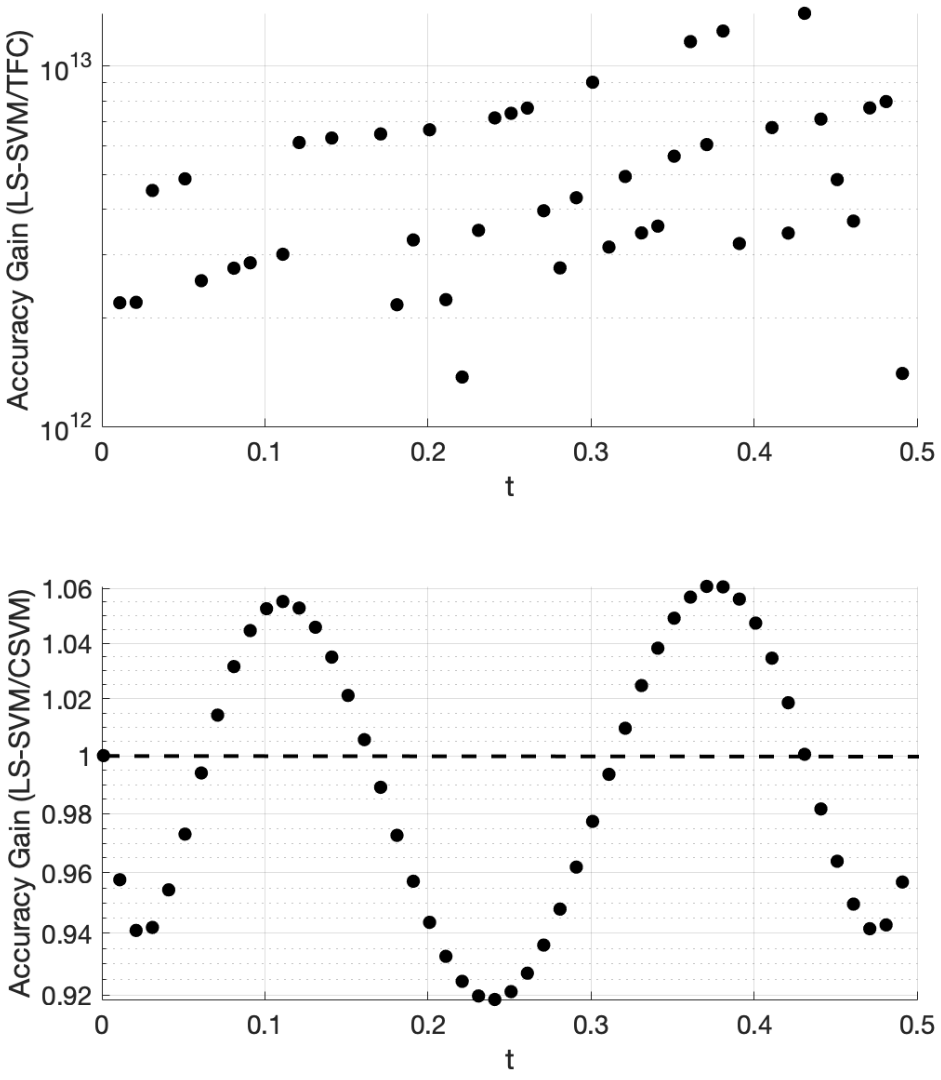


Figure 3. Accuracy gain for TFC and CSVM methods over LS-SVM for problem #2 using 100 training points.

Figure 3 shows that TFC is the most accurate of the two methods with the error being several orders of magnitude lower than the LS-SVM method. It was observed that the difference in accuracy between the CSVM and LS-SVM is negligible. The small variations in accuracy are a function of the specific method. For this problem, the solution accuracy for both methods monotonically decreases as *t* increases; however, the behavior of this decrease is not constant and is at different rates, which produces a sine wave-like plot of the accuracy gain.

Tables A4–A6 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 TFC is faster than using the LS-SVM method for all cases except the second case (using 16 training points). However, the speed gained using TFC is less than one order of magnitude. Furthermore, TFC is more accurate by multiple orders of magnitude as compared to the LS-SVM method over the entire range of test cases. In addition, TFC continues to reduce the MSE and maximum error on the test and training set as more training points are added, whereas the LS-SVM method error increases slightly between 8 and 16 points and then stays approximately the same. The CSVM method follows the same trend as the LS-SVM method; however, it requires more time to train than the LS-SVM method. This is highlighted in an accuracy versus speed comparison, shown graphically in Figure 4, where the MSE on the test set is plotted against training time for five specific cases: 8, 16, 32, 50, and 100 training points.

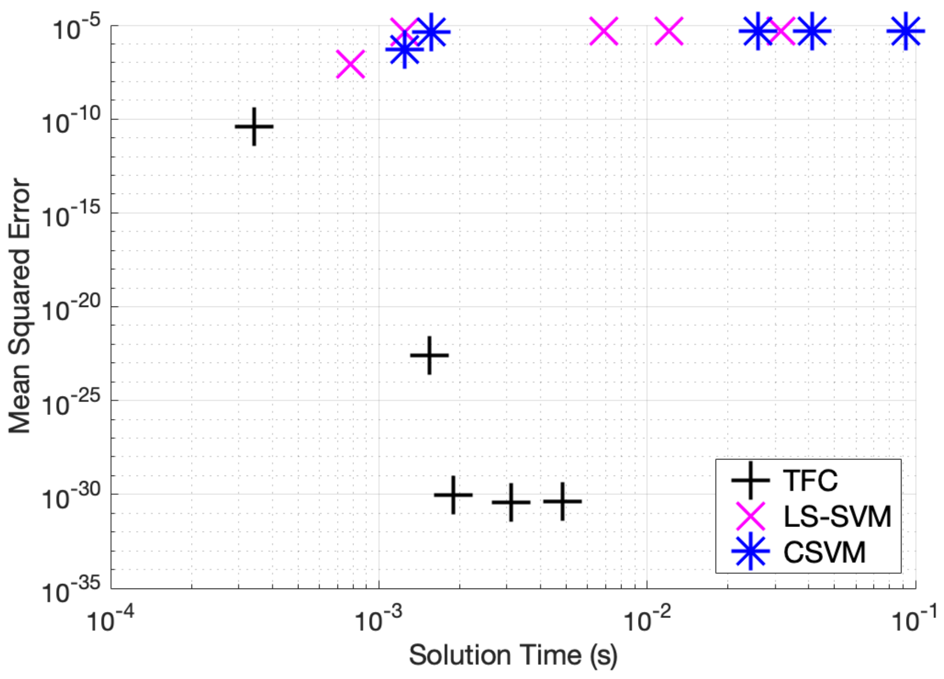


Figure 4. Mean squared error vs. solution time for problem #2.

4.3. Problem #3

Problem #3 is the second order linear ODE given by,

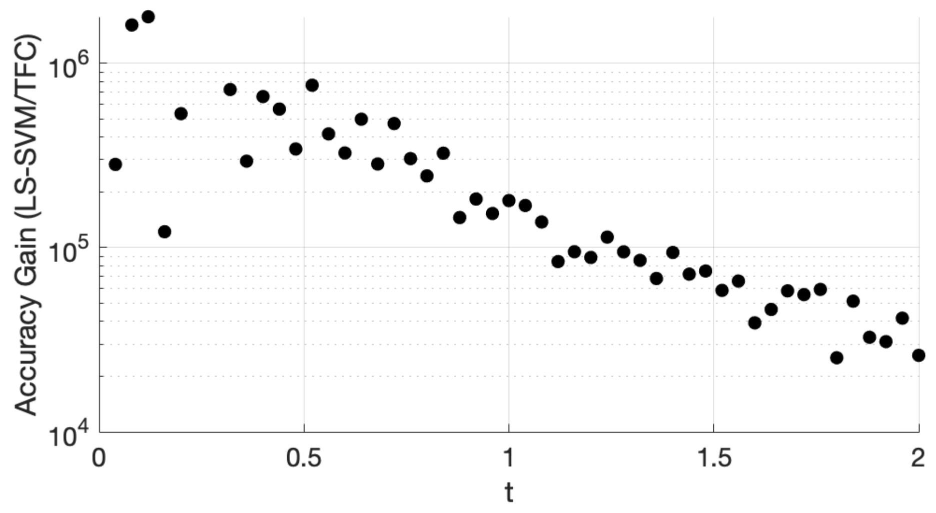
$$\ddot{y} + \frac{1}{5}\dot{y} + y = -\frac{1}{5}\,e^{-t/5}\cos t, \quad \text{subject to: } \begin{cases} y(0) = 1 \\ y'(0) = 1 \end{cases} \qquad t \in [0, 2], \tag{13}$$

which has the analytic solution,

$$y(t) = \frac{\sin(t)}{e^{t/5}}$$

The accuracy gain of TFC and CSVM compared to LS-SVM for problem #3 is shown in Figure 5. The figure was created using 100 training points. The top plot shows the error in the LS-SVM solution divided by the TFC solution, and the bottom plot shows the error in the LS-SVM solution divided by the CSVM solution.

Tables A7–A9 in the appendix compare the two methods for various numbers of training points when solving problem #3. These tables show that solving the DE using TFC is approximately an order of magnitude faster than using the LS-SVM method for all cases. Furthermore, TFC is more accurate than the LS-SVM method for all of the test cases. One interesting note is that when moving from 16 to 32 training points TFC actually loses a bit of accuracy, whereas the LS-SVM method continues to gain accuracy. Despite this, TFC is still multiple orders of magnitude more accurate than the LS-SVM method. Additionally, these tables show that the CSVM method is faster than the LS-SVM for all cases. The speed difference varies from approximately twice as fast to an order of magnitude faster. The LS-SVM and CSVM methods have a similar amount of error, and which method is more accurate depends on how many training points were being used. However, LS-SVM is slightly more accurate than CSVM for more cases than CSVM is slightly more accurate than LS-SVM. An accuracy versus speed comparison is shown graphically in Figure 6, where the MSE on the test set is plotted against training time for five specific cases: 8, 16, 32, 50, and 100 training points.



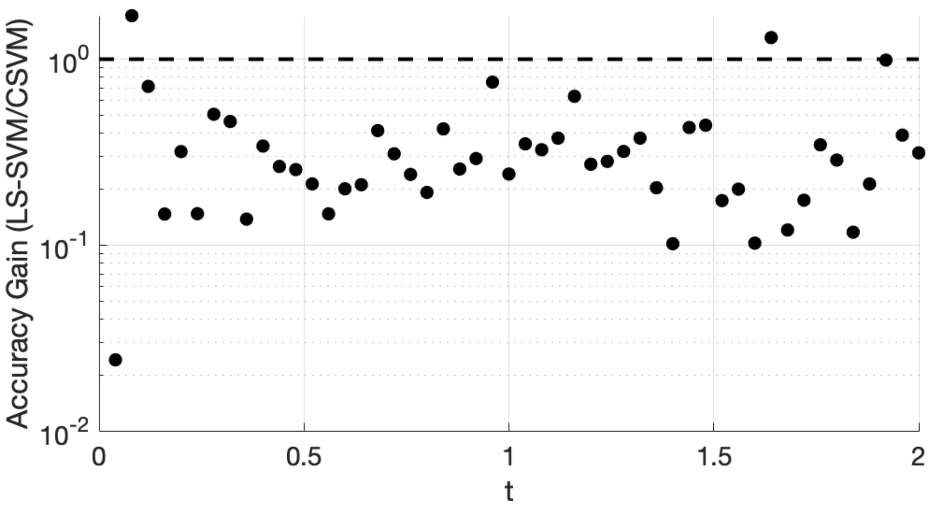


Figure 5. Accuracy gain for TFC and CSVM methods over LS-SVMs for problem #3 using 100 training points.

Figure 5 shows that TFC is the most accurate of the three methods. The TFC error is 4–6 orders of magnitude lower than the LS-SVM method. The LS-SVM method has error that is lower than the error in the CSVM method by an order of magnitude or less.

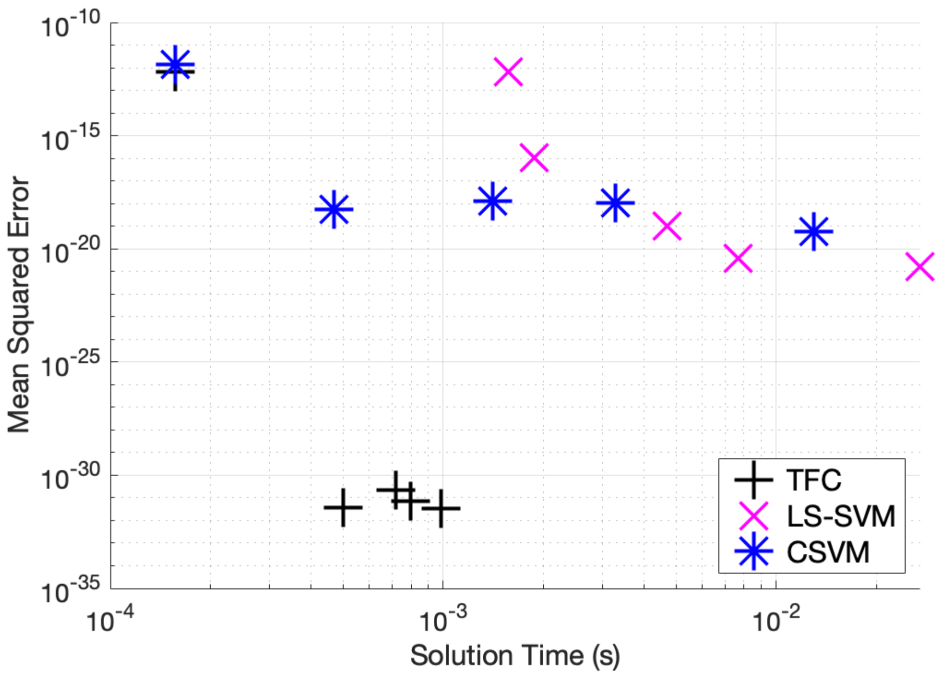


Figure 6. Mean squared error vs. solution time for problem #3 accuracy vs. time.

4.4. Problem #4

Problem #4 is the second order linear PDE on $(x, y) \in [0, 1] \times [0, 1]$ given by,

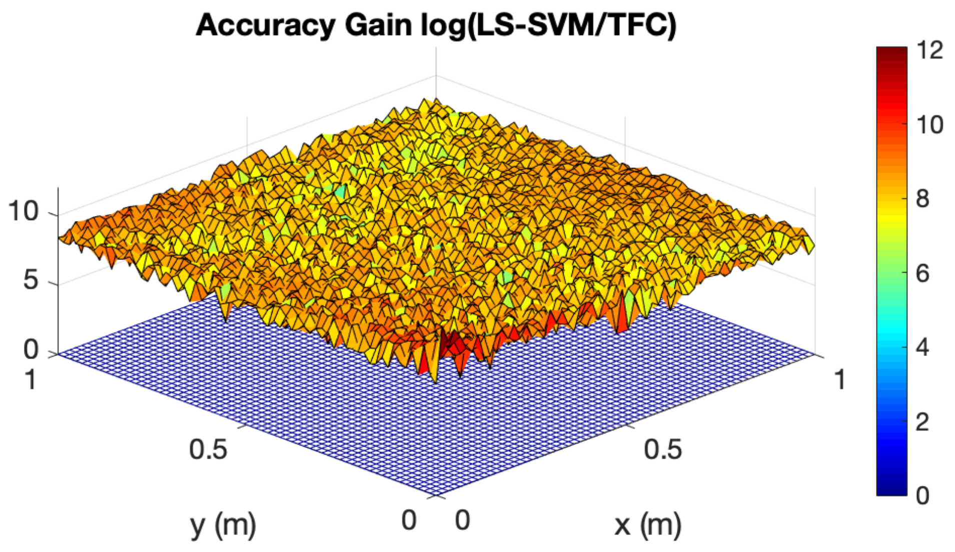
$$\nabla^2 z(x,y) = e^{-x}(x - 2 + y^3 + 6y) \quad \text{subject to:} \begin{cases} z(x,0) = xe^{-x} \\ z(0,y) = y^3 \\ z(x,1) = e^{-x}(x+1) \\ z(1,y) = (1+y^3)e^{-1} \end{cases} \tag{14}$$

which has the analytical solution,

$$z(x,y) = (x + y^3)e^{-x}$$

The accuracy gain of TFC and CSVM compared to LS-SVM for problem #4 is shown in Figure 7. The figure was created using 100 training points in the domain—training points in the domain means training points that do not lie on one of the four boundaries. The top plot shows the log base 10 of the error in the LS-SVM solution divided by the TFC solution, and the bottom plot shows the log base 10 of the error in the LS-SVM solution divided by the error in the CSVM solution.

Figure 7 shows that TFC is the most accurate of the two methods. The TFC error is orders of magnitude lower than the LS-SVM method. The CSVM error is, on average, approximately one order of magnitude lower than the LS-SVM method, but the error is still orders of magnitude higher than the error when using TFC.



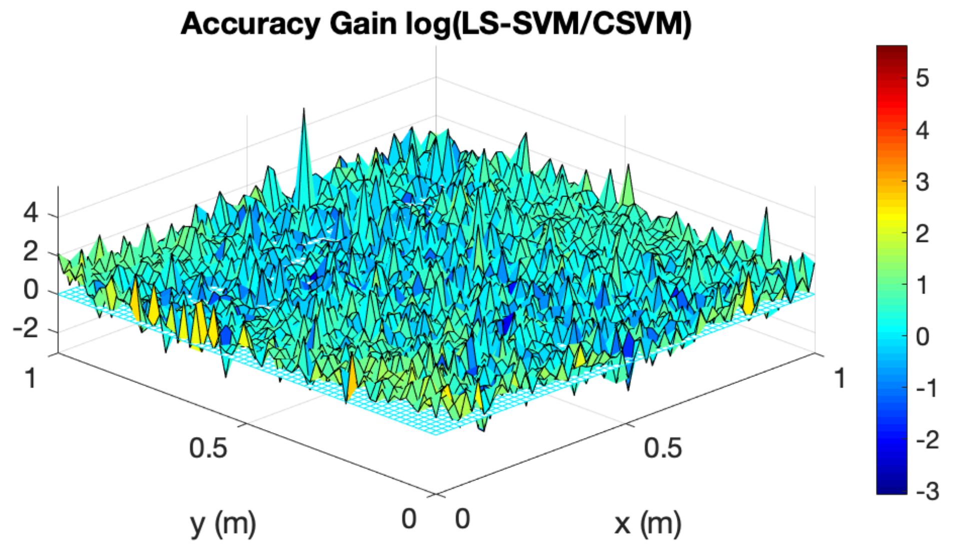


Figure 7. Accuracy gain for TFC and CSVM methods over LS-SVMs for problem #4 using 100 training points in the domain.

Tables A10–A12 in the appendix compare the two methods for various numbers of training points in the domain when solving problem #4. These tables show that solving the DE using TFC is slower than LS-SVM by less than an order of magnitude for all test cases. The MSE error on the test set for TFC is less than LS-SVM for all of the test cases. The amount by which the MSE error on the test set differs between the two methods varies between 7 and 18 orders of magnitude. In addition, these tables show that the training time for CSVM is greater than LS-SVM by approximately an order of magnitude or less. The MSE error on the test set for CSVM is less than the MSE error on the test set for LS-SVM for all the test cases. The amount by which the MSE error on the test set differs between the two methods varies between one and three orders of magnitude. An accuracy versus speed comparison is shown graphically in Figure 8, where the MSE on the test set is plotted against training time for five specific cases: 9, 16, 36, 64, and 100 training points in the domain.

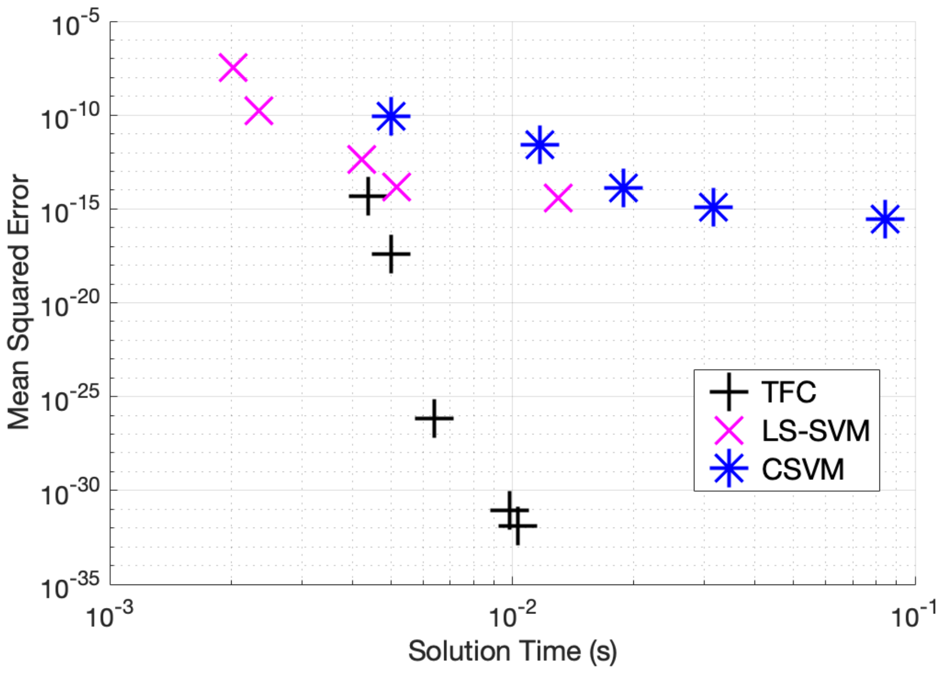


Figure 8. Mean squared error vs. solution time for problem #4 accuracy vs. time.

5. Conclusion

This article presented three methods to solve various types of DEs: TFC, LS-SVM, and CSVM. The CSVM method was a combination of the other two methods; it incorporated LS-SVM into the TFC framework. Four problems were presented that include a linear first order ODE, a nonlinear first order ODE, a linear second order ODE, and a linear second order PDE. The results showed that, in general, TFC is faster, by approximately an order of magnitude or less, and more accurate, by multiple orders of magnitude. The CSVM method has similar performance to the LS-SVM method, but the CSVM method satisfies the boundary constraints exactly whereas the LS-SVM method does not. While the CSVM method underperforms vanilla TFC, it showed the ease with which machine learning algorithms can be incorporated into the TFC framework. This capability is extremely important, as it provides a systematic way to analytically embed many different types of constraints into machine learning algorithms.

This feature will be exploited in future studies and specifically for higher-dimensional PDEs, where the scalability of machine learning algorithms may give a major advantage over the orthogonal basis functions used in TFC. In this article, the authors found that the number of terms when using SVMs may become prohibitive at higher dimensions. Therefore, future work should focus on other machine learning algorithms, such as neural networks, that do not have this issue. 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.

Furthermore, future work should analyze the effect of the regularization term. One observation from the experiments is that the parameter γ is very large, about 10^{13}, making the contribution from $\boldsymbol{w}^{\mathsf{T}}\boldsymbol{w}$ insignificant. However, $\boldsymbol{w}^{\mathsf{T}}\boldsymbol{w}$ is the term meant to provide the best separating boundary surfaces. Going farther in this direction, one could analyze whether applying the kernel trick is beneficial (if the separating margin is not really achieved), or if an expansion similar to Chebyshev polynomials (CP), but using Gaussians, could provide a more accurate solution with a simpler algorithm.

Author Contributions: Conceptualization, C.L., H.J. and L.S.; Software, C.L. and H.J.; Supervision, L.S. and D.M.; Writing original draft, C.L., H.J. and L.S.; Writing review and editing, C.L., H.J. and L.S.

Funding: This work was supported by a NASA Space Technology Research Fellowship, Leake [NSTRF 2019] Grant #: 80NSSC19K1152 and Johnston [NSTRF 2019] Grant #: 80NSSC19K1149.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BVP	boundary-value problem
CP	Chebyshev polynomial
CSVM	constrained support vector machines
DE	differential equation
IVP	initial-value problem
LS	least-squares
LS-SVM	least-squares support vector machines
MSE	mean square error
MVP	multi-value problem
ODE	ordinary differential equation
PDE	partial differential equation
RBF	radial basis function
SVM	support vector machines
TFC	*Theory of Functional Connections*

Appendix A Numerical Data

The rows in Tables A1–A9 correspond to 8, 16, 32, 50, and 100 training points, respectively.

Table A1. TFC 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	7.813×10^{-5}	6.035×10^{-6}	1.057×10^{-11}	6.187×10^{-6}	8.651×10^{-12}	7
16	1.406×10^{-4}	2.012×10^{-11}	1.257×10^{-22}	1.814×10^{-11}	8.964×10^{-23}	17
32	5.000×10^{-4}	2.220×10^{-16}	1.887×10^{-32}	3.331×10^{-16}	2.086×10^{-32}	25
50	7.500×10^{-4}	2.220×10^{-16}	9.368×10^{-33}	2.220×10^{-16}	1.801×10^{-32}	25
100	1.266×10^{-3}	4.441×10^{-16}	1.750×10^{-32}	2.220×10^{-16}	1.138×10^{-32}	26

Table A2. LS-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	γ	σ
8	1.719×10^{-3}	1.179×10^{-5}	5.638×10^{-11}	1.439×10^{-5}	7.251×10^{-11}	5.995×10^{17}	3.162×10^{0}
16	1.719×10^{-3}	1.710×10^{-6}	1.107×10^{-12}	1.849×10^{-6}	1.161×10^{-12}	3.594×10^{15}	6.813×10^{-1}
32	2.188×10^{-3}	9.792×10^{-8}	3.439×10^{-15}	9.525×10^{-8}	3.359×10^{-15}	3.594×10^{15}	3.162×10^{-1}
50	4.375×10^{-3}	1.440×10^{-8}	2.983×10^{-17}	8.586×10^{-9}	2.356×10^{-17}	3.594×10^{15}	3.162×10^{-1}
100	1.031×10^{-2}	3.671×10^{-9}	3.781×10^{-18}	3.673×10^{-9}	3.947×10^{-18}	2.154×10^{13}	3.162×10^{-1}

Table A3. 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	γ	σ
8	3.125×10^{-4}	1.018×10^{-5}	4.131×10^{-11}	1.357×10^{-5}	5.547×10^{-11}	2.154×10^{13}	3.162×10^{0}
16	1.406×10^{-3}	2.894×10^{-7}	2.588×10^{-14}	2.818×10^{-7}	2.468×10^{-14}	5.995×10^{17}	6.813×10^{-1}
32	5.313×10^{-3}	2.283×10^{-8}	1.355×10^{-16}	2.576×10^{-8}	1.494×10^{-16}	3.594×10^{15}	3.162×10^{-1}
50	3.281×10^{-3}	8.887×10^{-9}	2.055×10^{-17}	1.072×10^{-8}	2.783×10^{-17}	7.743×10^{8}	3.162×10^{-1}
100	1.078×10^{-2}	2.230×10^{-9}	5.571×10^{-19}	2.163×10^{-9}	5.337×10^{-19}	3.594×10^{15}	1.468×10^{-1}

Table A4. TFC 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	3.437×10^{-4}	8.994×10^{-6}	2.242×10^{-11}	1.192×10^{-5}	4.132×10^{-11}	8
16	1.547×10^{-3}	4.586×10^{-12}	6.514×10^{-24}	9.183×10^{-12}	2.431×10^{-23}	16
32	1.891×10^{-3}	3.109×10^{-15}	9.291×10^{-31}	4.885×10^{-15}	9.590×10^{-31}	32
50	3.125×10^{-3}	1.110×10^{-15}	2.100×10^{-31}	2.665×10^{-15}	3.954×10^{-31}	32
100	4.828×10^{-3}	1.776×10^{-15}	3.722×10^{-31}	2.665×10^{-15}	4.321×10^{-31}	32

Table A5. LS-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	γ	σ
8	7.813×10^{-4}	1.001×10^{-3}	1.965×10^{-7}	1.001×10^{-3}	7.904×10^{-8}	1.000×10^{10}	3.704×10^{-1}
16	1.250×10^{-3}	4.017×10^{-3}	4.909×10^{-6}	3.872×10^{-3}	4.514×10^{-6}	1.000×10^{10}	4.198×10^{-1}
32	6.875×10^{-3}	4.046×10^{-3}	4.834×10^{-6}	3.900×10^{-3}	4.575×10^{-6}	1.000×10^{10}	4.536×10^{-1}
50	1.203×10^{-2}	4.048×10^{-3}	4.792×10^{-6}	3.902×10^{-3}	4.580×10^{-6}	1.000×10^{10}	4.666×10^{-1}
100	3.156×10^{-2}	4.050×10^{-3}	4.752×10^{-6}	3.903×10^{-3}	4.582×10^{-6}	1.000×10^{10}	4.853×10^{-1}

Table A6. CSVM 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	γ	σ
8	1.250×10^{-3}	1.556×10^{-3}	7.644×10^{-7}	1.480×10^{-3}	5.325×10^{-7}	1.000×10^{10}	3.452×10^{-1}
16	1.563×10^{-3}	4.021×10^{-3}	4.914×10^{-6}	3.876×10^{-3}	4.517×10^{-6}	1.000×10^{10}	4.719×10^{-1}
32	2.594×10^{-2}	4.047×10^{-3}	4.834×10^{-6}	3.901×10^{-3}	4.575×10^{-6}	1.000×10^{10}	5.109×10^{-1}
50	4.109×10^{-2}	4.050×10^{-3}	4.792×10^{-6}	3.903×10^{-3}	4.580×10^{-6}	1.000×10^{10}	5.252×10^{-1}
100	9.219×10^{-2}	4.051×10^{-3}	4.753×10^{-6}	3.904×10^{-3}	4.583×10^{-6}	1.000×10^{10}	5.469×10^{-1}

Table A7. TFC 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	m
8	1.563×10^{-4}	1.313×10^{-6}	5.184×10^{-13}	1.456×10^{-6}	6.818×10^{-13}	8
16	7.969×10^{-4}	5.551×10^{-16}	6.123×10^{-32}	8.882×10^{-16}	7.229×10^{-32}	15
32	7.187×10^{-4}	1.221×10^{-15}	2.377×10^{-31}	9.992×10^{-16}	2.229×10^{-31}	15
50	5.000×10^{-4}	7.772×10^{-16}	3.991×10^{-32}	5.551×10^{-16}	3.672×10^{-32}	15
100	9.844×10^{-4}	7.772×10^{-16}	5.525×10^{-32}	6.661×10^{-16}	3.518×10^{-32}	15

Table A8. LS-SVM 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	γ	σ
8	1.563×10^{-3}	1.420×10^{-6}	8.300×10^{-13}	1.638×10^{-6}	6.522×10^{-13}	5.995×10^{17}	6.813×10^{0}
16	1.875×10^{-3}	1.811×10^{-8}	1.015×10^{-16}	1.871×10^{-8}	1.014×10^{-16}	3.594×10^{15}	3.162×10^{0}
32	4.687×10^{-3}	5.455×10^{-10}	1.025×10^{-19}	9.005×10^{-10}	1.015×10^{-19}	5.995×10^{17}	1.468×10^{0}
50	7.656×10^{-3}	8.563×10^{-11}	3.771×10^{-21}	8.391×10^{-11}	3.646×10^{-21}	2.154×10^{13}	1.468×10^{0}
100	2.688×10^{-2}	6.441×10^{-11}	1.500×10^{-21}	6.128×10^{-11}	1.640×10^{-21}	2.154×10^{13}	1.468×10^{0}

Table A9. CSVM 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	γ	σ
8	1.563×10^{-4}	1.263×10^{-6}	7.737×10^{-13}	2.017×10^{-6}	1.339×10^{-12}	1.000×10^{20}	6.813×10^{0}
16	4.687×10^{-4}	1.269×10^{-9}	4.961×10^{-19}	1.631×10^{-9}	5.342×10^{-19}	3.594×10^{15}	3.162×10^{0}
32	1.406×10^{-3}	1.763×10^{-9}	8.308×10^{-19}	2.230×10^{-9}	1.248×10^{-18}	3.594×10^{15}	3.162×10^{0}
50	3.281×10^{-3}	1.429×10^{-9}	1.045×10^{-18}	1.569×10^{-9}	1.017×10^{-18}	2.154×10^{13}	1.468×10^{0}
100	1.297×10^{-2}	8.261×10^{-10}	8.832×10^{-20}	7.209×10^{-10}	5.589×10^{-20}	2.154×10^{13}	1.468×10^{0}

The rows in Tables A10–A12 correspond to 9, 16, 35, 64, and 100 training points, respectively.

Table A10. TFC results for problem #4.

Number of Training Points in Domain	Training Time (s)	Maximum Error on Training Set	MSE on Training Set	Maximum Error on Test Set	MSE on Test Set	m
9	4.375×10^{-3}	1.107×10^{-7}	1.904×10^{-15}	1.543×10^{-7}	4.633×10^{-15}	8
16	5.000×10^{-3}	3.336×10^{-9}	2.131×10^{-18}	4.938×10^{-9}	3.964×10^{-18}	9
36	6.406×10^{-3}	6.628×10^{-14}	5.165×10^{-28}	2.333×10^{-13}	6.961×10^{-27}	12
64	9.844×10^{-3}	4.441×10^{-16}	2.091×10^{-32}	8.882×10^{-16}	8.320×10^{-32}	15
100	1.031×10^{-2}	3.331×10^{-16}	1.229×10^{-32}	6.661×10^{-16}	1.246×10^{-32}	15

Table A11. LS-SVM results for problem #4.

Number of Training Points in Domain	Training Time (s)	Maximum Error on Training Set	MSE on Training Set	Maximum Error on Test Set	MSE on Test Set	γ	σ
9	2.031×10^{-3}	2.578×10^{-4}	9.984×10^{-9}	3.941×10^{-4}	3.533×10^{-8}	1.000×10^{14}	6.635×10^{0}
16	2.344×10^{-3}	2.229×10^{-5}	6.277×10^{-11}	3.794×10^{-5}	1.731×10^{-10}	1.000×10^{14}	3.577×10^{0}
36	4.219×10^{-3}	1.254×10^{-6}	2.542×10^{-13}	2.435×10^{-6}	4.517×10^{-13}	1.000×10^{14}	1.894×10^{0}
64	5.156×10^{-3}	2.916×10^{-7}	1.193×10^{-14}	4.962×10^{-7}	1.390×10^{-14}	1.000×10^{14}	1.589×10^{0}
100	1.297×10^{-2}	1.730×10^{-7}	3.028×10^{-15}	2.673×10^{-7}	3.668×10^{-15}	1.000×10^{14}	9.484×10^{-1}

Table A12. CSVM results for problem #4.

Number of Training Points in Domain	Training Time (s)	Maximum Error on Training Set	MSE on Training Set	Maximum Error on Test Set	MSE on Test Set	γ	σ
9	5.000×10^{-3}	1.305×10^{-5}	1.936×10^{-11}	3.325×10^{-5}	8.262×10^{-11}	1.000×10^{14}	6.948×10^{0}
16	1.172×10^{-2}	2.121×10^{-6}	7.965×10^{-13}	5.507×10^{-6}	2.530×10^{-12}	1.000×10^{14}	4.894×10^{0}
36	1.891×10^{-2}	2.393×10^{-7}	6.242×10^{-15}	3.738×10^{-7}	1.341×10^{-14}	1.000×10^{14}	2.154×10^{0}
64	3.156×10^{-2}	9.501×10^{-8}	1.021×10^{-15}	1.251×10^{-7}	1.165×10^{-15}	1.000×10^{14}	1.371×10^{0}
100	8.453×10^{-2}	4.362×10^{-8}	2.687×10^{-16}	5.561×10^{-8}	2.951×10^{-16}	1.000×10^{14}	8.891×10^{-1}

Appendix B Nonlinear ODE LS-SVM and CSVM Derivation

This appendix shows how the method of Lagrange multipliers is used to solve nonlinear ODEs using the LS-SVM and CSVM methods. Equation (A1) shows the Lagrangian for the LS-SVM method. The values where $\mathcal{L}$ are zero give candidates for the minimum.

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{e}, \boldsymbol{y}, \boldsymbol{\alpha}, \beta, \boldsymbol{\eta}) = \frac{1}{2}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + \gamma \boldsymbol{e}^{\mathrm{T}}\boldsymbol{e}) &- \sum_{i=1}^{N} \alpha_i \left[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}'(t_i) - f(t_i, y_i) - e_i\right] - \beta[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(t_0) + b - y_0] \\ &- \sum_{i=1}^{N} \eta_i \left[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(t_i) + b - y_i\right]\end{aligned} \tag{A1}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \mathbf{0} \quad &\rightarrow \quad \boldsymbol{w} = \sum_{i=1}^{N} \alpha_i \boldsymbol{\varphi}'(t_i) + \sum_{i=1}^{N} \eta_i \boldsymbol{\varphi}(t_i) + \beta \boldsymbol{\varphi}(t_0) \\
\frac{\partial \mathcal{L}}{\partial e_i} = 0 \quad &\rightarrow \quad \gamma e_i = -\alpha_i \\
\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \quad &\rightarrow \quad \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}'(t_i) = f(t_i, y_i) + e_i \\
\frac{\partial \mathcal{L}}{\partial \eta_i} = 0 \quad &\rightarrow \quad y_i = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(t_i) + b \\
\frac{\partial \mathcal{L}}{\partial \beta} = 0 \quad &\rightarrow \quad \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\varphi}(t_0) + b = y_0 \\
\frac{\partial \mathcal{L}}{\partial b} = 0 \quad &\rightarrow \quad \beta + \sum_{i=1}^{N} \eta_i = 0 \\
\frac{\partial \mathcal{L}}{\partial y_i} = 0 \quad &\rightarrow \quad \alpha_i f_y(t_i, y_i) + \eta_i = 0
\end{aligned}$$

A system of equations can be set up by substituting the results found by differentiating $\mathcal{L}$ with respect to $\boldsymbol{w}$ 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 α_i, η_i, y_i, β, and b. This system of equations is given in Equation (A2),

$$\begin{aligned}
&\sum_{j=1}^{N} \alpha_j \boldsymbol{\varphi}'(t_j)^{\mathrm{T}}\boldsymbol{\varphi}'(t_i) + \sum_{j=1}^{N} \eta_j \boldsymbol{\varphi}(t_j)^{\mathrm{T}}\boldsymbol{\varphi}'(t_i) + \beta \boldsymbol{\varphi}(t_0)^{\mathrm{T}}\boldsymbol{\varphi}'(t_i) + \frac{\alpha_i}{\gamma} = f(t_i, y_i) \\
&\sum_{j=1}^{N} \alpha_j \boldsymbol{\varphi}'(t_j)^{\mathrm{T}}\boldsymbol{\varphi}(t_i) + \sum_{j=1}^{N} \eta_j \boldsymbol{\varphi}(t_j)^{\mathrm{T}}\boldsymbol{\varphi}(t_i) + \beta \boldsymbol{\varphi}(t_0)^{\mathrm{T}}\boldsymbol{\varphi}(t_i) + b - y_i = 0 \\
&\sum_{j=1}^{N} \alpha_j \boldsymbol{\varphi}'(t_j)^{\mathrm{T}}\boldsymbol{\varphi}(t_0) + \sum_{j=1}^{N} \eta_j \boldsymbol{\varphi}(t_j)^{\mathrm{T}}\boldsymbol{\varphi}(t_0) + \beta \boldsymbol{\varphi}(t_0)^{\mathrm{T}}\boldsymbol{\varphi}(t_0) + b = y_0 \\
&\beta + \sum_{i=j}^{N} \eta_j = 0 \\
&\alpha_i f_y(t_i, y_i) + \eta_i = 0
\end{aligned} \tag{A2}$$

where $i = 1, \ldots, N$. The system of equations given in Equation (A2) is the same as the system of equations in Equation (20) of reference [12] with one exception: The regularization term, I/γ, in the second row of the second column entry is 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 problem. Once the set of equations has been solved, the model solution is given in the dual form by,

$$\begin{aligned}\hat{y}(t) =& \sum_{i=1}^{N} \alpha_i \boldsymbol{\varphi}'(t_i)^{\mathrm{T}} \boldsymbol{\varphi}(t) + \sum_{i=1}^{N} \eta_i \boldsymbol{\varphi}(t_i)^{\mathrm{T}} \boldsymbol{\varphi}(t) + \\ &+ \beta \boldsymbol{\varphi}(t_0)^{\mathrm{T}} \boldsymbol{\varphi}(t) + b.\end{aligned}$$

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. The method for solving the nonlinear ODEs with CSVM is the same, except the Lagrangian function is,

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}, \boldsymbol{e}, \boldsymbol{y}, \boldsymbol{\alpha}, \boldsymbol{\eta}) =& \frac{1}{2}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + \gamma \boldsymbol{e}^{\mathrm{T}}\boldsymbol{e}) - \sum_{i=1}^{N} \alpha_i \left[\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\varphi}'(t_i) - f(t_i, y_i) - e_i\right] \\ &- \sum_{i=1}^{N} \eta_i \left[\boldsymbol{w}^{\mathrm{T}}(\boldsymbol{\varphi}(t_i) - \varphi(t_0)) + y_0 - y_i\right],\end{aligned}$$

where the initial value constraint has been embedded via TFC by taking the primal form of the solution to be,

$$\hat{y}(t) = \boldsymbol{w}^{\mathrm{T}}(\boldsymbol{\varphi}(t_i) - \varphi(t_0)) + y_0.$$

Similar to the SVM derivation, taking the gradients of $\mathcal{L}$ and setting them equal to zero leads to a system of equations,

$$\begin{aligned}&\sum_{j=1}^{N} \alpha_j \boldsymbol{\varphi}'(t_j)^{\mathrm{T}} \boldsymbol{\varphi}'(t_i) + \sum_{j=1}^{N} \eta_j \Big(\boldsymbol{\varphi}(t_j) - \boldsymbol{\varphi}(t_0)\Big)^{\mathrm{T}} \boldsymbol{\varphi}\prime(t_i) - f(t_i, y_i) + \alpha_i/\gamma = 0 \\ &\sum_{j=1}^{N} \alpha_j \boldsymbol{\varphi}'(t_j)^{\mathrm{T}} \Big(\boldsymbol{\varphi}(t_i) - \boldsymbol{\varphi}(t_0)\Big) + \sum_{j=1}^{N} \eta_j \Big(\boldsymbol{\varphi}(t_j) - \boldsymbol{\varphi}(t_0)\Big)^{\mathrm{T}} \Big(\boldsymbol{\varphi}(t_i) - \boldsymbol{\varphi}(t_0)\Big) + y_0 - y_i = 0 \\ &\alpha_i f_y(t_i, y_i) + \eta_i = 0\end{aligned}$$

where $i = 1, ..., N$. This can be solved using Newton's method for the unknowns α_i, η_i, and y_i. In addition, the gradients can be used to re-write the estimated solution, $\hat{y}$, in the dual form,

$$\hat{y}(t) = \sum_{i=1}^{N} \alpha_i \boldsymbol{\varphi}'(t_i)^{\mathrm{T}} \Big(\boldsymbol{\varphi}(t) - \boldsymbol{\varphi}(t_0)\Big) + \sum_{i=1}^{N} \eta_i \Big(\boldsymbol{\varphi}(t_i) - \boldsymbol{\varphi}(t_0)\Big)^{\mathrm{T}} \Big(\boldsymbol{\varphi}(t) - \boldsymbol{\varphi}(t_0)\Big) + y_0.$$

As with the SVM derivation, the system of equations that must be solved and the dual form of the estimated solution can each be written in terms of the kernel matrix and its derivatives.

Appendix C Linear PDE CSVM Derivation

This appendix shows how to solve the PDE given by,

$$\nabla^2 z(x, y) = f(x, y) \quad \text{subject to:} \begin{cases} z(x, 0) = c_1(x, 0) \\ z(0, y) = c_2(0, y) \\ z(x, 1) = c_3(x, 1) \\ z(1, y) = c_4(1, y) \end{cases}$$

using the CSVM method. Note, that this is the same PDE as shown in problem number four of the numerical results section where the right-hand side of the PDE has been replaced by a more general function $f(x, y)$ and the boundary-value type constraints have been replaced by more general functions $c_k(x, y)$ where $k = 1, ..., 4$. Note, that throughout this section all matrices will be written using tensor notation rather than vector-matrix form for compactness. In this article, we will let superscripts denote a derivative with respect to the superscript variable and a subscript will be a normal tensor index. For example, the symbol A_{ij}^{xx} would denote a second-order derivative of the second-order tensor A_{ij} with respect to the variable x (i.e. $\frac{\partial^2 A_{ij}}{\partial x^2}$).

Using the Multivariate TFC [23], the constrained expression for this problem can be written as,

$$
\begin{aligned}
\hat{z}(x,y) &= A_{ij}v_iv_j + w_j\varphi_j(x,y) - w_kB_{ijk}v_iv_j \\
A_{ij} &= \begin{bmatrix} 0 & c_1(x,0) & c_3(x,1) \\ c_2(0,y) & -c_1(0,0) & -c_3(0,1) \\ c_4(1,y) & -c_1(1,0) & -c_3(1,1) \end{bmatrix} \\
B_{ijk} &= \begin{bmatrix} 0 & \varphi_k(x,0) & \varphi_k(x,1) \\ \varphi_k(0,y) & -\varphi_k(0,0) & -\varphi_k(0,1) \\ \varphi_k(1,y) & -\varphi_k(1,0) & -\varphi_k(1,1) \end{bmatrix} \\
v_i &= \begin{bmatrix} 1 & 1-x & x \end{bmatrix} \\
v_j &= \begin{bmatrix} 1 & 1-y & y \end{bmatrix}.
\end{aligned}
$$

where $\hat{z}$ will satisfy the boundary constraints $c_k(x,y)$ regardless of the choice of w and φ_k. Now, the Lagrange multipliers are added in to form $\mathcal{L}$,

$$\mathcal{L}(\boldsymbol{w},\boldsymbol{\alpha},\boldsymbol{e}) = \frac{1}{2}w_iw_i + \frac{\gamma}{2}e_ie_i - \alpha_I(\hat{z}_I^{xx} + \hat{z}_I^{yy} - f_I - e_I),$$

where $\hat{z}_I$ is the vector composed of the elements $\hat{z}(x_n,y_n)$ where $n = 1,...,N_p$ and there are N_p training points. The gradients of $\mathcal{L}$ give candidates for the minimum,

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_k} &= w_k - \alpha_I(\varphi_{Ik}^{xx} - B_{Iijk}^{xx}v_iv_j + \varphi_{Ik}^{yy} - B_{Iijk}^{yy}v_iv_j) = 0 \\
\frac{\partial \mathcal{L}}{\alpha_k} &= \hat{z}_I^{xx} + \hat{z}_I^{yy} - f_I - e_I = 0 \\
\frac{\partial \mathcal{L}}{e_k} &= \frac{\gamma}{2}e_k - \alpha_k = 0,
\end{aligned}
$$

where φ_{Ik} is the second order tensor composed of the vectors $\varphi_i(x_n,y_n)$, B_{Iijk} is the fourth order tensor composed of the third order tensors $B(x_n,y_n)_{ijk}$, and $n = 1,...,N_p$. The gradients of $\mathcal{L}$ can be used to form a system of simultaneous linear equations to solve for the unknowns and write $\hat{z}$ in the dual form. The system of simultaneous linear equations is,

$$\mathcal{A}_{IJ}\alpha_J = \mathcal{B}_I$$

$$
\begin{aligned}
\mathcal{A}_{IJ} =& \varphi_{Ik}^{xx}\varphi_{Jk}^{xx} - \varphi_{Ik}^{xx}B_{Jijk}^{xx}v_iv_j + \varphi_{Ik}^{xx}\varphi_{Jk}^{yy} - \varphi_{Ik}^{xx}B_{Jijk}^{yy}v_iv_j - B_{Iijk}^{xx}v_iv_j\varphi_{Jk}^{xx} + B_{Iijk}^{xx}v_iv_jB_{Jmnk}^{xx}v_mv_n \\
&- B_{Iijk}^{xx}v_iv_j\varphi_{Jk}^{yy} + B_{Iijk}^{xx}v_iv_jB_{Jmnk}^{yy}v_mv_n + \varphi_{Ik}^{yy}\varphi_{Jk}^{xx} - \varphi_{Ik}^{yy}B_{Jijk}^{xx}v_iv_j + \varphi_{Ik}^{yy}\varphi_{Jk}^{yy} - \varphi_{Ik}^{yy}B_{Jijk}^{yy}v_iv_j \\
&- B_{Iijk}^{yy}v_iv_j\varphi_{Jk}^{xx} + B_{Iijk}^{yy}v_iv_jB_{Jmnk}^{xx}v_mv_n - B_{Iijk}^{yy}v_iv_j\varphi_{Jk}^{yy} + B_{Iijk}^{yy}v_iv_jB_{Jmnk}^{yy}v_mv_n + \frac{1}{\gamma}\delta_{IJ}
\end{aligned}
$$

$$\mathcal{B}_I = f_I - A_{Iij}^{xx}v_iv_j - A_{Iij}^{yy}v_iv_j$$

where $v_m = v_i$, $v_n = v_j$, and A_{Iijk} is the fourth order tensor composed of the third order tensors $A(x_n, y_n)_{ijk}$ where $n = 1, ..., N_p$. The dual-form of the solution is,

$$\begin{aligned}\hat{z}(x,y) =&A_{ij}v_iv_j + \alpha_I\left[\varphi^{xx}_{Ik}\varphi(x,y)_k - B^{xx}_{Iijk}v_iv_j\varphi_k(x,y) + \varphi^{yy}_{Ik}\varphi_k(x,y) - B^{yy}_{Iijk}v_iv_j\varphi_k(x,y)\right]\\ &- \alpha_I\left[\varphi^{xx}_{Ik}B_{ijk}v_iv_j - B^{xx}_{Iijk}v_iv_jB_{mnk}v_mv_n + \varphi^{yy}_{Ik}B_{ijk}v_iv_j - B^{yy}_{Iijk}v_iv_jB_{mnk}v_mv_n\right].\end{aligned}$$

The system of simulatenous linear equations as well as the dual form of the solution can be written and were solved using the kernel matrix and its partial derivatives.

References

1. Dormand, J.; Prince, P. A Family of Embedded Runge-Kutta Formulae. *J. Comp. Appl. Math.* **1980**, *6*, 19–26. [CrossRef]
2. Berry, M.M.; Healy, L.M. Implementation of Gauss-Jackson integration for orbit propagation. *J. Astronaut. Sci.* **2004**, *52*, 351–357.
3. Bai, X.; Junkins, J.L. Modified Chebyshev-Picard Iteration Methods for Orbit Propagation. *J. Astronaut. Sci.* **2011**, *58*, 583–613. [CrossRef]
4. Junkins, J.L.; Younes, A.B.; Woollands, R.; Bai, X. Picard Iteration, Chebyshev Polynomials, and Chebyshev Picard Methods: Application in Astrodynamics. *J. Astronaut. Sci.* **2015**, *60*, 623–653. [CrossRef]
5. Reed, J.; Younes, A.B.; Macomber, B.; Junkins, J.L.; Turner, D.J. State Transition Matrix for Perturbed Orbital Motion using Modified Chebyshev Picard Iteration. *J. Astronaut. Sci.* **2015**, *6*, 148–167. [CrossRef]
6. Driscoll, T.A.; Hale, N. Rectangular spectral collocation. *IMA J. Numer. Anal.* **2016**, *36*, 108–132. [CrossRef]
7. Mortari, D. The Theory of Connections: Connecting Points. *Mathematics* **2017**, *5*, 57. [CrossRef]
8. Mortari, D. Least-squares Solutions of Linear Differential Equations. *Mathematics* **2017**, *5*, 48. [CrossRef]
9. Mortari, D.; Johnston, H.; Smith, L. High accuracy least-squares solutions of nonlinear differential equations. *J. Comput. Appl. Math.* **2019**, *352*, 293–307. [CrossRef]
10. Johnston, H.; Mortari, D. Linear Differential Equations Subject to Relative, Integral, and Infinite Constraints. In Proceedings of the 2018 AAS/AIAA Astrodynamics Specialist Conference, Snowbird, UT, USA, 19–23 August 2018.
11. Johnston, H.; Leake, C.; Efendiev, Y.; Mortari, D. Selected Applications of the Theory of Connections: A Technique for Analytical Constraint Embedding. *Mathematics* **2019**, *7*, 537. [CrossRef]
12. Mehrkanoon, S.; Falck, T.; Johan, A.K. Approximate Solutions to Ordinary Differential Equations using Least-squares Support Vector Machines. *IEEE Trans. Neural Netw. Learn. Syst.* **2012**, *23*, 1356–1367. [CrossRef] [PubMed]
13. Freire, R.Z.; Santos, G.H.d.; Coelho, L.d.S. Hygrothermal Dynamic and Mould Growth Risk Predictions for Concrete Tiles by Using Least Squares Support Vector Machines. *Energies* **2017**, *10*, 1093. [CrossRef]
14. Zhao, X.; Chen, X.; Xu, Y.; Xi, D.; Zhang, Y.; Zheng, X. An EMD-Based Chaotic Least Squares Support Vector Machine Hybrid Model for Annual Runoff Forecasting. *Water* **2017**, *9*, 153. [CrossRef]
15. Gedik, N. Least Squares Support Vector Mechanics to Predict the Stability Number of Rubble-Mound Breakwaters. *Water* **2018**, *10*, 1452. [CrossRef]
16. Gao, C.; Xue, W.; Ren, Y.; Zhou, Y. Numerical Control Machine Tool Fault Diagnosis Using Hybrid Stationary Subspace Analysis and Least Squares Support Vector Machine with a Single Sensor. *Appl. Sci.* **2017**, *7*. [CrossRef]
17. Vapnik, V.N. *Statistical Learning Theory*; Wiley: Hoboken, NJ, USA, 1998.
18. Kramer, M.A.; Thompson, M.L.; Bhagat, P.M. Embedding Theoretical Models in Neural Networks. In Proceedings of the 1992 American Control Conference, Chicago, IL, USA, 24–26 June 1992; pp. 475–479.
19. Pathak, D.; Krähenbühl, P.; Darrell, T. Constrained Convolutional Neural Networks for Weakly Supervised Segmentation. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 11–18 December 2015; pp. 1796–1804.
20. Márquez-Neila, P.; Salzmann, M.; Fua, P. Imposing Hard Constraints on Deep Networks: Promises and Limitations. *arXiv* **2017**, arXiv:1706.02025.

21. Lanczos, C. Applied Analysis. In *Progress in Industrial Mathematics at ECMI 2008*; Dover Publications, Inc.: New York, NY, USA, 1957; Chapter 7, p. 504.
22. Wright, K. Chebyshev Collocation Methods for Ordinary Differential Equations. *Comput. J.* **1964**, *6*, 358–365. [CrossRef]
23. Mortari, D.; Leake, C. The Multivariate Theory of Connections. *Mathematics* **2019**, *7*, 296. [CrossRef]
24. Leake, C.; Mortari, D. An Explanation and Implementation of Multivariate Theory of Functional Connections via Examples. In Proceedings of the 2019 AAS/AIAA Astrodynamics Specialist Conference, Portland, ME, USA, 11–15 August 2019.
25. Theodoridis, S.; Koutroumbas, K. *Pattern Recognition*; Academic Press: Cambridge, MA, USA, 2008.
26. Mehrkanoon, S.; Suykens, J.A. LS-SVM Approximate Solution to Linear Time Varying Descriptor Systems. *Automatica* **2012**, *48*, 2502–2511. [CrossRef]
27. Mehrkanoon, S.; Suykens, J. Learning Solutions to Partial Differential Equations using LS-SVM. *Neurocomputing* **2015**, *159*, 105–116. [CrossRef]