Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 56-3 (2023) 37–42

Subspace Structured Neural Network for Rapid Trajectory Optimization

Luis Tituaña*, Yunjun Xu*

*Mechanical and Aerospace Engineering, University of Central Florida, Orlando, FL 32826 USA
(Tel: 407-823-1745; e-mail: luis.tituana@knights.ucf.edu).

Abstract: In this study, finite horizon constrained trajectory optimization is tackled by using Artificial Neural Network with an embedded subspace manifold. The resulting network takes advantage of the reduced dimension search space guided by a bio-inspired motion rule. The input nodes of the network are interpreted as collocation points over the time domain transcribed by a pseudospectral discretization method. The activation function for each node is the inverse of the dynamical system. The weights and biases to be optimized in the network are analogous to the parameters of the motion rule. The network is optimized during training by minimizing an augmented loss function where the constraints are considered penalties. The proposed method is simulated in a collision avoidance trajectory planning problem of a mobile robot with two driving wheels and an attitude slewing maneuver problem of an asymmetric rigid body spacecraft.

Keywords: Trajectory Optimization, Neural Networks, Optimal Control.

1. INTRODUCTION

Nonlinear constrained optimal trajectory design problems have been studied for decades, and they have been extensively applied in autonomous vehicles or robotics, manufacturing, precision agriculture, space exploration, biomechanics, etc. (Xin and Pan, 2009; Li et al., 2015; Gros and Schild, 2017).

An abundance of optimization theories and algorithms have been suggested with effective implementations to tackle various practical problems. Those methods include, but are not limited to, indirect approaches (Bryson, 1975; Lewis et al., 2012), direct approaches (Fahroo et al., 2002; Benson et al., 2006), adaptive dynamic programming (Bellman, 1961; Bertsekas, 2020; Ng et al., 2006), subspace approaches (Kurenkov et al., 2019), and learning based algorithms (Sutton and Barto, 1998). To date, real-time implementation of many existing algorithms is still challenging for problems with high nonlinearities and needing to consider stringent and/or pop-up constraints due to issues related to the curse of dimensionality (Powel, 2007). Thus, there is an uptrend of emerging approximation solutions (Busoniu et al., 2010), finding optimal solutions in carefully selected subspaces (Belkin and Niyogi, 2003; Liu et al., 2021), or along certain solution manifolds (Xu and Li, 2014).

As one of the motion phenomena observed in insects, the virtual motion camouflage (VMC) rule (Srinivasan and Davey, 2005; Xu and Li, 2014) has been used to iteratively select a subspace, in which an optimal solution is searched for in nonlinear constrained optimal control problems. Two variances of the algorithms have been investigated, one in the category of direct collocation (Xu and Basset, 2012; Xu and Li, 2014), while the other in the category of adaptive dynamic programming (Qiang and Xu. 2022). Both have shown satisfactory performance in terms of computational speed and optimality.

Meanwhile, with the development of methods and tools that facilitate the implementation of different kinds of artificial neural networks (ANNs), optimal control problems have been addressed from the machine learning perspective. After all, the training of an ANN is itself an optimization problem. Several efforts have used ANNs to create optimal control policies by approximating the objective function, system dynamics, or both (Janner et al., 2019). Good results have been obtained by using ANNs in approximate dynamic programming, which has resulted in the development of reinforcement learning algorithms (Schulman et al. 2017; Fujimoto et al., 2018), just as one example.

There are similarities between ANN approaches and bio-inspired subspace-based methods. The structure of an artificial neuron (perceptron unit) (Omondi and Rajapackse, 2006), the fundamental part of an ANN, is shown in Fig. 1(a). During a forward pass (Omondi and Rajapackse, 2006), the output y is determined by an input vector of features x, a dense matrix of weights W, a vector of biases b, and an activation function f as

$$y(W,b) = f(b + Wx). \tag{1}$$

The weights and biases are updated during a training process by minimizing a loss function $\mathcal{L}(W,b)$. Similarly, the structure of a VMC unit reorganized from (Xu and Basset, 2012) is shown in Fig. 1(b). This model shares some similarities with the structure of the perceptron if we consider $W = diag\{v\}$, $x = x_p - x_r$, and $b = x_r\mathbf{1}$, in which $\mathbf{1}$ is a vector with all elements being 1. During a forward pass, the output y, seen as the control u, is

$$u(v,x_r) = z(x_a(v,x_r)) = z\left(x_r + v(x_p - x_r)\right), \tag{2}$$

where z is a function to compute control commands, which involves the derivative calculation of the trajectory of x_a (actual or aggressor trajectory) and dynamic inversion. Updating v (PCP) and possibly x_r (reference point), during the training process, is similar to solving the trajectory optimization problem by minimizing the loss function $\mathcal{L}(u(v, x_r))$ as in (Xu and Basset, 2012). One obvious difference between the models lies in that their weighting matrices are dense and diagonal, respectively, whereas the channels between the output of summation operations and the activation are single and fully connected, respectively, for the regular perceptron model (Fig. 1a) and the VMC network unit (Fig. 1b).

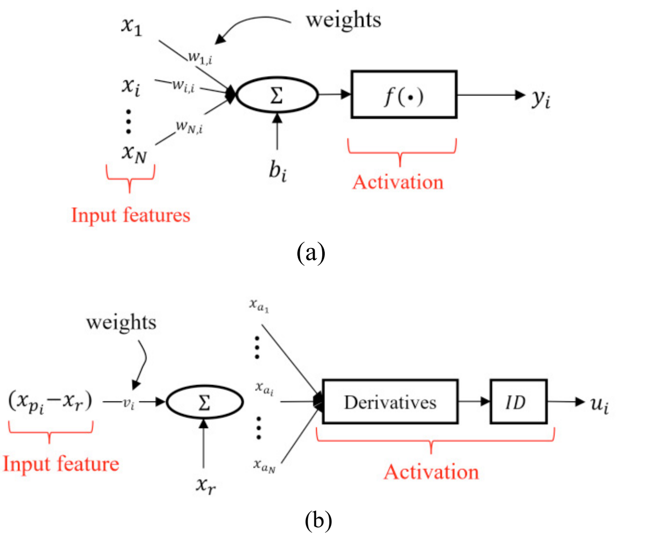

Figure 1. Similarities between, (a) a regular perceptron unit and, (b) the VMC unit.

In this study, we take advantage of the potential of neural networks as universal approximators (Abramowitz and Stegan, 1970), and use this concept to approximate the solution of a discretized finite-horizon nonlinear constrained optimal trajectory problem. We construct an ANN perceptron model using the VMC rule at its core, limiting the optimization process to happen in an iteratively selected subspace. The input to the VMC unit is the initial guess of the problem's solution and its output is the control command of the system. The shallow network has only one hidden layer consisting of three sequential stages. The first stage calculates the candidate trajectory x_a, the second stage approximates the derivative of this trajectory, and the third stage outputs the system's control commands by means of an inverse dynamics (ID) function. Each node of the network represents a collocation point from a discretized trajectory using the Legendre-Gauss-Lobatto (LGL) pseudospectral method (Fahroo and Ross, 2002).

The implementation of the VMC equation as the perceptron unit in NN reduces drastically the quantity of parameters requiring optimization, as well as the time complexity when compared to standard perceptron models. At the same time, formulating the direct collocation type VMC problems in a NN structure can utilize the backpropagation mechanism in NN aiming to increase the convergence speed, as well as different optimization algorithms therein.

The structure of the rest of the paper is as follows. Section 2 presents some preliminary concepts used in the study including optimal control problem formulation, numerical differentiation, and the VMC rule. In Section 3 we show the structure of our VMC-based neural network (VMC-NN), and explain its training and constraints inclusion. Section 4 shows the simulation results of two dynamical systems and shows their performance. Finally, in Section 5 we draw some conclusions based on simulation results and propose extensions of the concepts presented here.

2. PRELIMINARIES

2.1 Nonlinear constrained optimal trajectory design problems

In a classic finite horizon optimal trajectory control problem (Bryson, 1975), the goal is to find appropriate control commands $u(t)$ that minimize the loss function

$$J = \Phi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t)\, dt, \tag{3}$$

subject to the inequality constraints

$$g(x(t), u(t), t) \leq 0, \tag{4}$$

the equality constraints

$$h(x(t), u(t), t) = 0, \tag{5}$$

and the system's dynamical equation

$$\dot{x}(t) = f(x) + Bu(t). \tag{6}$$

We assume that the control command can be explicitly computed by inverting the dynamics in Eq. (6). For example, this assumption can be satisfied if matrix B is square and invertible or the system can be written in a control canonical form.

2.2 Discretization method

The Legendre-Gauss-Lobatto (LGL) based pseudospectral method (Fahroo and Ross, 2002) is used to discretize the trajectory, approximate the time derivatives, and integral of the loss function. It is worth mentioning that other discretization methods can also be used. The discretized functions are polynomial approximations at the LGL points. The original time domain, $t \in [t_0, t_f]$, is transformed to lie in the interval $\tau \in [\tau_0, \tau_N] = [-1,1]$, with $\tau_0 = -1$, and $\tau_N = 1$. The remaining LGL points are situated at the solutions of $\dot{L}_N(t) = 0$, that is when the time derivative of the Legendre polynomial of degree N is equal to zero (Fahroo and Ross, 2002). In the original time scale t, the nth-order derivative is calculated by

$$\frac{d^n \boldsymbol{x}}{dt^n} = \frac{2}{(t_f - t_0)^n} D^n \boldsymbol{x}, \tag{7}$$

where D is a $(N+1) \times (N+1)$ differentiation matrix (Fahroo and Ross, 2002). The calculation of the values of the matrix can be found in Fahroo and Ross, (2002). The vector $\boldsymbol{x}$ consists of $(N+1)$ discretized nodes of the state component x. The integral part of Eq. (3) is approximated with

$$\int_{t_0}^{t_f} f(x(t))dt = \frac{t_f - t_0}{2} \sum_{k=0}^{N} f(x(\tau_k))\, w_k, \tag{8}$$

where w_k are weights given by $w_k = \frac{2}{N(N+1)}\frac{1}{(L_N(\tau_k))^2}$ and $L_N(\tau_k)$ is the Legendre polynomial of degree N evaluated at the LGL point τ_k (Fahroo and Ross, 2002).

2.3 Virtual motion camouflage equation

In the observed motion camouflage phenomenon, an aggressor conceals its motion by appearing to be in the same direction with respect to a prey and a reference point (Srinivasan and Davey, 1995; Xu and Basset, 2012) as

$$x_a(t) = x_r + v(t)\big(x_p(t) - x_r\big), \tag{9}$$

where $x_a(t)$ is the aggressor's trajectory, x_r is the reference point, $x_p(t)$ is the prey's trajectory, and $v(t)$ is the path control parameter (PCP) that modifies the curvature and velocity of the prey's trajectory (Xu and Basset, 2012).

2.4 Dynamical constraints in the network

Similar to augmented Lagrangian or penalty methods (Bertsekas, 1996), we redefine the initial optimization problem and convert it into an unconstrained one by adding the constraint functions to the loss function and setting their relative importance by using weights. Equality constraints, defined for the initial and terminal states, are eliminated by construction of the optimization problem in the VMC subspace. Inequality constraints are naturally discontinuous at their boundary. Thus, they are approximated by a continuous function and treated as soft constraints, i.e., violating them does not incur an infinite penalty. The continuous approximation is done using the sigmoid function (Smith et al., 2017) as

$$c(x,u,t) \approx \frac{1}{1+e^{-\gamma g(x,u,t)}}, \tag{10}$$

where γ controls the steepness of the function and is chosen to be large enough to improve the approximation. Thus, the augmented performance index is

$$J = \int_{t_0}^{t_f} [L(x,u,t) + \sum_j \beta_j c_j(x,u,t)]dt, \tag{11}$$

where β_j is a positive penalty value for the jth inequality constraint.

3. VMC SUBSPACE NN ARCHITECTURE

Figure 2 shows the information flow in a forward pass of the network as well as the backpropagation operation. The VMC-NN is a shallow network with the same number of nodes (artificial neurons) as the number of discretization points. In a forward pass, the input (initially guessed trajectory) passes through the hidden layer and is mapped into the actual trajectory by the transformation of the VMC rule with the weights of the network (PCPs) and the reference point x_r. This trajectory goes through a nonlinear activation that consists of a numerical differentiation operation and inverse dynamics to produce the control commands. The loss of the network is then calculated by evaluating the control effort and the penalties related to the constraints of the system. The network is optimized, and thus the original optimization problem is solved during the backpropagation process.

It is worth mentioning that the dynamics of the system are not propagated (forward propagation) during training since it is assumed that the inverse dynamics will produce the necessary control to drive the system to match the actual trajectory and satisfy the boundary constraints. This means that the dynamics, with the predicted control from the network, are not propagated to verify if the constraints have been met. Simulation results in Section 4 show the effect of this mismatch due to numerical errors in discretization in terms of a root mean square average error (RMSE) (Zhang and Scaramuzza, 2018) between the predicted and propagated trajectories. Nevertheless, dynamics propagation can be included during the forward pass to increase accuracy at the expense of increasing the training time. This could allow the use of low-order discretization methods to compensate for the added time.

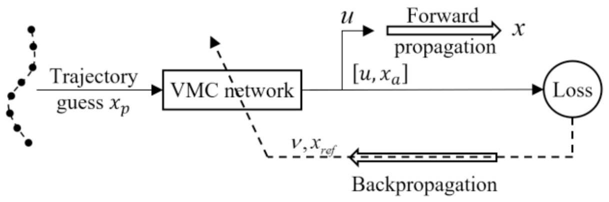

Figure 2. Data flow in the VMC-NN.

Figure 3 shows the structure of the studied VMC-NN layers. All the points of the trajectory are used, making it a fully connected layer with high accuracy when calculating the derivatives. Note that the network can be customized to fit other discretization methods and the order of accuracy can be regarded as a new hyperparameter.

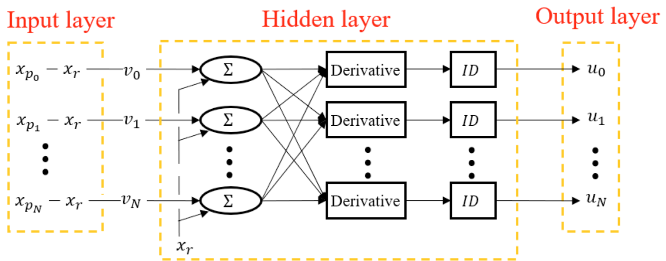

Figure 3. VMC-NN structure with the LGL discretization method.

4. SIMULATION RESULTS

The proposed VMC-NN structure is compared against a regular fully connected (FC) neural network, with one hidden layer, trying to solve the same optimization problem in the first simulation example. The results are used to show the advantages of the proposed subspace NN-based optimization structure in terms of the loss function, time to convergence, error between the optimized trajectory and the propagated dynamics, and number of optimizable parameters. Additionally, in the first simulation example, we compare the results when optimizing VMC-NN using two readily available optimizers: LBFGS and Adam (Kingma and Ba, 2014). After training, the dynamics are propagated using a fourth-order

Runge-Kutta method (RK4) to verify that constraints have been met using the predicted control.

The VMC-NN is implemented using Pytorch 2.0 with Python 3.8 in a laptop with an Intel i7-11800H @ 2.0 GHz processor and 16 GB of RAM. The reference point x_r was considered an optimizable parameter. Changing the reference point changes the search space, but modifying the PCP explores it. This is the reason why x_r was optimized later after the first 150 iterations (epochs) with a different learning rate. The training stops after the moving average of the loss function, with a window of 30, has reached a change rate of less than 5×10^{-5} (convergence). The number of collocation points is 21 (i.e., *N*=20). To measure how different the propagated trajectory is from the optimized trajectory, we use RMSE.

The input layer of the FC-NN contains the same points as that of VMC-NN. The number of hidden nodes is also 21, which is the same size the VMC-NN is composed of, and their activation is the hyperbolic tangent function. The derivative and inverse dynamics operations are in the output layer. To meet the initial and terminal conditions, like the VMC-NN, the first and last nodes in the output layer are fixed to match such points. The network weights and biases are updated using the Adam optimizer. The learning rate is set to 0.001 for the first 4000 iterations, following a decrease to 0.0001 for the remaining of the training. The same stop condition is considered only after the FC-NN has surpassed the average performance of the VMC-NN.

4.1 Two-wheel robot

The dynamic model of the system is given in Laumond et al., (1998), and the objective is to drive the robot in a planar motion from the initial point $x(0) = [1, 1]\ m$ to the terminal state $x(10) = [6, 6]\ m$. The initially guessed motion, for each of the *i*th components, is assumed to follow a convex quadratic function. The initial guess for the reference point is assumed to be $x_r = [5.4,\ 2]\ m$, and the maximum values of control are set to be $|u_{1_{max}}| \leq 1$ and $|u_{2_{max}}| \leq 135°$. The obstacles are circles with radii of 1 m and 0.6 m whose centers are located at $[3.5,\ 3.5]\ m$ and $[5.3,\ 4.4]\ m$, respectively. The penalty values for those four inequality constraints are $\beta_1 = \beta_2 = 30$ for touching the obstacles, and $\beta_3 = \beta_4 = 3.5$ for surpassing the maximum control. For this problem, the learning rate was 0.01 for the PCPs and 0.4 for x_r, respectively.

Figure 4 illustrates how the VMC-NN solves the optimization problem. The network produces a control where the boundary and obstacle constraints are all met. However, the propagated dynamics do not match exactly with the actual propagated trajectory due to the assumption explained in Section 2.4. Table 1 shows a comparison between four metrics to assess the performance of the VMC-NN and the FC-NN. Both networks produced a similar trajectory that avoids the obstacles and meets the terminal state constraint within an error tolerance of 5×10^{-3}. Due to the use of the motion rule, the VMC-NN contains only 21 optimizable parameters and manages to converge to an optimal solution in 4.03 s, while the FC-NN, with 1722 parameters, requires 8.56 s to solve the problem. However, the reduced time comes at the expense of optimality. The VMC-NN reaches a performance index of 1.0327 with an RMSE of 0.00395, which are only a little bit higher than their FC-NN counterparts of 1.0138 and 0.00124, respectively. Nevertheless, the propagated dynamics closely follow the optimized trajectory using the generated control. The better performance of the FC-NN is expected as its search space is encompassed by the whole domain of the real numbers, which also explains the almost two times longer time it takes to train.

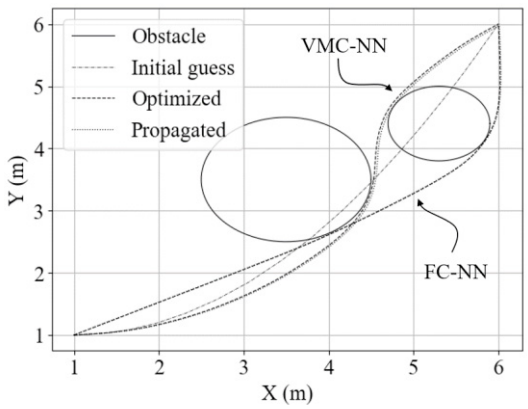

Figure 4. Trajectories in the *x-y* plane of the optimized, initially guessed, and propagated dynamics.

Table 1. Performance comparison for the two-wheel robot simulation example.

	FC-NN	VMC-NN
Performance index	1.0138	1.0327
Convergence time (s)	8.56	4.03
RMSE	0.00124	0.00395
# of optimizable parameters	1722	21

The comparison of training results using the LBFGS and Adam optimizers for the VMC-NN is shown in Table 2. The LBFGS optimizer, with a learning rate of 0.5 and history size of 5, can reach a performance index of 1.0114, which is slightly better than the value of 1.0327 achieved using the training strategy mentioned at the beginning of this section. However, LBFGS is a very memory expensive algorithm and requires longer to converge, 6.67 s versus 4.03 s when using Adam. These results show how our network can be implemented with already existing training algorithms, and how it can be easily adapted to a custom update rule that considers the reduced dimension nature of its structure.

Table 2. Comparison of the VMC-NN optimization using LBFGS and Adam as optimizers for the two-wheel robot example.

Optimizer	Performance index	Convergence time (s)
LBFGS	1.0114	6.67
Adam	1.0327	4.03

4.2 Asymmetric rigid spacecraft

In this example, the objective is to stop the rotation motion of an asymmetric rigid spacecraft, whose dynamics and constraints are given in Jaddu, (2002). The initial and final states are $x(0) = [0.01, 0.005, 0.001]\ rad/s$ and $x(100) = [0, 0, 0]\ rad/s$. The initially guessed trajectory for the angular velocity follows a concave quadratic function, whose vertex is at the initial point and passes through the final point. The initial guess for the reference point is $x_r = [0.9,\ 3.1, 0.07]\ rad/s$, and all control torques are bounded by $|u_{max}| \leq 0.01\ Nm$. An inequality constraint in (Jaddu, 2002) is adopted here for the first angular velocity as

$$\omega_1 \leq 5 \times 10^{-6} t^2 - 5 \times 10^{-4} t + 0.016. \qquad (12)$$

The penalty values for the four constraints are $\beta_1 = 6$ for violating the constraint in (12), and $\beta_2 = \beta_3 = \beta_4 = 1.5$ for exceeding the maximum control. For this problem, we used a learning rate of 0.001 for the PCPs and 0.1 for x_r. In Fig. 5, the trajectories of the angular velocities are shown. Like the previous example, our network successfully finds a solution that meets the constraints, and the terminal condition is met with an RMSE of 0.0156. The performance index converges to an average value of 0.033 in about 0.86 s. The propagated dynamics match well with the optimized one with an overall RMSE of 0.0073, and are not visibly different in Fig. 5.

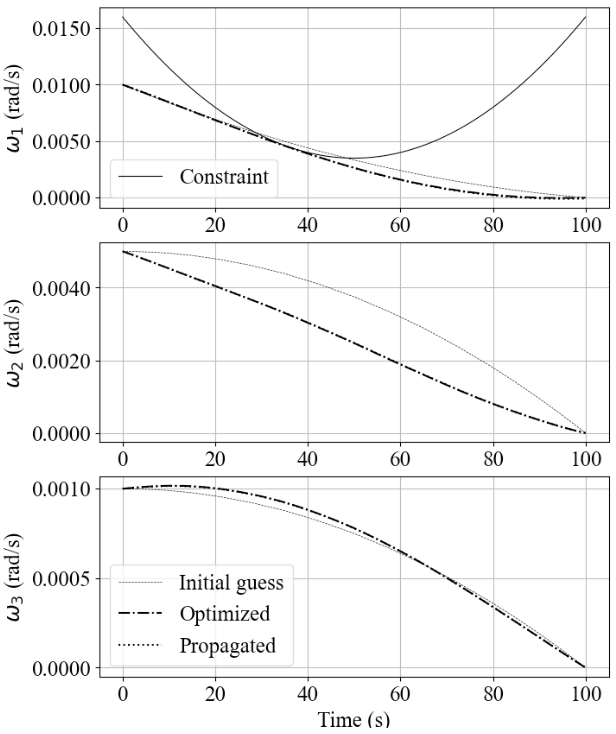

Figure 5. Trajectories of the angular velocities versus time.

It is worth noting that the solutions found are suboptimal as the search space is restricted by the motion rule. This is explained since changing the value of the PCPs modifies, at the same time, the trajectories of all the states in the problems. Nevertheless, this limitation is compensated for by the reduced number of optimizable parameters needed to train. Moreover, considering the reference point x_r an optimizable parameter played a big role in finding a solution to the optimization problems. In our experience, when the network got stuck in a local minimum, changing the reference created a different path for which a solution could be found. Thus, updating the reference point with a higher learning rate than that of PCPs will allow the network to explore more areas of the domain.

On a separate note, since the initially guessed trajectory is a virtual one, it is not restricted to follow any physical or dynamical law. Thus, manually selected points could be used to create a trajectory that already meets the constraints. This characteristic can also be exploited to expand the domain of the solution.

Future direction 1: A second network can be developed to address the numerical mismatch introduced by the discretization method and other noise or uncertainties. This network could act as a "critic" and learn to compensate for the discrepancy in the calculation of the derivatives.

Future direction 2: The proposed VMC-NN can be integrated into a receding horizon framework to enable its real-time usage.

Future direction 3: The stability of the closed-loop system will be analyzed for the feedforward network, and the robustness will be analyzed after the critic network is added to handle noise and uncertainties.

6. CONCLUSIONS AND FUTURE DIRECTIONS

In this work, we presented the VMC-NN, an artificial neural network whose perceptron structure is modified using a bio-inspired motion rule. The network benefits from the reduced dimension of the varying subspace to find an optimal control by optimizing a vector of weights instead of a dense matrix as in regular ANNs. The network's forward pass employs a numerical differentiation method followed by an inverse dynamics operation. The network is shown to find an optimal solution while meeting a set of constraints. Future work will be focused on increasing the accuracy of the prediction by propagating the system dynamics in the forward pass and using low-order numerical differentiation methods to speed up training. This network can be regarded as a policy generation function, similar to an actor-critic based agent in the context of reinforcement learning.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation under Grants No.1924622.

REFERENCES

Abramowitz, M. and Stegan, I. A. (1970). Handbook of mathematical functions with formulas, graphs, and mathematical tables. *NY: Dover Publications*.

Belkin, M. and Niyogi, P. (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation, *Neural Computation*, 15(6), 1373-1396.

Bellman, R. (1961). Adaptive Control Processes: A Guided Tour. Princeton, *Princeton University Press*.

Benson, D. A., Huntington, G. T., Thorvaldsen, T. P. and Rao, A. V. (2006). Direct trajectory optimization and costate estimation via an orthogonal collocation method, *Journal of Guidance, Control, and Dynamics*, 26(6), 1435-1440.

Bertsekas, D.P. (1996). Constrained optimization and Lagrange multiplier methods, *Athena Scientific*.

Bertsekas, D. P. (2020). Rollout, Policy Iteration, and Distributed Reinforcement Learning, *Athena Scientific.*

Bryson, A. E. (1975). Applied optimal control: optimization, estimation and control, *CRC Press*.

Busoniu, L., Babuska, R., De Schutter, B. and & Ernst, D. (2010). Reinforcement learning and dynamic programming using function approximators, *CRC Press*.

Butcher, J. C. (2003). Numerical Methods for Ordinary Differential Equations, *New York: John Wiley & Sons*.

Fahroo, F. and Ross, I. M. (2001). Costate Estimation by a Legendre Pseudospectral Method. *Journal of Guidance, Control and Dynamics*. 24(2), 370-275.

Fahroo, F. and Ross, I. M. (2002). Direct trajectory optimization by a Chebyshev pseudospectral method, *Journal of Guidance, Control, and Dynamics*, 25(1), 160-166.

Fujimoto, S., van Hoof, H. and Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods, *Proceedings of the 35th International Conference on Mahcine Learning (ICML)*, 80, 1582-1591.

Garg, D. (2011). Direct trajectory optimization and costate estimation of finite-horizon and infinite-horizon optimal control problems using a Radau pseudospectral method, *Computational Optimization, and Applications*, 49(2), 335-358.

Gros, S. and Schild, A. (2017). Real-time economic nonlinear model predictive control for wind turbine control. *International Journal of Control*, 90(12), 2799-2812.

Hornik, K. (1989). Multilayer Feedforward Networks are Universal Approximators. *Neural Neworks*, 2, 359-366.

Janner, M., Fu, J., Zhang, M. and Levine, S. (2019). When to Trust your model: Model-based policy optimization, *Neural Information Processing Systems (NeurIPS)*.

Jaddu, H. (2002) Direct solution of nonlinear optimal control problems using quasilinearization and Chebyshev polynomials, *Journal of the Franklin Institute*, 339, 479-498.

Junkins, K. L. and Turner, J. D. (1986). *Optimal Spacecraft Rotational Maneuvers*, Elsevier, Amsterdam.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization, *arXiv:1412.6980*.

Kurenkov, A. et al. (2019). AC-Teach: A Bayesian Actor-Critic Method for Policy Learning with an Ensemble of Suboptimal Teachers, *3rd Conference on Robot Learning (CoRL)*.

Laumond, J. P., Sekhavat, S. and Lamiraux, F. (1998). Guidelines in nonholonomic motion planning for mobile robots. Robot Motion Planning and Control. *Lecture Notes in Control and Information Sciences, vol 229*, Springer, Berlin, Heidelberg.

Lewis, F. L., Vrabie, D. and Syrmos V. L. (2012). Optimal control, *John Wiley & Sons*.

Li, N., Remeikas, C., Xu, Y., Jayasuriya, S. and Ehsani, R. (2015). Task Assignment and trajectory planning algorithm for a class of cooperative agricultural robots. *Journal of Dynamic Systems, Measurements and Control*, 137(5), 051004.

Li, Q. and Xu, Y. (2022). Dimension reduction based adaptive dynamic programming for optimal control of discrete-time nonlinear control-affine systems, *International Journal of Control*, 1-13.

Liu, X., Wen, Z. and Ya-Xiang, Y. (2021). Subspace Methods for Nonlinear Optimization, *CSIAM Transactions on Applied Mathematics*, 2(4), 585-651

Ng, A. G. et al. (2006). Autonomous inverted helicopter flight via reinforcement learning, *Experimental Robotics IX*, 363–372.

Omondi, A. and Rajapackse, J. (2006). *FPGA Implementations of Neural Networks*, Springer.

Powell, W. (2007). Approximate Dynamic Programming: Solving the Curses of Dimensionality. *John Wiley & Sons*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017). Proximal Policy Optimization Algorithms, *arXiv:1707.06347*.

Smith, N.E., Arendt, C. D., Cobb, R.G. and Reeger, J. A. (2017). Implementing conditional inequality constraints for optimal collision avoidance, *Journal of Aeronautics & Aerospace Engineering*, 6(195).

Srinivasan, M. and Davey, M. (1995). Strategies for active camouflage of motion, *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 259(1354), 19–25.

Sutton, R. and Barto, A. (2018) *Reinforcement Learning: An Introduction* (2nd ed.), The MIT Press.

Xim, M., Pan, H. (2009). Integrated nonlinear optimal control of spacecraft in proximity operations. *International Journal of Control*. 83(2), 347-363.

Xu, Y. and Basset, G. (2012). Sequential virtual motion camouflage method for optimal nonlinear constrained optimal trajectory control. *Automatica*. 48(7), 1273-1285.

Xu, Y. and Li, N. (2014). Bio-inspired varying subspace based computational framework for a class of nonlinear constrained optimal trajectory planning problems. *Bioinspiration & Biomimetics*. 9, 036010.

Zhang, Z. and Scaramuzza, D. (1018). A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry, *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 7244-7251.