Direct Methods Comparison for the Active Target Defense Scenario

Isaac E. Weintraub[*]
Air Force Research Laboratory, Wright-Patterson AFB, OH 45433

Richard Cobb [†], William Baker[‡], and Meir Pachter[§]
Air Force Institute of Technology, Wright-Patterson AFB, OH 45433

The goal of optimal control is to obtain an admissible function which minimizes an objective functional subject to specified constraints and boundary conditions. Four popular methods of computing the optimal control directly are the Single Shooting Method, Multiple Shootings Method, Even Collocation Method, and Pseudospectral Method. In this paper, the four direct methods of computing the optimal control for the Active Target Defense Scenario are presented and compared. The Active Target Defense Scenario is a three-agent engagement where an evading agent tries to escape a pursuing agent with the assistance of a defending agent. The goal of the Evader is to maneuver in such a way as to assist the Defender in capturing the Pursuer while maximizing his range from the Pursuer. This work begins by describing each direct method and summarizing advantages and disadvantages of each. Next, each method is used to solve the optimal control for the Active Target Defense Scenario; results are presented and a comparison of the performance of each method is made. Finally, concluding remarks describing the results, performance, and real-time implementation are presented.

I. Nomenclature

$\mathbf{D}$	=	differentiation matrix
$\mathbf{f}(\cdot)$	=	dynamics
$g(\cdot)$	=	path constraints
$h(\cdot)$	=	inequality constraints
J	=	objective cost functional
k	=	discrete time index
$l(\cdot)$	=	discrete time objective cost weight
L_i	=	Lagrange interpolating polynomial
$L(\cdot)$	=	Lagrangian
$\dot{\mathbf{x}}$	=	derivative of the state with respect to time
n	=	state space dimension
N	=	number of points
N_p	=	proportional navigation gain for the pursuer
N_d	=	proportional navigation gain for the defender
M	=	number of points
R	=	number of iterations
R_{dp}	=	range from defender to pursuer
R_{pe}	=	range from pursuer to evader
t	=	time
u	=	control
V	=	velocity
w	=	quadrature weight

[*]Electronics Engineer, Aerospace Systems Directorate, Address/Mail Stop, AIAA Senior Member.
[†]Professor, Department of Aerospace Engineering, Address/Mail Stop, and AIAA Member Grade (if any) for second author.
[‡]Professor, Department of Mathematic, Address/Mail Stop, and AIAA Member Grade (if any) for third author.
[§]Professor, Department of Electrical Engineering, Address/Mail Stop, and AIAA Member Grade (if any) for fourth author (etc.).

x	=	x-position
$\mathbf{x}$	=	state
$\mathbf{x}$	=	state vector
$\hat{x}$	=	state estimate
y	=	y-position
Δt	=	change in time
γ	=	heading angle
λ	=	line of sight angle
$\phi(\cdot)$	=	terminal objective cost
ϕ	=	angle between the velocity of an agent and its light of sight to its evader
σ	=	angle between an agent's velocity and his pursuer's line of sight
τ	=	non-dimensionalized time
ω	=	angular rate

Subscripts

0	=	initial time
f	=	final time
j	=	index j
k	=	index k
p	=	Pursuer
d	=	Defender
e	=	Evader

II. Introduction

The objective of optimal control theory is to determine the control signals that will cause a process to satisfy the physical constraints and at the same time minimize (or maximize) some performance criterion [[1]]. The goal of indirect methods is to find the optimal control analytically, leveraging the Calculus of Variations. By satisfying the necessary conditions for optimal control, the indirect method finds a closed form solution to the optimal control. One disadvantage of the indirect method is that finding the optimal control, analytically, is not always tractable. As and as a result, direct methods of finding the optimal control are made possible with the use of computational methods.

The introduction, to be written will introduce the topic of direct methods of finding optimal control citing the relavant work in the public domain. Paragraphs will introduce each of the four direct methods: Single Shooting Method, Multiple Shootings Method, Even Collocation Methods, and Pseudospectral Method. Motivation for the use of direct techniques to solve the Active Target Defense Scenario will be composed with citation of previous work which uses direct methods to solve the problem. This motivation will start with the problem description as well as a definition of the objective.

III. Description of Direct Methods

In this section we begin with a qualitative description of all four direct methods: Single Shooting Method, Multiple Shootings Method, Even Collocation Method, and Pseudospectral Method. Special emphasis hilghlingthing similarities and differences of each method are also provided. Comments on the size of each problem and requirements to produce initial guesses are also made.

A. Single Shooting Method Description

The Single Shooting Method (SSM), outlined in Figure 1, is a numerical technique for finding the optimal control to a dynamic system which achieves given objective subject to boundary conditions and path constraints. The procedure begins with some initial state, $\mathbf{x}_0$, and a guess for the control for the entire time series, $\{u_k | k = 1, \ldots, N\}$. Using the guess for the control and the initial state, a Nonlinear Program (NLP) solver, such as MATLAB's `FMINCON()`, computes the optimal control through iterative search. The NLP performs the search for the optimal control by first "shooting" the dynamics forward through time using an Ordinary Differential Equation (ODE) solver such as the Runge-Kutta Method [2]. Next, the NLP computes the objective cost functional using the propagated state trajectories. After the cost is computed, the NLP evaluates if a local minimum has been found. If the cost is a minimum, by some convergence criteria, the guess is considered the optimal control. However, if the objective cost functional is not considered a minimum, then an update to the guess for the control is made, and the process starts again.

There exist multiple numerical techniques to update the guess for the control, one popular technique is the method of Sequential Quadratic Programming (SQP) [3]. The SQP algorithm, in short, is a gradient-based technique for finding the optimal control which minimizes the objective cost subject to equality and inequality constraints. It strives to satisfy the first-order necessary conditions for optimality, called Karush-Kuhn-Tucker (KKT) conditions, by updating guesses until the objective cost is at a minimum and all constraints are satisfied to some tolerance. To determine how to update the guess for the control, the algorithm perturbs the control throughout its trajectory. By observing the change in the objective cost, as a function of the perturbations, updates are chosen to improve the guess for optimal control. In the event the KKT conditions are satisfied, the optimal guess for the control sequence, $\{u_k^* | k = 1, \dots, N\}$, is produced. Since the NLP produces only the optimal guess for optimal control, the dynamics must be "shot" forward one more time in order to obtain the optimal state trajectories, $\{\mathbf{x}_k^* | k = 1, \dots, N\}$.

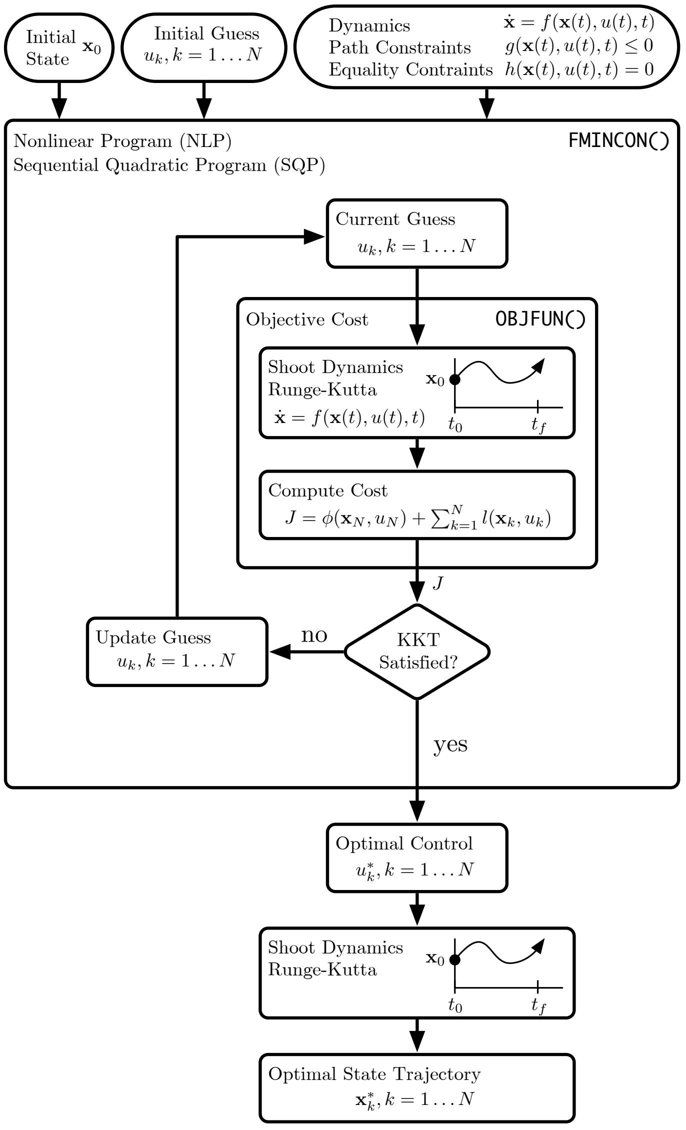
Initial State $\mathbf{x}_0$
Initial Guess $u_k, k = 1 \dots N$
Dynamics $\dot{\mathbf{x}} = f(\mathbf{x}(t), u(t), t)$
Path Constraints $g(\mathbf{x}(t), u(t), t) \leq 0$
Equality Contraints $h(\mathbf{x}(t), u(t), t) = 0$
Nonlinear Program (NLP)
Sequential Quadratic Program (SQP)
FMINCON()
Current Guess $u_k, k = 1 \dots N$
Objective Cost
OBJFUN()
Shoot Dynamics Runge-Kutta $\dot{\mathbf{x}} = f(\mathbf{x}(t), u(t), t)$
$\mathbf{x}_0$ t_0 t_f
Compute Cost $J = \phi(\mathbf{x}_N, u_N) + \sum_{k=1}^{N} l(\mathbf{x}_k, u_k)$
J
KKT Satisfied?
no
Update Guess $u_k, k = 1 \dots N$
yes
Optimal Control $u_k^*, k = 1 \dots N$
Shoot Dynamics Runge-Kutta
$\mathbf{x}_0$ t_0 t_f
Optimal State Trajectory $\mathbf{x}_k^*, k = 1 \dots N$

Fig. 1 Shooting Method Flow Diagram

The requirements to begin the shooting method is a guess for the control sequence u_k. If we assume that time is discretized evenly from t_0 to t_f into N points, then the initial guess is described as:

$$\{u_k \mid k = 1 \ldots N\} \tag{1}$$

The overall size to compute the optimal control using the shooting method is quite large. If we assume that time is evenly discretized into N points, we would have to forward propagate the dynamics every time we compute the objective cost. Further, let R be the number of iterations required to obtain an optimal control. Since perturbations are made at every time step, the number of times the dynamics are forward propagated would be approximately $R(N + 1)$. (The 1 represents the evaluation of the current guess, while the N represents the evaluation for each perturbation made of the guess.)

Since the dynamics are guaranteed by the accuracy of the ODE solver, the feasibility of solution is ideal. Although the computational time associated with using an ODE solver is large, it ensures that the dynamics are upheld. Since the initial state, $\mathbf{x}_0$, is provided and the final state is free, no equality constraints are applied.

B. Multiple Shooting Method Description

The Multiple Shootings Method (MSM) differs from single shooting in that it aims at reducing computational time by parallelizing the shooting operation. Breaking the shooting operation into smaller segments, the MSM makes use of multiple processors to shoot each segment in parallel. In exchange for parallelizing the shooting operations, discontinuities occur in-between each connecting interval. A series of equality constraints, one at each transition from one interval to the next are formed as "continuity constraints." Figure 2 describes the process of conducting a multiple shooting direct optimization.

The procedure begins with a guess for the control for the entire time series, $\{u_k | k = 1, \ldots, N\}$, and a value for the states at the beginning of each segment. Assuming there are M segments, n states, and the initial state is known to be $\mathbf{x}_0$, the size of the initial guess for the states at each segment is $n(M - 1)$. In total, the size of the guess is $N + n(M - 1)$ when the initial states are known.

Using the SQP algorithm, we iteratively search for the optimal states and control until the KKT conditions are satisfied. If the step size of the guess is below some threshold, the change in the objective cost functional is also below some threshold, and if the feasibility (defined as the equality constraints from the continuity conditions) is within some tolerance, then the NLP returns the optimal control as well as the values of the states at the beginning of each of the segments.

Although the overall size of the MSM is less than the shooting method, it still requires the use of an ODE solver such as the Runge-Kutta Method. Since the shootings are conducted in parallel, the size of the algorithm is reduced. If the number of computational threads is equal to the number of segments, the effective computational time for shooting the dynamics is reduced by a factor of M. However, the introduction of the continuity constraints requires the addition of $n(M - 1)$ unknowns to be solved in addition to the control.

C. Even Collocation Method

The Even Collocation Method (ECM) takes advantage of computational efficiency by transcribing a dynamic optimization problem, which requires numerical methods for solving differential equations, into a static optimization problem. The method begins with considering the dynamics and control at evenly spaced points in time from t_0 to t_f. If we consider N points, the guess for the states and control are:

$$\mathbf{x}_k,\ k = 1, \ldots, N \tag{2}$$

$$u_k,\ k = 1, \ldots, N \tag{3}$$

Using the guess for the state and control trajectories, the ECM imposes the dynamics as a set of equality constraints rather than using an ODE solver such as the Runge-Kutta Method. The equality constraints are formed by a simple subtraction:

$$\dot{\mathbf{x}} - f(\mathbf{x}(t), u(t), t) = 0 \quad t \in [t_0, t_f] \tag{4}$$

Assuming that there are n states, the number of equality constraints formed from imposing the dynamics are nN given by the first-order Euler approximation:

$$\mathbf{h}_k(\mathbf{x}_k, u_k, k) = \mathbf{x}_{k+1} - \mathbf{x}_k - f(\mathbf{x}_k, u_k, k)\Delta t \quad k = 1, 2, ..., N \tag{5}$$

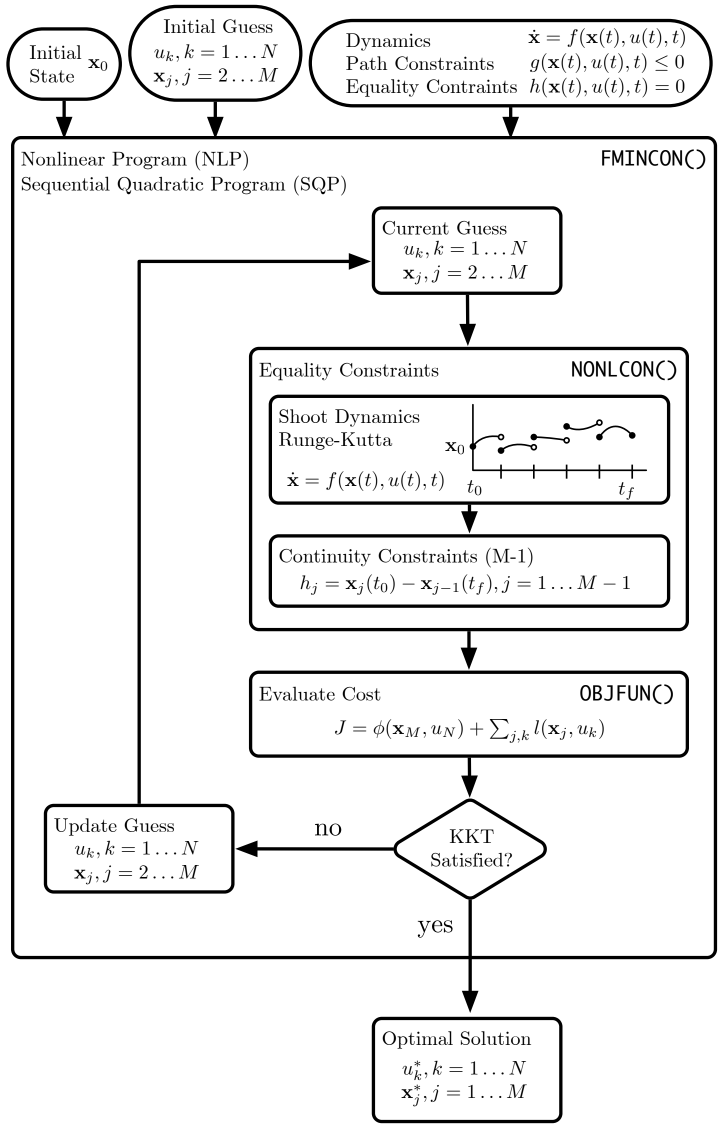
Initial State $\mathbf{x}_0$
Initial Guess $u_k, k = 1 \dots N$ $\mathbf{x}_j, j = 2 \dots M$
Dynamics $\dot{\mathbf{x}} = f(\mathbf{x}(t), u(t), t)$
Path Constraints $g(\mathbf{x}(t), u(t), t) \leq 0$
Equality Contraints $h(\mathbf{x}(t), u(t), t) = 0$
Nonlinear Program (NLP)
Sequential Quadratic Program (SQP)
FMINCON()
Current Guess $u_k, k = 1 \dots N$ $\mathbf{x}_j, j = 2 \dots M$
Equality Constraints
NONLCON()
Shoot Dynamics
Runge-Kutta
$\dot{\mathbf{x}} = f(\mathbf{x}(t), u(t), t)$
$\mathbf{x}_0$
t_0
t_f
Continuity Constraints (M-1)
$h_j = \mathbf{x}_j(t_0) - \mathbf{x}_{j-1}(t_f), j = 1 \dots M-1$
Evaluate Cost
OBJFUN()
$J = \phi(\mathbf{x}_M, u_N) + \sum_{j,k} l(\mathbf{x}_j, u_k)$
Update Guess $u_k, k = 1 \dots N$ $\mathbf{x}_j, j = 2 \dots M$
no
KKT Satisfied?
yes
Optimal Solution $u_k^*, k = 1 \dots N$ $\mathbf{x}_j^*, j = 1 \dots M$

Fig. 2 Multiple Shooting Method Flow Diagram

Using the definition of the equality constraints to satisfy the dynamics of our problem, we no longer are required to use an ODE solver to shoot the dynamics. Rather, we satisfy them as a constraint which the NLP considers feasibility criteria. The flow diagram which describes even collocation can be seen in Figure 3. In the procedure, we begin by providing a guess for the state and control trajectories. As described, the dynamics are transcribed into an equality constraint, and a `FOR()` loop is used to compute each equality constraint at every time step. Additionally, the objective cost functional is computed as a function of the states and control. Since the collocation points are evenly spaced, any terms inside the integral of the cost objective functional are approximated using a quadrature method such as trapezoidal or rectangular integration. Using the SQP search described earlier, the objective cost functional and equality constraints are minimized through iterative search until the KKT conditions are satisfied. Upon completion, the solution provides the optimal control and corresponding state trajectories at each collocation point.

In this work, we assume that there are N evenly spaced collocation points. However, a finer or coarser mesh may be implemented to increase or reduce the number of collocation points. By reducing the number of points, the number of function evaluations may be reduced. If we use the First-Euler approximation of the dynamics as an equality constraint, and the meshing becomes too coarse, then the small time steps assumption made earlier may no longer hold. Methods of changing the spacing between nodes are described as adaptive meshing techniques. The general concept of adaptive meshing is to decrease the spacing between nodes where more accurate computations of the states are required and allow the spacing of other areas to be larger to reduce the total number of points. For this comparative study, the use of adaptive meshing is not considered. Since the ODE solvers employed in the shooting method are fixed time-step solvers, fixed meshing is used for the collocated methods to make a fair comparison.

The overall size of the ECM is much smaller than SSM and MSM. By allowing the dynamics to be accurate to a prescribed threshold, the use of ODE solvers is eliminated. In our example, the derivative is computed using a first-order Euler's Method which isn't the most accurate, but for small time steps is sufficient. Furthermore, using a low-order quadrature method such as rectangular integration, the cost functional is computed. Since there are no uses of ODE solvers, the size of the ECM is to find the optimal solution to the state and control trajectories, a size of $N(n+1)$. If we assume there are R iterations, the number of function evaluations will be around the order of $RN(n+1)$.

D. Pseudospectral Methods

The Pseudospectral Method (PSM) refers to a series of techniques which are employed to solve optimal control problems by taking advantage of efficient computational methods. The PSM enforces dynamics at a set of collocation points (just as before) by means of equality constraints and the objective cost functional is computed through Gaussian quadrature. The states are approximated using Lagrange interpolating polynomials:

$$\hat{x}(t) \approx \sum_{i=1}^{n+1} x_i L_i(t) \tag{6}$$

where the Lagrange polynomial basis is efficiently computed as:

$$L_i(t) = \prod_{\substack{j=1 \\ j\neq i}}^{n+1} \frac{t-t_j}{t_i-t_j} \quad i = 1, \ldots, n+1 \tag{7}$$

By using Lagrange interpolating polynomials, there is zero error of the interpolating polynomial at each collocation point, and since the interpolated function is a polynomial, any number of derivatives are guaranteed to exist. In-between the collocation points, the error can be found by:

$$x(t) - \hat{x}(t) = \frac{x^{(n+1)}\xi(t)}{(n+1)!} \prod_{i=0}^{n} (t-t_i) \tag{8}$$

The point ξ is the value of t where the $(n+1)^{\text{st}}$ derivative of the sate x is equal to zero. Upon the first investigation, the error can be reduced by adding more points, but the Runge Phenomenon, where high-order derivatives cause the error to increase near the endpoints can be problematic. In order to overcome this phenomenon, we space the collocation points defined by the roots of an N^{th}-order Legendre polynomial on the interval [-1,1]. This not only reduces the Runge Phenomenon but also leads to exponential convergence in quadrature. Using an affine transformation, the time series must be mapped to the [-1,1] domain where the collocation points are determined by the roots of the Legendre Polynomial. Using, more specifically, the +1 point, we call the basis of collocation points Legendre-Gauss-Radau (LGR) points. The mapping to the affine transformation is as follows:

$$\tau = \frac{2t - (t_f - t_0)}{t_f - t_0} \tag{9}$$

Derivatives may also be computed by means of a matrix multiplication which is more efficient than first-order Euler approximations, and faster since operations can be computed by means of matrix multiplication rather than for-loops. Using standard techniques, a differentiation matrix (10) may be computed [4]. Using the differentiation matrix, equality constraints (11) are used to ensure the dynamics are upheld.

$$\dot{\mathbf{x}} \approx \mathbf{D}\mathbf{x} \tag{10}$$

$$\mathbf{h}(\mathbf{x}, u) = \mathbf{D}\mathbf{x} - \frac{\Delta t}{2}\mathbf{f}(\mathbf{x}, u) = \mathbf{0} \tag{11}$$

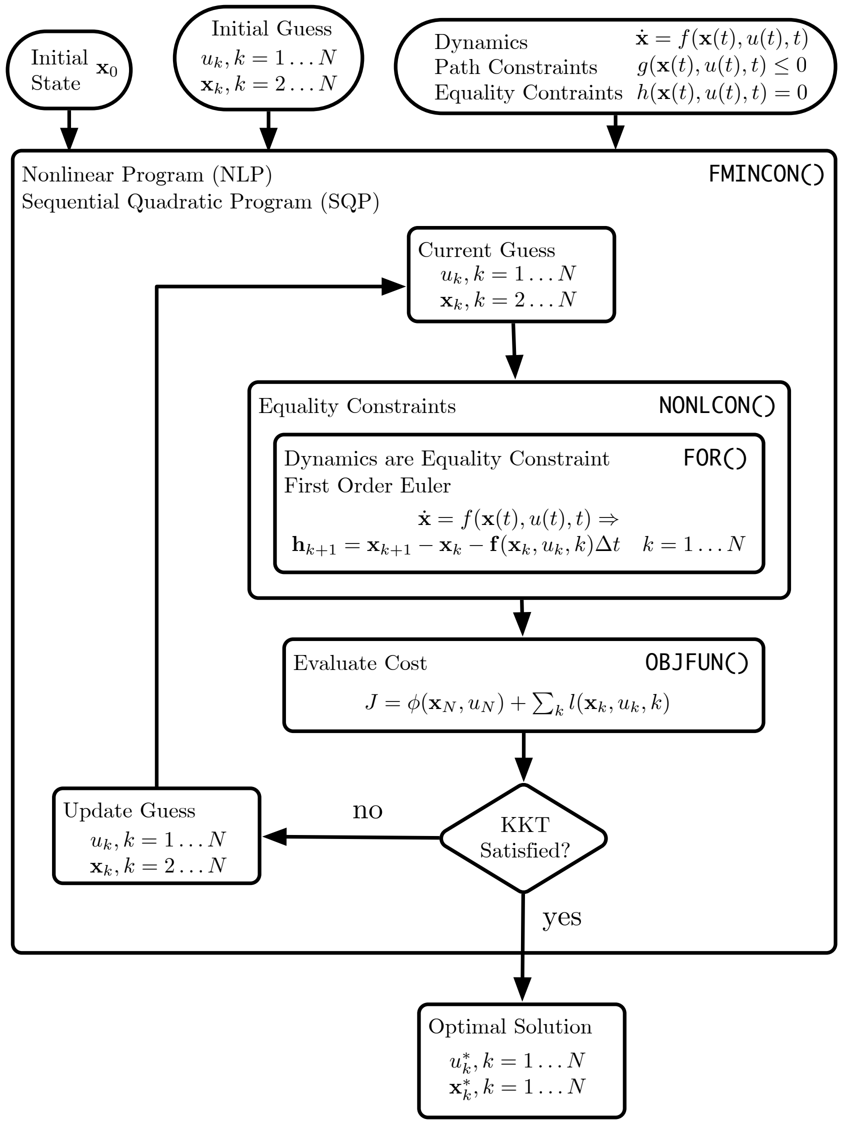
Initial State $\mathbf{x}_0$
Initial Guess $u_k, k = 1 \dots N$ $\mathbf{x}_k, k = 2 \dots N$
Dynamics $\dot{\mathbf{x}} = f(\mathbf{x}(t), u(t), t)$
Path Constraints $g(\mathbf{x}(t), u(t), t) \leq 0$
Equality Contraints $h(\mathbf{x}(t), u(t), t) = 0$
Nonlinear Program (NLP)
Sequential Quadratic Program (SQP)
FMINCON()
Current Guess $u_k, k = 1 \dots N$ $\mathbf{x}_k, k = 2 \dots N$
Equality Constraints
NONLCON()
Dynamics are Equality Constraint
FOR()
First Order Euler
$\dot{\mathbf{x}} = f(\mathbf{x}(t), u(t), t) \Rightarrow$
$\mathbf{h}_{k+1} = \mathbf{x}_{k+1} - \mathbf{x}_k - \mathbf{f}(\mathbf{x}_k, u_k, k)\Delta t \quad k = 1 \dots N$
Evaluate Cost
OBJFUN()
$J = \phi(\mathbf{x}_N, u_N) + \sum_k l(\mathbf{x}_k, u_k, k)$
KKT Satisfied?
no
yes
Update Guess $u_k, k = 1 \dots N$ $\mathbf{x}_k, k = 2 \dots N$
Optimal Solution $u_k^*, k = 1 \dots N$ $\mathbf{x}_k^*, k = 1 \dots N$

Fig. 3 Even Collocation Direct Method Flow Diagram

Quadrature weights, w_k, are associated with the collocation scheme and approximate the integration of the running cost, as:

$$\int_{t_0}^{t_f} L(\mathbf{x}, u, t)dt \approx \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k L(\mathbf{x}(\tau_k), u(\tau_k), \tau_k) \tag{12}$$

Similar to the previous section, constraints (inequality or equality) may be enforced at any collocation point; equality constraints used to enforce boundary conditions and inequality constraints used to implement path constraints.

The use of adaptive meshing for the PSM aids in computational efficiency. By increasing the polynomial order for the number of points in a mesh, it is feasible to map the entire time sequence to one mesh of points and one polynomial, however, with higher order polynomials, Runge Phenomenon, (although reduced because of the LGR collocation point selection) can be a source of error. In an effort to reduce the error from this phenomenon, HP-adaptive [5] meshing turns the mesh into a series of segments which are of no higher order than a prescribed maximum number. This means that mesh refinement occurs where necessary, and in the event nonlinearities in the dynamics require more precise computation of the states, the mesh is able to adapt.

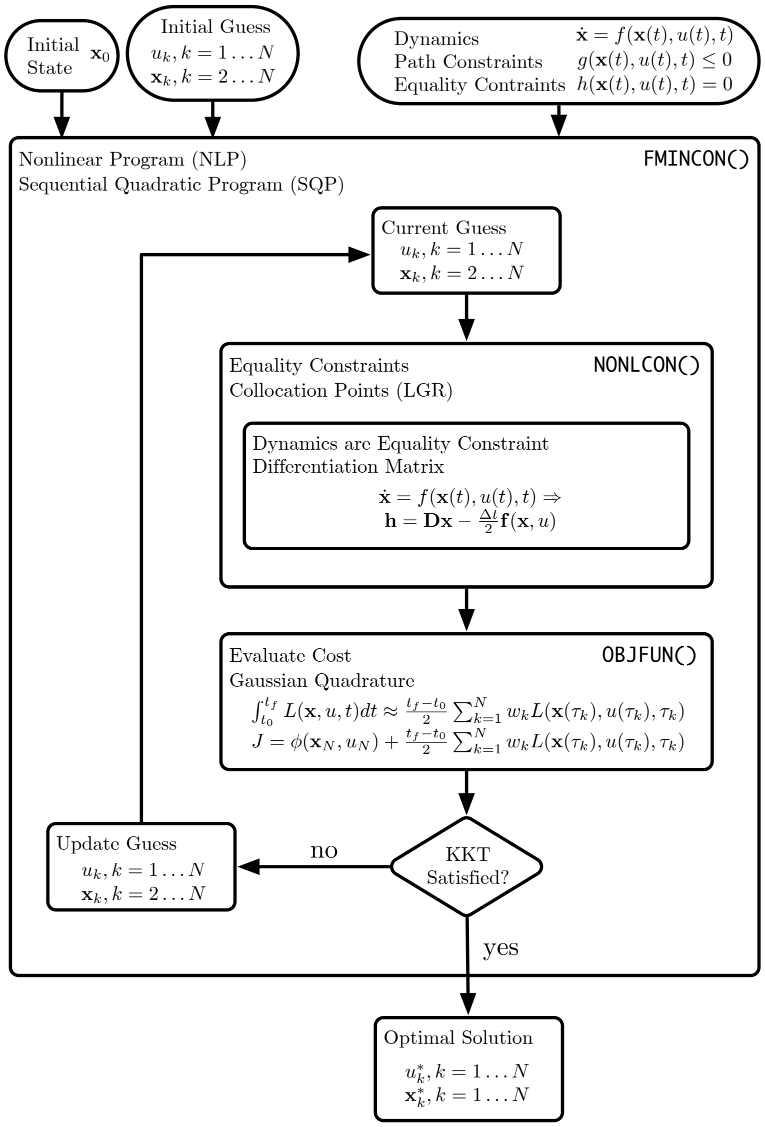
Initial State $\mathbf{x}_0$
Initial Guess $u_k, k = 1 \dots N$ $\mathbf{x}_k, k = 2 \dots N$
Dynamics $\dot{\mathbf{x}} = f(\mathbf{x}(t), u(t), t)$
Path Constraints $g(\mathbf{x}(t), u(t), t) \leq 0$
Equality Contraints $h(\mathbf{x}(t), u(t), t) = 0$
Nonlinear Program (NLP)
Sequential Quadratic Program (SQP)
FMINCON()
Current Guess $u_k, k = 1 \dots N$ $\mathbf{x}_k, k = 2 \dots N$
Equality Constraints
Collocation Points (LGR)
NONLCON()
Dynamics are Equality Constraint
Differentiation Matrix
$\dot{\mathbf{x}} = f(\mathbf{x}(t), u(t), t) \Rightarrow$
$\mathbf{h} = \mathbf{D}\mathbf{x} - \frac{\Delta t}{2}\mathbf{f}(\mathbf{x}, u)$
Evaluate Cost
Gaussian Quadrature
OBJFUN()
$\int_{t_0}^{t_f} L(\mathbf{x}, u, t)dt \approx \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k L(\mathbf{x}(\tau_k), u(\tau_k), \tau_k)$
$J = \phi(\mathbf{x}_N, u_N) + \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k L(\mathbf{x}(\tau_k), u(\tau_k), \tau_k)$
Update Guess $u_k, k = 1 \dots N$ $\mathbf{x}_k, k = 2 \dots N$
no
KKT Satisfied?
yes
Optimal Solution $u_k^*, k = 1 \dots N$ $\mathbf{x}_k^*, k = 1 \dots N$

Fig. 4 Pseudospectral Methods Flow Diagram

As far as computational size, PS methods tend to be the most efficient means of computing the optimal control and state trajectories to this author's knowledge. By leveraging sparce matrices and using matrix multiplications, the computational size is on the order of N for enforcing the dynamics and the order of N for computing objective cost functional. Exponential convergence is also seen as a result of using PS Methods. Since adaptive meshing is employed, the number of collocation points are reduced, and the size of the problem is less than the even collocation problem. Further, the use of Gaussian quadrature methods ensures that integrations are exact for the approximated polynomial. Also, the integral can be computed by a weighted summation of the polynomial evaluated at the collocation points. The number of function evaluations, assuming with N points, n states, and R iterations, will be of the order of $RN(n+1)$. The size of the PSM is equal to the ECM, but the efficient methods which are employed by the PSM produces more accurate approximation of the objective cost functional and more accurate implementation of the dynamics.

E. Comparison

Each of the four methods described has various advantages and disadvantages; Table 1 summarizes the various properties of the four methods. To begin with, the SSM is simple to construct, and initial guesses only require the control sequence. However, the SSM inefficiently computes the cost by shooting the dynamics for each evaluation. Moreover, shooting methods are very sensitive to the initial guess due to the gradient search methods employed by Sequential Quadratic Programming. Furthermore, since the initial guess is that of the control, the output of the NLP is only the optimal control, and obtaining the state trajectories which correspond to the solution requires an additional shooting of the dynamics.

Similar to the ECM, the MSM imposes dynamics as a series of inequality constraints; but, rather than using a first-order Euler approximation to the dynamics, an ODE solver is used to forward propagate the dynamics. Also, the introduction of states to the guess reduces sensitivity to the initial guess and allows for state and control to be provided at the end of the iterative search.

As compared to the SSM and MSM, the accuracy of the solution that ECM produces is poorer. While the SSM and MSM provide accurate integration using ODE solvers, the ECM uses a first-order Euler approximation. Furthermore, the use of rectangular integration by the ECM produces a coarse approximation of analytic integration. While the ECM aims at outperforming the speed of solution provided by SSM and MSM, it does so at the cost of accuracy. However, by setting the dynamics as a constraint rather than imposing the dynamics through shooting, the ECM is less sensitive to initial guess than SSM and MSM.

The PSM is for all intents and purpose, an accurate and efficient version of the ECM. Since it uses LGR collocation points and computes integration using Gaussian quadrature, integration is exact for the approximating polynomial and is found by a simple weighted sum. The PSM also makes use of a differentiation matrix to enforce the dynamics through equality constraints. This is much more efficient than the ECM's for-loop. The main disadvantage of PSM is the complexity and level of understanding required to implement the algorithm. Tools such as GPOPS-II are available to aid the novice control-theorist with setting up an optimization method using the PSM.

Table 1 Method Property Comparison

Metric	SSM	MSM	ECM	PSM
Easy to Formulate	x		x	
Guess includes Control	x	x	x	x
Guess includes State		x	x	x
Solves for Control Trajectory	x	x	x	x
Solves for State Trajectories		x	x	x
Solutions Sensitive to Init. Guess	x	x		
Employs ODE Solvers	x	x		
Eq. Constr. to Satisfy Dynamics		x	x	x
Low Convergence Times		x	x	x

Table 2 presents a summary of the size of the guess and equality constraints to set up each of the four methods. While at first glance, the shooting method appears to be the smallest size, it is the most computationally inefficient means of searching for the optimal control. Due to the number of ODE solver calls, shooting takes much more time to compute than the other three direct methods. Multiple shooting aimed at reducing the computational time by shooting segments in parallel but introduced continuity constraints. Although the computational size of the problem is reduced by parallelizing the ODE solver calls, the method requires special hardware and software to perform this operation. The

ECM removed the requirement to shoot the dynamics; instead, the dynamics were implemented through a series of equality constraints. Since the ECM does not require shooting the dynamics, a fine grid of points is required for the first-order Euler approximation for the dynamics to be sufficient. Additionally, the ECM is much less sensitive to initial guesses due to gradient-based search techniques. Finally, the PSM aims at reducing the error of rectangular integration and low-order dynamics by using Gaussian quadrature and spacing the points accordingly. In spacing the LGR points, the number of collocation points is reduced to achieve the same accuracy as the other direct methods, and as a result, solutions to the optimal control problem are found much faster. Table 6 shows a comparison of the size required to run each of the four direct methods.

Table 2 Method Size

Method	Initial Guess	No. Ukn.	Eq. Const.	NLP Output
SSM	`u[k], k = 1..N`	N	0	$u_k^*,\ k = 1\dots N$

MSM	`u[k], k = 1..N`
	`x[j], j = 2..M`
		$N + n(M-1)$	$n(M-1)$	$u_k^*,\ k = 1\dots N$
				$\mathbf{x}_j^*,\ j = 2\dots M$

ECM	`u[k], k = 1..N`
	`x[k], k = 2..N`
		$N + n(N-1)$	$n(N-1)$	$u_k^*,\ k = 1\dots N$
				$\mathbf{x}_k^*,\ k = 2\dots N$

PSM	`u[k], k = 1..N`
	`x[k], k = 2..N`
		$N + n(N-1)$	$n(N-1)$	$u_k^*,\ k = 1\dots N$
				$\mathbf{x}_k^*,\ k = 2\dots N$

IV. Direct Method Setup

The Pursuer-Defender-Evader engagement scenario in two-dimensional space is illustrated in Figure 5. In the engagement scenario, some of the underlying assumptions are that the velocities of the Defender, Pursuer, and Evader (V_d, V_p, and V_e) are assumed to be constant. Using Cartesian Coordinates, the dynamics of the three agents are described by:

$$\dot{x}_d = V_d \cos(\gamma_d) \qquad \dot{y}_d = V_d \sin(\gamma_d) \tag{13a}$$

$$\dot{x}_p = V_p \cos(\gamma_p) \qquad \dot{y}_p = V_p \sin(\gamma_p) \tag{13b}$$

$$\dot{x}_e = V_e \cos(\gamma_e) \qquad \dot{y}_e = V_e \sin(\gamma_e) \tag{13c}$$

The input to the system being the heading rate of the Evader aircraft:

$$\dot{\gamma}_e(t) = u(t) \tag{14}$$

Using this formulation, γ_x represents the heading angle for the agent x in the scenario. We have also made the assumption that the Pursuer is faster than the Evader, so that $V_p > V_e$. In this work, the Defender is assumed to be the same speed as the Pursuer, $V_d = V_p$, meaning that the Defender can not capture the Pursuer vehicle unless the Evader maneuvers in such a way as to aid the closure between the Defender and the Pursuer.

Using Cartesian Space, we define the line of sight (LOS) distance between agents to be the distance between the Defender and the Pursuer: R_{dp}, and the distance between the Pursuer and the Evader: R_{pe}. Further, the heading angles γ_x and line of sight angles λ_{xy} are used to describe the relative position information required to represent the dynamics of the three-agent-scenario.

The equations of motion are developed using the geometry in Fig. 5. We begin by calculating the line of sight angles derived from the Cartesian Space:

$$\lambda_{dp} = \tan^{-1}\left(\frac{y_p - y_d}{x_p - x_d}\right) \tag{15a}$$

$$\lambda_{pe} = \tan^{-1}\left(\frac{y_e - y_p}{x_e - x_p}\right) \tag{15b}$$

The angle between the velocity of an agent and its line of sight to its evader, σ, can be found by subtracting the line of sight to the next evader from the known heading:

$$\sigma_p = \gamma_p - \lambda_{pe} \tag{16a}$$

$$\sigma_d = \gamma_d - \lambda_{dp} \tag{16b}$$

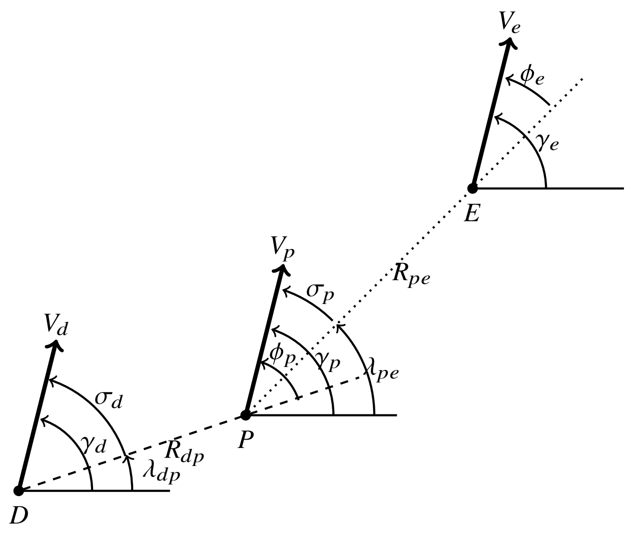

Fig. 5 Defender-Pursuer-Evader Geometry

Moreover, the angle between an agents velocity and his pursuer's line of sight can be calculated by subtracting the line of sight of the pursing agent from the evading agents heading:

$$\phi_p = \gamma_p - \lambda_{dp} \tag{17a}$$

$$\phi_e = \gamma_e - \lambda_{pe} \tag{17b}$$

The distance between the agents can be determined using the Euclidean distance:

$$R_{dp} = \sqrt{\left(y_p - y_d\right)^2 + \left(x_p - x_d\right)^2} \tag{18a}$$

$$R_{pe} = \sqrt{\left(y_e - y_p\right)^2 + \left(x_e - x_p\right)^2} \tag{18b}$$

The line of sight rates:

$$\dot{\lambda}_{dp} = \frac{V_p \sin\phi_p - V_d \sin\sigma_d}{R_{dp}} \tag{19a}$$

$$\dot{\lambda}_{pe} = \frac{V_e \sin\phi_e - V_p \sin\sigma_p}{R_{pe}} \tag{19b}$$

The angular heading rates are given by:

$$\dot{\gamma}_d = N_d \dot{\lambda}_{dp} \tag{20}$$

$$\dot{\gamma}_p = N_p \dot{\lambda}_{pe} \tag{21}$$

Where N_d and N_p are the proportional constants. Given the preceding analysis and identifying the input to be the heading rate of the Evader, the first-order model can be described as a system of differential equations:

$$\begin{bmatrix} \dot{x}_d \\ \dot{y}_d \\ \dot{\gamma}_d \\ \dot{x}_a \\ \dot{y}_a \\ \dot{\gamma}_p \\ \dot{x}_t \\ \dot{y}_t \\ \dot{\gamma}_e \end{bmatrix} = \begin{bmatrix} V_d \cos(\gamma_d) \\ V_d \sin(\gamma_d) \\ N_d \dot{\lambda}_{dp} \\ V_p \cos(\gamma_p) \\ V_p \sin(\gamma_p) \\ N_p \dot{\lambda}_{pe} \\ V_e \cos(\gamma_e) \\ V_e \sin(\gamma_e) \\ u \end{bmatrix} \tag{22}$$

Although the problem may be tractable using these dynamic equations of motion, it is possible to reduce the order of the dynamics; in order to formulate the problem in this reduced form, we transform the dynamics from (22) into polar form. By calculating and solving the optimal control problem in relative polar coordinates, calculation time is reduced since the number of nonlinear dynamic equations of motion is reduced. Once an optimal solution is found in the relative polar space, the optimal control may be fed back into Cartesian Space for visualization and evaluation.

The polar representation can be formed by taking the velocities along the line of sights and analyzing their closure based upon the line of sight angles:

$$\dot{R}_{dp} = -V_d \cos\sigma_d + V_p \cos\phi_p \tag{23a}$$

$$\dot{R}_{pe} = -V_p \cos\sigma_p + V_e \cos\phi_e \tag{23b}$$

Continuing the transformation of the dynamics from Cartesian to relative polar space: the headings with respect to the line of sight σ_p and σ_d can be found by differentiating (16) and (17) with respect to time.

To complete the polar form of the dynamics equations we need the angular rates of each of the agents in the engagement. The rate of change between the Pursuer's heading and its line of sight to the Evader may be found to be the difference between the heading rate of the Pursuer and the line of sight rate between the Pursuer and the Evader, namely:

$$\dot{\sigma}_a = \dot{\gamma}_p - \dot{\lambda}_{pe} = (N_p - 1)\dot{\lambda}_{pe} = \frac{(N_p - 1)(V_e \sin\phi_e - V_p \sin\sigma_p)}{R_{pe}} \tag{24}$$

Similarly, the rate of change between the Defender's heading and its line of sight to the Pursuer may be found to be the difference between the heading rate of the Defender and the line of sight rate between the Defender and pursuer:

$$\dot{\sigma}_d = \dot{\gamma}_d - \dot{\lambda}_{dp} = (N_d - 1)\dot{\lambda}_{dp} = \frac{(N_d - 1)(V_p \sin\phi_p - V_d \sin\sigma_d)}{R_{dp}} \tag{25}$$

Next, by differentiating (17) we find the angular rate of change the Pursuer's heading and Defender's line of sight to be:

$$\dot{\phi}_p = \dot{\gamma}_p - \dot{\lambda}_{dp} = N_p\dot{\lambda}_{pe} - \dot{\lambda}_{dp} = \frac{N_p(V_e \sin\phi_e - V_p \sin\sigma_p)}{R_{pe}} - \frac{V_p \sin(\phi_p) - V_d \sin(\sigma_d)}{R_{dp}} \tag{26}$$

And similarly for the Evader, the angular rate of change of the Evader's heading to the Pursuer's line of sight is calculated. Recall that the input, u, is the rate of the Evader's heading. This will appear in the Evader's angular rate of change relative to the line of sight rate:

$$\dot{\phi}_e = \dot{\gamma}_e - \dot{\lambda}_{pe} = u - \frac{V_e \sin(\phi_e) - V_p \sin(\sigma_p)}{R_{pe}} \tag{27}$$

The final reduced order form in relative polar space is:

$$\begin{bmatrix} \dot{R}_{dp} \\ \dot{R}_{pe} \\ \dot{\sigma}_d \\ \dot{\sigma}_a \\ \dot{\phi}_p \\ \dot{\phi}_e \end{bmatrix} = \begin{bmatrix} V_p \cos\phi_p - V_d \cos\sigma_d \\ V_e \cos\phi_e - V_p \cos\sigma_p \\ (N_d - 1)\dot{\lambda}_{dp} \\ (N_p - 1)\dot{\lambda}_{pe} \\ N_p\dot{\lambda}_{pe} - \dot{\lambda}_{dp} \\ u - \dot{\lambda}_{pe} \end{bmatrix} \tag{28}$$

where

$$\begin{aligned} \dot{\lambda}_{dp} &= \frac{V_p \sin\phi_p - V_d \sin\sigma_d}{R_{dp}} \\ \dot{\lambda}_{pe} &= \frac{V_e \sin\phi_e - V_p \sin\sigma_p}{R_{pe}} \end{aligned} \tag{29}$$

The objective of the three-agent scenario is for the Evader to out maneuver the Pursuer by maneuvering in such a manner to close the distance between the Defender and the Pursuer while distancing itself from the Pursuer as far as possible at the time of capture. Formulating in terms of closure rates we strive at maximizing the separation rate of the Pursuer-Evader range, and minimize the range rate from the Defender to the Pursuer:

$$\min_u J = \int_{t_0}^{t_f} \left(-\dot{R}_{pe} + \dot{R}_{dp}\right) dt \tag{30}$$

Evaluating the integral:

$$\min_u J = -\left(R_{pe}(t_f) - R_{pe}(t_0)\right) + \left(R_{dp}(t_f) - R_{dp}(t_0)\right) \tag{31}$$

Since the initial conditions are unaffected by the control, an equivalent minimization can be written in terms of the final states as:

$$\min_u J = -R_{pe}(t_f) + R_{dp}(t_f) \tag{32}$$

Now that the state dynamics and objective have been outlined, we define the constraints on the input (heading rate constraint) to be:

$$|u(t)| \leq \omega_{max} \ \forall \ t \in [t_0, t_f] \tag{33}$$

The boundary conditions for the states of the fixed final time, free final state scenario are:

$$\begin{bmatrix} R_{dp}(t_0) \\ R_{pe}(t_0) \\ \sigma_d(t_0) \\ \sigma_p(t_0) \\ \phi_p(t_0) \\ \phi_e(t_0) \end{bmatrix} = \begin{bmatrix} R_{da_0} \\ R_{at_0} \\ \sigma_{d_0} \\ \sigma_{a_0} \\ \phi_{a_0} \\ \phi_{t_0} \end{bmatrix} \quad \& \quad \begin{bmatrix} R_{dp}(t_f) \\ R_{pe}(t_f) \\ \sigma_d(t_f) \\ \sigma_p(t_f) \\ \phi_p(t_f) \\ \phi_e(t_f) \end{bmatrix} = \begin{bmatrix} \text{free} \\ \text{free} \\ \text{free} \\ \text{free} \\ \text{free} \\ \text{free} \end{bmatrix} \tag{34}$$

A. Direct Optimization Problem Definition

In an effort to compare the different methods of optimal control definitions for each method are as follows. Find the optimal control: $u^*(t)$ which produces a minimum:

$$\min_u J = -R_{pe}(t_f) + R_{dp}(t_f) \tag{35}$$

Subject to dynamics:

$$\begin{bmatrix} \dot{R}_{dp} \\ \dot{R}_{pe} \\ \dot{\sigma}_d \\ \dot{\sigma}_a \\ \dot{\phi}_p \\ \dot{\phi}_e \end{bmatrix} = \begin{bmatrix} V_p \cos\phi_p - V_d \cos\sigma_d \\ V_e \cos\phi_e - V_p \cos\sigma_p \\ (N_d - 1)\dot{\lambda}_{dp} \\ (N_p - 1)\dot{\lambda}_{pe} \\ N_p \dot{\lambda}_{pe} - \dot{\lambda}_{dp} \\ u - \dot{\lambda}_{pe} \end{bmatrix} \tag{36}$$

with

$$\dot{\lambda}_{dp} = \frac{V_p \sin\phi_p - V_d \sin\sigma_d}{R_{dp}} \tag{37a}$$

$$\dot{\lambda}_{pe} = \frac{V_e \sin\phi_e - V_p \sin\sigma_p}{R_{pe}} \tag{37b}$$

The constraints on the input are:

$$|u(t)| \leq \omega_{max} \ \forall \ t \in [t_0, t_f] \tag{38}$$

The boundary conditions on the states are:

$$\begin{bmatrix} R_{dp}(t_0) \\ R_{pe}(t_0) \\ \sigma_d(t_0) \\ \sigma_p(t_0) \\ \phi_p(t_0) \\ \phi_e(t_0) \end{bmatrix} = \begin{bmatrix} R_{da_0} \\ R_{at_0} \\ \sigma_{d_0} \\ \sigma_{a_0} \\ \phi_{a_0} \\ \phi_{t_0} \end{bmatrix} \quad \& \quad \begin{bmatrix} R_{dp}(t_f) \\ R_{pe}(t_f) \\ \sigma_d(t_f) \\ \sigma_p(t_f) \\ \phi_p(t_f) \\ \phi_e(t_f) \end{bmatrix} = \begin{bmatrix} \text{free} \\ \text{free} \\ \text{free} \\ \text{free} \\ \text{free} \\ \text{free} \end{bmatrix} \tag{39}$$

The final time, t_f, is judiciously chosen prior to capture.

B. Numerical Consideration

For all four methods, each use the SQP search implemented within MATLAB using the function `FMINCON()`. Furthermore, the initial conditions and final time are the same for each simulation. In order to ensure that the same problem is being posed for each direct method, the parameters outline in Table 3 provide a summary of the constants within the equations of motion. These values are implemented for each direct method.

Table 3 Simulation Parameters

Parameter	Variable	Value
Pursuer Velocity	V_p	1.00 m/s
Defender Velocity	V_d	1.00 m/s
Evader Velocity	V_e	0.50 m/s
Pursuer PN Gain	N_a	3.00
Defender PN Gain	N_d	3.00
Final Time	t_f	11 sec
Max Turn Rate	ω_{max}	0.200 rad/sec
Number Points	N	50

The initial conditions of the states are identical for all four simulations. Figure 6 describes the initial conditions used for each simulation. Numerically, the values in Cartesian Space are summarized in Table 4.

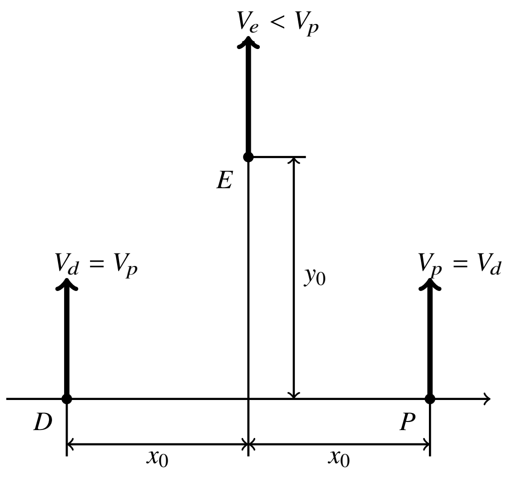

Fig. 6 Defender-Pursuer-Evader Geometry

Converting the Cartesian coordinates to the relative polar coordinate frame we have the initial conditions defined in Table 5.

Table 4 Initial State (Cartesian Coords)

State	Variable	Value
Defender X Coord	$x_d(t_0)$	-5
Defender Y Coord	$y_d(t_0)$	0
Defender Heading	$\gamma_d(t_0)$	$\pi/2$ rad
Pursuer X Coord	$x_a(t_0)$	5
Pursuer Y Coord	$y_a(t_0)$	0
Pursuer Heading	$\gamma_p(t_0)$	$\pi/2$ rad
Evader X Coord	$x_t(t_0)$	0
Evader Y Coord	$y_t(t_0)$	10
Evader Heading	$\gamma_e(t_0)$	$\pi/2$ rad

Table 5 Initial State (Relative Polar)

State	Variable	Value
Defender-Pursuer Distance	$R_{dp}(t_0)$	10.0000
Pursuer-Evader Distance	$R_{pe}(t_0)$	11.1803
Closing Heading Angle (Defender)	$\sigma_d(t_0)$	1.5708
Closing Heading Angle (Pursuer)	$\sigma_p(t_0)$	-0.4636
Escaping Heading Angle (Pursuer)	$\phi_p(t_0)$	1.5708
Escaping Heading Angle (Evader)	$\phi_e(t_0)$	-0.4636

C. Direct Method Setup

In an effort to make a fair comparison between each of the direct methods, the initial conditions, dynamics, and number of points for the trajectories are held constant. Moreover, the convergence criteria, used to determine if a minimum has been found, is also held constant. Aside from the actual algorithmic implementation of each direct method, the guess size and corresponding discretization are varied to meet the requirements of each direct method.

In the case of the SSM, we define the discretization of time and control to be evenly spaced into 50 points. By implementing the Runge-Kutta Method for forward propagation of the equations of motion, the fixed-time step solver provides trajectories of the states at every point. The initial guess for the optimal control is a max positive heading rate command. The total size of the initial guess is 50.

For the MSM, time is discretized evenly from t_0 to t_f with a total of N points, just as defined for the SSM. Similar to the SSM, the Runge-Kutta Method propagates states forward through time. The number of segments used for the MSM is 5; resulting in 24 equality constraints (See Table 7). At each of the continuity constraints, the initial guess for the states is the the value of the state at the initial condition. The guess for the optimal control, just like the SSM, is a max positive heading rate command. The total size of the guess is 74.

The ECM implements the dynamics as an equality constraint, and as a result, requires a much larger guess than the SSM and MSM. Time is discretized evenly from t_0 to t_f with a total of 50 points. Since the initial state is provided as presented in Table 5, the size of the guess for the states is 294. Adding the size for the control makes the total guess size 344. The dynamics are upheld using a lower order method: first-order Euler approximation. In order to make the initial guess a fair comparison, the state trajectories are guessed to be the initial conditions at every point and the optimal control is guessed to be a max positive heading rate command at every point.

For the case of the PSM, time is broken into 3 segments, and for each segment the LGR points set the discretization of time so that differentiation matrices may be implemented. Just as implemented for the ECM, the number of points for the state trajectories and control are the same. Using 3 segments, the size of the Differentiation Matrices are 16×16 for the first segment, and 17×17 for the remaining two segments. Numerically, this is more work than implementing the dynamics described by the ECM, but much more accurate. The total size of the guess, for the PSM is the same as the ECM: 344. In an effort to make the initial guess a fair comparison to the other methods, the state trajectories are guessed to be the initial conditions at every point and the optimal control is guessed to be a max positive heading rate command at every point. To conclude the setup of each of the four direct methods, Table 7 describes the objective cost, initial guess, and type of discretization.

Table 6 Method Size

Method	Initial Guess	No. Ukn.	Eq. Const.	NLP Output
SSM	`u[k], k = 1..50`	50	0	$u_k^*,\ k = 1\ldots50$
MSM	`u[k], k = 1..50` `x[j], j = 2..5`	$50 + 6(5-1) = 74$	$6(5-1) = 24$	$u_k^*,\ k = 1\ldots N$ $x_j^*,\ j = 2\ldots M$
ECM	`u[k], k = 1..50` `x[k], k = 2..50`	$50 + 6(50-1) = 344$	$6(50-1) = 294$	$u_k^*,\ k = 1\ldots50$ $x_k^*,\ k = 2\ldots50$
PSM	`u[k], k = 1..50` `x[k], k = 2..50`	$50 + 6(50-1) = 344$	$6(50-1) = 294$	$u_k^*,\ k = 1\ldots50$ $x_k^*,\ k = 2\ldots50$

Table 7 Direct Method Setup

Method	Objective Cost	Initial Guess	Guess Size	Discretization
SSM	$J = -R_{pe}(t_f) + R_{dp}(t_f)$	$u_k = \omega_{max},\ k = 1\ldots50$	50	Evenly Spaced 1 Shooting
MSM	$J = -R_{pe}(t_f) + R_{dp}(t_f)$	$u_k = \omega_{max},\ k = 1\ldots50$ $\mathbf{x}_j = \mathbf{x}_0,\ j = 2\ldots5$	74	Evenly Spaced 5 Shootings
ECM	$J = -R_{pe}(t_f) + R_{dp}(t_f)$	$u_k = \omega_{max},\ k = 1\ldots50$ $\mathbf{x}_k = \mathbf{x}_0,\ k = 2\ldots50$	344	Evenly Spaced Collocation Points
PSM	$J = -R_{pe}(t_f) + R_{dp}(t_f)$	$u_k = \omega_{max},\ k = 1\ldots50$ $\mathbf{x}_k = \mathbf{x}_0,\ k = 2\ldots50$	344	LGR Collocation 5 Segments

V. Simulation Results

A summary of the four direct-method simulations can be seen in Table 8. In the table, multiple columns representing interesting metrics for each of the direct methods are presented. The metrics presented are computation time, number of iterations, number of functional evaluations, function evaluation, and feasibility of solution. The computation time represents the total time it takes from feeding the NLP the initial guess to the time the optimal control and state trajectories are produced. The number of iterations is the number of times the NLP updates the guess being optimized. Function evaluations are the number of times the objective cost function is calculated until the convergence criteria of the NLP occurred. The 'Fval' column represents the value of the objective cost functional. The feasibility represents 2-norm of the vector containing all the equality constraints upon convergence. In the MSM the feasibility represents the total error in the continuity constraints for each shooting interval. In the case of the ECM and PSM, the feasibility represents the accuracy by which the model of the dynamics is upheld. For the sake of comparison, the PSM uses LGR collocation points and a differentiation matrix which is much more accurate than using first-order Euler approximation. Although the feasibility of the ECM is better than the PSM, the method of implementing the dynamics performed by the PSM is more accurate than the ECM.

The total computation time for each direct method is presented in the third column of Table 8. From this column we observe that the fastest direct method is the ECM, while the slowest is the SSM. The relative slow convergence times produced by the shooting methods are a reminder than forward propagation of the dynamics by means of an ODE solver is not computationally efficient. Moreover, we observe that the MSM makes efficient use of computing the shootings in parallel, and that the addition of the continuity constraints does not out-weigh the cost of breaking the problem into multiple segments. The collocation methods employed by the ECM and PSM are an efficient means of transcribing the dynamic problem into a static one. Although the PSM is slower than the ECM, this is attributed to the number of collocation points begin held constant across all methods. Since the LGR selected points give rise to the efficient computational methods previously described, the number of points required the the PSM to obtain a solution with the same accuracy is less. In the effort to make a fare computational comparison, we kept the number of collocation points constant. By using 50 points broken into 3 segments, the interpolating polynomials are either 16th or 17th order. Further, the implementation of fixed meshing makes it challenging to have enough points were necessary. As a result, the PSM would have faster convergence if fewer points and adaptive meshing were implemented.

The number of iterations and function evaluations conducted by the even collocation method were much larger than the other direct methods. Since the method converted the dynamics into nonlinear constraints, a large number of iterations and evaluations were required to satisfy the feasibility requirement. Even though more function evaluations were required, the effort to compute the equality constraints was far less expensive than using an ODE solver.

The final objective cost functional produced by all four methods are very similar. From Table 8 we observe that the range of the Pursuer to the Evader upon capture of the Pursuer by the Defender is approximately 2.7. The lower performance of the ECM and PSM are attributed to the lack of adaptive meshing for both the ECM and PSM.

Table 8 Direct Method Performance

Method	NLP	Comp. Time	Iter	F. Evals	Fval	Feasibility
SSM	`FMINCON()`	14.1 sec	32	1632	-2.76	0
MSM	`FMINCON()`	5.4 sec	27	2214	-2.76	1.788E − 9
ECM	`FMINCON()`	1.2 sec	31	10881	-2.75	1.019E − 9
PSM	`FMINCON()`	8.2 sec	21	6930	-2.71	3.590E − 8

In Figure 7 we observe the engagement computed by each of the direct methods. From this figure we observe that each of the four methods computes a similar solution for the optimal evasion engagement. Moreover, the states of the optimization are visualized in Figures 7-10. In Figure 8 we observe the heading angles for each of the agents. From the figure we observe that around 7.5-8.5 seconds, the Evader's Heading makes a drastic change, while the Pursuer and Defender have smooth curves described by the PN equations. The line of sight ranges shown in Figure 9 show that a race to zero between the two line of sight rates occurs. If the line of sight range R_{pe} reaches zero prior to R_{dp} then the Evader is captured; but, if instead R_{dp} goes to zero faster than R_{pe} then the Evader successfully escapes the Pursuer. In our engagement, the Evader successfully escapes the Pursuer by maneuvering in such a way, as to aid the Defender. In Figure 10 the optimal control determined by each of the direct methods is presented. We can see the bang-bang behavior, an outcome due to the nature of the limit on control. In all four cases the transition from a positive max turn rate to negative max turn rate occurs around 8 seconds.

VI. Conclusion

A three-agent engagement involving a Pursuer, Evader, and Defender is considered. The control strategy of the Evader is to maneuver in such a way as to aid the Defender in capturing the Pursuer before he captures the Evader. From the simulation results we compare multiple direct methods capable of computing the optimal control to the three-agent engagement. The four methods implemented are the Single Shooting Method, Multiple Shootings Method, Even Collocation Method, and Pseudospectral Method. In all four cases the optimal heading rate of the Evader was solved to complete a desired objective: to aid the Defender in capturing the Pursuer while distancing itself from the Pursuer. In an effort to make the problem more realistic, turning rate constraints were imposed on the Evader, while a Proportional Navigation guidance algorithm was implemented for the Defender and Pursuer. The optimal control involved a maximum turn rate command toward the Defender followed by a maximum turn rate command away from the Defender. Using MATLAB's `FMINCON()`, the four methods were implemented and an iterative search using a sequential quadratic program was implemented to find the optimal control. Considering the three-player engagement, the four methods found similar solutions for the optimal control and the corresponding state trajectories.

The Single Shooting Method and Multiple Shootings Method both required the use of Ordinary Differential Equation (ODE) solvers to forward propagate the states through time. While the Single Shooting Method forward propagated the entire time series in one shooting, the Multiple Shooting Method parallelized the shooting into segments in an effort to reduce the convergence time. Introducing segments allowed the Multiple Shooting Method to take advantage of parallel computing techniques, but required the introduction of continuity constraints between every segment. For the specific scenario simulated, we observed the implementation of the Multiple Shootings Method reduced computation time by 8.7 seconds, with nearly identical solutions for control, state trajectories, and cost.

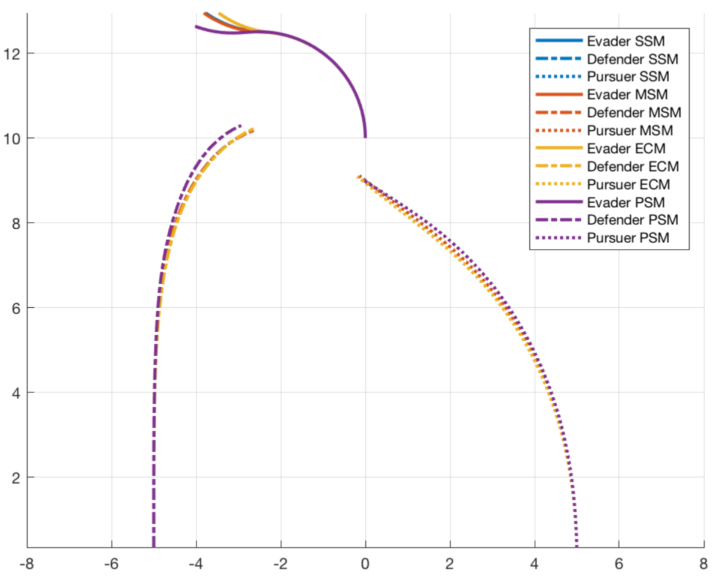

Fig. 7 Engagement Comparison

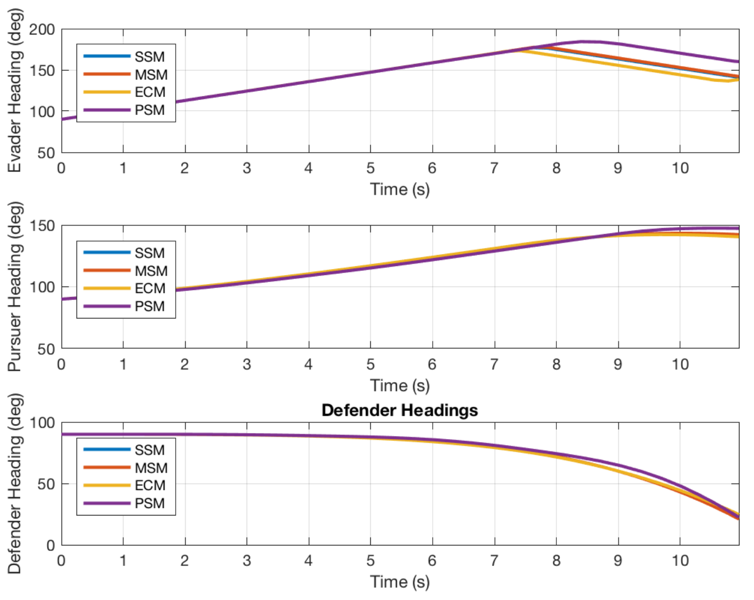

Fig. 8 Agent Heading Comparison

Next, we considered collocated direct methods such as the Even Collocation and Pseudospectral Method. By implementing the dynamics as a series of equality constraints at each collocation point, these methods were able to transcribe the dynamic optimization problem into a static one. This transcription relaxed the requirement of using an ODE solver to forward propagate the dynamics. Another added benefit of transcription was the reduced sensitivity to the initial guess. Furthermore, the Even Collocation Method used the first-order Euler Method for constraining the dynamics, while the Pseudospectral Method used Lagrange interpolating polynomials and differentiation matrices to

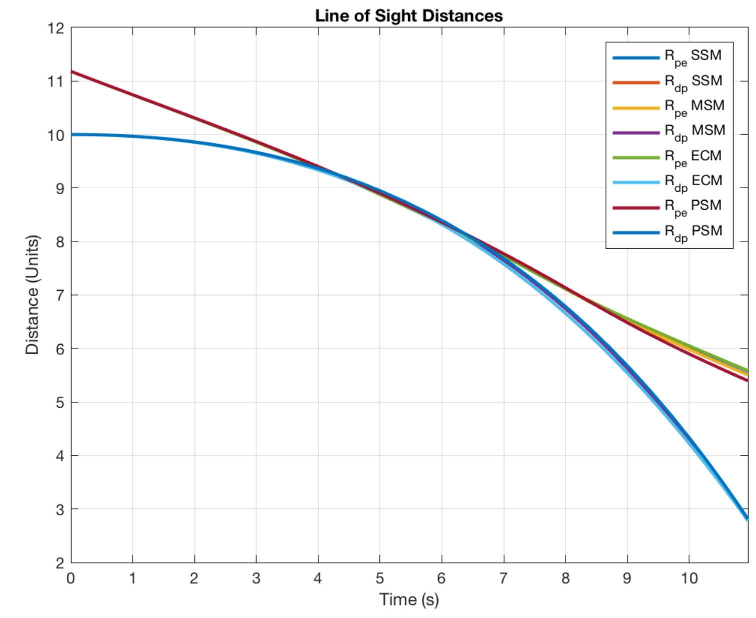

Fig. 9 Line of Sight Comparison

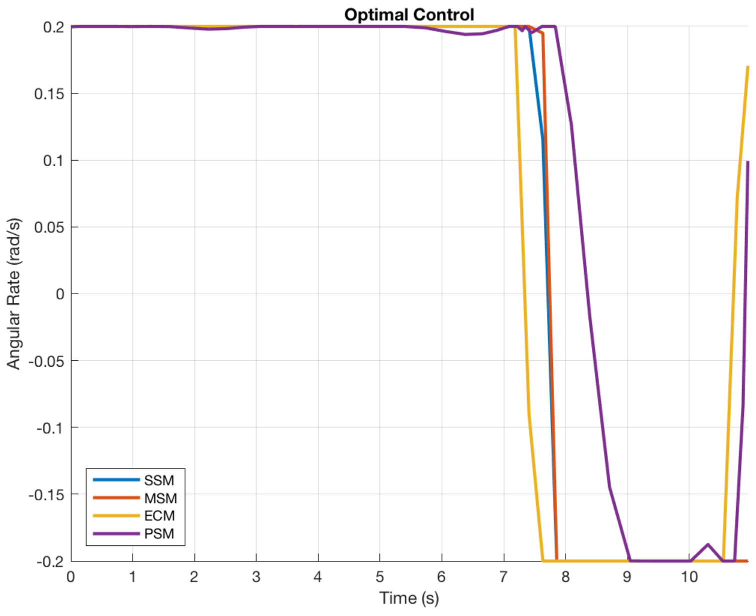

Fig. 10 Optimal Control Comparison

form the equality constraints. Because of the 16th and 17th order polynomials as well as fixed meshing techniques, the Pseudospectral Method did not converge to an optimal solution as fast as the ECM. Furthermore, the accuracy of the Pseudospectral Method solution was hindered by the fixed segmentation of the problem, and could have benefited by implementing adaptive meshing techniques. Since fixed-time step ODE solvers were implemented for the shooting methods the use of fixed meshing of the collocation points seemed to be a fair comparison. It should also be noted, all methods could have benefited by using adaptive meshing techniques. Rather than using fixed time step ODE solvers and meshes, adaptive techniques would aid solution accuracy of all four direct methods. In summary, the Even Collocation Method outperformed the shooting methods, only taking 1.2 seconds to find the optimal control and state trajectories.

Assuming a fixed mesh, the Even Collocation Method outperformed the convergence time of the other direct methods. Since the dynamics are approximated using low-order methods, and the use of ODE solvers is no longer required, the solution converges very fast. However, the use of ODE solvers to forward propagate the dynamics makes shooting methods more accurate. In a real-world scenario, were the Evader is a piloted aircraft, fast solutions of low fidelity are more valuable than late solutions of high-fidelity. For this reason, the Even Collocation Method outperforms the other direct methods and is more appropriate for real-time applications.

References

[1] Kirk, D. E., *Optimal Control Theory: An Introduction*, Dover Books on Electrical Engineering Series, Dover Publications, 2004.

[2] Butcher, J. C., "Runge–Kutta Methods," *Numerical Methods for Ordinary Differential Equations*, Wiley-Blackwell, 2008, Chap. 3, pp. 137–316. doi:10.1002/9780470753767.ch3.

[3] Nocedal, J., and Springer, S. J. W., *Numerical Optimization*, Springer-Verlag New York, Inc., 1999. doi:10.1007/978-0-387-40065-5{_}18.

[4] Fahroo, F., and Ross, I. M., "Costate Estimation by a Legendre Pseudospectral Method," ???? doi:10.2514/6.1998-4222, URL `https://arc-aiaa-org.wrs.idm.oclc.org/doi/pdf/10.2514/6.1998-4222`.

[5] Darby, C. L., and Rao, A. V., "A State Approximation-Based Mesh Refinement Algorithm for Solving Optimal Control Problems Using Pseudospectral Methods," *AIAA Guidance, Navigation and Control Conference*, AIAA, Chicago, IL, 2009, pp. 1–26. doi:10.2514/6.2009-5791.