MPSP Whitepaper

Sparsh Yadav

May 11, 2020

Contents

1 Introduction

Numerous complex problems in engineering can be formulated as optimal control problems. A general optimal control problem can be formulated as:

$$J(\mathbf{x}(.), \mathbf{u}(.)) = E\left(\mathbf{x}\left(t_0\right), \mathbf{x}\left(t_f\right)\right) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t))dt \tag{1}$$

Subject to

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{2}$$

$$\mathbf{x}\left(t_0\right) = \mathbf{x_0} \tag{3}$$

$$\mathbf{x}\left(t_f\right) = \mathbf{x_f} \tag{4}$$

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

The techniques for solving such optimal control problems can be broadly classified into direct and indirect techniques.

Indirect methods rely on the first-order necessary conditions and Pontryagin's minimum principle. These methods lead to a two-point boundary value problem and try to find the optimal trajectories for the state and the control by solving the TPBVP. Indirect methods are computationally intensive and the control solution is an open-loop solution which is undesirable as it can not account for disturbances.

Alternatively, direct methods transform the optimal control problem into a non-linear constrained optimization problem by discretization of the control input and the system dynamics at finite grid points. Mode Predictive static programming falls into this category.

2 MPSP Philosophy

MPSP combines the philosophy of MPC and Adaptive Dynamic Programming to solve a class of finite-horizon optimal control problems. It is an iterative technique that starts from "guess history" of the control solution. This guess solution is iterated upon to achieve the objective of minimum

control effort. The key advantage here is the computational efficiency i.e. the iterations can be calculated with very less computational load.

The flow chart in Figure 1.1 describes the iterative procedure of the algorithm.

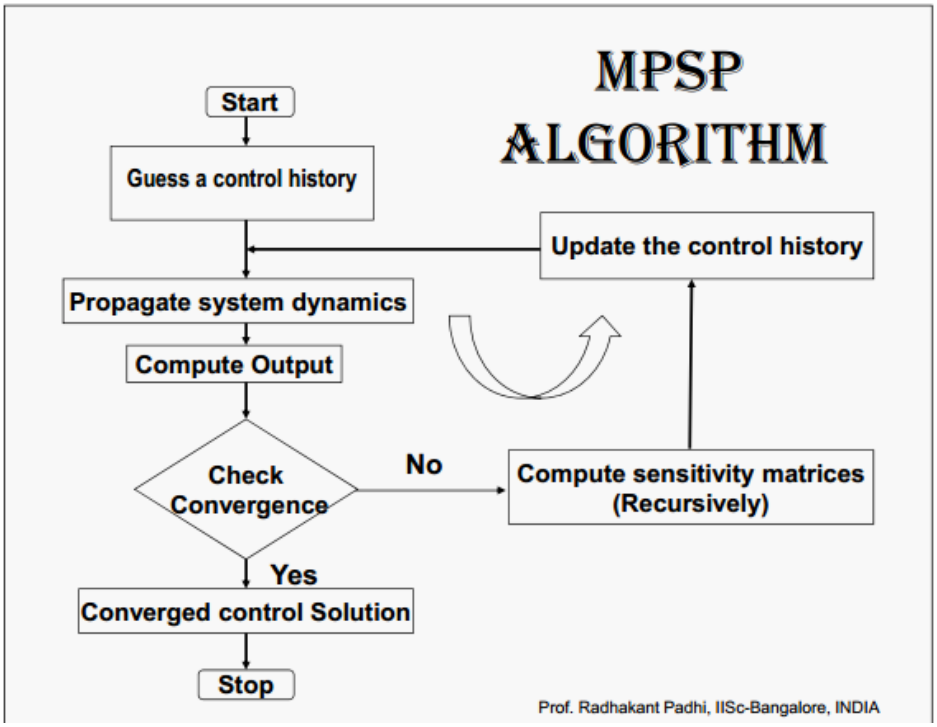

Figure 1: Flowchart of the MPSP Algorithm

3 MPSP Mathematical Background

In regular MPSP, a general nonlinear system is considered (in the discrete domain), the state and output equations are given by

$$X_{k+1} = F_k\left(X_k, U_k\right) Y_k = h_k\left(X_k\right) \tag{6}$$

where $X \in \Re^n, U \in \Re^m, Y \in \Re^p$ and $k = 1, 2, \ldots, N$ are the time steps. The primary objective is to obtain a suitable control history $U_k, k = 1, 2, \ldots, N-1$, so the output at the final time step Y_N reaches a desired value Y_N^* i.e. Y_N^*, i.e. $Y_N \to Y_N^*$

For mathematical development, the error is defined as $\triangle Y_N \triangleq Y_N - Y_N^*$ Next, using small error approximation we get

$$\Delta Y_N \cong dY_N = \left[\frac{\partial Y_N}{\partial X_N}\right] dX_N \tag{7}$$

However, we can write the error in state at time $(k+1)$ as

$$dX_{k+1} = \left[\frac{\partial F_k}{\partial X_k}\right] dX_k + \left[\frac{\partial F_k}{\partial U_k}\right] dU_k \tag{8}$$

where dX_k and dU_k are the errors in state and control, respectively, at time k.

Expanding iteratively until $k = 1$, we get

$$dY_N = AdX_1 + B_1 dU_1 + B_2 dU_2 + \cdots + B_{N-1} dU_{N-1} \tag{9}$$

where,

$$\begin{aligned} A &\triangleq \left[\frac{\partial Y_N}{\partial X_N}\right]\left[\frac{\partial F_{N-1}}{\partial X_{N-1}}\right]\cdots\left[\frac{\partial F_1}{\partial X_1}\right] \\ B_k &\triangleq \left[\frac{\partial Y_N}{\partial X_N}\right]\left[\frac{\partial F_{N-1}}{\partial X_{N-1}}\right]\cdots\left[\frac{\partial F_{k+1}}{\partial X_{k+1}}\right]\left[\frac{\partial F_k}{\partial U_k}\right], \quad k = 1, \ldots, N-2 \\ B_{N-1} &\triangleq \left[\frac{\partial Y_N}{\partial X_N}\right]\left[\frac{\partial F_{N-1}}{\partial U_{N-1}}\right] \end{aligned} \tag{10}$$

Equation 9 is solved with the cost function in 11 using static optimization techniques to get the update Equation 12 for the control, such that deviation at final-time is minimized.

$$J = \frac{1}{2}\sum_{k=1}^{N-1}\left(U_k^0 - dU_k\right)^T R_k \left(U_k^0 - dU_k\right) \tag{11}$$

$$U_k = U_k^0 - dU_k = R_k^{-1} B_k^T A_\lambda^{-1}\left(dY_N - b_\lambda\right), \quad k = 1, 2, \ldots, (N-1) \tag{12}$$

where,

$$A_\lambda \triangleq \left[-\sum_{k=1}^{N-1} B_k R_k^{-1} B_k^T\right], b_\lambda \triangleq \left[\sum_{k=1}^{N-1} B_k U_k^0\right] \tag{13}$$

3.1 Why MPSP is computationally efficient?

1. A dynamic programming problem is converted to a static programming problem, and thus it requires only a static costate vector for the control update
2. The symbolic costate vector has a closed-form solution, which reduces the computational load.
3. The sensitivity matrices that are necessary to compute the static costate vector can be calculated recursively.

4 Applications of MPSP

The algorithm's feasibility and efficiency has been demonstrated on several benchmarks (double integrator, Zermelo) and complex engineering problems. The focus here is on real-time aerospace applications.

- Aerospace Applications
  - Ascent phase guidance of ballistic missiles [1]
  - Guidance of interceptors with hard constraints [2, 3]
  - Guidance of solid-motor propelled flight vehicles [4]
  - Guidance of solid-motor propelled flight vehicles with flexible final time [5]
  - Re-entry guidance of Reusable Launch Vechicles [6, 7]
  - Guidance of Air-launched missile to a predetermined target [8]
  - Soft Lunar landing [9]
- Robotics
  - Trajectory tracking for robotics [10]
  - Inverted Pendulum [11]

In addition to the above applications, the algorithm is equally applicable to any area which necessitates the formulation of an optimal control problem, such as

- Biomedical: Artificial Pancreas
- Electrical and Process Control

5 A brief history of MPSP techniques

5.1 MPSP 2009

MPSP combines the philosophy of non-linear model predictive control and approximate dynamic programming. The basic version of MPSP is applicable for finite-horizon non-linear problems with terminal constraints was presented in [1]. It brings the concept of trajectory optimization into guidance laws which are optimal in regards to the control effort required.

MPSP technique is computationally efficient and can be implemented online, providing for a wide range of real-time applications. The effectiveness of the technique is demonstrated for the ascent phase guidance of a ballistic missile using solid motors in [1].

5.2 GMPSP 2014

A generalized version of MPSP was presented in [12] that extends upon the MPSP philosophy for solving a class of finite-horizon nonlinear optimal control problems with a hard terminal constraint. Two key features for its high computational efficiency include one-time backward integration of a small-dimensional weighting matrix dynamics, followed by a static optimization formulation that requires only a static Lagrange multiplier to update the control history. Moreover, it turns out that under Euler integration and rectangular approximation of finite integrals it is equivalent to model predictive static programming technique [1].

5.3 QS-MPSP 2018

Quasi-spectral MPSP was proposed in [3] where the control variable is expressed as a weighted sum of basis functions. The coefficients of the basis functions are then determined using optimization techniques, in contrast, to control input at every grid point. This reduces the complexity of the problem and also ensures the smoothness of the control variable. Such a procedure also ensures the spread of the control over the available time to go.

QS-MPSP guidance was demonstrated in [3] by formulating and solving an angle-constrained missile guidance problem to successfully engage an incoming high-speed ballistic missile target in three dimensions. The proposed QS-MPSP guidance showed considerable improvement in the lateral

acceleration demand over the BPN guidance scheme.

5.4 T-MPSP 2019

A nonlinear optimal command tracking technique was presented in [10] called the Tracking-oriented MPSP. In this technique, like MPC a model-based prediction-correction approach is adopted.

In [10] TMPSP was used for trajectory tracking problem of a two-wheel differential drive mobile robot both in simulations and on real hardware. Successful results in both simulation and hardware implementation study demonstrated that the proposed, computationally efficient, T-MPSP algorithm can be implemented online and is promising for controlling dynamic systems under the paradigm of fast non-linear MPC.

5.5 U-MPSP 2019

Unscented MPSP was presented in [11], which applies to a class of problems where there are uncertainties in the time-invariant system parameters and/or initial conditions. This technique is a fusion of two recent ideas, namely MPSP and Riemann–Stieltjes optimal control problems. In UMPSP, first an unscented transform is utilized to construct a low-dimensional finite number of deterministic problems. The philosophy of MPSP is utilized next so that the solution can be obtained in a computationally efficient manner. The control solution not only ensures that the terminal constraint is met accurately with respect to the mean value, but it also ensures that the associated covariance matrix (i.e., the error ball) is minimized.

Significance of U-MPSP has been demonstrated in [11] by successfully solving two benchmark problems, namely the Zermelo problem and inverted pendulum problem, which contain parametric and initial condition uncertainties.

6 The MPSP Optimal Control Software Package

The MPSP Optimal Control Software Package builds upon a decade of research into optimal control algorithms for real-time applications. The motivation of the package is to provide researchers, labs, and industry an easy to

use, state-of-the-art Optimal Control problem solver. Anyone with a basic understanding of optimal control problem formulation will be able to use it. The package will include computationally efficient algorithms with mathematically proven convergence properties fit for real-time applications.

The source code of the package is developed generically with APIs provided to set up the problem and solve it. It is also ensured the code is developer-friendly, new functionality can be added to the code, as well as user friendly.

6.1 Python Package

Python has been chosen as the language to implementation of the optimal control package because of its widespread adoption and ease of use. Moreover, this language is extremely popular with academia and the industry as it a high-level language that provides for rapid testing and deployment. The language is also open-source and has a rich collection of scientific libraries.

The overall block diagram of the package is shown in Figure. 2, which is subject to change as new modules are added.

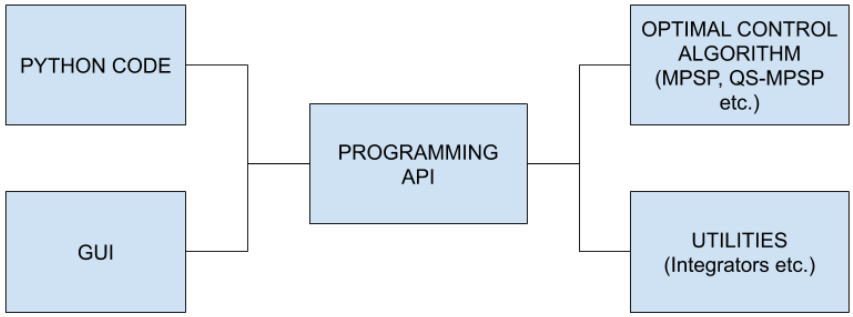

Figure 2: Block diagram of the software package.

6.2 User Friendliness

The aim of the software is collaboration and extensibility, these principles have been the guiding lights for the development. It should be easy for a newcomer to get started with the code and extend it to other algorithms.

The package provides an easy to use front-end API which can be used to define the model of the system, configure the algorithm and start the solution process. The solution process is straight forward as shown in Figure 3.

```
# Configure the model
sys = model.SystemModel(1,2,2, dF_dX, dF_dU, dH_dX)
sys.setStateFlow(F_x)
sys.setOutputFlow(H_x)

# Configure the algorithm
solver = MPSPSolver(sys, 0, 10)
solver.setInitialState(np.array([1,1]))
solver.setFinalOutput(np.array([0,0]))
solver.setInitalControlHistory(np.zeros((1,100)))
solver.setInputWeights(np.identity(1))

# Solve using MPSP
solver.solve()
```

Figure 3: Code sample to use the package.

Salient points:

- It is easy to configure the model and define the dynamics.
- It is easy to configure a particular algorithm and its parameters.
- A simple front-end API for the user, that can be used for coding or to create a GUI.
- Providing verbose and descriptive messages to guide the user.
- The code can be easily extended to include more functionality, using the principles of OOP (Object-oriented programming).
- The aim is to write extensive tests for regression testing.
- Thorough documentation autogenerated with pydoc.
- Version Control using GIT.

References

[1] Radhakant Padhi and Mangal Kothari. Model predictive static programming: a computationally efficient technique for suboptimal control design. *International journal of innovative computing, information and control*, 5(2):399–411, 2009.

[2] Prasiddha Nath Dwivedi, Abhijit Bhattacharya, and Radhakant Padhi. Suboptimal midcourse guidance of interceptors for high-speed targets with alignment angle constraint. *Journal of Guidance, Control, and Dynamics*, 34(3):860–877, 2011.

[3] Sabyasachi Mondal and Radhakant Padhi. Angle-constrained terminal guidance using quasi-spectral model predictive static programming. *Journal of Guidance, Control, and Dynamics*, 41(3):783–791, 2018.

[4] Mangal Kothari and Radhakant Padhi. A nonlinear suboptimal robust guidance scheme for long range flight vehicles with solid motors. *Automatic Control in Aerospace*, 3(1), 2010.

[5] Arnab Maity, Radhakant Padhi, Sanjeev Mallaram, G Mallikarjuna Rao, and M Manickavasagam. A robust and high precision optimal explicit guidance scheme for solid motor propelled launch vehicles with thrust and drag uncertainty. *International Journal of Systems Science*, 47(13):3078–3097, 2016.

[6] Charu Chawla, Pranjit Sarmah, and Radhakant Padhi. Suboptimal reentry guidance of a reusable launch vehicle using pitch plane maneuver. *Aerospace Science and Technology*, 14(6):377–386, 2010.

[7] Omkar Halbe, Ramsingh G Raja, and Radhakant Padhi. Robust reentry guidance of a reusable launch vehicle using model predictive static programming. *Journal of Guidance, Control, and Dynamics*, 37(1):134–148, 2014.

[8] Harshal B Oza and Radhakant Padhi. Impact-angle-constrained suboptimal model predictive static programming guidance of air-to-ground missiles. *Journal of Guidance, Control, and Dynamics*, 35(1):153–164, 2012.

[9] Kapil Sachan and Radhakant Padhi. Waypoint constrained multi-phase optimal guidance of spacecraft for soft lunar landing. *Unmanned Systems*, 07(02):83–104, April 2019.

[10] Prem Kumar, B Bhavya Anoohya, and Radhakant Padhi. Model predictive static programming for optimal command tracking: A fast model predictive control paradigm. *Journal of Dynamic Systems, Measurement, and Control*, 141(2), 2019.

[11] S Mathavaraj and Radhakant Padhi. Unscented mpsp for optimal control of a class of uncertain nonlinear dynamic systems. *Journal of Dynamic Systems, Measurement, and Control*, 141(6), 2019.

[12] Arnab Maity, Harshal B Oza, and Radhakant Padhi. Generalized model predictive static programming and angle-constrained guidance of air-to-ground missiles. *Journal of Guidance, Control, and Dynamics*, 37(6):1897–1913, 2014.

List of Figures