Experimental Application of Stochastic Approximated MPC

1st Michaela Horváthová
Institute of Information Engineering, Automation, and Mathematics, Slovak University of Technology in Bratislava
Slovakia
michaela.horvathova@stuba.sk

2nd Patrik Valábek
Institute of Information Engineering, Automation, and Mathematics, Slovak University of Technology in Bratislava
Slovakia
patrik.valabek@stuba.sk

3rd Karol Kiš
Institute of Information Engineering, Automation, and Mathematics, Slovak University of Technology in Bratislava
Slovakia
karol.kis@stuba.sk

4th Lenka Galčíková
Institute of Information Engineering, Automation, and Mathematics, Slovak University of Technology in Bratislava
Slovakia
lenka.galcikova@stuba.sk

***Abstract*—This paper aims to experimentally validate the stochastic approximated model predictive control (MPC) called the random shooting (RS)-based control. This method was designed to address the computational challenges of standard MPC, mainly the computational complexity. However, RS-based approximated MPC in its original formulation lacks guarantees of recursive feasibility and closed-loop stability. Therefore, a modification of the RS-based approach, a recursive RS-based method, was also implemented to overcome these limitations. The recursive RS-based method features a dual-mode control strategy, which reduces the computational burden. The recursive feasibility and closed-loop stability are enforced through the integration of a support controller. The case study demonstrates a successful implementation of offset-free reference tracking by RS-based methods on a system with fast dynamics. The validated control methods were implemented under strict computational constraints, proving to be a fast, solver-free alternative that does not compromise stability or recursive feasibility.**

***Index Terms*—Model Predictive control, Stochastic Control, Random Sequences, Process Control, Computational Complexity**

I. Introduction

Model Predictive Control (MPC) represents the standard control method for advanced process control in many areas of the industry [1]. MPC is an optimization-based control method that predicts future system behavior and computes optimal control inputs while enforcing constraint satisfaction in a receding horizon fashion.

However, the high computational demand associated with solving an optimization problem of MPC at each control step remains a challenge for real-time applications, particularly in embedded systems with limited computational power and memory. The optimal control law can be determined for all possible initial conditions by using explicit MPC [2]. The primary limitation of this method is the significant memory required to store the explicit solution, and the point-location problem can be computationally demanding for high-dimensional systems. Practical challenges in hardware synthesis of explicit MPC controllers are discussed in [3].

To address these challenges of implicit and explicit MPC, various methods were developed to approximate the optimal feedback control law. In some works [4, 5, 6], neural networks were used to design an explicit linear feedback law. However, these approaches are suboptimal and lack stability guarantees. Additionally, generating training data requires solving a large set of optimal control problems.

Together with neural networks, move-blocking MPC and reduced horizon MPC were designed to decrease the computational complexity of conventional MPC. Move blocking is a technique that keeps control inputs constant or otherwise predefined over fixed intervals, which decreases the number of decision variables. This speeds up the optimization of control inputs but may introduce suboptimality if blocking intervals are too long [7]. Reduced or adaptive horizon MPC shortens the prediction horizon to decrease computational demands while maintaining reasonable performance. However, it may lead to instability in some cases [8].

Stochastic methods, which approximate MPC, have been used in [9, 10, 11]. The RS-based framework for robust

The authors gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grants 1/0297/22 and 1/0239/24. M. Horváthová acknowldges the contribution of the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00636. P. Valábek, K. Kiš are supported by the Slovak Research and Development Agency under the project APVV-21-0019, and the European Union's Horizon Europe under grant no. 101079342 (Fostering Opportunities Towards Slovak Excellence in Advanced Control for Smart Industries). P. Valábek also acknowledges the STU Grant Scheme for Support of Young Researchers.

controller design was analyzed in [9]. A detailed tutorial on randomized algorithms for robust control synthesis was presented in [10]. In contrast, [11] introduced a generalized approach to solving convex optimization problems with randomized algorithms, such as random shooting (RS), achieving near-optimal solutions. The application of RS-based methods in nonlinear chemical reactor control [12] and agile maneuvers of a pendulum on a cart [13] validate their efficiency in handling complex systems and systems with fast dynamics. Although effective and library-free, RS-based methods do not guarantee the stability or feasibility of the control, limiting their reliability in safety-critical applications.

To overcome this limitation, recursive RS-based approximation of MPC [14] introduced a support controller, which ensures feasibility under all operating conditions by providing a solution when no feasible random shoot is found. Moreover, a dual-mode control strategy was incorporated, where a stabilizing Linear Quadratic Regulator (LQR) is used within a terminal invariant set. The dual mode control enhances the approach's real-time computational effectivity.

This paper presents the first experimental validation of recursive RS-based MPC applied to a real-world system with fast dynamics. The application is focused on reference tracking and is compared to the standard MPC approach, explicit MPC, and standard RS-based approximation of MPC in terms of computational time, suboptimality, and memory footprint.

II. Preliminaries

MPC is an optimization-based method that predicts the behavior of a controlled system using its model. Its main benefits include effective handling of constraints and multivariable systems and real-time responsiveness to feedback, i.e., receding horizon approach. MPC is considered as the baseline approach in this work in the following formulation:

$$\min_{X,U} \ \ell_{\mathrm{N}}(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k) \tag{1a}$$

$$\text{s.t.: } x_{k+1} = A\,x_k + B\,u_k, \tag{1b}$$

$$u_k \in \mathcal{U}, \tag{1c}$$

$$x_k \in \mathcal{X}, \tag{1d}$$

$$x_N \in \mathcal{T}, \tag{1e}$$

$$x(0) = x_0, \tag{1f}$$

$$k = 0, 1, \ldots, N-1, \tag{1g}$$

where the model of the system is defined by the state matrix A and input matrix B. The controlled system states x_k and control inputs u_k are subject to constraints defined by the compact and convex sets $\mathcal{X}$ and $\mathcal{U}$, respectively. The optimization problem involves the decision variables X and U, where $X = [x_1^\top, x_2^\top, \ldots, x_N^\top]^\top$ is the sequence of future states, and $U = [u_0^\top, u_1^\top, \ldots, u_{N-1}^\top]^\top$ represents the corresponding control inputs.

MPC only implements the first control action $u(0)$ from the solution of the optimization problem at the current time step based on the current state measurement $x(0)$. During the next time step, MPC recomputes the control sequence over a prediction horizon of length N based on the recent state measurement.

Throughout the paper, we assume that the cost function uses quadratic terms to penalize the terminal state x_N and the stage cost for states and inputs. Specifically, the terminal penalty $\ell_N(x_N)$ is defined as $x_N^\top P x_N$, where $P \succ 0$ is the Lyapunov matrix. The stage cost $\ell(x_k, u_k)$ is given by $x_k^\top Q x_k + u_k^\top R u_k$, where $Q \succeq 0$ and $R \succ 0$ are the weighting matrices. The terminal set $\mathcal{T} \subset \mathcal{X}$, constructed considering the well-known stabilizing LQR controller, is meant to contain the final state x_N to ensure stability.

III. Random shooting-based Approximated MPC

As discussed in the Introduction section, the MPC framework still poses challenges for embedded hardware due to its high computational and memory demands; these challenges call for alternative approaches. One of them is random shooting-based approximated MPC, which relies on the stochastic random generation of control sequences, i.e., random shoots. These sequences are inquired for feasibility (constraint satisfaction) and performance, with the best-performing feasible sequence $\widetilde{U} = \{\widetilde{u}_0, \widetilde{u}_1, \ldots, \widetilde{u}_{N-1}\}$ selected at each time step. While suboptimal, the chosen sequence ensures feasibility and safe operation. Suboptimality decreases as the number of generated sequences increases, and for convex control problems, both runtime and suboptimality can be estimated in advance [11]. Additionally, this method can handle scenarios with discontinuous cost functions and/or nonconvex constraints. In this work, we adopt the formulation of the method presented in [12]. Notably, the method is library-independent, with very low computational demands, and allows users to tune the suboptimality of the solution based on their preferences.

Algorithm 1 describes the basic principles of the random shooting-based approximated MPC. Key variables include a counter for each random selection of the control sequence–random shoot (i), feasible shoots (r), a feasibility flag (δ), and the maximum number of feasible shoots (N_{F}).

A limitation of Algorithm 1 is that it does not guarantee to find N_{F} feasible shoots within the sampling time t_{s}, potentially failing to achieve the desired suboptimality. In the worst case, no feasible solution may be found, leaving no valid control input $\widetilde{u}_0$ available. Additionally, the algorithm does not ensure recursive feasibility, meaning that future control steps might also lack feasible solutions.

IV. Recursive Random Shooting-based Approximated MPC

The goal of recursive random shooting-based approximated MPC proposed in [14] is to combine the advantages of MPC (Section II) and RS-based MPC approximation (Section III). The approach ensures constraint satisfaction, recursive feasibility, and asymptotic stability similar to MPC, while offering

Algorithm 1 RS-based approximation of MPC [12].

Require: initial conditions $x(0)$, N_{F} , maximal number of random shoots $N_{\max}$, $A, B, N, \mathcal{U}, \mathcal{X}, \mathcal{T}, \ell(x,u), \ell_{\mathrm{N}}(x_n)$

Ensure: sequence of control inputs $\{\widetilde{u}_0, \widetilde{u}_1, \ldots, \widetilde{u}_{N-1}\}$

$\widetilde{J} \leftarrow \infty$; $\widetilde{U} \leftarrow \varnothing$; $r \leftarrow 0$, $i \leftarrow 0$ `// Initialize`
while $r < N_{\mathrm{F}}$ **and** $i < N_{\max}$ **do**
  $\hat{x}_0 \leftarrow x_0$, $\hat{J}_r \leftarrow 0$, $\delta \leftarrow 1$, $i \leftarrow i+1$ `// Reset`
  for $j = 0, 1, \ldots, N-1$ **do**
    if $\hat{x}_j \in \mathcal{X}$ **then**
      $\hat{u}_j \leftarrow \mathrm{random}\{\hat{u} : \hat{u} \in \mathcal{U}\}$ `// Random shoot`
      $\hat{J}_r \leftarrow \hat{J}_r + \ell(\hat{x}_j, \hat{u}_j)$ `// Penalty update`
      $\hat{x}_{j+1} \leftarrow A\hat{x}_j + B\hat{u}_j$ `// System prediction`
    else
      $\delta \leftarrow 0$ `// Infeasible`
      break
  if $\delta == 1$ **and** $x_N \in \mathcal{T}$ **then**
    $\hat{J}_r \leftarrow \hat{J}_r + \ell_{\mathrm{N}}(x_N)$ `// Terminal penalty update`
    $r \leftarrow r+1$ `// Update feasibility counter`
    if $\hat{J}_r < \widetilde{J}$ **then**
      $\widetilde{U} \leftarrow \{\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_{N-1}\}$ `// Update best approximated control action`
      $\widetilde{J} \leftarrow \hat{J}_r$ `// Update cost`
return $\{\widetilde{u}_0, \widetilde{u}_1, \ldots, \widetilde{u}_{N-1}\} \leftarrow \widetilde{U}$

a library-free, easy-to-implement solution for close-to-optimal control on embedded platforms.

To achieve this, two modifications to the method described in Algorithm 1 were made. Firstly, the dual-mode control approach, increases the computational efficiency of the approach, as described in [15]. It consists of two modes: (i) Mode I: LQ optimal control when system states $x_k \in \mathcal{T}$, and (ii) Mode II: random shooting control input when the states are outside the terminal set $\mathcal{T}$. Compared to conventional MPC, the dual-mode control reduces the average real-time effort for evaluating the control input, as the computation simplifies to a matrix multiplication in (2) once the system states enter the terminal set $\mathcal{T}$. The terminal set controller F_{LQ} corresponding to the terminal set $\mathcal{T}$, computes the optimal control action as follows:

$$u(k) = F_{\mathrm{LQ}}\, x(k). \tag{2}$$

The controller gain F_{LQ} is obtained as the solution to the discrete algebraic Riccati equation for a linear time-invariant (LTI) system, as discussed in [16]. It guarantees the asymptotic stability of the controlled LTI system for $\forall x \in \mathcal{T}$. This improves the resource efficiency of the controller, enhancing battery life and optimizing hardware performance.

The second modification is the presence of a support controller $\bar{F}$ that guarantees recursive feasibility and closed-loop stability, as proven in [17] and [14]. The key advantage of using the support controller in the RS-based MPC approximation is that it initializes the evaluation of an asymptotically stable controller while its performance improves continuously through the recursive method described in Algorithm 1. Unlike the method described in Algorithm 1, where the best solution from the previous step is discarded, the recursive RS approach reuses the best solution found in the last step for the current one. If no better solution than the one coming from the support controller is found, the previous best solution is applied in a receding horizon control fashion, as discussed in [18].

The support controller $\bar{F}$ is designed to ensure feasibility under a conservative set of physical constraints, which may lead to suboptimal control performance. However, it serves as a backup control input when no better solution is found. In Algorithm 2, $\hat{u}, \hat{x}, \hat{J}_r$ represent variables for an individual random shoot, while $\widetilde{u}, \widetilde{J}$ correspond to the approximated solution of the MPC problem in (1). The parameters i, r, δ, N_{F} represent the same as in Algorithm 1.

In contrast to Algorithm 1, Algorithm 2 introduces two auxiliary inputs, $\bar{U}$ and $\bar{J}$, which are initialized before executing the closed-loop control and contribute to the recursive nature of the random shooting-based evaluation. The sequence of support control inputs, $\bar{U} = \{\bar{u}_0, \bar{u}_1, \ldots, \bar{u}_{N-1}\}$, is initialized using the state-feedback control law defined as $\bar{u}(k) = \bar{F}\bar{x}(k)$. The support penalty $\bar{J}(\bar{x}, \bar{u})$ is computed as

$$\bar{J}(\bar{x}, \bar{u}) = \sum_{k=0}^{N-1} \ell(\bar{x}_k, \bar{u}_k) + \ell_{\mathrm{N}}(\bar{x}_N),$$

where $\bar{u}_k$ and $\bar{x}_k$, for all $k = 0, 1, \ldots, N-1$, represent the support control inputs and the corresponding support system state trajectory, respectively. Algorithm 2 also evaluates $(N_{\mathrm{F}} - 1)$ random shoots, effectively skipping the evaluation of one random shoot. This modification comes from the availability of a support sequence of control inputs $\bar{U}$, which guarantees that $\widetilde{U}$ ensures the constraints satisfaction, maintains recursive feasibility, and asymptotic stability.

V. Reference Tracking

For the successful implementation of the offset-free control using strategies described in this paper, an integral action was incorporated into the model of the system. This is done by augmenting the state vector of the system as follows:

$$\hat{x}(k) = \begin{bmatrix} x(k) \\ \sum_{j=0}^{k} e(j) \end{bmatrix}, \tag{3}$$

where $\sum_{j=0}^{k} e(j) = \sum_{j=0}^{k} (y_{\mathrm{ref}}(j) - y(j))$ represents the integral (sum) of the control error at a given discrete time step k, and $y_{\mathrm{ref}} \in \mathbb{R}^{n_y}$ denotes the reference. The augmented system, defined with respect to the original state, input and output matrices A, B, C and the sampling time t_{s} is represented as:

$$\hat{A} = \begin{bmatrix} A & 0 \\ -t_{\mathrm{s}}C & I \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}, \quad \hat{C} = \begin{bmatrix} C & 0 \end{bmatrix}. \tag{4}$$

This method of state augmentation is a standard practice for achieving offset-free reference tracking.

Algorithm 2 Recursive RS-based approximation of MPC problem [14].

Require: initial conditions $x(0)$, N_{F}, $N_{\max}$, $A, B, N, \mathcal{U}, \mathcal{X}, \mathcal{T}, \ell(x,u), \ell_{\mathrm{N}}(x_n)$, terminal controller F_{LQ}, support controller $\bar{F}$, initialization of support control inputs $\bar{U}$

Ensure: sequence of control inputs $\{\widetilde{u}_0, \widetilde{u}_1, \ldots, \widetilde{u}_{N-1}\}$, support control inputs $\bar{U}$, support penalty $\bar{J}$

$r \leftarrow 0$, $i \leftarrow 0$ // Initialize
$\widetilde{U} \leftarrow \bar{U}$, $\bar{u}_N \leftarrow \varnothing$ // Initialize recursive RS
for $i = 0, 1, \ldots, N-1$ **do**
  $\bar{J} \leftarrow \bar{J} + \ell(\bar{x}_i, \widetilde{u}_i)$ // Penalty update
  $\bar{x}_{i+1} \leftarrow A\,\bar{x}_i + B\,\widetilde{u}_i$ // System prediction
$\bar{J} \leftarrow \bar{J} + \ell_{\mathrm{N}}(\bar{x}_N)$ // Terminal penalty update
$\widetilde{J} \leftarrow \bar{J}$
while $r < (N_{\mathrm{F}} - 1)$ **and** $i < N_{\max}$ **do**
  $\hat{x}_0 \leftarrow x_0$, $\hat{J}_r \leftarrow 0$, $\delta \leftarrow 1$, $i \leftarrow i+1$ // Reset
  for $j = 0, 1, \ldots, N-1$ **do**
    if $\hat{x}_j \in \mathcal{T}$ **then**
      $\hat{u}_j \leftarrow F_{\mathrm{LQ}}\,\hat{x}_j$ // LQ optimal control input
    else
      $\hat{u}_j \leftarrow \mathrm{random}\{\hat{u} : \hat{u} \in \mathcal{U}\}$ // Random shoot
    if $\hat{x}_j \in \mathcal{X}$ **then**
      $\hat{J}_r \leftarrow \hat{J}_r + \ell(\hat{x}_j, \hat{u}_j)$ // Penalty update
      $\hat{x}_{j+1} \leftarrow A\,\hat{x}_j + B\,\hat{u}_j$ // System state prediction
    else
      $\delta \leftarrow 0$ // Infeasible
      break
  if $\delta == 1$ **and** $\hat{x}_N \in \mathcal{T}$ **then**
    $\hat{J}_r \leftarrow \hat{J}_r + \ell_{\mathrm{N}}(\hat{x}_N)$ // Terminal penalty update
    $r \leftarrow r + 1$ // Update feasibility counter
    if $\hat{J}_r < \widetilde{J}$ **then**
      $\widetilde{U} \leftarrow \{\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_{N-1}\}$ // Update best approximated solution
      $\widetilde{J} \leftarrow \hat{J}_r$ // Update reference penalty
      $\bar{u}_N \leftarrow \bar{F}\,\widetilde{x}_N$ // Update support control
      $\bar{x}_{N+1} \leftarrow A\,\widetilde{x}_N + B\,\bar{u}_N, \quad \bar{\ell}_N(\bar{x}_{N+1})$ // Update terminal penalty
if $\bar{u}_N == \varnothing$ **then**
  $\bar{u}_N \leftarrow \bar{F}\bar{x}_N$ // Update recursive support control
return $\{\widetilde{u}_0, \widetilde{u}_1, \ldots, \widetilde{u}_{N-1}\} \leftarrow \widetilde{U}$,
$\bar{U} \leftarrow \{\widetilde{u}_1, \widetilde{u}_2, \ldots, \bar{u}_N\}$, $\bar{J}$

VI. Controlled Plant

The Flexy device is a single-input, single-output (SISO) system and it is characterized by fast dynamics [19]. Specifically, the sampling time of the device was set to $t_{\mathrm{s}} = 0.1$ s. The device consists of a fan with adjustable rotation speed acting as an actuator, which allows to produce an air flow. The air flow is exerted towards a flexible sensor that detects a bending (controlled output) caused by the airflow. The bending is detected through changes in electrical resistance and it is scaled to a normalized range of $[0, 100]\%$. The goal of the controller is the reference tracking to maintain the sensor bend at a desired level.

Figure 1. Controlled Plant Flexy [19].

The system's response is asymmetric as it changes when performing a negative or positive step change. To address this, step changes in both positive and negative directions were identified using the Strejc method for step-response identification [20]. The obtained mathematical model was converted into a discrete state-space form. To successfully remove the control offset, the integral action was added to the model, increasing the dimensionality of the states as described in (4). The augmented state, input, and output matrices are as follows:

$$\hat{A} = \begin{bmatrix} 0.5394 & 0 \\ -0.5464 & 1 \end{bmatrix}, \hat{B} = \begin{bmatrix} 0.0763 \\ 0 \end{bmatrix}, \hat{C} = \begin{bmatrix} 4.2317 \\ 0 \end{bmatrix}^{\top}. \quad (5)$$

VII. Control Setup

The experimental results were generated using MATLAB/Simulink R2022b on a PC equipped with an i5 2.7 GHz CPU and 8 GB of RAM. The optimization problems were modeled using the YALMIP toolbox [21], and the optimiation problems were solved using MATLAB's built-in

solver, Quadprog. All control methods were designed using weighting matrices Q and R defined as:

$$Q = \begin{bmatrix} 100 & 0 \\ 0 & 10 \end{bmatrix}, \quad R = 0.01. \tag{6}$$

The control output was defined within the range of $[0, 100]\%$, where the steady-state output of the Flexy device was set to 50%, and the control input was constrained within $[0, 80]\%$, with the steady-state value 40%. Therefore, the input and output constraints in the deviation form were specified as:

$$-50\% \leq y \leq 50\%, \quad -40\% \leq u \leq 40\%. \tag{7}$$

Additionally, the augmented state vector from (3) was constrained to:

$$[-50, -1000]^\top \preceq \hat{x}(k) \preceq [50, 1000]^\top. \tag{8}$$

The recursive RS-based approach considered a required number of feasible shoots $N_{\mathrm{F}} = 500$, with a maximum number of random shoots set to $N_{\max} = 1000$. the same $N_{\max} = 1000$ was also used for the non-recursive RS-based approach. The support controller gain was designed using the method proposed in [22] as follows:

$$\bar{F} = [-0.2103, -0.0291]. \tag{9}$$

The LQ optimal controller gain F_{LQ} in (2) was computed as:

$$F_{\mathrm{LQ}} = [-0.4022, -0.8720]. \tag{10}$$

VIII. Results

This section analyzes the performance of three control strategies. The first approach involves the RS-based approximation of MPC as described in Algorithm 1, detailed in Section III. The second method implements the recursive RS-based approach [14] described in Section IV. The conventional MPC presented in Section II was employed to evaluate the suboptimality level and computational demands.

Figure 2 illustrates the closed-loop control trajectories generated by the implemented methods. As shown in Figure 2, all three strategies successfully ensured offset-free reference tracking for the Flexy device. Algorithm 2 generates control inputs using one of the following methods: (i) random shooting, (ii) the support controller defined, or (iii) the LQ optimal controller described in (2). The dashed blue trajectory in Figure 2 corresponds to the control sequence obtained via Algorithm 2.

The first three control steps of the blue trajectory were computed using the support controller in (10), followed by 17 steps determined through random selection, i.e., shooting. Once the system states entered the terminal set $\mathcal{T}$, the remaining control actions were computed using the LQR controller from (2) designed for the terminal set $\mathcal{T}$. Consequently, these control actions aligned with the optimal actions produced by the conventional MPC method.

As discussed in Section IV, control inputs derived using Algorithm 2 ensure both closed-loop stability and adherence to physical constraints on system inputs and states. In contrast,

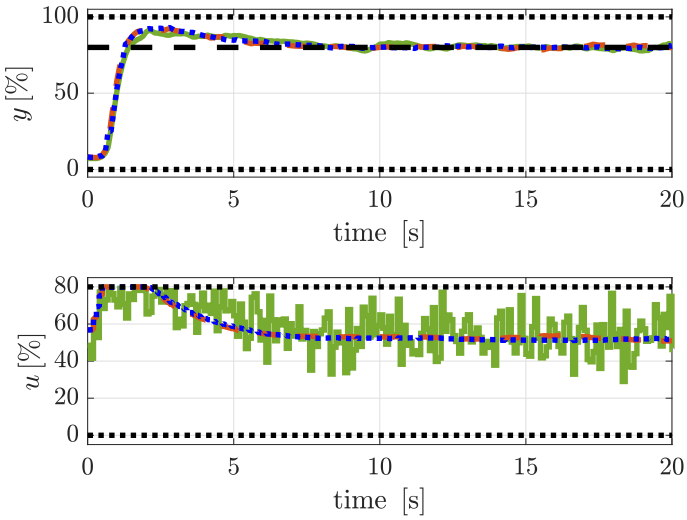

Figure 2. The control performance of reference tracking for the Flexy device using: the RS-based controller (solid green), the recursive RS-based controller (dotted blue), the conventional MPC controller (dashed red). The physical constraints on control input are depicted in black-dotted lines, and the reference is in black dashed lines.

Table I
Comparison of the selected control methods: MPC, Explicit MPC (EMPC), RS, Recursive RS.

Control Method	MPC	EMPC	RS	Recursive RS
t_{avg} [ms]	35.9	1.5	17.7	5.9
J_{total} [$\times 10^7$]	†	2.363	2.446	2.371
Memory footprint [kB]	0.296	7.056	0.312	0.424

control actions generated by Algorithm 1 (solid green) may not consistently maintain closed-loop stability and feasibility.

The numerical comparison of the results is summarized in Table I. For the conventional MPC, the control performance criterion J_{total} is the closed-loop cost J in (1a). For the random shooting-based approaches the criterion $J_{\mathrm{cl},i}$ corresponds to $\widetilde{J}$, as specified in Algorithm 1 and Algorithm 2.

In Table I, the symbol " † " indicates that the corresponding criterion was not applicable due to the fact that the computational time of the first control input using MPC exceeded the sampling time of the Flexy device. Implementation of RS-based and recursive RS-based methods decreased the average computational time t_{avg} by 51 % and 84 %, which was less than the sampling time of the flexy device.

Explicit MPC (EMPC) was also evaluated as a benchmark to analyze the suboptimality of the approximated approaches. When comparing the RS and recursive RS methods, the implementation of the recursive RS approach resulted in a decrease in t_{avg}. The suboptimality level of the closed-loop performance, as evaluated by J_{total}, was approximately 0.4% for the recursive RS-based method, compared to roughly 5% for the RS-based control method. Notably, the suboptimality of this approach can be tuned using the number of feasible shots, allowing a trade-off between control performance and computational time. Although highly efficient, implicit MPC demonstrated a significant increase in memory usage, with

explicit MPC leading to a 96 % increase in the case of RS-based approximation of MPC and 94 % for recursive RS-based approximation of MPC.

This section demonstrated the effectiveness of the RS-based approaches through experiments performed using the Flexy device. The results show that the proposed control strategies can handle systems with fast dynamics, even in the context of the reference tracking problem and under very limited availability of computational resources.

IX. Conclusion

This paper presents an experimental implementation of an RS-based approximation of MPC, focused explicitly on reference tracking. RS-based methods offer a solver-free, stochastic alternative to MPC. However, they lack the recursive feasibility and closed-loop stability guarantees. A recursive RS-based approach was later introduced to ensure recursive feasibility and closed-loop stability while maintaining computational efficiency. Both methods were validated on a system with fast dynamics and compared with implicit and explicit MPC in terms of computation time, suboptimality, and memory usage. Implicit MPC was not usable due to high computational demands, while RS-based methods significantly reduced computation time. Recursive RS-based MPC achieved near-optimal performance with lower memory usage than explicit MPC while maintaining performance and ensuring closed-loop stability and recursive feasibility guarantees. These methods can be particularly suitable for systems with strict computational constraints, such as embedded systems.

Bibliography

[1] Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117(5):1327–1349, 2021.

[2] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming – the explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1974–1985, 2002.

[3] T. A. Johansen, W. Jackson, R. Schreiber, and P. Tondel. Hardware synthesis of explicit model predictive controllers. *IEEE Transactions on Control Systems Technology*, 15(1):191–197, 2007.

[4] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas., and M. Morari. Approximating explicit model predictive control using constrained neural networks. In *2018 Annual American Control Conference (ACC)*, pages 1520–1527, 2018.

[5] Dustin Kenefake, Rahul Kakodkar, Sahithi S. Akundi, Moustafa Ali, and Efstratios N. Pistikopoulos. A multi-parametric approach to accelerating relu neural network based model predictive control. *Control Engineering Practice*, 151:106041, 2024.

[6] Saket Adhau, Sébastien Gros, and Sigurd Skogestad. Reinforcement learning based mpc with neural dynamical models. *European Journal of Control*, 80:101048, 2024.

[7] R. Cagienard, P. Grieder, E.C. Kerrigan, and M. Morari. Move blocking strategies in receding horizon control. *Journal of Process Control*, 17(6):563–570, 2007.

[8] Arthur J Krener. Adaptive horizon model predictive control∗∗this work was supported in part by the afosr. *IFAC-PapersOnLine*, 51(13):31–36, 2018. 2nd IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNON 2018.

[9] M. Vidyasagar. Randomized algorithms for robust controller synthesis using statistical learning theory. *Automatica*, 37(10):1515–1528, 2001.

[10] M. Vidyasagar. Randomized algorithms for robust controller synthesis using statistical learning theory: A tutorial overview. *European Journal of Control*, 7(2):287–310, 2001.

[11] M. Dyer, R. Kannan, and L. Stougie. A simple randomised algorithm for convex optimisation. *Mathematical Programming*, 147:207–229, 2014.

[12] P. Bakaráč and M. Kvasnica. Fast nonlinear model predictive control of a chemical reactor: a random shooting approach. *Acta Chimica Slovaca*, 11(2):175–181, 2018.

[13] K. Fedorová, P. Bakaráč, and M. Kvasnica. Agile manoeuvres using model predictive control. *Acta Chimica Slovaca*, 12(1):136–141, 2019.

[14] P. Bakaráč, M. Horváthová, L. Galčíková, J. Oravec, and M. Bakošová. Approximated MPC for embedded hardware: Recursive random shooting approach. *Computers & Chemical Engineering*, 165, 2022.

[15] M. S. Darup and M. Mönnigmann. Optimization-free robust MPC around the terminal region. *Automatica*, 95:229–235, 2018.

[16] F. Borrelli. *Constrained Optimal Control of Linear and Hybrid Systems*. Springer Berlin Heidelberg, 2017.

[17] J. Oravec, M. Kvasnica, and M. Bakošová. Quasi-nonsymmetric input and output constraints in LMI-based robust MPC. In *Preprints of the 20th IFAC World Congress, Toulouse, France*, volume 20, pages 11829–11834, 2017.

[18] J. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, London, 2000.

[19] Ľ. Čirka, M. Kalúz, D. Dzurková, and R. Valo. Educational device Flexy2 in the teaching of experimental identication. In M. Fikar and M. Kvasnica, editors, *Proceedings of the 22nd International Conference on Process Control*, pages 239–244, Štrbské Pleso, Slovakia, June 11-14, 2019 2019.

[20] J. Mikleš and M. Fikar. *Process Modelling, Identification, and Control*. Springer Verlag, Berlin Heidelberg, 2007.

[21] J. Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[22] M. V. Kothare, V. Balakrishnan, and M. Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32:1361–1379, 1996.