BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 70(4), 2022, Article number: e140696
DOI: 10.24425/bpasts.2022.140696

Computationally efficient nonlinear model predictive controller using parallel particle swarm optimization

Supriya P. DIWAN[1] **and Shraddha S. DESHPANDE**[2]*

[1] Government College of Engineering, Karad-415124, Maharashtra, India
[2] Walchand College of Engineering, Sangli-416415, Maharashtra, India

Abstract. As nonlinear optimization techniques are computationally expensive, their usage in the real-time era is constrained. So this is the main challenge for researchers to develop a fast algorithm that is used in real-time computations. This work proposes a fast nonlinear model predictive control approach based on particle swarm optimization for nonlinear optimization with constraints. The suggested algorithm divide and conquer technique improves computing speed and disturbance rejection capability, demonstrating its suitability for real-time applications. The performance of this approach under constraints is validated using a highly nonlinear fast and dynamic real-time inverted pendulum system. The solution presented through work is computationally feasible for smaller sampling times and it gives promising results compared to the state of art PSO algorithm.

Key words: nonlinear model predictive control; particle swarm optimization; fast dynamic systems; rotary inverted pendulum; divide and conquer approach.

1. INTRODUCTION

Model predictive control (MPC), has become one of the first choice of control strategy in all chemical plant operations as well as refineries, because it is able to manage multivariable systems with constraints. Many processes in industry sectors like paper and pulp, mining as well as food are nonlinear, and hence models that are linear are inadequate to describe the process dynamics over the required range of operating points. Therefore, different nonlinear model predictive control (NMPC) techniques have been developed [1–3]. For real-time fast dynamic applications in the area of automotive manufacturing, aerodynamics, robotics, cryogenics, etc. a fast sampling rate is required. It is very difficult to compute online control action as the optimization problem is required to be solved at every instant of sampling. It is also possible, but only for the class of nonlinear models, to use successive linearization, that provides an exact linear model, and then use MPC algorithms [4,5]. However, a change of variables in the feedback may provide problems with nonlinear constraints. Various efforts have been made towards the development of efficient optimization techniques and reviewed by [6]. Methods for improving computational speed have been discussed by adjusting nonlinear programming algorithms to adapt to the framework of online optimization, using the optimal control formulation of the receding horizon problem, constraint and cost approximations based on state-space partitioning, and re-parameterization of the degree of freedom in predictions.

Although some online NMPC solution techniques that are very fast have been developed, they depend on iterative, approximate solution approaches based on rapid convex optimization solvers [7, 8]. As a result, they do not necessarily attain the solution speeds obtained by linear formulations of MPC. Some authors have developed explicit NMPC which moves the optimization offline, but it is known to scale poorly as the look up table that results goes up exponentially in keeping with the length of the horizon as well as the number of constraints. Due to this, NMPC, especially on systems with strict computational constraints, is not suitable for fast, real-time operation. Usually sequential quadratic programming (SQP) algorithm or its modified versions prove to be effective as a nonlinear, non-convex optimization problem solver in the NMPC [9–12]. But it greatly depends on the strong initial guesses for the starting of computations which may lead to the local minimum solution and the resulting NMPC control may be far from an optimal point.

To adapt to these issues, heuristic algorithms can try not to get trapped in the local minima and benefit from its independence on the initial guesses. The primary motivation for selecting the PSO as a heuristic algorithm is the fact that its ease of implementation and shown good performance in many applications based on the literature. Though PSO does not guarantee an exact optimum but, because of the availability of a diversity space searching technique in the algorithm, it has a great chance to reach towards the global optimum. Various efforts have been taken by researchers to apply the PSO to the NMPC strategy for its implementation [13–15] for fast dynamic systems but still there is scope to improve the PSO algorithm to tackle with the large horizon length.

In the light of current approaches and their limitations, the NMPC with modified parallel algorithm of PSO based on

*e-mail: supriya.diwan8@gmail.com

Manuscript submitted 2021-04-23, revised 2021-10-31, initially accepted for publication 2021-12-13, published in August 2022.

divide-and-conquer approach (PPSO-DAC) is proposed for encapsulating the required control input, which applied to constrained NMPC for executing within prescribed sampling time. In this approach batch wise division of the population is used to compute the fitness function in parallel to conquer the high speed computation in the NMPC. The key contributions of the work are:

- a comparison of proposed strategy PPSO-DAC with PSO in NMPC algorithm;
- the proposition of efficient constrained PPSO NMPC algorithm with divide-and-conquer approach, tailored for implementation using fast hardware with on-line optimization;
- validation with SRIP case study, which is the classical example of the system with fast dynamics that plays a important role as prototype of several robotic applications.

The contents of the following sections of the paper are as follows. Section 2 gives a brief review of the NMPC formulation. The proposed PPSO-DAC algorithm used to solve the NMPC optimization problem is introduced in Section 3. Section 4 gives implementation details of the proposed algorithm which is followed by experimentation results explained in Section 5 for single rotary inverted pendulum (SRIP) as a case study, and the conclusion is presented in Section 6.

2. NMPC FORMULATION

In general, model predictive control recurrently calculates control actions which optimize the predicted system behaviour. The system to be controlled is considered as a nonlinear state-space model (1) with constraints (2) as follows:

$$x(t+1) = f(x(t),u(t)); \quad t>0 \text{ at } t_0,\ x(0), \tag{1a}$$

$$y(t) = h(x(t),u(t)). \tag{1b}$$

Depending on the constraints imposed by input and output in the form:

$$u_{\min} \le u(t) \le u_{\max}, \tag{2a}$$

$$y_{\min} \le y(t) \le y_{\max}, \tag{2b}$$

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

where $x(t) \in R^{(n_x)}$ is the state vector, $u(t) \le R^{(n_u)}$ is the input vector, $y(t) \le R^{(n_c)}$ denotes the controlled output with the t as the current sampling instant. f and h are system functions of the process model. Furthermore, $u_{\min}$, $u_{\max}$ and $y_{\min}$, $y_{\max}$ are constant vectors. The working principle of the NMPC is described in Fig. 1. At sample t a dynamic model of the controlled system is utilized to forecast a series of N_p future output behaviours of the system up to time $t+N_p$, i.e., $y(t+N_p \mid t)$ for $N_p = 1,2,\ldots,N_p$. Based on the forecast, N_m optimal future inputs $u(t+N_c \mid t)$ for $N_c = 0,1,\ldots,N_c-1$ are calculated to reach the desired output y_{ref}, as closely as possible as shown in Fig. 1. The parameters N_p and N_c are the prediction and control horizons respectively. The problem of optimization (3) for NMPC can be explained as below on the basis of the dynamic model of

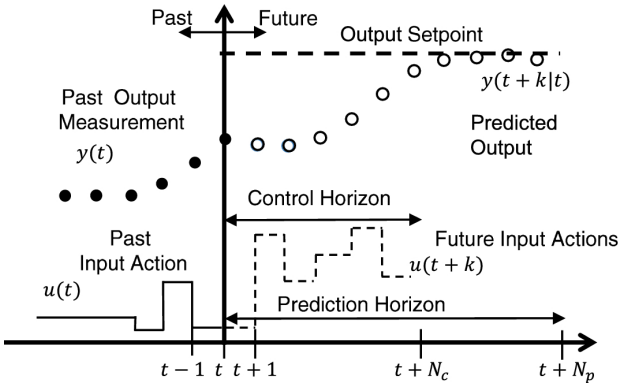


Fig. 1. A graphical representation of NMPC

form (1):

$$\min_{U_k} J(y(t),u(t)). \tag{3}$$

The criterion for minimizing the calculation of optimal movements is usually the quadratic function of the difference between the expected output signal and the desired reference output. This cost function J includes the control moves $(u(t+N_m \mid t))$ to minimize the control efforts. A cost function is as given below:

$$J = \sum_{p=1}^{N_P} \|\bar{y}(t+p \mid t) - y_{\text{ref}}(t+p \mid t)\|_Q^2 + \sum_{m=0}^{N_c-1} \|\Delta u(t+m \mid t)\|_R^2, \tag{4}$$

where Q and R, are weighting matrices. Here, $\|.\|$ is the vector 2-norm, $|.|$ is the absolute value of the vector, and $\Delta u(t+m \mid t) = u(t+m \mid t) - u(t+m-1 \mid t)$. Usually, only the first N_c control inputs are calculated, and the following $(N_p - N_c)$ control inputs are assumed to be zero.

Only the first control input is used in the N_c control input calculated from the J minimization; the rest are discarded. The performance is calculated at the next sampling moment and the process is repeated with the new measured values and by moving the horizon of control and prediction ahead. The estimation of the optimal control inputs for the future is based on the J that can be accomplished with various optimization algorithms. In this research paper, what we propose is a parallel PSO method on the basis of the divide and conquer approach which is discussed in the next section.

3. PPSO-DAC ALGORITHM

In this section the proposed PPSO-DAC is explain with focusing the classic PSO to the proposed PPSO-DAC.

3.1. PSO

PSO makes use of the population of possible solutions to probe the search space and relies on data that is being exchanged between individuals (particles) that make up the population otherwise known as the swarm [16–18]. Each particle modifies its

trajectory based on the information exchange, going towards the best position held by it previously, as well as the best position held previously that is achieved by the total swarm. These updated, improved positions will guide the movement of the swarm. This process will be re-run till a satisfactory solution is reached. The movement of the particles is updated based on the particle velocity and its position, which is mentioned in (5) and (6).

$$P_i(q+1) = P_i(q) + V_i(q+1), \tag{5}$$

$$V_i(q+1) = \omega V_i(q) + c_1 r_1 \left(P_i^L(q) - P_i(q)\right) + c_2 r_2 \left(P_i^G(q) - P_i(q)\right), \tag{6}$$

where, $P_i(q) = (P_{i1}(q), P_{i2}(q), \ldots, P_{iN}(q))$ is the position of the particle i with $i = 1,2,\ldots,S_P$, population size S_P, $V_i(q) = (V_{i1}(q), V_{i2}(q), \ldots, V_{iN}(q))$ is the velocity of the particle i, N is the dimension of the search space, $P_i^L = (P_{i1}^L, P_{i2}^L, \ldots, P_{iN}^L)$ is the local best within the particle, $P_i^G = (P_{i1}^G, P_{i2}^G, \ldots, P_{iN}^G)$ is the global best within the particle of the swarm. q is nothing but $= 1,2,\ldots,n_i$, where n_i is the number of iterations. c_1 and c_2 are constant positive values called as learning factors, r_1 and r_2 are random numbers which uniformly distributed within the range [0, 1] at each iteration, ω is the inertia weight which is decreased with the range $\omega_{\max}$ to $\omega_{\min}$ and each iteration as $\omega = (\omega_{\max} - \omega_{\min})/n_i$. Block diagram shown in Fig. 2 gives the details regarding the PSO based NMPC.

The selection of the parameters for the designing of PSO algorithm is crucial. Based on these parameters the convergence speed and the performance of the PSO algorithm is necessary. In this paper the parameters are selected empirically. The pseudo-code of the PSO based NMPC algorithm is given as Algorithm 1. PSO relies on a population (or swarm) of particles, which change positions and velocites while searching the space for global minimum of the cost function, J (4 or 8). When PSO is applied for NMPC, the particle position is a vector that corresponds to n_u inputs and N_c steps. Hence, the initial position is $P_i(0)$ of $n_u N_c$ elements. The entries of vector $P_i(0)$ for i-th particle are calculated as random numbers using uniform distribution between $u_{\max}$ and $u_{\min}$ (which is positions $p_{\max}$, $p_{\min}$), which is referred to in line 3 of algorithm 1. In the algorithm, P^G is the best position visited by the whole population as mentioned above, with cost function value F^*; Pi^L is the best position visited by the i-th particle, with cost function value F_i^L. Searching for the minimum of J proceeds through succeeding $j = 1,\ldots,t$ iterations. The choice of the ω, c_1, c_2, t and n_i, as well as prediction control parameters (i.e. Q, R, N_p, N_c) influence the quality of control algorithm and the time of optimization. Considering that the sequence $U(k)$ provided by Algorithm 1 is $U(k) = \{u_1, u_2, \ldots, u_{N_c}\}$. Within these sequence first optimal control input u_1^* calculated by the PSO is finally applied to the plant and others are discarded.

Constraints of the input signal can be handled by penalty function which is given by (8). The problem of state constraints is always more challenging. Constraints of any kind can be included in optimization process, resulting in minimisation problem of cost function subject to constraints.

Remark 1. The working mechanism of Algorithm 1 resembles the working principle of the PSO based NMPC algorithm described in the paper [19] but the only difference is the method of fitness function calculation which is described by (8).

As PSO carrying the unconstrained optimization nature, the penalty function approach is used to tackle the constraints in

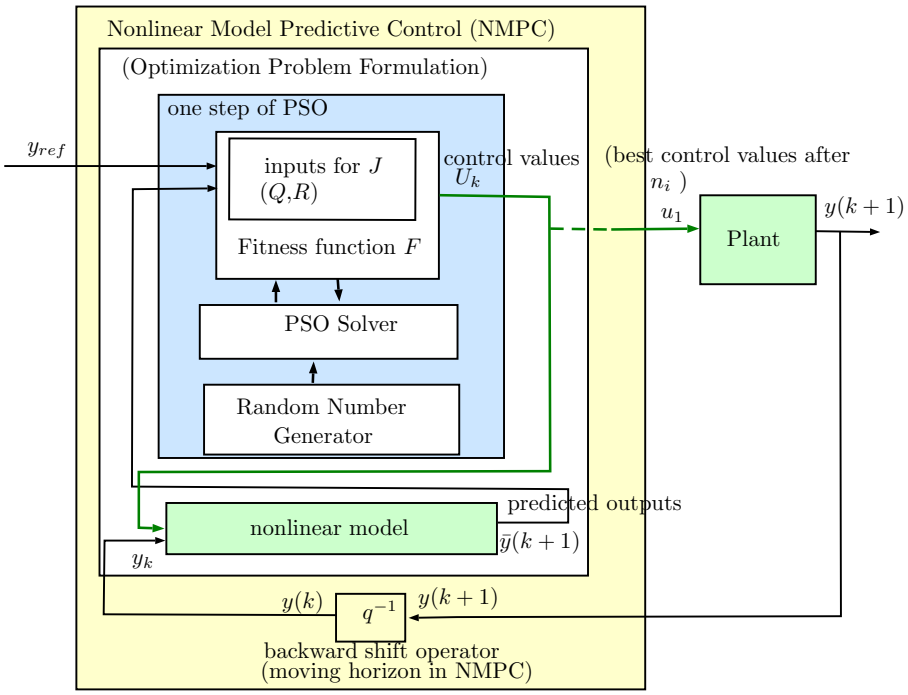


Fig. 2. Block diagram of PSO based NMPC

Algorithm 1: PSO algorithm for NMPC

Input: $y(k)$: Measured outputs from Plant
Output: $u_1(k),\ldots,u_{N_c}(k)$: control input
Data: N_p, N_c, Q, R, C_1, C_2, N, i, $\omega_{\min}$, $\omega_{\max}$, $N=N_c$, $Tolerance$, y_{ref}, $p_{\max}$, $p_{\min}$, q

$k \leftarrow 0$ // For each sample
for $i \leftarrow 1$ **to** S_p **do**
  /* generate random values from normal uniform distribution within range $(p_{\min}, p_{\max})$ of N dimensional swarm where $N=N_c$; use best position (P_i^L) obtained at previous NMPC step for $i=S_p$ */
  declare $P_i(0)$
  $V_i(0) \leftarrow 0$
  $P_i^L(0) \leftarrow P_i(0)$
  $F_i^* \leftarrow J(P_i(0))$
end
$F^* \leftarrow min(F_i^*)$ for all i // (best function value for all $F_i, i=1,2,\ldots,S_p$)
$P_i^G(0) \leftarrow \arg\min(F^*)$ // calculate best position
for $q \leftarrow 1$ **to** n_i **do** // for every iteration and
  for $i \leftarrow 1$ **to** S_p **do** // for every particle:
    /* random values normal distribution within (0,1)) */
    declare r_1, r_2
    /* Update velocity $V_i(q+1)$ and position $P_i(t+1)$ */
    $V_i(q+1) = \omega V_i(q) + c_1 r_1 (P_i^L(q) - P_i(t)) + c_2 r_2 (P_i^G(q) - P_i(t))$
    $P_i(q+1) = P_i(q) + V_i(q+1)$
    if $P_i(q+1) < p_{\min}$ **then**
      /* consider the constraints;in NMPC $p_{\max}$, $p_{\min}$ are nothing but $u_{\max}$, $u_{\min}$ */
      $P_i(q+1) \leftarrow p_{\min}$
      $V_i(qt+1) \leftarrow P_i(q+1) - P_i(q)$
    end
    if $P_i(q+1) > p_{\max}$ **then**
      $P_i(q+1) \leftarrow p_{\max}$
      $V_i(q+1) \leftarrow P_i(q+1) - P_i(q)$
    end
    $F \leftarrow J(P_i(q+1))$ // (calculate cost function)
    if $F < F_i^*$ **then**
      $F_i^* \leftarrow F$ // (update the best cost function value,
      $P_i^L \leftarrow P_i(q+1)$ // and position for particle i)
    end
    if $F > F_i^*$ **then**
      $F_i^* \leftarrow F$ // (update the best cost function value,
      $P_i^G \leftarrow P_i(q+1)$ // and global best position for particle i)
    end
  end
  If stopping criterion satisfied, output the solution $U_k = P_i^G(q+1)$ which is best so far.
end
The first output solution u_1^* form the set U_k is applied to the plant until next sampling instant.
$k \leftarrow k+1$

this paper. The constrained optimization problem (7) can be represented with the following constraints:

$$\begin{aligned} \min_{u} \quad & J(u) \\ \text{subject to} \quad & h_i(u) \leq 0, \quad i=1,2,\ldots,m, \end{aligned} \tag{7}$$

where J is the objective function and u is the decision vector with n_u variables which is defined in (1) as input vector. The formulation of the constraints in (7) is not restrictive, since an inequality constraint of the form $h_i(u) \geq 0$ can be also represented as $-h_i(u) \leq 0$. In the following equation (8), the PSO algorithm with penalty function approach is introduced to solve the constrained optimization problem which is generally defined as:

$$F(u,\sigma) = \begin{cases} J(u) & \text{if } u \text{ is feasible,} \\ J(u) + \sigma \sum_{i=1}^{m} [\max\{0, h_i(u)\}]^2 & \text{otherwise.} \end{cases} \tag{8}$$

In this way the constrained optimization problem is represented by unconstrained problem. Three aspects influence how efficiently a nonlinear programming (NLP) problem may be solved: the magnitude of the problem, the kind of the problem, and the optimization algorithm to be applied. The problem scale or problem style are not addressed in this paper. Because of nonlinear dynamic systems, improving the optimization algorithm is preferred in this paper for engineering problems.

The swarm particles will explore a pre-defined search space determined by equation conditions (6), (5) and find the best initial positions around the global optimum for u. As the NMPC computation of the optimal input applied to the real-time system mainly depends on the number of predictions required to compute along with the control horizon, the classic PSO algorithm finds some more time requirements because of the $O(n^2+1)$ complexity. Maximizing the number of particles produced better results than repeated runs with fewer particles for issues with a large number of design variables and several local minima. This motivates us to find a solution towards parallel computations in the NMPC with a reduced computational load of a single PSO solver.

3.2. PPSO-DAC

The divide-and-conquer approach is proposed by dividing the computational load of a single solver using multiple parallel solvers with the division of the swarm population [18, 20, 21]. It achieves the speed of computation within each sampling instant, which is the major challenge of the NMPC. This approach is beneficial for implementing it on the hardware with reconfigurable parallel architecture. All particles also known as design points are independent of each other for each step of time (design iteration) and can be easily studied in parallel. PSO as an iterative method, which depends on the size of the population and the number of iterations, requires more time to reach towards the optimum value. Therefore, the divide-and-conquer approach is suggested to decrease the computational efforts. In this approach, the populations of the swarm (SP) are divided into two sub-swarms, i.e. $\{S_{P_1}, S_{P_2}\}$ so that these sub-swarms will be used to solve the sub-optimum global solutions

in parallel fashioned PSO solvers. The final global particle position is $U_{\text{opt}}(k)$ which satisfies the condition $J_{\min}(S_{P_G})$, where $S_{P_G} = \{S_{P_{G1}}, S_{P_{G2}}\}$. In online optimization, which is a core part of NMPC, this benefit of the proposed technique can be surprisingly helpful. A block diagram of the proposed parallel solution is shown Fig. 3.

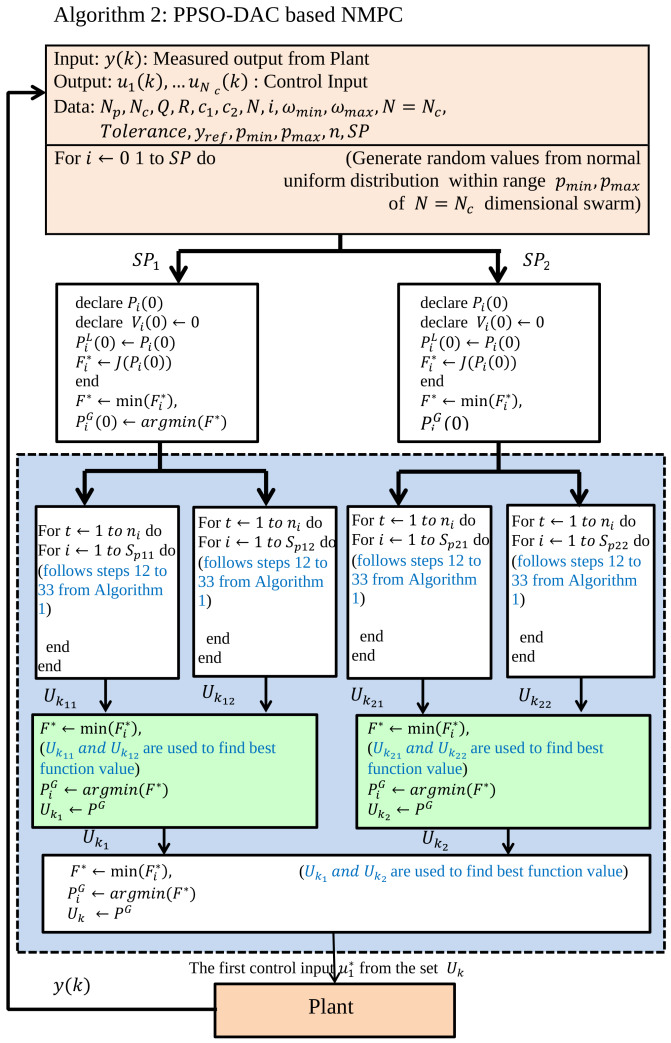


Fig. 3. Block diagram which shows the PPSO-DAC based NMPC with algorithm

Remark 2. The working mechanism of PPSO-DAC resembles the working principle of the based algorithm described in the paper [21]. The swarm is divided and the computed set of control inputs of each batch is again tested with fitness function and after this the feasible most fitted u_1^* from U_k is applied to the plant, is not provided therein.

Remark 3. Because of the iterative and random variable-based process, the output will vary in magnitude even if the mean is small for some applications. This issue could be solved by smoothing the inputs through interpolation of the obtained input sequence with the initial sequence or by using the curve fitting method [14].

4. APPLICATION OF THE PPSO-DAC-NMPC SCHEME TO SRIP

It is possible to extend this PPSO-based NMPC scheme and its real-time implementation to different systems. In this part, in order to show the superior performance and effectiveness of the proposed algorithm, we applied the proposed PPSO-based NMPC algorithm to an SRIP system supplied by QUANSER (National Instruments). The SRIP is chosen as it requires online computing efficiency. A simplified control relevant model of SRIP with 2 degree-of-freedom is used to prove the effective utilization of the proposed algorithm. The system consists of a rotational arm and a pendulum where the rotation of the arm is actuated by a motor to balance the pendulum in an inverted position [22, 23]. The design parametrs used for SRIP are mentioned in Table 1. A nonlinear state space model is developed for the same.

Table 1
Parameters of SRIP system

Symbols	Description	Value (unit)
R_m	Motor armature resistance	2.60 (Ω)
k_t	Motor torque constant	0.00767 (N m/A)
k_m	Motor back-EMF constant	0.00767 (V s/rad)
K_g	Total gear ratio	70 (–)
J_r	Motor armature moment of inertia	0.0010 (kg m^2)
g	Gravitational constant	9.81 (m/s^2)
m_p	Pendulum mass with T-fitting	0.127 (kg)
L_p	Full length of the pendulum (w/T-fitting)	0.337 (m)
L_r	Distance from pivot to center of gravity	0.216 (m)
J_p	Pendulum moment of inertia	0.00120 (kg m^2)
B_r	Viscous damping coefficient as seen at the rotary arm axis	0.00240 (N m s/rad)
B_p	Viscous damping coefficient as seen at the pendulum axis	0.00240 (N m s/rad)

4.1. Control relevant modelling

The plane of the pendulum is orthogonal to the radial arm. The SRIP system is derived with mathematical expression as follows [24]:

$$\left(m_p L_r^2 + \frac{1}{4} m_p L_p^2 \cos(\alpha^2) + J_r\right) \ddot{\theta} - \left(\frac{1}{2} m_p L_p L_r \cos(\alpha)\right) \ddot{\alpha} + \left(\frac{1}{2} m_p L_p^2 \sin(\alpha)\cos(\alpha)\right) \dot{\theta}\dot{\alpha} + \left(\frac{1}{2} m_p L_p L_r \sin(\alpha)\right) \dot{\alpha}^2 = \tau - B_r \dot{\theta}, \tag{9a}$$

$$-\frac{1}{2}m_pL_pL_r\cos(\alpha)\ddot{\theta}+\left(J_p+\frac{1}{4}m_pL_p^2\right)\ddot{\alpha}$$
$$-\frac{1}{4}m_pL_p^2\sin(\alpha)\cos(\alpha)\dot{\theta}^2$$
$$-\frac{1}{2}m_pLpg\sin(\alpha)=-B_p\dot{\alpha}, \tag{9b}$$

where the torque applied to the base of the rotary arm (i.e., at the load gear) is generated by a servo motor described by $\tau=\dfrac{\eta_g K_g \eta_m k_t (V_m - K_g k_m \dot{\theta})}{R_m}$. Based on the above nonlinear dynamic equation (9) the nonlinear state-space model is derived where the function f maps the current state and input to the next state $\dot{x}$ [5]. This model is used to forecast the state trajectory over a prediction horizon N_p and to move the state from a starting condition to a location that is desired with the required control action taken. The continuous model described by the nonlinear state space model is discretized with sample time T_s. For this, the equations of motion of the pendulum and the rotary arm are defined as equations of difference (10). Provided the notation vector $v(k)$ and sample time T_s, using the forward Euler discretization, the differential system equations are obtained, yielding:

$$x_{e_{k+1}} \approx x_{e_k} + T_s f_e(x_{e_k}, u_{e_k}) = f_d(x_{e_k}, u_{e_k}). \tag{10}$$

The model is extended with an additional delay state to model the delay between the instant the state variables x_{e_k} are evaluated and the time when a new control action $u_{e_{(k+1)}}$ is made available (11). Considering the new state vector $x_k = [x_{e_k}^T \quad x_{u_k}^T]^T$, the input vector $u_k = u_{e_k}$, and $f(x_k, u_k) = [f_d(x_{e_k}, x_{u_k})^T \quad u_{e_k}^T]^T$, the model takes the form:

$$x_{k+1} = f(x_k, u_k) \tag{11}$$

For the discretization of the model proper selection of sampling time is required. For obtaining greater information about the process model a smaller sampling time will be useful to improve the controller performance; but if the sampling time is too small, then it will increase the time required to compute the control input. If the selected sampling time is large, then a large error will be generated during the control process despite then decreased computational burden. This may cause losing the stability of the system. The system has a damping factor ζ which is 0.7 and the natural frequency (ω_n) is 4 rad/sec. The sampling time is selected as 20 ms by considering the dynamics of the system.

4.2. Control mechanism

The system has two main control actions i.e., balance control and the swing-up control [22]. In this paper, the authors have focused their research on a balanced control of the inverted pendulum with fast disturbance rejection. According to the current states of the RIP system at instant k, it is possible to predict the dynamic states of the next N_p steps depending on equation (10) and N_c is considered for the control prediction. A large prediction horizon gives more information about the dynamics of the system and results in a better balancing performance of the SRIP. However, a large value of N_p increases the computational time. A larger value of N_c results in a smooth control action, but on the other hand, a larger value of N_c is responsible for a longer computational time. Therefore, a trade-off between performance and the computational time has to be done. Hence, the value of $N_p = 23$ and then $N_c = 3$ has been selected by performing iterative simulations. The selection of Q and R is also based on the consideration of the required output performance of the rotary inverted pendulum (SRIP) system. The weighing matrices are selected as $Q = \text{diag}(1,5)$ and $R = \text{diag}(0.1)$ after various regulations are performed. The constraint on the position of the pendulum (output) is $-20 \leq x_2 \leq +20$ (degree) and the input constraint is considered as $-10 \leq u \leq +10$ (Volt) based on the actual output range of the actuator. The reference for the states is $y_{\text{ref}} = [0; \quad 0; \quad 0; \quad 0]$. The values for penalty are $\sigma = 100$ and $\beta = 2$. The cost function formulated based on these design parameters is solved by the PPSO-DAC-NMPC algorithm based on the divide and conquer approach which will be compared with the PSO-NMPC algorithm.

4.3. Design parameters for PPSO-DAC based NMPC

The proposed PPSO-DAC based NMPC algorithm is extension of the based PSO based NMPC algorithm. The basic PSO algorithm is developed with the following parameters: the number of iterations (n_i) at 50 with population size (S_P) 60 is sufficient for the computation of the optimum output which is decided based on a trial-and-error method as is essential for the practical engineering design to balance between computational efficiency and optimality. Search space, $N = N_c = 3$ such that better response within less computational time will be achieved. The cognitive coefficient c_1 is 0.5 and the social coefficient c_2 is 1.5. The inertial weight (ω) varies from $(\omega_{\max})$ 0.9 to $(\omega_{\min})$ 0.4 per iteration. The values of r_1 and r_2 are selected randomly in the program. For the convergence of the algorithm, a tolerance of $10e^{-4}$ is considered. Initially, velocity (V_i) is 0. Various combinations with population sizes and number of iterations have been tested. And based on these results best possible combination of $n_i = 50$ and $S_P = 80$ has been found sufficient for the desired response of the SRIP. For the PPSO-DAC-NMPC algorithm the population size is divided into two equal groups i.e., $S_{P_1} = 40$ and $S_{P_2} = 40$ with consideration of the same number of iterations i.e., 50. Each group of S_{P_1} and S_{P_2} is again equally divided into $S_{p_{11}} = S_{p_{12}} = S_{p_{21}} = S_{p_{22}} = 20$. The real-time implemented PPSO-DAC-NMPC algorithms results are discussed in the Section 5 below followed with Subsection 4.4.

4.4. Real-time implementation

An advanced model based design platform for control algorithms verification and prototyping must make easy the practical implementation with the same used hardware target and software tools. In this paper, the hardware-in-loop (HIL) simulation is carried out for testing of the algorithm which depicted in Fig. 4. The optical encoders used to measure the arm position and pendulum position and the power amplifier gain is used to boost the small control input applied to the SRIP. All the interconnections of the set-up are developed with the LabVIEW

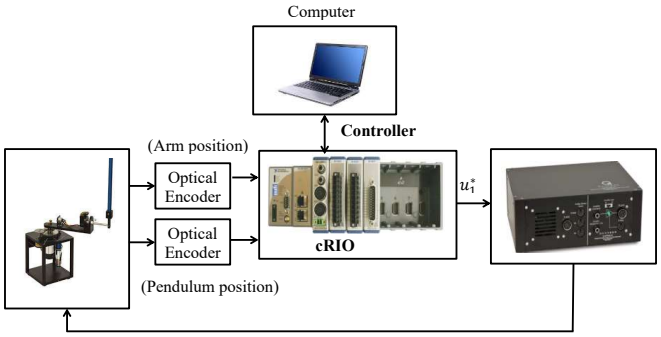


Fig. 4. Schematic diagram of the experimental setup of the SRIP

platform. The controller is implemented on the national instruments cRIO, hybrid embedded hardware which has processors of ARM and the field programmable gate array (FPGA) of parallel computing ability, which leverage fast processing speed. Because of this hardware and software co-design, the feasibility of the proposed control algorithm for the particular real-time hardware will be tested rapidly.

5. RESULTS AND DISCUSSION

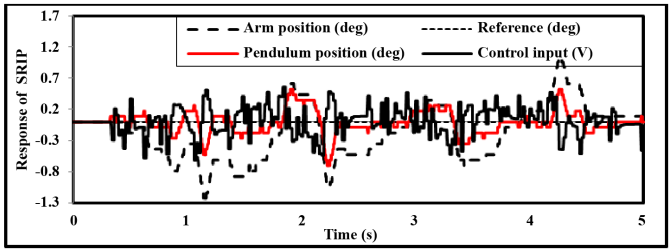
Several experiments were carried out to find the best SRIP optimised with the PPSO in terms of variation in the number of iterations in order to demonstrate the performance of the PPSO with the divide-and-conquer approach. The main goal is to compare the results obtained by NMPC with the Basic PSO and the PPSO-DAC. The response of the SRIP to the proposed PPSO-DAC-NMPC algorithm is depicted in Fig. 5a and 5b. As previously stated, only the balance control of the inverted pendulum is observed in the first case, which yields excellent results because the root mean-square error of the pendulum position for a 5 second simulation is only 0.7108 degree within 8.27 ms when compared to the basic PSO based NMPC Table 2. This proves the efficacy of the algorithm as its main motivation is to compute the control input within sampling time i.e. 20 ms.

Table 2

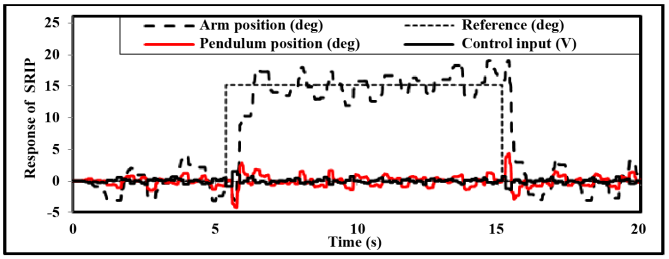
Computational time and the RMS (error) results for balancing the inverted pendulum without applying the disturbance

Drive system	RMS error (α) deg.	Computational time (sec)		Control input (volts)	
		Average	Max	Min	Max
PSO based NMPC	0.7525	0.05601	0.1024	–0.7201	0.7168
PPSO-DAC based NMPC	**0.7108**	**0.00827**	**0.009532**	**–0.4661**	**0.4331**

For testing the robustness of the controller, one step input of 15 degrees with for 5 s to 15 s is applied as a reference for the rotary arm position. The performance of the controller in Fig. 5b shows that the pendulum balanced within a second to reject the disturbance. It shows the 0.8625 degrees root



(a) Response of the the SRIP balancing in normal inverted position



(b) Response of the balance control of SRIP for the step changed arm position as a disturbance to Inverted pendulum

Fig. 5. Response of the inverted pendulum by applying PPSO-DAC based NMPC controller

mean square (RMS) within the average computational time of 8.68 ms which shows the fast disturbance rejection capability of the proposed controller Table 3.

Table 3

Computational time and the RMS (error) result for balancing the inverted pendulum by applying the disturbance to the rotary arm

Drive system	RMS error (α) deg.	Computational time (sec)		Control input (volts)	
		Average	Max	Min	Max
PSO based NMPC	0.9423	0.06021	0.1101	−1.962	1.9922
PPSO-DAC based NMPC	**0.8625**	**0.00868**	**0.01601**	**–1.48**	**1.15**

The particle swarm global optimizer was implemented in parallel in this study. When each fitness evaluation took the same amount of time, the findings for speedup and parallel efficiency were good. Overall, parallel PSO is a new solution for computationally intensive engineering optimization problems that makes efficient use of computer resources.

6. CONCLUSION

Tables 2 and 3 present a comparison of the basic PSO-based NMPC and the proposed PPSO-based NMPC. This demonstrates that the proposed algorithm is significantly more efficient in terms of average computational time, RMS of the inverted pendulum position (deg), and an optimal control input (volts).

Within the defined constraints, the computing time, which is dependent on the number of iterations, population, and constraints in the prediction horizon, does not exceed 20 ms for the entire process. As a result, the proposed PPSO-NMPC algorithm is fast enough to meet the computational speed requirement. It should be noted that classic PSO does not meet the 20 ms time step requirement, despite having a lower RMS than the proposed PPSO-NMPC algorithm. The outcome may differ due to the randomness of the population and the time required to compute the optimal output for each parallel batch.

In conclusion, the parallel particle swarm optimization algorithm introduced in this paper performs well in parallel as long as individual fitness evaluations take the same amount of time. In order to save wasted CPU cycles while maintaining high parallel efficiency, an improved modified approach may be required for optimization problems when the time required for each fitness evaluation varies significantly.

REFERENCES

[1] E.F. Camacho and C. Bordons, "Nonlinear model predictive control: An introductory review," *Lecture Notes in Control and Information Sciences*, vol. 358, pp. 1–16, 2007.

[2] L. Magni, D.M. Raimondo, and F. Allgöwer, "Nonlinear model predictive control," *Lecture Notes in Control and Information Sciences*, vol. 384, 2009.

[3] F. Manenti, "Considerations on nonlinear model predictive control techniques," *Comput. Chem. Eng.*, vol. 35, no. 11, pp. 2491–2509, 2011.

[4] M. Cannon, D. Ng, and B. Kouvaritakis, "Successive linearization nmpc for a class of stochastic nonlinear systems," in *Nonlinear Model Predictive Control*. Springer, 2009, pp. 249–262, doi: 10.1007/978-3-642-01094-1_20.

[5] S.P. Diwan and S.S. Deshpande, "Nonlinear model predictive controller for the real-time control of fast dynamic system," in *2019 International Conference on Communication and Electronics Systems (ICCES)*. IEEE, 2019, pp. 289–294, doi: 10.1109/ICCES45898.2019.9002380.

[6] M. Cannon, "Efficient nonlinear model predictive control algorithms," *Annu. Rev. Control*, vol. 28, no. 2, pp. 229–237, 2004, doi: 10.1016/j.arcontrol.2004.05.001.

[7] Y. Chen, M. Bruschetta, D. Cuccato, and A. Beghi, "An adaptive partial sensitivity updating scheme for fast nonlinear model predictive control," *IEEE Trans. Autom. Control*, vol. 64, no. 7, pp. 2712–2726, 2018, doi: 10.1109/TAC.2018.2867916.

[8] L. Wirsching, H.J. Ferreau, H.G. Bock, and M. Diehl, "An online active set strategy for fast adjoint based nonlinear model predictive control," *IFAC Proc. Vol.*, vol. 40, no. 12, pp. 234–239, 2007, doi: 10.3182/20070822-3-za-2920.00039.

[9] M. Diehl, A. Walther, H.G. Bock, and E. Kostina, "An adjoint-based sqp algorithm with quasi-newton jacobian updates for inequality constrained optimization," *Optim. Methods Software*, vol. 25, no. 4, pp. 531–552, 2010, doi: 10.1080/10556780903027500.

[10] X. Feng, S. Di Cairano, and R. Quirynen, "Inexact adjoint-based sqp algorithm for real-time stochastic nonlinear mpc," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6529–6535, 2020, doi: 10.1016/j.ifacol.2020.12.068.

[11] K. Antoniewicz and K. Rafal, "Model predictive current control method for four-leg three-level converter operating as shunt active power filter and grid connected inverter," *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 65, no. 5, 2017, doi: 10.1515/bpasts-2017-0065.

[12] M. Ławryńczuk and R. Nebeluk, "Computationally efficient nonlinear model predictive control using the l1 cost-function," *Sensors*, vol. 21, no. 17, p. 5835, 2021, doi: 10.3390/s21175835.

[13] Z. Tang and Y. Yang, "Two-stage particle swarm optimization-based nonlinear model predictive control method for reheating furnace process," *ISIJ Int.*, vol. 54, no. 8, pp. 1836–1842, 2014, doi: 10.2355/isijinternational.54.1836.

[14] F. Xu, H. Chen, X. Gong, and Q. Mei, "Fast nonlinear model predictive control on fpga using particle swarm optimization," *IEEE Trans. Ind. Electron.*, vol. 63, no. 1, pp. 310–321, 2015, doi: 10.1109/TIE.2015.2464171.

[15] J. Ziętkiewicz, "Pso-based nonlinear predictive control for unmanned bicycle robot stabilization," *Studia z Automatyki i Informatyki*, 2017.

[16] B. Chopard and M. Tomassini, "Particle swarm optimization," in *An Introduction to Metaheuristics for Optimization*. Springer, 2018, pp. 97–102, doi: 10.1007/978-3-319-93073-2_6.

[17] S. Kiranyaz, T. Ince, and M. Gabbouj, "Particle swarm optimization," in *Multidimensional Particle Swarm Optimization for Machine Learning and Pattern Recognition*. Springer, 2014, pp. 45–82, doi: 10.1007/978-3-642-37846-1_3.

[18] S. Sengupta, S. Basak, and R.A. Peters, "Particle swarm optimization: A survey of historical and recent developments with hybridization perspectives," *Mach. Learn. Knowl. Extr.*, vol. 1, no. 1, pp. 157–191, 2019. doi: 10.3390/make1010010.

[19] J. Zietkiewicz, P. Kozierski, and W. Giernacki, "Particle swarm optimisation in nonlinear model predictive control; comprehensive simulation study for two selected problems," *Int. J. Control*, vol. 94, no. 10, pp. 2623–2639, 2021, doi: 10.1080/00207179.2020.1727957.

[20] J.F. Schutte, *Applications of parallel global optimization to mechanics problems*. University of Florida, 2005.

[21] G. Venter and J. Sobieszczanski-Sobieski, "Parallel particle swarm optimization algorithm accelerated by asynchronous evaluations," *J. Aerosp. Comput. Inf. Commun.*, vol. 3, no. 3, pp. 123–137, 2006.

[22] K.J. Åström and K. Furuta, "Swinging up a pendulum by energy control," *Automatica*, vol. 36, no. 2, pp. 287–295, 2000, doi: 10.1016/S0005-1098(99)00140_5.

[23] M.F. Hamza, H.J. Yap, I.A. Choudhury, A.I. Isa, A.Y. Zimit, and T. Kumbasar, "Current development on using rotary inverted pendulum as a benchmark for testing linear and nonlinear control algorithms," *Mech. Syst. Signal Process.*, vol. 116, pp. 347–369, 2019, doi: 10.1016/j.ymssp.2018.06.054.

[24] Quanser, "Srv02 – et rotary servo and rotary pendulum model," *Instructor Manual*, 2012.