Time-Jerk Optimal Trajectory Planning of Industrial Robot based on Hybrid Particle Swarm Optimization Algorithm

1st Ying Gao,2nd Weinan Xie,3rd Qinghua Li*,4th Xinnian Li
Space Control and Inertial Technology Research Center , Harbin Institute of Technology, Harbin 150001, China
yinggao@stu.hit.edu.cn, xieweinan@hit.edu.cn,huahit@hit.edu.cn,18713596133@163.com

5th Meiling Hu
Research Institute of Intelligent Control Systems
Harbin Institute of Technology
Harbin 150001, China
mlhu1996@163.com

6th Liu Zhao
Department of Control Science and Engineering
Harbin Institute of Technology
Harbin 150001, China
z201107201402@163.com

Abstract—Industrial robots need to execute specific trajectories on production lines, which requires rational planning of robot joint trajectories in Cartesian space. To improve the efficiency and control accuracy of industrial robots, time-jerk multi-objective trajectory optimization is required. The main research of this topic is to apply the particle swarm optimization(PSO) algorithm based on crossover and subgroup to complete the trajectory optimization of a six-degree-of-freedom robot in Cartesian space with time-jerk multi-objective function and compare the optimization effect of the hybrid particle swarm optimization algorithm relative to the traditional particle swarm optimization algorithm. Firstly, a seven segment S-Spline interpolation function is applied to complete the trajectory planning and the basic PSO algorithm is applied to optimize the trajectory for the interpolation time. Then based on the shortcomings of the basic PSO algorithm, the particle swarm optimization algorithm based on crossover and subgroup is proposed to improve the speed and effectiveness of the trajectory optimization algorithm. Finally, simulation experiments show that the total time and joint shocks of robot trajectory planning are greatly reduced,which verifies the effectiveness of the hybrid PSO algorithm applied to robot time-shock optimal trajectory optimization.

Index Terms—Improved PSO algorithm, time-jerk optimal, Seven-segment S-spline curve, Cartesian space trajectory optimization

I. Introduction

The application scenarios of industrial robots are increasing and playing an increasingly powerful role. The purpose of trajectory planning is to establish the function between the motion time and workspace of the industrial robot, and to plan the trajectory of each joint of the robot, so that the end of the robot can complete the specified tasks stably and accurately. The trajectory optimization of industrial robot based on time-jerk optimization is to find the trajectory that makes the multi-objective function of robot time and jerk optimal. Its purpose is to improve the efficiency of industrial robot to complete the task, reduce the error of robot trajectory tracking, and avoid joint jerk increasing robot vibration [1].

Elementary interpolation algorithms mainly include basic polynomial interpolation, mixed polynomial interpolation and spline interpolation. Spline interpolation can better ensure the smoothness of joint trajectory, mainly including B-spline curve, NURBS curve and S-Spline curve. Guo Mingming et al. [2] applied 3-5-3 piecewise polynomial interpolation to realize the smooth operation of robot joint position, velocity and acceleration. Li Zhenna et al. [3] used seven S-type velocity planning to complete the Cartesian space trajectory planning of linear and circular motion based on quaternion to ensure the continuous acceleration of robot joints. However, scholars did not consider the optimization of interpolation time, so the robot could not complete the motion quickly and smoothly with multiple constraints.

The traditional interpolation method is difficult to complete the traditional interpolation algorithm optimization, scholars often use genetic algorithm, particle swarm optimization algorithm and convex optimization algorithm for trajectory optimization. Zhu Shiqiang et al. [4] used 7-degree B-spline curve interpolation method to complete the joint trajectory with continuous velocity, acceleration and pulse, and used the sequential quadratic programming method to complete the optimization of time optimal pulsating continuous trajectory. Lin et al. [5] proposed a joint jerk trajectory optimization method based on particle swarm optimization (PSO), which has the advantages of fast calculation speed and unified calculation method in different tasks. Gasparetto et al. [6] considered the factors of time and jerk at the same time, put forward the comprehensive objective function of robot task time and joint acceleration derivative, applied cubic and quintic spline curve interpolation, adjusted the weight of two optimization objectives of running time and joint jerk, and ensured the improvement of robot work efficiency and smooth motion trajectory.

II. Trajectory planning

Trajectory planning in Cartesian space is the establishment of robot end positions and poses as a function of time. The performance of the trajectory planning algorithm directly determines the operational performance of the robot for a specific work task.

A. Seven-segment S-spline curve

Compared with the polynomial interpolation and trapezoidal velocity interpolation algorithms commonly used in trajectory planning, seven segment S-Spline curve can ensure the continuity of joint angle, angular velocity and angular acceleration, and S-Spline interpolation is easier to set kinematic constraints [7]. The seven segment S-Spline interpolation function de image as follows:

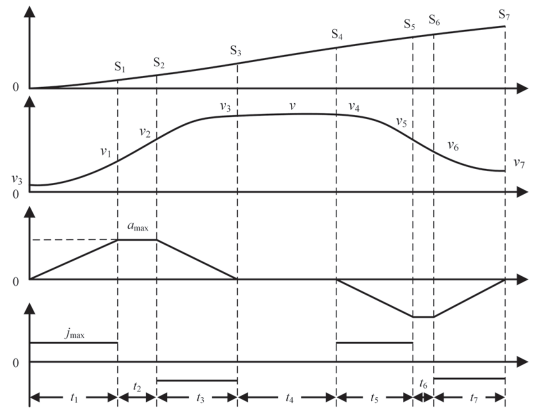


Fig. 1. Seven-segment S-spline function

The velocity function of the seven segment S-Spline is as follows:

$$v(t)=\begin{cases}\frac{1}{2}J_1t & 0\le t<t_1\\ \frac{1}{2}J_1t_1^2+a_{max}(t-t_1) & t_1\le t<t_2\\ v_{max}-\frac{1}{2}J_1(t_3-t)^2 & t_2\le t<t_3\\ v_{max} & t_3\le t<t_4\\ v_{max}-\frac{1}{2}J_1(t-t_4)^2 & t_4\le t<t_5\\ v_{max}-\frac{1}{2}J_1(t_5-t_4)^2-a_{max}(t-t_5) & t_5\le t<t_6\\ \frac{1}{2}J_1(t_7-t)^2 & t_6\le t<t_7\end{cases} \tag{1}$$

Where t is the independent variable time, v is the velocity function, $t_i(i=1,2,...,7)$ is the segmentation time, v_{max} is the maximum velocity, a_{max} is the maximum acceleration and J is the constant acceleration value.

As shown in the formula (1), seven segment S-Spline curves can be divided into the following seven segments:

1) The jerk is constant at J until the desired acceleration is reached a_{max};
2) The acceleration is constant at a_{max};
3) The jerk is constant at $-J$ until the acceleration reaches 0 and the velocity reaches v_{max};
4) The velocity is constant at v_{max};
5) The jerk is constant at J until the acceleration reaches $-a_{max}$;
6) The acceleration is constant at $-a_{max}$;
7) The jerk is constant at $-J$ until the desired acceleration is reached 0 and the velocity reaches v_{max};

Seven times spline interpolation is chosen as the interpolation function, and the optimization of the interpolation time will be continued later.

B. Time-jerk multi-objective function

The purpose of trajectory optimization is to optimize a multi-objective function consisting of minimizing joint running time and joint jerk with kinematic constraints.

The model of time objective function is as follows:

$$T=\sum_{i=1}^{n-1}\Delta t_i \tag{2}$$

Where T is the total time of trajectory planning, n is the number of path points of trajectory interpolation, i is the sequence number of path points of trajectory interpolation, and Δt_i is the length of interpolation time between the $i+1^{st}$ path point and the i^{st} path point.

The model of shock objective function is as follows:

$$J=\sum_{j=1}^{6}\sqrt{\frac{1}{T}\int_0^T\left(jerk_j(t)\right)^2dt} \tag{3}$$

Where J is the derivative of the angular acceleration of the joint, which measures the joint jerk and vibration during robot motion, j is the joint serial number, and $jerk_j(t)$ is the jerk function of the j^{st} joint.

Applying the weighting method to calculate the time performance function and the jerk performance function, the optimized fitness objective function is derived as follows

$$fit=a\sum_{i=1}^{n-1}\Delta t_i+b\sum_{j=1}^{6}\sqrt{\frac{1}{T}\int_0^T\left(jerk_j(t)\right)^2dt} \tag{4}$$

Where fit is the multi-objective optimization function, and a and b are weight coefficients of time performance index and acceleration performance index respectively.

III. Hybrid PSO algorithm for trajectory optimization

A. Basic PSO algorithm

The elementary particle swarm optimization algorithm is generally a bird swarm composed of M particles. The search space of each particle is D-dimensional, and the particles perform the optimization in the search space through iterative calculation of speed and position information. When searching, each particle will consider the searched best historical points and the best historical points of other particles in the group, and change its speed and position on this basis [8].

The velocity of the i-th particle and The position of the i-th particle are expressed as:

$$x_i = (x_i^1, x_i^2, x_i^3, ..., x_i^D) \tag{5}$$

$$v_i = (v_i^1, v_i^2, v_i^3, ..., v_i^D) 1 \le i \le M \tag{6}$$

Where M is the total number of particles in the population, D is the search space dimension, i is the particle number, v_i is the velocity of the particle i in the D-dimensional space, and x_i is the position of the particle i in the D-dimensional space.

Algorithm 1 is the pseudo-code of the standard particle swarm optimization algorithm.

```
Algorithm 1 Basic Particle Swarm Optimization
Require: Number of iterations Ger;Population size
    M;Problem dimension D
Ensure: Globally optimal position vector x*(ger)
 1: $ger \leftarrow 1(initialization)$;
 2: Initializes the position vector and the velocity vector
    of the particle
        $x_i^d = x_{min} + rand() * (x_{max} - x_{min})$
        $v_i^d = v_{min} + rand() * (v_{max} - v_{min})$
 3: while $(|f(x(ger)) * \ge \varepsilon)|) and (ger \le Ger)$ do
 4:     for i=1 to M do
 5:         $f(x_i(ger)) \leftarrow$ Calculate the fitness value of the
              particle $x_i(ger)$);
 6:         Update the value of $pBest_i$ and $gBest_i$;
 7:     end for
 8:     for $i = 1$ $to$ $N$ do
 9:         Update the velocity vetor $V_{ger}$
              and the position vector $X_{ger}$
10:         for $j = 1$ $to$ $D$ do
11:             $v_i^j \leftarrow w * v_i^j + c_1 r_1 * (pBest_i^j - x_i^j)$
                $+ c_2 r_2 * (gBest^j - x_i^j)$;
12:             $x_i^j \leftarrow x_i^j + v_i^j)$
13:         end for
14:     end for
15:     $ger \leftarrow ger + 1$;
16: end while
17: return x*(ger);
```

Where ger is the number of iterations, Ger is the maximum number of iterations, x_{max} and x_{min} represent the range of particle search space, $v_{max}$and v_{min} represent the maximum and minimum search speed of particles, pBest is the individual optimal position of the ith particle, gBest is the global optimal position, c1, c2 are called learning factors, i is the particle number, j is the search space number, w becomes an inertia factor, r1 and r2 are random numbers between 0 and 1.

B. Hybrid PSO algorithm with crossover and subgroup

When the search space dimension is high, the standard PSO algorithm has the problem of slow solution speed and easy to fall into the global optimal solution, so a hybrid particle swarm optimization algorithm with crossover and subgroup is proposed [9].

The way to improve the crossover of the PSO algorithm is as follows:

At each iteration, in each dimension of the search space, the velocity and position of the parent particles are still calculated according to the following formula:

$$\begin{aligned} v_i^j \leftarrow v_i^j + c_1 r_1 * (pBest_i^j - x_i^j) \\ + c_2 r_2 * (gBest^j - x_i^j) \end{aligned} \tag{7}$$

$$x_i^j \leftarrow x_i^j + v_i^j) \tag{8}$$

Then select a certain number of particle swarms according to the probability and put them into the pool according to the probability. The particles in the pool are randomly crossed in pairs to generate a corresponding number of child particles, and the child particles are used to replace their parent particles. In this way, the population size remains unchanged. Through the cross calculation of the position of the parent particle, the position obtained by the child particle can be obtained. The updated velocity and position formula are as follows:

$$\begin{aligned} child_1(v) = \frac{parent_1(v) + parent_2(v)}{|parent_1(v) + parent_2(v)|} |parent_1(v)| \\ child_2(v) = \frac{parent_1(v) + parent_2(v)}{|parent_1(v) + parent_2(v)|} |parent_2(v)| \end{aligned} \tag{9}$$

$$\begin{aligned} child_1(x_i) = p_i * parent_1(x_i) + (1 - p_i) * parent_2(x_i) \\ child_2(x_i) = p_i * parent_2(x_i) + (1 - p_i) * parent_1(x_i) \end{aligned} \tag{10}$$

Where p_i is a random number between 0 and 1. $child_i(v)$(i=1,2) represents the speed of the child particle. $child_i(x_i)$(i=1,2) represents the position of the child particle. $parent_i(v)$(i=1,2) represents the speed of the parent particle. $parent_i(x_i)$(i=1,2) represents the position of the parent particle.

The hybrid particle swarm algorithm obtains the velocity and position vector of the child particle by normalizing the sum of the velocity and position vector of the parent particle. Crossover operations and subgroup operations can benefit both particles, improve search capabilities, and easily find global optimal solutions quickly.

Taking the time-jerk multi-objective function as the optimization goal, the hybrid particle swarm algorithm based on crossover and subgroup is applied to optimize the trajectory. The specific implementation steps of the algorithm are as follows: [10]:

Step1: The seven segment S-curve function is used to interpolate the target path, and the angle vector of each joint of the robot is obtained.
Step2: Set the hybrid particle swarm solution parameters.
Step3: Set robot kinematics constraint conditions.
Step4: Initialize the velocity vector and position vector of the particle.

Step5: Calculate the fitness value of the particle by the function of formula (4), initialize the individual optimal position and the global optimal position.

Step6: Calculate the position and velocity of the next generation of particles from the updated formula (7),(8),(9),(10)of particle position and velocity.

Step7: From the known particle position vector, that is, the trajectory time vector, the target path is interpolated through the seven segment S-Spline curve function to obtain the position, velocity, acceleration and jerk of each joint of the robot under this condition, and check whether it conforms to the kinematics Restrictions.

Step8: Calculate the fitness value of the particle by the function of formula (4), update the historical optimal position of the particle and the global historical optimal position.

Step9: Determine whether the algorithm meets the iteration termination condition, and return to step 6 if it does not. When the iteration ends, the particle position vector is the trajectory time interval sequence, and step 7 has calculated the corresponding joint motion trajectory information, which is the optimized trajectory of each joint.

IV. Simulation Experiments

A. Experimental Parameters

The UR10 robot is selected as the research object for trajectory optimization in Cartesian space, and the target trajectory consists of four spatial circular arcs and two spatial straight lines. The experiments are based on the HPSO algorithm to complete the time-jerk optimal trajectory optimization. The input of the trajectory optimization algorithm is the interpolation time of the seven segment S-Spline curve, and the experimentally selected parameters are listed below.

TABLE I
Simulation parameter setting

Algorithms	Algorithm parameters	Kinematic constraints
Uniform	$T = 20s, N = 100$	$V_{max} = 1rad/s$
		$a_{max} = 2rad/s^2$
PSO	$N = 100, Ger_{max} = 400$	$V_{max} = 1rad/s$
	$c_1 = c_2 = 1.49445, \omega = 0.8$	$a_{max} = 2rad/s^2$
HPSO	$N = 100, Ger_{max} = 400$	$V_{max} = 1rad/s$
	$c_1 = c_2 = 1.49445, \omega = 0.8$	$a_{max} = 2rad/s^2$

B. Experiment Results

As Fig. 2 shows the histogram of the time interval between adjacent interpolation points of the time scale function function s(t) of the uniform interpolation algorithm Uniform, the standard PSO algorithm and the hybrid particle swarm algorithm, it can be seen that each time interval in the Uniform algorithm is uniformly 0.2s After the optimization of the two particle swarm optimization algorithms, the time interval value becomes uneven and the overall time running time becomes shorter.

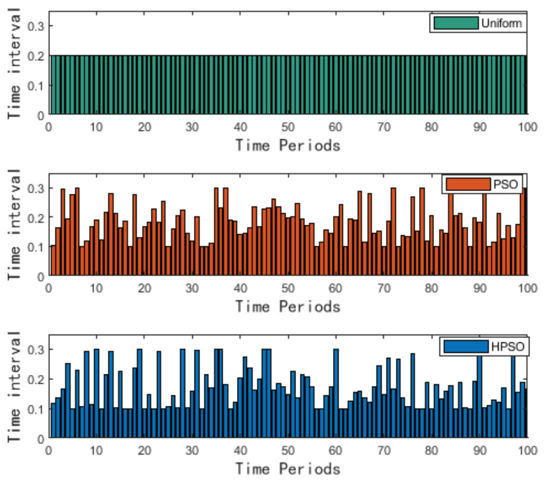


Fig. 2. Three algorithms interpolation time interval

As Fig. 3 shows the convergence curves of the values of the objective fitness functions of the two particle swarm algorithms with the number of iterations, it can be seen that the improved HPSO algorithm by adding crossovers and subgroups improves the stochasticity of the particle swarm algorithm, allowing the HPSO algorithm to find interpolation times that makes the fitness function better for the high-dimensional space search problem of trajectory optimisation under Cartesian space.

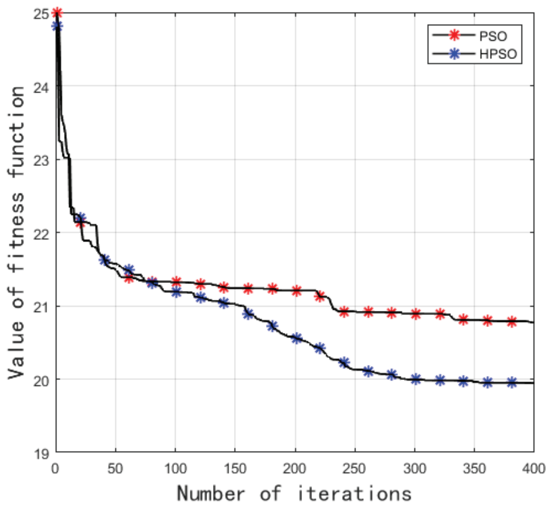
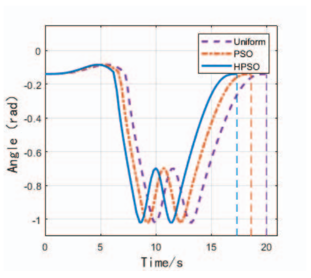
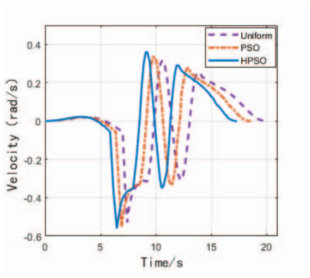


Fig. 3. Convergence process of the fitness function

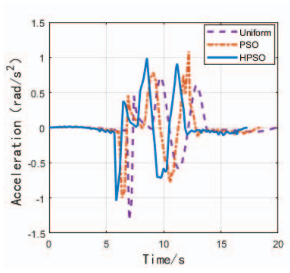
As Fig. 4 shows the comparison of joint position, velocity, acceleration and jerk of the robot Joint2. As can be seen from Fig. 4(a), the trajectory completion time be shortened sequentially from the basic trajectory interpolation,standard PSO algorithm and HPSO algorithm. It is obvious from Fig. 4(d) that the joint jerk of the trajectory optimized by HPSO algorithm is optimized overall compared with the joint jerk of the basic trajectory, and the spikes of the joint jerk disappear after optimization.



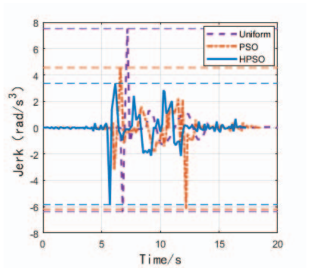
(a) Joint 2 position



(b) Joint 2 velocity



(c) Joint 2 acceleration



(d) Joint 2 jerk

Fig. 4. Three algorithms joint 2 trajectory

TABLE II shows the trajectory planning results of the three algorithms.It can be seen that for the same target trajectory and the same target trajectory, both optimization algorithms achieve the optimization of robot completion runtime and joint impact function than the basic trajectory planning The HPSO algorithm optimizes better compared to the standard PSO algorithm and shortens the completion trajectory time from 20s to 17.3158s.

TABLE II
Experimental results of the three algorithms

Algorithms	Time function	Shock function	Fitness function
Uniform	20s	19.3154 rad/s^3	19.93154
PSO	18.5858s	7.2879 rad/s^3	17.45601
HPSO	17.3158	7.9377 rad/s^3	16.37799

V. Conclusions

Using the six-degree-of-freedom UR10 robot as the object of study, the kinematic modeling and forward and inverse motion analysis were first completed by improving the D-H parameter method. The seven segment S-Spline curve is choose as the time-scale function to complete the trajectory planning in Cartesian space. The adaptive degree function for time-jerk multi-objective optimization is derived and detailed kinematic constraints are described. The HPSO algorithm based on crossover and subgroup ideas is used for trajectory optimization solver. The experimental conclusion gives a comparison of the performance of the basic trajectory planning algorithm, PSO algorithm and HPSO algorithm for trajectory optimization in Cartesian space. It was demonstrated that the trajectory optimisation based on the HPSO algorithm reduced the robot running time from 20s to 17.3158s and the jerk function value from 19.3154 rad/s^3 to 7.9377 rad/s^3, resulting in a significant improvement in the speed and stability of the industrial robot operation.

References

[1] Li L, Shang JY, Feng YL, Huai YAW. A review of research on trajectory planning of articulated industrial robots[J]. Computer Engineering and Applications,2018,54(05):36-50.

[2] Guo Ming-ming,Liu Man-lu,Zhuang Hua,et al. Improved robot time-optimal trajectory planning algorithm optimized by differential evolutionary algorithm[J]. Automation Instrumentation,2018,39(01):35-39(in Chinese).

[3] Li Zhen-na, Wang Tao, Wang Bin-rui,et al. Cartesian spatial trajectory planning of manipulator based on S-shaped velocity profile with constraints[J]. Transactions on Intelligent Systems,2019,14(04):655-661(in Chinese).

[4] Zhu Shi-qiang,Liu Song-guo,Wang Xuan-yin. Time-optimal pulsating continuous trajectory planning algorithm for robots[J]. Chinese Journal of Mechanical Engineering,2010,46(03):47-52(in Chinese).

[5] Lin H I . A fast unified method for finding minimum shortcut robot joint trajectories using particle swarm optimization[J]. Journal of Intelligent Robotic Systems, 2014, 75(3-4):379-392

[6] Gasparetto A,Zanotto V . A new method for smooth trajectory planning of robot manipulators[J]. Mechanism Machine Theory, 2007, 42(4):455-471.

[7] Pan Hai-hong,Yuan Shan-shan,Huang Xu-feng,et al.Research on full type asymmetric seven-stage S-curve acceleration/deceleration control algorithm[J].Mechanical Science and Technology,2018,12(in chinese).

[8] Banks A , Vincent J , Anyakoha C . A review of particle swarm optimization. Part I: Background and development[J]. Natural Computing, 2007, 6(4):467-484.

[9] Qian X , Cao M , Su Z , et al. A Hybrid Particle Swarm Optimization (PSO)-implex Algorithm for Damage Idenfication of Delaminated Beams[J]. mathematical Problems in Engineering,2012,(2012-11-7), 2012, 2012(pt.10):1239-125

[10] W. Zhang and S. Fu, ”Time-optimal Trajectory Planning of Dulcimer Music Robot Based on PSO Algorithm,” 2020 Chinese Control And Decision Conference (CCDC), 2020, pp. 4769-4774, doi: 10.1109/CCDC49329.2020.9164017.