第3章 基于轨迹规划的机械手控制

机械系统在运动过程中会产生振荡，而振荡会造成额外的能量消耗。在定点运动过程中，不同的运动轨迹会造成不同程度的振荡，因此有必要研究如何设计最优轨迹控制器使得机械系统在整个运动过程所消耗的能量最小。

以机器人轨迹规划为例，轨迹规划方法分为两方面：对于移动机器人是指移动的路径轨迹规划，如在有地图条件或没有地图的条件下，移动机器人按什么样的路径轨迹来行走；对于工业机器人则指机械臂末端行走的曲线轨迹规划，或机械臂在运动过程中的位移、速度和加速度的曲线轮廓规划。

本章介绍固定时间定点位置运动的最优轨迹规划设计问题，最优轨迹的目标是使整个运动过程中消耗的能量最小。智能优化算法（如遗传算法、粒子群算法、差分进化算法等）可以用来优化任何形式的定点运动，为了设计最优轨迹，这里介绍基于差分进化的轨迹优化算法。

3.1 差分进化算法

3.1.1 差分进化算法的提出

差分进化（Differential Evolution，DE）算法是模拟自然界生物种群以“优胜劣汰、适者生存”为原则的进化发展规律而形成的一种随机启发式搜索算法，是一种新兴的进化计算技术。它由 Rainer Storn 和 Kenneth Price 于 1995 年提出[1]。由于其简单易用、稳健性好以及强大的全局搜索能力，已在多个领域取得成功。

差分进化算法保留了基于种群的全局搜索策略，采用实数编码、基于差分的简单变异操作和一对一的竞争生存策略，降低了遗传操作的复杂性。同时，差分进化算法特有的记忆能力使其可以动态跟踪当前的搜索情况，以调整其搜索策略，具有较强的全局收敛能力和鲁棒性，且不需要借助问题的特征信息，适于求解一些利用常规的数学规划方法所无法求解的复杂环境中的优化问题，采用差分进化算法可实现轨迹规划[2]。

实验结果表明，差分进化算法的性能优于粒子群算法和其他进化算法，该算法已成为一种求解非线性、不可微、多极值和高维的复杂函数的一种有效的和鲁棒性好的方法。

3.1.2 标准差分进化算法

差分进化算法是基于群体智能理论的优化算法，通过群体内个体间的合作与竞争产生的群体智能指导优化搜索。它保留了基于种群的全局搜索策略，采用实数编码、基于差分的简单变异操作和一对一的竞争生存策略，降低了遗传操作的复杂性，它特有的记忆能力使其可以动态跟踪当前的搜索情况已调整其搜索策略。具有较强的全局收敛能力和鲁棒性。差分进化算法的主要优点可以总结为三点：一是待定参数少，二是不易陷入局部最优，三是收敛速度快。

差分进化算法根据父代个体间的差分矢量进行变异、交叉和选择操作，其基本思想是从某一随机产生的初始群体开始，通过把种群中任意两个个体的向量差加权后按一定的规则与第3个个体求和来产生新个体，然后将新个体与当代种群中某个预先决定的个体相比较，如果新个体的适应度值优于与之相比较的个体的适应度值，则在下一代中就用新个体取代旧个体，否则旧个体仍保存下来，通过不断地迭代运算，保留优良个体，淘汰劣质个体，引导搜索过程向最优解逼近。

在优化设计中，差分进化算法与传统的优化方法相比，具有以下主要特点：

(1) 差分进化算法从一个群体即多个点而不是从一个点开始搜索，这是它能以较大的概率找到整体最优解的主要原因；

(2) 差分进化算法的进化准则是基于适应性信息的，无须借助其他辅助性信息(如要求函数可导或连续)，大大地扩展了其应用范围；

(3) 差分进化算法具有内在的并行性，这使得它非常适用于大规模并行分布处理，减小时间成本开销；

(4) 差分进化算法采用概率转移规则，不需要确定性的规则。

3.1.3 差分进化算法的基本流程

差分进化算法是基于实数编码的进化算法，整体结构上与其他进化算法类似，由变异、交叉和选择三个基本操作构成。标准差分进化算法主要包括以下4个步骤。

1. 生成初始群体

在 n 维空间里随机产生满足约束条件的 M 个个体，实施措施如下：

$$x_{ij}(0)=\mathrm{rand}_{ij}(0,1)(x_{ij}^{\mathrm{U}}-x_{ij}^{\mathrm{L}})+x_{ij}^{\mathrm{L}} \tag{3.1}$$

其中 x_{ij}^{U} 和 x_{ij}^{L} 分别是第 j 个染色体的上界和下界，$\mathrm{rand}_{ij}(0,1)$是[0,1]之间的随机小数。

2. 变异操作

从群体中随机选择3个个体 x_{p1}、x_{p2} 和 x_{p3}，且 $i\neq p_1\neq p_2\neq p_3$，则基本的变异操作为

$$h_{ij}(t+1)=x_{\mathrm{p1j}}(t)+F(x_{\mathrm{p2j}}(t)-x_{\mathrm{p3j}}(t)) \tag{3.2}$$

如果无局部优化问题，变异操作可写为

$$h_{ij}(t+1)=x_{\mathrm{bj}}(t)+F(x_{\mathrm{p2j}}(t)-x_{\mathrm{p3j}}(t)) \tag{3.3}$$

其中，$x_{\mathrm{p2j}}(t)-x_{\mathrm{p3j}}(t)$为差异化向量，此差分操作是差分进化算法的关键，$F$ 为缩放因子，p_1、p_2、p_3 为随机整数，表示个体在种群中的序号，$x_{\mathrm{bj}}(t)$为当前代中种群中最好的个体。

由于式(3.3)借鉴了当前种群中最好的个体信息,可加快收敛速度。

3. 交叉操作

交叉操作是为了增加群体的多样性,具体操作如下:

$$v_{ij}(t+1)=\begin{cases}h_{ij}(t+1), & \text{rand}l_{ij}\leqslant CR\\ x_{ij}(t), & \text{rand}l_{ij}>CR\end{cases} \tag{3.4}$$

其中,$\text{rand}l_{ij}$ 为[0,1]之间的随机小数,CR 为交叉概率,$CR\in[0,1]$。

4. 选择操作

为了确定 $x_i(t)$是否成为下一代的成员,试验向量 $v_i(t+1)$和目标向量 $x_i(t)$对评价函数进行比较:

$$x_i(t+1)=\begin{cases}v_i(t+1), f(v_{i1}(t+1),\cdots,v_{in}(t+1))<f(x_{i1}(t),\cdots,x_{in}(t))\\ x_{ij}(t), f(v_{i1}(t+1),\cdots,v_{in}(t+1))\geqslant f(x_{i1}(t),\cdots,x_{in}(t))\end{cases} \tag{3.5}$$

反复执行步骤 2～4 的操作,直至达到最大的进化代数 G,差分进化基本运算流程如图 3.1 所示。

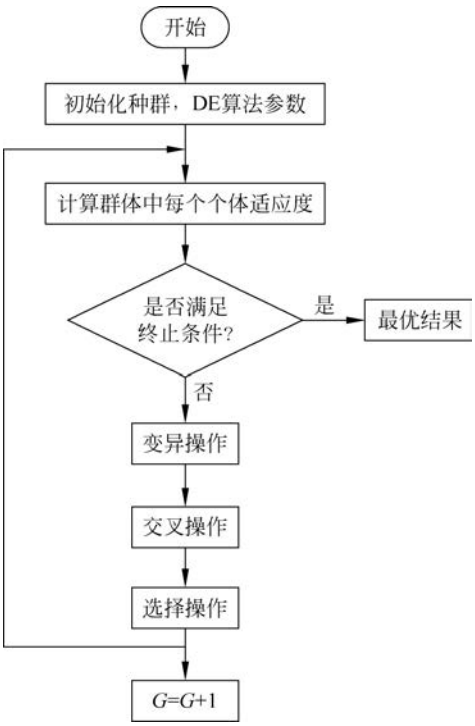

图 3.1 差分进化基本运算流程

3.1.4 差分进化算法的参数设置

对于进化算法而言,为了取得理想的结果,需要对差分进化算法的各参数进行合理的设置。针对不同的优化问题,参数的设置往往也是不同的。另外,为了使差分进化算法的收敛速度得到提高,学者们针对差分进化算法的核心部分—变异向量的构造形式提出了多种的扩展模式,以适应更广泛的优化问题。

差分进化算法的运行参数主要有缩放因子 F、交叉因子 CR、群体规模 M 和最大进化代数 G。

1. 变异因子 F

变异因子 F 是控制种群多样性和收敛性的重要参数。一般在[0,2]之间取值。变异因子 F 值较小时,群体的差异度减小,进化过程不一跳出局部极值导致种群过早收敛。变异因子 F 值较大时,虽然容易跳出局部极值,但是收敛速度会减慢。一般可选在 $F=0.3\sim0.6$。

另外,可以采用下式[3]线性调整变异因子 F:

$$F=(F_{\max}-F_{\min})\frac{T-t}{T}+F_{\min}$$

其中,t 为当前进化代数,T 为最大进化代数,$F_{\max}$ 和 $F_{\min}$ 为选定的变异因子最大和最小值。在算法搜索初期,F 取值较大,有利于扩大搜索空间,保持种群的多样性;在算法后期,收敛

的情况下，F 取值较小，有利于在最佳区域的周围进行搜索，从而提高了收敛速率和搜索精度。

2. 交叉因子 CR

交叉因子 CR 可控制个体参数的各维对交叉的参与程度，以及全局与局部搜索能力的平衡，一般在[0,1]之间。交叉因子 CR 越小，种群多样性减小，容易受骗，过早收敛。CR 越大，收敛速度越大。但过大可能导致收敛变慢，因为扰动大于了群体差异度。根据文献一般应选在[0.6,0.9]之间。

CR 越大，F 越小，种群收敛逐渐加速，但随着交叉因子 CR 的增大，收敛对变异因子 F 的敏感度逐渐提高。

同样，可以采用下式线性调整交叉因子 CR：

$$CR = CR_{\min} + \frac{CR_{\max} - CR_{\min}}{T}t$$

其中 $CR_{\max}$ 和 $CR_{\min}$ 为交叉因子 CR 的最大值和最小值。

为了保证算法的性能，$CR_{\max}$ 和 $CR_{\min}$ 应选取合理的值。随着进化代数的增加，F 线性递减，CR 线性递增，目的是希望改进的 DE 算法在搜索初期能够保持种群的多样性，到后期有较大的收敛速率。

3. 群体规模 M

群体所含个体数量 M 一般介于 $5D$ 与 $10D$ 之间（D 为问题空间的维度），但不能少于 4，否则无法进行变异操作。M 越大，种群多样性越强，获得最优解概率越大，但是计算时间更长，一般取 20～50。

4. 最大迭代代数 G

最大迭代代数 G 一般作为进化过程的终止条件。迭代次数越大，最优解更精确，但同时计算的时间会更长，需要根据具体问题设定。

以上四个参数对差分进化算法的求解结果和求解效率都有很大的影响，因此要合理设定这些参数才能获得较好的效果。

3.1.5 基于差分进化算法的函数优化

利用差分进化算法求 Rosenbrock 函数的极大值

$$\begin{cases} f(x_1,x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 \\ -2.048 \leqslant x_i \leqslant 2.048 \quad (i = 1,2) \end{cases} \tag{3.6}$$

该函数有两个局部极大点，分别是 $f(2.048,-2.048)=3897.7342$ 和 $f(-2.048,-2.048)=3905.9262$，其中后者为全局最大点。

函数 $f(x_1,x_2)$ 的三维图如图 3.2 所示，可以发现该函数在指定的定义域上有两个接近的极点，即一个全局极大值和一个局部极大值。因此，采用寻优算法求极大值时，需要避免陷入局部最优解。

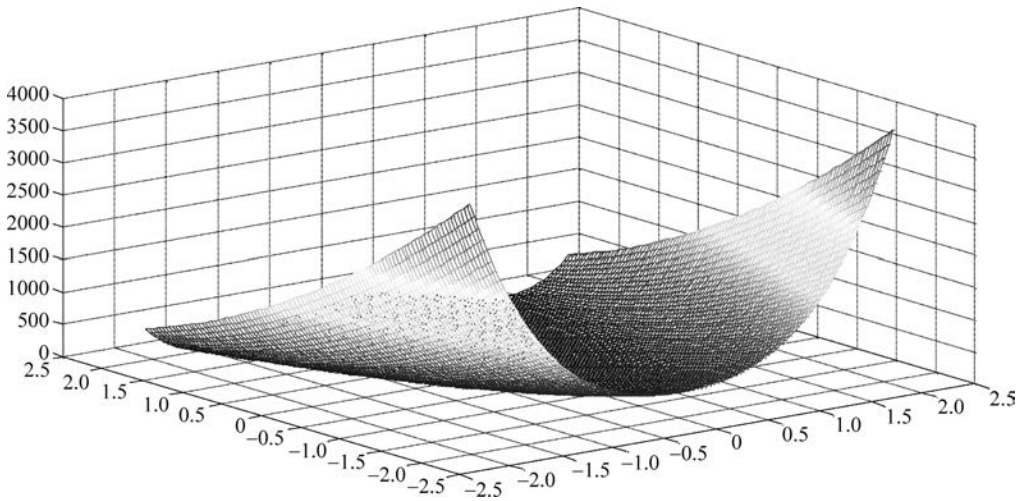

图 3.2　$f(x_1,x_2)$的三维图

仿真程序：chap3_1.m。

```
clear all;
close all;

x_min = -2.048;
x_max = 2.048;

L = x_max - x_min;
N = 101;
for i = 1:1:N
    for j = 1:1:N
       x1(i) = x_min + L/(N-1) * (i-1);
       x2(j) = x_min + L/(N-1) * (j-1);
       fx(i,j) = 100 * (x1(i)^2 - x2(j))^2 + (1 - x1(i))^2;
    end
end
figure(1);
surf(x1,x2,fx);
title('f(x)');

display('Maximum value of fx = ');
disp(max(max(fx)));
```

采用实数编码求函数极大值，用两个实数分别表示两个决策变量 x_1、x_2，分别将 x_1、x_2 的定义域离散化为从离散点 −2.048 到离散点 2.048 的 Size 个实数。个体的适应度直接取为对应的目标函数值，越大越好。即取适应度函数为 $F(x)=f(x_1,x_2)$。

在差分进化算法仿真中，取 $F=1.2$，$CR=0.90$，样本个数为 Size=50，最大迭代次数 $G=30$。按式(3.1)至式(3.5)设计差分进化算法，经过 30 步迭代，最佳样本为 BestS=[−2.048　−2.048]，即当 $x_1=-2.048$，$x_2=-2.048$ 时，Rosenbrock 函数具有极大值，极大值为 3905.9。

适应度函数 F 的变化过程如图 3.3 所示，通过适当增大 F 值及增加样本数量，有效地避免了陷入局部最优解，仿真结果表明正确率接近 100%。

差分进化算法优化程序包括以下两个部分。

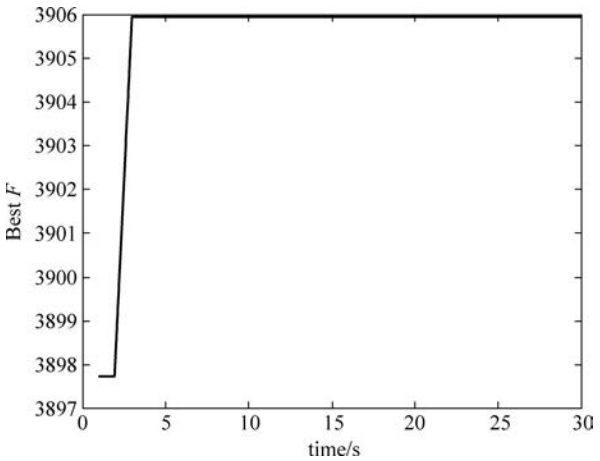

图 3.3　适应度函数 *F* 的优化过程

（1）主程序：chap3_2.m。

```
% To Get maximum value of function f(x1,x2) by Differential Evolution
clear all;
close all;

Size = 30;
CodeL = 2;

MinX(1) = -2.048;
MaxX(1) = 2.048;
MinX(2) = -2.048;
MaxX(2) = 2.048;

G = 50;

F = 1.2;                                          % 变异因子[0,2]
cr = 0.9;                                         % 交叉因子[0.6,0.9]
% 初始化种群
for i = 1:1:CodeL
    P(:,i) = MinX(i) + (MaxX(i) - MinX(i)) * rand(Size,1);
end

BestS = P(1,:);                                   % 全局最优个体
for i = 2:Size
        if(chap3_2obj( P(i,1),P(i,2))> chap3_2obj( BestS(1),BestS(2)))
        BestS = P(i,:);
    end
end

fi = chap3_2obj( BestS(1),BestS(2));

% 进入主要循环,直到满足精度要求
for kg = 1:1:G
     time(kg) = kg;
% 变异
    for i = 1:Size
        r1 = 1;r2 = 1;r3 = 1;
        while(r1 == r2|| r1 == r3 || r2 == r3 || r1 == i || r2 == i || r3 == i )
            r1 = ceil(Size * rand(1));
```

```
            r2 = ceil(Size * rand(1));
            r3 = ceil(Size * rand(1));
        end
        h(i,:) = P(r1,:) + F * (P(r2,:) - P(r3,:));

        for j = 1:CodeL                          % 检查位置是否越界
            if h(i,j)< MinX(j)
                h(i,j) = MinX(j);
            elseif h(i,j)> MaxX(j)
                h(i,j) = MaxX(j);
            end
        end

% 交叉
        for j = 1:1:CodeL
            tempr = rand(1);
            if(tempr < cr)
                v(i,j) = h(i,j);
            else
                v(i,j) = P(i,j);
            end
        end

% 选择
        if(chap3_2obj( v(i,1),v(i,2))> chap3_2obj( P(i,1),P(i,2)))
            P(i,:) = v(i,:);
        end
% 判断和更新
       if(chap3_2obj( P(i,1),P(i,2))> fi)        % 判断当此时的位置是否为最优的情况
          fi = chap3_2obj( P(i,1),P(i,2));
          BestS = P(i,:);
        end
    end
Best_f(kg) = chap3_2obj( BestS(1),BestS(2));
end
BestS                                            % 最佳个体
Best_f(kg)                                       % 最大函数值

figure(1);
plot(time,Best_f(time),'k','linewidth',2);
xlabel('Times');ylabel('Best f');
```

(2) 函数计算程序：chap3_2obj. m。

```
function J = evaluate_objective(x1,x2)           % 计算函数值
  J = 100 * ( x1^2 - x2)^2 + (1 - x1)^2;
end
```

3.2 轨迹规划算法的设计

3.2.1 一个简单的样条插值实例

三次样条插值(简称 Spline 插值)是通过一系列形值点的一条光滑曲线,数学上通过求解三弯矩方程组得出曲线函数组的过程。

定义 3.1 设$[a,b]$上有插值点,$a=x_1<x_2<\cdots<x_n=b$,对应的函数值为 $y_1,y_2,\cdots,$

y_n。若函数 $S(x)$满足 $S(x_j)=y_j(j=1,2,\cdots,n)$上都是不高于三次的多项式。当 $S(x)$在$[a,b]$上具有二阶连续导数，则称 $S(x)$为三次样条插值函数，如图 3.4 所示。

要求 $S(x)$只需在$[x_j,x_{j+1}]$上确定一个三次多项式，设

$$S_j(x)=a_jx^3+b_jx^2+c_jx+d_j,\quad j=1,2,\cdots,n-1 \tag{3.7}$$

其中，a_j,b_j,c_j,d_j 待定，并满足

$$S(x_j)=y_j,\quad S(x_j-0)=S(x_j+0),\quad j=1,2,\cdots,n-1$$

$$S'(x_j-0)=S'(x_j+0),S''(x_j-0)=S''(x_j+0),\quad j=1,2,\cdots,n-1$$

以一个简单的三次样条插值为例，横坐标取 0 至 10 且间隔为 1 的 11 个插值点，纵坐标取正弦函数，以横坐标间距为 0.25 的点形成插值曲线，利用 MATLAB 提供的插值函数 spline 可实现三次样条插值，仿真结果如图 3.5 所示。

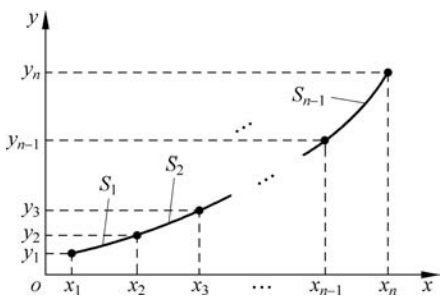

图 3.4　三次样条插值函数

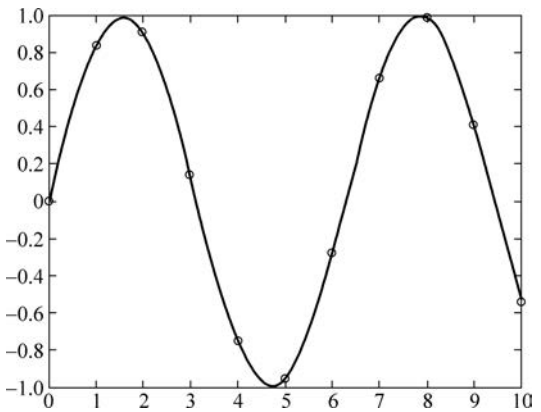

图 3.5　插值效果

仿真实例：chap3_3.m。

```
clear all;
close all
x = 0:10;
y = sin(x);
xx = 0:.25:10;
yy = spline(x,y,xx);
plot(x,y,'o',xx,yy)
```

3.2.2　轨迹规划算法介绍

当前轨迹规划主要应用于移动机器人和工业机器人，常用的轨迹规划方法有如下三种。

1. A 星搜索算法

A 星算法是一种启发式的图搜索算法，可以在有限的条件中得到一个最优解[4]，并在理论上可以保证全局最优解的收敛性，对轨迹规划问题中的各种约束条件可以较好地满足。

A 星算法的核心思想是建立启发函数：

$$f(n)=g(n)+h(n)$$

式中，$g(n)$是从起始点到当前节点 n 的实际代价值，$h(n)$是从当前节点 n 到目标点的估计值。两者相加得到的就是当前节点的总估计值 $f(n)$，然后再对 $f(n)$的大小做出比较，选取 $f(n)$最小的节点作为有效节点，有效节点作为新的起始点，继续搜索下一有效节点，直至到达目标点。

2. 人工势场法

人工势场法是通过设计目标和障碍的势能函数，使机器人处于人工势场中，同时受到目标点的引力和障碍物的斥力，选取合适的势能函数参数和移动步长，根据合力生成一系列路径点，最终完成路径规划[5]。

机械手在人工势场中受到的合力表达式为

$$F(n)=F_{\mathrm{a}}(n)+\sum F_{\mathrm{r}}(n)$$

式中，向量 $F_{\mathrm{a}}(n)$为当前点 n 受到目标点的引力，方向由机器人质心指向目标点，向量 $F_{\mathrm{r}}(n)$为受到障碍物的斥力，方向由障碍物质心指向机器人质心，$\sum F_{\mathrm{r}}(n)$ 为斥力合力，$F(n)$ 为机器人受到的总合力。

若当前点坐标和步长分别为 $p(n)$和 δ，则机器人下一节点的坐标为

$$p(n+1)=p(n)+\delta\frac{F(n)}{\|F(n)\|}$$

依此方法搜索，一步一步到达目标点。

3. 智能优化算法

前面两种方法轨迹规划每一步并不是朝着目标点方向，这样得到的路径并非最优。智能算法通过随机搜索获得最优路径，在最优轨迹方面具有较好的应用。

3.2.3 最优轨迹的设计

采用差分进化方法，可设计机械手轨迹规划的智能优化算法。不失一般性，最优轨迹可在定点运动—摆线运动轨迹的基础上进行优化。摆线运动的表达式如下：

$$\theta_{\mathrm{r}}=(\theta_{\mathrm{d}}-\theta_{0})\left[\frac{t}{T_{\mathrm{E}}}-\frac{1}{2\pi}\sin\left(\frac{2\pi t}{T_{\mathrm{E}}}\right)\right]+\theta_{0} \tag{3.8}$$

其中 T_{E} 是摆线周期，θ_0 和 θ_{d} 分别为角的初始角度和目标角度。

由于差分进化算法是一种离散型的算法，因此需要对连续型的参考轨迹式(3.8)进行等时间隔采样，取时间间隔为$\frac{T_{\mathrm{E}}}{2n}$，则可得到离散化的参考轨迹为

$$\bar{\boldsymbol{\theta}}_{\mathrm{r}}=[\bar{\theta}_{\mathrm{r},0},\bar{\theta}_{\mathrm{r},1},\cdots,\bar{\theta}_{\mathrm{r},2n-1},\bar{\theta}_{\mathrm{r},2n}] \tag{3.9}$$

其中，$\bar{\theta}_{\mathrm{r},j}$ 表示在时刻 $t=\frac{j}{2n}T_{\mathrm{E}}$ 对于 θ_{r} 的采样值($j=0,1,\cdots,2n$)，$\bar{\boldsymbol{\theta}}_{\mathrm{r}}$ 是离散的参考轨迹。

定义 $\Delta\bar{\theta}_j(k)$为与参考轨迹的偏差($j=1,2,\cdots,n-1$)，k 表示差分进化算法中的第 k 次迭代，则得

$$\bar{\theta}_{\mathrm{opj}}(k)=\bar{\theta}_{\mathrm{r},j}+\Delta\bar{\theta}_j(k) \tag{3.10}$$

其中，$\bar{\theta}_{\mathrm{opj}}(k)$表示在时刻 $t=\frac{j}{2n}T_{\mathrm{E}}$ 由差分进化算法的第 k 次迭代得到的关节角的修正角度。

3.3 单关节机械手最优轨迹控制

为了使实际生成的轨迹平滑，在保持轨迹接近参考轨迹的同时，还应确保系统在运动过程中消耗的总能量尽量小，可采用三次样条函数插值并结合差分进化方法来进行轨迹规划。

3.3.1 问题的提出

将单关节机械手简化为一个简单的二阶线性系统：

$$I\ddot{\theta}+b\dot{\theta}=\tau+d \tag{3.11}$$

其中，θ 为角度，I 为转动惯量，b 为黏性系数，τ 为控制输入，d 为加在控制输入上的扰动。

通过差分进化方法，沿着参考路径进行最优规划，从而保证运动系统在不偏离参考路径的基础上，采用 PD 控制方法，实现对最优轨迹的跟踪，使整个运动过程中消耗的能量最小。

3.3.2 最优轨迹的优化

最优轨迹能够通过优化与参考轨迹的偏差来间接地得到。假设系统达到稳态的最大允许时间为 $t=3T_{\mathrm{E}}$，考虑到能量守恒定理，用非保守力做功来表示系统在运动过程中消耗的总能量，目标函数选择为

$$J=\omega\int_0^{3T_{\mathrm{E}}}|\tau\dot{\theta}|\,\mathrm{d}t+(1-\omega)\int_0^{3T_{\mathrm{E}}}|\,\mathrm{dis}(t)\,|\,\mathrm{d}t \tag{3.12}$$

其中，ω 为权值，τ 为控制输入信号，$\mathrm{dis}(t)$为实际跟踪轨迹与理想轨迹之间的距离。

通过采用差分进化算法，优化轨迹式(3.12)，使目标函数最小，从而获得最优轨迹。差分进化算法的设定参数如下：最大迭代次数 G，种群数 Size，搜索空间的维数 D，放大因子 F，交叉因子 CR。经过差分进化算法可得到一组最优偏差，进而得到最优的离散轨迹如下：

$$\bar{\theta}_{\mathrm{op}}=[\bar{\theta}_{\mathrm{op},0},\bar{\theta}_{\mathrm{op},1},\cdots,\bar{\theta}_{\mathrm{op},2n-1},\bar{\theta}_{\mathrm{op},2n}] \tag{3.13}$$

为了获得连续型的最优轨迹，采用三次样条插值进行轨迹规划，即利用三次样条插值的方法对离散轨迹进行插值。插值的边界条件如下：

$$\theta_{\mathrm{op}}(0)=\bar{\theta}_{\mathrm{op},0}=\theta_0$$

$$\theta_{\mathrm{op}}(T_{\mathrm{E}})=\bar{\theta}_{\mathrm{op},2n}=\theta_{\mathrm{d}}$$

$$\dot{\theta}_{\mathrm{op}}(0)=\dot{\bar{\theta}}_{\mathrm{op},0}=\dot{\theta}_0=0$$

$$\dot{\theta}_{\mathrm{op}}(T_{\mathrm{E}})=\dot{\bar{\theta}}_{\mathrm{op},2n}=\dot{\theta}_{\mathrm{d}}=0$$

插值节点为

$$\theta_{\mathrm{op}}(t_j)=\bar{\theta}_{\mathrm{op},j},\quad t_j=\frac{j}{2n}T_{\mathrm{E}},\quad j=1,2,\cdots,2n-1$$

通过差分进化优化插值点的位置，将插值得到的连续函数 $\theta_{\mathrm{op}}(k)$作为关节的最优轨迹。定义跟踪误差为 $e=\theta_{\mathrm{op}}-\theta$，设计 PD 控制律为

$$\tau=k_{\mathrm{p}}e+k_{\mathrm{d}}\dot{e} \tag{3.14}$$

其中，$k_{\mathrm{p}}>0$，$k_{\mathrm{d}}>0$。

3.3.3 仿真实例

单关节机械手可简化为如下被控对象：

$$I\ddot{\theta}+b\dot{\theta}=\tau+d$$

其中，$I=\frac{1}{133}$，$b=\frac{25}{133}$，$d=\sin t$。

采样时间为 $ts=0.001$，采用 Z 变换进行离散化。仿真中，最大允许时间为 $3T_E$，摆线周期 $T_E=1$，取摆线周期的一半离散点数为 $n=500$，则采样时间为 $t_s=\frac{T_E}{2n}=0.001$。

采用样条插值方法，插值点选取 4 个点，即 $D=4$。通过插值点的优化来初始化路径，具体方法为：插值点横坐标固定取第 200、400、600 和第 800 个点，纵坐标取初始点和终止点之间的 4 个随机值，第 i 个样本($i=1,2,\cdots,$Size)第 j 个插值点($j=1,2,3,4$)的值取

$$\theta_{op}(i,j)=\mathrm{rand}(\theta_d-\theta_0)+\theta_0$$

其中，rand 为 0～1 之间的随机值。

根据式(3.8)求 θ_r，采用差分进化算法设计最优轨迹 θ_{op}，取权值 $\omega=0.30$，样本个数 Size=50，变异因子 $F=0.5$，交叉因子 $CR=0.9$，优化次数为 30 次。通过差分进化方法不断优化 4 个插值点的纵坐标值，直到达到满意的优化指标或优化次数为止。

跟踪指令为 $\theta_d=0.5$，采用 PD 控制律式(3.14)，取其中 $k_p=300$，$k_d=10$，仿真结果如图 3.6 至图 3.9 所示。

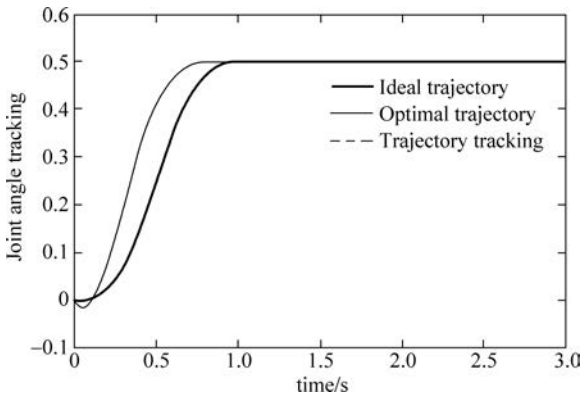

图 3.6 理想轨迹、最优轨迹及轨迹跟踪

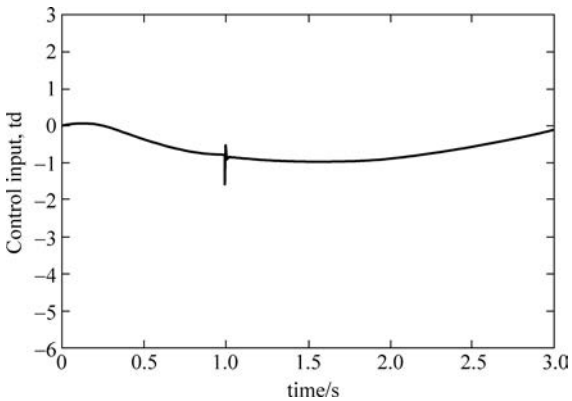

图 3.7 控制输入信号

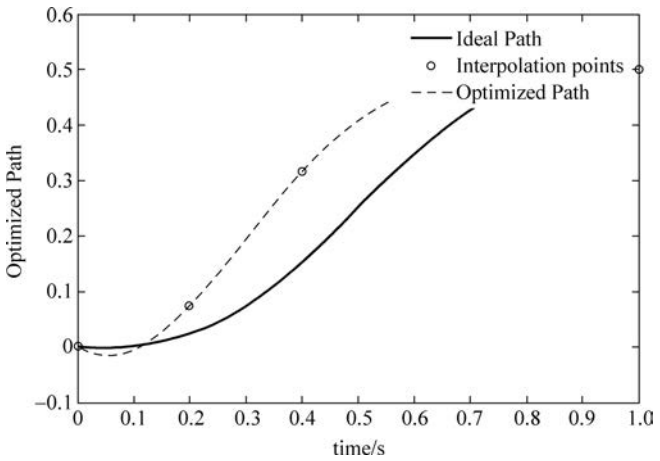

图 3.8　最优轨迹的优化效果

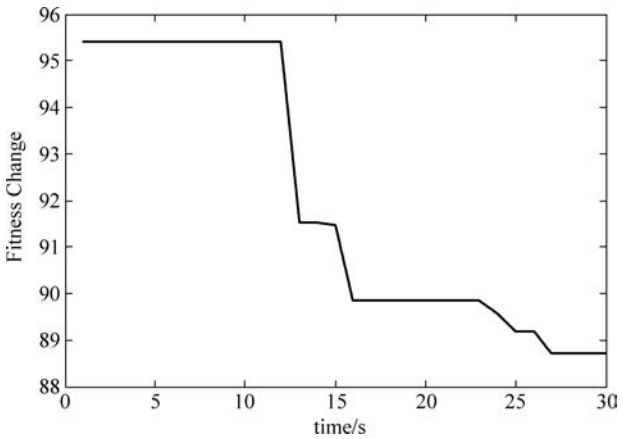

图 3.9　目标函数的优化过程

仿真程序：

(1) 优化主程序：chap3_4.m。

```
clear all;
close all;
global TE G ts
Size = 50;                                  % 样本个数
D = 4;                                      % 每个样本有 4 个固定点,即分成 4 段
F = 0.5;                                    % 变异因子
CR = 0.9;                                   % 交叉因子

Nmax = 30;                                  % DE 优化次数

TE = 1;                                     % 参考轨迹参数 TE
thd = 0.50;
aim = [TE;thd];                             % 摆线路径终点

start = [0;0];                              % 路径起点
tmax = 3 * TE;                              % 仿真时间

ts = 0.001;                                 % 采样时间
```

```
G = tmax/ts;                                          % 仿真时间为 G = 3000
% ********** 摆线参考轨迹 ********** %
th0 = 0;
dT = TE/1000;                                         % 将 TE 分为 1000 个点,每段长度(步长)为 dT

for k = 1:1:G
t(k) = k * dT;                                        % t(1) = 0.001;t(2) = 0.002;.....
if t(k)< TE
     thr(k) = (thd - th0) * (t(k)/TE - 1/(2 * pi) * sin(2 * pi * t(k)/TE)) + th0;
                                                      % 不含原点的参考轨迹(1)
else
    thr(k) = thd;
end
end
% ********** 初始化路径 ********** %
for i = 1:Size
    for j = 1:D
    Path(i,j) = rand * (thd - th0) + th0;
    end
end

% ********** 差分进化计算 ********** %
for N = 1:Nmax
% ********** 变异 ********** %
    for i = 1:Size
        r1 = ceil(Size * rand);
        r2 = ceil(Size * rand);
        r3 = ceil(Size * rand);
        while(r1 == r2||r1 == r3||r2 == r3||r1 == i||r2 == i||r3 == i)
                                                      % 选取不同的 r1、r2、r3,且不等于 i
              r1 = ceil(Size * rand);
              r2 = ceil(Size * rand);
              r3 = ceil(Size * rand);
        end
        for j = 1:D
            mutate_Path(i,j) = Path(r1,j) + F. * (Path(r2,j) - Path(r3,j));
                                                      % 选择前半部分产生变异个体
        end
% ********** 交叉 ********** %
        for j = 1:D
            if rand < = CR
                cross_Path(i,j) = mutate_Path(i,j);
            else
                cross_Path(i,j) = Path(i,j);
            end
        end
% 先进行三次样条插值,此为 D = 4 时的特殊情况 %
        XX(1) = 0;XX(2) = 200 * dT;XX(3) = 400 * dT;XX(4) = 600 * dT;XX(5) = 800 * dT;XX(6) = 1000 * dT;
        YY(1) = th0;YY(2) = cross_Path(i,1);YY(3) = cross_Path(i,2);YY(4) = cross_Path(i,3);
YY(5) = cross_Path(i,4);YY(6) = thd;
        dY = [0 0];
        cross_Path_spline = spline(XX,YY,linspace(0,1,1000));
                          % 输出插值拟合后的曲线,注意步长 nt 的一致,此时输出 1000 个点
        YY(2) = Path(i,1);YY(3) = Path(i,2);YY(4) = Path(i,3);YY(5) = Path(i,4);
        Path_spline = spline(XX,YY,linspace(0,1,1000));
```

```
% *** 计算指标并比较 *** %
        for k = 1:1000
            distance_cross(i,k) = abs(cross_Path_spline(k) - thr(k));
                                    % 计算交叉后的轨迹与参考轨迹的距离值
            distance_Path(i,k) = abs(Path_spline(k) - thr(k));
                                    % 计算插值后的轨迹与参考轨迹的距离值
        end
        new_object = chap3_4obj(cross_Path_spline,distance_cross(i,:),0);
                                    % 计算交叉后的能量消耗最低及路径逼近最佳值的和
        formal_object = chap3_4obj(Path_spline,distance_Path(i,:),0);
                                    % 计算插值后的能量消耗最低及路径逼近最佳值的和

%%%%%%%%%% 选择算法 %%%%%%%%%%
        if new_object <= formal_object
            Fitness(i) = new_object;
            Path(i,:) = cross_Path(i,:);
        else
            Fitness(i) = formal_object;
            Path(i,:) = Path(i,:);
        end
    end
    [iteraion_fitness(N),flag] = min(Fitness);   % 记下第 NC 次迭代的最小数值及其维数

    lujing(N,:) = Path(flag,:)                   % 第 NC 次迭代的最佳路径
    fprintf('N = %d Jmin = %g\n',N,iteraion_fitness(N));
end
[Best_fitness,flag1] = min(iteraion_fitness);
Best_solution = lujing(flag1,:);
YY(2) = Best_solution(1);YY(3) = Best_solution(2);YY(4) = Best_solution(3);YY(5) = Best_solution(4);

Finally_spline = spline(XX,YY,linspace(0,1,1000));
chap3_4obj(Finally_spline,distance_Path(Size,:),1);

figure(3);
plot((0:0.001:1),[0,thr(1:1:1000)],'k','linewidth',2);
xlabel('Time (s)');ylabel('Ideal Path');
hold on;
plot((0:0.2:1), YY,'ko','linewidth',2);
hold on;
plot((0:0.001:1),[0,Finally_spline],'k-.','linewidth',2);
xlabel('Time (s)');ylabel('Optimized Path');
legend('Ideal Path','Interpolation points','Optimized Path');

figure(4);
plot((1:Nmax),iteraion_fitness,'k','linewidth',2);
xlabel('Time (s)');ylabel('Fitness Change');
```

（2）目标函数程序：chap3_4obj.m。

```
% ********** 计算控制输入能量消耗最低及路径逼近最佳值之和的子函数 ********** %
function Object = object(path,distance,flag)      % path,distance 是 2000 维
global TE G ts
w = 0.60;
th_1 = 0;tol_1 = 0;e_1 = 0;
tmax = 3 * TE;                                    % 目标函数积分上限为 3TE
thd = 0.5;
```

```
thop_1 = 0;dthop_1 = 0;
x1_1 = 0;x2_1 = 0;
for k = 1:1:G                                    % Begin th(k)从 2 开始和 thop(1)对应

t(k) = k * ts;
    if t(k)<= TE
       thop(k) = path(k);                        % 要逼近的最优轨迹
       dthop(k) = (thop(k) - thop_1)/ts;
       ddthop(k) = (dthop(k) - dthop_1)/ts;
    else
        thop(k) = thd;
        dthop(k) = 0;
        ddthop(k) = 0;
    end

% 离散模型
I = 1/133;b = 25/133;
d(k) = 1 * sin(k * ts);

x2(k) = x2_1 + ts * 1/I * (tol_1 - b * x2_1 + d(k));
x1(k) = x1_1 + ts * x2(k);

th(k) = x1(k);
dth(k) = x2(k);

e(k) = thop(k) - th(k);
de(k) = (e(k) - e_1)/ts;

kp = 300;kd = 0.30;

    tol(k) = kp * e(k) + kd * de(k);
    energy(k) = abs(tol(k) * dth(k));

    tol_1 = tol(k);
    x1_1 = x1(k);
    x2_1 = x2(k);
    e_1 = e(k);
    thop_1 = thop(k);
    dthop_1 = dthop(k);
end
% ************ 计算总能量 ****************** %
energy_all = 0;
for k = 1:1:G
    energy_all = energy_all + energy(k);
end
dis = sum(distance);                             % 参考轨迹的逼近误差
% ******** 计算目标 ******** %
Object = w * energy_all + (1 - w) * dis;         % used for main.m
if flag == 1
    t(1) = 0;
    th0 = 0;
    for k = 1:1:G                                % > TE 不包含原点
        t(k) = k * ts;
    if t(k)< TE
        thr(k) = (thd - th0) * (t(k)/TE - 1/(2 * pi) * sin(2 * pi * t(k)/TE)) + th0;
```

```
                                                    %不含原点的参考轨迹
        else
            thr(k) = thd;
        end
        end
            figure(1);
            plot(t,thr,'k. - ',t,thop,'k',t,th,'k - .','linewidth',2);
            legend('Ideal trajectory','Optimal trajectory', 'Trajectory tracking');
            xlabel('Time (s)');ylabel('Joint angle tracking');
            figure(2);
            plot(t,tol,'k','linewidth',2);
            xlabel('Time (s)');ylabel('Control input,tol');
    end
    end
```

3.4 双关节机械手最优轨迹控制

3.4.1 系统描述

双关节机械手动力学方程为

$$\boldsymbol{D}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}=\boldsymbol{\tau} \tag{3.15}$$

其中，$\boldsymbol{D}(\boldsymbol{q})$为 2×2 阶正定惯性矩阵，$\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$为 2×2 阶离心和哥氏力项，$\boldsymbol{\tau}$ 为控制输入。

通过差分进化方法，沿着参考路径进行最优规划，从而保证运动系统在不偏离参考路径的基础上，采用 PD 控制方法，实现对最优轨迹的跟踪，使整个运动过程中消耗的能量最小。

3.4.2 规划器设计

最优轨迹能够通过优化与参考轨迹的偏差来间接地得到。假设系统达到稳态的最大允许时间为 $t=3T_{\mathrm{E}}$，考虑到能量守恒定理，用非保守力做功来表示系统在运动过程中消耗的总能量，第 i 个机械臂的目标函数选择为

$$J_i=\omega_i\int_0^{3T_{\mathrm{E}}}|\tau_i\dot{q}_i|\,\mathrm{d}t+(1-\omega_i)\int_0^{3T_{\mathrm{E}}}|\Delta_i(t)|\,\mathrm{d}t \tag{3.16}$$

其中，$i=1,2$，ω_i 为权值，τ_i 为控制输入信号，Δ_i 为实际跟踪轨迹与理想轨迹之间的距离，$\Delta_i=q_{\mathrm{op}_i}(t_j)-q_{\mathrm{r}_i}(t_j)$。

针对双关节机械手，总的目标函数为

$$J=J_1+J_2 \tag{3.17}$$

通过采用差分进化算法，优化轨迹式(3.17)，使目标函数最小，从而获得最优轨迹。差分进化算法的设定参数如下：最大迭代次数 G，种群数 Size，搜索空间的维数 D，放大因子 F，交叉因子 CR。经过差分进化算法可得到一组最优偏差，进而得到第 i 个机械臂的最优离散轨迹如下：

$$\bar{\boldsymbol{q}}_{\mathrm{op}_i}=[\bar{q}_{\mathrm{op}_i,0},\bar{q}_{\mathrm{op}_i,1},\cdots,\bar{q}_{\mathrm{op}_i,2n-1},\bar{q}_{\mathrm{op}_i,2n}],\ i=1,2 \tag{3.18}$$

为了获得连续型的最优轨迹，采用三次样条插值进行轨迹规划，即利用三次样条插值的方法对离散轨迹进行插值。第 i 个机械臂插值的边界条件如下：

$$q_{\mathrm{op}_i}(0)=\bar{q}_{\mathrm{op}_i,0}=q_{0i}$$

$$q_{\mathrm{op}_i}(T_{\mathrm{E}})=\bar{q}_{\mathrm{op}_i,2n}=q_{\mathrm{d}i}$$

$$\dot{q}_{\mathrm{op}_i}(0)=\dot{\bar{q}}_{\mathrm{op}_i,0}=\dot{q}_{0i}=0$$

$$\dot{q}_{\mathrm{op}_i}(T_{\mathrm{E}})=\dot{\bar{q}}_{\mathrm{op}_i,2n}=\dot{q}_{\mathrm{d}i}=0$$

插值节点为

$$q_{\mathrm{op}_i}(t_j)=\bar{q}_{\mathrm{op}_i,j},\quad t_j=\frac{j}{2n}T_{\mathrm{E}},\quad j=1,2,\cdots,2n-1 \tag{3.19}$$

将插值得到的连续函数 $q_{\mathrm{op}_i}(k)$ 作为第 i 个机械臂关节角度跟踪的最优轨迹。定义跟踪误差为 $e_i=q_{\mathrm{op}_i}-q_i$，$\boldsymbol{e}=[e_1\quad e_2]^{\mathrm{T}}$，$\dot{\boldsymbol{e}}=[\dot{e}_1\quad \dot{e}_2]^{\mathrm{T}}$，当忽略重力和外加干扰时，采用独立的 PD 控制，能满足机器人定点控制的要求。设计独立的 PD 控制律为

$$\boldsymbol{\tau}=\boldsymbol{K}_{\mathrm{d}}\dot{\boldsymbol{e}}+\boldsymbol{K}_{\mathrm{p}}\boldsymbol{e} \tag{3.20}$$

其中，$\boldsymbol{K}_{\mathrm{p}}>0$，$\boldsymbol{K}_{\mathrm{d}}>0$。

取跟踪误差为 $\boldsymbol{e}=\boldsymbol{q}_{\mathrm{d}}-\boldsymbol{q}$，采用定点控制时，$\boldsymbol{q}_{\mathrm{d}}$ 为常值，则 $\dot{\boldsymbol{q}}_{\mathrm{d}}=\ddot{\boldsymbol{q}}_{\mathrm{d}}\equiv 0$。则根据 LaSalle 不变集定理，则有 $t\to\infty$ 时，从任意初始条件 $(\boldsymbol{q}_0,\dot{\boldsymbol{q}}_0)$ 出发，均有 $\boldsymbol{q}\to\boldsymbol{q}_{\mathrm{d}}$，$\dot{\boldsymbol{q}}\to 0$。

3.4.3 仿真实例

针对被控对象式(3.15)，选二关节机器人系统(不考虑重力、摩擦力和干扰)，其动力学模型为

$$\boldsymbol{D}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}=\boldsymbol{\tau}$$

其中

$$\boldsymbol{D}(\boldsymbol{q})=\begin{bmatrix}p_1+p_2+2p_3\cos q_2 & p_2+p_3\cos q_2\\ p_2+p_3\cos q_2 & p_2\end{bmatrix}$$

$$\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})=\begin{bmatrix}-p_3\dot{q}_2\sin q_2 & -p_3(\dot{q}_1+\dot{q}_2)\sin q_2\\ p_3\dot{q}_1\sin q_2 & 0\end{bmatrix}$$

取 $\boldsymbol{p}=[2.90\quad 0.76\quad 0.87\quad 3.04\quad 0.87]^{\mathrm{T}}$，$\boldsymbol{q}_0=[0.0\quad 0.0]^{\mathrm{T}}$，$\dot{\boldsymbol{q}}_0=[0.0\quad 0.0]^{\mathrm{T}}$。位置指令为 $\boldsymbol{q}_{\mathrm{d}}(0)=[1.0\quad 1.0]^{\mathrm{T}}$，在控制器式(3.20)中，取 $\boldsymbol{K}_{\mathrm{p}}=\begin{bmatrix}1500 & 0\\ 0 & 1500\end{bmatrix}$，$\boldsymbol{K}_{\mathrm{d}}=\begin{bmatrix}150 & 0\\ 0 & 150\end{bmatrix}$。

采样时间为 $t_{\mathrm{s}}=0.001$，仿真中最大允许时间为 $3T_{\mathrm{E}}$，摆线周期 $T_{\mathrm{E}}=1$，取摆线周期的一半离散点数为 $n=500$，则采样时间为 $t_{\mathrm{s}}=\dfrac{T_{\mathrm{E}}}{2n}=0.001$。

采用样条插值方法，针对第 i 个机械臂，插值点选取 4 个点，即 $D=4$。通过插值点的优化来初始化路径，具体方法为：插值点横坐标固定取第 200、400、600 和第 800 个点，纵坐标取初始点和终止点之间的 4 个随机值，第 m 个样本($m=1,2,\cdots,\mathrm{Size}$)第 j 个插值点($j=1,2,3,4$)的值取

$$q_{\mathrm{op}_i}(m,j)=\mathrm{rand}(q_{\mathrm{d}_i}-q_{0_i})+q_{0_i}$$

其中，rand 为 0～1 之间的随机值。

根据式(3.8)求 q_{r_i}，采用差分进化算法设计最优轨迹 q_{op_i}，取权值 $\omega_i=0.20$，样本个数 $\mathrm{Size}=50$，变异因子 $F=0.5$，交叉因子 $CR=0.9$，优化次数为 30 次。通过差分进化方法不断优化 4 个插值点的纵坐标值，直到达到优化次数为止。仿真结果见图 3.10 至图 3.13 所示。

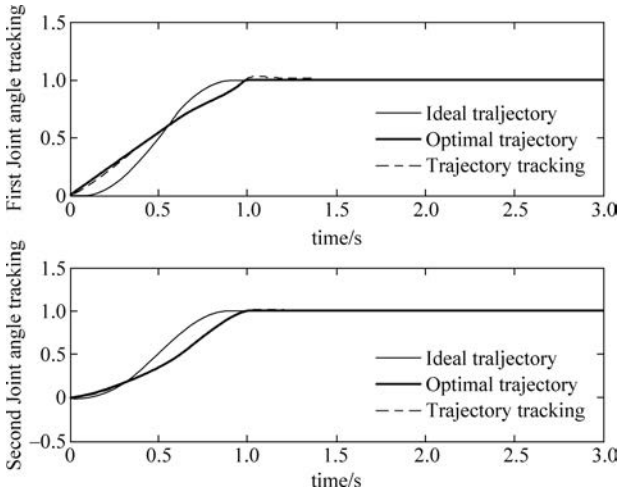

图 3.10　理想轨迹、最优轨迹及轨迹跟踪

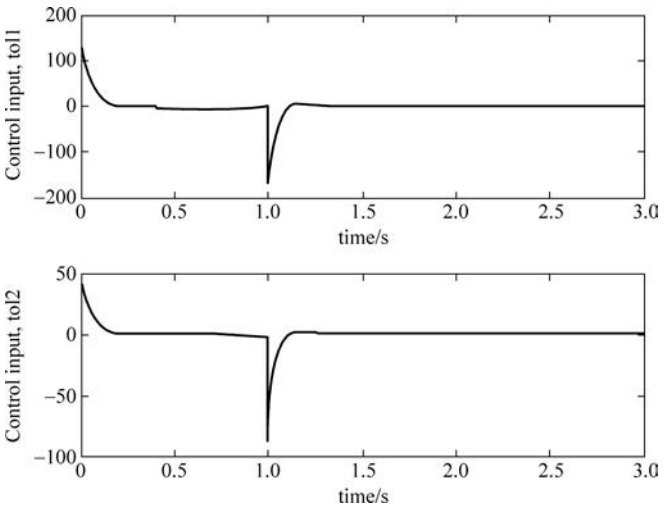

图 3.11　控制输入信号

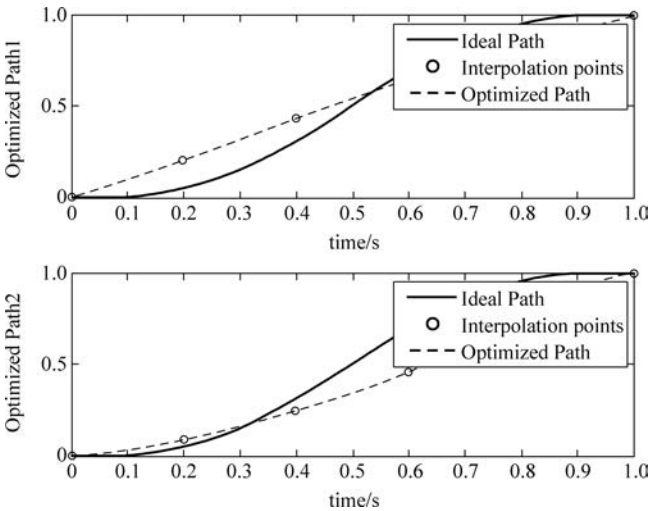

图 3.12　最优轨迹的优化效果

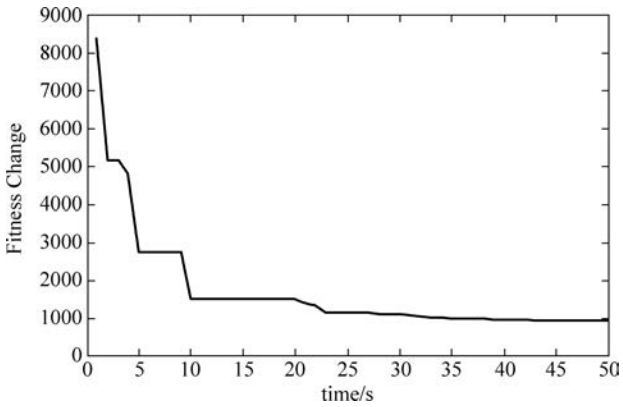

图 3.13　目标函数的优化过程

仿真程序：

(1) 优化主程序：chap3_5.m。

```
clear all;
close all;
global TE G ts
Size = 30;                                          % 样本个数
D = 4;                                              % 每个样本有 4 个固定点,即分成 4 段
F = 0.5;                                            % 变异因子
CR = 0.9;                                           % 交叉因子

Nmax = 50;                                          % DE 优化次数

TE = 1;
TE1 = 1;TE2 = 1;                                    % 参考轨迹参数 TE
q1d = 1.0;q2d = 1.0;

aim1 = [TE1;q1d];                                   % 摆线路径终点
aim2 = [TE2;q2d];                                   % 摆线路径终点

start = [0;0];                                      % 路径起点
tmax = 3 * TE;                                      % 仿真时间

ts = 0.001;                                         % Sampling time
G = tmax/ts;                                        % 仿真时间为 G = 3000
% ********** 摆线参考轨迹 ********** %
q10 = 0;q20 = 0;
q0 = [q10 q20];

dT = TE/1000;                                       % 将 TE 分为 1000 个点,每段长度(步长)为 dT

for k = 1:1:G
t(k) = k * dT;                                      % t(1) = 0.001;t(2) = 0.002;.....
if t(k)< TE
    qr1(k) = (q1d - q10) * (t(k)/TE - 1/(2 * pi) * sin(2 * pi * t(k)/TE)) + q10;
                                                    % 不含原点的参考轨迹(1)
else
    qr1(k) = q1d;
end
```

```
if t(k)< TE
    qr2(k) = (q2d - q20) * (t(k)/TE - 1/(2 * pi) * sin(2 * pi * t(k)/TE)) + q20;
                                                    % 不含原点的参考轨迹(1)
else
    qr2(k) = q2d;
end

end
% *************** 初始化路径 ************** %
for i = 1:Size
    for j = 1:D
    Path1(i,j) = rand * (q1d - q10) + q10;
    Path2(i,j) = rand * (q2d - q20) + q20;
    end
end

% ********** 差分进化计算 *************** %
for N = 1:Nmax
% ************** 变异 ************** %
    for i = 1:Size
        r1 = ceil(Size * rand);
        r2 = ceil(Size * rand);
        r3 = ceil(Size * rand);
        while(r1 == r2||r1 == r3||r2 == r3||r1 == i||r2 == i||r3 == i)
                                                    % 选取不同的 r1,r2,r3,且不等于 i
              r1 = ceil(Size * rand);
              r2 = ceil(Size * rand);
              r3 = ceil(Size * rand);
        end
        for j = 1:D
            mutate_Path1(i,j) = Path1(r1,j) + F. * (Path1(r2,j) - Path1(r3,j));
                                                    % 选择前半部分产生变异个体
            mutate_Path2(i,j) = Path2(r1,j) + F. * (Path2(r2,j) - Path2(r3,j));
                                                    % 选择前半部分产生变异个体
        end
% ***************** 交叉 ***************** %
        for j = 1:D
            if rand < = CR
                cross_Path1(i,j) = mutate_Path1(i,j);
                cross_Path2(i,j) = mutate_Path2(i,j);
            else
                cross_Path1(i,j) = Path1(i,j);
                cross_Path2(i,j) = Path2(i,j);
            end
        end
% 先进行三次样条插值,此为 D = 4 时的特殊情况
        XX(1) = 0;XX(2) = 200 * dT;XX(3) = 400 * dT;XX(4) = 600 * dT;XX(5) = 800 * dT;XX(6) = 1000 * dT;
        YY1(1) = q10;YY1(2) = cross_Path1(i,1);YY1(3) = cross_Path1(i,2);YY1(4) = cross_Path1(i,3);
YY1(5) = cross_Path1(i,4);YY1(6) = q1d;
        YY2(1) = q20;YY2(2) = cross_Path2(i,1);YY2(3) = cross_Path2(i,2);YY2(4) = cross_Path2(i,3);
YY2(5) = cross_Path2(i,4);YY2(6) = q2d;

        dY = [0 0];
        cross_Path1_spline = spline(XX,YY1(1:6),linspace(0,1,1000));
                                % 输出插值拟合后的曲线,注意步长 nt 的一致,此时输出 1000 个点
```

```
        cross_Path2_spline = spline(XX,YY2(1:6),linspace(0,1,1000));
                                    %输出插值拟合后的曲线,注意步长 nt 的一致,此时输出 1000 个点

        YY1(2) = Path1(i,1);YY1(3) = Path1(i,2);YY1(4) = Path1(i,3);YY1(5) = Path1(i,4);
        Path1_spline = spline(XX,YY1,linspace(0,1,1000));

        YY2(2) = Path2(i,1);YY2(3) = Path2(i,2);YY2(4) = Path2(i,3);YY2(5) = Path2(i,4);
        Path2_spline = spline(XX,YY2,linspace(0,1,1000));
% *** 计算指标并比较 *** %
        for k = 1:1000
            delta1_cross(i,k) = abs(cross_Path1_spline(k) - qr1(k));
                                            %计算交叉后的轨迹与参考轨迹的距离值
            delta2_cross(i,k) = abs(cross_Path2_spline(k) - qr2(k));
                                            %计算交叉后的轨迹与参考轨迹的距离值
            delta_Path1(i,k) = abs(Path1_spline(k) - qr1(k));
                                            %计算插值后的轨迹与参考轨迹的距离值
            delta_Path2(i,k) = abs(Path2_spline(k) - qr2(k));
                                            %计算插值后的轨迹与参考轨迹的距离值
        end

        new_object = chap3_5obj(cross_Path1_spline,cross_Path2_spline,delta1_cross(i,:),
delta2_cross(i,:),0);                       %计算交叉后的能量消耗最低及路径逼近最佳值的和
        formal_object = chap3_5obj(Path1_spline,Path2_spline,delta_Path1(i,:),delta_Path2
(i,:),0);                                   %计算插值后的能量消耗最低及路径逼近最佳值的和

%%%%%%%%%% 选择算法 %%%%%%%%%%
        if new_object <= formal_object
            Fitness(i) = new_object;
            Path1(i,:) = cross_Path1(i,:);
            Path2(i,:) = cross_Path2(i,:);
        else
            Fitness(i) = formal_object;
            Path1(i,:) = Path1(i,:);
            Path2(i,:) = Path2(i,:);
        end
    end
    [iteraion_fitness(N),flag] = min(Fitness);  %记下第 NC 次迭代的最小数值及其维数

    lujing1(N,:) = Path1(flag,:)                %第 NC 次迭代的最佳路径
    lujing2(N,:) = Path2(flag,:)                %第 NC 次迭代的最佳路径

    fprintf('N = %d Jmin = %g\n',N,iteraion_fitness(N));

end

[Best_fitness,flag1] = min(iteraion_fitness);
Best_solution1 = lujing1(flag1,:);
Best_solution2 = lujing2(flag1,:);

YY1(2) = Best_solution1(1);YY1(3) = Best_solution1(2);YY1(4) = Best_solution1(3);YY1(5) = Best_
solution1(4);
YY2(2) = Best_solution2(1);YY2(3) = Best_solution2(2);YY2(4) = Best_solution2(3);YY2(5) = Best_
solution2(4);

Finally_spline1 = spline(XX,YY1,linspace(0,1,1000));
```

```
Finally_spline2 = spline(XX,YY2,linspace(0,1,1000));
chap3_5obj(Finally_spline1,Finally_spline2,delta_Path1(Size,:),delta_Path2(Size,:),1);

figure(3);
subplot(211);
plot((0:0.001:1),[0,qr1(1:1:1000)],'k','linewidth',2);
xlabel('Time (s)');ylabel('Ideal Path1');
hold on;
plot((0:0.2:1), YY1,'ko','linewidth',2);
hold on;
plot((0:0.001:1),[0,Finally_spline1],'k-.','linewidth',2);
xlabel('Time (s)');ylabel('Optimized Path1');
legend('Ideal Path','Interpolation points','Optimized Path');
subplot(212);
plot((0:0.001:1),[0,qr2(1:1:1000)],'k','linewidth',2);
xlabel('Time (s)');ylabel('Ideal Path2');
hold on;
plot((0:0.2:1), YY2,'ko','linewidth',2);
hold on;
plot((0:0.001:1),[0,Finally_spline2],'k-.','linewidth',2);
xlabel('Time (s)');ylabel('Optimized Path2');
legend('Ideal Path','Interpolation points','Optimized Path');

figure(4);
plot((1:Nmax),iteraion_fitness,'k','linewidth',2);
xlabel('Time (s)');ylabel('Fitness Change');
```

（2）目标函数程序：chap3_5obj.m。

```
function Object = object(path1,path2,delta1,delta2,flag)   % path,delta 是 2000 维
global TE G ts

q1_1 = 0;q2_1 = 0;
tol1_1 = 0;tol2_1 = 0;
e1_1 = 0;e2_1 = 0;

tmax = 3 * TE;                                             % 目标函数积分上限为 3TE
q1d = 1.0;q2d = 1.0;
q1op_1 = 0;dq1op_1 = 0;
q2op_1 = 0;dq2op_1 = 0;

x1_1 = 0;x2_1 = 0;
x3_1 = 0;x4_1 = 0;
for k = 1:1:G

t(k) = k * ts;

    if t(k)<= TE
       q1op(k) = path1(k);                                 % 要逼近的最优轨迹
       dq1op(k) = (q1op(k) - q1op_1)/ts;
       ddq1op(k) = (dq1op(k) - dq1op_1)/ts;
    else
        q1op(k) = q1d;
        dq1op(k) = 0;
        ddq1op(k) = 0;
    end
```

```
    if t(k)<=TE
       q2op(k)=path2(k);                                    %要逼近的最优轨迹
       dq2op(k)=(q2op(k)-q2op_1)/ts;
       ddq2op(k)=(dq2op(k)-dq2op_1)/ts;
    else
        q2op(k)=q2d;
        dq2op(k)=0;
        ddq2op(k)=0;
    end

%离散模型
p=[2.9 0.76 0.87 3.04 0.87];
g=9.8;

D=[p(1)+p(2)+2*p(3)*cos(x3_1) p(2)+p(3)*cos(x3_1);
    p(2)+p(3)*cos(x3_1) p(2)];
C=[-p(3)*x4_1*sin(x3_1) -p(3)*(x2_1+x4_1)*sin(x3_1);
     p(3)*x2_1*sin(x3_1) 0];
tol=[tol1_1 tol2_1]';

dq=[x2_1;x4_1];

ddq=inv(D)*(tol-C*dq);

x2(k)=x2_1+ts*ddq(1);
x1(k)=x1_1+ts*x2(k);

x4(k)=x4_1+ts*ddq(2);
x3(k)=x3_1+ts*x4(k);

q1(k)=x1(k);
dq1(k)=x2(k);
e1(k)=q1op(k)-q1(k);
de1(k)=(e1(k)-e1_1)/ts;

q2(k)=x3(k);
dq2(k)=x4(k);
e2(k)=q2op(k)-q2(k);
de2(k)=(e2(k)-e2_1)/ts;

e=[e1(k);e2(k)];
de=[de1(k);de2(k)];

Kp=[1500 0;0 1000];
Kd=[150 0;0 150];

tol=Kp*e+Kd*de;

energy(k)=0.3*abs(tol(1)*dq1(k))+0.7*abs(tol(2)*dq2(k));

    x1_1=x1(k);x2_1=x2(k);
    x3_1=x3(k);x4_1=x4(k);

    e1_1=e1(k);
    e2_1=e2(k);
    q1op_1=q1op(k);dq1op_1=dq1op(k);
    q2op_1=q2op(k);dq2op_1=dq2op(k);
```

```
    tol1_1 = tol(1);
    tol2_1 = tol(2);

    tol1k(k) = tol(1);
    tol2k(k) = tol(2);
end
% ************ 计算总能量 ****************** %
energy_all = 0;
for k = 1:1:G
    energy_all = energy_all + energy(k);
end
d1 = sum(delta1);                                          % 参考轨迹的逼近误差
d2 = sum(delta2);                                          % 参考轨迹的逼近误差
% ******** 计算目标 ******** %
delta_all = 0.5 * d1 + 0.5 * d2;

Object = 0.20 * energy_all + 0.80 * delta_all;             % used for main.m

if flag == 1
    t(1) = 0;
    q10 = 0;q20 = 0;
    for k = 1:1:G                                          % > TE 不包含原点
        t(k) = k * ts;
    if t(k)< TE
        qr1(k) = (q1d - q10) * (t(k)/TE - 1/(2 * pi) * sin(2 * pi * t(k)/TE)) + q10;
                                                           % 不含原点的参考轨迹
        qr2(k) = (q2d - q20) * (t(k)/TE - 1/(2 * pi) * sin(2 * pi * t(k)/TE)) + q20;
                                                           % 不含原点的参考轨迹
    else
        qr1(k) = q1d;
        qr2(k) = q2d;
    end

    end
        figure(1);
        subplot(211);

        plot(t,qr1,'b',t,q1op,'r',t,q1,'k - .','linewidth',2);
        legend('Ideal trajectory','Optimal trajectory', 'Trajectory tracking');
        xlabel('Time (s)');ylabel('First Joint angle tracking');


        subplot(212);
        plot(t,qr2,'b',t,q2op,'r',t,q2,'k - .','linewidth',2);
        legend('Ideal trajectory','Optimal trajectory', 'Trajectory tracking');
        xlabel('Time (s)');ylabel('Second Joint angle tracking');

        figure(2);
        subplot(211);
        plot(t,tol1k,'k','linewidth',2);
        xlabel('Time (s)');ylabel('Control input,tol1');
        subplot(212);
        plot(t,tol2k,'k','linewidth',2);
        xlabel('Time (s)');ylabel('Control input,tol2');
end
end
```

参考文献

[1] R Storn, K Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, 1997,11: 341-59.

[2] M Vasile, E Minisci, M Locatelli, An inflationary differential Evolution algorithm for space trajectory optimization, IEEE Transactions on Evolutionary Computation, 2011,15(2): 267-281.

[3] Das S, Konar A, Chakraborty U K. Two improved differential evolution schemes for faster global search[C],Proceedings of the 2005 conference on Genetic and evolutionary computation. ACM, 2005: 991-998.

[4] Richards N D, Sharma M, Ward D G. A hybrid A * /automaton approach to on-line path planning with obstacle avoidance[C], AIAA 1st Intelligent Systems Technical Conference. 2004, 1: 141-157.

[5] Lee M C, Park M G. Artificial potential field based path planning for mobile robots using a virtual obstacle concept[C], Advanced Intelligent Mechatronics, 2003. AIM 2003. Proceedings. 2003 IEEE/ASME International Conference on. IEEE, 2003, 2: 735-740.

[6] Petros A. Ioannou, Sun Jing. Robust Adaptive Control[M]. PTR Prentice-Hall, 1996, 75-76.