文章编号: 1001-0920 (2015) 11-2048-07 DOI: 10.13195/j.kzyjc.2014.1410

自适应变步长迭代动态规划方法及其在间歇过程优化中的应用

李宏光, 刘骥鹏, 黄静雯

(北京化工大学 信息科学与技术学院，北京 100029)

摘 要: 迭代动态规划(IDP)作为一种求解非线性问题的离散算法, 其寻优精度和收敛速度受到时间段划分的影响. 通常, 时间段划分依赖主观经验, 缺乏科学有效的指导. 针对终端时刻固定的动态优化问题, 提出一种自适应变步长IDP算法, 综合考虑控制变量与目标函数值的变化, 对时间段数量、长度和切换点进行优化. 将该方法应用于间歇过程优化, 结果表明其能够智能分配时间段数量与长度, 可有效提升寻优精度.

关键词: 优化控制；迭代动态规划；自适应变步长；间歇过程

中图分类号: TP273 **文献标志码:** A

Self-adaptive variable-step approach for iterative dynamic programming with applications in batch process optimization

LI Hong-guang, *LIU Ji-peng*, *HUANG Jing-wen*

(School of Information Science and Technology，Beijing University of Chemical Technology，Beijing 100029, China. Correspondent：HUANG Jing-wen，E-mail：huangjwg@mail.buct.edu.cn)

Abstract: As a discrete algorithm to solve nonlinear optimization problems, iterative dynamic programming(IDP) algorithm is rather vulnerable to the stage of time in several aspects such as accuracy as well as the convergence rate. Traditionally, the time division associated with IDP algorithm relies on human's subjective experiences, lacking effective guidance. Motivated by this observation and targeted at fixed terimal time optimizaton problem, a self-adaptive variable-step IDP algorithm is introduced in this paper, which can adjust the number, length and switching point of the time stages taking account of the performance and control variables, in order to improve the performance of IDP. The approach is applied to batch process optimization simulations. The results show that the time stages can be self-adjusted and the optimization performance can be improved.

Keywords: optimizing control；iterative dynamic programming；self-adaptive and variable-step；batch process

0 引 言

动态规划是研究决策过程最优化的一种有效方法. 该方法最初用于时间离散问题, 后来基于哈密顿-雅克比理论推广到连续动态系统[1]. 然而, 由于求解HJB方程过程复杂, 且随着维数的增高导致路径点激增以及插值方法的局限性, 使得动态规划仍多用于低维离散系统.

Luus[2]提出了迭代动态规划(IDP)概念, 采用动态规划的基本思想, 巧妙地从时间与空间两个维度对系统进行离散化、网格化处理, 通过迭代计算缩小搜索范围、提升寻优精度, 最终求取全局最优解. 网格点的使用大幅降低了计算量, 使其可以处理高维非线性问题, 扩展了动态规划方法的适用范围[3].

IDP中的时间段数P对于优化结果具有极其重要的影响[4]. 分段过多会增加计算负担, 降低求解速度; 而分段过少会降低求解精度, 影响对目标函数的优化. 目前, 时间段数P的选取大多依靠人的主观经验, 且每段长度平均分配, 缺乏一种科学、高效的处理方式, 从而严重制约了IDP算法的发展.

一些学者对时间段划分问题进行了一系列研究, 提出了改进的方法. Bojkov等[5]利用变化时间段长度的IDP算法, 成功处理了终端时间不固定条件

收稿日期: 2014-09-12；**修回日期:** 2015-03-12.

基金项目: 中央高校基本科研业务费专项资金项目(YS1404, ZZ1310).

作者简介: 李宏光(1963−), 男, 教授, 博士生导师, 从事过程性能监督与优化、工业过程智能控制等研究；刘骥鹏(1990−), 男, 硕士生, 从事过程控制优化的研究.

下的优化问题; Mekarapiruk 等[6]在 IDP 中加入区域大小灵活变动的方案, 能在一定范围内随机调整步长; Luus[7]采用罚函数的方式选择适当的步长, 保证了终端时刻的约束条件.

上述方法对时间段划分提供了一定参考, 提升了 IDP 算法的寻优精度. 但是, 这些方法并没有对时间段数量进行调整, 同时缺少对化工过程动态特性的分析, 没有考虑目标函数、状态量与控制量的动态变化. 由于随机、无方向地对步长进行选择, 严重降低了寻优效率, 同时大幅提高的迭代次数造成了计算量的激增.

本文针对终端时刻固定的动态优化问题, 提出一种自适应变步长法的时间划分方法. 利用两个新函数分析迭代过程中的目标函数与控制变量的变化, 动态改变时间段的数量、长度和切换点. 将此方法应用于间歇过程的优化, 获得了较好的仿真结果.

1　自适应变步长判定函数

1.1　动态优化问题

终端时间固定的动态优化问题一般可以表示为如下形式:

目标函数

$$J = \max \varphi[(x(t_f), t_f]; \tag{1}$$

约束条件

$$\begin{aligned} &\frac{\mathrm{d}x}{\mathrm{d}t} = f(x,u),\ x(0) = x_0,\\ &c[u(t), x(t)] = 0,\\ &g[u(t), x(t)] \leqslant 0,\\ &a \leqslant u(t) \leqslant b. \end{aligned} \tag{2}$$

其中: $x(t)$ 为状态量, 其初始状态 x_0 已知; $u(t)$ 为控制量; c 为等式约束; g 为不等式约束; a、b 为控制量 $u(t)$ 的界限; t_f 为终端时刻.

IDP 算法并不针对模型本身进行离散化, 而是先利用控制量生成连续的状态量, 再对状态量进行采样, 得到离散的网格点, 从而将连续系统离散处理[8].

因为间歇过程的系统频率极低, 不能依据香农采样定理选择 IDP 步长, 所以 IDP 通常会增加时间段数量, 产生更为精细的网格点, 使采样信号包含更多连续系统的信息, 并使采样系统与连续系统更相近, 从而保证所求得的离散解能有效地用于实际系统. 然而, 单纯缩短步长、增加步数并不一定能提升控制效果. 对于控制变量变动剧烈的非线性系统, 选择最理想的控制变量切换的时间点, 对优化结果同样具有重要影响[9].

针对 IDP 算法中时间采样方法固定不变的弊端, 本文构建 Gap 函数和 Division 函数, 采用自适应变步长的方法优化时间段的分配以提升 IDP 的寻优能力.

1.2　Gap 函数

利用 IDP 算法进行优化求解, 特别是求解控制变量波动剧烈的系统, 控制变量切换点, 即时间采样点对优化结果至关重要. 本节构建 Gap 函数, 优化时间采样点以提升寻优能力.

通常, IDP 算法将整个反应时间平均分为 P 段, 每段时间长度分别为

$$v(k) = t_k - t_{(k-1)},\ k = 1, 2, \cdots, P. \tag{3}$$

在此, 引入一个新的时间变量 τ, 将不同长度的时间段归一化处理, 即

$$\mathrm{d}t = v(k) P \mathrm{d}\tau, \tag{4}$$

$$t_k - t_{-1} = v(k) P (\tau_k - \tau_{k-1}), \tag{5}$$

$$\tau_k - \tau_{k-1} = \frac{1}{P}, \tag{6}$$

$$\frac{\mathrm{d}x}{\mathrm{d}t} = v(k) P f(x,u). \tag{7}$$

设在第 j 次迭代后, 得到的时间向量为 $v^j(i) = [v^j(1), v^j(2), \cdots, v^j(P)]$, 求得的控制向量为 $u^j(i) = [u^j(1), u^j(2), \cdots, u^j(P)]$.

定义 1　Gap 函数为

$$G(i) = \frac{\Delta u^j(i)}{\Delta u^j_{\max}},\ i = 1, 2, \cdots, P-1. \tag{8}$$

其中

$$u^j_{\max} = \max_{m,n=1,2,\cdots,P} |u^j(m) - u^j(n)|, \tag{9}$$

$$u^j(i) = |u^j(i+1) - u^j(i)|,\ i = 1, 2, \cdots, P-1. \tag{10}$$

Gap 函数从控制变量的角度出发, 根据相邻控制变量的差值大小与本次迭代结果中所有控制变量间最大差值之比, 自动选取大波动的控制变量切换点, 进而对切换点进行优化.

设 Gap 的阈值为 ω. 若第 k 个采样点处, 有 $G(k) \geqslant \omega$, 则对第 k 个采样点重新寻优. 保持 $v^j(k)$ 与 $v^j(k+1)$ 之和不变, 随机产生 b 个采样点, 即 b 组 $[v^{j\prime}(k), v^{j\prime}(k+1)]$.

从 t_k 时刻的网格点出发, 保持 $u^j(k)$ 与 $u^j(k+1)$ 之和不变, 将生成的 b 组时间段代入方程 (2), 计算目标函数 φ, 有

$$\begin{aligned} &\varphi[x^j(t_{k+1}), t_{k+1}] =\\ &\int_{t_{k-1}}^{t_{k-1}+v(k)} f[x^j(t_k), u^j(k)]\mathrm{d}t +\\ &\int_{t_k}^{t_k+v(k+1)} f[x^{j\prime}, u^j(k+1)]\mathrm{d}t, \end{aligned} \tag{11}$$

$$x^{j\prime}(t_k) = \int_{t_{k-1}}^{t_{k-1}+v(k)} f[x^j(t_k), u^j(k)]\mathrm{d}t. \tag{12}$$

选择最优目标函数值对应的切换点, 得到新的时

间段序列 $v^j(i)=[v^j(1),\cdots,v^{j'}(k),v^{j'}(k+1),v^j(P)]$.

1.3 Division 函数

虽然IDP算法不能直接使用香农采样定理, 但仍可依照其思想: 采样间隔越密集, 获得的控制过程信息越多, 控制效果也越好.

以第 j 次迭代为例, 以平方的形式表征控制变量间的波动程度, 记为 $\Delta u^{j'}$, 有

$$\Delta u^{j'}(i)=\begin{cases}[u^j(i)-u^j(i+1)]^2,\ i=1;\\ [u^j(i)-u^j(i+1)]^2+[u^j(i)-u^j(i-1)]^2,\\ \qquad i\in[2,P-1];\\ [u^j(i)-u^j(i-1)]^2,\ i=P.\end{cases}\tag{13}$$

将控制变量 $u^j(i)=[u^j(1),u^j(2),\cdots,u^j(P)]$ 代入方程(2), 得到每个时间采样点的目标值 $J^j(i)$, 计算相邻目标函数值差值的平方, 记为 $\Delta J^j(i)$, 即

$$\Delta J^j(i)=[J^j(i)-J^j(i+1)]^2,\ i=1,2,\cdots,P-1.\tag{14}$$

定义 2 Division 函数为

$$D(i)=\alpha\frac{\Delta u^{j'}(i)}{\sum\limits_{k=1}^{P}\Delta u^{j'}(k)}+\beta\frac{\Delta J^j(i)}{\sum\limits_{k=1}^{P}\Delta J^j(k)}.\tag{15}$$

由于控制变量与目标函数对采样时间优化影响不同, 引入式(15)中 α、β, 分别为目标函数与控制变量的权重. 对 $D(i)$ 进行归一化处理, 令 $\alpha+\beta=1$, 则有 $\sum\limits_{i=1}^{P}D(i)=1$.

Division 函数针对采样间隔问题, 根据控制变量与目标函数在单一时间段内部的变化, 自动筛选出需要更密集采样的时间段, 并细分该时间段以提升控制变量的精确度, 产生最优的采样密度.

设 $D(i)$ 的阈值为 λ. 当 $D(i)\geqslant\lambda$ 时, 即认为在第 i 个时间段内, 控制变量的波动较大且目标函数灵敏度高. 因此, 保持控制变量不变, 将时间段分为两段, 以便下次迭代求得更精确的控制策略.

对于 λ 的设计, λ 过小会导致时间段激增, 而 λ 过大并不能有效地划分时间段. 经作者大量仿真实例发现, 一般而言, λ 的经验值在0.1附近时, 得到的目标函数值较为理想.

2 自适应变步长IDP算法

将上述函数加入原IDP算法中, 即可得到自适应变步长IDP算法的步骤.

1) 设初始时间段数为 P, 将时间区间 $[0,t_f]$ 平均分为 P 段. 选择 N 个控制变量用于生成网格点, 对于每个可行的控制变量设定 M 个可行值. 设定Gap函数和Division函数的阈值 ω 和 λ.

2) 给每个时间段一个初始控制变量值 u_0, 其余 $N-1$ 个控制变量在 $[u_0-r^j,u_0+r^j]$ 内随机生成.

3) 将第2)步生成的 N 组控制向量分别代入状态方程(2), 得到 N 组 x 的状态轨迹, 即 $P\times N$ 个的网格点.

4) 从最后一个时间段 P 开始, 其对应的时间区间是 $[t_f-v^j(P),t_f]$, 对于 $t_f-v^j(P)$ 时刻上的每个网格点, 用 M 个可行的控制变量代入状态方程(2). M 个可行控制变量通过下式计算得到:

$$u^j(P-1)=u^{j-1*}(P-1)+\eta r^j.\tag{16}$$

其中: u^{j-1*} 是上次迭代得到的控制量(第1次迭代使用初始值), η 是 $[-1,1]$ 间的随机数. 从 M 个目标值中选择使目标函数值最优的控制变量, 储存控制变量值以便下一步使用.

5) 退至前一个时间段 $P-1$, 对应的时间区间为 $[t_f-v^j(P)-v^j(P-1),t_f-v^j(P)]$, 每个网格点用 M 个可行控制变量值再代入状态方程(2), 得到新的网格点. 选择与新网格点距离最近的上一步网格点中储存的控制变量值, 继续迭代, 最终得到 M 个目标函数值. 储存目标函数值最优的网格点对应的控制变量值.

6) 对 $P-2,P-3$ 直到初始时间段, 重复上述过程. 对于得到的控制变量向量进行正向搜索, 寻找使目标函数值最优的一组, 记为 $u^j(i)=[u^j(1),u^j(2),\cdots,u^j(P)]$.

7) 利用Gap函数和Division函数, 对每个时间段的切换点、数量和长度进行优化, 重新生成时间向量和控制向量.

8) 利用缩小因子 γ 减小控制变量的搜索范围, 即

$$r^{j+1}=\gamma*r^j.\tag{17}$$

增加迭代次数到 $j+1$, 然后回到第2)步进行下一次迭代, 直到完成规定的迭代次数.

值得注意的是, 适当进行若干次第7)步的判定便可以得到良好的时间段优化效果. 特别在初始几次的迭代后, 求得的控制变量并不精确, 此时进行上述两种判定不但意义不大, 反而可能导致错误的时间段划分, 降低算法的寻优能力. 因此, 本文实例中, 算法的第7)步只需在迭代次数 i 为偶数且 $i>5$ 的迭代求解过程中进行.

3 自适应变步长IDP算法收敛性证明

假设 1 U 是一个在子空间 $\boldsymbol{R}^{N*n_u}$ 上的紧子集.

假设 2 $J_0(u)$ 是一个在 U 上的凸函数.

假设 3 $L\{u^{[j]*}\}=\{u\in U|J_0(u)\leqslant J_0(u^{[j]*})\}$ 是有限且紧的水平集.

定理 1 自适应变步长IDP算法的每次求解过

程, 都是在子空间 $U \in \boldsymbol{R}^{N*n_u}$ 内的有序搜索 $\{U(1), U(2), \cdots, U(N)\} \subseteq U \in \boldsymbol{R}^{N*n_u}$, 且 $\{U(1), \cdots, U(k-1), U(k), \cdots, U(N)\}$ 在对应时间段内是分段常量[1,9].

定理 2　在求解的序列 $\{u^{[j]*} \in U | j = 1, 2, \cdots\}$ 中, 点集 $u^* \in \{\arg\min_{u\in U} J_0(U)\}$ 是自适应变步长 IDP 算法中的最优解且是稳定点.

证明　每次迭代过程会产生 M 个可行值, 有

$$\hat{u}^{[q]} = u^{[j+1]*} + w^{[j,q]}(k),\ q = 1, 2, \cdots, M-1. \quad (18)$$

迭代过程中控制变量的选择可表示为

$$u^{[j+1]*} = u^{[j]*} + w^{[j]} = u^{[j]*} + \sum_{k=0}^{N-1} w^{[j]}(k) = u^{[0]*} + \sum_{i=1}^{j}\sum_{k=0}^{N-1} w^{[l]}(k). \quad (19)$$

其中: $w^{[j]}(k)$ 是在 $r^{[j]}$ 内的随机向量.

$$u^{[j+1]*}(k) = \begin{cases} u^{[j+1]*}(k),\ J_0(\hat{u}^{[q]}) \geqslant J_0(u^{[j+1]*}); \\ \arg\min J_0(u),\ J_0(\hat{u}^{[q]}) < J_0(u^{[j+1]*}). \end{cases}$$

如果 $u^{[j]*} = u^* \in \{\arg\min_{u\in U} J_0(u)\}$, 因 $J_0(\hat{u}^{[q]}) \geqslant J_0(u^{[j+1]*})$ 成立, 则对于任意 q, 有 $u^{[j+1]*}(k) = u^{[j]*}(k) = u^*(k)$. 因此, 对于所有 k, 可以证明 $u^{[j+1]*}(k) = u^*(k)$. □

定理 3　自适应变步长 IDP 算法产生的序列是收敛的.

证明　由上文已知, $w^{[j]} = \sum_{k=1}^{N} w_k^{[j]}$, $|w_{k,D}^{[j]}| \leqslant \gamma^{j-1} r$, 其中 $D = 1, 2, \cdots, n_u$. 因此, 由三角不等式 $|\alpha + \beta| \leqslant |\alpha| + |\beta|$, 有

$$\|w^{[j]}\|_2 = \left\|\sum_{k=1}^{N}\sum_{D=1}^{n_u} w_{k,D}^{[j]}\right\|_2 \leqslant \|\gamma^{j-1} r\|_2, \quad (20)$$

从而可得 $u^{[j+1]*} - u^{[j]*} = w^{[j]}$. 同时有

$$\lim_{j\to\infty} \|u^{[j+1]*} - u^{[j]*}\|_2 = \lim_{j\to\infty} \|w^{[j]}\|_2 \leqslant \lim_{j\to\infty} \|\gamma^{j-1} r\|_2 = \|r\|_2 \lim_{j\to\infty} \gamma^{j-1} = \|r\|_2 \times 0 = 0 \Rightarrow \lim_{j\to\infty} \|u^{[j+1]*} - u^{[j]*}\|_2 = 0. \quad (21)$$

因此, $\lim_{j\to\infty} u^{[j+1]*} = u^{[j]*} = u^{[\infty]*}$. 由于已知

$$\begin{cases} u^{[\infty]*} \in L\{u^{[0]*}\}, \\ L\{u^{[0]*}\} \text{ 在假设中是有限的}, \end{cases}$$

由自适应变步长 IDP 算法产生的序列是收敛的. □

推论 1　IDP 最大的收敛半径是

$$\|u^{[\infty]*} - u^{[0]*}\|_2 \leqslant \|r\| \times \frac{\gamma}{1-\gamma}.$$

推论 2　对于充分大的 γ (即 $\gamma \to 1$) 且 $r \neq 0$, IDP 的搜索范围是整个可行域.

以上两个推论的证明较简单, 本文不再赘述.

定理 4　对于充分大的 γ (即 $\gamma \to 1$) 且 $r \neq 0$, 自适应变步长 IDP 的解 $u^{[j]*} \in U\ (j = 1, 2, \cdots)$ 收敛到 u^* 的概率是 1.

证明　记

$$L_K\{u^{[j]*}\} = \{u(k) \in U(k) | J_0(u^{[j]*}|_{u^{[j]*}(k)=u(k)}) \leqslant J_0(u^{[j]*})\},$$

并且 $L_k\{u^{[j+1]*}\}$ 在边界有一个无穷小的误差, 有

$$P_r(u^{[j+1]*}(k) \neq u^{[j]*}(k)) = P_r(u^{[j+1]*} + w^{[j,q]}(k) \in L_k\{u^{[j+1]*}\}) = \frac{\mathrm{Vol}(L_k\{u^{[j+1]*}\})}{\prod_{D=1}^{n_u}(2\gamma^{j-1} r_{k,D})}. \quad (22)$$

其中: $\mathrm{Vol}(\cdot)$ 在 $n_u = 1, n_u = 2, n_u \geqslant 3$ 下分别代表长度、面积与体积.

$J_0(u)$ 在 U 上是严格凸的, $u^{[j+1]*}$ 必然会落在 $L\{u^{[j]*}\}$ 的边界上, 因此

$$0 \leqslant \mathrm{Vol}(L\{u^{[j+1]*}\}) \leqslant \prod_{D=1}^{n_u}(\gamma^{j-1} r_{k,D}). \quad (23)$$

如果 $u^{[j+1]*} \neq u^*$, 则有

$$0 < \Pr(u^{[j+1]*}(k) \neq u^{[j]*}(k)) < 0.5 \Rightarrow 0.5 < \Pr(u^{[j+1]*}(k) = u^{[j]*}(k)) < 1 \Rightarrow 0.5 < \Pr(u^{[j+1]'}(k) = u^{[j]}(k)) < 1 \Rightarrow \Pr(u^{[j+1]*'} = u^{[j]*}) = \Pr(u^{[j+1]*'} = u^{[j]*}(k)) < 1 \Rightarrow \lim_{j\to\infty} \Pr(u^{[j+1]*'} = u^{[j]*}) = 0 \Rightarrow \lim_{j\to\infty} \Pr(J(u^{[j]*'} < J(u^{[j]*}) = 1. \quad (24)$$

u^* 存在且唯一, 在反复迭代之后, $\gamma \to 1$ 时会有唯一的稳定点 $u^{*'}$, 因此

$$\lim_{j\to\infty} \Pr(u^{[j]*'} = u^{[j]*}) = 1. \quad (25)$$

4　间歇过程动态优化实例

4.1　连续间歇过程实例

在一个连续间歇反应中进行 $A \to B \to C$ 的化学反应. 其中: A 是反应原料, B 是目标产物, C 是副产品. 本例要求控制反应温度, 使 B 的产量在终端时刻达到最大值. 该优化问题可描述如下:

目标函数

$$\max_{u(t)} J = x_2(t_f); \quad (26)$$

约束条件

$$\dot{x_1} = -4\,000 \exp(-2\,500/T) x_1^2,$$

$$\dot{x_2} = 4\,000\exp(-2\,500/T)x_1^2 - 620\,000\exp(-5\,000/T)x_2,$$
$$x_1(0) = 1,\ x_2(0) = 0,$$
$$298 \leqslant T \leqslant 398,\ t_f = 1. \tag{27}$$

其中: x_1、x_2 分别为 A、B 的浓度分率 $(\mathrm{mol}\cdot L^{-1})$, 其初始值分别为 1 和 0; T 为反应温度 (K); t_f 为反应时间长度.

对于这个间歇反应优化问题, 许多学者采用不同方法对其进行了研究. Renfro 等[10]采用同步优化策略求得的最优值是 0.610; Rajesh 等[11]利用蚁群结构优化方法进行研究, 将最优值提升至 0.610 45; 刘兴高等[12]利用传统的 IDP 算法得到最优值为 0.610 632; Logsdon 等[13]采用 SQP 法求解该问题, 得到的最优值为 0.610 775.

采用原 IDP 算法, 设状态变量离散点数 $N = 5$, 控制变量数 $M = 5$, 控制变量初始值 $T = 348$ K, 初始搜索范围 $r = 50$ K, 迭代次数为 15 次. 设时间段长度平均分配, 段数分别为 $P = 5$, $P = 10$, $P = 20$ 和 $P = 25$, 求得目标函数值见表 1. 最优控制变量轨迹如图 1 所示. 可以看出, 随着时间段数 P 的增多, 求得的控制变量更为精细, 目标函数值的精度更高.

表 1 不同时间段数量 P 下的目标函数值 J

P	5	10	20	25
J	0.609 400	0.610 070	0.610 453	0.610 535

利用自适应变步长 IDP 算法再对本例进行求解. 设 $P = 10$ 为初始时间段数量, 保持 M、N 与 r 等参数值不变, 对 Division 函数 λ 的阈值和权重 α 选取不同数值, 求得的目标函数值如表 2 所示. 可见, 当 $\lambda = 0.1, \alpha = 0.9$ 时, J 有最大值 0.610 786.

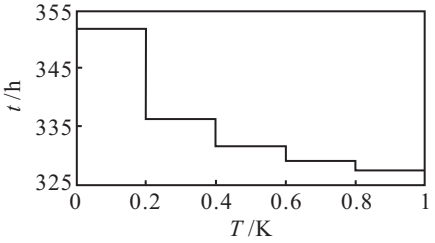
由于 P 值在迭代过程中动态变化, Gap 函数的阈值 ω 也随 P 值而变化, 设阈值 ω 为

$$\omega = h/P,\ P = 10, 11, 12\cdots, \tag{28}$$

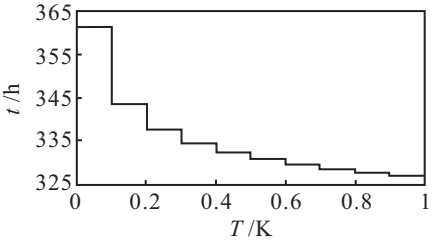
其中 h 表征阈值的高低.

为了求得更为精确的控制变量, h 的取值应较小. 将 h 的取值区间设为 $[2, 5]$ 间的整数, 表 3 给出了不同 h 值下的目标函数值. 可见, 当 $h = 4$ 时, 目标函数值最优. 因此, 选择 $h = 4$, 经过 15 次迭代后, P 由 10 变为 24, 求得 $J = 0.610\,786$. 该结果优于上文 $P = 25$ 条件下的 0.610 535, 也优于刘兴高等[12]用原 IDP 算法求得的 0.610 632 和 Logsdon 等[13]用 SQP 法求得的 0.610 775.

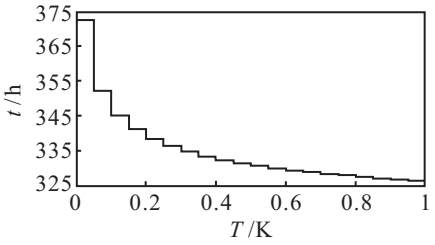
加入 Gap 函数后的目标函数值 $J = 0.610\,786\,4$ 与只考虑 Division 函数的目标函数值 $J = 0.610\,786\,2$ 相比, 只有极微小的提升, 得到的最优控制曲线如图 2 所示.



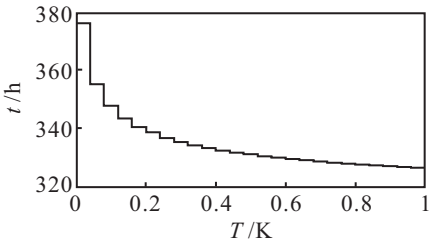
(a) P=5



(b) P=10



(c) P=20



(d) P=25

图 1 不同 P 值下的最优控制轨迹

表 2 不同参数下的目标函数值 J

α	λ		
	0.1	0.2	0.3
0.9	0.610 786	0.610 763	0.610 679
0.8	0.610 785	0.610 757	0.610 631
0.7	0.610 784	0.610 757	0.610 615

表 3 不同 h 值下目标函数值 J

h	2	3	4	5
J	0.610 781	0.610 784	0.610 786	0.610 786

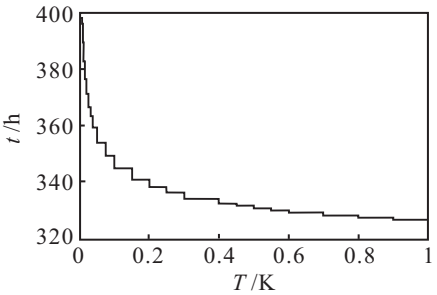


图 2 自适应变步长 IDP 算法下的最优控制轨迹

由于本例中控制量 u 相对平稳, 除了 $t \in [0, 0.2]$ 时变动较大, 其他时刻趋于稳定, 引入 Gap 函数的效果并不明显. 但是, 下一个实例将充分证明 Gap 函数在控制变量波动大的情况下, 能有效地优化时间切换点, 提升算法的寻优能力.

4.2　半间歇过程实例

采用 Park 等[14]提出的半间歇生化反应器模型, 其原料随反应的进行逐渐添加, 但并不伴随有产品的产出. 该过程优化问题描述如下:

目标函数

$$J = \max x_1(t_f)x_5(t_f); \tag{29}$$

约束条件

$$\begin{aligned}
&\frac{\mathrm{d}x_1}{\mathrm{d}t} = g_1(x_2 - x_1) - \frac{u}{x_5}x_1,\\
&\frac{\mathrm{d}x_2}{\mathrm{d}t} = g_2x_3 - \frac{u}{x_5}x_2,\\
&\frac{\mathrm{d}x_3}{\mathrm{d}t} = g_3x_3 - \frac{u}{x_5}x_3,\\
&\frac{\mathrm{d}x_4}{\mathrm{d}t} = -7.3g_3x_3 + \frac{u}{x_5}x_2(20 - x_4),\\
&\frac{\mathrm{d}x_5}{\mathrm{d}t} = u,\\
&g_3 = \frac{21.78x_4}{(x_4 + 0.4)(x_4 + 62.5)},\\
&g_2 = \frac{x_4\mathrm{e}^{-5x_4}}{0.1 + x_4},\\
&g_1 = \frac{4.75g_3}{0.12 + g_3},\\
&x(0) = [0, 0, 1, 5, 1]^{\mathrm{T}}.
\end{aligned} \tag{30}$$

其中: x_1 为环境中细胞分泌的 SCU-s2 的浓度 (gL^{-1}), x_2 为环境中 SCU-s2 的浓度 (gL^{-1}), x_3 为细胞浓度水平 (gL^{-1}), x_4 为葡萄糖浓度水平 (gL^{-1}), x_5 为反应液的总体积 (L); 反应的总时长 $t_f = 15$ h; 反应液的滴加速度 u 的约束为

$$0 \leqslant u \leqslant 2; \tag{31}$$

J 代表最终时刻 SUC2-s2 的产量. 本例要求控制加料速度, 使终端时刻 SUC2-s2 的产量最大.

采用自适应变步长 IDP 算法, 设 $P = 10$, $M = 5$, $N = 5$, Division 函数阈值取 $\lambda = 0.1$, 暂不引入 Gap 函数. 迭代 20 次后, 得到 $P = 22$, $J = 32.426\,9$, 控制变量的轨迹如图 3 所示.

由图 3 可见, $u(t)$ 的波动剧烈, 此时引入 Gap 函数对控制变量切换点进行优化应有显著效果.

令 $\omega = h/P$ 中 $h = 3$. 迭代 20 次, 得到 $J = 32.622\,2$, 该结果优于未引入 Gap 函数的 32.426 9, 也略优于 Luus 等[15]利用 $P = 31$ 等长度时间段得到的 32.613 4.

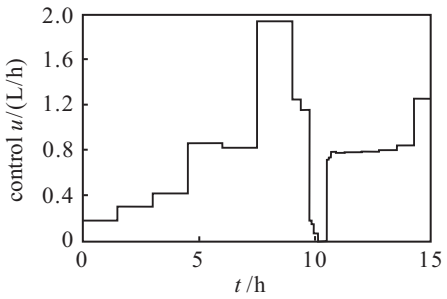


图 3　引入 Division 函数的最优控制轨迹

为了进一步提升寻优精度, 针对这个多变量的复杂过程, 对 J、u、$x_1 \sim x_5$ 等相关变量进行分析.

如图 4 所示, 当 $t < 9.54$ 时, $x_1 \approx 0$, $J \approx 0$, 此时, 以 $J = \max x_1(t_f)x_5(t_f)$ 为依据进行步长优化没有实际意义. 因此, 考虑在 $t < 9.54$ 区间内, 重新建立评价函数.

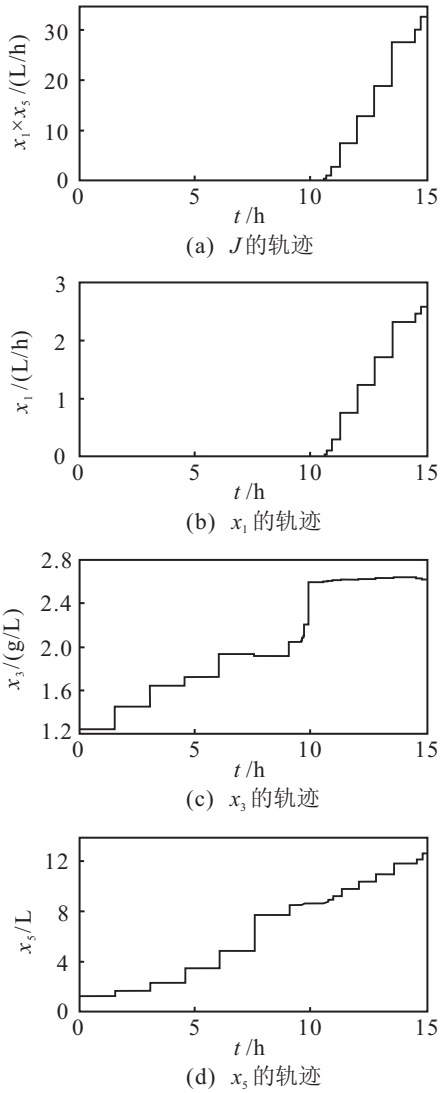


图 4　J, x_1, x_3, x_5 的轨迹

x_3 代表细胞浓度的变化, 在 x_3 达到峰值后, 即 $t = 9.54$ 后, 目标产物 x_1 才开始大量产生. 因此, 针对细胞总量建立新的评价函数

$$J_2 = x_3(t)x_5(t),\ t \leqslant 9.54. \tag{32}$$

以$t = 9.54$为界, 前后分别采用J_2和J作为Division函数的判别依据. 重新迭代运算20次, 求得$J = 32.626\,8$.

由于本例的控制变量波动明显, 考虑适当缩小λ值以增加时间段数量, 使求得的控制变量能更精确地拟合理想控制变量的轨迹.

$\lambda = 0.07$时, 经25次迭代得到$J = 32.671\,5$, 该结果优于Luus[9]采用基于罚函数法变步长IDP算法迭代2 000次求得的32.669 1 , 控制轨迹如图5所示.

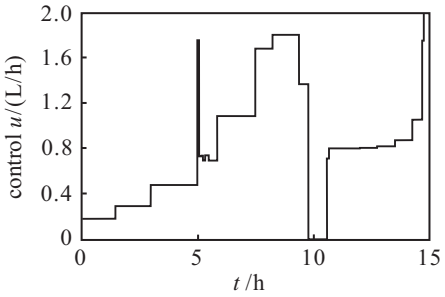


图5 改变λ后的控制变量轨迹

5 结 论

本文通过对IDP算法的时间分段理论进行分析, 提出了一种自适应变步长的IDP算法. 建立了两个自适应变步长的判定函数, 并给出了新算法的步骤. 将其应用于两个经典间歇过程, 实例仿真结果表明, 本文算法能够实现自适应的分段功能, 有效提升了算法对目标函数的求解精度.

参考文献(References)

[1] Aiying Rong, JoséRui Figueira. Dynamic programming algorithms for the bi-objective integer knapsack problem[J]. European J of Operational Research, 2014, 236(1): 85-99.

[2] Luus R. Optimal control by dynamic programming using systematic reduction in grid size[J]. Int J of Control, 1990, 51(5): 995-1013.

[3] Adrian M Thompson, William R Cluett. Stochastic iterative dynamic programming: A Monte Carlo approach to dual control[J]. Automatica, 2005, 41(5): 767-778.

[4] 李前兴. 工业过程迭代动态规划算法研究[D]. 浙江大学控制科学与工程系, 2011.
(Li Q X. Research of iterative dynamic programming for industrial process[D]. Department of Control Science and Engineering, Zhejiang University, 2011.)

[5] Bojkov B, Luus R. Optimal control of nonlinear systems with unspecified final times[J]. Chemical Engineering Science, 1996, 51(6): 905-919.

[6] Mekarapiruk W, Luus R. Iterative dynamic programming with adaptive scheme for region size determination[J]. Hungarian J of Industrial Chemistry, 1999, 27(3): 235-240.

[7] Luus R. Parametrization in nonlinear optimal control problems[J]. Optimization, 2006, 55(1/2): 65-89.

[8] Min Ho Chang, Young Cheol Park, Tai-yong Lee. Iterative dynamic programming of optimal control problem using a new global optimization technique[J]. Computer Aided Chemical Engineering, 2003(3): 416-421.

[9] Luus R. Iterative dynamic programming[M]. Floriad: CRC Press, 2000.

[10] Renfro J G, Morshedi A M, Asbjornsen O A. Simultaneous optimization and solution of systems described by differential/algebraic equations[J]. Computers & Chemical Engineering, 1987, 11(5): 503-517.

[11] Rajesh J, Gupta K, Kusumakar H S, et al. Dynamic optimization of chemical processes using ant colony framework[J]. Computers & Chemistry, 2001, 25(6): 583-595.

[12] 刘兴高, 吴高辉. 一种间歇反应过程迭代动态规划方法[J]. 计算机与应用化学, 2009, 25(7): 792-794.
(Liu X G, Wu G H. An approach to iterative dynamic programming for batch reactor systems[J]. Computer and Applied Chemistry, 2009, 25(7): 792-794.)

[13] Logsdon J S, Biegler L T. A relaxed reduced space SQP strategy for dynamic optimization problems[J]. Computers & Chemical Engineering, 1993, 17(4): 367-372.

[14] Park S, Fred Ramirez W. Optimal production of secreted protein in fed-batch reactors[J]. AIChE Journal, 1988, 34(9): 1550-1558.

[15] Luus R, Hennessy D. Optimization of fed-batch reactors by the Luus-Jaakola optimization procedure[J]. Industrial & Engineering Chemistry Research, 1999, 38(5): 1948-1955.

(责任编辑：李君玲)