文章编号:1000-2243(2007)01-0064-06

一阶惯性延时系统的免分析建模法

—— MATLAB在大林算法建模仿真中的应用

郑剑翔

(福州大学自动化研究所,福建 福州 350002)

摘要:利用MATLAB软件对一阶惯性延时系统免分析建模.通过对实际被控对象施加阶跃激励可获得实际系统的阶跃响应数据,利用MATLAB软件的"fminsearch"命令可方便地求得与实际数据有最小均方差的响应曲线数学表达式,通过变换运算得到被控对象的传递函数$G_p(Z)$.根据大林(Dahlin)控制算法理论,先设定期望的闭环系统函数$\Phi(Z)$,由此二函数便可得到控制器的函数$G_c(Z)$.利用MATLAB中的SIMULINK工具建立仿真模型,通过对此模型的仿真运行,便可得到控制器的最佳参数.

关键词:大林控制算法;建模;仿真;MATLAB

中图分类号:TP271.2 **文献标识码**:A

A method of modeling for first-order inertia delay system

——the application of MATLAB in simulation for algorithm of Dahlin controller

ZHENG Jian-xiang

(Institute of Automation, Fuzhou University, Fuzhou, Fujian 350002, China)

Abstract: This article introduces a simple and precise way to establish a first-order inertia delay system's simulation model without quantitative analysis. We can get the data of step response of the system by applying a step stimulate signal to the system and sampling its response. Then we use MATLAB software to get a mathematic expression which has least mean square deviation to original data and further to derive system's Z function $G_p(Z)$. According to Dahlin control algorithm, we can determine a expecting close loop system function $\Phi(Z)$ and then the controller function $G_c(Z)$ can derive from them. Using SIMULINK tool the simulation model is easy established for these function. After runing the simulation, we can get the best parameters of the controller.

Keywords: Dahlin algorithm controller; modeling; emulating; MATLAB

1 概述

一阶惯性延时系统是实际工业控制中常见的控制问题.使用大林(Dahlin)控制算法解决此类问题比较理想[1].该算法可简单表达为:如图1所示的单回路控制系统,$G_p(Z)$为被控对象传递函数,$G_c(Z)$为数字控制器函数,则闭环系统的传递函数$\Phi(Z)$可用式(1)表示.

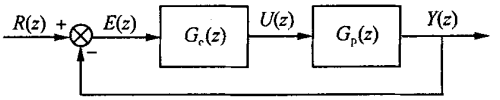

图1 单回路控制系统框图

Fig. 1 Block-diagram of a close-loop control system

$$\Phi(Z)=\frac{Y(Z)}{R(Z)}=\frac{G_c(Z)G_p(Z)}{1+G_c(Z)G_p(Z)} \tag{1}$$

收稿日期:2006-04-10

作者简介:郑剑翔(1955-),男,助理研究员.

从式(1)中可解出数字控制器 $G_c(Z)$，如式(2)所示.

$$G_c(Z) = \frac{U(Z)}{E(Z)} = \frac{1}{G_p(Z)} \cdot \frac{\Phi(Z)}{1+\Phi(Z)} \tag{2}$$

如果已知被控对象的传递函数，并确定闭环系统的函数，则可得到控制器函数 $G_c(Z)$. 由此可知，该算法的特点是根据被控对象的函数并设定期望的闭环响应(其特性为一阶惯性加纯延时)，然后由此得到满足这种响应的控制器.

对于被控对象的传递函数，按常规方法，需要对研究对象进行物理定量的数学分析，建立微分方程或状态方程，得到描述系统的数学模型，才能进行计算机仿真. 但在实际问题中，系统的惯性和延时特性由许多复杂的因素组成，难以进行定量分析. 从系统分析理论知道，系统的冲击响应的拉氏变换式即为系统的传递函数. 实际操作中获得冲击响应难以实现，而阶跃响应却比较容易得到. 从理论分析知道，系统的阶跃响应为系统的冲击响应与阶跃信号的卷积，经过拉氏变换后就是乘积的关系. 即：

$$F(S) = H(S) \cdot U(S) = H(S) \cdot (1/S) \tag{3}$$

式中：$F(S)$ 为系统的阶跃响应的 S 函数；$H(S)$ 为系统的传递函数；$U(S)$ 为阶跃信号的拉氏变换. 如果知道了阶跃响应 $F(S)$，从式(3)很容易解得系统的传递函数：$H(S) = F(S) \cdot S$. 利用 MATLAB 软件的连续到离散系统的转换命令“c2d”，可方便地把 S 函数 $H(S)$ 转换为 Z 函数 $H(Z)$. 式(2)中的 $G_p(Z)$ 就是被控对象的传递函数. 有了准确描述被控对象的传递函数 $G_p(Z)$ 并设定期望的闭环系统函数，从公式(2)可得到控制器函数 $G_c(Z)$. 有了数学模型，就可以利用 MATLAB 软件中的 SIMULINK 工具建立仿真模型，在计算机上对控制过程做精确的仿真调试，由此可减少在实际机器上的调试操作. 这点对某些问题是非常重要的. 将该方法应用于测厚和液压的无卡轴旋切机[2]的仿真实验，结果较为理想.

2　阶跃响应的获得

从图 1 可知，只要令 $R(Z)$ 不变，控制器 $G_c(Z)$ 输出一个阶跃量，便可在控制器输入端得到被控对象 $G_p(Z)$ 的阶跃响应. 专门编一段程序完成阶跃响应的采样记录和数据传送，便可得到 $G_p(Z)$ 的阶跃响应数据并通过通信接口送给 PC 计算机. 在编写采样记录程序时，需要注意采样周期和输出幅度的选取和记录. 对于旋切机例子，设输出阶跃幅度(送给 D/A 转换器的数值的变化量)为 5. 此数值经 D/A 转换变为控制液压电磁阀电压阶跃，进而引起进刀速度的变化，最后在厚度检测传感器中得到反映切割厚度电压信号变化. 此电压经 A/D 转换后就是所需要的被控对象的阶跃响应采样值. 采样周期为 0.01 s，把各采样点的时间值和幅度值分别以数组名为“xdata”和“ydata”输入到 MATLAB 工作空间. 其输入的方法有多种. 最简单的方法是，在采样送给 PC 计算机时将它保存为扩展名为“dat”的数据文件，如“ydata.dat”，数据间可以是逗号或空格相隔，也可为竖列排列. 然后将其拷贝入 MATLAB 的工作目录“work”文件夹内. 再用“load ydata.dat”命令就可将它载入 MATLAB 工作空间内存. 时间轴可用命令“xdata = 0:0.01:0.5”命令产生. 它们都是长度为 51 的一维数组. 图 2 中的“*”点线就是采样获得的原始数据曲线.

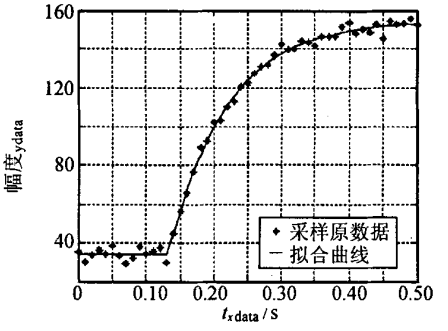

图 2　曲线拟合结果图

Fig. 2　Result of fitcurve

3 传递函数的导出

一阶惯性延时系统的阶跃响应为延时的指数上升曲线. 可用下式表示:

$$Y(t) = A \cdot (1 - \exp(-\alpha(t - t_0))) \cdot U(t - t_0) \tag{4}$$

其中: A 为幅度; α 指数系数; t_0 为延时常数.

现在的问题是如何找到合适的 A、α、t_0, 使得表达式(4)与采样获得的曲线相吻合. 只要曲线能吻合, 我们就获得了被控对象的阶跃响应的数学表达式, 经过拉氏变换有: $Y(S) = A \cdot \alpha \cdot \exp(-t_0 \cdot S)/(S \cdot (S + \alpha))$, 其传递函数则为:

$$H(S) = Y(S) \cdot S = A \cdot \exp(-t_0 \cdot S)/(t_r \cdot S + 1)$$

其中, $t_r = 1/\alpha$ 为上升时间常数.

这是一个曲线拟合的问题. 用 MATLAB 中多变量函数查找最小值的函数命令"fminsearch"可很方便地解决此问题. 在实际问题中, 用式(4)拟合图2的曲线还存在一个问题, 图2可以看到在起始延时阶段不为零, 有一个直流分量. 这是A/D转换所必须的, 但这个量与系统的动态特性无关. 为了求得曲线的精确拟合, 也必须给式(4)增加一个直流分量 d_0. 因此, 以 A、t_r、t_0、d_0 为自变量, 用"fminsearch"命令求得具有与实际曲线最小均方差的估计曲线, 也就得到了所求的数学表达式.

为此建立曲线拟合函数的 m 文件"fitcurve.m":

```
function [estimates, model] = fitcurve(xdata, ydata, start_point)
model = @expfun;                                    % 建立函数句柄
estimates = fminsearch(model, start_point);         % 调用求最小值函数
  function [sse, FittedCurve] = expfun(params)      % 定义 EXPFUN 子函数
      A = params(1);                                % A 参数赋值
      t_r = params(2);                              % t_r 参数赋值
      t_0 = params(3);                              % t_0 参数赋值
      d_0 = params(4);                              % d_0 参数赋值
      U = zeros(size(xdata - t_0));
      U((xdata - t_0) >=0) = 1 ;                    % 产生延时了 t_0 的 U(t) 序列
      FittedCurve = (1 - exp( -(xdata - t_0)./t_r)) * A ;
      FittedCurve = FittedCurve. * U + d_0;         % 求曲线值 = 式(4) + d_0
      ErrorVector = FittedCurve - ydata;            % 求误差
      sse = sum(ErrorVector .^ 2);                  % 求误差平方和
    end
end
```

以上程序中, 子函数 expfun 就是用确定的参数 A、t_r、t_0、d_0, 按照式(4)加直流 d_0 规则计算曲线值, 然后求出与已知数据误差的平方和. 子函数被"fminsearch"命令所调用. 为了使求最小值过程能最后收敛于正确值, 必须给 A、t_r、t_0、d_0 4参数一个估计值, 将此估计值赋予数组"start_point". 从图2很容易对此4个参数做出估计 $A = 150$, $t_r = 0.1$, $t_0 = 0.13$, $d_0 = 30$, t_r 值的估计是阶跃上升60%处的时刻减阶跃开始上升的时刻. 这些估计值不必很准确, 50%的误差都可以通过. 在 MATLAB 命令窗输入:

```
start_point = [150  0.1  0.12  30];
```

加上上述的 xdata, ydata 两个数组, 就有了调用"fitcurve"函数的必备参数. 但要注意 xdata, ydata

必须是同为行数组或列数组. 如不一致，必须转置. 执行：

[estimates, model] = fitcurve(*x*data, *y*data, start_point)

命令就可得到：

estimates = 120.831 1　0.086 3　0.132 2　34.093 4

model = @fitcurve/expfun

数组 estimates 的内容就是想要的 A、t_r、t_0、d_0 数值. Model 为所求函数的句柄. 为了将结果与原始数据绘图比较，执行：[sse, FittedCurve] = model(estimates);

命令可求出拟合后的曲线值在“FittedCurve”数组中，“sse”为最小均方差. 以下为绘图命令：

plot(*x*data, *y*data, '*')；

hold on;

plot(*x*data, FittedCurve, 'r')；

图 2 为拟合结果图. 由此得到所求的被控对象的阶跃响应的数学表达式为：

$$Y(t) = 120.831\,1 \cdot (1 - \exp(-(t - 0.132\,2) / 0.086\,3)) \cdot U(t - 0.132\,2)$$

这里忽略了直流分量，因为它与系统的动态特性无关. 经拉氏变换得到传递函数为：

$$H(S) = Y(S) \cdot S = 120.831\,1 \cdot \exp(-0.132\,2 \cdot S)/(0.0863 \cdot S + 1)$$

4　大林控制算法建模仿真

对于线性系统，常系数可归到系统的放大倍数 K 中考虑. 因此得归一化的被控对象函数为：

$$G_p(S) = \exp(-0.132\,2 \cdot S)/(0.086\,3 \cdot S + 1) = \exp(-t_0 \cdot S)/(t_r \cdot S + 1)$$

大林指出，通常期望的闭环响应是一阶惯性加纯延时形式，其延时时间等于对象的延时时间 t_0[1]，确定期望的闭环响应设计为：

$$\Phi(S) = \exp(-0.132\,2 \cdot S)/(0.035 \cdot S + 1) = \exp(-t_0 * S)/(t_b \cdot S + 1)$$

并取采样时间为 $t_s = 0.12$ s，$t_b = 0.035$ s. 采样时间 t_s 与闭环响应的时间常数 t_b 的选取是先根据经验取 $t_s \approx 0.9t_0$，t_b 取在 $0.3t_0$ 和 $0.5t_r$ 之间，通过仿真再加以调整.

以下命令是将两函数输入 MATLAB 并转换成 Z 函数. 再根据式(2)求出控制器函数 $G_c(Z)$. 把这些命令存为“gethf. m”文件到工作目录，调试时只要在编辑器中修改相应参数后保存. 然后执行此文件就立即得到新的 $G_c(Z)$ 数据.

```
gps = tf([1], [0.0863, 1], 'inputdelay', 0.1322);   % 输入 Gp(S).
sys = tf([1], [0.035, 1], 'inputdelay', 0.1322);    % 输入 Φ(S).
ts = 0.12;                                          % 输入采样时间.
gpz = c2d(gps, ts, 'zoh');                          % 将 Gp(S) 转换为 Gp(Z).
dsys = c2d(sys, ts, 'zoh');                         % 将 Φ(S) 转换为 Φ(Z).
gcz = 1/gpz * dsys/(1 - dsys);                      % 根据式(2)求出控制器函数 Gc(Z).
[num, den] = tfdata(dsys1, 'v');                    % 获得 Gc(Z) 的分子和分母的系数数组.
```

在命令窗输入“gethf”，便产生了控制器的传递函数. 再输入“gcz”便看到它的表达式：

$$H(z) = z^{-1} * \frac{0.954z^7 - 0.254\,9z^6 + 0.003\,897z^5 + 0.000\,109\,2z^4}{0.713\,2z^6 - 0.008\,463z^5 - 0.682\,2z^4 - 0.023\,6z^3 + 0.000\,971\,3z^2 + 1.658\mathrm{e}(-005)z} \quad (5)$$

用 simulink 工具创建如图 3 的仿真模型[3]. 其中控制器模块是 simulink 离散模块库中的传输函数“Discrete Transfer Fun”模块. 双击它可弹出参数设置对话框. 该对话框有 3 项内容：分子系数、分母系

数和采样时间. 可直接用对应的变量名填入. 但要注意, 由于产生的"den"数组的内容是降一阶的数据, 因此需在"den"后面加个零. 即在 3 个空格中分别填入"num"、"[den 0]"、"t_s". 这样当修改参数执行"gethf"文件后, 改变后的数据自动传递到模块中, 可立即运行仿真看结果, 十分便捷. 被控对象是用连续模块库中的"Transfer Fun"和"Transport Delay"模块组成. $G_p(S)$ 参数设置同样在其设置对话框中, 并在图 4 中可见其数据. 延时模块时间取 0.132 2 s. "Step"模块为阶跃信号源, "scope"是示波器.

做完这些工作就可以进行仿真运行了. 把仿真停止时间设为 1 s, 按下运行按钮, 双击"scope"就可以看到被控对象在阶跃信号驱动下的输出波形如图 4 . 当然这个理想的曲线是经过反复仿真调试的结果, 所选用的参数是 t_s = 0.12 s, t_b = 0.035 s.

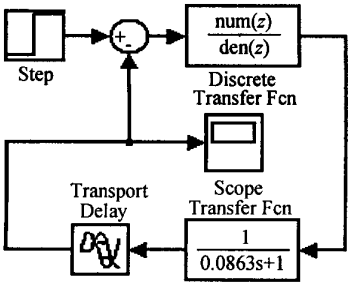

图 3 仿真模型图

Fig. 3 Simulink model

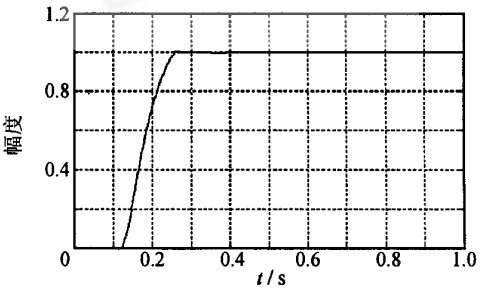

图 4 仿真闭环系统阶跃响应曲线

Fig. 4 Step response by simulating the closed – loop system

5 控制器编程

从图 1 和式(5) 可知控制器的输出为:

$$U(Z) = G_c(Z) \cdot E(Z) = (\mathrm{num}(Z)/\mathrm{den}(Z)) \cdot E(Z)$$

从上式解得:

$$U(Z) \cdot \mathrm{den}(Z) = E(Z) \cdot \mathrm{num}(Z)$$

即: $$U(Z) \cdot \sum a_r Z^{-r} = E(Z) \cdot \sum b_r Z^{-r} \qquad (r = 0, 1, 2, \cdots, 7)$$

求逆变换得: $$\sum a_r \cdot u(k-r) = \sum b_r \cdot e(k-r) \qquad (r = 0, 1, 2, \cdots, 7)$$

解得: $$u(k) = (b_0 \cdot e(k) + \sum (b_r \cdot e(k-r) - a_r \cdot u(k-r)))/a_0 \qquad (r = 1, 2, \cdots, 7)$$

即: $$u(k) = (b_0/a_0) \cdot e(k) + F(k) \tag{6}$$

其中: $$F(k) = (\sum (b_r \cdot e(k-r) - a_r \cdot u(k-r)))/a_0 \qquad (r = 1, 2, \cdots, 7) \tag{7}$$

对应于式(5) 中的各系数 a_r和 b_r 分别为 (忽略小于 0.002 的数):

$a_0 = 0.713\,2$, $a_1 = -0.008\,46$, $a_2 = -0.682\,2$, $a_3 = -0.023\,6$, $a_4 = a_5 = a_6 = a_7 = 0$

$b_1 = 0.954$, $b_1 = -0.254\,9$, $b_2 = 0.003\,9$, $b_3 = b_4 = b_5 = b_6 = b_7 = 0$

将这些系数代入式(6)、式(7) 有:

$$u(k) = 1.337\,6 \cdot e(k) + F(k-1) \tag{8}$$

$$F(k-1) = -0.357\,4 \cdot e(k-1) + 0.005\,6 \cdot e(k-2) - 0.011\,9 \cdot u(k-1) - 0.956\,5 \cdot u(k-2) - 0.033\,1 \cdot u(k-3) \tag{9}$$

这些系数都是归一化后的系数, 在编写实际控制器程序时, 必须考虑信号的幅度, 也就是要考虑被控对象(包括除了软件控制器以外的所有环节) 的放大倍数 K_p. 对于本文采集的数据来说, K_p =

120.83/5 = 24.166. 因此，软件控制器的放大倍数必须等于 1/24.166. 保证闭环增益等于 1，才能保证实际开发的控制器的效果符合仿真的结果. 按照式(8)、式(9)便可设计运算程序，其方框图如图 5 所示.

程序中，控制器的输出紧接在 $U(k)$ 运算完成之后，这样有利于减少系统的延时. 输出之后再计算 $F(k)$ 并保存，以备下次循环计算 $U(k)$ 使用. 本次的 $F(k)$ 即为下次的 $F(k-1)$，因此只需要计算 $F(k)$ 即可. 控制目标值可以是软件内部设定的某恒定值，也可以是外部给定的某变量. 总之是被控对象跟踪的目标.

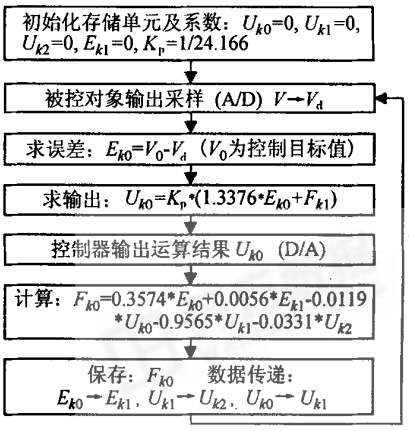

图 5　控制器运算程序方框图

Fig. 5　Flow – diagram of controlling software

6　结果与讨论

在开发基于测厚和液压的无卡轴旋切机的过程中，运用此方法进行调试，取得了满意的效果. 整个调试过程只在机器上进行了一次跃阶响应采样操作，其余调试工作都在计算机上进行仿真. 参数确定后，实际开机无需再调整，就能达到理想的切割效果. 图 6 是调试结果的实验曲线. 与图 4 的仿真结果曲线相比，除了有小幅噪声信号存在外，曲线的整体趋势基本相同. 而这噪声是由于测厚机构的振动和木片纹理粗糙造成的，这不影响切割质量. 实践证明，此方法是调试一阶惯性延时控制系统的行之有效的方法，具有操作简便、节约资源、保障安全、仿真精确等优点. 这些都是得益于功能强大的 MATLAB 软件. 本文使用的是 7.1 版本.

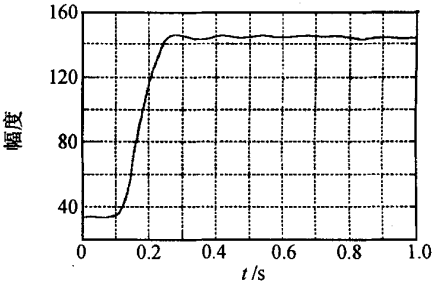

图 6　在阶跃信号驱动的实验曲线

Fig. 6　Step – response of real machine

上述方法适用于一阶惯性延时系统. 从理论上来说，也适用于其他系统，但如何实现还有待进一步研究. 至于如何知道所研究的对象就是一阶惯性延时系统，这只要看所获得的阶跃响应曲线是否是如图 2 所示的指数曲线，因为这由系统理论所决定了的.

参考文献：

[1] 刘金琨. 先进 PID 控制及其 MATLAB 仿真[M]. 北京：电子工业出版社，2003.

[2] 郑剑翔. 基于测厚和液压的无卡轴旋切机[J]. 福州大学学报：自然科学版，2006，34(4)：529 – 532.

[3] 姚 俊，马松辉. Simulink 建模与仿真[M]. 西安：西安电子科技大学出版社，2002.