Automatic Loop Tuning Using CAD Software Tools for PI, PID, pPI and PIP Strategies

PROFESSOR C.S. COX AND DR A.F. DOONAN
Control Systems Centre
University of Sunderland
Chester Road, Sunderland, SR1 3SD.
UNITED KINGDOM

Abstract: This paper discusses the features and operation of three software packages which have been developed to automate the procedure of parameter selection for PI(D), pPI and PIP algorithms. Screen shots of the software in action together with a brief outline of the encoded theory is included.

Key-Words: Computer aided control system design, Auto-tuning control, Predictive PI, PIP, relay tuning.

1. Introduction

In recent years the emphasis within many areas of manufacturing has moved from the construction aspects of the associated processing plant to one of investigating methods of optimising the operational efficiency of the existing hardware assets. During the same period, developments that have taken place in the area of microprocessor technology have realised a variety of hardware platforms to implement improved, more sophisticated control strategies. Consequently, it is something of an anomaly to discover that the vast majority of regulators used in industry are of the simple three-term (PID) form. However, even after careful instruction, plant operators still have difficulty in installing and operating such systems. The task of choosing the PID parameter values is referred to as 'tuning'. A feedback control system in of little value if it is improperly tuned.

This paper chronicles the attempt of a small group of academic practitioners at the University of Sunderland to develop a family of software tools especially introduced to facilitate the tuning of traditional and predictive control strategies. The paper will explain via a number of real industrial problems how the tools have been developed and subsequently refined. Two different software packages will be introduced. These are ***MasterTune***, for the tuning of PI and PID controllers which incorporates ***QuickTune*** for the tuning of PI and pPI strategies and ***ProDesign*** for the tuning of the PIP algorithm.

2. MasterTune and QuickTune

The kernel of the ***MasterTune*** software[1] is the Astrom and Hagglund[2] autotuner. However, it has a family of additional refinements which make it easy to use whilst at the same time producing consistent reliable responses far superior[3] to those obtained using well tried empirical formulae[4,5]. The tuning philosophy consists of three distinct stages, *Process Analysis*, *Relay Tuning* and *Performance Evaluation*. Each stage is outlined below after an introduction t o the theory behind the technique.

2.1 Relay Auto Tuning

In 1984, Astrom and Hagglund[2] described an automatic tuning (auto tuning) method which utilised a relay controller plus an integrator (see Figure 1) to force the system to oscillate with a constant amplitude, fixed frequency oscillation known as a limit cycle.

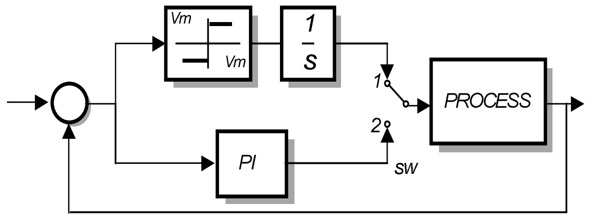


Figure 1 A block diagram showing the configuration of the Autotuner scheme and process

An important feature of the technique is that the amplitude of the oscillation can be altered by changing the magnitude of the relay characteristic.

Also, the addition of the integrator has the benefit of forcing the limit cycle to be sustained about the set point value and helps to ensure 'bumpless' transfer between tuning and control modes. Much has been published on the relay auto tuning technique so it is sufficient here to stress that the design condition requires that K_c and T_i be chosen such that

$$T_i = \frac{P_U \tan\phi_m}{2\pi} \quad \text{and} \quad K_c = \frac{2V_m P_U \sin\phi_m}{\pi^2 A}$$

.... *Equation 1*

where

P_u is the period of the limit cycle oscillation

A is the peak value of the limit cycle appearing at the input to the non-linearity, and,

ϕ_m is the phase margin.

Two problems exist for the untrained operator. The first is how do we choose the relay amplitude and secondly how do we choose ϕ_m? These questions are answered after a review of the theoretical background of the pPI design.

2.1.1 Predictive PI (pPI) Design

The problems associated with the occurrence of appreciable time delay in control are not a new phenomenon. Smith[6] highlighted the problem and proposed a scheme to overcome this time delay. The Smith predictor is shown in block diagram format as

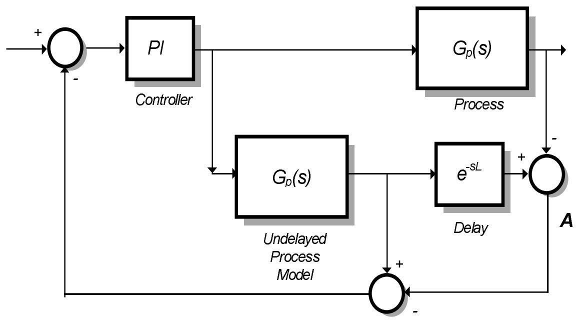


Figure 2 A block diagram of the Smith Predictor

A model of the process and the delay are used in conjunction with a normal PI controller in an attempt to remove the time delay from the characteristic equation. If the model and delay are an exact representation of the process being controlled and the disturbance is zero then the signal at point A will be zero. This means that the controller is left to act on the error between the reference and the undelayed process model. Hence the controller used is designed based purely on the delay free model of the plant.

Figure 2 can be expressed as

$$u(t) = G_c\{e(t) - \hat{G}_p[u(t) - u(t-L)]\}$$

............*Equation 2*

where G_c is the PI controller,

G_p is the process model without delay, and,

L is the time delay exhibited by the system

If we assume, as suggested by Hagglund[2] that

$$\hat{G}_p = K_p \frac{1}{1+s\tau} \qquad G_c = K_c \frac{1+sT_i}{sT_i}$$

............*Equation 3*

where

$K_c = 1/K_p = K$ and $T_i = \tau = T$

then Equation 2 can be re-arranged into the form

$$u(t) = Ke(t) + \frac{1}{K(1+sT)}u(t-L)$$

............*Equation 4*

which gives the straightforward block diagram as shown in Figure 3.

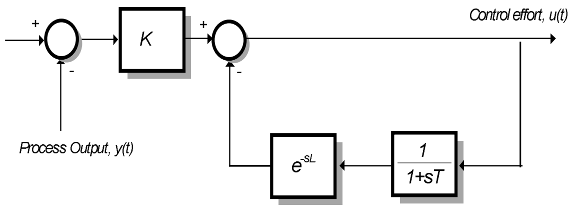


Figure 3 Block diagram of the simplified Smith Predictor (pPI controller)

Next consider the block diagram of Figure 4 which is the same as Figure 3 except that the delay term has been removed.

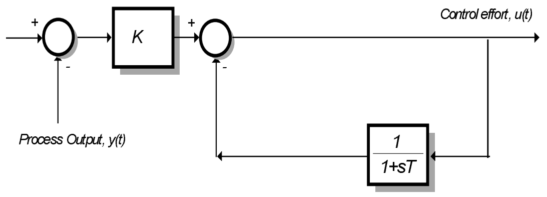


Figure 4 A block diagram of a standard PI controller

The transfer function between the control effort and the error in Figure 4 is

$$G_c(s) = \frac{K(1+sT)}{sT}$$

............*Equation 5*

which is the standard PI controller.

2.2 CAD software

CAD software called ***MasterTune*** has been developed to help in the automatic tuning of both the traditional PI and the pPI controllers and is briefly outlined below.

2.2.1 Set test parameters

The value of various test parameters used during the tuning phase are automatically set to default values. These values are set by the software to provide 'sensible' tuning parameters however, if the user so chooses these values can be modified to suit individual requirements. The parameters which can be modified include

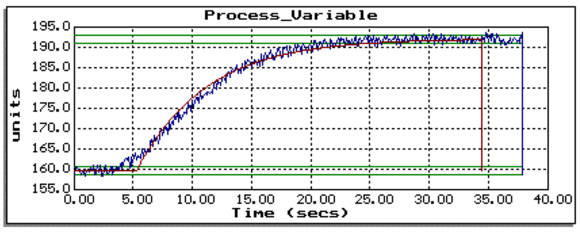
- The percentage overshoot: the amount of overshoot the closed loop system will exhibit when compensated by the tuned PI controller. The percentage overshoot is used to determine the phase margin required by the design equations (see Equation 1).
- The relay characteristics: the amplitude (to control the size of the limit cycle oscillations during tuning) and the hysteresis (used to prevent false switching caused by noisy signals).
- Tolerance between peaks: used by the software to detect when the limit cycle has become stable.
- Constraints: control over the allowable level variations of both the process and manipulated variables.

2.2.2 Process analysis phase

The process analysis phase is the preliminary step of the tuning procedure. Here an open loop step test is conducted and a first order plus dead time model automatically calculated using a characteristic area approach[7]. The process inputs and outputs are visually displayed to the process operator throughout and on completion of the test the model is overlaid onto the process reaction curve allowing an informed judgement to be made about the quality of the model (Figure 5).

2.2.2.1 PI QuickTune

If the time delay is not appreciable in relation to the time constant the software recommends that a PI controller be used. This fast tune facility uses the transfer function obtained in the process analysis phase to calculate the controller settings using some empirical formula, typically those recommended by Daniels and Cox[3]. This feature can be used to get a control loop working satisfactorily in a very short period of time.



```
Estimated process parameters :
    Gain 3.259   Tau 5.855s   Dead time 3.381s
Recommended relay parameters :
    hysteresis 1.005   height 2.991
Predictive-PI settings :
    Xp 325.916   Ti 5.855s
Est. cycle period :
    31.270
```

Figure 5 Typical process analysis result

2.2.2.2 Predictive PI (pPI)

If the time delay is large compared to the principal time constant, then a PI controller will not be able to yield satisfactory performance. In this situation the software recommends that the pPI strategy is utilised and calculates the controller parameters as described earlier. For implementation of the pPI algorithm it is recommended to use a programmable process controller.

2.2.2.3 Setting the relay characteristics

If, as recommended, the fast tune procedure is by passed in favour of the relay feedback autotuning, then the relay characteristics must be set. The process analysis phase calculates a value for the relay height which will produce a limit cycle with an amplitude approximately equal to that set in the 'Set test parameters' phase above. Additionally, the software will analyse the level of noise imposed on the process variable and use this value to suggest a level of hysteresis. This stage relieves the process operator of the rather complex task of specifying the relay characteristic. The operator only needs to specify the tolerable process variable swing. Further, it should be noted that this stage may only be required when tuning a process for the first time. In subsequent sessions, the relay parameters can be entered directly based on previously recorded values.

2.2.3. Connection to the plant

Connection to the plant is achieved by utilising the RS232 communications port present on modern commercial controllers. Figure 6 illustrates this technique.

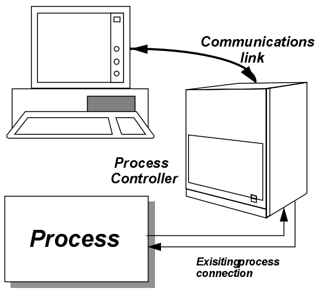
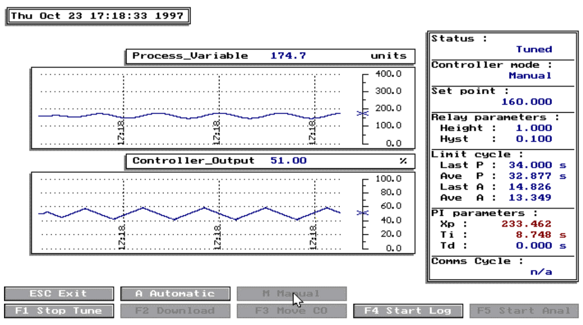


Figure 6 Hardware configuration

2.2.3 AutoTune

When the autotune in invoked, the relay and integrator force the process variable to oscillate with the specified amplitude. After a stable limit cycle is exhibited by the process, a PI controller is calculated which will result with the required level of overshoot being observed during compensated evaluation tests (see Figure 7).

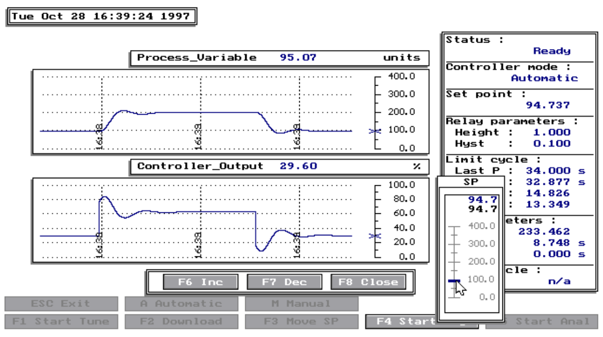


```
Thu Oct 23 17:18:33 1997

Process_Variable   174.7   units
Controller_Output  51.00   %

Status :
            Tuned
Controller mode :
           Manual
Set point :
          160.000
Relay parameters :
  Height :   1.000
  Hyst   :   0.100
Limit cycle :
  Last P :  34.000 s
  Ave  P :  32.877 s
  Last A :  14.826
  Ave  A :  13.349
PI parameters :
  Xp :     233.462
  Ti :       8.748 s
  Td :       0.000 s
Comms Cycle :
              n/a
```

Figure 7 CAD software screen dump showing Autotuning in process

2.2.4 Evaluation

Once the appropriate controller has been selected and tuned, the evaluation stage can be implemented. Here the controller is put into automatic mode and, if desired, a step change can be induced in the set-point (see Figure 8). This, as well as each other stage in the tuning cycle, can be logged and saved in a data file to provide a full record of the work completed.



```
Tue Oct 28 16:39:24 1997

Process_Variable   95.07   units
Controller_Output  29.60   %

Status :
            Ready
Controller mode :
        Automatic
Set point :
           94.737
Relay parameters :
  Height :   1.000
  Hyst   :   0.100
Limit cycle :
  Last P :  34.000 s
         :  32.877 s
         :  14.826
         :  13.349
eters :
           233.462
             8.748 s
             0.000 s
cle :
              n/a
```

Figure 8 CAD software screen dump showing evaluation of PI controller

3. ProDesign

The PIP control algorithm is one element of a total design environment that rejects the idea that a digital control system be initially designed in continuous-time terms. Rather it demands that the designer considers the design from a digital sampled data standpoint. The design method to be discussed here is based upon 'discrete-time state-space plant descriptions' but as we shall see later the corresponding 'pulse transfer function model' will also play a part in highlighting some of the features of the new algorithm.

3.1 PIP Control

Given a *Non-Minimal State Space (NMSS)* description of a process, i.e. a state space representation with the following state choice

$$\mathbf{x}(k) = \big[y(k), y(k-1), \dots, y(k-p+1), \dots \\ u(k-1), u(k-2), \dots, u(k-q+1), m(k) \big]'$$

............*Equation 6*

where *m(k),* the integral of error is given by

$$m(k) = m(k-1) + \big(y_d(k) - y(k)\big)$$

............*Equation 7*

and $y_d(k)$ is the desired output or set point.

Note that *x(k)* consists of the present value of the system output as well as past values of both plant input and output. *m(k)* is included to guarantee 'type 1' servo-mechanism performance. The plant in matrix form becomes

$$\mathbf{x}(k) = \mathbf{F}\mathbf{x}(k-1) + \mathbf{g}u(k-1) + \mathbf{d}y_d(k)$$
$$y(k) = \mathbf{h}\mathbf{x}(k)$$

............*Equation 8*

$$\mathbf{F} = \left[\begin{array}{cc:cccc:cc:c}
-a_1 & -a_2 & -a_{p-1} & -a_p & b_2 & b_3 & b_{q-1} & b_q & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
a_1 & a_2 & a_{p-1} & a_p & -b_2 & -b_3 & -b_{q-1} & -b_q & 1
\end{array}\right]$$

$$\mathbf{g} = [\; b_1 \quad 0 \quad 0 \;\vdots\; 0 \quad 1 \quad 0 \quad 0 \;\vdots\; 0 \quad -b_1 \;]'$$
$$\mathbf{d} = [\; 0 \quad 0 \quad 0 \;\vdots\; 0 \quad 0 \quad 0 \quad 0 \;\vdots\; 0 \quad 1 \;]'$$
$$\mathbf{h} = [\; 1 \quad 0 \quad 0 \;\vdots\; 0 \quad 0 \quad 0 \quad 0 \;\vdots\; 0 \quad 0 \;]$$

The use of the NMSS description of the plant and the LSVFB control law results in a control algorithm which has been named by Young *et al.*[8] as the *Proportional Integral Plus* (PIP) algorithm, thus using the following control law

$$u(k) = -\mathbf{v}'\,\mathbf{x}(k)$$

.... *Equation 9*

where the elements of the gain vector, $\mathbf{v}$, are conveniently grouped and given the following reference,

$$\mathbf{v}' = \left[f_0, f_1, f_2, \dots f_{p-1}, \quad g_1 g_2, \dots g_{q-1}, \quad -k_I \right],$$

and,

$\mathbf{x}(k)$ are the states chosen to produce a NMSS description.

Expressing the PIP control law in polynomial form allows greater insight into the algorithm and the reason for grouping the controller terms as above. Figure 9 shows PIP in block diagram form with Equation 10 expressing the control law.

$$\begin{aligned} u(k) &= -\left(f_0 + f_1 z^{-1} + f_2 z^{-2} + \dots\right) y(k) \\ &\quad - \left(g_1 z^{-1} + g_2 z^{-2} + \dots\right) u(k) + k_I m(k) \end{aligned}$$

.... *Equation 10*

where

$$m(k) = \frac{1}{\left(1 - z^{-1}\right)} \left[y_d(k) - y(k) \right]$$

.... *Equation 11*

and $y_d(k)$ is the set point.

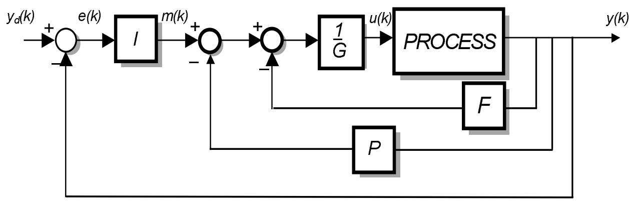


Figure 9 Block diagram of the PIP control algorithm

As can be seen from Figure 9, in addition to proportional (arising from the f_0 gain*)* and integral action (arising from *m(k)*) the controller also contains additional *feedforward* and *feedback discrete time filters* which are the *plus* part of PIP. The '*f*' gains of the gain vector operate on past values of the system *output* and the '*g*' gains operate on previously generated control actions. The order of these additional polynomials is dependent upon the order of the system under control. In fact, the PIP algorithm can be interpreted as containing a 'model' of the process in these polynomials and it is this which results in its impressive performance of the algorithm as presented in later sections. In Figure 9 the following polynomials can be identified,

$$F(z^{-1}) = f_1 z^{-1} + f_2 z^{-2} + \dots + f_{p-1} z^{-(p-1)}$$
$$G(z^{-1}) = 1 + g_1 z^{-1} + g_2 z^{-2} + \dots + g_{q-1} z^{-(q-1)}$$

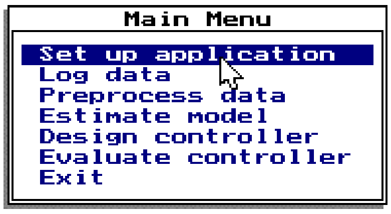
............*Equation 12*

3.1.1 Computing PIP gains

There remains the requirement of computing the controller gains k_i , f_0 to f_{p-1} and g_1 to g_{q-1}. Two distinct strategies for the completion of this task are available. Firstly the gains can be calculated by the designer specifying the closed loop pole position. Appropriate choice of these poles leaves the designer with the ability of adjusting the closed loop response to their satisfaction (provided the system is completely controllable[9]). However, in an industrial environment this flexibility (and the requirement of specifying *p+q* pole locations) can be seen as overly complex and hence a drawback. In overcoming this, an alternative suggestion is to use a multiple pole model where the location of the pole acts as a tuning parameter. Secondly a technique whereby a set of gains, optimal in the Linear-Quadratic sense, is also calculated.

3.2 CAD Software

The *Main Menu* (Figure 10) summarises the individual stages of the design strategy, and appears first on the screen. The menu is redisplayed after each stage in order to guide the user through the package. A preliminary step is the *Set up application* program which allows the user to assign names to the process signals and to specify their ranges in engineering units: for example the process variable could be tagged as "Flow" and given a range of 0-10 litres/min. Additionally rate limits may also be defined for the controller output signal. This set up data is the stored under a specific file name for use by the other programs in the suite.



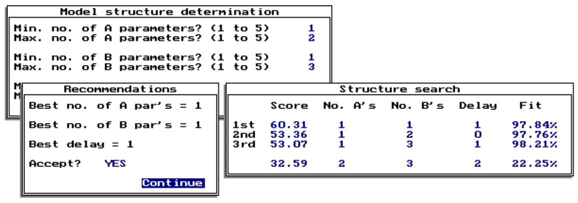
```
Main Menu
 Set up application
 Log data
 Preprocess data
 Estimate model
 Design controller
 Evaluate controller
 Exit
```

Figure 10 The main menu of the ***ProDesign*** CAD software

Having set up the application, the first proper stage is the *Log data* option which is used for performing system identification tests and recording the test data. The user has the opportunity to specify the test parameters as well as the ranges of the chart recorder type displays. Once the PC has initiated communications with the controller, real time trends of the process signals start to evolve. Note that the controller is still in command at this point, and the PC will only take over when remote mode is selected on the controller. The transfer is bumpless. Once in charge, the user may start recording and then introduce perturbations into the process. The controller output may be manipulated using a slider, or, alternatively, step functions and/or pseudo random binary sequences (PRBS) may be selected by clicking on the appropriate buttons. It is a good idea to try a step test first, in order to gain an appreciation of the process response time, from which a suitable sampling time may be deduced[10]. For example, if the process rise time was found to be approximately 10 seconds then a sampling time of 1 second is recommended for the PRBS test. This second test was allowed to run for about 100 seconds so that a statistically 'large' data set was obtained. Having carried out a system identification test, command may then be restored to the controller.

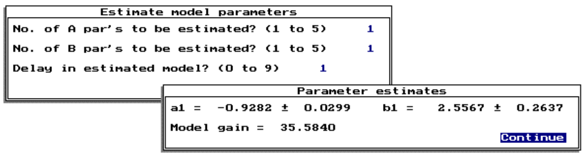
The second stage of the procedure is *Pre-process data* which hold a variety of operations designed to 'clean up' the bad data points and disturbances that are common features of real industrial data. These operations include: selecting data subsets, marking bad data points, removing low frequency trends and removing bias.

Estimate model is the next stage, which employs statistical methods to estimate a difference equation model from the pre-processed data. In its default mode, this program uses a three pass recursive-iterative instrumental variables algorithm. The program can be asked to recommend an appropriate model structure, i.e. the amount of pure time delay and the number of '*a*' and '*b*' parameters that are required. Users may accept these of specify their own structure. Figure 11 and Figure 12 present the screen dumps from the structure search and parameter estimation routines for typical data.



```
Model structure determination
Min. no. of A parameters? (1 to 5)      1
Max. no. of A parameters? (1 to 5)      2

Min. no. of B parameters? (1 to 5)      1
Max. no. of B parameters? (1 to 5)      3

Recommendations
Best no. of A par's = 1
Best no. of B par's = 1
Best delay = 1
Accept?   YES
                    Continue

Structure search
      Score  No. A's  No. B's  Delay   Fit
1st   60.31    1        1        1    97.84%
2nd   53.36    1        2        0    97.76%
3rd   53.07    1        3        1    98.21%

      32.59    2        3        2    22.25%
```

Figure 11 CAD software showing the structure selection stage



```
Estimate model parameters
No. of A par's to be estimated? (1 to 5)      1
No. of B par's to be estimated? (1 to 5)      1
Delay in estimated model? (0 to 9)      1

Parameter estimates
a1 =  -0.9282 ±  0.0299    b1 =   2.5567 ±  0.2637
Model gain =  35.5840
                                            Continue
```

Figure 12 CAD software showing the model parameter estimates

Finally, in this section the model 'fit' is presented graphically as shown in Figure 13.

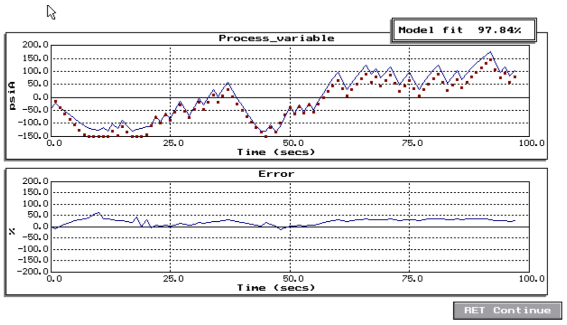


Figure 13 CAD software showing graphically the 'fit' of the estimated model

Having obtained a model, the *Design controller* program may be invoked This stage employs a dynamic programming algorithm to obtain 'optimal' controller gains[11], and the user is presented with a fast interactive facility for adjusting the design weights until a suitable closed loop response is achieved

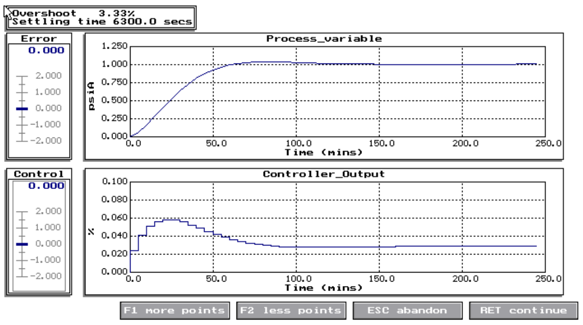


Figure 14 CAD software allowing the use to design the LQ optimal controller gains

Figure 14, a screen dump from this part of the program, shows the sliders on the left which are used to vary the design weights. Having made an adjustment the gains are recomputed, and the new response is displayed. On a typical 486 based PC with a maths co-processor the update time is typically less than a second thus making it extremely pleasant to use.

In the final stage, *Evaluate controller,* The PC once again establishes communications with the controller, and down loads the computed gains. As with the data logging program, real time trends are presented on the screen, and the user has the same facilities to define the display ranges. Set point changes may also be introduced, via a slider, and the data recorded for further analysis. When the user has evaluated the performance satisfactorily, local set point mode may be selected, and the PC disconnected, thus leaving the control law to execute within the programmable controller.

4. Conclusion

The PI(D) controller is still the dominant system used in current industrial practice. Much has been published about the design of PI(D) regulators, however, the majority of the strategies rely on accurate process information not normally available to the commissioning engineer installing systems on 'real plant'. For this reason, alternative methods have been sought which still provide good controller settings based upon imprecise process knowledge. The ***MasterTune*** software incorporates several of the latest techniques and provides an automatic medium for parameter determination. Another attractive feature is the ability to download the parameters directly into the actual process controller once the values are accepted.

Whilst PI(D) controllers are still ubiquitous, they still have some important limitations. For these situations, it is now well established that some form of prediction needs to be included within the algorithm. ***QuickTune*** provides a rapid means of incorporating the pPI strategy. An alternative approach is to use the ***ProDesign*** package which employs the PIP strategy. For those cases when a very long time delay clearly degrades the PI(D) response it is found that PIP usually produces an excellent response.

It is concluded that modern process engineers need to include both methodologies in their armoury to satisfy the ever increasing stringency of today's process specifications.

References:

[1] Doonan A.F., PhD Thesis : The Development, Evaluation and Implementation of True Digital Control, University of Sunderland 1997.

[2] Astrom K.J. and Hagglund T., *Automatica,* 20, 645-651 1984. 'Automatic tuning of simple regulators with specifications on phase and amplitude margins'.

[3] Cox C.S., Daniel P.R. and Lowdon A., *IFAC Preprints : Algorithms and Architectures for Real-Time Control, Vilamura, Portugal, 9-11 April* 1997, 'QUICKTUNE: A realiable automatic strategy for determining PI and pPI controller parameters using a FOPDT model'.

[4] Lopez A.M., Smith C.L. and Murril P.W., *Inst Tech* Vol 14 No 3 1967, 'Controller tuning relationships based on integral criteria'.

[5] Zhaung M. and Atherton D.P., *IEE Proceedings - D* Vol 140 No 3 1993, 'Automatic tuning of optimum PID controllers'.

[6] Smith O.J., *ISA Journal*, Vol 6, no 2 1959, 'A controller to overcome dead time'

[7] Nishikawa Y., Sannomiya N., Ohat T. and Tanaka H., *Automatica* 20 1984, 'A method of autotuning of PID parameters'.

[8] Young P.C., Behzadi M.A., Wang C.-L. and Chotai A., *Int. Jnl. of Control* 46 1987, 'Direct digital and adaptive control by input-output, state variable feedback pole assignment'.

[9] Kuo B.C., *Digital Control Systems,* CBS Publishing 1980.

[10] Boucher A.R., PhD Thesis : Management Strategies for a Water Treatment Plant, Sunderland Polytechnic 1991.

[11] Billington A.J., Boucher A.R. and Cox C.S., *Conference Publication No 332 : Control 91* Vol 2 1991, 'Optimal PIP control of scalar and multivariable processes'.