International Journal of Pharmaceutics, 55 (1989) 193–198
Elsevier

IJP 01871

Curve fitting of dissolution data by personal computer

Sverre A. Sande and Jan Karlsen

Department of Pharmaceutics, Institute of Pharmacy, University of Oslo, Oslo (Norway)

(Received 3 January 1989)
(Modified version received 10 April 1989)
(Accepted 12 April 1989)

Key words: Dissolution testing; Mathematical model; Curve fitting; Computer program

Summary

A computer program written in Pascal for use on personal computers is described. The program performs curve fitting to a mathematical model suitable for in vitro dissolution data. The initial values for the parameters are calculated from the raw data. The time used by the program was tested with several datasets and found to be acceptable also with a minimum hardware configuration. The performance of the program was compared to the performance of a program for general non-linear regression running on a mainframe computer. There were only minor differences in the calculated parameters and standard deviations between the two systems.

Introduction

The evaluation of the dissolution rate of solid preparations is usually done without any fitting of the data. The advantage of fitting data to a mathematical model is that one is able to describe the entire dissolution process with a few parameters, thus making statistical evaluations and comparisons easier. Several mathematical models have been proposed to describe the dissolution process.

Noyes and Whitney proposed an equation for the dissolution process in 1897. For plain tablets, where the dissolution process is the rate-limiting step, this would be an adequate model. Several modifications have been suggested, e.g. Wagner's log-normal distribution (Wagner, 1969) and the sigma-minus-plots (Martin, 1967).

The cube-root law was suggested by Hixon and Crowell (1931), and this is also still in current use (Bamba et al., 1979; Swarbrick and Ma, 1981).

A third approach is the Schering-plots, which bear a strong resemblance to the Lineweaver-Burk plots in enzyme kinetics (Fuchs et al., 1968).

A mathematical model of more general applicability was first described by Rosin and Rammler (1934) and "rediscovered" by Weibull in 1951. The first who suggested the model used for in vitro dissolution data was Langenbucher (1972). A thorough discussion of the model and significance of the parameters was presented in a later paper (Langenbucher, 1976). The model is:

$$M = M_u \cdot \left(1 - e^{-[(t-T_0)/T_d]^{\beta}}\right)$$

where M = amount dissolved at time t; M_u =

Correspondence: S.A. Sande, Department of Pharmaceutics, Institute of Pharmacy, University of Oslo, P.O. Box 1068, Blindern, 0316 Oslo 3, Norway.

amount dissolved after infinite time; T_0 = lag time; T_d = a time parameter; β = a shape parameter.

As can be seen, the model is an extension of the Noyes-Whitney equation. Basically the model contains two parameters (T_d and β), the other two serving only as scale parameters.

The advantage of this model is its capability in dealing both with S-shaped curves (drug dissolution from disintegrating tablets) and curves with a fast initial release followed by a slower release (sustained release tablets).

T_d represents the time for release of 63.2% of total dose. When $\beta = 1$, this corresponds to the mean dissolution time as defined by Dost (1958).

The determination of the 4 parameters may be performed by general programs for non-linear regression. The disadvantage of such programs is, however, that they cannot be run or are extremely slow on personal computers. In addition they require an initial value for the parameters.

In the paper by Langenbucher (1976), a method for linearization of the model was given, provided the T_0 and M_u was known. The determination of T_0 and M_u was suggested to be solved by an iterative method, where the two parameters were varied and the sum of squared residuals (SSR) were monitored for determination of the best fit.

This paper reports a program for a personal computer utilizing this linearization procedure, and, taking the initial values for the parameters from the raw-data, performs the necessary iterations to determine all 4 parameters. A comparison between the program and a traditional non-linear regression analysis is also presented.

Program

The program is written in Pascal and based on 3 procedures. The first iteratively varies the T_0 and calculates the best F_u for each T_0.

The second calculates the best F_u, given a T_0.

The third procedure calculates T_d and β by weighted linear regression acccording to the equations given in the paper of Langenbucher (1976).

The best fit criterion is based on calculation of the SSR from the difference between the raw-data and calculated values from the parameters.

An outline of the program is given in the Appendix.

Testing

The program was compiled by a Turbo Pascal compiler version 2 (Borland international U.S., Scotts Valley, CA). The time used by the program on a standard personal computer with Intel 8088 CPU, no co-processor and a clock-speed of 4.77 MHz was recorded for several datasets. The limit for the iterations was set to 10^{-7}. Datasets were generated from a perfect curve with a noise of $\pm 10\%$. Ten datasets were created for β-values of 0.5, 1.0 and 1.5, respectively. The values of the other parameters were: $M_u = 10$, $T_d = 10$ and $T_0 = 0.2$. Each dataset contained points for the time-values 1, 3, 5, 8, 12, 20 and 30.

Each dataset were analyzed on the program reported in this paper as well as on a program for general non-linear regression (NLIN-procedure in the SAS system (SAS Institute U.S., Cary, NC)).

Performance

Typical time-values for the iterative part of the program was 5–7 min. For a dataset where the SAS-NLIN program failed to converge, the time used was 11.5 min.

This may seem quite a while, but it should be noted that a most unfavorable computer was used. A higher clock-rate on 8 or 10 MHz, which is common on newer computers, and the use of a better processor or a co-processor would substantially decrease the time required for the calculations. Employing an Intel 8087 co-processor decreased the time used for the calculations from 11.5 min to 0.6 minutes for the above-mentioned dataset. The mean parameters from the test-runs are given in Table 1. The regression coefficient given (*Reg*) is:

$$Reg = 1 - (\text{SSR}/\text{Sum of squared observations})$$

For two datasets with $\beta = 0.5$ the NLIN-procedure failed to converge by using the standard setup. The comparison of the parameters is therefore restricted to the 8 remaining sets.

The method used to determine significant difference between the parameters was the Student's *t*-test for differences between paired samples.

TABLE 1

Comparison of mean parameters given by the two programs

β			$F_u = 10$	$T_d = 10$	β	$T_0 = 0.2$	*Reg*
0.5	Prg	Mean	12.0200	24.7799	0.4914	0.1915	0.9990
		SD%	26.87	97.51	17.27	127	0.11
	Non	Mean	11.9893	24.6143	0.4977	0.1825	**0.9991**
		S.D.%	27.10	97.47	18.15	132	0.11
1.0	Prg	Mean	9.9164	**9.3863**	**1.0749**	0.1194	0.9993
		S.D.%	6.89	9.76	11.36	150	0.04
	Non	Mean	9.9216	**9.3571**	**1.0883**	0.1177	0.9993
		S.D.%	6.82	9.87	11.96	152	0.04
1.5	Prg	Mean	**10.2546**	10.3663	**1.4468**	0.2257	**0.9983**
		S.D.%	6.29	7.65	10.14	118	0.07
	Non	Mean	**10.3187**	10.3895	**1.5056**	0.1803	**0.9985**
		SD%	5.57	7.89	11.83	161	0.06

Significant differences typed in boldface.

There were only minute differences between the resulting parameters and standard deviations from the two systems (differences mostly in the third digit). As can be seen from Table 1, SAS-NLIN produced better fits (smaller residuals) than the program. The reason for this is the small error introduced in the logarithmation done during calculation of T_d and β combined with the weighting of the data. The effect on the resulting curve is, however, marginal, so for all practical purposes the methods are equivalent.

Conclusion

The program has proven to give as good results as a larger general system for non-linear regression. It is simpler in use since starting values for the regression is chosen from the raw-data, and, unlike the general system, it will always converge. Last but not least, it works on a personal computer.

Appendix

Some non-standard Pascal expressions have been used for the sake of clarity. Comments in capital letters are explanatory comments, other comments are hints or indications for further programming.

```
program weifit (input,output);

 type
  parms=record
        Fu,T0,Td,beta,SSR:real;     {PARAMETERS, AND SQUARED SUM OF RESIDUALS}
       end;

 var
  time,fract: array [1..50] of real;
  parset    : array[1..3,1..3] of parms;
                        {parset[1..3,*] HAVE THE HIGH,MEDIUM AND LOW T0-VALUES
                        parset[*,1..3] HAVE HIGH,MEDIUM AND LOW Fu FOR EACH T0}
  endlevel  : real;
          {ENDLEVEL DETERMINES NUMBER OF SIGNIFICANT FIGURES IN FINAL RESULT}
```

```
{---------------------------------------------------------------------------}

 procedure linreg(sett:parms);
               {PERFORMS WEIGHTED LINEAR REGRESSION ON A SET,GIVEN TO AND Fu}
  var
   i:integer;
   fi,weight,y,x, ... and a lot of sums .... :real;

  begin
    {Set SSR and all sums equal to 0}
    for i:=1 to {number of points} do
     begin
       fi:=fract[i]/sett.Fu;
       if fi<1 then
        begin                                    {    ALL
         {calculate weight,x and y                    CALCULATIONS
          add up all sums }                           ACCORDING TO
                                                      EQUATIONS IN
                                                      (LANGENBUCHER,1976)    }
       end;
    end;
   {Calculate beta from sums
    Calculate Td from beta and sums }
    for i:= 1 to {number of points} do
      sett.SSR:=sett.SSR+sqr(fract[i]-{calculated value from parameters});
  end; {PROC LINREG}
{---------------------------------------------------------------------------}
{---------------------------------------------------------------------------}

 procedure Fufit (T0nr:integer,limit:real);
                                 {DETERMINES THE BEST Fu FOR A GIVEN T0}
  var
   step:real;

  begin
    linreg(parset[T0nr,2]);
    step:=0.1;
    while step > limit do
     begin
       parset[T0nr,1].Fu:=parset[T0nr,2].Fu/(1+step);linreg(parset[T0nr,1]);
       parset[T0nr,3].Fu:=parset[T0nr,2].Fu*(1+step);linreg(parset[T0nr,3]);
                                                  {INITIATION OF THE SETS}

       while not(parset[T0nr,1]>parset[T0nr,2]<parset[T0nr,3]) do
        begin

          {The iterations are performed in the same way as in T0fit
           parset[T0nr,1..3].Fu is increased or decreased depending
           on the relative magnitude of the SSR, folowed by a call
           to linreg to update Td,beta and SSR}
           ;
        end;
      step:=step*0.1;
    end {WHILE (step>limit)-LOOP }
       {parset[T0nr,2] NOW HAS THE BEST FIT Fu (WITH SATISFACTORY ACCURACY)
        FOR A GIVEN T0}
  end; {PROC FUFIT}

{---------------------------------------------------------------------------}

 procedure T0fit;
                  {DETERMINES THE BEST T0}
  var
    step:real;    {STEP IS THE DIFFERENCE BETWEEN THE T0's OF THE PARSETS}

  begin
    {parset[2,2] is filled with appropriate data,
      T0:=0 and Fu:=value of last element in fract array}

    Fufit(2,1E-4); {A HIGHER LEVEL THAN ENDLEVEL IS CHOSEN TO SPEED UP PROCESS}

    step:=time[1]/10;
```

```
while step > endlevel*time[1] do  {ITERATION TO FIND BEST TO }
 begin
   if parset[2,2].TO < step then parset[1,1..3].TO:=0
   else parset[1,1..3].TO:=parset[2,2].TO-step;
   parset[1,2].Fu:=parset[2,2].Fu;
   Fufit(1,1E-4);                                 {SETUP OF THE THREE}
                                                  {DIFFERENT TO's    }
   parset[3,1..3].TO:=parset[2,2].TO+step;
   Fufit(3,1E-4);
       while not ( parset[1,2].SSR>=parset[2,2].SSR<=parset[3,2].SSR ) do
                                          {PERFORM LOOP WHILE parset[2,2]
                                           DOES NOT HAVE THE LOWEST SSR}
        begin
          if parset[1,2].SSR > parset[2,2].SSR > parset[3,2].SSR then
                                               { INCREASE TO }
           begin
             parset[1,2]:=parset[2,2];parset[2,2]:=parset[3,2];
             parset[3,2].TO:=parset[3,2].TO+step;
             Fufit(3,1E-4);  {For small values of step this may be replaced
                              by a direct call to linreg, to speed up process}
           end
          else if parset[1,2].SSR < parset[2,2].SSR < parset[3,2].SSR then
                                                {DECREASE TO}
           begin
             parset[3,2]:=parset[2,2];parset[2,2]:=parset[1,2];
             parset[1,2].TO:=parset[1,2].TO-step;   {IF TO<step THEN TO:=0}

             Fufit(3,1E-4);  {For small values of step this may be replaced
                              by a direct call to linreg, to speed up process}

           end
          else  { HERE parset[2,2].SSR IS GREATEST. MOVE IN THE DIRECTION
                  FOR WHICH SSR IS LOWEST }
           begin
             if parset[1,2].SSR < parset[3,2].SSR then
                 {decrease TO , see above}
             else
                 {increase TO , see above}
           end;
        end; {WHILE (not parset[2,2] lowest SSR)-LOOP}
              {parset[2,2] NOW CONTAINS THE BEST TO FOR THIS STEP
               NOW DECREASE STEP TO GET A CLOSER ESTIMATE}
       step:=step/10;
     end; {WHILE (step >endlevel)-LOOP }
          {parset[2,2] NOW CONTAINS BEST TO WITH SATISFACTORY ACCURACY}
    Fufit(2,endlevel);  { TO ALSO GET A SATISFACTORY ESTIMATE OF FU}
  end; {PROC TOfit}

{*****************************************************************************}

 begin {*** MAIN ***}

  {Reading of rawdata into time and fract arrays,
                          excluding the point 0,0 if given}
  {Reading of endlevel and other options}
  TOfit;
  {Writing of best fit parameters (parset(2,2)}

 end.  {*** MAIN ***}
```

References

Bamba, M., Puisieux, F., Marty, J.P. and Carstensen, J.T., Release mechanism in gelforming sustained release preparations. *Int. J. Pharm.*, 2 (1979) 307–315.

Dost, F.H., Uber ein Einfaches Statistisches Dosis-Umsatz Gesetz. *Klin. Wschr.*, 36 (1958) 655–657.

Fuchs, P., Riemann, J., Richter, H., Röpke, H. and Gibian, H., Uber eine Bewertungsmethode der Lösungsgeschwindigkeit von Wirksubstansen in verschiedenen pharmazeutischen Präparationen. *Arzneim.-Forsch.*, 18 (1968) 112–116.

Hixon, A.W. and Crowell, J.H., Dependence of reaction velocity upon surface and agitation. *Ind. Engng. Chem.*, 23 (1931) 923–930.

Langenbucher, F., Linearization of dissolution rate curves by the Weibull distribution. *J. Pharm. Pharmacol.*, 24 (1972) 979–981.

Langenbucher, F., Parametric representation of dissolution–rate curves by the RRSBW distribution. *Pharm. Ind.*, 38 (1976) 472–477.

Martin, B.K., Treatment of data from drug urinary excretion. *Nature (Lond.)*, 214 (1967) 247–249.

Noyes, A.A. and Whitney, W.R., Ueber die Auflösungsgeschwindigkeit von festen stoffen in ihren eigenen Lösungen. *Z. Physikal. Chem.*, 23 (1897) 689–692.

Rosin, P. and Rammler, E., Die Kornzusammensetzung des Mahlgutes im Lichte der Warscheinlichkeitslehre. *Kolloid-Z.*, 67 (1934) 16–26.

Swarbrick, J. and Ma, D., In vitro dissolution of dapsone. *J. Pharm. Pharmacol.*, 31 (1981) 787–789.

Wagner, J.G., Interpretation of percent dissolved–time plots derived from in vitro testing of conventional tablets and capsules. *J. Pharm. Sci.*, 58 (1969) 1253–1257.

Weibull, W., A statistical distribution function of wide applicability. *J. Appl. Mech.*, 18 (1951) 293–297.