Separation Science and Technology
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713708471

Fast Algorithm for the Conversion of R to Lambda Values in Field-Flow Fractionation

Mark R. Schure[a]
[a] LABORATORY DATA PRODUCTS GROUP DIGITAL EQUIPMENT CORPORATION, MARLBOROUGH, MASSACHUSETTS

To cite this Article Schure, Mark R.(1987) 'Fast Algorithm for the Conversion of R to Lambda Values in Field-Flow Fractionation', Separation Science and Technology, 22: 12, 2403 — 2411
To link to this Article: DOI: 10.1080/01496398708057194
URL: http://dx.doi.org/10.1080/01496398708057194

SEPARATION SCIENCE AND TECHNOLOGY, **22**(12), pp. 2403-2411, 1987

NOTE

Fast Algorithm for the Conversion of *R* to Lambda Values in Field-Flow Fractionation

MARK. R. SCHURE

LABORATORY DATA PRODUCTS GROUP
DIGITAL EQUIPMENT CORPORATION
1 IRON WAY
MARLBOROUGH, MASSACHUSETTS 01752

Abstract

The numerical methodology needed to convert the retention ratio, R, to the dimensionless mean layer thickness, λ, commonly encountered in field-flow fractionation (FFF), is reviewed and a rigorous interpolation scheme which is suitable for any of the FFF techniques is presented. Computer implementation and error estimates are discussed in detail.

INTRODUCTION

A common operation found in all field-flow fractionation (FFF) experiments is the routine conversion of the fractogram on a time scale to a fractogram on a λ scale. Although this operation would at first seem trivial due to the wealth of theory which exists, in fact this operation is nontrivial because there is virtually no analytical form available for this operation with sufficient accuracy over the entire elution range. This is in contrast to a number of mathematical equations which express R, the retention ratio, as an analytical function of λ, the dimensionless mean layer thickness. Once the fractogram is converted to an elution scale in the λ domain, it is easy to convert to molecular weight or particle size for the techniques of thermal FFF (*1*), sedimentation FFF (*2*), electrical FFF (*3*), and flow FFF (*4*).

A number of numerical techniques exist for the interpolation of values in the independent variable, given the dependent variable value and a relationship relating one to the other. In this note we consider a most rigorous, noniterative approach to this inversion problem and provide specific details of its implementation with respect to any of the field-flow fractionation techniques.

THEORY

An experimental fractogram records the concentration or some detector response similar to concentration as a function of time. The first transformation to be made is to convert the retention time, t_r, to the dimensionless parameter R, the retention ratio, which is related to retention time by

$$R = t_0/t_r \tag{1}$$

where t_0 is the void time. From a fundamental standpoint, R is related to the average velocity of zone migration by

$$R = \frac{<c(x) \cdot v(x)>}{<c(x)><v(x)>} \tag{2}$$

where $c(x)$ is the concentration of the solute zone as a function of x, the distance above the lower channel wall, and $v(x)$ is the fluid velocity as a function of x. The operator $<>$ denotes a cross-sectional average. The quantity R can be analytically expressed as a function of λ:

$$R = 6\lambda \coth\left[\frac{1}{2\lambda}\right] - 12\lambda^2 \tag{3}$$

Although this expression has been found adequate for sedimentation FFF, flow FFF, and electrical FFF due to the uniform viscosity across the channel width, further research has shown that a much more complex expression is required for thermal FFF (*1*) to include the effect of temperature dependence on fluid viscosity. We will now review the numerical procedures which have been used to obtain λ when R is given.

NUMERICAL METHODOLOGY

It is beyond the scope of this note to review all of the available methodology for finding an independent variable, given the dependent variable. Some of these methods are reviewed in Ref. 5. The most common method of finding λ, given R, is to express Eq. (3) as a root-finding problem, i.e.,

$$R - 6\lambda \coth\left[\frac{1}{2\lambda}\right] - 12\lambda^2 = 0 \tag{4}$$

For this technique it is desired to find λ, given R, so that Eq. (4) is satisfied. A common approach to this is to use the bisection method (5), where it is assumed that λ is between some range $\{\lambda_i,\lambda_{i+1}\}$ and this range is iteratively searched via an algorithmic procedure which typically halves the search range every iteration. This is continued until some criteria of convergence is achieved. Although this method gives reasonable accuracy, it is slow and is not robust because of difficulties in picking the search range. It is not obvious, *a priori,* what this range should be, and some form of lookup table is needed for proper operation.

Another method of finding λ, given R, is to minimize the left-hand side of Eq. (4) using a minimization algorithm, such as the Newton-Raphson method (5), whereby an initial guess of λ is made, given R, and iteration is performed until some convergence criterion is met. Preliminary work has shown that even with a very robust routine from a major mathematical software vendor, convergence was not achieved for some R values greater than 0.3, even when the exact λ was used as the initial guess. In all of the methods similar to the Newton-Raphson method, there is no guarantee of convergence here even when the parameter to be obtained is under numerical constraints.

Another method to be considered and extensively investigated in preliminary work is the use of an interpolating polynomial of the form

$$\lambda = a_0 + a_1R + a_2R^2 + a_nR^n \tag{5}$$

where the coefficients a_0 through a_n are chosen so that the least-squares criterion is minimized:

$$\min = \sum_i (\lambda_i - [a_0 + a_1R_i + a_2R_i^2 + a_nR_i^n])^2 \tag{6}$$

In practice, this method may only be approached as a piecewise approximation, i.e., a valid range of R (and hence λ) is chosen so that

unique coefficients are determined over a narrow range. This is noted to occur because λ changes drastically with R when R approaches unity, yet λ changes only slowly with R at low values of R. This is shown in Fig. 1 using Eq. (3) with the axes reversed for plotting. This aspect also explains some of the difficulty encountered with the Newton-Raphson method. A further difficulty with the polynomial method is that most polynomials are numerically unstable for large values of the independent variable (*5*).

Along the lines of a polynomial expression, such as that used in Eq. (5), a more rigorous method is that of piecewise interpolation with Padé approximation (*5*), whereby the ratio of polynomials is used as a basis function over a narrow range of R:

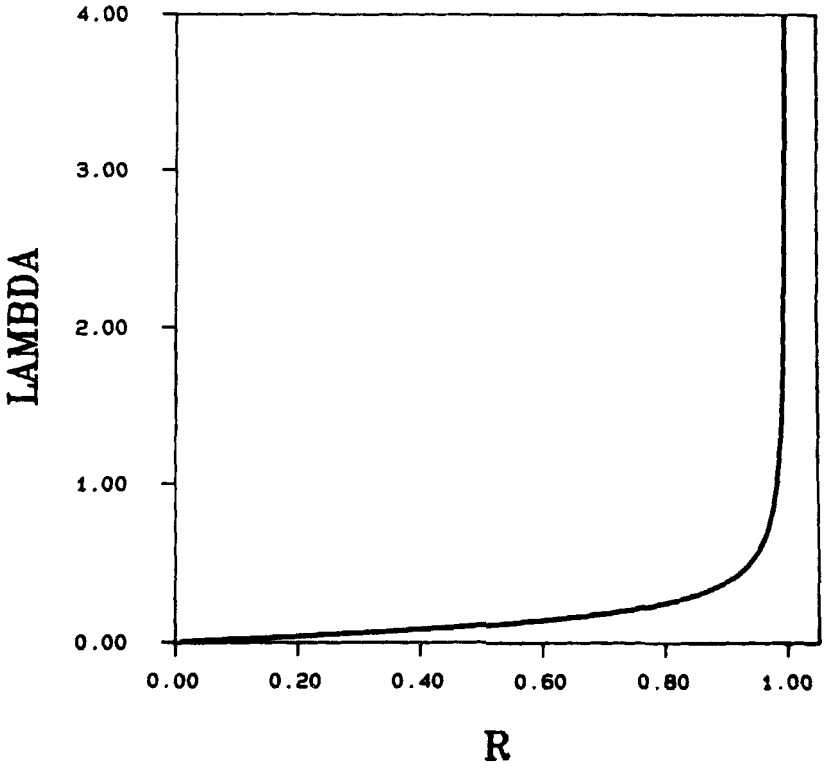

FIG. 1. λ as a function of R using the relationship established by Eq. (3).

$$\lambda = \frac{a_0 + a_1R + a_2R^2 + a_nR^n}{1 + b_1R + b_2R^2 + b_nR^n} \quad (7)$$

This approach has been used in preliminary work with some degree of success although six piecewise ranges must be employed to obtain accuracy of 1 part in 1000 in the higher λ range with a quadratic basis in the numerator and denominator of Eq. (7). A problem with the functional form of Eq. (7) is that a great deal of work is needed to optimize the coefficients, and the maximum degree of the basis functions is not clearly defined although Eq. (7) can be written so that orthogonal expressions are used. When orthogonal functions are used in Eq. (7), further accuracy can be obtained by increasing the basis degree (until numerical instability sets in) because the coefficients a and b are independent of the number of terms used in the numerator and the denominator. This approach is also inflexible in research because one would like to be able to change easily the functional form of R, as a function of λ, without the burden of determining a new set of coefficients. Toward this end, an extensively tested algorithm combining a sophisticated lookup table with quadratic interpolation in R rather than in λ is presented.

R TO LAMBDA CONVERSION

All software used here was written in FORTRAN 77 and executed on a microVAX II (Digital Equipment Corp., Maynard, Massachusetts); the code will also run on a small microcomputer since it is short in length and requires little memory. The software described here is in the form of a subroutine. The main program calls this subroutine and passes an array of R values and the number of R values to be analyzed. The subroutine passes back the λ values in an array. The R values are assumed to be monotonically decreasing, as would be the case in a normal fractogram. The R values are also assumed to be in the range $0.9999 > R > 0.00001$. If these conditions are not met, an error code is passed back to the calling program.

The first step taken in the program is to calculate a table of λ values and the corresponding R values via Eq. (3), although in research applications where Eq. (3) may not be used, the same procedure is used. Nonuniform spacing is used in this table as indicated in Table 1. This is done to ensure that the polynomial interpolation is performed on a very finely spaced grid where known deviations from quadratic behavior lie. The search is then started on the R lookup table data searching from high R values to low R values until R values are found such that $R_i > R$ and R_{i+1}

TABLE 1
Spacing and Number of Data Values in the Lookup Table

Range in λ	Corresponding R range	Number of data in lookup table
$1 \leqslant \lambda \leqslant 10$	$0.98372 \leqslant R \leqslant 0.99983$	451
$0.1 \leqslant \lambda \leqslant 0.995$	$0.48005 \leqslant R \leqslant 0.98356$	198
$0.01 \leqslant \lambda \leqslant 0.099$	$5.8800 \times 10^{-2} \leqslant R \leqslant 0.47643$	90
$0.0001 \leqslant \lambda \leqslant 0.0099$	$5.9988 \times 10^{-4} \leqslant R \leqslant 5.8224 \times 10^{-2}$	99
$0.0000001 \leqslant \lambda \leqslant 0.000099$	$5.9999 \times 10^{-7} \leqslant R \leqslant 5.9388 \times 10^{-4}$	99

$\leqslant R$, where the subscript denotes the index of R in the lookup table. If this condition is not found, an error condition is reported. Next, Lagrange interpolation (*5*) is performed to find the local quadratic equation

$$R = a_0 + a_1\lambda + a_2\lambda^2 \tag{8}$$

The formulas for Lagrange interpolation expressed in the form of this problem are

$$a_0 = b_0\lambda_{i+1}\lambda_{i+2} + b_1\lambda_i\lambda_{i+2} + b_2\lambda_i\lambda_{i+1} \tag{9}$$

$$a_1 = b_0(-\lambda_{i+1} - \lambda_{i+2}) - b_1(\lambda_i + \lambda_{i+2}) - b_2(\lambda_i + \lambda_{i+1}) \tag{10}$$

$$a_2 = b_0 + b_1 + b_2 \tag{11}$$

and

$$b_0 = \frac{R_i}{(\lambda_i - \lambda_{i+1}) \cdot (\lambda_i - \lambda_{i+2})} \tag{12}$$

$$b_1 = \frac{R_{i+1}}{(\lambda_{i+1} - \lambda_i) \cdot (\lambda_{i+1} - \lambda_{i+2})} \tag{13}$$

$$b_2 = \frac{R_{i+2}}{(\lambda_{i+2} - \lambda_i) \cdot (\lambda_{i+2} - \lambda_{i+1})} \tag{14}$$

Upon performing these calculations, the problem is set up as a root problem so that

$$a_2\lambda^2 + a_1\lambda + (a_0 - R) = 0 \tag{15}$$

Although solution of Eq. (15) appears to be straightforward, via the quadratic formula

$$\lambda = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_2(a_0 - R)}}{2a_2} \tag{16}$$

a number of conditions exist which warrant special attention. First, if a_2 is equal to 0, which may happen with very low λ values, although this has never been observed using Eq. (3) as the generator, Eq. (16) will not work; the problem is reduced to a linear solution:

$$\lambda = \frac{R - a_0}{a_1} \tag{17}$$

and λ is found for this R value. Using Eq. (3) as the generator, in practice one root of Eq. (16) is found to be the correct one and it is a small matter of programming to test both roots and see which root (λ) lies in the range $\{\lambda_i,\lambda_{i+2}\}$. The present subroutine does check a number of other conditions, mostly as safety mechanisms. If both roots (λ's) are in the range $\{\lambda_i,\lambda_{i+2}\}$, then these values are fed into Eq. (3) (or other suitable form), and both R values are calculated. The λ with the closest calculated R value to the given R to be converted is then used as the found λ value. Also tested is the case where the quantity inside the square-root operator in Eq. (16) is negative, indicating the roots are complex; if this occurs, then an error code is passed to the calling program and the program operation is aborted; however, this should never occur and has never been observed in practice. The search pointer for the next R value in the lookup table starts where the last value was found, avoiding the complete search for every R value.

ERROR ANALYSIS

A separate error analysis program is written which generates 7000 values of λ, equally spaced across the logarithm of λ between 1.0 and -5.75 inclusive. The coresponding R values, calculated from Eq. (3), are passed to the conversion subroutine where λ is calculated. The percentage difference between λ calculated from the conversion subroutine and λ from the error analysis program is shown in Fig. 2, plotted as a function of the logarithm of λ. As can be seen from Fig. 2, the error is totally negligible for the majority of the elution range. In the infrequently

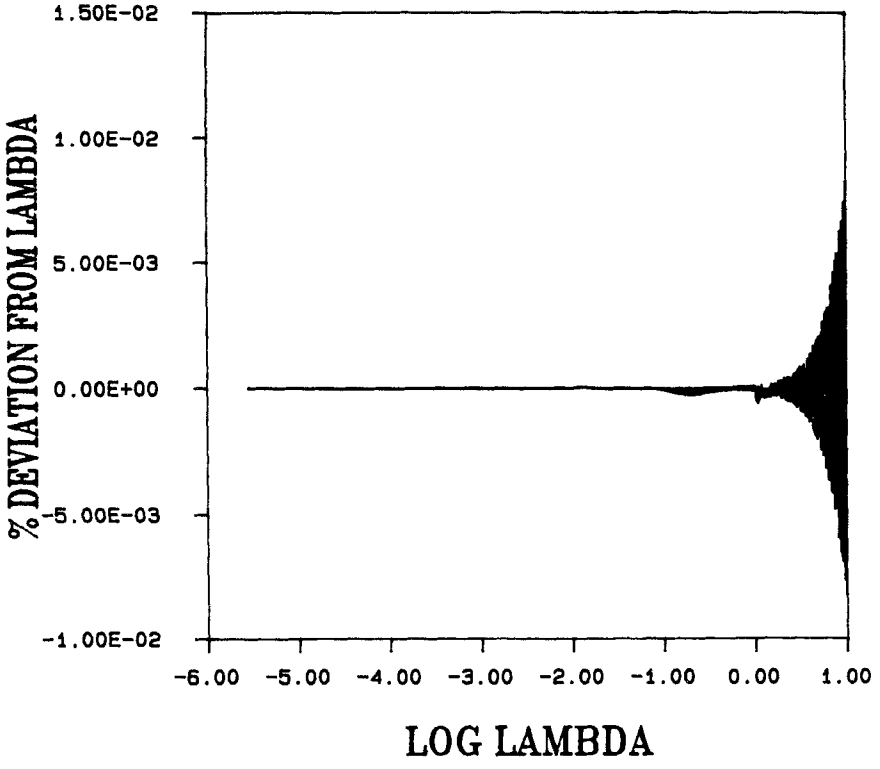

FIG. 2. The percentage error in λ as a function of the logarithm of λ.

used region near the void peak (large R and λ), λ has a maximum error of approximately 1 part in 10,000. This is of sufficient accuracy for any FFF experiment. One further consideration here is that occasionally one will call this subroutine more than once, for obtaining isolated λ values, rather than running this as an array-oriented subroutine. In the subroutine developed for this work, the lookup table is created only once. Subsequent calls to the subroutine avoid the recalculation of the table and set the lookup pointer back to the top of the arrays. This provides fast operation even for isolated point by point conversion.

CONCLUSIONS

The algorithm described above is seen to be a high accuracy, fast solution to the general problem of R to λ conversion which is performed

on all quantitative FFF experiments. It avoids iterative calculation and initial parameter estimation, and can be programmed in a minimum of memory space with very modest array storage requirements. All software described in this communication is available from the author upon request.

REFERENCES

1. J. J. Gunderson, K. D. Caldwell, and J. C. Giddings, *Sep. Sci. Technol., 19,* 667 (1984).
2. F. S. Yang, K. D. Caldwell, and J. C. Giddings, *J. Colloid Interface Sci., 92,* 81 (1983).
3. L. F. Kesner, K. D. Caldwell, and J. C. Giddings, *Anal. Chem., 48,* 1834 (1976).
4. J. C. Giddings, F. J. Yang, and M. N. Myers, *Anal. Biochem., 81,* 395 (1977).
5. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes,* Cambridge University Press, New York, 1986, Chap. 3.

Received by editor January 26, 1987
Revised March 5, 1987