*Journal of Thermal Analysis, 36 (1990) 145-169*

# A COMPUTATION SCHEME TO EVALUATE DEBYE AND TARASOV EQUATIONS FOR HEAT CAPACITY COMPUTATION WITHOUT NUMERICAL INTEGRATION

*R. Pan,* M. Varma-Nair* and *B. Wunderlich*

CHEMISTRY DIVISION, OAK RIDGE NATIONAL LABORATORY, OAK RIDGE, TENNESSEE 37831-6197;
DEPARTMENT OF CHEMISTRY, UNIVERSITY OF TENNESSEE, KNOXVILLE, TENNESSEE 37996-1600

*PROCTER AND GAMBLE, MIAMI VALLEY LABORATORIES, P.O. BOX 398707, CINCINNATI. OHIO 45239-8707; U.S.A.

(Received June 11, 1989)

The *ATHAS* computation scheme of heat capacities of solid, linear macromolecules has been developed for an IBM-compatible microcomputer on a Lotus 1-2-3 based software. In this effort the Debye functions that can only be integrated numerically have been approximated by polynomials and logarithmic polynomials, which describe these functions to an accuracy of better than ±0.1%. Heat capacities can now be computed much faster and with greater ease.

At *ATHAS*, our laboratory for Advanced THermal AnalysiS, the linking of the experimental heat capacities of nearly 100 linear macromolecules to an approximate vibrational spectrum has been completed [1]. It is based on the separation of the vibrational spectrum into skeletal (inter and intra molecular vibrations) and group vibrations. At present, the experimental heat capacities form the best means of approximating the strongly coupled, low frequency skeletal vibrations which dominate heat capacity in the 10 to 100 K region. From the knowledge of the experimental heat capacities fitted to two characteristic temperatures $\theta_1$ and $\theta_3$ (representing the intra-molecular and inter-molecular vibrations) and the group vibrations derived from normal-mode calculations based on i.r. and Raman data, it is possible to compute heat capacities over a wide temperature range (0 to 1000 K).

*John Wiley & Sons, Limited, Chichester*
*Akadémiai Kiadó, Budapest*

The values of $\theta_1$ and $\theta_3$ represent the upper frequencies of suitably chosen distributions. The $\theta$-temperatures are related to frequency via $\theta = h\nu/k$, where $h$ and $k$ are Planck's and Boltzmann's constants, respectively. A $\theta$ of 1 K corresponds to a frequency of 2.08 x $10^{10}$ Hz or 0.695 $\mathrm{cm}^{-1}$. The theory behind the computations is well documented is several publications from our laboratory [2-4]. The computer programs for these calculations have been described in detail [3]. The skeletal vibrations for linear macromolecules are approximated by a Tarasov model which is a combination of one and three dimensional Debye terms and is given by the equation [5]:

$$C_v\,/NR = D_1(\Theta_1/T\,)\,.\,(\Theta_3/\Theta_1)\,[D_1(\Theta_3/T)\,.D_3(\Theta_3/T\,)] \qquad (1)$$

where $D_1$ and $D_3$ are represented as

$$D_1(\Theta_1/T) = (2T/\Theta_1)\int_0^{\Theta_1/T}\frac{(\Theta/T)\,\mathrm{d}(\Theta/T)}{\exp(\Theta/T)-1} - \frac{(\Theta_1/T)}{\exp(\Theta_1/T)-1} \qquad (2)$$

$$D_3(\Theta_3/T) = (12T^3/\Theta_3^3)\int_0^{\Theta_3/T}\frac{(\Theta/T)^3\,\mathrm{d}(\Theta/T)}{\exp(\Theta/T)-1} - \frac{3(\Theta_3/T)}{\exp(\Theta_3/T)-1} \qquad (3)$$

The model seems to hold for most linear macromolecules, but its limits are reached in case of polymers with phenylene groups [1e] or alternating heavy and light mass backbone-units [1f]. The narrow frequency distributions of the group vibrations are inverted to their heat capacity contributions and vice-versa by Einstein functions represented by:

$$C_v\,/NR = \frac{(\Theta/T\,)^2\exp(\Theta/T\,)}{[\exp(\Theta/T\,)-1\,]^2} \qquad (4)$$

while wider frequency ranges are represented by a box distribution function which assumes a linear distribution function between lower and upper frequencies L and U is given by:

$$C_v / NR = B(\Theta_U / T, \Theta_L / T) =$$

$$\Theta_U / (\Theta_U - \Theta_L)[D_1(\Theta_U / T) - (\Theta_L / \Theta_U) / D_1(\Theta_L / T)] \tag{5}$$

While the Einstein functions are easily calculated using even a pocket calculator, a major difficulty arises in the evaluation of the integrals of the Einstein function involved in the Debye functions.

Functions 1, 2, 3 and 5 involve integrals of the type

$$C_v / NR = \int_0^B \frac{X^n}{\exp(X) - 1} \mathrm{d}X \tag{6}$$

that cannot be solved in closed form [3]. Although tables for Debye functions have been published [6-8], the use of these is quite involved, especially for the Tarasov equation. Cheban *et al.* [3] solved these integrals making use of the Clenshaw-Curtiss method [9], using a double-adaptive strategy [10] and published the computer programs written in FORTRAN. The application of the various methods was described by Lau *et al.* [3] and the latest progress is reported in Ref. [1a].

Although these computations are highly accurate, they require long computation times (e.g. about 80 s CPU time for polyethylene and 120 s for poly(vinylidene fluride). In addition, these computations are limited to the main-frame computer. These difficulties precluded the wider applications of our ATHAS (Advanced Thermal Analysis System).

In an effort to adapt the computation scheme to IBM compatible microcomputers we developed the approximate method of evaluation of Debye functions described in the present paper. Also given are some example calculations, showing the accuracy of the new computations of heat capacity.

## Result and discussion

The computation scheme was adapted to the IBM PC by dividing the Debye functions into several intervals and the fitting each of these by least squares techniques into either a polynomial or a polynominal logarithmic function. Computations of polynomial function are rather fast on microcomputers and, as will be shown below, the error can easily be made less than ±0.1%.

In Eq. (6) we assume $X$ to be $\Theta / T$. Then for $X \leq 1$, the three-dimensional Debye-function can be fitted to:

$$D_3(X) = -4.635519X^2 - 2.333748 \times 10^{-3}X + 1.000327 \tag{7}$$

The Root-Mean-Square (RMS) deviation for 48 data points in the range from $X = 0.1$ to 1.05 with an increment of 0.02 is ±0.007% For $1 < X < 1$, the three-dimensional Debye-function can next be fitted to:

$$D_3(X) =$$

$$= 6.482976 \times 10^{-3}X^3 - 5.684513 \times 10^{-2}X^2 - 1.473625 \times 10^{-3}X + 1.003855 \tag{8}$$

with an RMS deviation for 66 data points in the range from $X = 0.9$ to 4.1 with an increment of 0.05 of ±0.045%. This is followed for $4 \leq X \leq 6$, with

$$D_3(X) = 1.566803 \times 10^{-2}X^2 - 2.754144 \times 10^{-1}X + 1.354056 \tag{9}$$

giving an RMS deviation for 44 data points in the range from $X = 3.9$ to 6.1 with an increment of 0.05 of ±0.012%. For $6 < X \leq 15$, the fitted equation is:

$$D_3(X) =$$

$$\exp(4.664932 \times 10^{-1}V^3 - 3.697541V^2 + 6.790459V - 4.305865) \tag{10}$$

with $V = \ln(X)$.The RMS deviation for 92 data points in this range from $X = 5.9$ to 15 with an increment of 0.1 is ±0.024%. For $X > 15$, the well-known cubic, low-temperature approximation can be used with a similarly low error:

$$D_3(X) = 7.792454 \times 10^{-1}X^{-3} \tag{11}$$

Similar to the three-dimensional function, we also applied the least square fitting technique to the one-dimensional Debye function. For $X \leq 1$,

$$D_1(X) = -2.594584 \times 10^{-2}X^2 - 1.215488 \times 10^{-3}X + 1.000176 \qquad (12)$$

The RMS deviation of this fitting for 48 data points in the range form $X$ = 0.1 to 1.1 with increments of 0.02 is ±0.004%. For $1 < X \leq 4$, the one-dimensional Debye function can be represented by

$$D_1(X) =$$

$$= 3.346877 \times 10^{-3}X^3 - 3.250229 \times 10^{-2}X^2 + 1.795002X + 1.795002X + 1.000493 \qquad (13)$$

with an RMS deviation of this fitting for 64 data points in the range from $X = 0.9$ to 4.1 with increments of 0.05 of ±0.014%. This is followed for $4 < X \leq 6$, by:

$$D_1(x) = 6.779643 \times 10^{-3}X^2 - 1.549074 \times 10^{-1}X + 1.212911 \qquad (14)$$

and an RMS deviation of this fitting for 23 data points in the range from $X = 3.9$ to 6.1 with increments of 0.1 ±0.015%. Next, for $6 < X < 14$ the fitted function is:

$$D_1(X) =$$

$$= \exp(1.134231 \times 10^{-1}V^3 - 8.418919 \times 10^{-1}V^2 + 1.085236V - 5.331556 \times 10^{-1}) \qquad (15)$$

where $V = \ln(X)$. The RMS deviation for 83 data points in this range from $X = 5.9$ to 14.1 with increments of 0.1 is ±0.019%. Finally, for $14 \leq X < 21$, the fitted equation is :

$$D_1(X) = 3.66906 \leq 10^{-2}X^{-2} - 3.293774 \times 10^{-1}X^{-1} - 1.003587 \times 10^{-4} \qquad (16)$$

with an RMS deviation for 45 data points in the range from $X = 13.9$ to 21.9 with increments of 0.1 of ±0.000%. For $X \geq 21$, the well-known low temperature, linear approximation of the Debye function [3] is used.

$$D_1(X) = 3.2898688X^{-1} \qquad (17)$$

The Appendix shows the program named CPTOT which permits the calculation of heat capacity contributions of skeletal vibrations via the Tarasov function (Eq. 1) and of group vibrations via Einstein functions and box distributions (Eqs 4 and 5). This program, written in C language, is a remarkable improvement in terms of calculation speed over the old program using the integration algorithm based on the Clenshaw-Curtiss and Patterson methods [11]. For example, to calculate $C_p$ of poly(ethylene terephthalate), the old program [3] requires 85 seconds of CPU time on an IBM 3081D mainframe computer. The new program, after being compiled by Microsoft C compiler (version 3.0), takes only 5 seconds on a Compaq Personal Computer (equipped with 20-MHz 80386 Intel microprocessor and 80387 math coprocessor). Having established that the equations represent the Debye functions well, a single new computation scheme for Eq. (1) was developed using menu-driven programming. To compare the accuracies of the two computation schemes, the one-dimensional Debye function was computed using a $\theta_1$ of 100 K [$\theta_3$ = 0.001 K (since the computation does not accept $\theta_3$ = K for obvious reasons). The number of vibrators, $N$, was chosen to be one, for simplicity. The ratio of $\theta_3/\theta_1$ in the Tarasov equation (Eq. 1) is in this case $10^{-5}$ and the calculated function represents thus a one-dimensional Debye term. For the three-dimensional Debye function $\theta_1$ and $\theta_3$ were both set to 100 K; this leads to $\theta_3/\theta_1 = 1$ and cancellation of the $D_1$-term in Eq. (1). The results of the computation are shown is columns 5 of Tables 1 and 2. Column 3 shows the Debye functions obtained using the main-frame computation scheme. The error is in both cases 0.1% or less.

Similarly a computation scheme for group vibrations was developed using Eqs (4) and (5). The Einstein function was calculated using $\Theta_E$ = 100 K for $N = 1$, while a value of $\Theta_L$ = 50 K and $\Theta_Y$ = 100 K was used to determine the contribution of box distributions (Eq. 5) to the heat capacity. Tables 3 and 4 depict the results, together with those computed using the previous method. The error between the two computations was again less than 0.1% and could be taken as negligible as the present data is rounded at the third decimal position. The experimental data to which these computations are applied have typical errors of the order of magnitude of 1%.

To check the quality of the above fitted equations and the resulting program, calculation results of polyethylene by this new program are compared with that by the prior algorithm [1j and 12]. Table 5 lists the deviations of heat capacity contributions of skeletal vibration, group vibration, $C_v$ and $C_p$ of polyethylene calculated by the new program.

The standard deviation of the skeletal vibration contribution is $-0.001\pm 0.010\%$ between 0.1and 1000K.

**Table 1** Contribution of 1-dimensional Debye function to the heat capacity

| Temp., K | Θ/T | $D_1(\Theta_1/T)$ | Calculated[a] $C_v$, J/(kmol) | Calculated[b] $C_v$, J/(kmol) | Error, % |
|---|---|---|---|---|---|
| 0.1 | 1000 | 0.0032898 | 0.0274 | 0.027 | 0 |
| 0.2 | 500 | 0.0065797 | 0.0547 | 0.055 | 0 |
| 0.3 | 333.333 | 0.0098696 | 0.0821 | 0.082 | 0 |
| 0.4 | 250 | 0.0131594 | 0.1094 | 0.109 | 0 |
| 0.5 | 200 | 0.0164493 | 0.1368 | 0.137 | 0 |
| 0.6 | 166.667 | 0.0197392 | 0.1641 | 0.164 | 0 |
| 0.8 | 125 | 0.0263189 | 0.2188 | 0.219 | 0 |
| 0.9 | 111.111 | 0.0296088 | 0.2462 | 0.246 | 0 |
| 1.0 | 100 | 0.0328986 | 0.2735 | 0.274 | 0 |
| 2.0 | 50 | 0.0657973 | 0.5471 | 0.547 | 0 |
| 4.0 | 25 | 0.1315947 | 1.0941 | 1.094 | 0 |
| 5.0 | 20 | 0.1644934 | 1.3677 | 1.368 | 0.022 |
| 8.0 | 12.5 | 0.263134 | 2.1877 | 2.187 | -0.032 |
| 10.0 | 10.0 | 0.328433 | 2.7306 | 2.731 | 0.014 |
| 20.0 | 5.0 | 0.6078342 | 5.0537 | 5.054 | 0.005 |
| 25.0 | 4.0 | 0.7016666 | 5.8338 | 5.835 | +0.02 |
| 40 | 2.5 | 0.8540155 | 7.1005 | 7.102 | -0.02 |
| 50 | 2 | 0.9008593 | 7.4900 | 7.490 | 0 |
| 80 | 1.25 | 0.9585453 | 7.9696 | 7.969 | -0.007 |
| 100 | 1.0 | 0.9730325 | 8.0901 | 8.090 | 0 |
| 200 | 0.5 | 0.9931072 | 8.2569 | 8.257 | 0.001 |
| 250 | 0.4 | 0.9955767 | 8.2775 | 8.277 | -0.006 |
| 400 | 0.25 | 0.9982671 | 8.2998 | 8.300 | 0.002 |
| 500 | 0.20 | 0.9988902 | 8.3051 | 8.305 | -0.001 |
| 800 | 0.125 | 0.9995618 | 8.3107 | 8.311 | 0.003 |
| 1000 | 0.100 | 0.9997223 | 8.3120 | 8.313 | -0.012 |

[a] $C_v$ computed using the programs given in Ref. 3. The integrations made use of the algorithm based on Patterson's method [1a] (see column to the left).

[b] $C_v$ computed using *ATHAS* computation scheme on a PC. A value $\Theta_3 = 0.001$K, $\Theta_1 = 100$K and N = 1 was used

The group vibration contribution show a somewhat larger percentage deviation at low temperatures, due to less tolerance of curve fluctuation in the narrow frequency band box distributions in the ranges 14 $<\Theta/T21$ and $\Theta/T$

Table 2 Contribution of 3-dimensional Debye function to the heat capacity

| Temp., K | $\Theta/T$ | $D_3(\Theta_3/T)$ | Calculated[a] $C_V$, J/(K mol) | Calculated[b] $C_V$, J/(K mol) | Error, % |
|---|---|---|---|---|---|
| 0.1 | 1000 | 0.00000000 | 0.0000 | 0.000 | 0 |
| 0.2 | 500 | 0.0000006 | 0.0000 | 0.000 | 0 |
| 0.3 | 333.333 | 0.0000021 | 0.0000 | 0.000 | 0 |
| 0.4 | 250 | 0.0000031 | 0.0000 | 0.000 | 0 |
| 0.5 | 200 | 0.0000090 | 0.0000 | 0.000 | 0 |
| 0.6 | 166.667 | 0.0000168 | 0.0001 | 0.000 | 0 |
| 0.8 | 125.0 | 0.0000267 | 0.0002 | 0.000 | 0 |
| 0.9 | 111.111 | 0.0000398 | 0.0003 | 0.000 | 0 |
| 1.0 | 100 | 0.0000779 | 0.0006 | 0.001 | 0 |
| 2.0 | 50 | 0.0006234 | 0.0051 | 0.005 | 0 |
| 4.0 | 25 | 0.0049873 | 0.0414 | 0.041 | 0 |
| 5.0 | 20 | 0.0097407 | 0.0809 | 0.081 | 0 |
| 8.0 | 12.5 | 0.0397017 | 0.3309 | 0.330 | 0 |
| 10.0 | 10.0 | 0.0758210 | 0.6304 | 0.630 | 0 |
| 20.0 | 5.0 | 0.3686350 | 3.0649 | 3.065 | 0.003 |
| 25.0 | 4.0 | 0.5030594 | 4.1826 | 4.183 | 0.009 |
| 30 | 3.333 | 0.6082474 | 5.05717 | 5.051 | -0.122 |
| 40 | 2.5 | 0.7458531 | 6.20127 | 6.204 | 0.0045 |
| 50 | 2 | 0.8254081 | 6.8627 | 6.863 | 0.004 |
| 80 | 1.25 | 0.9260335 | 7.6993 | 7.698 | -0.016 |
| 100 | 1.00 | 0.9517321 | 7.9130 | 7.912 | -0.012 |
| 200 | 0.5 | 0.9876107 | 8.2113 | 8.211 | 0 |
| 250 | 0.4 | 0.9920454 | 8.2481 | 8.228 | 0 |
| 400 | 0.25 | 0.9968819 | 8.2884 | 8.288 | 0 |
| 500 | 0.20 | 0.9980028 | 8.2977 | 8.298 | 0 |
| 800 | 0.125 | 0.9992179 | 8.3078 | 8.309 | 0.014 |
| 1000 | 0.10 | 0.9995001 | 8.3102 | 8.311 | 0.009 |

[a] $C_V$ computed using the programs given in Ref. 3. The integrations given in Ref. 3. The integrationsmade use of the algorithm based on Patterson's method [1a](see column to the left

[b] $C_V$ computed using *ATHAS* computation scheme on a PC. A value of $\Theta = 1.00$K, $\Theta_3 = 100$K and N = 1 was used

**Table 3** Contribution of Einstein function to the heat capacity

| Temp., K | $\Theta/T$ | $E(\Theta/T)$ | Calculated[a] $C_v$, J/(K mol) | Calculated[b] $C_v$, J(K mol) | Error, % |
|---|---|---|---|---|---|
| 0.1 | 1000 | 0 | 0 | 0.000 | 0 |
| 0.2 | 500 | 0 | 0 | 0.000 | 0 |
| 0.3 | 333.333 | 0 | 0 | 0.000 | 0 |
| 0.4 | 250 | 0 | 0 | 0.000 | 0 |
| 0.5 | 200 | 0 | 0 | 0.000 | 0 |
| 0.6 | 166.667 | 0 | 0 | 0.000 | 0 |
| 0.7 | 125.0 | 0 | 0 | 0.000 | 0 |
| 0.8 | 111.111 | 0 | 0 | 0.000 | 0 |
| 1.0 | 100 | 0 | 0 | 0.000 | 0 |
| 2.0 | 50 | 0.0000 | 0 | 0.000 | 0 |
| 4.0 | 25 | 0.0000000 | 0.000000 | 0.000 | 0 |
| 5.0 | 20 | 0.00000082 | 0.0000066 | 0.000 | 0 |
| 8.0 | 12.5 | 0.0000465835 | 0.0038731 | 0.005 | 0 |
| 10.0 | 10.0 | 0.0045404 | 0.0377750 | 0.038 | 0 |
| 20.0 | 5.0 | 0.1707418 | 1.4196056 | 1.420 | 0 |
| 25.0 | 4.0 | 0.3040873 | 2.5282854 | 2.528 | 0 |
| 40 | 2.5 | 0.60889 | 5.0625189 | 5.063 | 0 |
| 50 | 2 | 0.7240616 | 6.0200948 | 6.020 | 0 |
| 80 | 1.25 | 0.8793662 | 7.3113505 | 7.311 | 0 |
| 100 | 1.00 | 0.9206736 | 7.6547933 | 7.655 | 0 |
| 200 | 0.5 | 0.9794245 | 8.1432685 | 8.143 | 0 |
| 250 | 0.4 | 0.9867726 | 8.2043634 | 8.204 | 0 |
| 400 | 0.25 | 0.9948079 | 8.2711712 | 8.271 | 0 |
| 500 | 0.20 | 0.9966733 | 8.2866812 | 8.287 | 0 |
| 800 | 0.125 | 0.9986999 | 8.3035231 | 8.304 | 0 |
| 1000 | 0.10 | 0.9991671 | 8.3074156 | 8.307 | 0 |

[a] $C_v$ computed using the main-frame computation scheme of *ATHAS*

[b] $C_v$ computed using *ATHAS* computation scheme on a PC. Calculations were done using $\Theta_E$ = 100K and $N$ = 1

**Table 4** Contribution of box distribution function to the heat capacity

| Temp.,K | $\theta U/T$ | $\theta L/T$ | Calculated box distribution function[a], $C_v/R$ | $C_v$ | Box distribution[b], $C_v$ | Error,% |
|---|---|---|---|---|---|---|
| 0.1 | 1000 | 500 | -0.0000001 | 0.000 | 0.000 | 0 |
| 0.2 | 500 | 250 | 0.0000000 | 0.000 | 0.000 | 0 |
| 0.3 | 333.333 | 166.666 | 0.0000000 | 0.000 | 0.000 | 0 |
| 0.4 | 250 | 125 | -0.0000001 | 0.000 | 0.000 | 0 |
| 0.5 | 200 | 100 | 0.0000000 | 0.000 | 0.000 | 0 |
| 0.6 | 166.666 | 83.333 | 0.0000000 | 0.000 | 0.000 | 0 |
| 0.7 | 142.857 | 71.428 | 0.0000000 | 0.000 | 0.000 | 0 |
| 0.8 | 125 | 62.5 | 0.0000000 | 0.000 | 0.000 | 0 |
| 1.0 | 100 | 50 | 0.0000000 | 0.000 | 0.000 | 0 |
| 2.0 | 50 | 25 | 0.0000000 | 0.000000 | 0.000 | 0 |
| 4.0 | 25 | 12.5 | 0.0000554 | 0.0004606 | 0.000 | 0 |
| 5.0 | 20 | 10 | 0.0005538 | 0.0046044 | 0.004 | -15.0 |
| 8.0 | 12.5 | 6.25 | 0.0164604 | 0.1368573 | 0.136 | -0.588 |
| 10.0 | 10 | 5 | 0.0490318 | 0.407667 | 0.408 | 0 |
| 20.0 | 5 | 2.5 | 0.3616529 | 3.0069052 | 3.006 | -0.03 |
| 25.0 | 4 | 2 | 0.5024739 | 4.1777388 | 4.181 | +0.08 |
| 40.0 | 2.5 | 1.25 | 0.7494857 | 6.2314789 | 6.234 | +0.041 |
| 50.0 | 2.0 | 1.0 | 0.8286860 | 6.8899779 | 6.890 | 0 |
| 80.0 | 1.25 | 0.625 | 0.9278162 | 7.7141793 | 7.713 | -0.014 |
| 100 | 1.00 | 0.5 | 0.9529578 | 7.9232152 | 7.923 | 0 |
| 200 | 0.5 | 0.25 | 0.9879473 | 8.2141298 | 8.214 | 0 |
| 250 | 0.4 | 0.20 | 0.9922632 | 8.2500136 | 8.249 | -0.012 |
| 400 | 0.25 | 0.125 | 0.9969724 | 8.2891675 | 8.288 | -0.013 |
| 500 | 0.20 | 0.10 | 0.9980581 | 8.2981944 | 8.298 | 0 |
| 800 | 0.125 | 0.0625 | 0.9992507 | 8.30811011 | 8.308 | 0 |
| 1000 | 0.10 | 0.050 | 0.9998153 | 8.3128043 | 8.311 | -0.021 |

[a] $C_V$ computed using the main frame computation scheme of *ATHAS*

[b] $C_V$ computed using *ATHAS* computation scheme on a PC. Calculations were done using $\theta_U = 100K$, $\theta_U = 50K$ and $N = 1$

**Table 5** Deviations of heat capacity contributions of polyethylene calculated by the fitted equations [Deviations %]

| Temp., K | Skelatal | Group | $C_v$ |
|---|---|---|---|
| 0.1 | -0.015 | | -0.015 |
| 0.2 | -0.015 | | -0.015 |
| 0.3 | -0.015 | | -0.015 |
| 0.4 | 0.000 | | 0.000 |
| 0.5 | -0.007 | | -0.007 |
| 0.6 | -0.001 | | -0.001 |
| 0.7 | -0.003 | | -0.003 |
| 0.8 | -0.003 | | -0.003 |
| 0.9 | -0.004 | | -0.004 |
| 1.0 | -0.003 | | -0.003 |
| 1.2 | -0.003 | | -0.003 |
| 1.4 | -0,003 | | -0.003 |
| 1.6 | -0.003 | | -0.003 |
| 1.8 | -0.003 | | -0.003 |
| 2 | -0.003 | | -0.003 |
| 3 | -0.015 | | -0.015 |
| 4 | +0.001 | | +0.001 |
| 5 | -0..007 | | -0.007 |
| 6 | -0.001 | | -0.001 |
| 7 | -0.003 | | -0.003 |
| 8 | -0.006 | | -0.006 |
| 9 | +0.003 | | +0.003 |
| 10 | +0.043 | | +0.043 |
| 15 | -0.010 | | -0.010 |
| 20 | -0.009 | | -0.009 |
| 25 | +0.033 | | +0.033 |
| 30 | -0.018 | | -0.018 |
| 40 | -0.0016 | | -0.016 |
| 50 | +0.006 | | +0.006 |
| 60 | +0.036 | -21.912 | +0.036 |
| 70 | -0.012 | -5.604 | -0.011 |
| 80 | -0.016 | +34.935 | -0.011 |
| 90 | -0.002 | +10.876 | +0.003 |
| 100 | +0.003 | -20.384 | -0.024 |
| 110 | -0.020 | +13.879 | +0.018 |
| 120 | -0.026 | -1.593 | -0.034 |
| 130 | +0.014 | -1.496 | +0.001 |
| 140 | -0.014 | +0.986 | -0.007 |
| 150 | -0.011 | -0.288 | -0.020 |
| 160 | -0.009 | -0.720 | -0.033 |

Table 5 Deviations of heat capacity contributions of polyethylene calculated by the fitted equations [Deviations %] (continued)

| Temp., K | Skeletal | Group | $C_v$ |
|---|---|---|---|
| 170 | -0.002 | -0.583 | -0.019 |
| 180 | +0.010 | +0.068 | +0.011 |
| 190 | +0.017 | +0.252 | +0.034 |
| 200 | +0.015 | +0.284 | +0.043 |
| 210 | +0.018 | +0.253 | +0.041 |
| 220 | +0.016 | +0.073 | +0.017 |
| 230 | +0.011 | +0.315 | +0.052 |
| 240 | +0.005 | +0.184 | +0.033 |
| 250 | 0.0000 | +0.058 | +0.010 |
| 260 | +0.001 | -0.190 | -0.036 |
| 270 | -0.001 | -0.133 | -0.029 |
| 280 | -0.009 | -0.289 | -0.074 |
| 290 | -0.010 | +0.024 | +0.003 |
| 300 | -0.012 | +0.036 | +0.001 |
| 310 | -0.012 | +0.091 | +0.018 |
| 320 | -0.011 | +0.000 | -0.008 |
| 330 | -0.012 | -0.077 | -0.038 |
| 340 | -0.012 | -0.066 | -0.030 |
| 350 | -0.015 | -0.028 | -0.016 |
| 360 | -0.011 | +0.083 | +0.020 |
| 370 | -0.015 | +0.143 | +0.050 |
| 380 | -0.009 | +0.176 | -0.061 |
| 390 | -0.011 | +0.018 | +0.043 |
| 400 | -0.007 | +0.039 | +0.013 |
| 410 | -0.006 | -0.020 | -0.015 |
| 420 | -0.009 | -0.119 | -0.059 |
| 430 | -0.006 | -0.156 | -0.075 |
| 440 | -0.003 | -0.109 | -0.053 |
| 450 | -0.001 | -0.067 | -0.033 |
| 460 | -0.001 | -0.026 | -0.013 |
| 470 | +0.002 | +0.032 | +0.017 |
| 480 | +0.001 | +0.020 | +0.011 |
| 490 | +0.002 | +0.019 | +0.010 |
| 500 | +0.004 | +0.011 | +0.010 |
| 510 | +0.006 | -0.018 | -0.004 |
| 520 | -0.004 | -0.036 | -0.021 |
| 530 | -0.001 | -0.046 | -0.026 |
| 540 | +0.001 | +0.050 | -0.027 |
| 550 | +0.001 | -0.048 | -0.026 |
| 560 | +0.000 | -0.045 | -0.026 |

**Table 5** Deviations of heat capacity contributions of polyethylene calculated by the fitted equations [Deviations %] (continued)

| Temp., K | Skeletal | Group | $C_v$ |
|---|---|---|---|
| 570 | +0.004 | -0.042 | -0.023 |
| 580 | +0.005 | -0.035 | -0.021 |
| 590 | +0.004 | -0.028 | -0.015 |
| 600 | +0.001 | -0.022 | -0.013 |
| 610 | +0.001 | -0.017 | -0.007 |
| 620 | +0.004 | -0.009 | -0.004 |
| 630 | +0.004 | -0.003 | 0.000 |
| 640 | +0.006 | +0.004 | +0.002 |
| 650 | +0.006 | +0.009 | +0.005 |
| 660 | +0.001 | +0.014 | +0.009 |
| 670 | +0.005 | +0.016 | +0.011 |
| 680 | +0.005 | +0.021 | +0.013 |
| 690 | +0.001 | +0.022 | +0.014 |
| 700 | +0.005 | +0.026 | +0.016 |
| 710 | +0.005 | +0.028 | +0.017 |
| 720 | 0.000 | +0.029 | +0.019 |
| 730 | +0.004 | +0.028 | +0.019 |
| 740 | +0.002 | +0.031 | +0.020 |
| 750 | +0.002 | +0.031 | +0.021 |
| 760 | +0.004 | +0.031 | +0.019 |
| 770 | +0.001 | +0.032 | +0.021 |
| 780 | +0.004 | +0.029 | +0.020 |
| 790 | +0.004 | +0.028 | +0.020 |
| 800 | +0.004 | +0.029 | +0.019 |
| 810 | -0.001 | +0.026 | +0.019 |
| 820 | +0.002 | +0.028 | +0.017 |
| 830 | -0.001 | +0.024 | +0.018 |
| 840 | -0.002 | +0.025 | +0.016 |
| 850 | +0.003 | +0.023 | +0.015 |
| 860 | +0.003 | +0.022 | +0.014 |
| 870 | -0.002 | +0.020 | +0.013 |
| 880 | -0.001 | +0.019 | +0.013 |
| 890 | +0.002 | +0.017 | +0.012 |
| 900 | -0.001 | +0.015 | +0.010 |
| 910 | -0.003 | +0.014 | +0.009 |
| 920 | +0.001 | +0.013 | +0.007 |
| 930 | +0.001 | +0.010 | +0.007 |
| 940 | 0.000 | +0.009 | +0.006 |
| 950 | +0.001 | +0.009 | +0.004 |
| 960 | +0.001 | +0.007 | +0.003 |

**Table 5** Deviations of heat capacity contributions of polyethylene calculated by the fitted equations [Deviations %] (continued)

| Temp. K | , Skeletal | Group | $C_V$ |
|---|---|---|---|
| 970 | -0.004 | +0.005 | +0.003 |
| 980 | -0.002 | +0.002 | +0.003 |
| 990 | +0.001 | +0.002 | +0.001 |
| 1000 | -0.003 | +0.002 | 0.000 |

For actual heat capacities of PE see Refs. 11 and 12

≥21. Since the heat capacity contributions of the group vibrations are extremely small at these temperatures, these larger errors have only little effect on the precision of the final $C_V$ and $C_p$. Actually, disregarding the low temperature fluctuations, the standard deviation of the group vibration contribution (between 140 and 1000 K) is 0.008 ± 0.172%. The overall average and RMS deviations of final $C_V$ and $C_p$ over the entire temperature range (0.1 to 1000 K) are 0.001 ± 0.023% and -0.007± 0.033%, respectively.

* * *

This research was sponsored by the Materials Division of the National Science Foundation of the U.S. Polymers Program, Grant # 8818412 and the Division of Materials Sciences, Office of Basic Energy Sciences, U.S. Department of Energy, under Contract DEAC05-840R21400 with Martin Marietta Energy Systems, Inc.

## APPENDIX

```
/***********************************************************************
 * This program reads the output files of SKELETAL.PRN, EINSTEIN.PRN *
 * and BOX.PRN of Lotus-1-2-3.  It then calculates the heat capacity *
 * contributions of skeletal vibration, Einstein function, box       *
 * distribution,  the total group vibration contribution,  Cv and Cp *
 * based on the parameters and frequecies in the above files.        *
 **********************************************************************/

 #include  < stdio.h >
 #include  < math.h >

 float tempsts[130];
 float thetae[100];
 float thetal[100];
 float thetau[100];
 float ne[100];
 float nb[100];
 double tarasov[130];
 double einstein[130];
 double box[130];
 double group[130];
 double Cv[130];
 double Cp[130];

 float thetal = 0, theta3 = 0, bn = 0, tm = 0, A = 0, sn = 0;
 double ress = 0;
 int item = 0;
 int einstitm = 0;
 int boxitm = 0;

/**********************************************************************
 *                            Main  Program                           *
 *********************************************************************/

 main(argc,argv)
 int argc;
 char *argv[];
 {    FILE *fopen(), *fclose(), *fps, *fpe, *fpb;

      if  ((fps = fopen(*++argv, "r")) == NULL)
          {
           printf("tfmain: can't open %s (Skeletal.Prn)\n", *argv);
           exit(1);
          }
      if ((fpe = fopen(*++argv, "r")) == NULL)
          {
           printf("\ntfmain: can't open %s (Einstein.Prn)\n",*argv);
           exit(1);
          }
```

```
     if ((fpb = fopen(*++argv, "r")) == NULL)
          {
           printf("\ntfmain: can't open %s (Box.prn)\n",*argv);
           exit(1);
          }
     gettemp();
     getskele(fps);
     geteinst(fpe);
     getbox(fpb);
     calculproc();
     prtproc();
}

/***********************************************************************
 * End of Main Program. Function gettemp() to load temperature scale *
 ***********************************************************************/

 gettemp()
 {
   char line[80];
   int i, j;

   i = 0;
   while ((getst(line,80)) != NULL)
         {
          sscanf(line,"%f", &tempsts[i]);
          i++;
         }
   item = i;
 }

/**********************************************************************
 *      Function getst() to scan each line of the input file        *
 **********************************************************************/

 getst(line, n)
 char line[80];
 int n;
 {
   char c;
   int i;

   i = 0;
   while ((c = getchar()) != EOF)
         {
           switch(c)
                {
                  case '\n':
                     line[i] = '\0';
                     return(1);
                  default:
                     line[i++] = c;
```

```
                    break;
          }
      }
    return(NULL);
  }

/**********************************************************************
 *      Function getskele() retrieves information of calculation
 *                              Parameters
 **********************************************************************

 getskele(fp)
 FILE *fp;
 {
   char line[80];
   int i;

   if ((fgets(line,80,fp)) == NULL)
      {
        fclose(fp);
        return;
      }

   sscanf(line, "%f", &thetal);
      if ( (i = strlen(line) ) == 0)
            thetal = 0;
      if ((fgets(line,80,fp)) == NULL)
         {
          fclose(fp);
          return;
         }
     if ((fgets(line,80,fp)) == NULL)
        {
         fclose(fp);
         return;
        }

   sscanf(line,"%f",&theta3);
      if ((i = strlen(line) ) == 0)
            theta3 = 0;
      if ((fgets(line,80,fp)) == NULL)
         {
          fclose(fp);
          return;
         }
      if ((fgets(line,80,fp)) == NULL)
         {
          fclose(fp);
          return;
         }

   sscanf(line, "%f", &bn);
```

```
      if ((i = strlen(line) ) == 0)
            bn = 0;
      if ((fgets(line,80,fp)) == NULL)
          {
           fclose(fp);
           return;
          }
      if ((fgets(line,80,fp)) == NULL)
          {
           fclose(fp);
           return;
          }

   sscanf(line,"%f",&tm);
      if ((i = strlen(line) ) == 0)
            tm = 0;
      if ((fgets(line,80,fp)) == NULL)
          {
           fclose(fp);
           return;
          }
      if ((fgets(line,80,fp)) == NULL)
          {
           fclose(fp);
           return;
          }

 sscanf(line,"%f",&A);
    if ((i = strlen(line) ) == 0)
        A = 0;
    if ((fgets(line,80,fp)) == NULL)
        {
         fclose(fp);
         return;
        }
   if ((fgets(line,80,fp)) == NULL)
        {
         fclose(fp);
         return;
        }

   sscanf(line,"%f",&sn);
      if ((i = strlen(line)) == 0)
          sn = 0;
   fclose(fp);
 }

/**********************************************************************
 * Function geteinst() retrieves the frequencies to be calulated by  *
 *                     the Einstein Function                          *
 **********************************************************************/
```

```
geteinst(fp)
FILE *fp;
{
   char line[80];
   int i;

   i = 0;
   einstitm = 0;
   while ((fgets(line,80,fp)) != NULL)
          {
           sscanf(line,"%f %f", &thetae[i], &ne[i]);
           if (thetae[i] == 0 && ne[i] == 0 && einstitm == 0)
            einstitm = i;
           i++;
          }
   if (einstitm == 0)
       einstitm = i - 1;
   fclose(fp);
 }

/***********************************************************************
 *  Function getbox() retrieves the frequencies to be calculated by *
 *                        the box distribution                        *
 ***********************************************************************/

 getbox(fp)
 FILE *fp;
 {
   char line[80];
   int i;

     i = 0;
     boxitm = 0;
     while ((fgets(line,80,fp)) != NULL)
            {
             sscanf(line,"%f %f %f",&thetal[i], &thetau[i], &nb[i]);
             if (thetal[i] == 0 && thetau[i] == 0 && nb[i] == 0 &&
                         boxitm == 0)
                  boxitm = i;
             i++;
            }
     fclose(fp);
     if (boxitm == 0)
          boxitm = i - 1;
}

/***********************************************************************
 *                  Calculation Function calcuproc()                  *
 ***********************************************************************/

 calculproc()
 {
     taraspróc();
     einstproc();
     boxproc();
     grouproc();
     Cvproc();
     Cpproc();
}

/*********************************************************************
 *  Function tarasproc() calculates the heat capacity contribution  *
 *                  based on the Tarasov function                    *
 *********************************************************************/

tarasproc()
{
  int i;
  double T, data1, data2, resul1, resul2, resul3;

  if (theta1 == 0 || theta3 == 0 || bn == 0)
     {
      setzero(tarasov, 0);
      return;
     }
  for (i = 0; i  <  item; i++)
      {
        T = tempsts[i];
        data1 = theta1 / T;
        getd1s(data1);
        resul1 = ress;
        data2 = theta3 / T;
        getd1s(data2);
        resul2 = ress;
        getd3s(data2);
        resul3 = ress;
        tarasov[i] = bn * (resul1 - (theta3 / theta1),
                             * (resul2 - resul3));
        if (tarasov[i]  <  0.1e-08)
            tarasov[i] = 0;
    }

 }

/*********************************************************************
 *   Function setzero() sets the negligiable contributions at very  *
 *                       temperatures to zero                        *
 *********************************************************************/

setzero(str,j)
double str[];
int j;
{
    int i;
    for (i = j; i  <  item; i++)
         {
          str[i] = 0;
         }
}

/*********************************************************************
 *  Function einstproc() calculates the heat capacity contributions  *
 *            of designated frequencies by Einstein function         *
 *********************************************************************/

einstproc()
{
  double exp();
  double T, datal, data2, resull, resul2, ans;
  int i = 0, j;

    if (thetae[i] == 0 && ne[i] == 0)
       {
         setzero(einstein,i);
         return;
       }

    for (i = 0; i  <  15; i++)
           einstein[i]=0;

    for (i = 15; i  <  item; i++)
         {
           T = tempsts[i];
           ans = 0;
           for (j = 0; j  <  einstitm; j++)
                {
                 datal = thetae[j] / T;
                 resull = exp(datal);
                 resul2 = resull - l;
                 ans += ne[j] * datal * datal * resull
                               / (resul2 * resul2);
                }
           einstein[i] = 8.31434 * ans;
       }
}

/*********************************************************************
 *    Function boxproc() calculates heat capacity contributions of   *
 *           the designated frequencies by box distribution          *
 *********************************************************************/

boxproc()
{
  double T, datal, data2, resull, resul2, ans;
  int i = 0, j;
```

```
    if (thetal[i] == 0 && thetau[i] == 0 && nb[i] == 0)
       {
        setzero(box, i);
        return;
       }

    for (i = 0; i  <  15; i++)
          box[i] = 0;

    for (i = 15; i  <  item; i++)
        {
          T = tempsts[i];
          ans = 0;
          for (j = 0; j  <  boxitm; j++)
              {
                datal = thetau[j] / T;
                getdls(datal);
                resull = ress;
                data2 = thetal[j] / T;
                getdls(data2);
                resul2 = ress;
                ans += (nb[j] * (thetau[j] / (thetau[j] - thetal[j]))
                              * (resull - (thetal[j] / thetau[j])
                                        * resul2));
              }
         if (ans  <  0.1e-08)
              box[i] = 0;
         else box[i] = ans;
       }
}

/**********************************************************************
 *     Function grouproc() sums the heat capacity contributions       *
 *      of both Einstein function and box distribution as the         *
 *               contribution of group vibration                      *
 **********************************************************************/

 grouproc()
 {
   int i;

   for (i = 0; i  <  item; i++)
         group[i] = einstein[i] + box[i];
 }

/**********************************************************************
 *  Function Cvproc() adds the heat capacity contributions from both *
 *               skeletal and group vibrations as Cv                  *
 **********************************************************************/

 Cvproc()
 {
  int i;

  if (theta1 == 0 || theta3 == 0 || bn == 0 )
     {
       setzero(Cv, 0);
       return;
     }

  for (i = 0; i  <  item; i++)
        Cv[i] = tarasov[i] + group[i];
}

/**********************************************************************
 *      Function Cpproc() converts the calculated Cv to Cp by         *
 *                   the Nernst-Lindemann equation                    *
 *********************************************************************/

Cpproc()
{
  int i;
  double T, a0, data1, resul1;

  if (tm == 0 || A == 0 || sn == 0 )
     {
       setzero(Cp, 0);
       return;
     }

  a0 = 0.001 * A / sn;

  for (i = 0; i  <  item; i++)
      {
        T = tempsts[i];
        data1 = (tm * tm) - (4 * a0 * T * tm * Cv[i]);
        if (data1  <  0)
           {
            setzero(Cp, i);
            return;
           }
        resul1 = sqrt(data1);
        Cp[i] = (tm - resul1) / (2 * a0 * T);
      }
}

/**********************************************************************
 *   Function prtproc() prints the output in a predetermined format   *
 *********************************************************************/

prtproc()
{
   int i;
   for (i = 0; i  <  15; i++)
        printf("%12.9f %12.9f %12.9f %12.9f %12.9f %12.9f\n",
                tarasov[i],einstein[i],box[i],group[i],Cv[i],Cp[i]);

   for (i = 15; i  <  23; i++)
        printf("%12.6f %12.9f %12.9f %12.9f %12.6f %12.6f\n",
                tarasov[i],einstein[i],box[i],group[i],Cv[i],Cp[i]);

   for (i = 23; i  <  31; i++)
        printf("%12.4f %12.9f %12.9f %12.9f %12.4f %12.4f\n",
                tarasov[i],einstein[i],box[i],group[i],Cv[i],Cp[i]);

   for (i = 31; i  <  36; i++)
        printf("%12.4f %12.6f %12.6f %12.6f %12.4f %12.4f\n",
                tarasov[i],einstein[i],box[i],group[i],Cv[i],Cp[i]);

   for (i = 36; i  <  43; i++)
        printf("%12.3f %12.4f %12.4f %12.4f %12.3f %12.3f\n",
                tarasov[i],einstein[i],box[i],group[i],Cv[i],Cp[i]);

   for (i = 43; i  <  item; i++)
        printf("%12.3f %12.3f %12.3f %12.3f %12,3f %12.3f\n",
                tarasov[i],einstein[i],box[i],group[i],Cv[i],Cp[i]);
}

/**********************************************************************
 *   Function getdls() calculates the Debye-1-dimensional function    *
 *                by the fitted polynomial functions                  *
 **********************************************************************/

getdls(x)
double x;
{
  double vl;

    if (x  < = 1.0)
         ress = -2.157225e-01*x*x - 1.010598e-02*x + 8.315802;
    else if (x  < = 4.0)
              ress = 2.782707e-02*x*x*x - 2.702351e-01*x*x
                        + 1.492426e-02*x + 8.318437;
    else if (x  < = 6.0)
              ress = 5.636826e-02*x*x - 1.287953*x + 10.08455;
    else if (x  <  14.0)
            {
              vl = log(x);
              ress = exp(1.134240e-01*vl*vl*vl - 8.418975e-01*vl*vl
                          + 1.085249*vl + 1.584817);
            }
    else if (x  <  21.0)
            {
              vl = 1 / x;
              ress = -3.050581e-01*vl*vl + 2.738556e+01*vl
```

## References

1a A. Xenopoulos and B. Wunderlich {polyamides}, Polymer, to be published.
b S. Z. D. Cheng, R. Pan and B. Wunderlich [poly(butylene terephthalete)], Makromol. Chem., 189 (1988) 2443.
c H. S. Bu, W. Aycock, S. Z. D. Cheng and B. Wunderlich [14 various polymers], Polymer, 29 (1988) 1485.
d H. S. Bu, W. Aycock and B. Wunderlich [various branched polymers], Polymer, 28 (1987) 1165.
e S. Z. D. Cheng, S. Lim, L. H. Judovits and B. Wunderlich [high melting phenylene containing polymers], Polymer, 28 (1987) 10.
f L. Judovits, R. C. Bopp, U. Gaur and B. Wunderlich {polystyrenes}, J. Polym. Sci., Polym. Phys. Ed., 24 (1986) 2725.
g S. Lim and B. Wunderlich [aliphatic polyesters], Polymer, 28 (1987) 777.
h K. Loufakis and B. Wunderlich [chlorine and fluorine containing derivatives of polyethylene], Polymer, 27 (1986) 563; 26 (1985) 1875.
i J. Grebowicz, W. Aycock and B. Wunderlich [1.4-polybutadienes], Polymer, 27 (1986) 575.
j J. Grebowicz, H. Suzuki and B. Wunderlich [polyethylene and aliphatic polyoxides], Polymer, 26 (1985) 561.
k S.-F. Lau, H. Suzuki and B. Wunderlich [polytetrafluoroethylene], J. Polym. Sci., Polym. Phys. Ed., 22 (1984) 379.
l J. Grebowicz, S.-F. Lau and B. Wunderlich [polypropylene], J. Polym. Sci., Polym. Symposium, 71 (1984) 19.
2 B. Wunderlich and H. Baur, Adv. Polym. Sci., 7 (1970) 151.
3 Yu. V. Cheban, S.-F. Lau and B. Wunderlich, Colloid and Polym. Sci., 260 (1982) 9; S.-F. Lau and B. Wunderlich, J. Thermal Anal., 28 (1983) 59.
4 H. S. Bu, S. Z. D. Cheng and B. Wunderlich, J. Phys. Chem., 91 (1987) 91.
5 V. V. Tarasov, Zh. Fiz. Khim., 24 (1950) 111; ibid., 24 (1953) 1430; ibid., 39 (1965) 2077; Dokl. Akad. Nauk SSSR, 100 (1955) 307.
6 B. Wunderlich [1D-Debye function table], J. Chem. Phys., 37 (1962) 1207
7 U. Gaur, G. Pultz, H. Wiedemeier and B. Wunderlich [2D-Debye function table], J. Thermal Anal., 43 (1981) 297.
8 J. A. Beattie, [3D-Debye function table], J. Math Phys. (MIT), 6 (1926/27) 1.
9 C. W. Clenshaw and A. R. Curtis, Num. Math, 2 (1960) 197.
10 J. Oliver, Comp. J., 15 (1972) 141.
11 T. N. L. Patterson, Math. Comp., 22 (1968) 847; 22 (1968) 877.
12 R. Pan, M. Varma-Nair and B. Wunderlich, J. Thermal Anal., to be published 35 (1989).

**Zusammenfassung** - Für einen IBM-kompatiblen Mikrocomputer mit einer Softwarebasis Lotus 1-2-3 wurde unter der Namen ATHAS ein Rechenschema für Wärmekapazitäten von festen, linearen Makromolekülen entwickelt. In diesem Beitrag wurden die ausschlielich numerisch integrierbaren Debye-Funktionen durch Funktionen aus Polynomen und exponentiellen Polynomen genähert, die Funktionen mit einer Genauigkeit von mindestens ±0.1% wiedergeben. Wärmekapazitäten können nun somit schneller und einfacher errechnet werden.