



LUND UNIVERSITY

Programpaket för utvärdering och presentation av fysikalisk mätdata.

Persson, Anders

1988

[Link to publication](#)

Citation for published version (APA):

Persson, A. (1988). *Programpaket för utvärdering och presentation av fysikalisk mätdata*. (Lund Reports in Atomic Physics; Vol. LRAP-93). Atomic Physics, Department of Physics, Lund University.

Total number of authors:

1

General rights

Unless other specific re-use rights are stated the following general rights apply:

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: <https://creativecommons.org/licenses/>

Take down policy

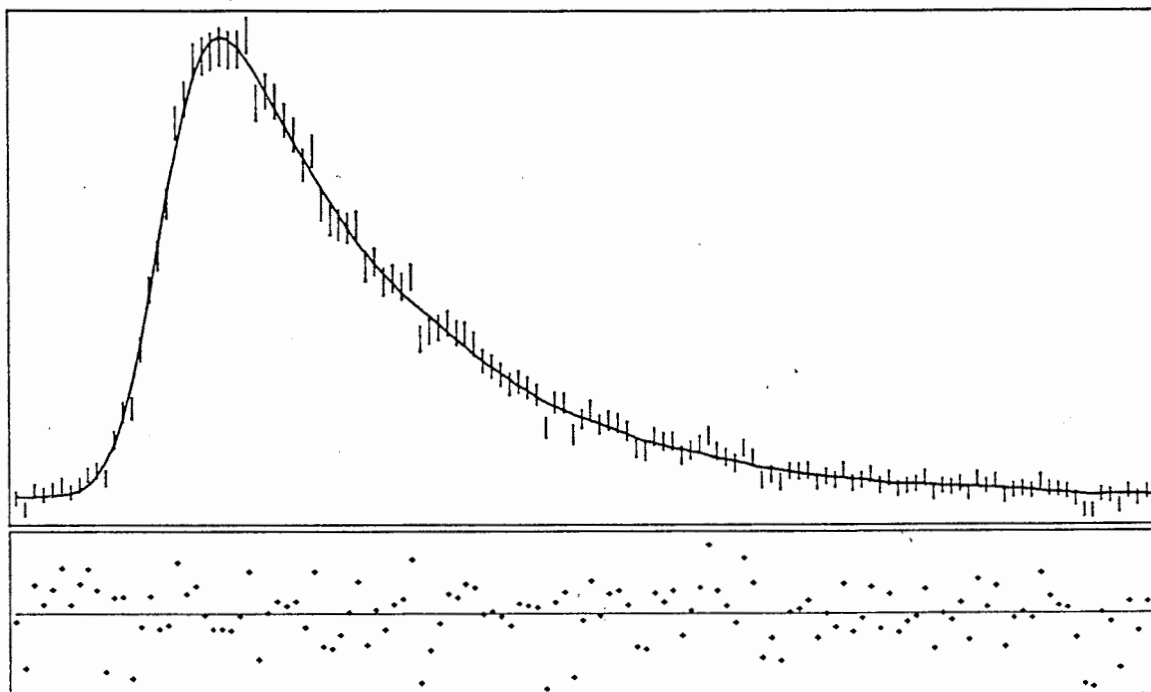
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

**PROGRAMPAKET FÖR UTVÄRDERING OCH
PRESENTATION AV FYSIKALISKA MÄTDATA**

ANDERS PERSSON



LRAP-93

INNEHÅLL

Inledning	1
Teori	2
Referenser	11
Programexempel	
POL1, anpassning till polynom av grad 4.	12
POL2, anpassning till polynom av grad n	17
POL3, anpassning till polynom av grad n,	22
CALCPOLC, generering av polynomkonstanter till POL3	26
ARCROSS, anpassning till argon fotojonisation- tvärsnitt, styckvisa polynom	37
EXP1, anpassning till en exponential + bakgrund	45
EXPN, anpassning till n exponentialer + bakgrund	55
GAUSSN, anpassning till n Gauss-toppar + bakgrund	68
LORN, anpassning till n Lorentz-toppar + bakgrund	77
PCONEXPA, anpassning till laserpuls faltad med n exponentialer + bakgrund + ströljus	86
PCONEXP, anpassning till laserpuls faltad med n exponentialer + bakgrund + ströljus	106
TESTMD, test av rutin för translation/derivation	122
TFEXP, test av rutin för beräkning av faltning	125
Hjälprutiner	
PVEC, CALCW, FFT, RANF, NORMAL	130
Matrisbibliotek	
MATLIB, DMATLIB	133
Ritrutiner	
SCRPLOT, ROLPLOT, JETPLOT	140
Ritbibliotek	
HERCLIB	152
PLLIB	159
HPLIB, HPFLIB	169
Länkningslista	181
Program och subrutinregister	182

Inledning

Dessa testprogram för anpassning till mätdata med hjälp av minsta kvadratmetoden är skrivna för en PC/XT/AT-kompatibel dator. Programspråket är FORTRAN-77 (med vissa delar skrivna i assembler t.ex. drivrutiner för skrivare och grafiskskärm). Programmen bör ganska lätt kunna omarbetas för andra anpassningsfunktioner än de som ingår i exemplen, anpassningsprogrammen behöver bara förses med nya rutiner för beräkning av funktionsvärdet och första och andraderivatorna med avseende på de anpassade parametrarna. Endast ett testprogram (ARCROSS) gör en anpassning till riktiga mätdata, de andra programmen tillverkar själva en testkurva. De delar av programmen som tillverkar testkurvorna bör dock lätt kunna bytas ut mot inläsningsrutiner för mätdata.

Programmen ställer vissa krav på hårdvara och mjukvara. Datorn måste (om Professional FORTRAN-77 kompilatorn från IBM används) vara försedd med matematikprocessorn 80x87 och med Hercules-kompatibel grafik. För att få ut bilder på papper är det lämpligt att ha tillgång till en HP-Laserjet Series II, laserskrivare eller en Roland DXY-980 pen-plotter men det är inte nödvändigt för att köra programmen. Eftersom både laserskrivaren och pen-plottern skall anslutas till samma skrivarutgång på datorn kan de inte användas samtidigt av en dator om inte drivrutinerna skrivs om. Programmen blir ganska stora, speciellt om de kompileras om för längre data-vektorer och fler parametrar. Det är därför tillrådligt att ha så mycket RAM-minne som möjligt i datorn (640 kB).

De program som behövs för att köra och kompilera/länka: är IBM Professional FORTRAN version 1.30 (eller senare), Microsoft MACRO-Assembler, LIB och LINK (levereras tillsammans med kompilatorn), HGC.COM och INT10.COM från Hercules, HPSCREEN från Hewlett-Packard för att dumpa skärmbilder på laserskrivaren kan också vara bra att ha men det är inte nödvändigt. HPSCREEN används i samband med testprogrammen TESTMD och TFEXP för att få ut bilder på papper. Vid kompilering med Fortran 77 kompilatorn från IBM bör option /Z användas. /Z betyder optimering=av, kompilatorn optimerar ibland koden till den grad att programmen ger felaktiga resultat.

Minsta kvadratmetoden

Antag att man i punkterna x_1, x_2, \dots, x_N har mätvärden y_1, y_2, \dots, y_N . Dessa mätvärden är behäftade med statistiska osäkerheter $\sigma_1, \sigma_2, \dots, \sigma_N$ där σ_1 är standardavvikelsen för mätvärdet y_1 (x_1 antas vara exakta). Vi har någon lämpligt vald funktion f som beskriver mätningen. Funktionen beror av x och ett antal parametrar som vi önskar skatta.

$$f_1 = f(\theta_1, \theta_2, \dots, \theta_L; x_1) = f(\underline{\theta}; x_1) \quad (1)$$

θ_i $i=1..L$ ($L \leq N$) är de önskade parametrarna. Minsta kvadratmetoden (MK) innebär att de bästa parametrarna är de som minimerar kvadratsumman

$$\chi^2 = \sum_{i=1}^N (y_i - f_i)^2 \cdot w_i \quad (2)$$

w_i är den vikt man vill sätta på mätning i . Om alla punkter anses lika viktiga används den oviktade kvadratsumman.

$$\chi^2 = \sum_{i=1}^N (y_i - f_i)^2 \quad (3)$$

Ofta används precisionen i mätning i som vikt, $w_i = 1/\sigma_i^2$ där σ_i^2 är variansen i mätning i

$$\chi^2 = \sum_{i=1}^N \left[\frac{y_i - f_i}{\sigma_i} \right]^2 \quad (4)$$

Ex. 1

Om y_1 är Poissonfördelade är $\sigma_1 = \sqrt{y_1} \Rightarrow$

$$\chi^2 = \sum_{i=1}^N \frac{(y_i - f_i)^2}{y_i} \quad \text{eller} \quad \chi^2 = \sum_{i=1}^N \frac{(y_i - f_i)^2}{f_i} \quad (5)$$

Några av MK metodens fördelar är att man inte behöver känna fördelningsfunktionen för y_1 (observablerna), linjär MK ger väntevärdesriktiga parameterskattningar och skattningar av parametrarna med minimal varians. Maximum likelihood (ML) skattning kan ha statistikteoretiska fördelar men man måste känna fördelningsfunktionen. Om y_1 är normalfördelade är ML och MK ekvivalenta.

Linjärt parameterberoende

Ex. 2

Funktionen linjär i parametrarna. Antag $f_1 = \theta = \text{konstant}$, y_1 har medelvärde m och varians σ^2 (samma varians för alla punkter).

Minimera:

$$\chi^2 = \sum_{i=1}^N \frac{(y_i - f_i)^2}{\sigma^2} \quad (5)$$

Minpunkt (eller max eller sadel) då

$$\frac{d\chi^2}{d\theta} = \sum_{i=1}^N \frac{-2(y_i - \theta)}{\sigma^2} = 0 \Rightarrow N \cdot \theta = \sum y_i \Rightarrow \theta = m = \frac{\sum y_i}{N} \quad (6)$$

Inversen av 1/2 gånger andraderivatan av χ^2 blir:

$$\left[\frac{1}{2} \cdot \frac{d\chi^2}{d\theta^2} \right]^{-1} = \left[\frac{1}{2} \sum_{i=1}^N \frac{2}{\sigma^2} \right]^{-1} = \frac{\sigma^2}{N} \quad (7)$$

vilket är variansen för θ om mätningarna (y_i) är oberoende.

Ex. 3

Anpassning till rät linje. Antag att felen är lika stora i alla mätpunkter och att $1/\sigma^2=1$.

$$f_i = \theta_1 + \theta_2 \cdot x_i \quad (8)$$

$$\chi^2 = \sum_{i=1}^N (y_i - \theta_1 - \theta_2 \cdot x_i)^2 \quad (9)$$

Vi får ett ekvationssystem med två obekanta:

$$\begin{cases} \frac{\partial \chi^2}{\partial \theta_1} = \sum_{i=1}^N (-2)(y_i - \theta_1 - \theta_2 \cdot x_i) = 0 \\ \frac{\partial \chi^2}{\partial \theta_2} = \sum_{i=1}^N (-2)(y_i - \theta_1 - \theta_2 \cdot x_i) \cdot x_i = 0 \end{cases} \quad (10)$$

⇔

$$\begin{pmatrix} \sum_{i=1}^N 1 & \sum_{i=1}^N x_i \\ \sum_{i=1}^N x_i & \sum_{i=1}^N x_i^2 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^N y_i \\ \sum_{i=1}^N x_i y_i \end{pmatrix} \quad (11)$$

vars lösning är:

$$\begin{cases} \theta_1 = \frac{\sum x_1^2 \cdot \sum y_1 - \sum x_1 y_1 \cdot \sum x_1}{N \cdot \sum x_1^2 - (\sum x_1)^2} \\ \theta_2 = \frac{N \cdot \sum x_1 y_1 - \sum x_1 \cdot \sum y_1}{N \cdot \sum x_1^2 - (\sum x_1)^2} \end{cases} \quad (12)$$

Det allmänna fallet där f är ett polynom av grad p

$$f_1 = \theta_1 + \theta_2 x_1 + \theta_3 x_1^2 + \dots + \theta_{p+1} x_1^p \quad (13)$$

ger ekvationssystemet:

$$\left[\sum_{i=1}^N x_1^{r+k-2} \right] \cdot \theta = \left[\sum_{i=1}^N y_1 x_1^{r-1} \right] \quad r, k=1, 2, \dots, p+1 \quad (14)$$

Matrisen kan vara svår att invertera, (illa konditionerat ekvationssystem) det kan hjälpa att centrera mätdata kring origo. Vid anpassning till en rät linje är detta ekvivalent med att använda ortogonala polynom. Med lämpliga transformationer kan även andra funktioner än polynom anpassas med hjälp av polynom Anpassning.

Ex. 4

$$f(x) = C \cdot e^{-a \cdot x} \quad \text{-- logaritmera och anpassa till rät linje}$$

$$f(x) = C \cdot e^{-a \cdot x^2} \quad \text{-- Gauss-topp, logaritmera och anpassa till 2-grads polynom.}$$

$$f(x) = A / (B + x^2) \quad \text{-- Lorentz-topp, invertera och anpassa till 2-grads polynom}$$

Matrisnotation

För att bestämma ekvationssystemet om f är en allmän funktion, linjär i parametrarna, behövs matrisnotation för att uttrycken skall bli hanterbara. Vi har N oberoende observationer (med felgränser) $y_1 \pm \sigma_1, y_2 \pm \sigma_2, \dots, y_N \pm \sigma_N$ i punkterna x_1, x_2, \dots, x_N . Sätt:

$$f_i = f(\theta_1, \theta_2, \dots, \theta_L; x_i) = \sum_{j=1}^L a_j(x_i) \cdot \theta_j \quad (15)$$

där $a_j(x)$ är en godtycklig funktion t.ex. polynom, exp, sin, cos, Kvadratsumman (4) blir:

$$\chi^2 = \sum_{i=1}^N \left[\frac{y_i - f_i}{\sigma_i} \right]^2 = \sum_{i=1}^N \left[\frac{1}{\sigma_i} \right]^2 \cdot \left[y_i - \sum_{j=1}^L a_j(x_i) \cdot \theta_j \right]^2 \quad (16)$$

döp om $a_j(x_i)$ till α_{ij}

$$\chi^2 = \sum_{i=1}^N \left(\frac{1}{\sigma_i} \right)^2 \left[y_i - \sum_{j=1}^L \alpha_{ij} \theta_j \right]^2 \quad (17)$$

derivatorna med avseende på θ_j (vilka skall vara noll i minpunkten) blir:

$$\frac{\partial \chi^2}{\partial \theta_k} = \sum_{i=1}^N (-2\alpha_{ik}) (1/\sigma_i^2) (y_i - \sum_{j=1}^L \alpha_{ij} \theta_j) = 0 \quad k=1..L \quad (18)$$

detta ekvationssystem kan skrivas på formen:

$$\left[\sum_{i=1}^N \frac{\alpha_{ir} \cdot \alpha_{ik}}{\sigma_i^2} \right] \underline{\theta} = \left[\sum_{i=1}^N \frac{\alpha_{ir} \cdot y_i}{\sigma_i^2} \right] \quad (19)$$

För att kunna skriva ovanstående uttryck med hjälp av matriser behövs några definitioner. Sätt:

$$\underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, \quad \underline{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{bmatrix}, \quad \underline{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_N \end{bmatrix}, \quad (20)$$

Felen, σ_i , introduceras genom kovariansmatrisen $V(\underline{y})$. Denna blir diagonal vid oberoende observationer.

$$V = V(\underline{y}) = \begin{bmatrix} \sigma_1^2 & & & 0 \\ & \sigma_2^2 & & \\ & & \sigma_3^2 & \\ & & & \ddots \\ 0 & & & & \sigma_N^2 \end{bmatrix} \quad (21)$$

Om observationerna ej är oberoende skall de icke diagonala elementen vara kovariansen mellan y_r, y_k . (Se ref. 1 sid 5/13 för definition av kovarians.) Kovariansmatrisen är symmetrisk. Sätt vidare:

$$A = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \dots & \alpha_{1L} \\ \alpha_{21} & \cdot & \cdot & \cdot & \cdot \\ \vdots & & & & \\ \alpha_{N1} & \cdot & \cdot & \cdot & \alpha_{NL} \end{bmatrix} \quad (22)$$

Vi kan nu skriva uttrycken (15) och (17) med hjälp av matriser på ett kompakt sätt

$$\underline{f} = A \cdot \underline{\theta} \quad (23)$$

$$\chi^2 = (\underline{y} - A \cdot \underline{\theta})^T V^{-1} (\underline{y} - A \cdot \underline{\theta}) \quad (24)$$

Minimum av χ^2 (då derivatorna är noll) (18) ger

$$\nabla_{\underline{\theta}} \chi^2 = -2(\mathbf{A}^T \mathbf{V}^{-1} \underline{y} - \mathbf{A}^T \mathbf{V}^{-1} \mathbf{A} \underline{\theta}) = \underline{0} \quad (25)$$

⇔ (19)

$$(\mathbf{A}^T \mathbf{V}^{-1} \mathbf{A})^{-1} \underline{\theta} = \mathbf{A}^T \mathbf{V}^{-1} \underline{y} \quad (26)$$

Detta är lösbart om $\mathbf{A}^T \mathbf{V}^{-1} \mathbf{A}$ ej är singulär och lösningen blir

$$\underline{\theta} = (\mathbf{A}^T \mathbf{V}^{-1} \mathbf{A})^{-1} \mathbf{A}^T \mathbf{V}^{-1} \underline{y} \quad (27)$$

Man kan notera att kovariansmatrisen kan multipliceras med godtycklig faktor ($\neq 0$) utan att parameterskattningen $\underline{\theta}$ ändras. Skall osäkerheterna i $\underline{\theta}$ skattas måste man veta kovariansmatrisen exakt. Man kan visa att [3] kovariansmatrisen för $\underline{\theta}$ blir:

$$\mathbf{V}(\underline{\theta}) = (\mathbf{A}^T \mathbf{V}^{-1} \mathbf{A})^{-1} \quad (28)$$

Då denna matris ingår som del i lösningen (27) till $\underline{\theta}$ behövs inga extra beräkningar för att få fram felen i parameterskattningarna.

$$\underline{\theta} = \mathbf{V}(\underline{\theta}) \mathbf{A}^T \mathbf{V}^{-1} \underline{y} \quad (29)$$

Standardavvikelsen σ_k för parameter θ_k erhålls som kvadratroten ur diagonalelementet $\mathbf{V}(\underline{\theta})_{kk}$. Om θ_i och θ_j är korrelerade är ρ_{ij}

$$\rho_{ij} = \frac{\mathbf{V}(\underline{\theta})_{ij}}{\sqrt{\mathbf{V}(\underline{\theta})_{ii}} \sqrt{\mathbf{V}(\underline{\theta})_{jj}}} \quad (30)$$

en skattning av korrelationskoefficienten (Se ref. 1, sid. 5/5).

Ex. 5

Polynom Anpassning, lika vikt för alla punkter. Om man ansätter funktionen f där

$$f_i = \sum_{j=1}^L x_i^{j-1} \cdot \theta_j \quad (31)$$

blir matrisinverteringen svår om gradtalet är högt ($\cong 6$). $\mathbf{V}(\underline{y}) = \sigma^2 \mathbf{I}$ ger lösningen

$$\underline{\theta} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \underline{y} \quad (32)$$

Om man istället anpassar till ortonormerade polynom $\alpha_j(x)$ där ortonormalitetskravet ges av:

$$\sum_{i=1}^N a_k(x_i) \cdot a_j(x_i) = \delta_{kj} \quad k, j=1..L \quad (33)$$

blir $\mathbf{A}^T \mathbf{A}$ enhetsmatrisen och lösningen $\underline{\theta}$ kan skrivas: $\underline{\theta} = \mathbf{A}^T \underline{y}$.

Kovariansmatrisen $\mathbf{V}(\underline{\theta})$ blir också enkel, $\mathbf{V}(\underline{\theta}) = \mathbf{A}^T \mathbf{V}(\underline{y}) \mathbf{A} = \sigma^2 \mathbf{A}^T \mathbf{A} = \sigma^2 \mathbf{I} \Rightarrow$ parametrarna är okorrelerade.

Ex. 6

Rät linje. Välj $a_1(x_1)=1$ och $a_2(x_1)=x_1-\bar{x}$ (mätpunkterna centrerade kring $x=0$). $A^T A$ blir visserligen inte enhetsmatrisen men den blir diagonal vilket medför trivial invertering.

Icke linjärt parameterberoende

Om funktionen $f(\underline{\theta}; \underline{x})$ beror av parametrarna θ_i på ett icke linjärt sätt och man inte kan transformera funktionen på något listigt sätt så att den blir linjär, har man ett mycket besvärligt problem att lösa. Man måste då använda någon iterativ metod för att hitta minpunkten. En av de bästa (om den fungerar) är Newtons metod. Vi har att:

$$\chi^2 = (\underline{y}-\underline{f})^T V^{-1} (\underline{y}-\underline{f}) \quad (34)$$

$$\underline{f} = \underline{f}(\underline{\theta}; \underline{x}) \quad (35)$$

Antag att vi har en startvektor $\underline{\theta}^m$:

$$\underline{\theta}^m = [\theta_1^m, \theta_2^m, \dots, \theta_L^m]^T \quad (36)$$

och att $V(\underline{y})_{ij} = 1/\sigma_i^2 \cdot \delta_{ij}$ (oberoende mätningar). Minpunkten ges av den punkt där derivatorna map θ är noll, startvektorn $\underline{\theta}^m$ behöver inte vara minpunkten. Sätt:

$$\frac{\partial \chi^2}{\partial \theta_j} = g_j(\underline{\theta}^m) = \sum_{i=1}^N \left[-\frac{2}{\sigma_i^2} \right] (y_i - f_i) \frac{\partial f_i}{\partial \theta_j} \quad j=1..L \quad (37)$$

Man vill hitta en vektor $\Delta \underline{\theta}^m$ sådan att \underline{g} blir noll

$$\underline{g}(\underline{\theta}^m + \Delta \underline{\theta}^m) = 0 \quad (38)$$

iteration $m+1$ skall ge

$$\underline{\theta}^{m+1} = \underline{\theta}^m + \Delta \underline{\theta}^m \quad (39)$$

För att hitta ett uttryck för $\Delta \underline{\theta}^m$ kan man utveckla \underline{g} kring $\underline{\theta}^m$ till första ordningen

$$g_j(\underline{\theta}^m) + \frac{\partial g_j}{\partial \theta_1} \cdot \Delta \theta_1^m + \frac{\partial g_j}{\partial \theta_2} \cdot \Delta \theta_2^m + \dots + \frac{\partial g_j}{\partial \theta_L} \cdot \Delta \theta_L^m = 0, \quad j=1..L \quad (40)$$

Derivatorna beräknas i punkten $\underline{\theta}^m$. Koefficienterna $\partial g_j / \partial \theta_k$ framför $\Delta \theta$ -termerna är andraderivatorna av χ^2 .

$$\frac{\partial g_j}{\partial \theta_k} = \frac{\partial^2 \chi^2}{\partial \theta_k \partial \theta_j} = G_{kj} \quad (41)$$

$$G_{kj} = \sum_{i=1}^N \left[-\frac{2}{\sigma_i^2} \right] \left[-\frac{\partial f_i}{\partial \theta_k} \cdot \frac{\partial f_i}{\partial \theta_j} + (y_i - f_i) \frac{\partial^2 f_i}{\partial \theta_k \partial \theta_j} \right] \quad (42)$$

G_{kj} är symmetrisk men behöver inte vara positivt definit vilket egentligen är ett krav för att iterationsformeln skall konvergera mot ett minimum. Med matrisnotation kan korrektionstermerna $\Delta \underline{\theta}$ skrivas

$$\Delta \underline{\theta}^m = -G(\underline{\theta}^m)^{-1} \cdot \underline{g}(\underline{\theta}^m) \quad (43)$$

När den nya $\underline{\theta}^{m+1}$ beräknats stoppar man in värdena i G och \underline{g} och utför beräkningarna igen osv. Om startvektorn är tillräckligt nära den rätta minpunkten konvergerar metoden snabbt mot det rätta värdet (kvadratisk konvergens). Man får dock problem om G matrisen blir singulär eller nästan singulär vilket illustreras bäst i en dimension (Newton-Raphson).

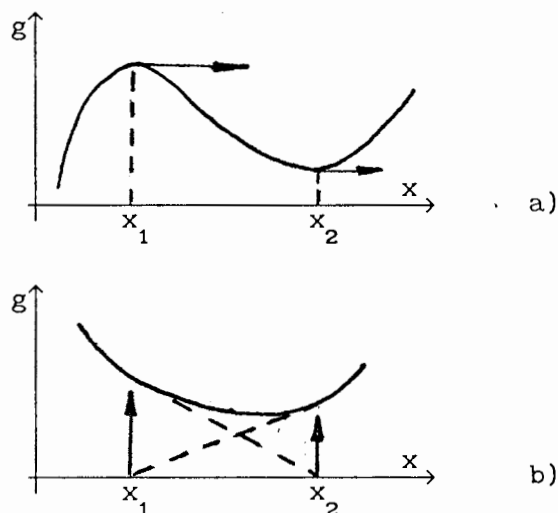


Fig. 1: $x^{k+1} = x^k - g(x^k)/g'(x^k)$. Problem uppstår i punkterna x_1 och x_2 . a) derivatan = 0, b) x^{k+1} hoppar mellan punkterna x_1 och x_2 .

Feluppskattning

I exempel 2 beräknades parametervariansen genom (7), i det allmänna icke linjära fallet erhålls en skattning av kovariansmatrisen med

$$V(\underline{\theta})_{ij}^{-1} = \frac{1}{2} \left[\frac{\partial^2 \chi^2}{\partial \theta_i \partial \theta_j} \right] \quad (44)$$

där derivatorna är beräknade i minpunkten. Denna är redan beräknad i iterationsformeln eftersom:

$$V(\underline{\theta})^{-1} = 1/2 \cdot G(\underline{\theta}) \Rightarrow V(\underline{\theta}) = 2 \cdot G(\underline{\theta})^{-1} \quad (45)$$

Kvadratroten av diagonalelementen i G^{-1} ger ett mått på felen i θ .

Vill man ha en bättre feluppskattning kan man följa χ^2 -värdet från minpunkten χ_{\min}^2 upp till $\chi_{\min}^2 + a$ och se hur mycket de olika parametrarna behöver ändras, detta ger bättre och ej säkert symmetriska feluppskattningar.

Linearisering

Ett av problemen med Newtons metod är att G -matrisen inte alltid är positivt definit. Genom att försumma andra termen i uttrycket för G (38) får man den lineariserade G -matrisen.

$$G_{kj} \approx \sum_{i=1}^N \left[\frac{2}{\sigma_i^2} \right] \frac{\partial f_i}{\partial \theta_k} \cdot \frac{\partial f_i}{\partial \theta_j} \quad (46)$$

Denna är alltid positivt definit om derivatorna existerar och om den ej är singular. Den är också enklare att beräkna än den fullständiga G -matrisen. Ofta konvergerar Newtons metod snabbare med denna approximation och metoden klarar sämre startvärden utan att divergera. Konvergensen sker mot samma minpunkt som med den fullständiga G -matrisen. Man får dock betala ett pris för dessa fördelar, den approximativa G -matrisen ger inga feluppskattningar eftersom $G^{-1} \neq V$. Vill man ha ett mått på felet måste den fullständiga G -matrisen beräknas när minpunkten är uppnådd.

Andra metoder

Andra minimeringsmetoder än Newtons metod är t. ex. Random search, Grid search, Simplex, Steepest descent, Davidon-Fletcher-Powell, Kvadratisk interpolation, och extrapolation, intervallhalvering mm. Några av dessa metoder har fördelen att vara stabilare än Newtons metod men nära minpunkten finns knappt någon metod som konvergerar snabbare. Dessa och andra metoder finns beskrivna i ref. 3,4,5,6.

Hur bra är anpassningen?

Ett mått på anpassningens godhet är det reducerade χ^2 värdet

$$\chi^2 = \frac{\chi^2}{n-p} \quad (47)$$

där n är antalet mätpunkter och p är antalet skattade parametrar. Enligt [7] skall χ^2 -värdet ligga nära ett för Poisson fördelade mätdata, ett värde lägre än 0.75 tyder på för kort mättid (för litet totalt antal counts). Värden mellan 0.8 och 1.2 är OK. Ett annat (och bättre) sätt att kontrollera anpassningen är att rita kurvan över de

viktade resterna (Weighted residuals) [7], $r(x_1)$

$$r(x_1) = (y_1 - f_1) \cdot \sqrt{w_1} = \frac{y_1 - f_1}{\sigma_1} \quad (48)$$

Resterna skall vid en godkänd anpassning vara slumpmässigt fördelade kring noll, alla systematiska variationer av $r(x)$ brukar synas bra i en graf av denna. Alla de testprogram som följer ritar denna punktföljd. Ett annat mått på anpassningen är den s.k. Durbin-Watson parametern, DW [7]

$$DW = \frac{\sum_{i=2}^n [r(x_i) - r(x_{i-1})]^2}{\sum_{i=1}^n [r(x_i)]^2} \quad (49)$$

DW skall (enl. ref. 7) ha ett värde på $\geq 1.7-1.8$ för anpassning till 1..3 exponentialer (256..512 datapunkter).

Medelvärdet och standardavvikelsen för de viktade resterna

$$\bar{r} = \sum r(x_i)/n, \quad \sigma_r = \frac{\left[\sum [r(x_i) - \bar{r}]^2 \right]^{1/2}}{(n-1)^{1/2}} \quad (50)$$

kan också användas som ett mått på anpassningen. \bar{r} skall ha ett värde nära noll och σ_r skall vara ≈ 1 , de är inte särskilt känsliga för dåliga anpassningar.

Referenser

- [1] G. Blom: Statistikteori med tillämpningar, Studentlitteratur, Lund, 1970
- [2] G. Blom: Sannolikhetsteori med tillämpningar, Studentlitteratur, Lund, 1970
- [3] A. G. Frodesen, O. Skjeggestad, H. Tøfte: Probability and Statistics in Particle Physics, Universitetsforlaget.
- [4] F. James: Function minimization, Proc. of the 1972 CERN Computing and Data Processing School, CERN 1972
- [5] T. Ekman: Numeriska metoder på dator och dosa, Sigma-Tryck TLTH, Lund, 1978.
- [6] G. Eriksson: Numerisk analys FK, Sigma-Tryck TLTH, Lund, 1979
- [7] D. V. O'Connor, D. Phillips: Time-correlated Single Photon Counting, Academic Press Inc. (London) Ltd. 1984.
- [8] R. W. Ramirez: The FFT: Fundamentals and Concepts, Tectronix Inc. (1975).
- [9] R. C. Jennison: Fourier Transforms and Convolutions for the Experimentalist, Pergamon Press, New York 1961
- [10] T. Ekman, G. Eriksson, Programmering i Fortran 77, Studentlitteratur, Lund, 1979.
- [11] B. Wichman, D. Hill: Building a Random-Number Generator. Byte March 1987, p. 127.
- [12] J. R. Van Aken, C. R. Killebrew Jr.: Better Bit-Mapped Lines, Byte March 1987, p. 249.
- [13] M. A. O'Neill: Faster than Fast Fourier: Byte April 1988 p. 293.
- [14] R. N. Bracewell: The Discrete Hartley Transform, JOSA 73, 1832, (1983).
- [15] J. W. Hartwell: Is Hartley Really Faster (Letter), Byte July 1988 p. 26.

POL1

Programmet anpassar ett fjärdegradspolynom till valbart antal punkter, anpassningsfunktionen är:

$$f(\underline{\theta}; x) = \theta_1 + \theta_2 x + \theta_3 x^2 + \theta_4 x^3 + \theta_5 x^4 \quad (51)$$

Test kurvan beräknas i programmet genom att de exakta värdena beräknas och därefter adderas ett approximativt normalfördelat brus där normalfördelningens standardavvikelse ges i % av skillnaden mellan kurvans max och minpunkt. Programmet skriver ut parametervärden med felgränser och skriver även ut kovariansmatrisen. Man kan rita anpassning och residualvektor på pen-plottern eller laserskrivaren. Från testkörningens kovariansmatris kan noteras att korrelationen mellan θ_3, θ_5 och θ_2, θ_4 är mycket större än mellan θ_2, θ_3 .

Utskrift från testkörning av programmet POL1:

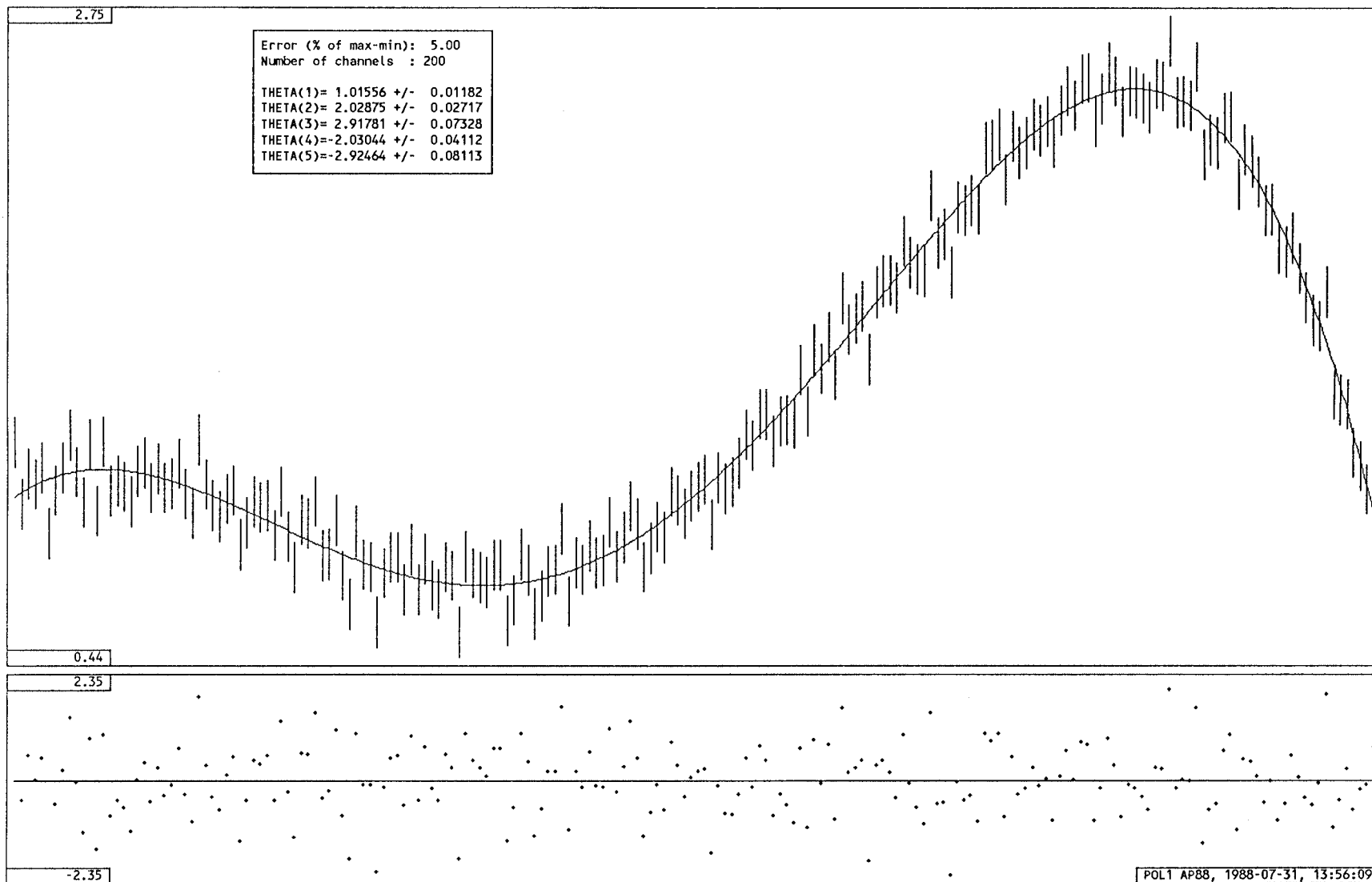
```
Sigma(Y) ( % of ymax-ymin )      :5
No of channels ( 5..1000)         :200
Plot errorbars? (Y/*)            :Y
Plot on Roland plotter? (Y/*)    :N
Plot on HP-Laserjet? (Y/*)       :Y
```

```
THETA(1)= 0.99908 +/- 0.01182
THETA(2)= 2.01402 +/- 0.02717
THETA(3)= 3.01692 +/- 0.07328
THETA(4)=-2.06049 +/- 0.04112
THETA(5)=-2.99541 +/- 0.08113
```

The covariance matrix for theta:

```
0.140E-03  0.278E-05 -0.646E-03 -0.643E-05  0.575E-03
0.278E-05  0.738E-03 -0.386E-04 -0.102E-02  0.573E-04
-0.646E-03 -0.386E-04  0.537E-02  0.891E-04 -0.570E-02
-0.643E-05 -0.102E-02  0.891E-04  0.169E-02 -0.132E-03
0.575E-03  0.573E-04 -0.570E-02 -0.132E-03  0.658E-02
```

New calculation? (Y/*) :N



Kommentarer till programlistningen:

RAD	KOMMENTAR
1-15	Deklarationer.
16-20	De exakta θ -värdena. ($\theta=[1,2,3,-2,-3]^T$)
22-24	Läs in hur stor standardavvikelse mätpunkterna skall ha.
26-29	Läs in antalet mätpunkter i anpassningen.
31-34	Välj om felgränser skall ritas i bilderna.
36-40	Beräkna (x_1, y_1) , $x \in [0,2]$, $y_1 = N(f_1, \sigma_1)$.
42-45	Nollställ matriserna A,B och SIGMA samt placera värdet SIGMAIN i vektorn SIGMA (samma osäkerhet i alla mätpunkter). För beskrivning av matrisrutinerna se kommentarer till biblioteket MATLIB.
47-60	Beräkna A och B matriserna där $A_{r,k} = \sum x_i^{r+k-2}$ och $B_r = \sum y_i x_i^{r-1}$ jfr. (14).
62-65	Multiplitera båda sidor i ekvationen med vikten $1/\sigma^2$ för att få den rätta kovariansmatrisen för feluppskattningen.
67-71	Invertera A-matrisen.
73	Beräkna $\underline{\theta}$, $\underline{\theta} = A^{-1} \cdot B$.
75	Rita anpassning och residualvektor på skärmen.
77-82	Rita anpassning och residualvektor på Roland pen-plotter.
84-132	Rita anpassning och residualvektor samt skriv ut diverse indata och resultat på HP-laserskrivare. (Se fig.)
134-149	Skriv ut de beräknade parametrarna med felgränser och kovariansmatrisen på skärmen.
151-159	Fråga om ny beräkning skall göras.
161-177	Rutinen beräknar värdet av funktionen $f(\underline{\theta}; x)$.

```

1 PROGRAM POL1
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5 INTEGER N,L
6 PARAMETER (N=1000, L=5)
7 REAL Y(1:N),X(1:N),DX,DET,RANF,THETAESTIMATE(1:L)
8 REAL SIGIN,RESIDUE(1:N),SIGMA(1:N),F,THETA(1:L),NORMAL
9 REAL B(1:L),A(1:L,1:L),JX,JY,JXL,JYL,DJY
10 INTEGER R,K,I,NCH
11 INTEGER*2 HO,MI,SE,HN,YEAR,MONTH,DAY
12 CHARACTER SV*1,JETT*40
13 LOGICAL SING,ERRORBAR
14 EXTERNAL F
15
16 THETA(1)= 1.0
17 THETA(2)= 2.0
18 THETA(3)= 3.0
19 THETA(4)=-2.0
20 THETA(5)=-3.0
21
22 10 WRITE(*,*) 'Sigma(Y) ( % of ymax-ymin ):'
23 READ(*,*,ERR=10) SIGIN
24 SIGIN=SIGIN*0.0178319
25
26 20 WRITE(*,1000) 'No of channels (',L,',',N,'):'
27 1000 FORMAT(' ',A,12,A,14,A)
28 READ(*,*,ERR=20) NCH
29 IF (NCH.LT.L.OR.NCH.GT.N) GOTO 20
30
31 WRITE(*,*) 'Plot errorbars? (Y/*):'
32 READ(*,1100) SV
33 1100 FORMAT(A)
34 ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'
35
36 DX=2.0/(NCH-1)
37 DO 30 I=1,NCH
38 X(I)=(I-NCH/2)*DX
39 Y(I)=NORMAL(F(THETA,L,X(I)),SIGIN)
40 30 CONTINUE
41
42 CALL ZEROMAT(A,L,L,L)
43 CALL ZEROMAT(B,L,1,L,1)
44 CALL ZEROMAT(SIGMA,NCH,1,N,1)
45 CALL ADDCMAT(SIGMA,SIGIN,NCH,1,N,1)
46
47 DO 60 I=1,NCH
48 B(1)=B(1)+Y(I)
49 DO 40 R=2,L
50 B(R)=B(R)+Y(I)*X(I)**(R-1)
51 40 CONTINUE
52 A(1,1)=A(1,1)+1.0
53 DO 50 R=1,L
54 DO 50 K=R,L
55 IF (R+K.NE.2) THEN
56 A(R,K)=A(R,K)+X(I)**(R+K-2)
57 ENDIF

```

```

58 A(K,R)=A(R,K)
59 50 CONTINUE
60 60 CONTINUE
61
62 IF (SIGIN.NE.0.0) THEN
63 CALL MULTCMAT(A,1.0/SIGIN**2,L,L,L)
64 CALL MULTCMAT(B,1.0/SIGIN**2,L,1,L)
65 ENDIF
66
67 CALL INVMAT(A,L,DET,SING,L)
68 IF (SING) THEN
69 WRITE(*,*) 'The A-matrix is singular.'
70 GOTO 120
71 ENDIF
72
73 CALL MULTMAT(A,B,THETAESTIMATE,L,L,1,L,L,1,1)
74
75 CALL SCRIPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAESTIMATE,L,ERRORBAR)
76
77 70 WRITE(*,*) 'Plot on Roland plotter? (Y/*):'
78 READ(*,1100) SV
79 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
80 CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAESTIMATE,L,ERRORBAR)
81 GOTO 70
82 ENDIF
83
84 80 WRITE(*,*) 'Plot on HP-Laserjet? (Y/*) : '
85 READ(*,1100) SV
86 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
87 CALL HPSTART('L')
88 CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAESTIMATE,L,ERRORBAR)
89 JX=200
90 JXL=394-JX
91 JYL=(3+1+L)*13
92 JY=700-JYL
93 CALL HPBOX(JX,JY,JXL,JYL)
94 JY=700-16
95 DJY=-13
96
97 WRITE(JETT*1600) 'Error (% of max-min):',SIGIN/0.0178319
98 1600 FORMAT(' ',A,F6.2)
99 CALL HPPRINT(JX,JY,40,JETT*)
100 JY=JY+DJY
101
102 WRITE(JETT*1700) 'Number of channels :',NCH
103 1700 FORMAT(' ',A,I4)
104 CALL HPPRINT(JX,JY,40,JETT*)
105 JY=JY+DJY
106
107 WRITE(JETT*1800) ' '
108 1800 FORMAT(' ',A)
109 CALL HPPRINT(JX,JY,40,JETT*)
110 JY=JY+DJY
111
112 DO 90 R=1,L
113 IF (SIGIN.EQ.0.0) THEN
114 WRITE(JETT*1200) 'THETA(',R,')=',THETAESTIMATE(R)
115 ELSE
116 WRITE(JETT*1200) 'THETA(',R,')=',THETAESTIMATE(R), ' +/- ',
117 f SQRT(ABS(A(R,R)))

```

```

118         ENDIF
119         CALL HPPRINT(JX,JY,40,JETTXT)
120         JY=JY+DJY
121 90      CONTINUE
122
123         CALL GETTIM(HO,MI,SE,HN)
124         CALL GETDAT(YEAR,MONTH,DAY)
125         WRITE(JETTXT,1900)
126         f 'POL1 AP88, ',YEAR,'-',MONTH,'-',DAY,' ',HO,':',MI,':',SE
127 1900    FORMAT(A,I4.4,5(A,I2.2))
128         CALL HPTBOX(921.0,0.0,31,JETTXT)
129
130         CALL HPEJECT
131         GOTO 80
132     ENDIF
133
134     DO 100 R=1,L
135         IF (SIGIN.EQ.0.0) THEN
136             WRITE(*,1200) 'THETA(',R,')=',THETAESTIMATE(R)
137         ELSE
138             WRITE(*,1200) 'THETA(',R,')=',THETAESTIMATE(R),' +/- ',
139         f          SQRT(ABS(A(R,R)))
140         ENDIF
141 1200    FORMAT(' ',A,I1,A,F8.5,A,F8.5)
142 100    CONTINUE
143
144     WRITE(*,1300) 'The covariance matrix for theta:'
145 1300    FORMAT('/' ',A)
146     DO 110 K=1,L
147         WRITE(*,1400) (A(R,K), R=1,L)
148 1400    FORMAT(' ',7G11.3)
149 110    CONTINUE
150
151 120    WRITE(*,1500) 'New calculation? (Y/*):'
152 1500    FORMAT('/' ',A)
153     READ(*,1100) SV
154     IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
155         GOTO 10
156     ELSE
157         STOP
158     ENDIF
159     END
160
161     REAL FUNCTION F(THETA,L,X)
162 C      Anders Persson, 1988
163
164     IMPLICIT LOGICAL (A-Z)
165     INTEGER L,I
166     REAL THETA(1:L),X,SLASK
167
168     SLASK=THETA(1)
169     IF (X.NE.0.0) THEN
170         DO 10 I=2,L
171             SLASK=SLASK+THETA(I)*X**(I-1)
172 10     CONTINUE
173     ENDIF
174     F=SLASK
175
176     RETURN
177     END

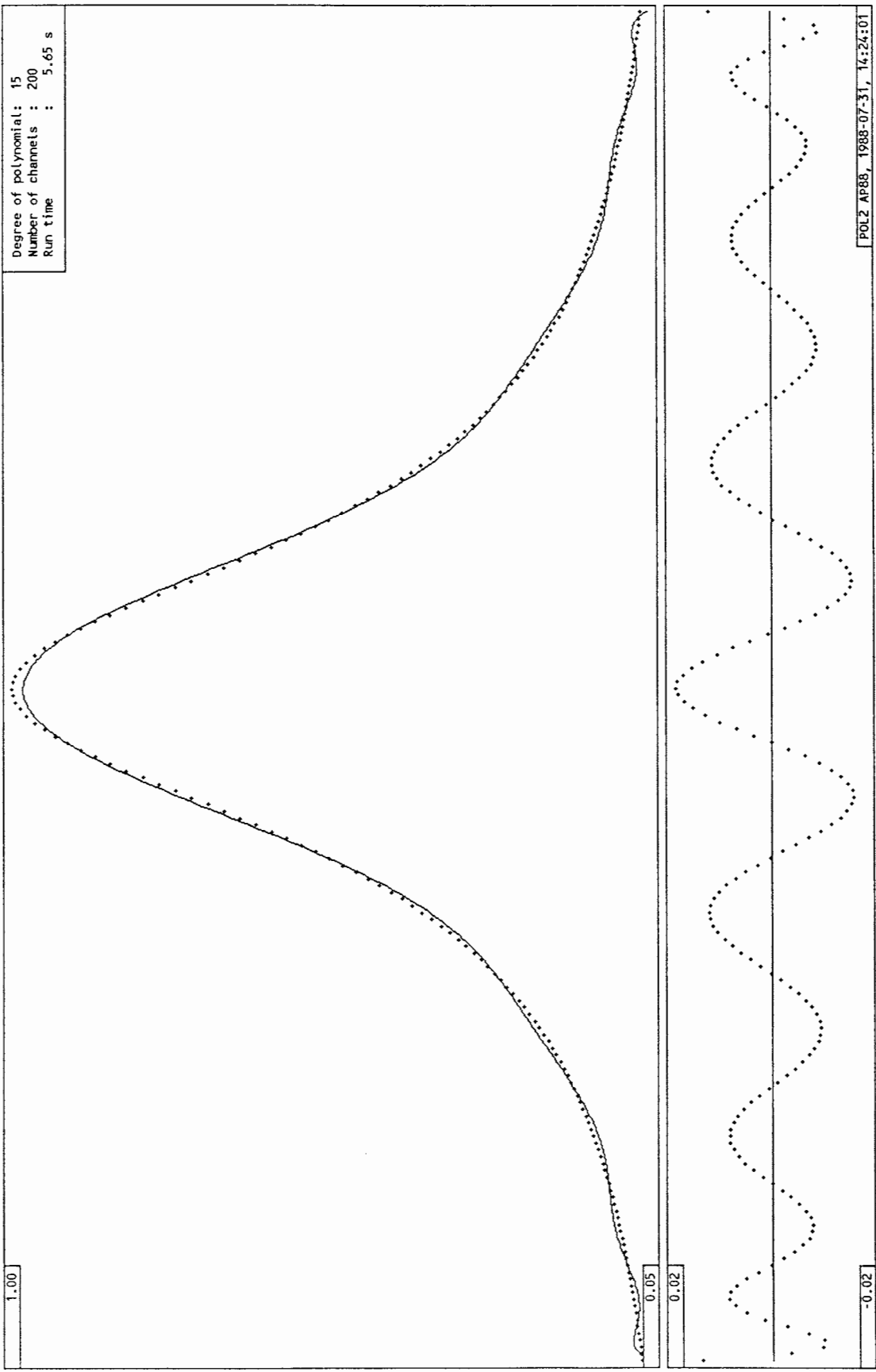
```

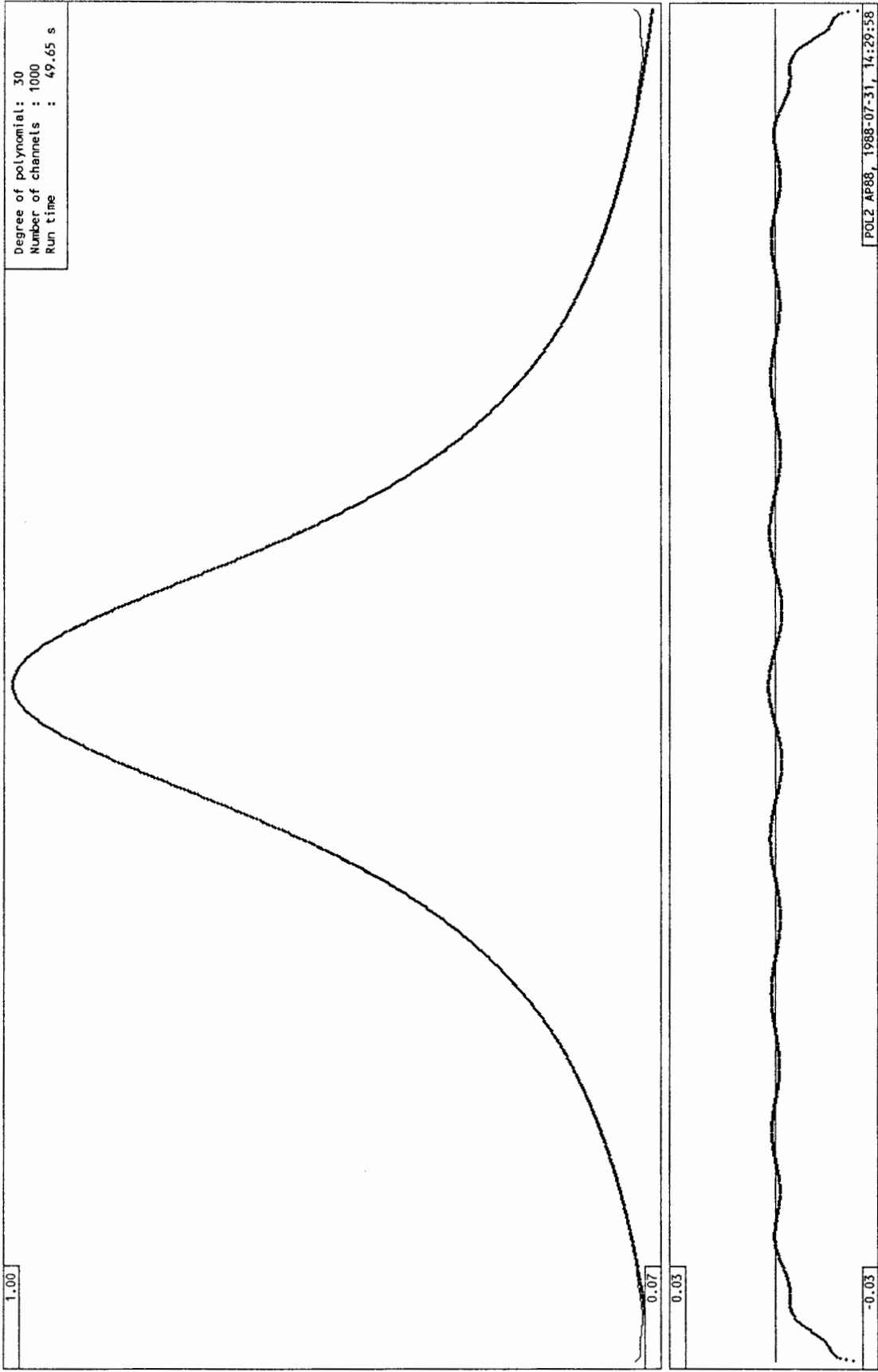
POL2

Programmet anpassar ett n:te grads-polynom ($n < 100$) till en Lorentz-topp. Programmet gör i stort sett exakt samma beräkningar som POL1 men beräkningen av A-matrisen är effektiviserad genom att summan $\sum x^p$ endast beräknas en gång för varje p-värde och beräkningen av B-vektorn går snabbare genom att så få beräkningar som möjligt sker i den innersta slingan. Man kan notera att anpassningen vid kanterna av Lorentz-toppen blir usel om höga gradtal används, dessa anpassningar är alltså oanvändbara för extrapolation.

Utskrift från testkörning av programmet POL2:

```
Degree of polynomial (0..99) :15
No of channels ( 16..1000) :200
Run time :6.65 s.
Press any key to continue. <Return>
Plot on Roland plotter? (Y/*) :N
Plot on HP-Laserjet? (Y/*) :Y
New calculation? (Y/*) :N
```





```

1 PROGRAM POL2
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5 INTEGER N,L,ASLASKRANGE
6 PARAMETER (N=1000, L=100, ASLASKRANGE=L+L-2)
7 REAL*8 Y(1:N),X(1:N),DX,DET,TIME,THETAESTIMATE(1:L),A(1:L,1:L)
8 REAL*8 B(1:L),SIGIN,RESIDUE(1:N),SIGMA(1:N),F
9 REAL*8 ASLASK(0:ASLASKRANGE),SL1,SL2,SL3
10 REAL JX,JY,JXL,JYL,DJY
11 INTEGER R,K,I,NCH,POLDEGREE,LRANGE
12 INTEGER*2 HO,MI,SE,HN,ANS,KBDCHK,KBDINC
13 INTEGER*2 YEAR,MONTH,DAY
14 CHARACTER SV*1,JETTXT*40
15 LOGICAL SING,ERRORBAR
16 EXTERNAL F
17
18 SIGIN=1.0D0
19 ERRORBAR=.FALSE.
20 CALL ZEROMAT(SIGMA,N,1,N,1)
21 CALL ADDCMAT(SIGMA,1.0D0,N,1,N,1)
22
23 10 WRITE(*,1000) 'Degree of polynomial (0..',L-1,'):'
24 1000 FORMAT(' ',A,12,A)
25 READ(*,*,ERR=10) POLDEGREE
26 IF (POLDEGREE.LT.0.OR.POLDEGREE.GT.L-1) THEN
27 GOTO 10
28 ENDIF
29 LRANGE=POLDEGREE+1
30
31 20 WRITE(*,1100) 'No of channels (' ,LRANGE,'..',N,'):'
32 1100 FORMAT(' ',A,13,A,14,A)
33 READ(*,*,ERR=20) NCH
34 IF (NCH.LT.LRANGE.OR.NCH.GT.N) THEN
35 GOTO 20
36 ENDIF
37
38 DX=1.5D0/(NCH-1)
39 DO 30 I=1,NCH
40 X(I)=(I-NCH*0.5D0)*DX
41 Y(I)=1.0D0/(1.0D0+25.0D0*X(I)*X(I))
42 30 CONTINUE
43
44 CALL GETTIM(HO,MI,SE,HN)
45 TIME=HO*3600.0D0+MI*60.0D0+SE+HN*0.01D0
46
47 CALL ZEROMAT(A,LRANGE,LRANGE,L,L)
48 CALL ZEROMAT(B,LRANGE,1,L,1)
49 CALL ZEROMAT(ASLASK,LRANGE+LRANGE-1,1,ASLASKRANGE+1,1)
50
51 DO 40 I=1,NCH
52 ASLASK(0)=ASLASK(0)+1.0D0
53 DO 40 R=1,LRANGE+LRANGE-2
54 ASLASK(R)=ASLASK(R)+X(I)**R
55 40 CONTINUE
56
57 DO 50 R=1,LRANGE

```

```

58 DO 50 K=R,LRANGE
59 A(R,K)=ASLASK(R+K-2)
60 A(K,R)=A(R,K)
61 50 CONTINUE
62
63 DO 70 I=1,NCH
64 SL1=Y(I)
65 SL2=X(I)
66 SL3=SL2
67 B(1)=B(1)+SL1
68 DO 60 R=2,LRANGE
69 B(R)=B(R)+SL1*SL2
70 SL2=SL2*SL3
71 60 CONTINUE
72 70 CONTINUE
73
74 CALL INVMAT(A,LRANGE,DET,SING,L,L)
75 IF (SING) THEN
76 WRITE(*,*) 'The A-matrix is singular.'
77 GOTO 110
78 ENDIF
79 CALL MULTMAT(A,B,THETAESTIMATE,LRANGE,LRANGE,1,L,L,1,L,1)
80
81 CALL GETTIM(HO,MI,SE,HN)
82 TIME=HO*3600.0D0+MI*60.0D0+SE+HN*0.01D0-TIME
83
84 WRITE(*,1200) 'Run time:',TIME,' s.'
85 1200 FORMAT(' ',A,F8.2,A)
86 WRITE(*,*) 'Press any key to continue.'
87 80 ANS=KBDCHK()
88 IF (ANS.EQ.0) THEN
89 GOTO 80
90 ENDIF
91 ANS=KBDINC()
92
93 CALL SCRIPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
94 f THETAESTIMATE,LRANGE,ERRORBAR)
95
96 90 WRITE(*,*) 'Plot on Roland plotter? (Y/*):'
97 READ(*,1300) SV
98 1300 FORMAT(A)
99 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
100 CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
101 f THETAESTIMATE,LRANGE,ERRORBAR)
102 GOTO 90
103 ENDIF
104
105 100 WRITE(*,*) 'Plot on HP-Laserjet? (Y/*): '
106 READ(*,1300) SV
107 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
108 CALL HPSTART('L')
109 CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
110 f THETAESTIMATE,LRANGE,ERRORBAR)
111 JX=900
112 JXL=1119-JX
113 JYL=(3+1)*13
114 JY=719-JYL
115 CALL HPBOX(JX,JY,JXL,JYL)
116 JX=906
117 JY=719-16

```

```

118      DJY=-13
119
120      WRITE(JETTXT,1400) 'Degree of polynomial:',POLDEGREE
121 1400  FORMAT(' ',A,I4)
122      CALL HPPRINT(JX,JY,32,JETTXT)
123      JY=JY+DJY
124
125      WRITE(JETTXT,1500) 'Number of channels : ',NCH
126 1500  FORMAT(' ',A,I4)
127      CALL HPPRINT(JX,JY,32,JETTXT)
128      JY=JY+DJY
129
130      WRITE(JETTXT,1600) 'Run time           :',TIME,' s.'
131 1600  FORMAT(' ',A,F8.2,A)
132      CALL HPPRINT(JX,JY,32,JETTXT)
133      JY=JY+DJY
134
135      CALL GETTIM(HO,MI,SE,HN)
136      CALL GETDAT(YEAR,MONTH,DAY)
137      WRITE(JETTXT,1800)
138      f 'POL2 AP88, ',YEAR,'-',MONTH,'-',DAY,', ',HO,':',MI,':',SE
139 1800  FORMAT(A,I4.4,5(A,I2.2))
140      CALL HPTBOX(921.0,0.0,31,JETTXT)
141
142      CALL HPEJECT
143      GOTO 100
144  ENDIF
145
146 110  WRITE(*,1700) 'New calculation? (Y/*):'
147 1700  FORMAT('/' ',A)
148      READ(*,1300) SV
149      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
150          GOTO 10
151      ELSE
152          STOP
153      ENDIF
154      END
155
156      REAL*8 FUNCTION F(THETA,L,X)
157  C    Anders Persson, 1988
158
159      IMPLICIT LOGICAL (A-Z)
160      INTEGER L,I
161      REAL*8 THETA(1:L),X,SLASK
162
163      SLASK=THETA(1)
164      IF (X.NE.0.0D0) THEN
165          DO 10 I=2,L
166              SLASK=SLASK+THETA(I)*X**(I-1)
167 10    CONTINUE
168      ENDIF
169      F=SLASK
170
171      RETURN
172      END

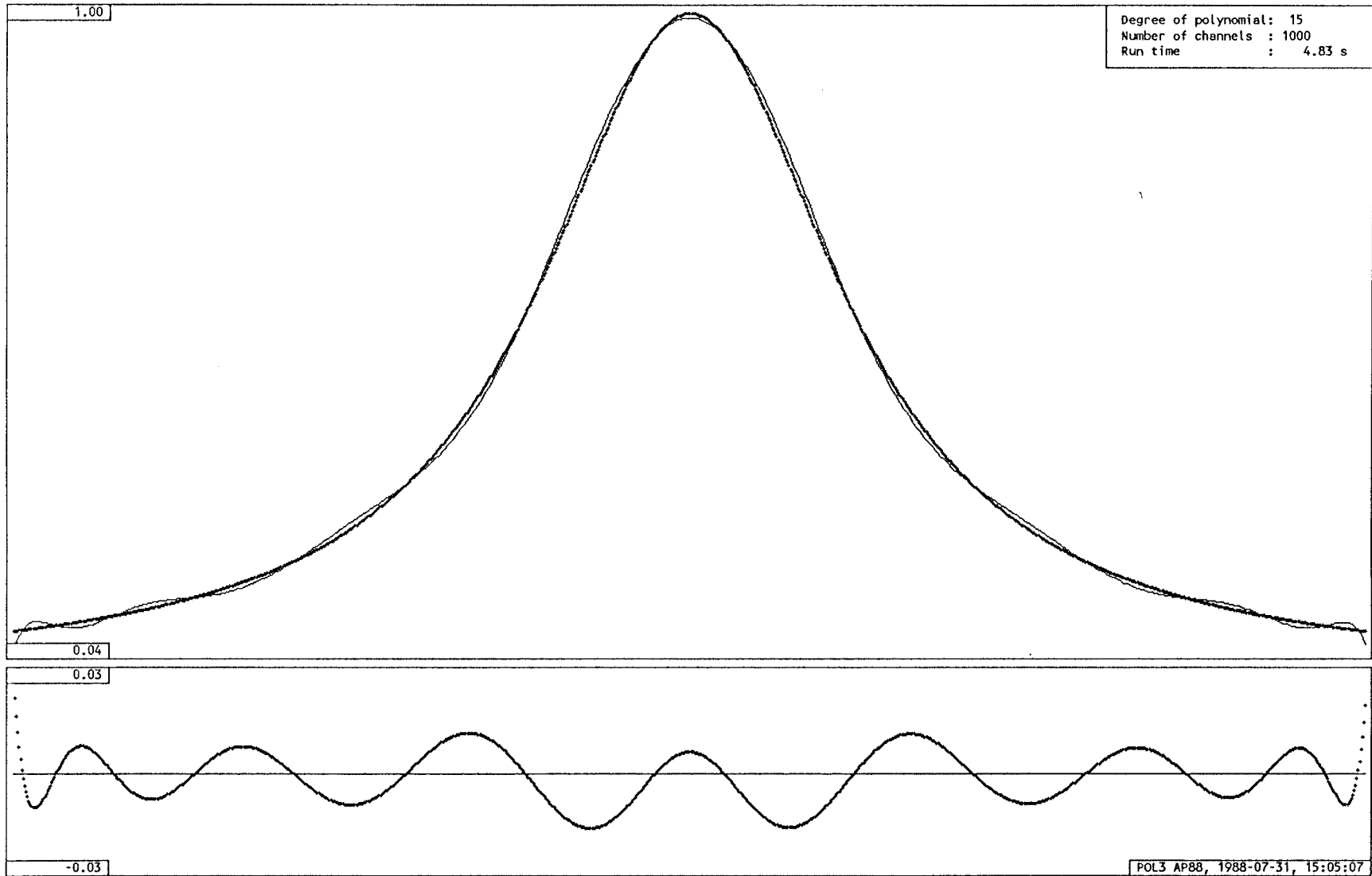
```


POL3

Programmet utför exakt samma beräkning som POL2 men genom att beräkna summorna, $\sum x^p$ på ett listigare sätt kan exekveringstiden drastiskt minskas, speciellt för anpassning av polynom av lågt gradtal till långa mätserier. POL3 exekverar ungefär fem gånger snabbare än POL2 vid anpassning till polynom av grad mindre än 10 till 1000 mätpunkter. Skillnaden blir större för längre mätserier eftersom beräkningen av A-matrisen i POL3 är oberoende av mätseriernas längd. Hur summaberäkningarna konverteras till beräkningar av polynom beskrivs närmare i nästa program: CALCPOLC.

Utskrift från testkörning av programmet POL3:

```
Degree of polynomial (0..49) :15
No of channels ( 16..1000) :200
Run time :2.91s.
Press any key to continue. <Return>
Plot on Roland plotter? (Y/*) :N
Plot on HP-Laserjet? (Y/*) :Y
New calculation? (Y/*) :N
```



```

1 PROGRAM POL3
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5 INTEGER N,L,POLVECRANGE,CRANGE
6 PARAMETER ( N=1000, L=50, POLVECRANGE=L+L-2 )
7 PARAMETER ( CRANGE=((L+L)*(L+L-1))/2 )
8 REAL*8 Y(1:N),X(1:N),DX,DET,TIME,THETAESTIMATE(1:L),A(1:L,1:L)
9 REAL*8 B(1:L),SIGIN,RESIDUE(1:N),SIGMA(1:N),F,SL1,SL2,SL3
10 REAL*8 POL,POLVEC(0:POLVECRANGE),C(1:CRANGE)
11 REAL JX,JY,JXL,JYL,DJY
12 INTEGER R,K,I,NCH,POLDEGREE,LRANGE,S
13 INTEGER*2 HO,MI,SE,HN,ANS,KBDINC,KBDCHK
14 INTEGER*2 YEAR,MONTH,DAY
15 CHARACTER SV*1,JETTXT*40
16 LOGICAL SING,ERRORBAR
17 EXTERNAL F
18
19 S =CRANGE
20 SIGIN =1.000
21 ERRORBAR=.FALSE.
22
23 CALL READPC(C,S)
24 CALL ZEROMAT(SIGMA,N,1,N,1)
25 CALL ADDCMAT(SIGMA,1.000,N,1,N,1)
26
27 10 WRITE(*,1000) 'Degree of polynomial (0..',L-1,'):'
28 1000 FORMAT(' ',A,I2,A)
29 READ(*,*,ERR=10) POLDEGREE
30 IF (POLDEGREE.LT.0.OR.POLDEGREE.GT.L-1) THEN
31 GOTO 10
32 ENDIF
33 LRANGE=POLDEGREE+1
34
35 20 WRITE(*,1100) 'No of channels (',LRANGE,'..',N,'):'
36 1100 FORMAT(' ',A,I3,A,I4,A)
37 READ(*,*,ERR=20) NCH
38 IF (NCH.LT.LRANGE.OR.NCH.GT.N) THEN
39 GOTO 20
40 ENDIF
41
42 DX=1.500/(NCH-1)
43 DO 30 I=1,NCH
44 X(I)=(I-NCH*0.500)*DX
45 Y(I)=1.000/(1.000+25.000*X(I)*X(I))
46 30 CONTINUE
47
48 CALL GETTIM(HO,MI,SE,HN)
49 TIME=HO*3600.000+MI*60.000+SE+HN*0.0100
50 CALL ZEROMAT(B,LRANGE,1,L,1)
51
52 DO 40 I=0,LRANGE+LRANGE-2
53 IF (MOD(I,2).EQ.0) THEN
54 POLVEC(I)=2.000*POL(I,NCH/2,C)*(DX**(I))
55 ELSE
56 POLVEC(I)=0.000
57 ENDIF

```

```

58 40 CONTINUE
59
60 POLVEC(0)=NCH
61 DO 50 R=1,LRANGE
62 DO 50 K=R,LRANGE
63 A(R,K)=POLVEC(R+K-2)
64 A(K,R)=POLVEC(R+K-2)
65 50 CONTINUE
66
67 DO 70 I=1,NCH
68 SL1=Y(I)
69 SL2=X(I)
70 SL3=SL2
71 B(1)=B(1)+SL1
72 DO 60 R=2,LRANGE
73 B(R)=B(R)+SL1*SL2
74 SL2=SL2*SL3
75 60 CONTINUE
76 70 CONTINUE
77
78 CALL INVMAT(A,LRANGE,DET,SING,L,L)
79 IF (SING) THEN
80 WRITE(*,*) 'The A-matrix is singular.'
81 GOTO 110
82 ENDIF
83
84 CALL MULTMAT(A,B,THETAESTIMATE,LRANGE,LRANGE,1,L,L,1,L,1)
85
86 CALL GETTIM(HO,MI,SE,HN)
87 TIME=HO*3600.000+MI*60.000+SE+HN*0.0100-TIME
88
89 WRITE(*,1200) 'Run time:',TIME,'s.'
90 1200 FORMAT(' ',A,FB.2,A)
91
92 WRITE(*,*) 'Press any key to continue.'
93 80 ANS=KBDCHK()
94 IF (ANS.EQ.0) THEN
95 GOTO 80
96 ENDIF
97 ANS=KBDINC()
98
99 CALL SCRIPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
100 f THETAESTIMATE,LRANGE,ERRORBAR)
101
102 90 WRITE(*,*) 'Plot on Roland plotter? (Y/*):'
103 READ(*,1300) SV
104 1300 FORMAT(A)
105 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
106 CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
107 f THETAESTIMATE,LRANGE,ERRORBAR)
108 GOTO 90
109 ENDIF
110
111 100 WRITE(*,*) 'Plot on HP-Laserjet? (Y/*):'
112 READ(*,1300) SV
113 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
114 CALL HPSTART('L')
115 CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
116 f THETAESTIMATE,LRANGE,ERRORBAR)
117 JX=900

```

```

118      JXL=1119-JX
119      JYL=(3+1)*13
120      JY=719-JYL
121      CALL HPBOX(JX,JY,JXL,JYL)
122      JX=906
123      JY=719-16
124      DJY=-13
125
126      WRITE(JETTXT,1400) 'Degree of polynomial:',POLDEGREE
127 1400  FORMAT(' ',A,14)
128      CALL HPPRINT(JX,JY,32,JETTXT)
129      JY=JY+DJY
130
131      WRITE(JETTXT,1500) 'Number of channels : ',NCH
132 1500  FORMAT(' ',A,14)
133      CALL HPPRINT(JX,JY,32,JETTXT)
134      JY=JY+DJY
135
136      WRITE(JETTXT,1600) 'Run time           :',TIME,' s.'
137 1600  FORMAT(' ',A,F8.2,A)
138      CALL HPPRINT(JX,JY,32,JETTXT)
139
140      CALL GETTIM(HO,MI,SE,HN)
141      CALL GETDAT(YEAR,MONTH,DAY)
142      WRITE(JETTXT,1800)
143 1800  f 'POL3 AP88, ',YEAR,'-',MONTH,'-',DAY,' ',HO,':',MI,':',SE
144      FORMAT(A,14.4,5(A,12.2))
145      CALL HPTBOX(921.0,0.0,31,JETTXT)
146
147      CALL HPEJECT
148      GOTO 100
149      ENDIF
150
151 110   WRITE(*,1700) 'New calculation? (Y/*):'
152 1700  FORMAT('/ ',A)
153      READ(*,1300) SV
154      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
155          GOTO 10
156      ELSE
157          STOP
158      ENDIF
159      END
160
161      REAL*8 FUNCTION F(THETA,L,X)
162  C    Anders Persson, 1988
163
164      IMPLICIT LOGICAL (A-Z)
165      INTEGER L,I
166      REAL*8 THETA(1:L),X,SLASK
167
168      SLASK=THETA(1)
169      IF (X.NE.0.0D0) THEN
170          DO 10 I=2,L
171              SLASK=SLASK+THETA(I)*X**(I-1)
172 10    CONTINUE
173      ENDIF
174      F=SLASK
175
176      RETURN
177      END

```

```

178
179      DOUBLE PRECISION FUNCTION POL(Q,N,C)
180  C    Anders Persson, 1987
181
182      IMPLICIT LOGICAL (A-Z)
183      REAL*8 C(1:*),PROD,SUM
184      INTEGER Q,N,I,J,OFFS
185
186      OFFS=(Q*(Q+1))/2
187      SUM=1.0D0
188      DO 10 I=1,Q+1
189          PROD=1.0D0
190          DO 20 J=1,I
191              PROD=PROD*(N-J)
192 20    CONTINUE
193          SUM=SUM+PROD*C(OFFS+I)
194 10    CONTINUE
195      POL=SUM
196
197      RETURN
198      END
199
200      SUBROUTINE READPC(C,S)
201  C    Anders Persson, 1988
202
203      IMPLICIT LOGICAL (A-Z)
204      REAL*8 C(1:*)
205      INTEGER S,I
206      LOGICAL EX
207
208      INQUIRE(FILE='CCOEFF.DAT',EXIST=EX)
209
210      IF (.NOT.EX) THEN
211          WRITE(*,*)
212          f'The unformatted file: CCoeff:DAT containing the Cq,i values is '
213          WRITE(*,*)
214          f'missing. You must supply one or create one by running the '
215          WRITE(*,*)
216          f'program CALCPOLC.'
217          STOP
218      ENDIF
219
220      OPEN(10,FILE='CCOEFF.DAT',ACCESS='DIRECT',
221  f    FORM='UNFORMATTED',RECL=8)
222
223      DO 10 I=1,S
224          READ(10,REC=I) C(I)
225 10    CONTINUE
226
227      RETURN
228      END

```

CALCPOLC

Summan

$$S(q, n) = \sum_{k=1}^n k^q, \quad q \text{ heltal } \geq 0, \quad n \text{ heltal } \geq 1 \quad (52)$$

kan skrivas som ett polynom $P(q, n)$, av grad $q+1$. $P(q, n)$ blir:

$$P(q, n) = 1 + \sum_{i=1}^{q+1} C_{q,i} \cdot \frac{(n-1)!}{(n-1-i)!} = 1 + \sum_{i=1}^{q+1} \left[C_{q,i} \cdot \prod_{j=1}^i (n-j) \right] =$$

$$1 + C_{q,1}(n-1) + C_{q,2}(n-1)(n-2) + \dots + C_{q,q+1}(n-1)(n-2) \cdots (n-(q+1)) \quad (53)$$

Koefficienterna $C_{q,i}$, $q=0, 1, 2, \dots$, $i=1, 2, \dots, q+1$, beräknas med formeln:

$$C_{q,i} = \frac{1}{i} \cdot \frac{1}{(i-1)!} \sum_{k=0}^{i-1} \binom{i-1}{k} (-1)^k (i+1-k)^q = \frac{1}{i} \sum_{k=0}^{i-1} \frac{(-1)^k (i+1-k)^q}{k! \cdot (i-1-k)!} \quad (54)$$

Delen av $C_{q,i}$ -formlerna efter den första $1/i$ termen blir alltid heltal ≥ 1 . Exempel: $q=5 \Rightarrow C_{5,1}=32, C_{5,2}=211/2, C_{5,3}=285/3, C_{5,4}=125/4, C_{5,5}=20/5$ och $C_{5,6}=1/6$. Koefficienten $C_{q,1}$ blir alltid 2^q , koefficienten $C_{q,q+1}$ blir alltid $1/(q+1)$. Den sista anmärkningen medför följande identitet:

$$q! = \sum_{k=0}^q \binom{q}{k} (-1)^k (q+2-k)^q \quad (55)$$

Koefficienterna för $q \leq 20$ går bra att beräkna i FORTRAN med dubbel precision, för större q -värden räcker inte antalet signifikanta siffror till. Speciellt blir $C_{q,i}$ -koefficienterna för stora i dåliga. Jag vet tyvärr inte hur man skall bevisa uttrycken för C -koefficienterna. Jag fick fram dem genom att notera symmetrier vid beräkning av den diskreta Fouriertransformen av punktföljden $f(k)=k^q \cdot e^{-k}$. Möjligen kan ett induktionsbevis användas. För beräkning av Fouriertransformen av ovanstående punktföljd se noteringar i anslutning till programmet TFEXP. Programmet CALCPOLC beräknar polynomkoefficienterna $C_{q,i}$ med hjälp av n -siffrig heltalsaritmetik. Programmet räknar exakt enligt formel (54). Som resultat erålls filer med exakta värden eller med värden i dubbel precision. Programmet POL3 använder en rutin READPC för att läsa in en fil: CCOEFF.DAT tillverkad med hjälp av CALCPOLC. Programmet utnyttjar heltalsvektorer där fyra siffror lagras i varje vektorelement. Endast fyra siffror lagras i ett vektorelement eftersom fem eller fler siffror skulle ge större tal än vad integer-variabler i den använda FORTRAN kompilatorn klarar av.

Lagringsformat: Ex. $A = -1234567$ lagras enl.

1 2 3 4 248 249 250 ← Element nr:
A= [0] [0] [0] [0] [0] [-123] [4567]
FNZA=249=First Non-Zero element in A ↑

På de följande två sidorna finns en lista över koefficienterna för $q \leq 30$.

C(0,1)=	1/1	C(9,7)=	11130/7	C(14,2)=	4766585/2	C(17,11)=	22210622434/11	C(20,11)=	101990314020959/11
C(1,1)=	2/1	C(9,8)=	1110/8	C(14,3)=	129442951/3	C(17,12)=	2766584522/12	C(20,12)=	21291952650150/12
C(1,2)=	1/2	C(9,9)=	54/9	C(14,4)=	885423630/4	C(17,13)=	235168752/13	C(20,13)=	3145861276831/13
C(2,1)=	4/1	C(9,10)=	1/10	C(14,5)=	2314233285/5	C(17,14)=	13549578/14	C(20,14)=	332528480620/14
C(2,2)=	5/2	C(10,1)=	1024/1	C(14,6)=	2873141271/6	C(17,15)=	517140/15	C(20,15)=	25236629540/15
C(2,3)=	1/3	C(10,2)=	58025/2	C(14,7)=	1925136213/7	C(17,16)=	12444/16	C(20,16)=	1369189248/16
C(3,1)=	8/1	C(10,3)=	465751/3	C(14,8)=	753655760/8	C(17,17)=	170/17	C(20,17)=	52351194/17
C(3,2)=	19/2	C(10,4)=	1132670/4	C(14,9)=	181092340/9	C(17,18)=	1/18	C(20,18)=	1370565/18
C(3,3)=	9/3	C(10,5)=	1144165/5	C(14,10)=	27457430/10	C(18,1)=	262144/1	C(20,19)=	23275/19
C(3,4)=	1/4	C(10,6)=	563409/6	C(14,11)=	2650648/11	C(18,2)=	387158345/2	C(20,20)=	230/20
C(4,1)=	16/1	C(10,7)=	147147/7	C(14,12)=	161070/12	C(18,3)=	33972448951/3	C(20,21)=	1/21
C(4,2)=	65/2	C(10,8)=	21120/8	C(14,13)=	5915/13	C(18,4)=	601616805790/4	C(21,1)=	2097152/1
C(4,3)=	55/3	C(10,9)=	1650/9	C(14,14)=	119/14	C(18,5)=	3612997293605/5	C(21,2)=	10458256051/2
C(4,4)=	14/4	C(10,10)=	65/10	C(14,15)=	1/15	C(18,6)=	9650629410813/6	C(21,3)=	2188563950925/3
C(4,5)=	1/5	C(10,11)=	1/11	C(15,1)=	32768/1	C(18,7)=	13461181659199/7	C(21,4)=	77279066272045/4
C(5,1)=	32/1	C(11,1)=	2048/1	C(15,2)=	14316139/2	C(18,8)=	10866668017920/8	C(21,5)=	835664518638180/5
C(5,2)=	211/2	C(11,2)=	175099/2	C(15,3)=	522538389/3	C(18,9)=	5440287930870/9	C(21,6)=	3779879619149865/6
C(5,3)=	285/3	C(11,3)=	1921029/3	C(15,4)=	4556561101/4	C(18,10)=	1771429211695/10	C(21,7)=	8599555939908780/7
C(5,4)=	125/4	C(11,4)=	6129101/4	C(15,5)=	14770823340/5	C(18,11)=	387549682091/11	C(21,8)=	11078105508262980/8
C(5,5)=	20/5	C(11,5)=	7997660/5	C(15,6)=	22426222182/6	C(18,12)=	58176221220/12	C(21,9)=	8758257497935506/9
C(5,6)=	1/6	C(11,6)=	5088028/6	C(15,7)=	18274230975/7	C(18,13)=	6058947050/13	C(21,10)=	4497968808080814/10
C(6,1)=	64/1	C(11,7)=	1740585/7	C(15,8)=	8708038053/8	C(18,14)=	438412422/14	C(21,11)=	1563339417432793/11
C(6,2)=	665/2	C(11,8)=	337227/8	C(15,9)=	2564579160/9	C(18,15)=	21823818/15	C(21,12)=	378785698472909/12
C(6,3)=	1351/3	C(11,9)=	37620/9	C(15,10)=	483124070/10	C(18,16)=	728688/16	C(21,13)=	65334010525784/13
C(6,4)=	910/4	C(11,10)=	2365/10	C(15,11)=	59265206/11	C(18,17)=	15504/17	C(21,14)=	8133788486131/14
C(6,5)=	245/5	C(11,11)=	77/11	C(15,12)=	4744558/12	C(18,18)=	189/18	C(21,15)=	736314553260/15
C(6,6)=	27/6	C(11,12)=	1/12	C(15,13)=	243880/13	C(18,19)=	1/19	C(21,16)=	48512846756/16
C(6,7)=	1/7	C(12,1)=	4096/1	C(15,14)=	7700/14	C(19,1)=	524288/1	C(21,17)=	2311510740/17
C(7,1)=	128/1	C(12,2)=	527345/2	C(15,15)=	135/15	C(19,2)=	1161737179/2	C(21,18)=	78391929/18
C(7,2)=	2059/2	C(12,3)=	7859215/3	C(16,1)=	65536/1	C(19,3)=	136276954149/3	C(21,19)=	1836065/19
C(7,3)=	6069/3	C(12,4)=	32566534/4	C(16,2)=	42981185/2	C(19,4)=	3042056477901/4	C(21,20)=	28105/20
C(7,4)=	5901/4	C(12,5)=	54115061/5	C(16,3)=	2104469695/3	C(19,5)=	22279600567420/5	C(21,21)=	252/21
C(7,5)=	2380/5	C(12,6)=	43613856/6	C(16,4)=	23305343894/4	C(19,6)=	71167403169296/6	C(21,22)=	1/22
C(7,6)=	434/6	C(12,7)=	19012708/7	C(16,5)=	93181501141/5	C(19,7)=	117340082684405/7	C(22,1)=	4194304/1
C(7,7)=	35/7	C(12,8)=	4775628/8	C(16,6)=	171754378614/6	C(19,8)=	111261193820479/8	C(22,2)=	31376865305/2
C(7,8)=	1/8	C(12,9)=	713427/9	C(16,7)=	168620069982/7	C(19,9)=	65269547326620/9	C(22,3)=	8764714059751/3
C(8,1)=	256/1	C(12,10)=	63635/10	C(16,8)=	96646573452/8	C(19,10)=	24926009259515/10	C(22,4)=	388583895311150/4
C(8,2)=	6305/2	C(12,11)=	3289/11	C(16,9)=	34353829653/9	C(19,11)=	6422025396787/11	C(22,5)=	5091266178101125/5
C(8,3)=	26335/3	C(12,12)=	90/12	C(16,10)=	7878943930/10	C(19,12)=	1143840557951/12	C(22,6)=	27294821852687235/6
C(8,4)=	35574/4	C(12,13)=	1/13	C(16,11)=	1194306542/11	C(19,13)=	143001479920/13	C(22,7)=	72576327138420105/7
C(8,5)=	20181/5	C(13,1)=	8192/1	C(16,12)=	120944460/12	C(19,14)=	12635133380/14	C(22,8)=	108302505514275600/8
C(8,6)=	5418/6	C(13,2)=	1586131/2	C(16,13)=	8158878/13	C(19,15)=	787593510/15	C(22,9)=	98660680487618040/9
C(8,7)=	714/7	C(13,3)=	31964205/3	C(16,14)=	359380/14	C(19,16)=	34211514/16	C(22,10)=	58235914386824460/10
C(8,8)=	44/8	C(13,4)=	170691885/4	C(16,15)=	9860/15	C(19,17)=	1007760/17	C(22,11)=	23258041817274330/11
C(8,9)=	1/9	C(13,5)=	357256900/5	C(16,16)=	152/16	C(19,18)=	19095/18	C(22,12)=	6487553497580610/12
C(9,1)=	512/1	C(13,6)=	359412053/6	C(16,17)=	1/17	C(19,19)=	209/19	C(22,13)=	1293461845833885/13
C(9,2)=	19171/2	C(13,7)=	195715520/7	C(17,1)=	131072/1	C(19,20)=	1/20	C(22,14)=	187340837817749/14
C(9,3)=	111645/3	C(13,8)=	61993360/8	C(17,2)=	129009091/2	C(20,1)=	1048576/1	C(22,15)=	19914821338291/15
C(9,4)=	204205/4	C(13,9)=	11909898/9	C(17,3)=	8460859965/3	C(20,2)=	3485735825/2	C(22,16)=	1561032948112/16
C(9,5)=	156660/5	C(13,10)=	1413412/10	C(17,4)=	118631189165/4	C(20,3)=	546269553775/3	C(22,17)=	90120040076/17
C(9,6)=	58107/6	C(13,11)=	103103/11	C(17,5)=	582394350740/5	C(20,4)=	15346559343654/4	C(22,18)=	3800957391/18
		C(13,12)=	4459/12	C(17,6)=	1295462151439/6	C(20,5)=	136719659882421/5	C(22,19)=	115113229/19
		C(13,13)=	104/13	C(17,7)=	1520714938470/7	C(20,6)=	520451422752492/6	C(22,20)=	2426270/20
		C(13,14)=	1/14	C(17,8)=	1038439231050/8	C(20,7)=	1009888064644536/7	C(22,21)=	33649/21
		C(14,1)=	16384/1	C(17,9)=	440184869982/9	C(20,8)=	1118690827068716/8	C(22,22)=	275/22
				C(17,10)=	121022212883/10	C(20,9)=	763956667086679/9	C(22,23)=	1/23
						C(20,10)=	339455649181285/10	C(23,1)=	8388608/1

C(23, 2)= 94134790219/2
C(23, 3)= 35090233104309/3
C(23, 4)= 1951684190615501/4
C(23, 5)= 30936180963917900/5
C(23, 6)= 196155019146911770/6
C(23, 7)= 607905438960048075/7
C(23, 8)= 1047298876766900505/8
C(23, 9)= 1094909310390456000/9
C(23, 10)= 739255738742687100/10
C(23, 11)= 337332416194116420/11
C(23, 12)= 107596237285822260/12
C(23, 13)= 24596019339255000/13
C(23, 14)= 4103574413100120/14
C(23, 15)= 505977979230405/15
C(23, 16)= 46452381456195/16
C(23, 17)= 3183193669480/17
C(23, 18)= 162338230505/18
C(23, 19)= 6103221971/19
C(23, 20)= 166064899/20
C(23, 21)= 3166548/21
C(23, 22)= 39974/22
C(23, 23)= 299/23
C(23, 24)= 1/24

C(24, 1)= 16777216/1
C(24, 2)= 282412759265/2
C(24, 3)= 140455067207455/3
C(24, 4)= 9793511186181814/4
C(24, 5)= 187568769974122901/5
C(24, 6)= 1404021314992300290/6
C(24, 7)= 5059398530827296370/7
C(24, 8)= 10033595329862152620/8
C(24, 9)= 11996391980671460505/9
C(24, 10)= 9226722436560014100/10
C(24, 11)= 4787244733072084140/11
C(24, 12)= 1736083500909805800/12
C(24, 13)= 451940508035392260/13
C(24, 14)= 86149635535756800/14
C(24, 15)= 12199222080786600/15
C(24, 16)= 1295668463985720/16
C(24, 17)= 103749867506835/17
C(24, 18)= 6267620049075/18
C(24, 19)= 284402669925/19
C(24, 20)= 9590584850/20
C(24, 21)= 235728955/21
C(24, 22)= 4085950/22
C(24, 23)= 47150/23
C(24, 24)= 324/24
C(24, 25)= 1/25

C(25, 1)= 33554432/1
C(25, 2)= 847255055011/2
C(25, 3)= 562102681589085/3
C(25, 4)= 49108010998116525/4
C(25, 5)= 1135206131030919220/5
C(25, 6)= 10015717974920224931/6
C(25, 7)= 41879209561610671250/7
C(25, 8)= 95361756499586669950/8
C(25, 9)= 129997515136576757670/9
C(25, 10)= 113490338782831615605/10

C(25, 11)= 66673659233425923780/11
C(25, 12)= 27356330244899559540/12
C(25, 13)= 8063250613405297440/13
C(25, 14)= 1744185041071744260/14
C(25, 15)= 281337188828342400/15
C(25, 16)= 34225585968543840/16
C(25, 17)= 3163166079108750/17
C(25, 18)= 222834648439260/18
C(25, 19)= 11955673447575/19
C(25, 20)= 485804951775/20
C(25, 21)= 14776621860/21
C(25, 22)= 329705805/22
C(25, 23)= 5217550/23
C(25, 24)= 55250/24
C(25, 25)= 350/25
C(25, 26)= 1/26

C(26, 1)= 67108864/1
C(26, 2)= 2541798719465/2
C(26, 3)= 2249257981411351/3
C(26, 4)= 246102157672171710/4
C(26, 5)= 6860344797183631845/5
C(26, 6)= 71245231955472493737/6
C(26, 7)= 345049394467805594931/7
C(26, 8)= 900135018057890700800/8
C(26, 9)= 1395336907865354246650/9
C(26, 10)= 1378391241747724529325/10
C(26, 11)= 913574249583931900965/11
C(26, 12)= 422305952417119297800/12
C(26, 13)= 140241838832573723700/13
C(26, 14)= 34226026229481461340/14
C(26, 15)= 6245580062325222660/15
C(26, 16)= 863172150293587680/16
C(26, 17)= 9116257392501340/17
C(26, 18)= 7397024399454690/18
C(26, 19)= 461948117390760/19
C(26, 20)= 22157577434850/20
C(26, 21)= 810890632695/21
C(26, 22)= 22359855375/22
C(26, 23)= 454927005/23
C(26, 24)= 6598800/24
C(26, 25)= 64350/25
C(26, 26)= 377/26
C(26, 27)= 1/27

C(27, 1)= 134217728/1
C(27, 2)= 7625463267259/2
C(27, 3)= 8999573724364869/3
C(27, 4)= 1232760046342269901/4
C(27, 5)= 41408170940773962780/5
C(27, 6)= 505576968485491088004/6
C(27, 7)= 2831640387697917253185/7
C(27, 8)= 8446264556988821902131/8
C(27, 9)= 14853504096711433167300/9
C(27, 10)= 16557640567090324069225/10
C(27, 11)= 12341282236754907340905/11
C(27, 12)= 6403551631006482772365/12
C(27, 13)= 2385691696073151429600/13
C(27, 14)= 65363223227479564800/14
C(27, 15)= 134155307226685023900/15

C(27, 16)= 20919506617316213220/16
C(27, 17)= 2504098507358611800/17
C(27, 18)= 231706038982140450/18
C(27, 19)= 16635986747269890/19
C(27, 20)= 927257243522610/20
C(27, 21)= 39997171354140/21
C(27, 22)= 1325167306320/22
C(27, 23)= 33278103495/23
C(27, 24)= 619897005/24
C(27, 25)= 8271900/25
C(27, 26)= 74529/26
C(27, 27)= 405/27
C(27, 28)= 1/28

C(28, 1)= 268435456/1
C(28, 2)= 22876524019505/2
C(28, 3)= 36005920360726735/3
C(28, 4)= 6172799805435714374/4
C(28, 5)= 249681785690986046581/5
C(28, 6)= 3580446950339211578808/6
C(28, 7)= 23158700070068829113484/7
C(28, 8)= 78848021400597314372364/8
C(28, 9)= 156981305524103153575131/9
C(28, 10)= 196987550334704997928775/10
C(28, 11)= 164653027408149212160085/11
C(28, 12)= 9587653439839183381650/12
C(28, 13)= 39803235376030602786765/13
C(28, 14)= 12190175180195086086600/14
C(28, 15)= 2800117147901756026200/15
C(28, 16)= 489786919721060648640/16
C(28, 17)= 65993279749771225620/17
C(28, 18)= 6906513248019280350/18
C(28, 19)= 564425773927538250/19
C(28, 20)= 36108388861244700/20
C(28, 21)= 1807195013313690/21
C(28, 22)= 70476019399500/22
C(28, 23)= 2123841790200/23
C(28, 24)= 48775528620/24
C(28, 25)= 834966405/25
C(28, 26)= 10284183/26
C(28, 27)= 85869/27
C(28, 28)= 434/28
C(28, 29)= 1/29

C(29, 1)= 536870912/1
C(29, 2)= 68629840493971/2
C(29, 3)= 144046557966926445/3
C(29, 4)= 30900004947539298605/4
C(29, 5)= 1504263513951351993860/5
C(29, 6)= 25312810438065467098237/6
C(29, 7)= 188850047510889844486680/7
C(29, 8)= 732790892675444658464760/8
C(29, 9)= 1648661076641628850123674/9
C(29, 10)= 2323844359205858130791656/10
C(29, 11)= 2172823879232495543849795/11
C(29, 12)= 1407289922126058596121535/12
C(29, 13)= 652832748704267622396360/13
C(29, 14)= 222655863078956894085765/14
C(29, 15)= 56992049546623182505800/15
C(29, 16)= 11126494783159787053080/16

C(29, 17)= 1677665955216942709800/17
C(29, 18)= 197217031462137552270/18
C(29, 19)= 18195026726570045350/19
C(29, 20)= 1322701940013676950/20
C(29, 21)= 75866679154145880/21
C(29, 22)= 3428143459502190/22
C(29, 23)= 121448222364300/23
C(29, 24)= 3343230005700/24
C(29, 25)= 70484655150/25
C(29, 26)= 1112639346/26
C(29, 27)= 12688515/27
C(29, 28)= 98455/28
C(29, 29)= 464/29
C(29, 30)= 1/30

C(30, 1)= 1073741824/1
C(30, 2)= 205890058352825/2
C(30, 3)= 576254861708199751/3
C(30, 4)= 154644071295663419470/4
C(30, 5)= 9056481088655651261765/5
C(30, 6)= 178693936580409621681519/6
C(30, 7)= 1536113190525184222991677/7
C(30, 8)= 6783968081589891770669520/8
C(30, 9)= 17219401659091733159701500/9
C(30, 10)= 27210949027906068288831890/10
C(30, 11)= 2839773090995804656989196/11
C(30, 12)= 20467592866871257293429750/12
C(30, 13)= 10546948403985805309670575/13
C(30, 14)= 3992670694888621033682835/14
C(30, 15)= 1134528655824927814178565/15
C(30, 16)= 246142460860339562408160/16
C(30, 17)= 41324481977064755829480/17
C(30, 18)= 5424789552997556202930/18
C(30, 19)= 561117605993538459270/19
C(30, 20)= 45971769466857261300/20
C(30, 21)= 2991768881404886310/21
C(30, 22)= 154713978722696250/22
C(30, 23)= 6342900796245390/23
C(30, 24)= 205028972506800/24
C(30, 25)= 5175831039600/25
C(30, 26)= 100525917492/26
C(30, 27)= 1467917766/27
C(30, 28)= 15543710/28
C(30, 29)= 112375/29
C(30, 30)= 495/30
C(30, 31)= 1/31

Kommentarer till programlistningen:

Rad	Kommentar
1-13	Deklarationer, SLASK är en hjälpvektor där delresultat lagras. C innehåller den senast beräknade koeficienten. SL är en hjälpvektor där talen $(i+1-k)^q$ lagras för att de inte skall behöva beräknas flera gånger om. FNZ, FNZS och FNZC är pekare till det första elementet skilt från noll i vektorerna.
15-22	Initiering av konstanter och vektorer.
24-28	Läs in till hur högt gradtal, q, som koeficienterna skall beräknas. $q \approx 20$ ger en exekveringstid på 10-20 sekunder men beräkningstiden ökar kraftigt med ökad q. För $q=100$ är det fråga om timmar.
30-63	Läs in i vilka format koeficienterna skall lagras.
65-69	Lagra talen $2, 3, 4, \dots, q_{\max} + 2$ i hjälpvektorn SL.
71-206	Beräkna och lagra koeficienterna.
72-76	Om $q \geq 2$ multipliceras talen i hjälpvektorn SL med $2, 3, 4, \dots, q_{\max} + 2$. Resultatet blir att SL innehåller talen $2^q, 3^q, 4^q, \dots, (q_{\max} + 2)^q$ dessa används senare i beräkningen av koeficienterna.
78-80	Skriv ut på skärmen (utan radframmatning) vilken C-koeficient som beräknas.
83-126	Beräkna delsumman $\binom{i}{k} (-1)^k (i+1-k)^q$ och addera den till C-vektorn.
84-88	Placera talet $(i+1-k)^q$ i vektorn SLASK.
90-118	Multiplicera talet i SLASK med $\binom{i-1}{k}$, multiplikationen sker med minimalt antal anrop av multiplikation- och divisions-rutinen, därav indexberäkningarna LOW och UP för slingorna och användandet av slaskvariablerna SLM och SLD där vanlig multiplikation sker tills talen blir av maximal storlek.
120-124	Om $(-1)^k$ är negativt subtraheras delsumman annars adderas den.
128-135	Division av C med talet $(i-1)!$ Nu fattas bara att dividera med i men eftersom resultatet då ej kan representeras exakt i vektorerna sparas talet som heltal.

- 137-141 Om koefficienterna skall lagras i formatet DOUBLE PRECISION (formatterat eller oformatterat) konverteras talet i C-vektorn till ett tal i dubbel precision. Observera att här divideras talet med i eftersom detta inte utfördes på raderna 128-135.
- 143-188 Beräkna formatsatsen och lagra det exakta talet om detta lagringsformat är valt.
- 190-197 Lagra talen som DOUBLE PRECISION-tal om detta lagringsformat är valt.
- 223-234 Rutinen nollställer talet lagrat i vektorn A.
- 236-248 Rutinen placerar talet 1 i vektorn A.
- 250-355 Rutinen adderar talet i vektor A med talet i vektor B och placerar resultatet i vektor A. Inga restriktioner på tecknet.
- 357-366 Rutinen subtraherar talet i vektor A med talet i vektor B och placerar resultatet i vektor A. Inga restriktioner på tecknet.
- 368-411 Rutinen multiplicerar talet i vektor A med talet MULT och placerar resultatet i vektor A. Inga restriktioner på tecknet.
- 413-451 Rutinen dividerar talet i vektor A med talet DIV och placerar resultatet i vektor A. Inga restriktioner på tecknet.
- 453-478 Rutinen konverterar talet i vektor A till ett tal i dubbel precision och placerar talet i variabeln C.
- 480-496 Rutinen kopierar talet i vektor A till vektor B.

IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
 Source File: CALCPOLC.FOR Options: /LBZ 08/02/88 14:14:35

```

1 PROGRAM CALCPOLC
2 C Anders Persson, 1988
3 C Must be compiled with compiler option /Z (optimization = off)
4 C
5
6 IMPLICIT LOGICAL (A-Z)
7 INTEGER SLASK(1:250),C(1:250),SL(1:250,2:252),FNZ(2:252)
8 INTEGER FIRSTNZ,NPRINT,QMAX,Q,ARRPOS,NSIFF,N,I,K,J,SLM,SLD,M
9 INTEGER FNZC,FNZS,LOW1,UP1,LOW2,UP2
10 REAL*8 DC
11 CHARACTER FMT*36,SLASK3*3,SV*1,FILECEX*40,FILECDPASCII*40
12 CHARACTER FILECDPUNFORM*40
13 LOGICAL CEX,DPASC,DPUNF
14
15 FMT = '(T2,A,Ia,A,Ib,A,T13,Ic,pppI4.4,A,Ib)'
16 N = 250
17 FNZC = 1
18 FNZS = 1
19 CALL ZEROVEC(SLASK,N,FNZS)
20 CALL ZEROVEC(C,N,FNZC)
21 NSIFF = 4
22 M = 10**NSIFF
23
24 10 WRITE(*,*) 'Qmax:'
25 READ(*,*,ERR=10) QMAX
26 IF (QMAX.LT.0.OR.QMAX.GT.250) THEN
27 GOTO 10
28 ENDIF
29
30 CEX = .FALSE.
31 DPASC = .FALSE.
32 DPUNF = .FALSE.
33
34 WRITE(*,*) 'Save file with the exact Cq,i values? (Y/*):'
35 READ(*,1000) SV
36 1000 FORMAT(A)
37 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
38 CEX=.TRUE.
39 WRITE(*,*) 'Filename:'
40 READ(*,1000) FILECEX
41 OPEN(UNIT=10,FILE=FILECEX)
42 ENDIF
43
44 WRITE(*,*) 'Save file with the formatted double precision Cq,i'
45 WRITE(*,*) 'values? (Y/*):'
46 READ(*,1000) SV
47 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
48 DPASC=.TRUE.
49 WRITE(*,*) 'Filename:'
50 READ(*,1000) FILECDPASCII
51 OPEN(UNIT=11,FILE=FILECDPASCII)
52 ENDIF
53
54 WRITE(*,*) 'Save file with the unformatted double precision Cq,i'
55 WRITE(*,*) 'values? (Y/*):'
56 READ(*,1000) SV
57 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN

```

```

58 DPUNF=.TRUE.
59 WRITE(*,*) 'Filename:'
60 READ(*,1000) FILECDPUNFORM
61 OPEN(UNIT=12,FILE=FILECDPUNFORM,ACCESS='DIRECT',
62 f FORM='UNFORMATTED',RECL=8)
63 ENDIF
64
65 DO 20 I=2,QMAX+2
66 FNZ(I)=1
67 CALL ZEROVEC(SL(1,I),N,FNZ(I))
68 SL(N,I)=1
69 20 CONTINUE
70
71 DO 90 Q=0,QMAX
72 IF (Q.GE.2) THEN
73 DO 30 I=2,QMAX+2
74 CALL MULTVEC(SL(1,I),N,NSIFF,I,FNZ(I))
75 30 CONTINUE
76 ENDIF
77
78 DO 80 I=1,Q+1
79 WRITE(*,1100) 'C(',Q,',',I,')'
80 1100 FORMAT('+',A,I3,A,I3,A)
81
82 CALL ZEROVEC(C,N,FNZC)
83 DO 60 K=0,I-1
84 IF (Q.NE.0) THEN
85 CALL COPYVEC(SL(1,I+1-K),SLASK,N,FNZ(I+1-K),FNZS)
86 ELSE
87 CALL ONEVEC(SLASK,N,FNZS)
88 ENDIF
89
90 IF (K.GT.I-1-K) THEN
91 LOW1 = K+1
92 UP1 = I-1
93 LOW2 = 2
94 UP2 = I-1-K
95 ELSE
96 LOW1 = I-K
97 UP1 = I-1
98 LOW2 = 2
99 UP2 = K
100 ENDIF
101
102 SLM=1
103 DO 40 J=LOW1,UP1
104 SLM=SLM*J
105 IF (SLM*(J+1).GE.M.OR.J.EQ.UP1) THEN
106 CALL MULTVEC(SLASK,N,NSIFF,SLM,FNZS)
107 SLM=1
108 ENDIF
109 40 CONTINUE
110
111 SLD=1
112 DO 50 J=LOW2,UP2
113 SLD=SLD*J
114 IF (SLD*(J+1).GE.M.OR.J.EQ.UP2) THEN
115 CALL DIVVEC(SLASK,N,NSIFF,SLD,FNZS)
116 SLD=1
117 ENDIF

```

```

118 50      CONTINUE
119
120      IF (MOD(K,2).EQ.0) THEN
121          CALL ADDVEC(C,SLASK,N,NSIFF,FNZC,FNZS)
122      ELSE
123          CALL SUBVEC(C,SLASK,N,NSIFF,FNZC,FNZS)
124      ENDIF
125
126 60      CONTINUE
127
128      SLD=1
129      DO 70 J=2,I-1
130          SLD=SLD*J
131          IF (SLD*(J+1).GE.M.OR.J.EQ.I-1) THEN
132              CALL DIVVEC(C,N,NSIFF,SLD,FNZC)
133              SLD=1
134          ENDIF
135 70      CONTINUE
136
137      IF (DPASC.OR.DPUNF) THEN
138          CALL CONVDP(C,N,NSIFF,DC,FNZC)
139          DC = DC/I
140          ARRPOS= Q*(Q+1)/2+I
141      ENDIF
142
143      IF (CEX) THEN
144          FIRSTNZ=FNZC
145          NPRINT=N-FNZC
146
147          IF (Q.LE.9) THEN
148              FMT(8:8)='1'
149          ELSEIF (Q.LE.99) THEN
150              FMT(8:8)='2'
151          ELSE
152              FMT(8:8)='3'
153          ENDIF
154
155          IF (I.LE.9) THEN
156              FMT(13:13)='1'
157              FMT(35:35)='1'
158          ELSEIF (I.LE.99) THEN
159              FMT(13:13)='2'
160              FMT(35:35)='2'
161          ELSE
162              FMT(13:13)='3'
163              FMT(35:35)='3'
164          ENDIF
165
166          IF (C(FIRSTNZ).LE.9) THEN
167              FMT(22:22)='1'
168          ELSEIF (C(FIRSTNZ).LE.99) THEN
169              FMT(22:22)='2'
170          ELSEIF (C(FIRSTNZ).LE.999) THEN
171              FMT(22:22)='3'
172          ELSE
173              FMT(22:22)='4'
174          ENDIF
175
176          WRITE(SLASK3,1200) NPRINT
177 1200      FORMAT(13)

```

```

178          FMT(24:26)=SLASK3
179
180          IF (NPRINT.EQ.0) THEN
181              FMT(24:31)=' '
182          ELSE
183              FMT(27:31)='14.4,'
184          ENDIF
185
186          WRITE(10,FMT) 'C(' ,Q ,',' ,I ,')=' ,C(FIRSTNZ) ,
187      f      (C(J) , J=FIRSTNZ+1,N) ,',' ,I
188          ENDIF
189
190          IF (DPASC) THEN
191              WRITE(11,1300) 'C(' ,ARRPOS ,')=' ,DC
192 1300      FORMAT(' ' ,A ,15 ,A ,D28.20)
193          ENDIF
194
195          IF (DPUNF) THEN
196              WRITE(12,REC=ARRPOS) DC
197          ENDIF
198
199 80      CONTINUE
200
201          IF (CEX) THEN
202              WRITE(10,1400) ' '
203 1400      FORMAT(' ' ,A)
204          ENDIF
205
206 90      CONTINUE
207
208          IF (CEX) THEN
209              CLOSE(10)
210          ENDIF
211
212          IF (DPASC) THEN
213              CLOSE(11)
214          ENDIF
215
216          IF (DPUNF) THEN
217              CLOSE(12)
218          ENDIF
219
220          STOP
221          END
222
223      SUBROUTINE ZEROVEC(A,N,FNZ)
224  C      Anders Persson, 1988
225
226      IMPLICIT LOGICAL (A-Z)
227      INTEGER A(1:*),N,I,FNZ
228
229      DO 10 I=FNZ,N
230          A(I)=0
231 10      CONTINUE
232      FNZ=N
233      RETURN
234      END
235
236      SUBROUTINE ONEVEC(A,N,FNZ)
237  C      Anders Persson, 1988

```

```

238
239 IMPLICIT LOGICAL (A-Z)
240 INTEGER A(1:*),N,I,FNZ
241
242 DO 10 I=FNZ,N
243   A(I)=0
244 10 CONTINUE
245   A(N)=1
246   FNZ=N
247   RETURN
248   END
249
250 SUBROUTINE ADDVEC(A,B,N,NSIFF,FNZA,FNZB)
251 C Anders Persson, 1988
252
253 IMPLICIT LOGICAL (A-Z)
254 INTEGER A(1:*),B(1:*),N,NSIFF,M,I,K,FNZA,FNZB,FNZ
255 LOGICAL NEG,DIFFSIGN,BLTZERO,ABSBLTABS
256
257 FNZ = MIN(FNZA,FNZB)
258 M = 10**NSIFF
259
260 IF (A(FNZA).GE.0.AND.B(FNZB).GE.0) THEN
261   NEG = .FALSE.
262   DIFFSIGN = .FALSE.
263 ELSEIF (A(FNZA).LE.0.AND.B(FNZB).LE.0) THEN
264   NEG = .TRUE.
265   DIFFSIGN = .FALSE.
266   A(FNZA) = -A(FNZA)
267   B(FNZB) = -B(FNZB)
268 ELSE
269   DIFFSIGN = .TRUE.
270 ENDIF
271
272 IF (.NOT.DIFFSIGN) THEN
273   DO 10 K=N,FNZ,-1
274     I=A(K)+B(K)
275     IF (I.GE.M) THEN
276       I = I-M
277       A(K-1)= A(K-1)+1
278     ENDIF
279     A(K)=I
280 10 CONTINUE
281
282   A(FNZ-1)=A(FNZ-1)+B(FNZ-1)
283   FNZA=N
284   DO 20 K=FNZ-1,N
285     IF (A(K).NE.0) THEN
286       FNZA=K
287       GOTO 21
288     ENDIF
289 20 CONTINUE
290 21 CONTINUE
291
292   IF (NEG) THEN
293     A(FNZA)=-A(FNZA)
294   ENDIF
295   RETURN
296 ENDIF
297

```

```

298 BLTZERO=B(FNZB).LT.0
299 IF (BLTZERO) THEN
300   B(FNZB)=-B(FNZB)
301 ELSE
302   A(FNZA)=-A(FNZA)
303 ENDIF
304
305 ABSBLTABS=.TRUE.
306 DO 30 K=FNZ,N
307   IF (B(K).LT.A(K)) THEN
308     ABSBLTABS=.TRUE.
309     GOTO 31
310   ELSEIF (B(K).GT.A(K)) THEN
311     ABSBLTABS=.FALSE.
312     GOTO 31
313   ENDIF
314 30 CONTINUE
315 31 CONTINUE
316
317 IF (.NOT.ABSBLTABS) THEN
318   DO 40 K=FNZ,N
319     I = A(K)
320     A(K) = B(K)
321     B(K) = I
322 40 CONTINUE
323
324   I = FNZA
325   FNZA = FNZB
326   FNZB = I
327 ENDIF
328
329 DO 50 K=N,FNZ,-1
330   I=A(K)-B(K)
331   IF (I.LT.0) THEN
332     I = I+M
333     A(K-1)= A(K-1)-1
334   ENDIF
335   A(K)=I
336 50 CONTINUE
337
338   A(FNZ-1)=A(FNZ-1)-B(FNZ-1)
339   FNZA=N
340   DO 60 K=FNZ-1,N
341     IF (A(K).NE.0) THEN
342       FNZA=K
343       GOTO 61
344     ENDIF
345 60 CONTINUE
346 61 CONTINUE
347
348 IF (BLTZERO.AND.(.NOT.ABSBLTABS)) THEN
349   A(FNZA)=-A(FNZA)
350 ELSEIF ((.NOT.BLTZERO).AND.ABSBLTABS) THEN
351   A(FNZA)=-A(FNZA)
352 ENDIF
353
354 RETURN
355 END
356
357 SUBROUTINE SUBVEC(A,B,N,NSIFF,FNZA,FNZB)

```

```

358 C Anders Persson, 1988
359
360 IMPLICIT LOGICAL (A-Z)
361 INTEGER A(1:*),B(1:*),N,NSIFF, FNZA, FNZB
362
363 B(FNZB)=-B(FNZB)
364 CALL ADDVEC(A,B,N,NSIFF, FNZA, FNZB)
365 RETURN
366 END
367
368 SUBROUTINE MULTVEC(A,N,NSIFF,MULT, FNZA)
369 C Anders Persson, 1988
370
371 IMPLICIT LOGICAL (A-Z)
372 INTEGER A(1:*),N,NSIFF,MULT,K,I,M,PLUS, FNZA, SAVEMULT
373 LOGICAL NEG
374
375 SAVEMULT = MULT
376 NEG = .FALSE.
377
378 IF (A(FNZA).LT.0.AND.MULT.GT.0) THEN
379 NEG = .TRUE.
380 ELSEIF (A(FNZA).GT.0.AND.MULT.LT.0) THEN
381 NEG = .TRUE.
382 ENDIF
383
384 A(FNZA) = ABS(A(FNZA))
385 MULT = ABS(MULT)
386 M = 10**NSIFF
387 PLUS = 0
388
389 DO 10 K=N, FNZA, -1
390 I=A(K)*MULT+PLUS
391 IF (I.GE.M) THEN
392 PLUS = I/M
393 I = I-PLUS*M
394 ELSE
395 PLUS = 0
396 ENDIF
397 A(K)=I
398 10 CONTINUE
399
400 A(FNZA-1)=A(FNZA-1)*MULT+PLUS
401 IF (A(FNZA-1).NE.0) THEN
402 FNZA=FNZA-1
403 ENDIF
404
405 IF (NEG) THEN
406 A(FNZA)=-A(FNZA)
407 ENDIF
408 MULT=SAVEMULT
409
410 RETURN
411 END
412
413 SUBROUTINE DIVVEC(A,N,NSIFF,DIV, FNZA)
414 C Anders Persson, 1988
415
416 IMPLICIT LOGICAL (A-Z)
417 INTEGER A(1:*),N,NSIFF,DIV,K,I,M,MD, FNZA, SAVEDIV

```

```

418 LOGICAL NEG
419
420 SAVEDIV = DIV
421 NEG = .FALSE.
422
423 IF (A(FNZA).LT.0.AND.DIV.GT.0) THEN
424 NEG = .TRUE.
425 ELSEIF (A(FNZA).GT.0.AND.DIV.LT.0) THEN
426 NEG = .TRUE.
427 ENDIF
428
429 A(FNZA) = ABS(A(FNZA))
430 DIV = ABS(DIV)
431 M = 10**NSIFF
432
433 DO 10 K=FNZA,N-1
434 I = A(K)/DIV
435 MD = A(K)-I*DIV
436 A(K+1) = A(K+1)+MD*M
437 A(K) = I
438 10 CONTINUE
439
440 A(N)=A(N)/DIV
441 IF (A(FNZA).EQ.0) THEN
442 FNZA=FNZA+1
443 ENDIF
444
445 IF (NEG) THEN
446 A(FNZA)=-A(FNZA)
447 ENDIF
448 DIV=SAVEDIV
449
450 RETURN
451 END
452
453 SUBROUTINE CONVDP(A,N,NSIFF,C, FNZA)
454 C Anders Persson, 1988
455
456 IMPLICIT LOGICAL (A-Z)
457 REAL*B C
458 INTEGER A(1:*),N,NSIFF,M,K, FNZA, LAST
459 LOGICAL NEG
460
461 NEG = A(FNZA).LT.0
462 A(FNZA) = ABS(A(FNZA))
463 LAST = MIN(N, FNZA+25/NSIFF)
464 C = 0.000
465
466 DO 10 K=FNZA, LAST
467 IF (A(K).NE.0) THEN
468 M=(N-K)*NSIFF
469 C=C+DBLE(A(K))*(10.000**M)
470 ENDIF
471 10 CONTINUE
472
473 IF (NEG) THEN
474 C=-C
475 ENDIF
476
477 RETURN

```

```
478     END
479
480     SUBROUTINE COPYVEC(A,B,N, FNZA, FNZB)
481 C     Anders Persson, 1988
482
483     IMPLICIT LOGICAL (A-Z)
484     INTEGER    A(1:*),B(1:*),N, FNZA, FNZB, I
485
486     DO 10 I=FNZB, FNZA
487         B(I)=0
488 10    CONTINUE
489
490     DO 20 I=FNZA, N
491         B(I)=A(I)
492 20    CONTINUE
493     FNZB=FNZA
494
495     RETURN
496     END
```

ARCROSS

Programmet anpassar polynom till mätdata från en mätning av fotojonisationstvårsnittet för Argon. Till de två räta delarna av kurvan anpassas räta linjer (polynom av grad 1) och till den första icke linjära delen anpassas polynom av valbart gradtal i tre intervall. Intervallen överlappar något för att få mjukare övergångar mellan de olika områdena. Istället för att överlappa intervallen för att få mjukare övergångar vore det naturligtvis snyggare att införa kravet att första och andraderivatorna skall vara kontinuerliga.

Utskrift från testexekvering av programmet ARCROSS:

```
Degree of polynomial (0..15)      :7
Plot errorbars? (Y/*)             :Y
Plot only up to the first edge? (Y/*) :Y
Plot on Roland plotter? (Y/*)     :N
Plot on HP-Laserjet? (Y/*)       :Y
Degree of polynomial:      7
Run time:      4.62 s.
```

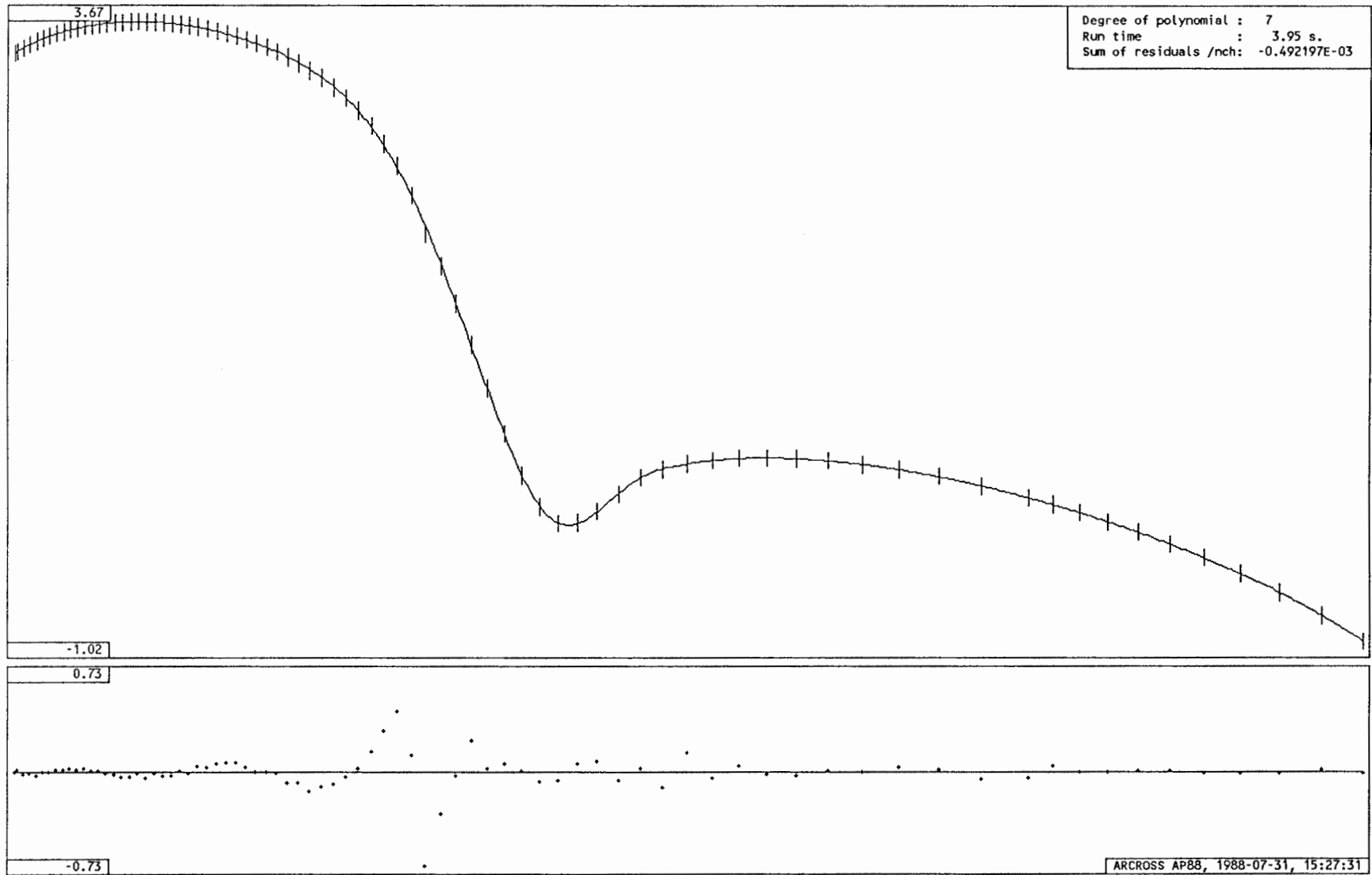
```
THETA( 1)= -1.2150      +/- 0.96106E-02
THETA( 2)= -2.3404      +/- 0.12465E-01
THETA( 3)= -4.7481      +/- 0.13769E-01
THETA( 4)= -2.7514      +/- 0.17519E-01
```

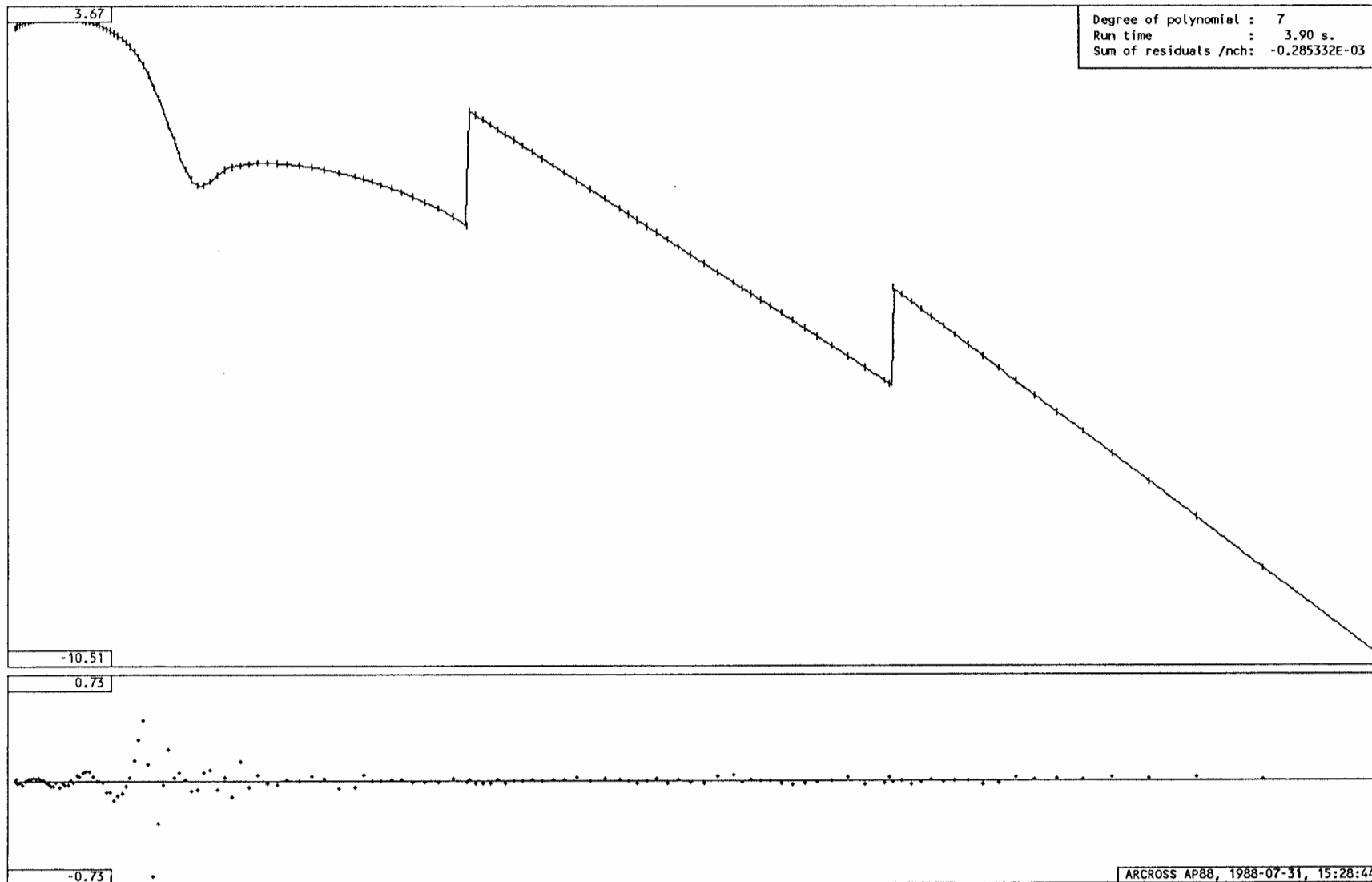
```
THETA( 5)=  3.5320      +/- 0.19287E-01
THETA( 6)= -0.89759     +/- 0.16465
THETA( 7)= -2.5185      +/-  1.1848
THETA( 8)=  0.34384     +/-  4.0394
THETA( 9)= -11.103      +/- 16.684
THETA(10)= -37.728      +/- 33.210
THETA(11)= -1.5623      +/- 62.820
THETA(12)=  97.717      +/- 104.75
```

```
THETA(13)= -0.67368E-01 +/- 0.35160E-01
THETA(14)=  1.9664      +/- 0.60406
THETA(15)= 22.693       +/- 5.3902
THETA(16)= -141.15     +/- 49.779
THETA(17)= -151.49     +/- 190.73
THETA(18)= 1669.0      +/- 1204.5
THETA(19)= 543.22      +/- 1796.5
THETA(20)= -7209.6     +/- 8862.5
```

```
THETA(21)= 0.16348     +/- 0.30442E-01
THETA(22)= -0.88166    +/- 0.19736
THETA(23)= -0.72682    +/- 0.69852
THETA(24)= 0.11928     +/- 2.6531
THETA(25)= -0.12968    +/- 3.7314
THETA(26)= -0.93966E-01 +/- 10.374
THETA(27)= -0.51016    +/- 5.3534
THETA(28)= 0.36433     +/- 12.133
```

```
Sum of residuals /nch: -0.492197E-03
Quit? (Y/*):Y
```



Indata till ARCROSS programmet.

1: 15.83 , 29.2	51: 42.75 , 1.77	101: 826.5 , .278
2: 15.89 , 29.5	52: 44.28 , 1.30	102: 885.6 , .237
3: 16.10 , 30.3	53: 45.92 , 1.03	103: 953.7 , .199
4: 16.31 , 31.1	54: 47.68 , .914	104: 1033. , .165
5: 16.53 , 31.8	55: 49.59 , .916	105: 1127. , .135
6: 16.75 , 32.5	56: 51.66 , 1.00	106: 1240. , .108
7: 16.98 , 33.1	57: 53.90 , 1.13	107: 1305. , .0955
8: 17.22 , 33.7	58: 56.35 , 1.28	108: 1378. , .0842
9: 17.46 , 34.2	59: 59.04 , 1.36	109: 1459. , .0736
10: 17.71 , 34.7	60: 61.99 , 1.42	110: 1550. , .0639
11: 17.97 , 35.1	61: 65.25 , 1.45	111: 1653. , .0549
12: 18.23 , 35.5	62: 68.88 , 1.48	112: 1771. , .0467
13: 18.50 , 35.8	63: 72.93 , 1.48	113: 1907. , .0393
14: 18.78 , 36.1	64: 77.49 , 1.47	114: 2066. , .0326
15: 19.07 , 36.3	65: 82.65 , 1.45	115: 2254. , .0266
16: 19.37 , 36.5	66: 88.56 , 1.41	116: 2480. , .0213
17: 19.68 , 36.6	67: 95.37 , 1.36	117: 2755. , .0166
18: 20.00 , 36.7	68: 103.3 , 1.29	118: 3100. , .0126
19: 20.32 , 36.8	69: 112.7 , 1.20	119: 3204. , .0117
20: 20.66 , 36.7	70: 124.0 , 1.10	120: 3263. , .0959
21: 21.01 , 36.7	71: 130.5 , 1.05	121: 3444. , .0827
22: 21.38 , 36.5	72: 137.8 , .987	122: 3646. , .0706
23: 21.75 , 36.3	73: 145.9 , .923	123: 3874. , .0598
24: 22.14 , 36.1	74: 155.0 , .856	124: 4133. , .0501
25: 22.54 , 35.7	75: 165.3 , .785	125: 4428. , .0414
26: 22.96 , 35.4	76: 177.1 , .709	126: 4768. , .0338
27: 23.39 , 34.9	77: 190.7 , .630	127: 5166. , .0271
28: 23.84 , 34.4	78: 206.6 , .547	128: 5635. , .0213
29: 24.31 , 33.8	79: 225.4 , .461	129: 6199. , .0164
30: 24.80 , 33.1	80: 245.0 , .381	130: 6888. , .0123
31: 25.30 , 32.3	81: 248.0 , 4.66	131: 7749. , .00889
32: 25.83 , 31.4	82: 258.3 , 4.23	132: 8856. , .00616
33: 26.38 , 30.5	83: 269.5 , 3.83	133: 10330. , .00403
34: 26.95 , 29.5	84: 281.8 , 3.45	134: 12400. , .00244
35: 27.55 , 28.3	85: 295.2 , 3.10	135: 15500. , .00132
36: 28.18 , 27.1	86: 310.0 , 2.76	136: 20660. , .000599
37: 28.83 , 25.7	87: 326.3 , 2.45	137: 31000. , .000196
38: 29.52 , 24.3	88: 344.4 , 2.16	138: 61990. , .000029
39: 30.24 , 22.7	89: 364.7 , 1.89	
40: 30.99 , 21.0	90: 387.4 , 1.64	
41: 31.79 , 19.1	91: 413.3 , 1.41	
42: 32.63 , 17.1	92: 442.8 , 1.20	
43: 33.51 , 15.0	93: 476.9 , 1.01	
44: 34.44 , 12.8	94: 516.6 , .836	
45: 35.42 , 10.3	95: 563.9 , .682	
46: 36.46 , 7.77	96: 619.9 , .546	
47: 37.57 , 6.10	97: 652.5 , .484	
48: 38.74 , 4.62	98: 688.8 , .426	
49: 39.99 , 3.41	99: 729.3 , .373	
50: 41.33 , 2.47	100: 774.9 , .324	

Kommentarer till programlistningen:

Rad	Kommentar
1-16	Deklarationer.
18-20	Antal mätpunkter är 138, data finns i filen ARGON.DAT, osäkerheten i mätvärden är 3% vilket i den LOG-skala som används blir lika stora fel för alla mätpunkter.
22-26	Läs in och logaritmera mätdata samt placera felet i vektorn SIGMA.
30-35	Välj gradtal på de tre polynom som används i anpassningen till den första delen av kurvan.
37-40	Välj om felgränser skall ritas.
42-43	Spara tiden före beräkningarna.
45-50	Anpassa polynom av valt gradtal till punkterna 1 till 51.
52-56	Anpassa polynom av valt gradtal till punkterna 48 till 62.
58-62	Anpassa polynom av valt gradtal till punkterna 59 till 80.
64-69	Anpassa rät linje till den första räta delen av kurvan (kanal 81-119).
71-75	Anpassa rät linje till den andra räta delen av kurvan (kanal 120-138).
77	Det totala antalet anpassade parametrar är $3*(POLDEG+1)+4$.
79-80	Spara exekveringstiden.
82-87	Välj om kurvor skall ritas med alla punkter eller endast den första delen innan de linjära områdena.
89	Rita anpassning, mätdata och residualvektor på skärmen.
91-95	Beräkna residualssumman för att få ett mått på hur bra anpassningen lyckats.
97-102	Rita på pen-plotter.
104-138	Rita och skriv ut diverse resultat på HP laserskrivare.
140-167	Skriv ut parametervärden med felgränser mm. på skärmen.
169-204	Rutinen beräknar funktionsvärdet $f(\theta;x)$. Funktionen är uppdelad i 5 intervall: $x = [-\infty, \ln(41)]$, $[\ln(41), \ln(62)]$, $[\ln(62), \ln(248)]$, $[\ln(248), \ln(3260)]$ och $[\ln(3260), \infty]$. I de tre första intervallen används polynom av valbart gradtal, i de två sista används förstgradspolynom.
206-256	Rutinen anpassar ett polynom av valbart gradtal till punkterna (x_i, y_i) , $i=FI, FI+1, \dots, LA$. x -värdena centreras kring $x=0$ innan summationerna av matriselementen (se ekvation 14) för att matrisen skall bli lättare att invertera, för övrigt utför rutinen samma beräkningar som i programmen POL1, POL2 och POL3.

```

1      PROGRAM ARCCROSS
2 C    Anders Persson, 1988
3
4      IMPLICIT LOGICAL (A-Z)
5      INTEGER N,L,MR
6      PARAMETER (N=138, L=16, MR=3*L+4)
7      REAL*8 Y(1:N),X(1:N),THETAEST(1:MR),THETASLASK(1:L)
8      REAL*8 RESIDUE(1:N),SIGMA(1:N),F,THETAERR(1:MR),ERRSLASK(1:L)
9      REAL*8 TIME,SLASK,XM(1:5),SUMRESDIVNCH
10     REAL JX,JY,JXL,JYL,DJY
11     INTEGER R,K,I,NCH,POLDEG,LR,J
12     INTEGER*2 HO,MI,SE,HN,YEAR,MONTH,DAY
13     CHARACTER SV*1,JETTXT*40
14     LOGICAL ERRORBAR
15     EXTERNAL F
16     COMMON XM
17 C
18     NCH=138
19     OPEN(UNIT=10,FILE='ARGON.DAT')
20     SLASK=LOG((1.0+0.03)/(1.0-0.03))
21
22     DO 10 I=1,138
23         READ(10,*) X(I),Y(I)
24         X(I)=LOG(X(I))
25         Y(I)=LOG(Y(I))
26         SIGMA(I)=SLASK
27 10    CONTINUE
28     CLOSE(10)
29
30 20    WRITE(*,1000) 'Degree of polynomial (0..',L-1,'):'
31 1000  FORMAT(' ',A,I2,A)
32     READ(*,*,ERR=20) POLDEG
33     IF (POLDEG.LT.0.OR.POLDEG.GT.L-1) THEN
34         GOTO 20
35     ENDIF
36
37     WRITE(*,*) 'Plot errorbars? (Y/*):'
38     READ(*,1100) SV
39 1100  FORMAT(A)
40     ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'
41
42     CALL GETTIM(HO,MI,SE,HN)
43     TIME=HO*3600.0+MI*60.0+SE+HN*0.01
44
45     LR=POLDEG+1
46     CALL FITPOL(THETASLASK,ERRSLASK,X,Y,SIGMA,XM(3),1,51,POLDEG)
47     DO 30 I=1,LR
48         THETAEST(I+4)=THETASLASK(I)
49         THETAERR(I+4)=ERRSLASK(I)
50 30    CONTINUE
51
52     CALL FITPOL(THETASLASK,ERRSLASK,X,Y,SIGMA,XM(4),48,62,POLDEG)
53     DO 40 I=1,LR
54         THETAEST(LR+I+4)=THETASLASK(I)
55         THETAERR(LR+I+4)=ERRSLASK(I)
56 40    CONTINUE
57

```

```

58     CALL FITPOL(THETASLASK,ERRSLASK,X,Y,SIGMA,XM(5),59,80,POLDEG)
59     DO 50 I=1,LR
60         THETAEST(2*LR+I+4)=THETASLASK(I)
61         THETAERR(2*LR+I+4)=ERRSLASK(I)
62 50    CONTINUE
63
64     LR=2
65     CALL FITPOL(THETASLASK,ERRSLASK,X,Y,SIGMA,XM(1),81,119,1)
66     DO 60 I=1,LR
67         THETAEST(I)=THETASLASK(I)
68         THETAERR(I)=ERRSLASK(I)
69 60    CONTINUE
70
71     CALL FITPOL(THETASLASK,ERRSLASK,X,Y,SIGMA,XM(2),120,138,1)
72     DO 70 I=1,LR
73         THETAEST(I+2)=THETASLASK(I)
74         THETAERR(I+2)=ERRSLASK(I)
75 70    CONTINUE
76
77     LR=3*(POLDEG+1)+4
78
79     CALL GETTIM(HO,MI,SE,HN)
80     TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
81
82     WRITE(*,*) 'Plot only up to the first edge? (Y/*):'
83     READ(*,1100) SV
84     NCH=138
85     IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
86         NCH=80
87     ENDIF
88
89     CALL SCRIPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,POLDEG+1,ERRORBAR)
90
91     SUMRESDIVNCH=0.0D0
92     DO 80 I=1,NCH
93         SUMRESDIVNCH=SUMRESDIVNCH + RESIDUE(I)
94 80    CONTINUE
95     SUMRESDIVNCH=SUMRESDIVNCH/NCH
96
97 90    WRITE(*,*) 'Plot on Roland plotter? (Y/*):'
98     READ(*,1100) SV
99     IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
100        CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,POLDEG+1,ERRORBAR)
101        GOTO 90
102    ENDIF
103
104 100   WRITE(*,*) 'Plot on HP-Laserjet? (Y/*): '
105     READ(*,1100) SV
106     IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
107         CALL HPSTART('L')
108         CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,POLDEG+1,ERRORBAR)
109         JX=870
110         JXL=1119-JX
111         JYL=(3+1)*13
112         JY=719-JYL
113         CALL HPBOX(JX,JY,JXL,JYL)
114         JX=876
115         JY=719-16
116         DJY=-13
117

```

```

118 WRITE(JETTXT,1300) 'Degree of polynomial :',POLDEG
119 CALL HPPRINT(JX,JY,40,JETTXT)
120 JY=JY+DJY
121
122 WRITE(JETTXT,1400) 'Run time           :',TIME,' s.'
123 CALL HPPRINT(JX,JY,40,JETTXT)
124 JY=JY+DJY
125
126 WRITE(JETTXT,1700) 'Sum of residuals /nch:',SUMRESDIVNCH
127 CALL HPPRINT(JX,JY,40,JETTXT)
128
129 CALL GETTIM(HO,MI,SE,HN)
130 CALL GETDAT(YEAR,MONTH,DAY)
131 WRITE(JETTXT,1800)
132 f 'ARCROSS AP88, ',YEAR,'-',MONTH,'-',DAY,' ',HO,':',MI,':',SE
133 1800 FORMAT(A,I4.4,5(A,I2.2))
134 CALL HPTBOX(903.0,0.0,34,JETTXT)
135
136 CALL HPEJECT
137 GOTO 100
138 ENDIF
139
140 WRITE(*,1200)
141 1200 FORMAT(/)
142 WRITE(*,1300) 'Degree of polynomial:',POLDEG
143 1300 FORMAT(' ',A,I4)
144 WRITE(*,1400) 'Run time:',TIME,' s.'
145 1400 FORMAT(' ',A,F8.2,A)
146 WRITE(*,1200)
147
148 DO 110 R=1,LR
149 WRITE(*,1500) 'THETA(',R,')=',THETAEST(R),' +/- ',THETAERR(R)
150 IF (R.EQ.2.OR.R.EQ.4.OR.MOD(R-4,POLDEG+1).EQ.0) THEN
151 WRITE(*,1600) ' '
152 ENDIF
153 110 CONTINUE
154 1500 FORMAT(' ',A,I2,A,G15.5,A,G15.5)
155 1600 FORMAT(' ',A)
156
157 WRITE(*,1700) 'Sum of residuals /nch:',SUMRESDIVNCH
158 1700 FORMAT(' ',A,G15.6)
159
160 WRITE(*,1600) 'Quit? (Y/*):'
161 READ(*,1100) SV
162 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
163 STOP
164 ELSE
165 GOTO 20
166 ENDIF
167 END
168
169 REAL*8 FUNCTION F(THETA,L,X)
170 C Anders Persson, 1988
171
172 IMPLICIT LOGICAL (A-Z)
173 INTEGER L,I
174 REAL*8 THETA(1:*),X,XM(1:5),SLASK,LOG41,LOG62,LOG248,LOG3260
175 PARAMETER (LOG41= 3.721588629, LOG62= 4.126973082)
176 PARAMETER (LOG248=5.513428746, LOG3260=8.089482474)
177 COMMON XM

```

```

178
179 IF (X.GE.LOG248.AND.X.LT.LOG3260) THEN
180 F=THETA(1)+THETA(2)*(X-XM(1))
181 ELSEIF (X.GE.LOG3260) THEN
182 F=THETA(3)+THETA(4)*(X-XM(2))
183 ELSEIF (X.LT.LOG41) THEN
184 SLASK=0.0D0
185 DO 10 I=1,L
186 SLASK=SLASK+THETA(I+4)*((X-XM(3))**(I-1))
187 10 CONTINUE
188 F=SLASK
189 ELSEIF (X.GE.LOG41.AND.X.LT.LOG62) THEN
190 SLASK=0.0D0
191 DO 20 I=1,L
192 SLASK=SLASK+THETA(L+I+4)*((X-XM(4))**(I-1))
193 20 CONTINUE
194 F=SLASK
195 ELSEIF (X.GE.LOG62.AND.X.LT.LOG248) THEN
196 SLASK=0.0D0
197 DO 30 I=1,L
198 SLASK=SLASK+THETA(2*L+I+4)*((X-XM(5))**(I-1))
199 30 CONTINUE
200 F=SLASK
201 ENDIF
202
203 RETURN
204 END
205
206 SUBROUTINE FITPOL(THETA,ERR,X,Y,SIGMA,XM,FI,LA,POLDEG)
207 C Anders Persson, 1988
208
209 IMPLICIT LOGICAL (A-Z)
210 INTEGER L
211 PARAMETER ( L=16 )
212 INTEGER FI,LA,POLDEG,I,LR,R,K
213 REAL*8 THETA(1:*),ERR(1:*),X(1:*),Y(1:*),SIGMA(1:*),XM
214 REAL*8 DET,A(1:L,1:L),B(1:L),C(1:L)
215 LOGICAL SING
216
217 XM=0.0D0
218 DO 10 I=FI,LA
219 XM=XM+X(I)
220 10 CONTINUE
221 XM=XM/(LA-FI+1)
222 LR=POLDEG+1
223
224 CALL ZEROMAT(A,L,L,L,L)
225 CALL ZEROMAT(B,L,1,L,1)
226 CALL ZEROMAT(C,L,1,L,1)
227
228 DO 40 I=FI,LA
229 B(1)=B(1)+Y(I)/SIGMA(I)**2
230 DO 20 R=2,LR
231 B(R)=B(R)+Y(I)*(X(I)-XM)**(R-1)/SIGMA(I)**2
232 20 CONTINUE
233 A(1,1)=A(1,1)+1.0D0/SIGMA(I)**2
234 DO 30 R=1,LR
235 DO 30 K=R,LR
236 IF (R+K.NE.2) THEN
237 A(R,K)=A(R,K)+(X(I)-XM)**(R+K-2)/SIGMA(I)**2

```

```
238         ENDIF
239         A(K,R)=A(R,K)
240 30      CONTINUE
241 40      CONTINUE
242
243      CALL INVMAT(A,LR,DET,SING,L,L)
244      IF (SING) THEN
245         WRITE(*,*) 'The A-matrix is singular.'
246         STOP
247      ENDIF
248
249      CALL MULTMAT(A,B,C,LR,LR,1,L,L,1,L,1)
250      DO 50 I=1,LR
251         THETA(I)=C(I)
252         ERR(I)=SQRT(ABS(A(I,I)))
253 50      CONTINUE
254
255      RETURN
256      END
```

EXP1

Programmet anpassar en exponential med bakgrund till en i datorn tillverkad testkurva. Eftersom anpassningsfunktionen inte kan transformeras till att bli linjär i parametrarna måste en iterativ metod användas för att hitta minimum av χ^2 summan. Programmet använder Newton's metod vilket innebär att första och andra derivatorna med avseende på θ måste beräknas för χ^2 . (ekv. 37,42 och/eller 46).

Anpassad funktion:

$$f(\theta; x) = \theta_1 \cdot \exp(-\theta_2 \cdot x) + \theta_3 \quad (56)$$

Förstaderivator:

$$\frac{\partial f}{\partial \theta_1} = \exp(-\theta_2 \cdot x) \quad (57)$$

$$\frac{\partial f}{\partial \theta_2} = -x \cdot \theta_1 \cdot \exp(-\theta_2 \cdot x) \quad (58)$$

$$\frac{\partial f}{\partial \theta_3} = 1 \quad (59)$$

Andraderivator:

$$\frac{\partial^2 f}{\partial \theta_1 \partial \theta_2} = -x \cdot \exp(-\theta_2 \cdot x) \quad (60)$$

$$\frac{\partial^2 f}{\partial \theta_2 \partial \theta_2} = x^2 \cdot \theta_1 \cdot \exp(-\theta_2 \cdot x) \quad (61)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} = \frac{\partial^2 f}{\partial \theta_k \partial \theta_j} \quad (62)$$

alla övriga andraderivator är noll.

Programmet beräknar startvärden till parametervektorn om detta väljs. Startvektorn beräknas enligt följande algoritm: Först beräknas bakgrunden (θ_3) genom att ta medelvärdet på den sista tiondelen av indatakurvan (dock alltid minst 2 punkter), därefter subtraheras bakgrunden bort och en rät linje anpassas till den logaritmerade kurvans första punkter. Antalet punkter bestäms genom att värdet i den sista kanalen skall vara större än en tiondel av värdet i den första kanalen. Från denna räta linje får man startvärden på intensitet (θ_1) och sönderfallskonstant (θ_2). Om startvärden istället ges från tangentbordet kan parametervärden låsas till startvärdet om så önskas.

Utskrift från testkörning av programmet EXP1:

No of channels (3..1024) :300
Plot errorbars? (Y/*) :Y

Parameter values.

Theta 1 (Intensity) :200
Theta 2 (Decay constant) :6
Theta 3 (Background) :10
Calculate start values from data? (Y/*) :Y

Press Q to quit iteration.

It:0, TH1: 207.26 ,TH2: 5.5107 ,TH3: 10.979
It:1, TH1: 201.66 ,TH2: 6.1885 ,TH3: 10.956 ,STEP: 0.139383
It:2, TH1: 202.46 ,TH2: 6.2591 ,TH3: 10.616 ,STEP: 0.473135E-01
It:3, TH1: 202.46 ,TH2: 6.2591 ,TH3: 10.612 ,STEP: 0.327852E-03
It:4, TH1: 202.81 ,TH2: 6.2850 ,TH3: 10.689 ,STEP: 0.129645E-01
It:5, TH1: 202.81 ,TH2: 6.2851 ,TH3: 10.689 ,STEP: 0.381779E-04

Press any key to continue. <Return>

Plot on Roland plotter? (Y/*) :N

Plot on HP-Laserjet? (Y/*) :Y

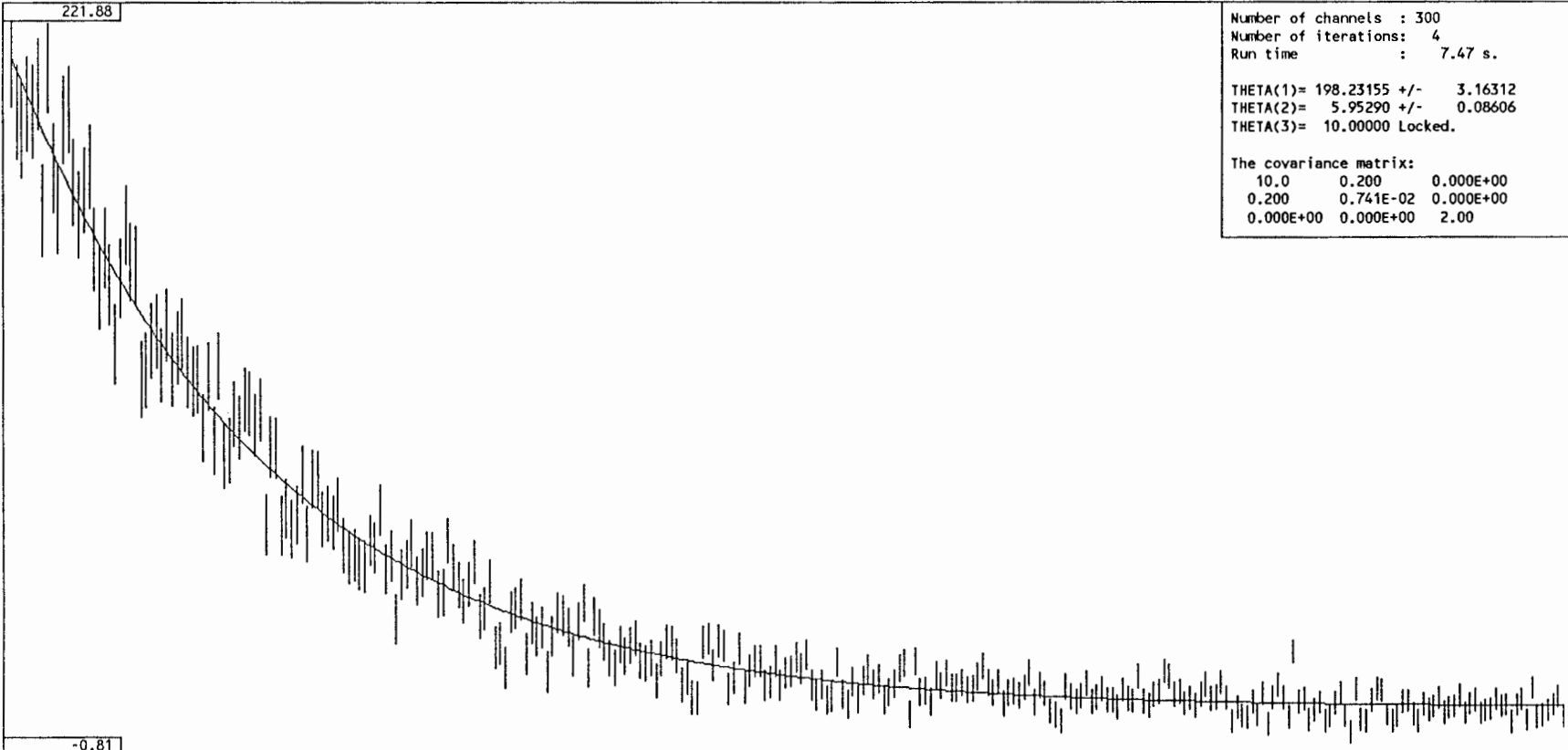
Number of channels : 300
Number of iterations : 5
Run time : 10.66 s.

THETA(1)= 202.80978 +/- 3.36685
THETA(2)= 6.28512 +/- 0.12131
THETA(3)= 10.68880 +/- 0.37307

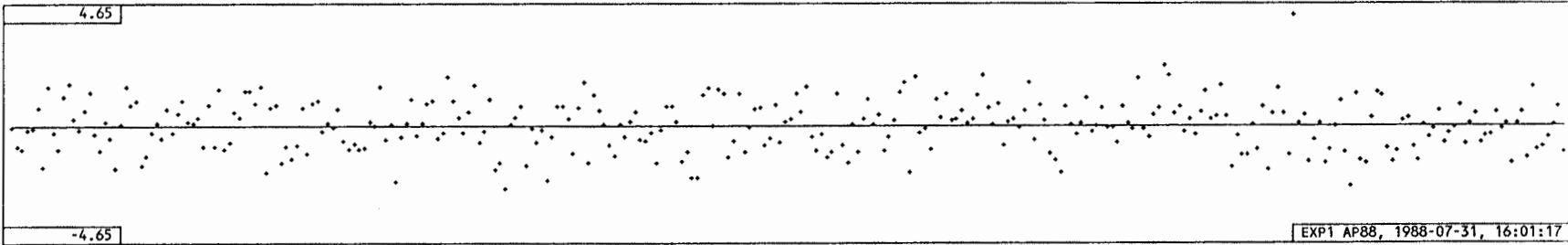
The covariance matrix for theta:

11.3	0.281	0.280
0.281	0.147E-01	0.294E-01
0.280	0.294E-01	0.139

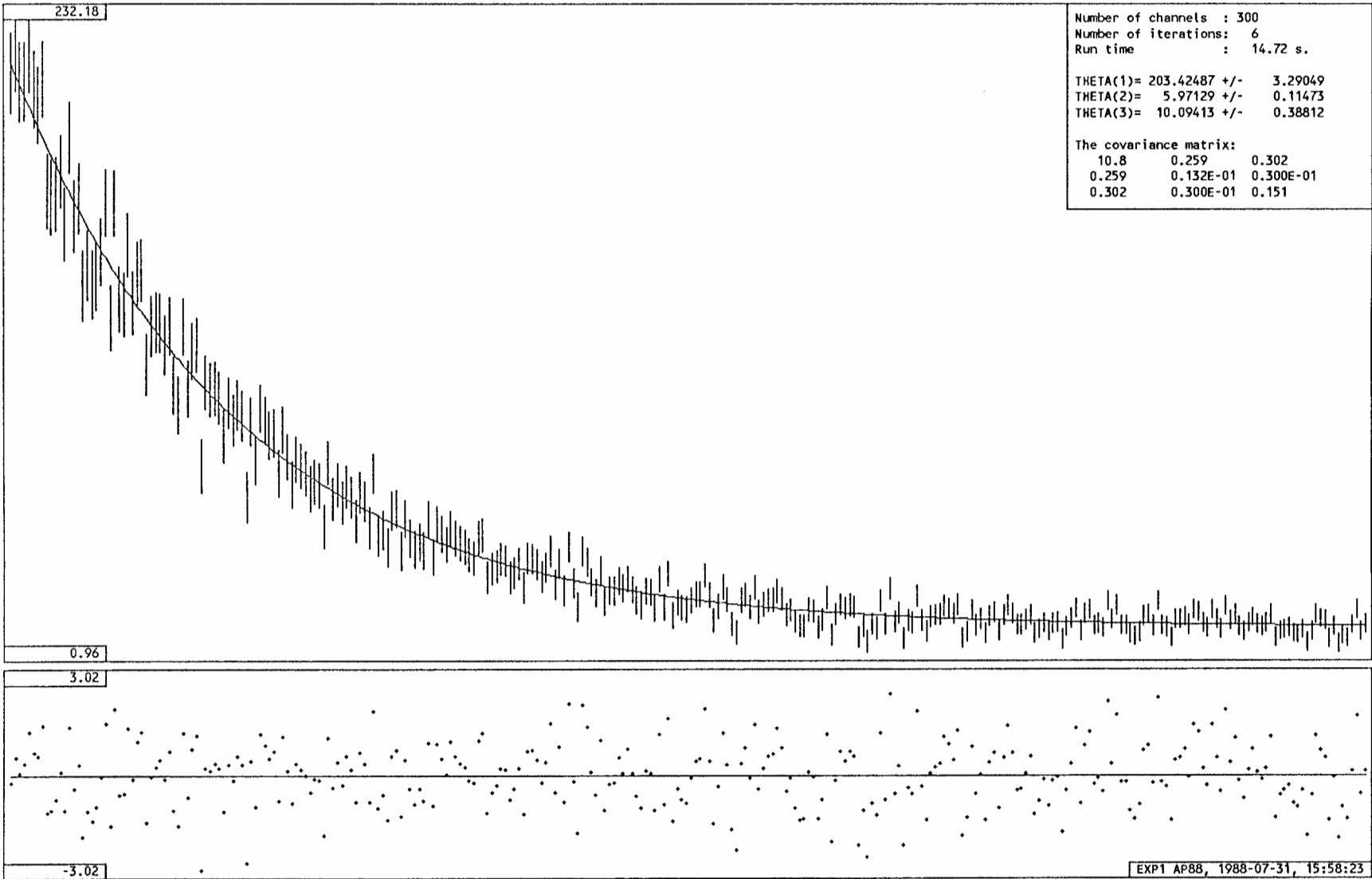
New calculation? (Y/*):N



Number of channels : 300
Number of iterations: 4
Run time : 7.47 s.
THETA(1)= 198.23155 +/- 3.16312
THETA(2)= 5.95290 +/- 0.08606
THETA(3)= 10.00000 Locked.
The covariance matrix:
10.0 0.200 0.000E+00
0.200 0.741E-02 0.000E+00
0.000E+00 0.000E+00 2.00



EXP1 AP88, 1988-07-31, 16:01:17



Kommentarer till programlistningen:

Rad	Kommentar
1-18	Deklarationer.
20-40	Läs in antal kanaler, parametervärden och om felgränser skall ritas.
42-48	Tillverka testkurvan $y_i = N(f(\underline{\theta}; x_i), \sqrt{f(\underline{\theta}; x_i)})$. Egentligen borde kanalinhållet vara Poissonfördelat (kan diskuteras se [7] sid. 181) men om kanalinhållet är tillräckligt stort är normalfördelningen en god approximation.
50-52	Från start är alla parametrar fria variabler i anpassningen.
54-110	Om man väljer att bestämma startvärden automatiskt enl. tidigare beskrivning utförs detta annars läses startvärden in från tangentbordet och möjlighet ges att låsa parametervärden.
112-114	Om Q eller q tangenten på tangentbordet trycks ned avbryts itereringen.
116-117	Tidtagning.
119-120	Skriv ut startvärden på skärmen.
122	Itereringen börjar med att använda den lineariserade G-matrisen eftersom den tillåter sämre startvärden utan att divergera.
123-198	Utför iterativa förbättringar med Newtons metod.
124-125	Nollställ g-vektorn och G-matrisen.
127-131	Om G-matrisen är lineariserad används vikten 1/2 annars används $1/\sigma^2 = 1/f(\underline{\theta}; x)$ jfr. (5).
132-150	g-vektorn och G-matrisen beräknas enl. formlerna (37),(42) och (46).
152-156	Om någon parameter är låst är motsvarande rad och kolumn i G-matrisen nollställd. Därför måste diagonalelementet sättas till 1 för att parameterns värde ej skall ändras vid lösningen av ekvationssystemet (dessutom blir G-matrisen singular om en rad eller kolumn är noll).
158-164	Invertera G-matrisen och multiplicera den med -1.
165	Utför beräkningen $-G \cdot \underline{g}$ enl. ekvation (43) och placera resultatet i vektorn DELTATHETA.
166	Addera DELTATHETA till THETAEST vilket ger den nya parametervektorn enl. ekv. (39).

- 168-173 Beräkna ett mått på hur nära minpunkten iterationerna har nått. SLASK-variabeln tilldelas värdet på den senaste steglängden i parameterrymden (normerat). Om denna steglängd är tillräckligt liten kan programmet tillåtas byta till den fullständiga G-matrisen istället för den lineariserade, alternativt kan iterationen avbrytas om G-matrisen redan beräknas med (42).
- 175-177 Skriv ut resultat från den senaste iterationen på skärmen.
- 179-182 Om steglängden är mindre än $0.5 \cdot 10^{-4}$ och den fullständiga G-matrisen använts är iterationen klar och slingan bryts.
- 183-185 Om steglängden är mindre än $0.5 \cdot 10^{-2}$ byter programmet från den lineariserade G-matrisen till den fullständiga för att felgränser skall kunna beräknas.
- 187-194 Här testas om knappen Q på tangentbordet varit nedtryckt, om så är fallet bryts itereringen.
- 196-198 Slut på iterationslingen. Om programmet hamnar på rad 197 har maximalt antal iterationer utförts utan att minpunkten uppnåtts.
- 200 Efter slingan ligger matrisen $-G^{-1}$ i fältet GMATRIX. Denna måste multipliceras med -2 för att vara en skattning av kovariansmatrisen (45).
- 201-202 Spara exekveringstiden.
- 203-208 Stanna programmet innan kurvan ritas på skärmen.
- 210 Rita anpassning och residualvektor på skärmen.
- 212-217 Rita anpassning och residualvektor på Roland pen-plotter.
- 219-283 Rita anpassning och residualvektor på HP laserskrivare samt skriv ut diverse värden bl.a. kovariansmatrisen.
- 284-309 Skriv ut parametervärden med felgränser (eller ordet 'Locked') och kovariansmatrisen mm. på skärmen.
- 311-319 Fråga om ny beräkning skall utföras.
- 321-331 Rutinen beräknar funktionsvärdet $f(\underline{\theta}; x)$, (56).
- 333-349 Rutinen beräknar förstaderivatorna av f , (57)..(59).
- 351-367 Rutinen beräknar andraderivatorna av f , (60)..(62).

```

1 PROGRAM EXP1
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5 INTEGER N,L,MAXITER
6 PARAMETER (N=1024, L=3, MAXITER=1000)
7 REAL Y(1:N),X(1:N),DX,DET,NORMAL,THETAEST(1:L)
8 REAL RESIDUE(1:N),SIGMA(1:N),F,THETA(1:L)
9 REAL GMATRIX(1:L,1:L),GVECTOR(1:L),DELTATHETA(1:L)
10 REAL BG,SX,SY,SXY,SX2,NOOFPOINTS,SLASK,DFDTH,D2FDTH2
11 REAL SL1,SL2,SL3,SL4,SL5,TIME
12 REAL JX,JY,JXL,JYL,DJY
13 INTEGER R,K,I,NCH,NOOFIT,J
14 INTEGER*2 HO,MI,SE,HN,ANS,KBDCHK,KBDINC
15 INTEGER*2 YEAR,MONTH,DAY
16 CHARACTER SV*1,JETTXT*40
17 LOGICAL SING,ERRORBAR,LINEARIZED,LOCKED(1:L)
18 EXTERNAL F
19
20 10 WRITE(*,1000) 'No of channels (' ,L,'..',N,'):'
21 1000 FORMAT(' ',A,I2,A,I4,A)
22 READ(*,*,ERR=10) NCH
23 IF (NCH.LT.L.OR.NCH.GT.N) THEN
24 GOTO 10
25 ENDIF
26
27 WRITE(*,*) 'Plot errorbars? (Y/*):'
28 READ(*,1100) SV
29 1100 FORMAT(A)
30 ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'
31
32 WRITE(*,*) 'Parameter values.'
33 WRITE(*,*) 'Theta 1 (Intensity):'
34 READ(*,*) THETA(1)
35
36 WRITE(*,*) 'Theta 2 (Decay constant):'
37 READ(*,*) THETA(2)
38
39 WRITE(*,*) 'Theta 3 (Background):'
40 READ(*,*) THETA(3)
41
42 DX=1.0/(NCH-1)
43 DO 20 I=1,NCH
44 X(I)=(I-1)*DX
45 SLASK=F(THETA,L,X(I))
46 Y(I)=NORMAL(SLASK,SQRT(SLASK))
47 SIGMA(I)=SQRT(SLASK)
48 20 CONTINUE
49
50 DO 30 I=1,L
51 LOCKED(I)=.FALSE.
52 30 CONTINUE
53
54 WRITE(*,*) 'Calculate start values from data? (Y/*):'
55 READ(*,1100) SV
56 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
57 R =NCH/10+2

```

```

58 K =NCH-R+1
59 BG=0.0
60
61 DO 40 I=K,NCH
62 BG=BG+Y(I)
63 40 CONTINUE
64
65 BG =BG/R
66 SX =0.0
67 SY =0.0
68 SXY =0.0
69 SX2 =0.0
70 NOOFPOINTS=0.0
71
72 DO 50 I=1,NCH
73 IF (Y(I)-BG.LE.(Y(1)-BG)*0.1) THEN
74 GOTO 60
75 ENDIF
76 NOOFPOINTS=NOOFPOINTS+1.0
77 SX=SX+X(I)
78 SY=SY+LOG(Y(I))
79 SXY=SXY+X(I)*LOG(Y(I))
80 SX2=SX2+X(I)**2
81 50 CONTINUE
82 60 CONTINUE
83
84 THETAEST(3)=BG
85 THETAEST(2)=(NOOFPOINTS*SX2-SX**2)/(SX*SY-NOOFPOINTS*SXY)
86 THETAEST(1)=(SY*THETAEST(2)+SX)/(THETAEST(2)*NOOFPOINTS)
87 THETAEST(1)=EXP(THETAEST(1))
88 THETAEST(2)=1.0/THETAEST(2)
89 ELSE
90 WRITE(*,*) 'Enter start values for theta.'
91
92 WRITE(*,*) 'Theta(1)='
93 READ(*,*) THETAEST(1)
94 WRITE(*,*) 'Lock parameter Theta(1)? (Y/*):'
95 READ(*,1100) SV
96 LOCKED(1)=SV.EQ.'Y'.OR.SV.EQ.'y'
97
98 WRITE(*,*) 'Theta(2)='
99 READ(*,*) THETAEST(2)
100 WRITE(*,*) 'Lock parameter Theta(2)? (Y/*):'
101 READ(*,1100) SV
102 LOCKED(2)=SV.EQ.'Y'.OR.SV.EQ.'y'
103
104 WRITE(*,*) 'Theta(3)='
105 READ(*,*) THETAEST(3)
106 WRITE(*,*) 'Lock parameter Theta(3)? (Y/*):'
107 READ(*,1100) SV
108 LOCKED(3)=SV.EQ.'Y'.OR.SV.EQ.'y'
109
110 ENDIF
111
112 WRITE(*,*) ' '
113 WRITE(*,*) 'Press Q to quit iteration.'
114 WRITE(*,*) ' '
115
116 CALL GETTIM(HO,MI,SE,HN)
117 TIME=HO*3600.0+MI*60.0+SE+HN*0.01

```

```

118 WRITE(*,1200) 'It:',0,'',TH1:',THETAEST(1),'',TH2:',THETAEST(2),
119 f ',TH3:',THETAEST(3)
120
121
122 LINEARIZED=.TRUE.
123 DO 120 J=1,MAXITER
124 CALL ZEROMAT(GMATRIX,L,L,L,L)
125 CALL ZEROMAT(GVECTOR,L,1,L,1)
126 DO 90 I=1,NCH
127 IF (LINEARIZED) THEN
128 SL1=-1.0
129 ELSE
130 SL1=-2.0/F(THETAEST,L,X(I))
131 ENDIF
132 SL2=Y(1)-F(THETAEST,L,X(I))
133 DO 80 R=1,L
134 IF (.NOT.LOCKED(R)) THEN
135 SL3=DFDTH(R,THETAEST,L,X(I))
136 GVECTOR(R)=GVECTOR(R)+SL1*SL2*SL3
137 DO 70 K=R,L
138 IF (.NOT.LOCKED(K)) THEN
139 SL4=DFDTH(K,THETAEST,L,X(I))
140 GMATRIX(R,K)=GMATRIX(R,K)+SL1*(-SL3*SL4)
141 IF (.NOT.LINEARIZED) THEN
142 SL5=D2FDTH2(R,K,THETAEST,L,X(I))
143 GMATRIX(R,K)=GMATRIX(R,K)+SL1*SL2*SL5
144 ENDIF
145 GMATRIX(K,R)=GMATRIX(R,K)
146 ENDIF
147 70 CONTINUE
148 ENDIF
149 80 CONTINUE
150 90 CONTINUE
151
152 DO 100 R=1,L
153 IF (LOCKED(R)) THEN
154 GMATRIX(R,R)=1.0
155 ENDIF
156 100 CONTINUE
157
158 CALL INVMAT(GMATRIX,L,DET,SING,L,L)
159 IF (SING) THEN
160 WRITE(*,*) 'G-matrix singular!'
161 GOTO 210
162 ENDIF
163
164 CALL MULTCMAT(GMATRIX,-1.0,L,L,L,L)
165 CALL MULTMAT(GMATRIX,GVECTOR,DELTATHETA,L,L,1,L,L,L,1,L,1)
166 CALL ADDMAT(THETAEST,DELTATHETA,THETAEST,L,1,L,1,L,1,L,1)
167
168 SLASK=0.0
169 DO 110 I=1,L
170 IF (THETAEST(I).NE.0.0) THEN
171 SLASK=SLASK+ABS(DELTATHETA(I)/THETAEST(I))
172 ENDIF
173 110 CONTINUE
174
175 WRITE(*,1200) 'It:',J,'',TH1:',THETAEST(1),'',TH2:',THETAEST(2),
176 f ',TH3:',THETAEST(3),'',STEP:',SLASK
177 1200 FORMAT(' ',A,13,A,G12.5,A,G12.5,A,G12.5,A,G13.6)

```

```

178
179 IF (SLASK.LE.0.5E-4.AND.(.NOT.LINEARIZED)) THEN
180 NOOFIT=J
181 GOTO 130
182 ENDIF
183 IF (SLASK.LE.0.5E-2.AND.LINEARIZED) THEN
184 LINEARIZED=.FALSE.
185 ENDIF
186
187 ANS=KBDCHK()
188 IF (ANS.NE.0) THEN
189 ANS=KBDINC()
190 IF (CHAR(ANS).EQ.'Q'.OR.CHAR(ANS).EQ.'q') THEN
191 NOOFIT=J
192 GOTO 130
193 ENDIF
194 ENDIF
195
196 120 CONTINUE
197 NOOFIT=MAXITER
198 130 CONTINUE
199
200 CALL MULTCMAT(GMATRIX,-2.0,L,L,L,L)
201 CALL GETTIM(HO,MI,SE,HN)
202 TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
203 WRITE(*,*) 'Press any key to continue.'
204 140 ANS=KBDCHK()
205 IF (ANS.EQ.0) THEN
206 GOTO 140
207 ENDIF
208 ANS=KBDINC()
209
210 CALL SCRXPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,L,ERRORBAR)
211
212 150 WRITE(*,*) 'Plot on Roland plotter? (Y/*):'
213 READ(*,1100) SV
214 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
215 CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,L,ERRORBAR)
216 GOTO 150
217 ENDIF
218
219 160 WRITE(*,*) 'Plot on HP-Laserjet? (Y/*): '
220 READ(*,1100) SV
221 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
222 CALL HPSTART('L')
223 CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,L,ERRORBAR)
224 JX=870
225 JXL=1119-JX
226 JYL=(12+1)*13
227 JY=719-JYL
228 CALL HPBOX(JX,JY,JXL,JYL)
229 JX=870
230 JY=719-16
231 DJY=-13
232
233 WRITE(JETTXT,1400) 'Number of channels :',NCH
234 CALL HPPRINT(JX,JY,40,JETTXT)
235 JY=JY+DJY
236
237 WRITE(JETTXT,1400) 'Number of iterations:',NOOFIT

```

```

238 CALL HPPRINT(JX,JY,40,JETTXT)
239 JY=JY+DJY
240
241 WRITE(JETTXT,1500) 'Run time           :',TIME,' s.'
242 CALL HPPRINT(JX,JY,40,JETTXT)
243 JY=JY+DJY
244
245 WRITE(JETTXT,1100) ' '
246 CALL HPPRINT(JX,JY,40,JETTXT)
247 JY=JY+DJY
248
249 DO 170 R=1,L
250   IF (.NOT.LOCKED(R)) THEN
251     WRITE(JETTXT,1600) 'THETA(',R,')=',THETAEST(R),' +/- ',
252     f      SQRT(ABS(GMATRIX(R,R)))
253   ELSE
254     WRITE(JETTXT,1600) 'THETA(',R,')=',THETAEST(R),' Locked.'
255   ENDIF
256   CALL HPPRINT(JX,JY,40,JETTXT)
257   JY=JY+DJY
258 170 CONTINUE
259
260 WRITE(JETTXT,1100) ' '
261 CALL HPPRINT(JX,JY,40,JETTXT)
262 JY=JY+DJY
263
264 WRITE(JETTXT,1100) ' The covariance matrix:'
265 CALL HPPRINT(JX,JY,40,JETTXT)
266 JY=JY+DJY
267
268 DO 180 K=1,L
269   WRITE(JETTXT,1800) (GMATRIX(R,K), R=1,L)
270   CALL HPPRINT(JX,JY,40,JETTXT)
271   JY=JY+DJY
272 180 CONTINUE
273
274 CALL GETTIM(HO,MI,SE,HN)
275 CALL GETDAT(YEAR,MONTH,DAY)
276 WRITE(JETTXT,2000)
277 f 'EXP1 AP88, ',YEAR,'-',MONTH,'-',DAY,', ',HO,':',MI,':',SE
278 2000 f FORMAT(A,14.4,5(A,12.2))
279 CALL HPTBOX(921.0,0.0,31,JETTXT)
280
281 CALL HPEJECT
282 GOTO 160
283 ENDIF
284
285 WRITE(*,1300)
286 1300 FORMAT(/)
287 WRITE(*,1400) 'Number of channels:',NCH
288 WRITE(*,1400) 'Number of iterations:',NOOFIT
289 1400 FORMAT(' ',A,I4)
290 WRITE(*,1500) 'Run time:',TIME,' s.'
291 1500 FORMAT(' ',A,F8.2,A)
292 WRITE(*,1300)
293
294 DO 190 R=1,L
295   IF (.NOT.LOCKED(R)) THEN
296     WRITE(*,1600) 'THETA(',R,')=',THETAEST(R),' +/- ',
297     f      SQRT(ABS(GMATRIX(R,R)))

```

```

298   ELSE
299     WRITE(*,1600) 'THETA(',R,')=',THETAEST(R),' Locked.'
300   ENDIF
301 1600 FORMAT(' ',A,I1,A,F10.5,A,F10.5)
302 190 CONTINUE
303
304 WRITE(*,1700) 'The covariance matrix for theta:'
305 1700 FORMAT(' ',A)
306 DO 200 K=1,L
307   WRITE(*,1800) (GMATRIX(R,K), R=1,L)
308 1800 FORMAT(' ',10G11.3)
309 200 CONTINUE
310
311 WRITE(*,1900) 'New calculation? (Y/*):'
312 1900 FORMAT(' ',A)
313 READ(*,1100) SV
314 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
315   GOTO 10
316 ELSE
317   STOP
318 ENDIF
319 END
320
321 REAL FUNCTION F(THETA,L,X)
322 C Anders Persson, 1988
323
324 IMPLICIT LOGICAL (A-Z)
325 INTEGER L
326 REAL THETA(1:L),X
327
328 F=THETA(1)*EXP(-THETA(2)*X)+THETA(3)
329
330 RETURN
331 END
332
333 REAL FUNCTION DFDTH(J,THETA,L,X)
334 C Anders Persson, 1988
335
336 IMPLICIT LOGICAL (A-Z)
337 INTEGER J,L
338 REAL THETA(1:L),X
339
340 IF (J.EQ.1) THEN
341   DFDTH=EXP(-THETA(2)*X)
342 ELSEIF (J.EQ.2) THEN
343   DFDTH=-X*THETA(1)*EXP(-THETA(2)*X)
344 ELSE
345   DFDTH=1.0
346 ENDIF
347
348 RETURN
349 END
350
351 REAL FUNCTION D2FDTH2(I,J,THETA,L,X)
352 C Anders Persson, 1988
353
354 IMPLICIT LOGICAL (A-Z)
355 INTEGER I,J,L
356 REAL THETA(1:L),X
357

```



```
358 IF (I+J.EQ.3) THEN
359   D2FDTH2=-X*EXP(-THETA(2)*X)
360 ELSEIF (I.EQ.2.AND.J.EQ.2) THEN
361   D2FDTH2=X*X*THETA(1)*EXP(-THETA(2)*X)
362 ELSE
363   D2FDTH2=0.0
364 ENDIF
365 RETURN
366 END
367
```

EXPN

Programmet anpassar till en summa av exponentialer med bakgrund. I stort sett utför programmet samma beräkningar som EXP1-programmet för en exponential, men man har här möjlighet att låsa vissa parametrar i en körning för att sedan släppa dem fria i en förnyad iteration på samma data. Programmet medger även att den lineariserade G-matrisen används hela tiden för att öka exekveringshastigheten på bekostnad av förlorad information om felgränser för parameterskattningarna.

Anpassad funktion:

$$f(\underline{\theta}; x) = \sum_{i=1}^L \theta_i \cdot \exp(-\theta_{L+i} \cdot x) + \theta_{2L+1} \quad (63)$$

Förstaderivator:

$$\frac{\partial f}{\partial \theta_j} (j=1..L) = \exp(-\theta_{L+j} \cdot x) \quad (64)$$

$$\frac{\partial f}{\partial \theta_j} (j=L+1..2L) = -x \cdot \theta_{j-L} \cdot \exp(-\theta_j \cdot x) \quad (65)$$

$$\frac{\partial f}{\partial \theta_j} (j=2L+1) = 1 \quad (66)$$

Andraderivator:

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left(\begin{matrix} j=1..L \\ k=L+j \end{matrix} \right) = -x \cdot \exp(-\theta_{L+j} \cdot x) \quad (67)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left(\begin{matrix} j=L+1..2L \\ k=j \end{matrix} \right) = x^2 \cdot \theta_{j-L} \cdot \exp(-\theta_j \cdot x) \quad (68)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} = \frac{\partial^2 f}{\partial \theta_k \partial \theta_j} \quad (69)$$

alla övriga andraderivator är noll.

Utskrift från testkörning av programmet EXPN:

```

No of decays (1.. 3)      :2
No of channels ( 5..1024) :400
Intensity, decay no 1     :1000
Decay rate, decay no 1   :20
Intensity, decay no 2     :300
Decay rate, decay no 2   :6
Background                 :50
Add noise to data? (Y/*) :Y
Plot errorbars? (Y/*)    :Y

```

Toggle Free/Locked with <shift>+Function key.

F1 : Intensity , decay no 1 1000.0000, Free
F2 : Intensity , decay no 2 300.0000, Free
F3 : Decay rate, decay no 1 20.0000, Free
F4 : Decay rate, decay no 2 6.0000, Free
F5 : Background 50.0000, Free
F8 : Preview.
F9 : Calculate.
F10: G-matrix = not linearized.

<F9>

Press Q to quit iteration.

It: 0
Intensity(1) : 1000.0 Decay rate(1): 20.000
Intensity(2) : 300.00 Decay rate(2): 6.0000
Background : 50.000

It: 1, STEP: 0.292837
Intensity(1) : 952.09 Decay rate(1): 20.710
Intensity(2) : 341.99 Decay rate(2): 6.5124
Background : 49.664

It: 2, STEP: 0.220971E-01
Intensity(1) : 948.66 Decay rate(1): 20.809
Intensity(2) : 345.87 Decay rate(2): 6.5192
Background : 49.591

It: 3, STEP: 0.527162E-03
Intensity(1) : 948.73 Decay rate(1): 20.808
Intensity(2) : 345.78 Decay rate(2): 6.5184
Background : 49.590

It: 4, STEP: 0.255050E-01
Intensity(1) : 948.72 Decay rate(1): 20.854
Intensity(2) : 346.47 Decay rate(2): 6.5150
Background : 50.643

It: 5, STEP: 0.886826E-03
Intensity(1) : 948.83 Decay rate(1): 20.851
Intensity(2) : 346.34 Decay rate(2): 6.5137
Background : 50.641

It: 6, STEP: 0.207379E-04
Intensity(1) : 948.83 Decay rate(1): 20.851
Intensity(2) : 346.34 Decay rate(2): 6.5138
Background : 50.641

It: 7, STEP: 0.412327E-06
Intensity(1) : 948.83 Decay rate(1): 20.851
Intensity(2) : 346.34 Decay rate(2): 6.5138
Background : 50.641

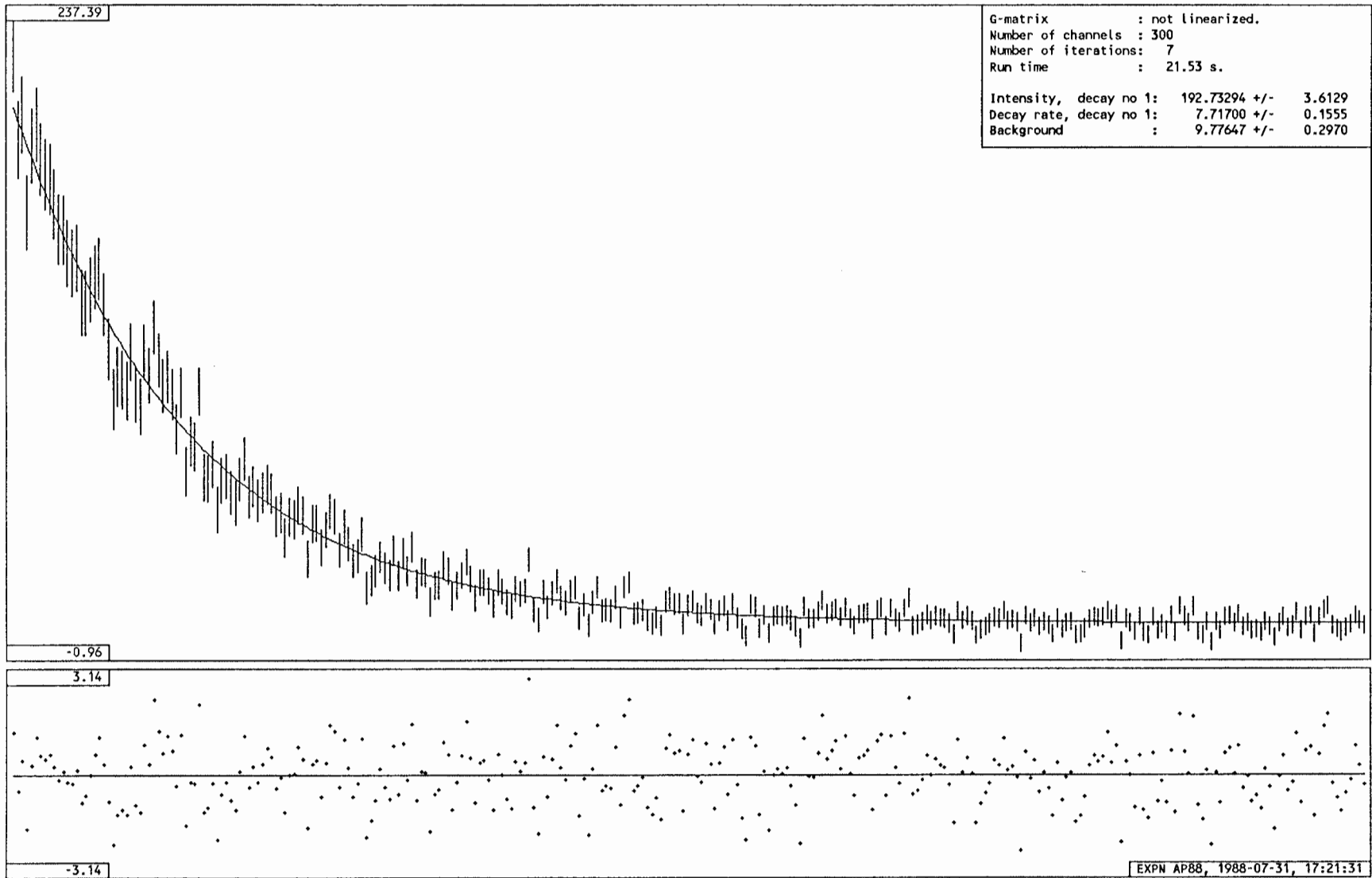
Press any key to continue. <Return>
Plot on Roland plotter? (Y/*) :N
Plot on HP-Laserjet? (Y/*) :Y

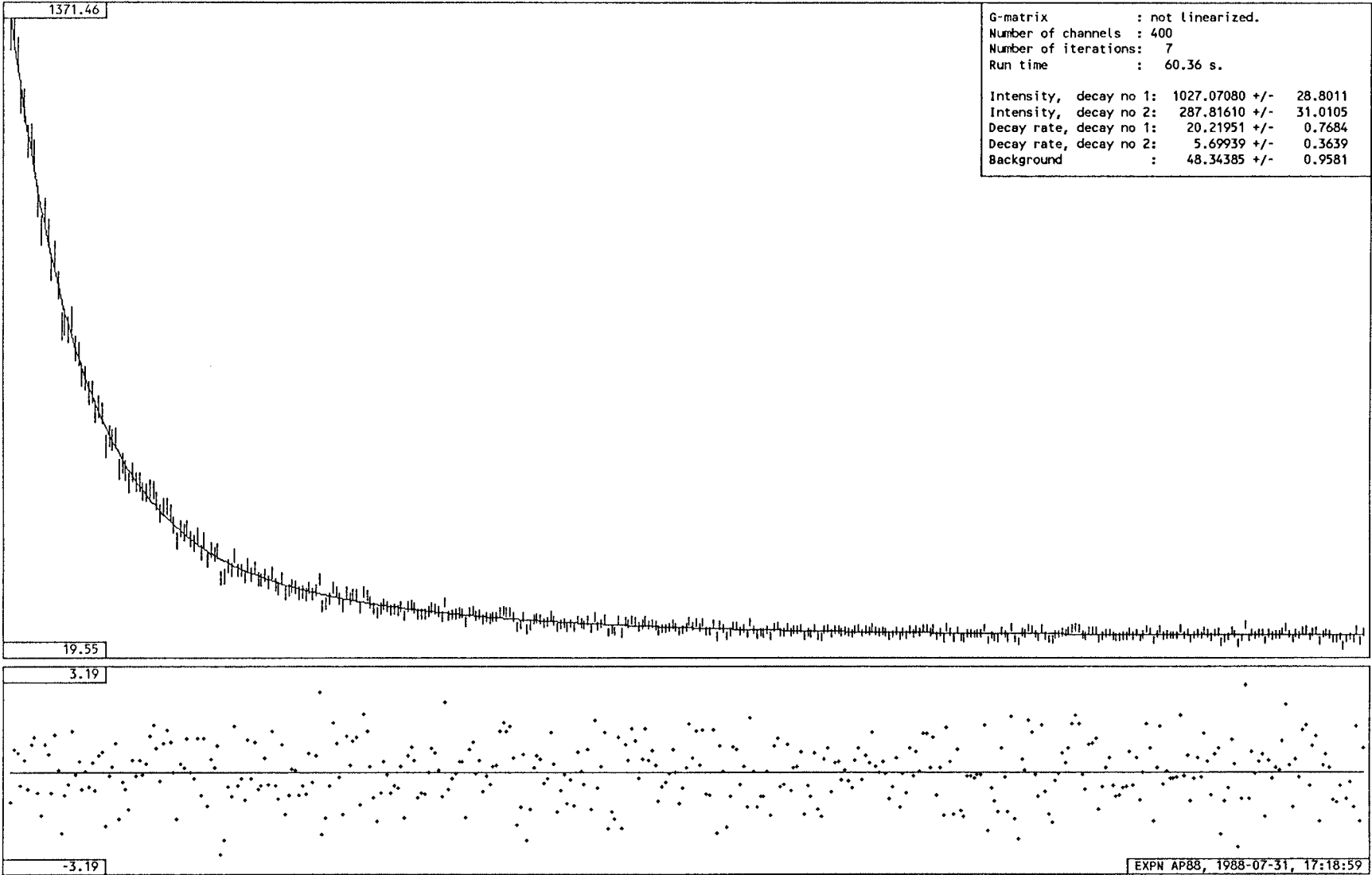
G-matrix : not linearized.
Number of points : 400
Number of iterations : 7
Run time : 55.31 s.

Intensity, decay no 1 : 948.82849 +/- 36.3149
Intensity, decay no 2 : 346.34073 +/- 39.3322
Decay rate, decay no 1 : 20.85087 +/- 0.9480
Decay rate, decay no 2 : 6.51378 +/- 0.3859
Background : 50.64080 +/- 0.8209

New iteration on the data? (Y/*):N

New calculation? (Y/*):N





Kommentarer till programlistningen:

Rad	Kommentar
1-20	Deklarationer.
22-62	Läs in data för konstruktion av testkurvan.
64-67	Från början är alla parametrar fria.
69-80	Beräkna testkurvan.
86	Programmet startar med att G-matrisen ej låses till den lineariserade varianten (46).
88-89	Sudda skärmen.
91-114	Skriv ut meny på skärmen.
116-120	Läs in vilken funktionstangent som tryckts in.
122-129	Om <shift>+funktionstangent trycks ned ändras status på motsvarande parameter från 'free' till 'locked' eller tillbaka.
130-133	Om Preview väljs ritas kurva och anpassning på skärmen för att man skall kunna avgöra om något startvärde behöver ändras. (Programmet är dock ganska snällt och tillåter relativt dåliga startvärden utan att divergera, speciellt för en exponential. Om anpassning sker till fler exponentieller blir det kinkigare med startvärden.)
134-136	Om F9 trycks ned - starta itereringen.
137-139	Om F10 trycks ned ändras valet av G-matris från (42) till (46) och tillbaka. Ger möjlighet att snabba upp exekveringen genom att använda den lineariserade G-matrisen. Detta kan ibland vara enda möjligheten att få programmet att konvergera på verkligt usla indata.
140-164	Övriga funktionstangenter ger möjlighet att ändra startvärdet på parametrarna.
166-170	Skriv ut meddelande om att itereringen avbryts om Q tangenten trycks ned.
172-177	Ändringen i parametervärden maximeras. Intensitet och avklingningskonstant tillåts variera från startvärdet/5 till startvärdet*5. Bakgrunden får variera ± 5 *startvärdet.
179-180	Ta tiden.
182-190	Skriv ut nollte iterationens parametervärden.
192-193	Från start är G-matrisen lineariserad och vikten låst till $1/y_i$.
195-222	Beräkna g-vektorn och G-matrisen enl. formlerna (37), (42), (46).
199-203	Om vikten är låst används $1/y_i$ som vikt annars används $1/f_i$.

- 224-228 Om någon parameter är låst blir elementet i motsvarande rad och kolumn i G-matrisen noll. För att göra ekvationssystemet lösbart och inte ändra den låsta parametern måste diagonalelementet sättas till ett.
- 230-234 Invertera G-matrisen.
- 236-237 Multiplicera G^{-1} -matrisen med -1 och beräkna $\Delta\theta$ -vektorn enl. (43).
- 239-245 Om någon parameter hamnar utanför tillåtna gränser ändras motsvarande $\Delta\theta$ -värde, parameterens värde hamnar på min eller maxgränsen.
- 247 Beräkna den nya parametervektorn enl. (39).
- 249-254 Beräkna steglängden, $(\sum |\Delta\theta_i / \theta_i|)$, ett mått på hur nära minpunkten programmet nått.
- 256-261 Skriv ut de nya parametervärdena på skärmen.
- 263-266 Om steglängden är mindre än $0.5 \cdot 10^{-4}$ och G-matrisen ej är lineariserad är iterationen klar och slingan bryts.
- 267-269 Om steget är mindre än $0.5 \cdot 10^{-4}$ och G-matrisen är lineariserad byter programmet till den fullständiga G-matrisen (42).
- 270-272 Om programmet är låst till att använda den lineariserade G-matrisen sätts LINEARIZED=on.
- 273-276 Om den lineariserade G-matrisen skall användas och steglängden är mindre än $0.5 \cdot 10^{-4}$ är iterationen klar och slingan bryts.
- 277-279 Om steglängden är mindre än $0.5 \cdot 10^{-2}$ används vikten $1/f_i$ istället för $1/y_i$.
- 281-288 Hoppa ur slingan om knappen Q tryckts ned på tangentbordet.
- 289-291 Slut på iterations-slingan. Om programmet hamnar på rad 290 har maximalt antal iterationer utförts utan att minpunkten uppnåtts.
- 293 I iterations-slingan multipliceras G^{-1} -matrisen med -1 inför beräkningen av $\Delta\theta$ (43), den måste nu multipliceras med -2 för att vara en skattning av kovariansmatrisen för θ (45).
- 294-295 Spara exekveringstiden.
- 296-301 Stanna programmet innan det ritar på skärmen.
- 303-304 Rita mätdata, anpassning och residualvektor på skärmen.
- 306-312 Rita mätdata, anpassning och residualvektor på Roland pen-plotter.
- 314-412 Rita mätdata, anpassning och residualvektor samt skriv ut diverse data på HP laserskrivare.

- 414-478 Skriv ut parametervärden med felgränser om den fullständiga G-matrisen använts, utan felgränser om den lineariserade G-matrisen använts och med noteringen 'Locked' om parametervärdet varit låst i anpassningen. Skriv även ut exekveringstid, antal kanaler, antal iterationer mm.
- 480-484 Här kan man välja att gå tillbaka för förnyad iteration på samma indata t.ex. om startvärden måste ändras eller om en låst parameter skall släppas fri.
- 486-494 Beräkning på nya data eller avsluta programmet.
- 496-509 Rutinen beräknar funktionsvärdet $f(\underline{\theta}; x)$, (63).
- 511-526 Rutinen beräknar förstaderivatorna av f , (64)..(66).
- 528-557 Rutinen beräknar andraderivatorna av f , (67)..(69).

```

1 PROGRAM EXPN
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5 INTEGER N,L,MAXITER,MAXNOOFDECAYS
6 PARAMETER (N=1024, MAXITER=1000, MAXNOOFDECAYS=3)
7 PARAMETER (L=2*MAXNOOFDECAYS+1)
8 REAL Y(1:N),X(1:N),DX,DET,NORMAL,THETAEST(1:L)
9 REAL RESIDUE(1:N),SIGMA(1:N),F,THETA(1:L)
10 REAL GMATRIX(1:L,1:L),GVECTOR(1:L),DELTATHETA(1:L)
11 REAL SLASK,DFDTH,D2FDTH2,THETAMIN(1:L),THETAMAX(1:L)
12 REAL SL1,SL2,SL3,SL4,SL5,TIME,STEP
13 REAL JX,JY,JXL,JYL,DJY
14 INTEGER R,K,I,NCH,NOOFIT,J,NOOFDECAYS,LR
15 INTEGER*2 HO,MI,SE,HN,ANS,KBDCHK,KBDINC
16 INTEGER*2 YEAR,MONTH,DAY
17 CHARACTER SV*1,LOCKEDORFREE(1:L)*6,JETTXT*50
18 LOGICAL SING,ERRORBAR,LINEARIZED,LOCKED(1:L),ADDNOISE
19 LOGICAL LOCKTOLINEARIZED,LOCKWEIGHT
20 EXTERNAL F
21
22 10 WRITE(*,1000) 'No of decays (1..' ,MAXNOOFDECAYS,'):'
23 1000 FORMAT(' ',A,I2,A)
24 READ(*,*,ERR=10) NOOFDECAYS
25 IF (NOOFDECAYS.LT.1.OR.NOOFDECAYS.GT.MAXNOOFDECAYS) THEN
26 GOTO 10
27 ENDIF
28 LR=2*NOOFDECAYS+1
29
30 20 WRITE(*,1100) 'No of channels (' ,LR,'..' ,N,'):'
31 1100 FORMAT(' ',A,I2,A,I4,A)
32 READ(*,*,ERR=20) NCH
33 IF (NCH.LT.L.OR.NCH.GT.N) THEN
34 GOTO 20
35 ENDIF
36
37 DO 50 I=1,NOOFDECAYS
38 30 WRITE(*,1200) 'Intensity, decay no ',I,': '
39 1200 FORMAT(' ',A,I1,A)
40 READ(*,*,ERR=30) THETA(I)
41 IF (THETA(I).LE.0.0) THEN
42 GOTO 30
43 ENDIF
44 40 WRITE(*,1200) 'Decay rate, decay no ',I,': '
45 READ(*,*,ERR=40) THETA(I+NOOFDECAYS)
46 IF (THETA(I+NOOFDECAYS).LE.0.0) THEN
47 GOTO 40
48 ENDIF
49 50 CONTINUE
50 60 WRITE(*,*) 'Background: '
51 READ(*,*,ERR=60) THETA(LR)
52
53 WRITE(*,*) 'Add noise to data? (Y/*): '
54 READ(*,1300) SV
55 1300 FORMAT(A)
56 ADDNOISE=SV.EQ.'Y'.OR.SV.EQ.'y'
57 ERRORBAR=ADDNOISE
    
```

```

58 IF (ADDNOISE) THEN
59 WRITE(*,*) 'Plot errorbars? (Y/*): '
60 READ(*,1300) SV
61 ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'
62 ENDIF
63
64 DO 70 I=1,LR
65 LOCKED(I)=.FALSE.
66 LOCKEDORFREE(I)='Free '
67 70 CONTINUE
68
69 DX=1.0/(NCH-1)
70 DO 80 I=1,NCH
71 X(I)=(I-1)*DX
72 SLASK=F(THETA,NOOFDECAYS,X(I))
73 IF (ADDNOISE) THEN
74 Y(I)=NORMAL(SLASK,SQRT(SLASK))
75 SIGMA(I)=SQRT(SLASK)
76 ELSE
77 Y(I)=SLASK
78 SIGMA(I)=0.0
79 ENDIF
80 80 CONTINUE
81
82 DO 90 I=1,LR
83 THETAEST(I)=THETA(I)
84 90 CONTINUE
85
86 LOCKTOLINEARIZED=.FALSE.
87
88 100 CALL GMODE
89 CALL TMODE
90
91 WRITE(*,1400) 'Toggle Free/Locked with <shift>+Function key.'
92 1400 FORMAT(//////,T10,A,/)
93 DO 110 I=1,NOOFDECAYS
94 WRITE(*,1500) I,I,THETAEST(I),LOCKEDORFREE(I)
95 1500 FORMAT(T10,'F',I1,' : Intensity , decay no ',
96 f I1,T45,F12.4,' ',',',T60,A)
97 110 CONTINUE
98 DO 120 I=1,NOOFDECAYS
99 WRITE(*,1600) I+NOOFDECAYS,I,THETAEST(I+NOOFDECAYS),
100 f LOCKEDORFREE(I+NOOFDECAYS)
101 1600 FORMAT(T10,'F',I1,' : Decay rate, decay no ',
102 f I1,T45,F12.4,' ',',',T60,A)
103 120 CONTINUE
104 WRITE(*,1700) LR,THETAEST(LR),LOCKEDORFREE(LR)
105 1700 FORMAT(T10,'F',I1,' : Background ',T45,F12.4,' ',',',T60,A)
106 WRITE(*,1800) 'F8 : Preview.'
107 1800 FORMAT(T10,A)
108 WRITE(*,1800) 'F9 : Calculate.'
109 IF (LOCKTOLINEARIZED) THEN
110 WRITE(*,1800) 'F10: G-matrix = linearized.'
111 ELSE
112 WRITE(*,1800) 'F10: G-matrix = not linearized.'
113 ENDIF
114 WRITE(*,1800) ' '
115
116 130 ANS=KBDCHK()
117 IF (ANS.NE.0) THEN
    
```

1-03-1

```

118     GOTO 130
119     ENDIF
120     ANS=KBDINC()
121
122     IF (ANS.GE.1084.AND.ANS.LE.1083+LR) THEN
123         I=ANS-1083
124         LOCKED(I)=.NOT.LOCKED(I)
125         LOCKEDORFREE(I)='Free '
126         IF (LOCKED(I)) THEN
127             LOCKEDORFREE(I)='Locked'
128         ENDIF
129         GOTO 100
130     ELSEIF (ANS.EQ.1066) THEN
131         CALL SCRIPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
132     f      THETAEST,NOOFDECAYS,ERRORBAR)
133         GOTO 100
134     ELSEIF (ANS.EQ.1067) THEN
135     C      Calculate.
136         GOTO 170
137     ELSEIF (ANS.EQ.1068) THEN
138         LOCKTOLINEARIZED=.NOT.LOCKTOLINEARIZED
139         GOTO 100
140     ELSEIF (ANS.GE.1059.AND.ANS.LE.1058+LR) THEN
141         I=ANS-1058
142         IF (I.LE.NOOFDECAYS) THEN
143     140      WRITE(*,1900) 'New intensity, decay no ',I,': '
144     1900      FORMAT(I10,A,I1,A)
145         READ(*,*,ERR=140) THETAEST(I)
146         IF (THETAEST(I).LE.0.0) THEN
147             GOTO 140
148         ENDIF
149         GOTO 100
150     ELSEIF (I.GT.NOOFDECAYS.AND.I.LT.LR) THEN
151     150      WRITE(*,1900) 'New decay rate, decay no ',I-NOOFDECAYS,': '
152         READ(*,*,ERR=150) THETAEST(I)
153         IF (THETAEST(I).LE.0.0) THEN
154             GOTO 150
155         ENDIF
156         GOTO 100
157     ELSEIF (I.EQ.LR) THEN
158     160      WRITE(*,1900) 'New background: '
159         READ(*,*,ERR=160) THETAEST(LR)
160         GOTO 100
161     ENDIF
162     GOTO 130
163     ENDIF
164     GOTO 130
165
166     C      *** Calculation ***
167     170      CONTINUE
168         WRITE(*,*) ' '
169         WRITE(*,*) 'Press Q to quit iteration.'
170         WRITE(*,*) ' '
171
172         DO 180 I=1,2*NOOFDECAYS
173             THETAMIN(I)=THETAEST(I)/5.0
174             THETAMAX(I)=THETAEST(I)*5.0
175     180      CONTINUE
176         THETAMAX(LR)= THETAEST(LR)*5.0
177         THETAMIN(LR)=-THETAMAX(LR)

```

```

178
179     CALL GETTIM(HO,MI,SE,HN)
180     TIME=HO*3600.0+MI*60.0+SE+HN*0.01
181
182     WRITE(*,2000) 'It:',0
183     2000    FORMAT(' ',49(' '),/,',',A,14,A,G13.6)
184     DO 190 R=1,NOOFDECAYS
185         WRITE(*,2100) 'Intensity(',R,'):',THETAEST(R),
186     f      'Decay rate(',R,'):',THETAEST(R+NOOFDECAYS)
187     2100    FORMAT(' ',A,I1,A,G12.5,A,I1,A,G12.5)
188     190      CONTINUE
189         WRITE(*,2200) 'Background:',THETAEST(LR)
190     2200    FORMAT(' ',A,G12.5)
191
192     LINEARIZED=.TRUE.
193     LOCKWEIGHT=.TRUE.
194
195     DO 270 J=1,MAXITER
196         CALL ZEROMAT(GMATRIX,LR,LR,L,L)
197         CALL ZEROMAT(GVECTOR,LR,1,L,1)
198         DO 220 I=1,NCH
199             IF (LOCKWEIGHT) THEN
200                 SL1=-2.0/MAX(Y(I),1.0)
201             ELSE
202                 SL1=-2.0/F(THETAEST,NOOFDECAYS,X(I))
203             ENDIF
204             SL2=Y(I)-F(THETAEST,NOOFDECAYS,X(I))
205             DO 210 R=1,LR
206                 IF (.NOT.LOCKED(R)) THEN
207                     SL3=DFDTH(R,THETAEST,NOOFDECAYS,X(I))
208                     GVECTOR(R)=GVECTOR(R)+SL1*SL2*SL3
209                 DO 200 K=R,LR
210                     IF (.NOT.LOCKED(K)) THEN
211                         SL4=DFDTH(K,THETAEST,NOOFDECAYS,X(I))
212                         GMATRIX(R,K)=GMATRIX(R,K)+SL1*(-SL3*SL4)
213                     IF (.NOT.LINEARIZED) THEN
214                         SL5=D2FDTH2(R,K,THETAEST,NOOFDECAYS,X(I))
215                         GMATRIX(R,K)=GMATRIX(R,K)+SL1*SL2*SL5
216                     ENDIF
217                     GMATRIX(K,R)=GMATRIX(R,K)
218                 ENDIF
219     200      CONTINUE
220             ENDIF
221     210      CONTINUE
222     220      CONTINUE
223
224     DO 230 R=1,LR
225         IF (LOCKED(R)) THEN
226             GMATRIX(R,R)=1.0
227         ENDIF
228     230      CONTINUE
229
230     CALL INVMAT(GMATRIX,LR,DET,SING,L,L)
231     IF (SING) THEN
232         WRITE(*,*) 'G-matrix singular.'
233         GOTO 360
234     ENDIF
235
236     CALL MULTCMAT(GMATRIX,-1.0,LR,LR,L,L)
237     CALL MULTMAT(GMATRIX,GVECTOR,DELTAETHETA,LR,LR,1,L,L,1,L,1)

```

```

238
239 DO 240 I=1,2*NOOFDECAYS+1
240 IF (THETAEST(I)+DELTATHETA(I).LT.THETAMIN(I)) THEN
241 DELTATHETA(I)=THETAMIN(I)-THETAEST(I)
242 ELSEIF (THETAEST(I)+DELTATHETA(I).GT.THETAMAX(I)) THEN
243 DELTATHETA(I)=THETAMAX(I)-THETAEST(I)
244 ENDIF
245 240 CONTINUE
246
247 CALL ADDMAT(THETAEST,DELTATHETA,THETAEST,LR,1,L,1,L,1,L,1)
248
249 STEP=0.0
250 DO 250 I=1,LR
251 IF (THETAEST(I).NE.0.0) THEN
252 STEP=STEP+ABS(DELTATHETA(I)/THETAEST(I))
253 ENDIF
254 250 CONTINUE
255
256 WRITE(*,2000) 'It:',J,', STEP:',STEP
257 DO 260 R=1,NOOFDECAYS
258 WRITE(*,2100) 'Intensity(',R,'):',THETAEST(R),
259 f 'Decay rate(',R,'):',THETAEST(R+NOOFDECAYS)
260 260 CONTINUE
261 WRITE(*,2200) 'Background :',THETAEST(LR)
262
263 IF (STEP.LE.0.5E-4.AND.(.NOT.LINEARIZED)) THEN
264 NOOFIT=J
265 GOTO 280
266 ENDIF
267 IF (STEP.LE.0.5E-4.AND.LINEARIZED) THEN
268 LINEARIZED=.FALSE.
269 ENDIF
270 IF (LOCKTOLINEARIZED) THEN
271 LINEARIZED=.TRUE.
272 ENDIF
273 IF (STEP.LE.0.5E-4.AND.LOCKTOLINEARIZED) THEN
274 NOOFIT=J
275 GOTO 280
276 ENDIF
277 IF (LOCKWEIGHT.AND.STEP.LE.0.5E-2) THEN
278 LOCKWEIGHT=.FALSE.
279 ENDIF
280
281 ANS=KBDCHK()
282 IF (ANS.NE.0) THEN
283 ANS=KBDINC()
284 IF (CHAR(ANS).EQ.'Q'.OR.CHAR(ANS).EQ.'q') THEN
285 NOOFIT=J
286 GOTO 280
287 ENDIF
288 ENDIF
289 270 CONTINUE
290 NOOFIT=MAXITER
291 280 CONTINUE
292
293 CALL MULTCMAT(GMATRIX,-2.0,L,L,L,L)
294 CALL GETTIM(HO,MI,SE,HN)
295 TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
296 WRITE(*,*) 'Press any key to continue.'
297 290 ANS=KBDCHK()

```

```

298 IF (ANS.EQ.0) THEN
299 GOTO 290
300 ENDIF
301 ANS=KBDINC()
302
303 CALL SCRXPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
304 f NOOFDECAYS,ERRORBAR)
305
306 300 WRITE(*,*) 'Plot on Roland plotter? (Y/*): '
307 READ(*,1300) SV
308 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
309 CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
310 f NOOFDECAYS,ERRORBAR)
311 GOTO 300
312 ENDIF
313
314 310 WRITE(*,*) 'Plot on HP-Laserjet? (Y/*): '
315 READ(*,1300) SV
316 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
317 CALL HPSTART('L')
318 CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
319 f NOOFDECAYS,ERRORBAR)
320
321 JX=800
322 JXL=1119-JX
323 JYL=(6+1+2*NOOFDECAYS)*13
324 JY=719-JYL
325 CALL HPBOX(JX,JY,JXL,JYL)
326 JX=800
327 JY=719-16
328 DJY=-13
329
330 IF (LOCKTOLINEARIZED) THEN
331 WRITE(JETTXT,2400) 'G-matrix : linearized.'
332 ELSE
333 WRITE(JETTXT,2400) 'G-matrix : not linearized.'
334 ENDIF
335 CALL HPPRINT(JX,JY,50,JETTXT)
336 JY=JY+DJY
337
338 WRITE(JETTXT,2400) 'Number of channels :',NCH
339 CALL HPPRINT(JX,JY,50,JETTXT)
340 JY=JY+DJY
341
342 WRITE(JETTXT,2400) 'Number of iterations:',NOOFIT
343 CALL HPPRINT(JX,JY,50,JETTXT)
344 JY=JY+DJY
345
346 WRITE(JETTXT,2500) 'Run time :',TIME,' s.'
347 CALL HPPRINT(JX,JY,50,JETTXT)
348 JY=JY+DJY
349
350 WRITE(JETTXT,1300)
351 CALL HPPRINT(JX,JY,50,JETTXT)
352 JY=JY+DJY
353
354 DO 320 I=1,NOOFDECAYS
355 IF ((.NOT.LOCKDEC(I)).AND.ADDNOISE.AND.
356 f (.NOT.LOCKTOLINEARIZED)) THEN
357 WRITE(JETTXT,2600) 'Intensity, decay no ',I,':',

```

```

358 f          THETAEST(I),' +/- ',SQRT(ABS(GMATRIX(I,I)))
359 ELSEIF (LOCKED(I)) THEN
360 WRITE(JETTXT,2600) 'Intensity, decay no ',I,':',
361 f          THETAEST(I),' Locked'
362 ELSEIF ((.NOT.LOCKED(I)).AND.
363 ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED)) THEN
364 f          WRITE(JETTXT,2600) 'Intensity, decay no ',I,':',
365 f          THETAEST(I)
366 ENDIF
367 CALL HPPRINT(JX,JY,50,JETTXT)
368 JY=JY+DJY
369 320 CONTINUE
370
371 DO 330 I=1,NOOFDECAYS
372 IF ((.NOT.LOCKED(I+NOOFDECAYS)).AND.ADDNOISE.AND.
373 (.NOT.LOCKTOLINEARIZED)) THEN
374 f          WRITE(JETTXT,2600) 'Decay rate, decay no ',I,':',
375 f          THETAEST(I+NOOFDECAYS),' +/- ',
376 f          SQRT(ABS(GMATRIX(I+NOOFDECAYS,I+NOOFDECAYS)))
377 ELSEIF (LOCKED(I+NOOFDECAYS)) THEN
378 f          WRITE(JETTXT,2600) 'Decay rate, decay no ',I,':',
379 f          THETAEST(I+NOOFDECAYS),' Locked'
380 ELSEIF ((.NOT.LOCKED(I+NOOFDECAYS)).AND.
381 ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED)) THEN
382 f          WRITE(JETTXT,2600) 'Decay rate, decay no ',I,':',
383 f          THETAEST(I+NOOFDECAYS)
384 ENDIF
385 CALL HPPRINT(JX,JY,50,JETTXT)
386 JY=JY+DJY
387 330 CONTINUE
388
389 IF ((.NOT.LOCKED(LR)).AND.ADDNOISE.AND.
390 (.NOT.LOCKTOLINEARIZED)) THEN
391 f          WRITE(JETTXT,2700) 'Background          :',
392 f          THETAEST(LR),' +/- ',SQRT(ABS(GMATRIX(LR,LR)))
393 ELSEIF (LOCKED(LR)) THEN
394 f          WRITE(JETTXT,2700) 'Background          :',
395 f          THETAEST(LR),' Locked'
396 ELSEIF ((.NOT.LOCKED(LR)).AND.
397 ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED)) THEN
398 f          WRITE(JETTXT,2700) 'Background          :',
399 f          THETAEST(LR)
400 ENDIF
401 CALL HPPRINT(JX,JY,50,JETTXT)
402
403 CALL GETTIM(HO,MI,SE,HN)
404 CALL GETDAT(YEAR,MONTH,DAY)
405 WRITE(JETTXT,2900)
406 f 'EXPN AP88, ',YEAR,'-',MONTH,'-',DAY,' ',HO,':',MI,':',SE
407 2900 FORMAT(A,14.4,5(A,12.2))
408 CALL HPTBOX(921.0,0.0,31,JETTXT)
409
410 CALL HPEJECT
411 GOTO 310
412 ENDIF
413
414 WRITE(*,2300)
415 2300 FORMAT(/)
416
417 IF (LOCKTOLINEARIZED) THEN

```

```

418 WRITE(*,2400) 'G-matrix          : linearized.'
419 ELSE
420 WRITE(*,2400) 'G-matrix          : not linearized.'
421 ENDIF
422
423 WRITE(*,2400) 'Number of points   :',NCH
424
425 WRITE(*,2400) 'Number of iterations :',NOOFIT
426 2400 FORMAT(' ',A,14)
427
428 WRITE(*,2500) 'Run time:',TIME,' s.'
429 2500 FORMAT(' ',A,F8.2,A)
430 WRITE(*,2300)
431
432 DO 340 I=1,NOOFDECAYS
433 IF ((.NOT.LOCKED(I)).AND.ADDNOISE.AND.
434 (.NOT.LOCKTOLINEARIZED)) THEN
435 f          WRITE(*,2600) 'Intensity, decay no ',I,':',
436 f          THETAEST(I),' +/- ',SQRT(ABS(GMATRIX(I,I)))
437 2600 FORMAT(' ',A,I1,A,F12.5,A,F9.4)
438 ELSEIF (LOCKED(I)) THEN
439 f          WRITE(*,2600) 'Intensity, decay no ',I,':',
440 f          THETAEST(I),' Locked'
441 ELSEIF ((.NOT.LOCKED(I)).AND.
442 ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED)) THEN
443 f          WRITE(*,2600) 'Intensity, decay no ',I,':',
444 f          THETAEST(I)
445 ENDIF
446 340 CONTINUE
447
448 DO 350 I=1,NOOFDECAYS
449 IF ((.NOT.LOCKED(I+NOOFDECAYS)).AND.ADDNOISE.AND.
450 (.NOT.LOCKTOLINEARIZED)) THEN
451 f          WRITE(*,2600) 'Decay rate, decay no ',I,':',
452 f          THETAEST(I+NOOFDECAYS),' +/- ',
453 f          SQRT(ABS(GMATRIX(I+NOOFDECAYS,I+NOOFDECAYS)))
454 ELSEIF (LOCKED(I+NOOFDECAYS)) THEN
455 f          WRITE(*,2600) 'Decay rate, decay no ',I,':',
456 f          THETAEST(I+NOOFDECAYS),' Locked'
457 ELSEIF ((.NOT.LOCKED(I+NOOFDECAYS)).AND.
458 ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED)) THEN
459 f          WRITE(*,2600) 'Decay rate, decay no ',I,':',
460 f          THETAEST(I+NOOFDECAYS)
461 ENDIF
462 350 CONTINUE
463
464 IF ((.NOT.LOCKED(LR)).AND.ADDNOISE.AND.
465 (.NOT.LOCKTOLINEARIZED)) THEN
466 f          WRITE(*,2700) 'Background          :',
467 f          THETAEST(LR),' +/- ',SQRT(ABS(GMATRIX(LR,LR)))
468 2700 FORMAT(' ',A,F12.5,A,F9.4)
469 ELSEIF (LOCKED(LR)) THEN
470 f          WRITE(*,2700) 'Background          :',
471 f          THETAEST(LR),' Locked'
472 ELSEIF ((.NOT.LOCKED(LR)).AND.
473 ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED)) THEN
474 f          WRITE(*,2700) 'Background          :',
475 f          THETAEST(LR)
476 ENDIF
477

```

```

478 WRITE(*,2300)
479
480 360 WRITE(*,*) 'New iteration on the data? (Y/*):'
481 READ(*,1300) SV
482 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
483     GOTO 100
484 ENDIF
485
486 WRITE(*,2800) 'New calculation? (Y/*):'
487 2800 FORMAT(/' ',A)
488 READ(*,1300) SV
489 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
490     GOTO 10
491 ELSE
492     STOP
493 ENDIF
494 END
495
496 REAL FUNCTION F(THETA,L,X)
497 C Anders Persson, 1988
498
499 IMPLICIT LOGICAL (A-Z)
500 INTEGER L,I
501 REAL THETA(1:*),X,SLASK
502
503 SLASK=THETA(2*L+1)
504 DO 10 I=1,L
505     SLASK=SLASK+THETA(I)*EXP(-MIN(THETA(L+I)*X,80.0))
506 10 CONTINUE
507 F=SLASK
508 RETURN
509 END
510
511 REAL FUNCTION DFDTH(J,THETA,L,X)
512 C Anders Persson, 1988
513
514 IMPLICIT LOGICAL (A-Z)
515 INTEGER J,L
516 REAL THETA(1:*),X
517
518 IF (J.LE.L) THEN
519     DFDTH=EXP(-MIN(THETA(L+J)*X,80.0))
520 ELSEIF (J.GT.L.AND.J.LE.2*L) THEN
521     DFDTH=-X*THETA(J-L)*EXP(-MIN(THETA(J)*X,80.0))
522 ELSE
523     DFDTH=1.0
524 ENDIF
525 RETURN
526 END
527
528 REAL FUNCTION D2FDTH2(J,K,THETA,L,X)
529 C Anders Persson, 1988
530
531 IMPLICIT LOGICAL (A-Z)
532 INTEGER I,J,K,L
533 REAL THETA(1:*),X
534 LOGICAL JGTK
535
536 JGTK=J.GT.K
537 IF (JGTK) THEN

```

```

538     I=J
539     J=K
540     K=I
541 ENDIF
542 D2FDTH2=0.0
543
544 IF (J.LE.L.AND.K.EQ.L+J) THEN
545     D2FDTH2=-X*EXP(-MIN(THETA(K)*X,80.0))
546 ELSEIF ((J.GT.L.AND.J.LE.2*L).AND.(K.EQ.J)) THEN
547     D2FDTH2=X*X*THETA(J-L)*EXP(-MIN(THETA(J)*X,80.0))
548 ENDIF
549
550 IF (JGTK) THEN
551     I=J
552     J=K
553     K=I
554 ENDIF
555
556 RETURN
557 END

```

GAUSSN

Programmet anpassar en summa av Gauss-toppar med bakgrund till indata. Programmet liknar EXP1 och EXPN. Dock kan parametervärden ej läsas och det går inte att välja att alltid använda den lineariserade G-matrisen. Programmet anpassar och ger felgränser för intensitet och position för de olika topparna. Halvvärdesbredden anpassas och förutsätts lika för alla toppar. Bakgrunden anpassas men ges utan felgränser. Standardavvikelsen för mätdata förutsätts lika för alla punkter och ges i absoluta enheter. Bruset i testkurvorna är approximativt normalfördelat.

Anpassad funktion:

$$f(\underline{\theta}; x) = \sum_{i=1}^L \theta_i \cdot \exp(-\theta_{2L+1} \cdot (x - \theta_{L+i})^2) + \theta_{2L+2} \quad (70)$$

$$\text{Position, topp nr. } i: \quad \theta_{L+i} \quad (71)$$

$$\text{Intensitet, topp nr. } i: \quad \theta_i \quad (72)$$

$$\text{Halvvärdesbredd (alla): } 2 \cdot \sqrt{\frac{\ln 2}{\theta_{2L+1}}} \quad (73)$$

Förstaderivator:

$$\frac{\partial f}{\partial \theta_j} (j=1..L) = \exp(-\theta_{2L+1} \cdot (x - \theta_{L+j})^2) \quad (74)$$

$$\frac{\partial f}{\partial \theta_j} (j=L+1..2L) = 2 \cdot \theta_{j-L} \cdot \theta_{2L+1} (x - \theta_j) \cdot \exp(-\theta_{2L+1} \cdot (x - \theta_j)^2) \quad (75)$$

$$\frac{\partial f}{\partial \theta_j} (j=2L+1) = \sum_{i=1}^L -\theta_i (x - \theta_{L+i})^2 \cdot \exp(-\theta_{2L+1} \cdot (x - \theta_{L+i})^2) \quad (76)$$

$$\frac{\partial f}{\partial \theta_j} (j=2L+2) = 1 \quad (77)$$

Andraderivator:

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left(\begin{matrix} j=1..L \\ k=L+j \end{matrix} \right) = 2 \cdot \theta_{2L+1} (x - \theta_{L+j}) \cdot \exp(-\theta_{2L+1} \cdot (x - \theta_{L+j})^2) \quad (78)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left(\begin{matrix} j=1..L \\ k=2L+1 \end{matrix} \right) = -(x - \theta_{L+j})^2 \cdot \exp(-\theta_{2L+1} \cdot (x - \theta_{L+j})^2) \quad (79)$$

$$\begin{aligned} \frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left(\begin{matrix} j=L+1..2L \\ k=j \end{matrix} \right) &= -2\theta_{j-L} \cdot \theta_{2L+1} \cdot \exp(-\theta_{2L+1} \cdot (x - \theta_j)^2) + \\ &+ 4\theta_{j-L}^2 \cdot \theta_{2L+1} \cdot (x - \theta_j)^2 \cdot \exp(-\theta_{2L+1} \cdot (x - \theta_j)^2) \end{aligned} \quad (80)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left(\begin{matrix} j=L+1..2L \\ k=2L+1 \end{matrix} \right) = 2\theta_{j-L} \cdot (x-\theta_j) \cdot \exp(-\theta_{2L+1} \cdot (x-\theta_j)^2) -$$

$$-2\theta_{j-L} \cdot \theta_{2L+1} \cdot (x-\theta_j)^3 \cdot \exp(-\theta_{2L+1} \cdot (x-\theta_j)^2) \quad (81)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left(\begin{matrix} j=2L+1 \\ k=2L+1 \end{matrix} \right) = \sum_{i=1}^L \theta_i (x-\theta_{L+i})^4 \cdot \exp(-\theta_{2L+1} \cdot (x-\theta_{L+i})^2) \quad (82)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} = \frac{\partial^2 f}{\partial \theta_k \partial \theta_j} \quad (83)$$

alla övriga andraderivator är noll.

Utskrift från testkörning av programmet GAUSSN:

```

No of Gauss-peaks (1.. 5)      :4
No of channels (10..1000)     :300
Position, peak no  1 (0..1000) :200
Intensity, peak no  1          :25
Position, peak no  2 (0..1000) :300
Intensity, peak no  2          :100
Position, peak no  3 (0..1000) :550
Intensity, peak no  3          :150
Position, peak no  4 (0..1000) :800
Intensity, peak no  4          :25
FWHM                      :100
Background                 :100
Sigma(Y)                   :5
Plot errorbars? (Y/*)      :Y

```

Enter start values for intensity,
position, FWHM and background.

```

-----
Position, peak no  1 (0..1000) :200
Intensity, peak no  1          :25
Position, peak no  2 (0..1000) :300
Intensity, peak no  2          :100
Position, peak no  3 (0..1000) :550
Intensity, peak no  3          :150
Position, peak no  4 (0..1000) :800
Intensity, peak no  4          :25
FWHM                      :100
Background                 :100

```

Press Q to quit iteration.

```

-----
IT:  0
FWHM      : 100.00   Background   : 100.00
Position( 1) : 200.00   Intensity( 1) : 25.000
Position( 2) : 300.00   Intensity( 2) : 100.00
Position( 3) : 550.00   Intensity( 3) : 150.00
Position( 4) : 800.00   Intensity( 4) : 25.000
-----

```


IT: 1 ,STEP : 0.181801
FWHM : 97.619 Background : 101.27
Position(1) : 198.67 Intensity(1) : 23.623
Position(2) : 298.83 Intensity(2) : 100.41
Position(3) : 549.91 Intensity(3) : 150.05
Position(4) : 799.68 Intensity(4) : 23.847

IT: 2 ,STEP : 0.648877E-02
FWHM : 97.424 Background : 101.29
Position(1) : 198.65 Intensity(1) : 23.629
Position(2) : 298.81 Intensity(2) : 100.45
Position(3) : 549.92 Intensity(3) : 150.16
Position(4) : 799.71 Intensity(4) : 23.831

IT: 3 ,STEP : 0.472963E-03
FWHM : 97.415 Background : 101.29
Position(1) : 198.65 Intensity(1) : 23.627
Position(2) : 298.80 Intensity(2) : 100.45
Position(3) : 549.92 Intensity(3) : 150.17
Position(4) : 799.72 Intensity(4) : 23.829

IT: 4 ,STEP : 0.288897E-04
FWHM : 97.414 Background : 101.29
Position(1) : 198.65 Intensity(1) : 23.627
Position(2) : 298.80 Intensity(2) : 100.45
Position(3) : 549.92 Intensity(3) : 150.17
Position(4) : 799.72 Intensity(4) : 23.828

Press any key to continue. <Return>

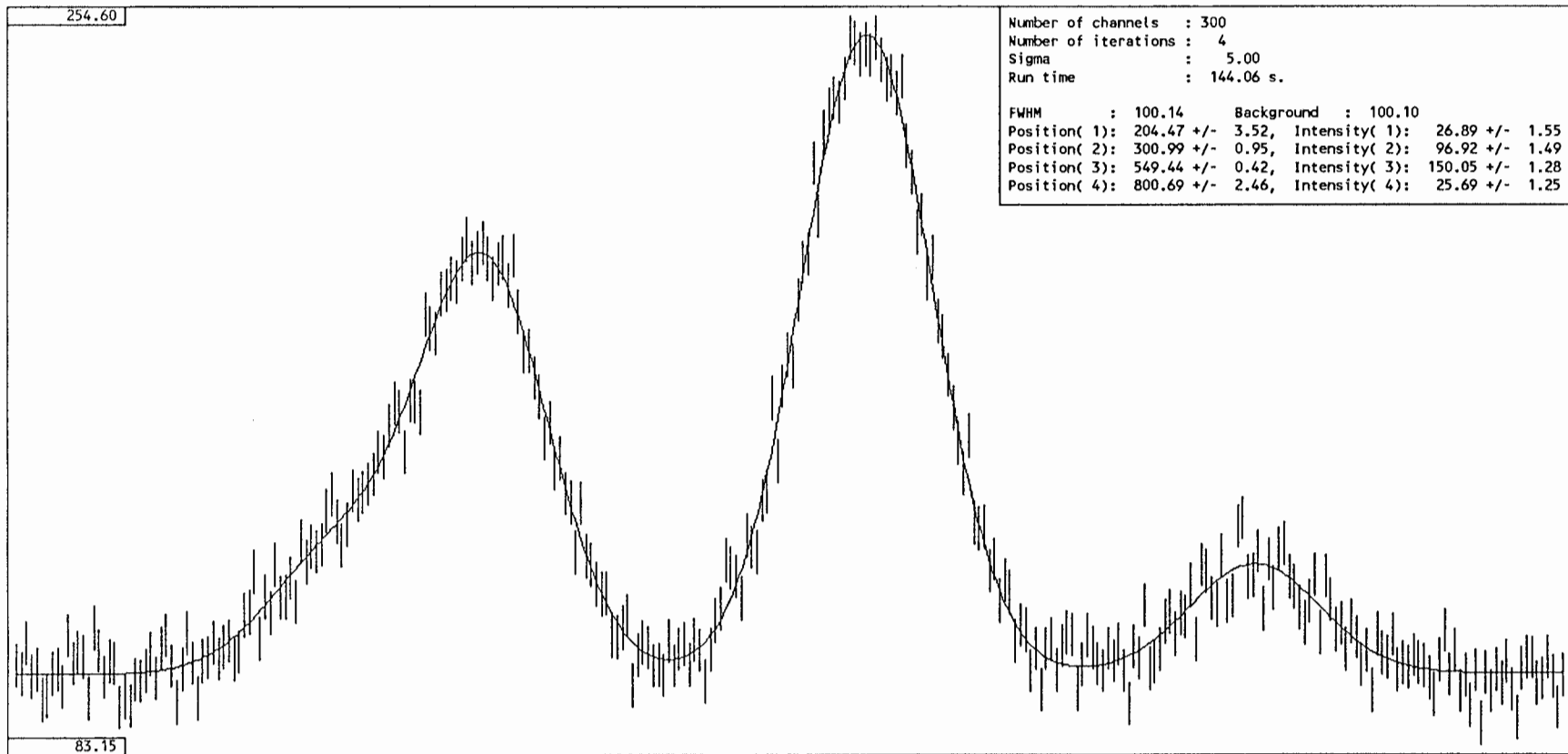
Plot on Roland plotter? (Y/*) :N

Plot on HP-Laserjet? (Y/*) :Y

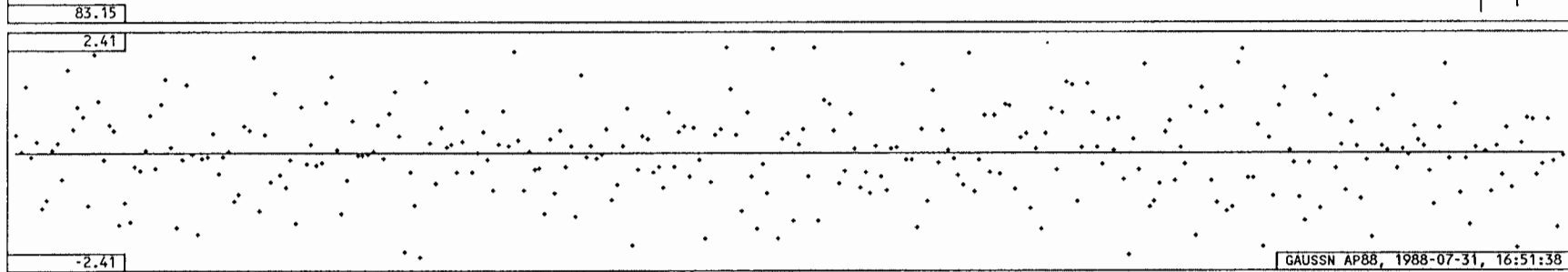
Number of channels : 300
Number of iterations : 4
Sigma : 5.00
Run time : 148.08 s.

FWHM : 97.414 Background : 101.29
Position(1) : 198.65 +/- 3.93, Intensity(1): 23.63 +/- 1.45
Position(2) : 298.80 +/- 0.87, Intensity(2): 100.45 +/- 1.42
Position(3) : 549.92 +/- 0.41, Intensity(3): 150.17 +/- 1.29
Position(4) : 799.72 +/- 2.63, Intensity(4): 23.83 +/- 1.26

New calculation? (Y/*):N



Number of channels : 300
 Number of iterations : 4
 Sigma : 5.00
 Run time : 144.06 s.
 FWHM : 100.14 Background : 100.10
 Position(1): 204.47 +/- 3.52, Intensity(1): 26.89 +/- 1.55
 Position(2): 300.99 +/- 0.95, Intensity(2): 96.92 +/- 1.49
 Position(3): 549.44 +/- 0.42, Intensity(3): 150.05 +/- 1.28
 Position(4): 800.69 +/- 2.46, Intensity(4): 25.69 +/- 1.25



GAUSSN AP88, 1988-07-31, 16:51:38

Kommentarer till programlistningen (för utförligare kommentarer till samma typ av program se EXPN-programmet):

Rad	Kommentar
1-20	Deklarationer.
22-81	Tillverka testkurva.
83-106	Läs in startvärden.
108-204	Anpassa med Newtons metod.
206-332	Skriv ut resultat och rita på skärmen, Roland pen-plotter eller HP laserskrivare.
334-343	Avsluta eller kör på nytt.
345-361	Rutinen beräknar funktionsvärdet $f(\theta;x)$, (70).
363-388	Rutinen beräknar förstaderivatorna av f , (74)..(77).
390-442	Rutinen beräknar andraderivatorna av f , (78)..(83).

```

1 PROGRAM GAUSSN
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5 INTEGER N,L,MAXITER,MAXNOOFPEAKS
6 PARAMETER (N=1000, MAXNOOFPEAKS=5, MAXITER=1000)
7 PARAMETER (L=2*MAXNOOFPEAKS+2)
8 REAL Y(1:N),X(1:N),DX,DET,RANF,NORMAL,THETAEST(1:L),SLASK
9 REAL RESIDUE(1:N),SIGMA(1:N),F,THETA(1:L),INVSIGMA2(1:N)
10 REAL GMATRIX(1:L,1:L),GVECTOR(1:L),DELTATHETA(1:L),D2FDTH2
11 REAL DFDTH,SIGMAIN,FWHM,POSITION(1:MAXNOOFPEAKS)
12 REAL INTENSITY(1:MAXNOOFPEAKS),SL1,SL2,SL3,SL4,SL5,TIME
13 REAL ERRPOS(1:MAXNOOFPEAKS),ERRINT(1:MAXNOOFPEAKS)
14 REAL JX,JY,JXL,JYL,DJY
15 INTEGER R,K,I,NCH,NOOFIT,NOOFPEAKS,LR,J
16 INTEGER*2 HO,MI,SE,HN,ANS,KBDCHK,KBDINC
17 INTEGER*2 YEAR,MONTH,DAY
18 CHARACTER SV*1,JETTXT*70
19 LOGICAL SING,ERRORBAR,LINEARIZED
20 EXTERNAL F
21
22 10 WRITE(*,1000) 'No of Gauss-peaks (1..',MAXNOOFPEAKS,') : '
23 1000 FORMAT(' ',A,I2,A)
24 READ(*,*,ERR=10) NOOFPEAKS
25 IF (NOOFPEAKS.LT.1.OR.NOOFPEAKS.GT.MAXNOOFPEAKS) THEN
26 GOTO 10
27 ENDIF
28 LR=2*NOOFPEAKS+2
29
30 20 WRITE(*,1100) 'No of channels (' ,LR,'..',N,') : '
31 1100 FORMAT('+',A,I2,A,I4,A)
32 READ(*,*,ERR=20) NCH
33 IF (NCH.LT.LR.OR.NCH.GT.N) GOTO 20
34
35 DO 30 I=1,NOOFPEAKS
36 WRITE(*,1200) 'Position, peak no ',I,' (0..1000): '
37 1200 FORMAT('+',A,I2,A)
38 READ(*,*) POSITION(I)
39 WRITE(*,1300) 'Intensity, peak no ',I,' : '
40 1300 FORMAT('+',A,I2,A)
41 READ(*,*) INTENSITY(I)
42 30 CONTINUE
43
44 WRITE(*,1200) 'FWHM : '
45 READ(*,*) FWHM
46 FWHM=ABS(FWHM)
47
48 WRITE(*,1200) 'Background : '
49 READ(*,*) THETA(LR)
50
51 WRITE(*,1200) 'Sigma(Y) : '
52 READ(*,*) SIGMAIN
53 SIGMAIN=ABS(SIGMAIN)
54
55 ERRORBAR=.FALSE.
56 IF (SIGMAIN.NE.0.0) THEN
57 WRITE(*,1200) 'Plot errorbars? (Y/*) : '

```

```

58 READ(*,1400) SV
59 1400 FORMAT(A)
60 ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'
61 ENDIF
62
63 THETA(LR-1)=LOG(2.0)/(0.5*FWHM)**2
64
65 DO 40 I=1,NOOFPEAKS
66 THETA(NOOFPEAKS+I)=POSITION(I)
67 THETA(I)=INTENSITY(I)
68 40 CONTINUE
69
70 DX=1000.0/(NCH-1)
71 DO 50 I=1,NCH
72 X(I)=(I-1)*DX
73 SLASK=F(THETA,NOOFPEAKS,X(I))
74 Y(I)=NORMAL(SLASK,SIGMAIN)
75 SIGMA(I)=SIGMAIN
76 IF (SIGMAIN.NE.0.0) THEN
77 INVSIGMA2(I)=-2.0/SIGMAIN**2
78 ELSE
79 INVSIGMA2(I)=-2.0
80 ENDIF
81 50 CONTINUE
82
83 WRITE(*,*) 'Enter start values for intensity, '
84 WRITE(*,*) 'position, FWHM and background. '
85 WRITE(*,*) '-----'
86 WRITE(*,*) ' '
87
88 DO 60 I=1,NOOFPEAKS
89 WRITE(*,1200) 'Position, peak no ',I,' (0..1000): '
90 READ(*,*) POSITION(I)
91 WRITE(*,1300) 'Intensity, peak no ',I,' : '
92 READ(*,*) INTENSITY(I)
93 60 CONTINUE
94
95 WRITE(*,1200) 'FWHM : '
96 READ(*,*) FWHM
97 FWHM=ABS(FWHM)
98
99 WRITE(*,1200) 'Background : '
100 READ(*,*) THETAEST(LR)
101 THETAEST(LR-1)=LOG(2.0)/(0.5*FWHM)**2
102
103 DO 70 I=1,NOOFPEAKS
104 THETAEST(NOOFPEAKS+I)=POSITION(I)
105 THETAEST(I)=INTENSITY(I)
106 70 CONTINUE
107
108 CALL GETTIM(HO,MI,SE,HN)
109 TIME=HO*3600.0+MI*60.0+SE+HN*0.01
110
111 WRITE(*,*) ' '
112 WRITE(*,*) 'Press Q to quit iteration.'
113 WRITE(*,*) ' '
114
115 FWHM=2.0*SQRT(ABS(LOG(2.0)/THETAEST(LR-1)))
116 DO 80 I=1,NOOFPEAKS
117 POSITION(I)=THETAEST(NOOFPEAKS+I)

```

```

118      INTENSITY(I)=THETAEST(I)
119 80    CONTINUE
120
121      WRITE(*,1500) 'IT:',J,' ',STEP:',SLASK
122      WRITE(*,1600) 'FWHM',':',FWHM,'Background  :',THETAEST(LR)
123      DO 90 I=1,NOOFPEAKS
124          WRITE(*,1700) 'Position(',I,'): ',POSITION(I),
125          f      Intensity(',I,'): ',INTENSITY(I)
126 90    CONTINUE
127
128      LINEARIZED=.TRUE.
129      DO 170 J=1,MAXITER
130          CALL ZEROMAT(GMATRIX,LR,LR,L,L)
131          CALL ZEROMAT(GVECTOR,LR,1,L,1)
132          DO 130 I=1,NCH
133              SL1=INVSIGMA2(I)
134              SL2=Y(I)-F(THETAEST,NOOFPEAKS,X(I))
135              DO 120 R=1,LR
136                  SL3=DFDTH(R,THETAEST,NOOFPEAKS,X(I))
137                  GVECTOR(R)=GVECTOR(R)+SL1*SL2*SL3
138                  DO 110 K=R,LR
139                      SL4=DFDTH(K,THETAEST,NOOFPEAKS,X(I))
140                      GMATRIX(R,K)=GMATRIX(R,K)+SL1*(-SL3*SL4)
141                      IF (.NOT.LINEARIZED) THEN
142                          SL5=D2FDTH2(R,K,THETAEST,NOOFPEAKS,X(I))
143                          GMATRIX(R,K)=GMATRIX(R,K)+SL1*SL2*SL5
144                      ENDIF
145                  GMATRIX(K,R)=GMATRIX(R,K)
146 110    CONTINUE
147 120    CONTINUE
148 130    CONTINUE
149
150      CALL INVMAT(GMATRIX,LR,DET,SING,L,L)
151      IF (SING) THEN
152          WRITE(*,*) 'The G-matrix is singular.'
153          GOTO 260
154      ENDIF
155
156      CALL MULTCMAT(GMATRIX,-1.0,LR,LR,L,L)
157      CALL MULTMAT(GMATRIX,GVECTOR,DELTATHETA,LR,LR,1,L,L,1,L,1)
158      CALL ADDMAT(THETAEST,DELTATHETA,THETAEST,LR,1,L,1,L,1,1)
159
160      SLASK=0.0
161      DO 140 I=1,LR
162          IF (THETAEST(I).NE.0.0) THEN
163              SLASK=SLASK+ABS(DELTATHETA(I)/THETAEST(I))
164          ENDIF
165 140    CONTINUE
166
167      FWHM=2.0*SQRT(ABS(LOG(2.0)/THETAEST(LR-1)))
168      DO 150 I=1,NOOFPEAKS
169          POSITION(I)=THETAEST(NOOFPEAKS+I)
170          INTENSITY(I)=THETAEST(I)
171 150    CONTINUE
172
173      WRITE(*,1500) 'IT:',J,' ',STEP:',SLASK
174 1500    FORMAT(' ',49(' '),',',',',A,I3,A,G13.6)
175      WRITE(*,1600) 'FWHM',':',FWHM,'Background  :',THETAEST(LR)
176 1600    FORMAT(' ',A,T14,A,G12.5,T29,A,G12.5)
177

```

```

178      DO 160 I=1,NOOFPEAKS
179          WRITE(*,1700) 'Position(',I,'): ',POSITION(I),
180          f      Intensity(',I,'): ',INTENSITY(I)
181 1700    FORMAT(' ',A,I2,A,G12.5,A,I2,A,G12.5)
182 160    CONTINUE
183
184      IF (SLASK.LE.0.5E-2.AND.LINEARIZED) THEN
185          LINEARIZED=.FALSE.
186      ENDIF
187      IF (SLASK.LE.0.5E-4.AND.(.NOT.LINEARIZED)) THEN
188          NOOFIT=J
189          GOTO 180
190      ENDIF
191
192      ANS=KBDCHK()
193      IF (ANS.NE.0) THEN
194          ANS=KBDINC()
195          IF (CHAR(ANS).EQ.'Q'.OR.CHAR(ANS).EQ.'q') THEN
196              NOOFIT=J
197              GOTO 180
198          ENDIF
199      ENDIF
200
201 170    CONTINUE
202      NOOFIT=MAXITER
203 180    CONTINUE
204      CALL MULTCMAT(GMATRIX,-2.0,LR,LR,L,L)
205
206      CALL GETTIM(HO,MI,SE,HN)
207      TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
208      WRITE(*,1400) ' '
209      WRITE(*,*) 'Press any key to continue.'
210 190    ANS=KBDCHK()
211          IF (ANS.EQ.0) THEN
212              GOTO 190
213          ENDIF
214          ANS=KBDINC()
215
216      CALL SCRIPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,NOOFPEAKS,ERRORBAR)
217
218 200    WRITE(*,*) 'Plot on Roland plotter? (Y/*):'
219          READ(*,1400) SV
220          IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
221              CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,NOOFPEAKS,ERRORBAR)
222              GOTO 200
223          ENDIF
224
225 210    WRITE(*,*) 'Plot on HP-Laserjet? (Y/*):'
226          READ(*,1400) SV
227          IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
228              CALL HPSTART('L')
229              CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
230          f      NOOFPEAKS,ERRORBAR)
231              JX=710
232              JXL=1119-JX
233              JYL=(6+1+NOOFPEAKS)*13
234              JY=719-JYL
235              CALL HPBOX(JX,JY,JXL,JYL)
236              JY=719-16
237              DJY=-13

```

```

238 WRITE(JETTXT,1900) 'Number of channels      :',NCH
239 CALL HPPRINT(JX,JY,70,JETTXT)
240 JY=JY+DJY
241
242 WRITE(JETTXT,1900) 'Number of iterations :',NOOFIT
243 CALL HPPRINT(JX,JY,70,JETTXT)
244 JY=JY+DJY
245
246 WRITE(JETTXT,2000) 'Sigma                :',SIGMAIN
247 CALL HPPRINT(JX,JY,70,JETTXT)
248 JY=JY+DJY
249
250 WRITE(JETTXT,2000) 'Run time                :',TIME,' s.'
251 CALL HPPRINT(JX,JY,70,JETTXT)
252 JY=JY+DJY
253
254 WRITE(JETTXT,1400) ' '
255 CALL HPPRINT(JX,JY,70,JETTXT)
256 JY=JY+DJY
257
258 FWHM=2.0*SQRT(ABS(LOG(2.0)/THETAEST(LR-1)))
259
260 DO 220 I=1,NOOFPEAKS
261   POSITION(I)=THETAEST(NOOFPEAKS+I)
262   ERRPOS(I)=SQRT(GMATRIX(NOOFPEAKS+I,NOOFPEAKS+I))
263   IF (SIGMAIN.EQ.0.0) THEN
264     ERRPOS(I)=0.0
265   ENDIF
266   INTENSITY(I)=THETAEST(I)
267   ERRINT(I)=SQRT(GMATRIX(I,I))
268   IF (SIGMAIN.EQ.0.0) THEN
269     ERRINT(I)=0.0
270   ENDIF
271 CONTINUE
272
273 WRITE(JETTXT,1600)
274 f 'FWHM',' :',FWHM,'Background  :',THETAEST(LR)
275 CALL HPPRINT(JX,JY,70,JETTXT)
276 JY=JY+DJY
277
278 DO 230 I=1,NOOFPEAKS
279   WRITE(JETTXT,2100) 'Position(' ,I,'):',POSITION(I),
280     ' +/- ',ERRPOS(I),
281     ' , Intensity(' ,I,'):',INTENSITY(I),
282     ' +/- ',ERRINT(I)
283   CALL HPPRINT(JX,JY,70,JETTXT)
284   JY=JY+DJY
285 CONTINUE
286
287 CALL GETTIM(HO,MI,SE,HN)
288 CALL GETDAT(YEAR,MONTH,DAY)
289 WRITE(JETTXT,2300)
290 f 'GAUSSN AP88, ' ,YEAR,'-',MONTH,'-',DAY,' ',HO,':',MI,':',SE
291 FORMAT(A,I4.4,5(A,I2.2))
292 CALL HPTBOX(909.0,0.0,33,JETTXT)
293
294 CALL HPEJECT
295 GOTO 210
296
297 ENDIF

```

```

298 WRITE(*,1800)
299 FORMAT(/)
300 1800 WRITE(*,1900) 'Number of channels :',NCH
301 WRITE(*,1900) 'Number of iterations:',NOOFIT
302 FORMAT(' ',A,I4)
303 1900 write(*,2000) 'Sigma                :',SIGMAIN
304 WRITE(*,2000) 'Run time                :',TIME,' s.'
305 FORMAT(' ',A,F8.2,A)
306 2000 WRITE(*,1800)
307
308 FWHM=2.0*SQRT(ABS(LOG(2.0)/THETAEST(LR-1)))
309
310 DO 240 I=1,NOOFPEAKS
311   POSITION(I)=THETAEST(NOOFPEAKS+I)
312   ERRPOS(I)=SQRT(GMATRIX(NOOFPEAKS+I,NOOFPEAKS+I))
313   IF (SIGMAIN.EQ.0.0) THEN
314     ERRPOS(I)=0.0
315   ENDIF
316   INTENSITY(I)=THETAEST(I)
317   ERRINT(I)=SQRT(GMATRIX(I,I))
318   IF (SIGMAIN.EQ.0.0) THEN
319     ERRINT(I)=0.0
320   ENDIF
321 CONTINUE
322
323 WRITE(*,1600) 'FWHM',' :',FWHM,'Background  :',THETAEST(LR)
324
325 DO 250 I=1,NOOFPEAKS
326   WRITE(*,2100) 'Position(' ,I,'):',POSITION(I),
327     ' +/- ',ERRPOS(I),
328     ' , Intensity(' ,I,'):',INTENSITY(I),
329     ' +/- ',ERRINT(I)
330 CONTINUE
331
332 250 FORMAT(' ',A,I2,A,F8.2,A,F5.2,A,I2,A,F8.2,A,F5.2)
333
334 260 WRITE(*,2200) 'New calculation? (Y/*):'
335 2200 FORMAT('/' ',A)
336 READ(*,1400) SV
337 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
338   GOTO 10
339 ELSE
340   STOP
341 ENDIF
342
343 END
344
345 REAL FUNCTION F(THETA,L,X)
346 C Anders Persson, 1988
347
348 IMPLICIT LOGICAL (A-Z)
349 INTEGER L,I,M
350 REAL THETA(1:*),X,SLASK,SL
351
352 SL(M)=EXP(-MIN(THETA(2*L+1)*(X-THETA(M))**2,80.0))
353
354 SLASK=THETA(2*L+2)
355 DO 10 I=1,L
356   SLASK=SLASK+THETA(I)*SL(L+I)
357 10 CONTINUE

```

```

358 F=SLASK
359 RETURN
360
361 END
362
363 REAL FUNCTION DFDTH(J,THETA,L,X)
364 C Anders Persson, 1988
365
366 IMPLICIT LOGICAL (A-Z)
367 INTEGER J,L,I,M
368 REAL THETA(1:*),X,SLASK,SL
369
370 SL(M)=EXP(-MIN(THETA(2*L+1)*(X-THETA(M))**2,80.0))
371
372 DFDTH=0.0
373 IF (J.LE.L) THEN
374 DFDTH=SL(L+J)
375 ELSEIF (J.GT.L.AND.J.LE.2*L) THEN
376 DFDTH=2.0*THETA(J-L)*THETA(2*L+1)*(X-THETA(J))*SL(J)
377 ELSEIF (J.EQ.2*L+1) THEN
378 SLASK=0.0
379 DO 10 I=1,L
380 SLASK=SLASK-THETA(I)*((X-THETA(L+I))**2)*SL(L+I)
381 10 CONTINUE
382 DFDTH=SLASK
383 ELSEIF (J.EQ.2*L+2) THEN
384 DFDTH=1.0
385 ENDIF
386
387 RETURN
388 END
389
390 REAL FUNCTION D2FDTH2(J,K,THETA,L,X)
391 C Anders Persson, 1988
392
393 IMPLICIT LOGICAL (A-Z)
394 INTEGER J,K,L,I,M
395 REAL THETA(1:*),X,SL,SLASK
396 LOGICAL JGTK
397
398 SL(M)=EXP(-MIN(THETA(2*L+1)*(X-THETA(M))**2,80.0))
399
400 JGTK=J.GT.K
401 IF (JGTK) THEN
402 I=J
403 J=K
404 K=I
405 ENDIF
406 D2FDTH2=0.0
407
408 IF (J.LE.L) THEN
409 IF (K.EQ.L+J) THEN
410 D2FDTH2=2.0*THETA(2*L+1)*(X-THETA(L+J))*SL(L+J)
411 ELSEIF (K.EQ.2*L+1) THEN
412 D2FDTH2=-((X-THETA(L+J))**2)*SL(L+J)
413 ENDIF
414 ELSEIF (J.GT.L.AND.J.LE.2*L) THEN
415 IF (K.EQ.J) THEN
416 SLASK=THETA(J-L)*THETA(2*L+1)*SL(J)
417 D2FDTH2=SLASK*(4.0*THETA(2*L+1)*(X-THETA(J))**2-2.0)

```

```

418 ELSEIF (K.EQ.2*L+1) THEN
419 SLASK=2.0*THETA(J-L)*(X-THETA(J))*SL(J)
420 D2FDTH2=SLASK*(1.0-THETA(2*L+1)*(X-THETA(J))**2)
421 ENDIF
422 ELSEIF (J.EQ.2*L+1) THEN
423 IF (K.GT.L.AND.K.LE.2*L) THEN
424 SLASK=2.0*THETA(K-L)*(X-THETA(K))*SL(K)
425 D2FDTH2=SLASK*(1.0-THETA(2*L+1)*(X-THETA(K))**2)
426 ELSEIF (K.EQ.2*L+1) THEN
427 SLASK=0.0
428 DO 10 I=1,L
429 SLASK=SLASK+THETA(I)*((X-THETA(L+I))**4)*SL(L+I)
430 10 CONTINUE
431 D2FDTH2=SLASK
432 ENDIF
433 ENDIF
434
435 IF (JGTK) THEN
436 I=J
437 J=K
438 K=I
439 ENDIF
440
441 RETURN
442 END

```

LORN

Programmet utför exakt samma beräkningar som GAUSSN-programmet men på en summa av Lorentz-toppar istället.

Anpassad funktion:

$$f(\underline{\theta}; x) = \sum_{i=1}^L \theta_i [(x-\theta_{L+i})^2 + \theta_{2L+1}]^{-1} + \theta_{2L+2} \quad (84)$$

$$\text{Position, topp nr. } i: \quad \theta_{L+i} \quad (85)$$

$$\text{Intensitet, topp nr. } i: \quad \theta_i / \theta_{2L+1} \quad (86)$$

$$\text{Halvvärdesbredd (alla): } 2 \cdot \sqrt{\theta_{2L+1}} \quad (87)$$

Förstaderivator:

$$\frac{\partial f}{\partial \theta_j} (j=1..L) = [(x-\theta_{L+j})^2 + \theta_{2L+1}]^{-1} \quad (88)$$

$$\frac{\partial f}{\partial \theta_j} (j=L+1..2L) = 2 \cdot \theta_{j-L} (x-\theta_j) \cdot [(x-\theta_j)^2 + \theta_{2L+1}]^{-2} \quad (89)$$

$$\frac{\partial f}{\partial \theta_j} (j=2L+1) = \sum_{i=1}^L -\theta_i [(x-\theta_{L+i})^2 + \theta_{2L+1}]^{-2} \quad (90)$$

$$\frac{\partial f}{\partial \theta_j} (j=2L+2) = 1 \quad (91)$$

Andraderivator:

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left(\begin{matrix} j=1..L \\ k=L+j \end{matrix} \right) = 2(x-\theta_k) \cdot [(x-\theta_k)^2 + \theta_{2L+1}]^{-2} \quad (92)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left(\begin{matrix} j=1..L \\ k=2L+1 \end{matrix} \right) = -[(x-\theta_{L+j})^2 + \theta_{2L+1}]^{-2} \quad (93)$$

$$\begin{aligned} \frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left(\begin{matrix} j=L+1..2L \\ k=j \end{matrix} \right) &= 8\theta_{j-L} (x-\theta_j)^2 \cdot [(x-\theta_j)^2 + \theta_{2L+1}]^{-3} - \\ &\quad - 2\theta_{j-L} \cdot [(x-\theta_j)^2 + \theta_{2L+1}]^{-2} \end{aligned} \quad (94)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left(\begin{matrix} j=L+1..2L \\ k=2L+1 \end{matrix} \right) = -4\theta_{j-L} (x-\theta_j) \cdot [(x-\theta_j)^2 + \theta_{2L+1}]^{-3} \quad (95)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left(\begin{matrix} j=2L+1 \\ k=2L+1 \end{matrix} \right) = \sum_{i=1}^L 2\theta_i \cdot [(x-\theta_j)^2 + \theta_{2L+1}]^{-3} \quad (96)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} = \frac{\partial^2 f}{\partial \theta_k \partial \theta_j} \quad (97)$$

alla övriga andraderivator är noll.

Utskrift från testkörning av programmet LORN:

```
No of Lorentz-peaks (1.. 5)      :4
No of channels (10..1000)       :300
Position, peak no  1 (0..1000)  :200
Intensity, peak no  1              :25
Position, peak no  2 (0..1000)  :300
Intensity, peak no  2              :100
Position, peak no  3 (0..1000)  :550
Intensity, peak no  3              :150
Position, peak no  4 (0..1000)  :800
Intensity, peak no  4              :25
FWHM                             :100
Background                       :100
Sigma(Y)                          :5
Plot errorbars? (Y/*)            :Y
```

Enter start values for intensity,
position, FWHM and background.

```
-----
Position, peak no  1 (0..1000)  :200
Intensity, peak no  1              :25
Position, peak no  2 (0..1000)  :300
Intensity, peak no  2              :100
Position, peak no  3 (0..1000)  :550
Intensity, peak no  3              :150
Position, peak no  4 (0..1000)  :800
Intensity, peak no  4              :25
FWHM                             :100
Background                       :100
```

Press Q to quit iteration.

```
-----
IT:  0
FWHM      : 100.00   Background   : 100.00
Position( 1) : 200.00   Intensity( 1) : 25.000
Position( 2) : 300.00   Intensity( 2) : 100.00
Position( 3) : 550.00   Intensity( 3) : 150.00
Position( 4) : 800.00   Intensity( 4) : 25.000
```

```
-----
IT:  1 ,STEP : 0.323363
FWHM      : 97.930   Background   : 101.27
Position( 1) : 201.82   Intensity( 1) : 27.169
Position( 2) : 301.25   Intensity( 2) : 98.798
Position( 3) : 550.39   Intensity( 3) : 150.34
Position( 4) : 799.94   Intensity( 4) : 23.319
```

```
-----
IT:  2 ,STEP : 0.115276E-01
FWHM      : 97.903   Background   : 101.28
Position( 1) : 201.50   Intensity( 1) : 27.008
Position( 2) : 301.18   Intensity( 2) : 98.963
Position( 3) : 550.39   Intensity( 3) : 150.37
Position( 4) : 799.99   Intensity( 4) : 23.356
-----
```

IT: 3 ,STEP : 0.538773E-03
FWHM : 97.900 Background : 101.28
Position(1) : 201.51 Intensity(1) : 27.013
Position(2) : 301.18 Intensity(2) : 98.959
Position(3) : 550.39 Intensity(3) : 150.37
Position(4) : 799.99 Intensity(4) : 23.355

IT: 4 ,STEP : 0.180735E-04
FWHM : 97.900 Background : 101.28
Position(1) : 201.51 Intensity(1) : 27.013
Position(2) : 301.18 Intensity(2) : 98.959
Position(3) : 550.39 Intensity(3) : 150.37
Position(4) : 799.99 Intensity(4) : 23.355

Press any key to continue. <Return>

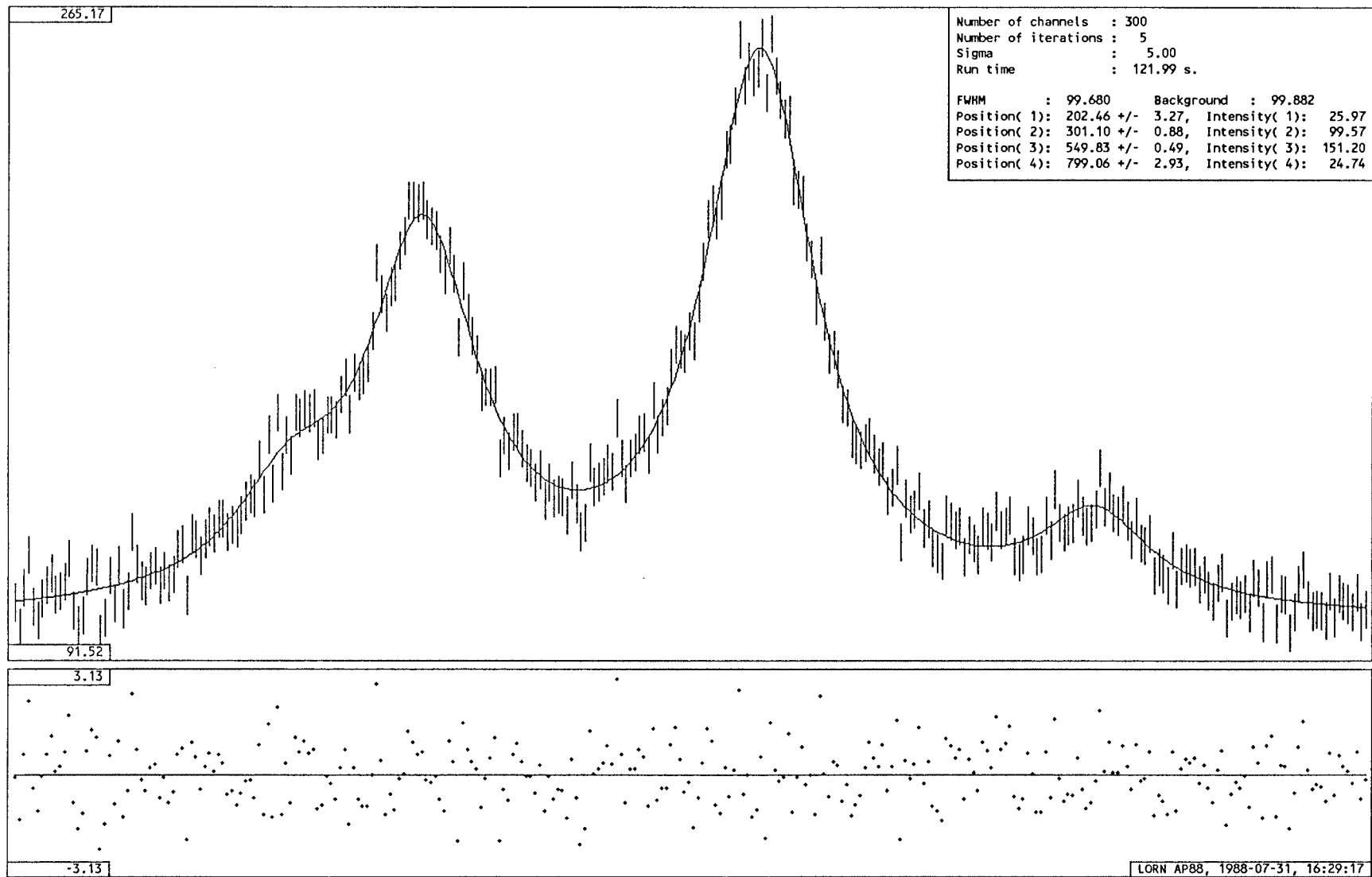
Plot on Roland plotter? (Y/*) :N

Plot on HP-Laserjet? (Y/*) :Y

Number of channels : 300
Number of iterations : 4
Sigma : 5.00
Run time : 90.13 s.

FWHM : 97.900 Background : 101.28
Position(1) : 201.51 +/- 3.25, Intensity(1): 27.01
Position(2) : 301.18 +/- 0.87, Intensity(2): 98.96
Position(3) : 550.39 +/- 0.48, Intensity(3): 150.37
Position(4) : 799.99 +/- 3.34, Intensity(4): 23.36

New calculation? (Y/*):N



Kommentarer till programlistningen (för utförligare kommentarer till samma typ av program se EXPN programmet.):

Rad	Kommentar
1-19	Deklarationer.
21-80	Tillverka testkurva.
82-105	Läs in startvärden.
107-207	Anpassa med Newtons metod.
208-320	Skriv ut resultat och rita på skärmen, Roland pen-plotter eller HP laserskrivare.
322-330	Avsluta eller kör på nytt.
332-346	Rutinen beräknar funktionsvärdet $f(\theta; x)$, (84).
348-373	Rutinen beräknar förstaderivatorna av f , (88)..(91).
375-420	Rutinen beräknar andraderivatorna av f , (92)..(97).

```

1 PROGRAM LORN
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5 INTEGER N,L,MAXITER,MAXNOOFPEAKS
6 PARAMETER (N=1000, MAXNOOFPEAKS=5, MAXITER=1000)
7 PARAMETER (L=2*MAXNOOFPEAKS+2)
8 REAL Y(1:N),X(1:N),DX,DET,RANF,NORMAL,THETAEST(1:L),SLASK
9 REAL RESIDUE(1:N),SIGMA(1:N),F,THETA(1:L),INVSIGMA2(1:N)
10 REAL GMATRIX(1:L,1:L),GVECTOR(1:L),DELTATHETA(1:L),D2FDTH2
11 REAL DFDTH,SIGMAIN,FWHM,POSITION(1:MAXNOOFPEAKS)
12 REAL INTENSITY(1:MAXNOOFPEAKS),SL1,SL2,SL3,SL4,SL5,TIME
13 REAL ERRPOS(1:MAXNOOFPEAKS),JX,JY,JXL,JYL,DJY
14 INTEGER R,K,I,NCH,NOOFIT,NOOFPEAKS,LR,J
15 INTEGER*2 HO,MI,SE,HN,ANS,KBDCHK,KBDINC
16 INTEGER*2 YEAR,MONTH,DAY
17 CHARACTER SV*1,JETTXT*60
18 LOGICAL SING,ERRORBAR,LINEARIZED
19 EXTERNAL F
20
21 10 WRITE(*,1000) 'No of Lorentz-peaks (1..',MAXNOOFPEAKS,') : '
22 1000 FORMAT(' ',A,12,A)
23 READ(*,*,ERR=10) NOOFPEAKS
24 IF (NOOFPEAKS.LT.1.OR.NOOFPEAKS.GT.MAXNOOFPEAKS) THEN
25 GOTO 10
26 ENDIF
27 LR=2*NOOFPEAKS+2
28
29 20 WRITE(*,1100) 'No of channels (' ,LR,'..',N,') : '
30 1100 FORMAT('+',A,12,A,I4,A)
31 READ(*,*,ERR=20) NCH
32 IF (NCH.LT.LR.OR.NCH.GT.N) GOTO 20
33
34 DO 30 I=1,NOOFPEAKS
35 WRITE(*,1200) 'Position, peak no ',I, ' (0..1000): '
36 1200 FORMAT('+',A,12,A)
37 READ(*,*) POSITION(I)
38 WRITE(*,1300) 'Intensity, peak no ',I, ' : '
39 1300 FORMAT('+',A,12,A)
40 READ(*,*) INTENSITY(I)
41 30 CONTINUE
42
43 WRITE(*,1200) 'FWHM : '
44 READ(*,*) FWHM
45 FWHM=ABS(FWHM)
46
47 WRITE(*,1200) 'Background : '
48 READ(*,*) THETA(LR)
49
50 WRITE(*,1200) 'Sigma(Y) : '
51 READ(*,*) SIGMAIN
52 SIGMAIN=ABS(SIGMAIN)
53
54 ERRORBAR=.FALSE.
55 IF (SIGMAIN.NE.0.0) THEN
56 WRITE(*,1200) 'Plot errorbars? (Y/*) : '
57 READ(*,1400) SV
    
```

```

58 1400 FORMAT(A)
59 ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'
60 ENDIF
61
62 THETA(LR-1)=(0.5*FWHM)**2
63
64 DO 40 I=1,NOOFPEAKS
65 THETA(NOOFPEAKS+I)=POSITION(I)
66 THETA(I)=INTENSITY(I)*THETA(LR-1)
67 40 CONTINUE
68
69 DX=1000.0/(NCH-1)
70 DO 50 I=1,NCH
71 X(I)=(I-1)*DX
72 SLASK=F(THETA,NOOFPEAKS,X(I))
73 Y(I)=NORMAL(SLASK,SIGMAIN)
74 SIGMA(I)=SIGMAIN
75 IF (SIGMAIN.NE.0.0) THEN
76 INVSIGMA2(I)=-2.0/SIGMAIN**2
77 ELSE
78 INVSIGMA2(I)=-2.0
79 ENDIF
80 50 CONTINUE
81
82 WRITE(*,*) 'Enter start values for intensity, '
83 WRITE(*,*) 'position, FWHM and background.'
84 WRITE(*,*) '-----'
85 WRITE(*,*) ' '
86
87 DO 60 I=1,NOOFPEAKS
88 WRITE(*,1200) 'Position, peak no ',I, ' (0..1000): '
89 READ(*,*) POSITION(I)
90 WRITE(*,1300) 'Intensity, peak no ',I, ' : '
91 READ(*,*) INTENSITY(I)
92 60 CONTINUE
93
94 WRITE(*,1200) 'FWHM : '
95 READ(*,*) FWHM
96 FWHM=ABS(FWHM)
97
98 WRITE(*,1200) 'Background : '
99 READ(*,*) THETA(LR)
100
101 THETAEST(LR-1)=(0.5*FWHM)**2
102 DO 70 I=1,NOOFPEAKS
103 THETAEST(NOOFPEAKS+I)=POSITION(I)
104 THETAEST(I)=INTENSITY(I)*THETAEST(LR-1)
105 70 CONTINUE
106
107 CALL GETTIM(HO,MI,SE,HN)
108 TIME=HO*3600.0+MI*60.0+SE+HN*0.01
109
110 WRITE(*,*) ' '
111 WRITE(*,*) 'Press Q to quit iteration.'
112 WRITE(*,*) ' '
113
114 FWHM=2.0*SQRT(ABS(THETAEST(LR-1)))
115 DO 80 I=1,NOOFPEAKS
116 POSITION(I)=THETAEST(NOOFPEAKS+I)
117 INTENSITY(I)=THETAEST(I)/THETAEST(LR-1)
    
```

```

118 80 CONTINUE
119
120 WRITE(*,1500) 'IT:',0
121 WRITE(*,1600) 'FWHM',':',FWHM,'Background  :',THETAEST(LR)
122 DO 90 I=1,NOOFPEAKS
123   WRITE(*,1700) 'Position(',I,'):',POSITION(I),
124   f      ' Intensity(',I,'):',INTENSITY(I)
125 90 CONTINUE
126
127 LINEARIZED=.TRUE.
128 DO 170 J=1,MAXITER
129   CALL ZEROMAT(GMATRIX,LR,LR,L,L)
130   CALL ZEROMAT(GVECTOR,LR,1,L,1)
131   DO 130 I=1,NCH
132     SL1=INVSIGMA2(I)
133     SL2=Y(I)-F(THETAEST,NOOFPEAKS,X(I))
134     DO 120 R=1,LR
135       SL3=DFDTH(R,THETAEST,NOOFPEAKS,X(I))
136       GVECTOR(R)=GVECTOR(R)+SL1*SL2*SL3
137       DO 110 K=R,LR
138         SL4=DFDTH(K,THETAEST,NOOFPEAKS,X(I))
139         GMATRIX(R,K)=GMATRIX(R,K)+SL1*(-SL3*SL4)
140         IF (.NOT.LINEARIZED) THEN
141           SL5=D2FDTH2(R,K,THETAEST,NOOFPEAKS,X(I))
142           GMATRIX(R,K)=GMATRIX(R,K)+SL1*SL2*SL5
143         ENDIF
144         GMATRIX(K,R)=GMATRIX(R,K)
145 110 CONTINUE
146 120 CONTINUE
147 130 CONTINUE
148
149 CALL INVMAT(GMATRIX,LR,DET,SING,L,L)
150 IF (SING) THEN
151   WRITE(*,*) 'The G-matrix is singular.'
152   GOTO 260
153 ENDIF
154
155 CALL MULTCMAT(GMATRIX,-1.0,LR,LR,L,L)
156 CALL MULTMAT(GMATRIX,GVECTOR,DELTATHETA,LR,LR,1,L,L,1,L,1)
157 CALL ADDMAT(THETAEST,DELTATHETA,THETAEST,LR,1,L,1,L,1,L,1)
158
159 SLASK=0.0
160 DO 140 I=1,LR
161   IF (THETAEST(I).NE.0.0) THEN
162     SLASK=SLASK+ABS(DELTATHETA(I)/THETAEST(I))
163   ENDIF
164 140 CONTINUE
165
166 FWHM=2.0*SQRT(ABS(THETAEST(LR-1)))
167 DO 150 I=1,NOOFPEAKS
168   POSITION(I)=THETAEST(NOOFPEAKS+I)
169   INTENSITY(I)=THETAEST(I)/THETAEST(LR-1)
170 150 CONTINUE
171
172 WRITE(*,1500) 'IT:',J,' ,STEP:',SLASK
173 1500 FORMAT(' ',49(' '),/,/,',A,13,A,G13.6)
174 WRITE(*,1600) 'FWHM',':',FWHM,'Background  :',THETAEST(LR)
175 1600 FORMAT(' ',A,T14,A,G12.5,T29,A,G12.5)
176
177 DO 160 I=1,NOOFPEAKS

```

```

178   WRITE(*,1700) 'Position(',I,'):',POSITION(I),
179   f      ' Intensity(',I,'):',INTENSITY(I)
180 1700 FORMAT(' ',A,I2,A,G12.5,A,I2,A,G12.5)
181 160 CONTINUE
182
183 IF (SLASK.LE.0.5E-2.AND.LINEARIZED) THEN
184   LINEARIZED=.FALSE.
185 ENDIF
186
187 IF (SLASK.LE.0.5E-4.AND.(.NOT.LINEARIZED)) THEN
188   NOOFIT=J
189   GOTO 180
190 ENDIF
191
192 ANS=KBDCHK()
193 IF (ANS.NE.0) THEN
194   ANS=KBDINC()
195   IF (CHAR(ANS).EQ.'q'.OR.CHAR(ANS).EQ.'q') THEN
196     NOOFIT=J
197     GOTO 180
198   ENDIF
199 ENDIF
200
201 170 CONTINUE
202 NOOFIT=MAXITER
203 180 CONTINUE
204 CALL MULTCMAT(GMATRIX,-2.0,LR,LR,L,L)
205
206 CALL GETTIM(HO,MI,SE,HN)
207 TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
208 WRITE(*,1400) ' '
209 WRITE(*,*) 'Press any key to continue.'
210 190 ANS=KBDCHK()
211 IF (ANS.EQ.0) THEN
212   GOTO 190
213 ENDIF
214 ANS=KBDINC()
215
216 CALL SCRPLLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,NOOFPEAKS,ERRORBAR)
217
218 200 WRITE(*,*) 'Plot on Roland plotter? (Y/*):'
219 READ(*,1400) SV
220 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
221   CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,NOOFPEAKS,ERRORBAR)
222   GOTO 200
223 ENDIF
224
225 210 WRITE(*,*) 'Plot on HP-Laserjet? (Y/*):'
226 READ(*,1400) SV
227 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
228   CALL HPSTART('L')
229   CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
230   f      NOOFPEAKS,ERRORBAR)
231   JX=770
232   JXL=1119-JX
233   JYL=(6+1+NOOFPEAKS)*13
234   JY=719-JYL
235   CALL HPBOX(JX,JY,JXL,JYL)
236   JY=719-16
237   DJY=-13

```

```

238
239 WRITE(JETTXT,1900) 'Number of channels      :',NCH
240 CALL HPPRINT(JX,JY,60,JETTXT)
241 JY=JY+DJY
242
243 WRITE(JETTXT,1900) 'Number of iterations :',NOOFIT
244 CALL HPPRINT(JX,JY,60,JETTXT)
245 JY=JY+DJY
246
247 WRITE(JETTXT,2000) 'Sigma                :',SIGMAIN
248 CALL HPPRINT(JX,JY,60,JETTXT)
249 JY=JY+DJY
250
251 WRITE(JETTXT,2000) 'Run time                :',TIME,' s.'
252 CALL HPPRINT(JX,JY,60,JETTXT)
253 JY=JY+DJY
254
255 WRITE(JETTXT,1400) ' '
256 CALL HPPRINT(JX,JY,60,JETTXT)
257 JY=JY+DJY
258
259 FWHM=2.0*SQRT(ABS(THETAEST(LR-1)))
260 DO 220 I=1,NOOFPEAKS
261   POSITION(I)=THETAEST(NOOFPEAKS+I)
262   ERRPOS(I)=SQRT(GMATRIX(NOOFPEAKS+I,NOOFPEAKS+I))
263   IF (SIGMAIN.EQ.0.0) THEN
264     ERRPOS(I)=0.0
265   ENDIF
266   INTENSITY(I)=THETAEST(I)/THETAEST(LR-1)
267 220 CONTINUE
268
269 WRITE(JETTXT,1600)
270 f 'FWHM',' ',FWHM,'Background      ',THETAEST(LR)
271 CALL HPPRINT(JX,JY,60,JETTXT)
272 JY=JY+DJY
273
274 DO 230 I=1,NOOFPEAKS
275   WRITE(JETTXT,2100) 'Position('',I,''):',POSITION(I),
276 f ' +/- ',ERRPOS(I),
277 f ' , Intensity('',I,''):',INTENSITY(I)
278   CALL HPPRINT(JX,JY,60,JETTXT)
279   JY=JY+DJY
280 230 CONTINUE
281
282 CALL GETTIM(HO,MI,SE,HN)
283 CALL GETDAT(YEAR,MONTH,DAY)
284 WRITE(JETTXT,2300)
285 f 'LORN AP88, ',YEAR,'-',MONTH,'-',DAY,' ',HO,':',MI,':',SE
286 2300 FORMAT(A,14.4,5(A,12.2))
287 CALL HPTBOX(921.0,0.0,31,JETTXT)
288
289 CALL HPEJECT
290 GOTO 210
291 ENDIF
292
293 WRITE(*,1800)
294 1800 FORMAT(/)
295 WRITE(*,1900) 'Number of channels      :',NCH
296 WRITE(*,1900) 'Number of iterations:',NOOFIT
297 1900 FORMAT(' ',A,14)

```

```

298 WRITE(*,2000) 'Sigma                :',SIGMAIN
299 WRITE(*,2000) 'Run time                :',TIME,' s.'
300 2000 FORMAT(' ',A,F8.2,A)
301 WRITE(*,1800)
302
303 FWHM=2.0*SQRT(ABS(THETAEST(LR-1)))
304 DO 240 I=1,NOOFPEAKS
305   POSITION(I)=THETAEST(NOOFPEAKS+I)
306   ERRPOS(I)=SQRT(GMATRIX(NOOFPEAKS+I,NOOFPEAKS+I))
307   IF (SIGMAIN.EQ.0.0) THEN
308     ERRPOS(I)=0.0
309   ENDIF
310   INTENSITY(I)=THETAEST(I)/THETAEST(LR-1)
311 240 CONTINUE
312
313 WRITE(*,1600) 'FWHM',' ',FWHM,'Background      :',THETAEST(LR)
314
315 DO 250 I=1,NOOFPEAKS
316   WRITE(*,2100) 'Position('',I,''):',POSITION(I),
317 f ' +/- ',ERRPOS(I),
318 f ' , Intensity('',I,''):',INTENSITY(I)
319 250 CONTINUE
320 2100 FORMAT(' ',A,12,A,F8.2,A,F5.2,A,12,A,F8.2)
321
322 260 WRITE(*,2200) 'New calculation?      (Y/*):'
323 2200 FORMAT('/' ',A)
324 READ(*,1400) SV
325 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
326   GOTO 10
327 ELSE
328   STOP
329 ENDIF
330 END
331
332 REAL FUNCTION F(THETA,L,X)
333 C Anders Persson, 1988
334
335 IMPLICIT LOGICAL (A-Z)
336 INTEGER L,I
337 REAL THETA(1:*),X,SLASK
338
339 SLASK=THETA(2*L+2)
340 DO 10 I=1,L
341   SLASK=SLASK+THETA(I)/((X-THETA(L+1))**2+THETA(2*L+1))
342 10 CONTINUE
343 F=SLASK
344
345 RETURN
346 END
347
348 REAL FUNCTION DFDTH(J,THETA,L,X)
349 C Anders Persson, 1988
350
351 IMPLICIT LOGICAL (A-Z)
352 INTEGER J,L,I
353 REAL THETA(1:L),X,SLASK1,SLASK2
354
355 IF (J.GE.1.AND.J.LE.L) THEN
356   DFDTH=1.0/((X-THETA(L+J))**2+THETA(2*L+1))
357 ELSEIF (J.GE.L+1.AND.J.LE.2*L) THEN

```

```

358     SLASK1=2.0*THETA(J-L)*(X-THETA(J))
359     SLASK2=((X-THETA(J))**2+THETA(2*L+1))**2
360     DFDTH=SLASK1/SLASK2
361     ELSEIF (J.EQ.2*L+1) THEN
362     SLASK1=0.0
363     DO 10 I=1,L
364         SLASK2=((X-THETA(L+I))**2+THETA(2*L+1))**2
365         SLASK1=SLASK1-THETA(I)/SLASK2
366 10    CONTINUE
367     DFDTH=SLASK1
368     ELSEIF (J.EQ.2*L+2) THEN
369     DFDTH=1.0
370     ENDIF
371
372     RETURN
373     END
374
375     REAL FUNCTION D2FDTH2(J,K,THETA,L,X)
376 C    Anders Persson, 1988
377
378     IMPLICIT LOGICAL (A-Z)
379     INTEGER J,K,L,I,M
380     REAL THETA(1:*),X,SL,SLASK
381     LOGICAL JGTK
382
383     SL(M)=1.0/((X-THETA(M))**2+THETA(2*L+1))
384     JGTK=J.GT.K
385     IF (JGTK) THEN
386         I=J
387         J=K
388         K=I
389     ENDIF
390     D2FDTH2=0.0
391
392     IF (J.LE.L) THEN
393         IF (K.EQ.L+J) THEN
394             D2FDTH2=2.0*(X-THETA(K))*SL(K)**2
395         ELSEIF (K.EQ.2*L+1) THEN
396             D2FDTH2=-(SL(L+J)**2)
397         ENDIF
398     ELSEIF (J.GT.L.AND.J.LE.2*L) THEN
399         IF (K.EQ.J) THEN
400             SLASK=8.0*THETA(J-L)*((X-THETA(J))**2)*SL(J)**3
401             D2FDTH2=SLASK-2.0*THETA(J-L)*SL(J)**2
402         ELSEIF (K.EQ.2*L+1) THEN
403             D2FDTH2=-4.0*THETA(J-L)*(X-THETA(J))*SL(J)**3
404         ENDIF
405     ELSEIF (J.EQ.2*L+1.AND.K.EQ.2*L+1) THEN
406         SLASK=0.0
407         DO 10 I=1,L
408             SLASK=SLASK+2.0*THETA(I)*SL(L+I)**3
409 10    CONTINUE
410         D2FDTH2=SLASK
411     ENDIF
412
413     IF (JGTK) THEN
414         I=J
415         J=K
416         K=I
417     ENDIF
418
419     RETURN
420     END

```


PCONEXPA

Programmet anpassar en summa av exponentieller faltad med en laserpuls plus bakgrund och ströljus till indata. Fria parametrar är ströljusintensitet, laserpulsens position, bakgrund samt intensitet och sönderfallskonstant för exponentialerna. För att kunna beräkna g-vektorn och G-matrisen (37),(42) behövs ett antal faltningar av laserpuls och exponentieller och derivator av laserpulsens m.m. Dessa vektorer beräknas genom att Fouriertransformera puls och exponentieller och därefter utföra derivation, faltning och translation i transformplanet. Efter beräkning transformerar programmet tillbaka till tidsplanet, denna omväg går fortare om vektorernas längder är större än ≈ 64 . Detta program minimerar inte beräkningsarbetet genom att använda diverse trick vid beräkning av faltningar och Fouriertransformer utan använder standardformler (nästa program PCONEXP försöker minimera exekveringstiden genom att utföra beräkningarna på ett effektivt sätt). Detta program ger möjlighet att undvika s.k. cirkulär faltning genom att använda dubbla längden på datavektorer. Nyttan av detta kan diskuteras, om någorlunda hyfsade indata används (d.v.s. tillräckligt lång tid före och efter laserpulsens) har inte inverkan av cirkulär faltning särskilt stor betydelse för resultatet, däremot är kostnaden i exekveringstid stor, den mer än fördubblas. För anpassningar till verkliga mätdata rekommenderas därför programmet PCONEXP men programmet PCONEXPA har fördelen att det är lättare att sätta sig in i beräkningarna om man vill förstå exakt vad programmen gör.

Anpassad funktion:

(\otimes = faltning, $\dot{}$ = derivation, $P(x)$ = laserpuls)

$$f(\underline{\theta}; x) = \theta_1 \cdot P(x-\theta_2) + P(x-\theta_2) \otimes \sum_{i=1}^L \theta_{i+2} \cdot \exp(-\theta_{L+i+2} \cdot x) + \theta_{2L+3} \quad (98)$$

Förstaderivator:

$$\frac{\partial f}{\partial \theta_j} (j=1) = P(x-\theta_2) \quad (99)$$

$$\frac{\partial f}{\partial \theta_j} (j=2) = -\theta_1 \cdot \dot{P}(x-\theta_2) - \dot{P}(x-\theta_2) \otimes \sum_{i=1}^L \theta_{i+2} \cdot \exp(-\theta_{L+i+2} \cdot x) \quad (100)$$

$$\frac{\partial f}{\partial \theta_j} (j=3..L+2) = P(x-\theta_2) \otimes \exp(-\theta_{L+j} \cdot x) \quad (101)$$

$$\frac{\partial f}{\partial \theta_j} (j=L+3..2L+2) = -P(x-\theta_2) \otimes [\theta_{j-L} \cdot x \cdot \exp(-\theta_j \cdot x)] \quad (102)$$

$$\frac{\partial f}{\partial \theta_j} (j=2L+3) = 1 \quad (103)$$

Andraderivator:

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left[\begin{matrix} j=1 \\ k=2 \end{matrix} \right] = -\dot{P}(x-\theta_2) \quad (104)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left[\begin{matrix} j=2 \\ k=2 \end{matrix} \right] = \theta_1 \cdot \ddot{P}(x-\theta_2) + \ddot{P}(x-\theta_2) \otimes \sum_{i=1}^L \theta_{i+2} \cdot \exp(-\theta_{L+i+2} \cdot x) \quad (105)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left[\begin{matrix} j=2 \\ k=3 \dots L+2 \end{matrix} \right] = -\dot{P}(x-\theta_2) \otimes \exp(-\theta_{L+k} \cdot x) \quad (106)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left[\begin{matrix} j=2 \\ k=L+3 \dots 2L+2 \end{matrix} \right] = \dot{P}(x-\theta_2) \otimes [x \cdot \theta_{k-L} \cdot \exp(-\theta_k \cdot x)] \quad (107)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left[\begin{matrix} j=3 \dots L+2 \\ k=L+j \end{matrix} \right] = -P(x-\theta_2) \otimes [x \cdot \exp(-\theta_k \cdot x)] \quad (108)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} \left[\begin{matrix} j=L+3 \dots 2L+2 \\ k=j \end{matrix} \right] = P(x-\theta_2) \otimes [x^2 \cdot \theta_{j-L} \cdot \exp(-\theta_j \cdot x)] \quad (109)$$

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} = \frac{\partial^2 f}{\partial \theta_k \partial \theta_j} \quad (110)$$

alla övriga andraderivator är noll.

Utskrift från testkörning av programmet PCONEXPA:

```

No of decays (1.. 2)      : 1
No of channels ( 5.. 512) : 128
Stray-light intensity    : 0
Pulse position (channels) : 0
Intensity, decay no 1    : 0.1
Decay rate, decay no 1   : 10
Background                : 20
Add noise to data? (Y/*) : Y
Plot errorbars? (Y/*)   : Y
Pulse FWHM (channels)    : 10
Add noise to pulse? (Y/*) : N

```

Toggle Free/Locked with <shift>+Function key.

```

F1 : Stray-light intensity          0.0000, Locked
F2 : Pulse position                 0.0000E+00, Locked
F3 : Intensity , decay no 1         0.1000, Free
F4 : Decay rate, decay no 1         10.0000, Free
F5 : Background                     20.0000, Free
F10 : Calculate.
<Alt>+ F10 : Preview.
<Shift>+ F10 : G-matrix = not linearized.
<Ctrl>+ F10 : Avoid circular convolution = False.
<Alt>+ F1 : Weight = 1/F.

```

<<shift>+F2>

Toggle Free/Locked with <shift>+Function key.

F1	: Stray-light intensity	0.0000,	Locked
F2	: Pulse position	0.0000E+00,	Free
F3	: Intensity , decay no 1	0.1000,	Free
F4	: Decay rate, decay no 1	10.0000,	Free
F5	: Background	20.0000,	Free
F10	: Calculate.		

<Alt>+ F10 : Preview.
<Shift>+ F10 : G-matrix = not linearized.
<Ctrl>+ F10 : Avoid circular convolution = False.
<Alt>+ F1 : Weight = 1/F.

<F10>

Press Q to quit iteration.

It: 0
Pulse position : 0.00000E+00
Intensity(1) : 0.10000 ,Decay rate(1): 10.000
Background : 20.000

It: 1, STEP: 0.846875E-01
Pulse position : 0.69136E-01
Intensity(1) : 0.10133 ,Decay rate(1): 10.188
Background : 19.001

It: 2, STEP: 0.515620E-01
Pulse position : 0.72790E-01
Intensity(1) : 0.10138 ,Decay rate(1): 10.194
Background : 19.005

It: 3, STEP: 0.252140E-02
Pulse position : 0.72970E-01
Intensity(1) : 0.10138 ,Decay rate(1): 10.195
Background : 19.006

It: 4, STEP: 0.556980E-01
Pulse position : 0.73708E-01
Intensity(1) : 0.10152 ,Decay rate(1): 10.201
Background : 19.873

It: 5, STEP: 0.154279E-01
Pulse position : 0.74810E-01
Intensity(1) : 0.10154 ,Decay rate(1): 10.204
Background : 19.880

It: 6, STEP: 0.553574E-03
Pulse position : 0.74850E-01
Intensity(1) : 0.10154 ,Decay rate(1): 10.204
Background : 19.880

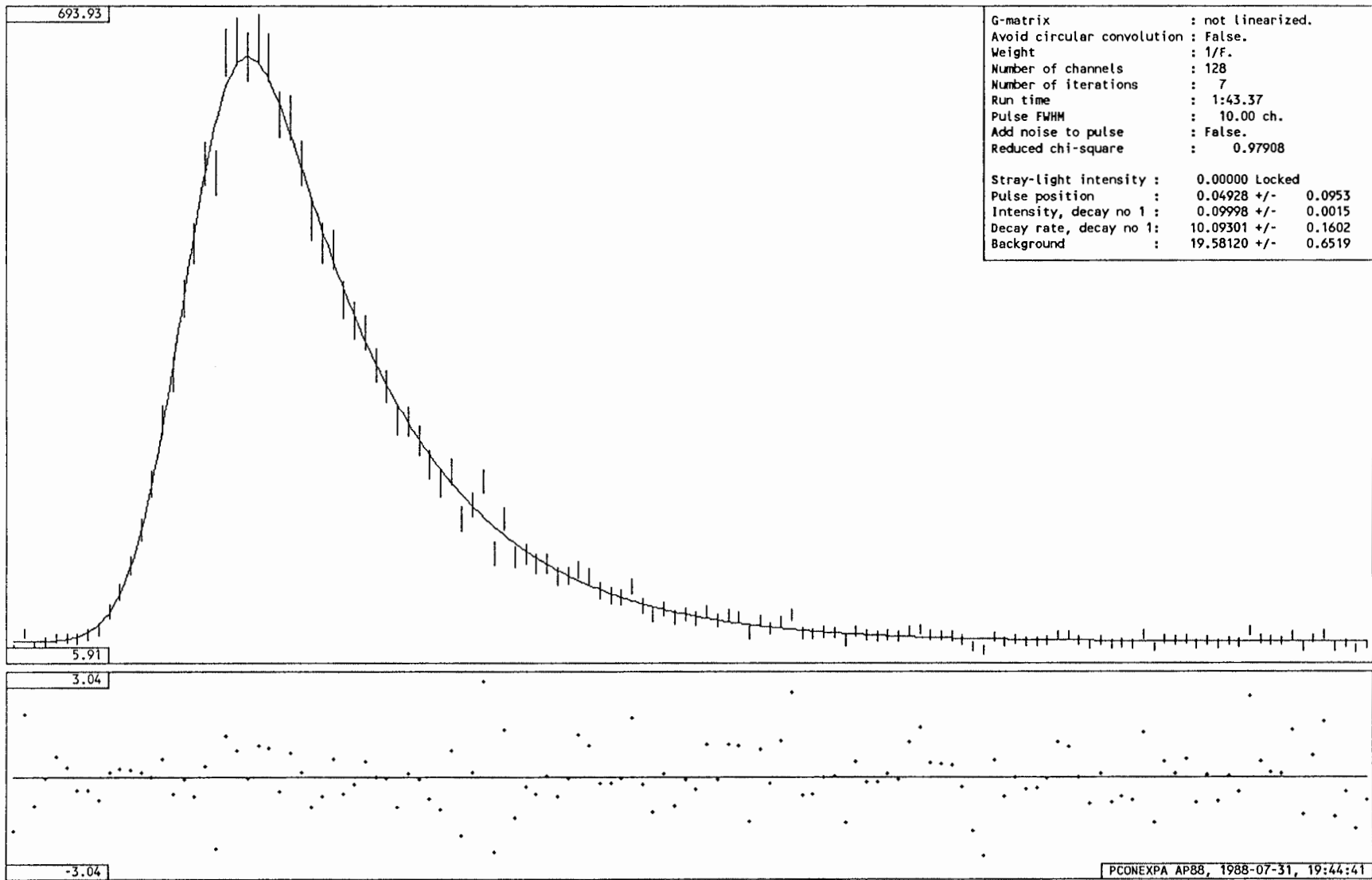
It: 7, STEP: 0.190543E-04
Pulse position : 0.74851E-01
Intensity(1) : 0.10154 ,Decay rate(1): 10.204
Background : 19.880

Press any key to continue. <Return>
Plot on Roland plotter? (Y/*) : N
Plot on HP-Laserjet? (Y/*) : Y

G-matrix : not linearized.
Avoid circular convolution : False.
Weight : 1/F.
Number of channels : 128
Number of iterations : 7
Run time : 1:53.14
Pulse FWHM : 10.00 ch.
Add noise to pulse : False.
Reduced chi-square : 0.96477

Stray-light intensity : 0.00000 Locked
Pulse position : 0.07485 +/- 0.0957
Intensity, decay no 1 : 0.10154 +/- 0.0015
Decay rate, decay no 1 : 10.20361 +/- 0.1609
Background : 19.88001 +/- 0.6513

New iteration? (Y/*) :N
New calculation? (Y/*) :N



Kommentarer till programlistningen:

Rad	Kommentar
1-45	Deklarationer. Användning av vektorer och matriser: P0 - laserpulsens. P - den translaterade laserpulsens. PPRIM - förstaderivatans av laserpulsens. PBIS - andraderivatans av laserpulsens. PCONVE - pulsen faltad med exp. PCONVXE - pulsen faltad med $x \cdot \exp$. PCONVX2E - pulsen faltad med $x^2 \cdot \exp$. PPRIMCONVE - förstaderivatans faltad med exp. PPRIMCONVXE - förstaderivatans faltad med $x \cdot \exp$. PBISCONVE - andraderivatans faltad med exp. FTP0 - Fouriertransformen av laserpulsens. FTE - Fouriertransformen av exp. FTEX - Fouriertransformen av $x \cdot \exp$. FTEX2 - Fouriertransformen av $x^2 \cdot \exp$. S1,S2 - hjälpvektorer för mellanresultat. W1,W2 - hjälpvektorer, sinus, cosinustabeller till fouriertransformrutinen.
47-53	Läs in antal sönderfall.
55-61	Läs in antal kanaler i indata.
63-67	Läs in ströljusintensitet.
69-71	Läs in laserpulsens position.
73-85	Läs in intensitet och sönderfallskonstant för de olika exponentialtermerna.
87-88	Läs in bakgrund.
90-99	Läs in om en approximation till Poisson-brus skall adderas till testkurvan. Om brus skall adderas, fråga om felgränser skall ritas i figurerna.
101	Från början antas att cirkulär faltning inte påverkar resultatet (vilket antagligen är sant om indata är bra).
103-108	Beräkna hur många punkter programmet skall använda vid beräkning av Fouriertransformer/faltningar.
110-123	Startvärden för låsparametrar och programkontrollparametrar.
125-133	Läs in laserpulsens halvvärdesbredd och om brus (normalfördelat) skall adderas till laserpulsens. Laserpulsens är en Gausstopp centrerad i position: $x_0 = \text{antal kanaler} / 7.0$.
135-145	Beräkna laserpulsens.
147-148	Beräkna sinus-cosinustabeller till Fouriertransformrutinen.

- 149-150 Beräkna laserpulsens Fouriertransform.
- 151 Beräkna alla derivator och faltningar av puls och exponentialer samt flytta laserpulsens till vald position. SLOCKED-vektorn indikerar att alla parametrar är låsta till sina startvärden eftersom det medför minimalt räknearbete i CALCCONV-rutinen. I det här läget i programmet behövs ett minimum av faltningar, bara de för beräkning av testkurvan.
- 153-163 Beräkna test(fluorescens)kurvan.
- 165-166 Sudda skärmen.
- 168-211 Skriv ut meny på skärmen (se utskrift från testkörning).
- 213-217 Ta reda på vilken funktionstangent som tryckts ned.
- 219-290 Utför den valda operationen, t.ex. ändra parameter till låst läge, läs in nytt startvärde, rita kurvan på skärmen m.m.
- 293-447 Anpassa med Newtons metod.
- 297-309 Sätt max och minvärde för tillåtna parametervariationer.
- 311-312 Tidtagning.
- 314-330 Skriv ut parametervärden för iteration nr:0
- 332-333 Programet startar med den lineariserade G-matrisen och med vikten $1/y_i$.
- 335-338 Nollställ g -vektorn och G-matrisen samt beräkna alla nödvändiga faltningar, translationer och derivator av laserpulsens.
- 339-367 Summera \underline{g} och G enl. ekv. (37),(42) och (46).
- 369-373 Om någon parameter är låst skall motsvarande diagonalelement sättas till 1 för att ekvationssystemet skall bli lösbart.
- 375-379 Invertera G-matrisen.
- 381-382 Beräkna $\Delta\underline{\theta}$ enl. ekv. (43).
- 384-390 Maximera svinget i parametervärden.
- 392 Addera $\Delta\underline{\theta}^m$ till $\underline{\theta}^m$ enl. ekv. (39).
- 394-401 Beräkna steglängden i $\underline{\theta}$, $STEP = \sum |\Delta\theta_i / \theta_i|$ om $|\Delta\theta_i| < |\theta_i|$.
- 403-416 Skriv ut parametervärden och steglängd för iteration j.
- 418-434 Avgör om itereringen skall avbrytas eller om programmet skall byta till den fullständiga G-matrisen och till $1/f_i$ -vikt istället för $1/y_i$ -vikt.
- 436-443 Avbryt itereringen om Q-knappen tryckts ned på tangentbordet.
- 445-447 Slut på iterations slingan. Om programmet hamnar på rad 446 har maximalt antal iterationer (MAXNOOFITERATIONS) använts utan att minpunkten uppnåtts.

- 449-450 Beräkna faltningar med hjälp av de senaste parameterskattningarna. Multiplicera G-matrisen med -2 för att få en skattning av kovariansmatrisen för $\underline{\theta}$ (45).
- 452-453 Spara exekveringstiden.
- 455-460 Stanna programmet innan det ritar på skärmen.
- 462-464 Beräkna vektorn SIGMA, $\sigma_i = \sqrt{f_i}$ om data är Poissonfördelade.
- 466-467 Rita anpassning och residualvektor på skärmen.
- 469-475 Rita anpassning och residualvektor på Roland pen-plotter.
- 477-489 Beräkna den reducerade χ^2 parametern (47).
- 491-670 Rita anpassning och residualvektor samt skriv ut diverse indata och parameterdata på HP laserskrivare.
- 672-792 Skriv ut parametervärden med felgränser på skärmen. Skriv även ut diverse indata: antal kanaler, vikt m.m.
- 794-798 Fråga om anpassning till nuvarande indata skall köras igen, t.ex. om startvärden behöver ändras eller om parametrar skall släppas fria eller låsas i anpassningen.
- 800-808 Avsluta programmet eller kör från start igen.
- 810-844 Rutinen beräknar funktionsvärdet $f(\underline{\theta}; x)$ (98) med hjälp av de faltningar som beräknats i CALCCONV-rutinen. Eftersom dessa är diskreta punktföljder (som inte gör sig särskilt bra på figurerna) utför rutinen linjär interpolation mellan de diskreta punkterna.
- 846-886 Rutinen beräknar förstaderivatorna av f (99)..(103).
- 888-960 Rutinen beräknar andraderivatorna av f (104)..(110).
- 962-1171 Rutinen beräknar all nödvändiga faltningar och pulstranslationer/derivationer.
- 994-1007 Om pulsens position är noll får P-vektorn samma värden som den ursprungliga laserpulsens P_0 , annars beräknas den translaterade pulsen med hjälp av rutinen MOVEDIFF och placeras i P-vektorn.
- 1009-1028 Om pulsens position inte är låst måste pulsens förstaderivata beräknas. Om G-matrisen inte är lineariserad behövs även pulsens andraderivata.
- 1030-1062 Beräkna exponentialvektorerna $\exp(-\theta x_i)$, $x_i \cdot \exp(-\theta x_i)$ och $x_i^2 \cdot \exp(-\theta x_i)$.
- 1064-1074 Beräkna Fouriertransformerna av exponentialvektorerna.
- 1076-1114 Beräkna faltningar mellan laserpuls och exponentialer.
- 1116-1146 Beräkna faltningar mellan laserpulsens förstaderivata och exponentialer.

- 1148-1168 Beräkna faltningar mellan laserpulsens andraderivata och exponentialer.
- 1173-1208 Rutinen beräknar en translaterad och deriverad kurva med hjälp av kurvans Fouriertransform.
- 1183-1189 Translation DC kanaler blir i transformplanet en multiplikation med fasfaktorn $\exp(-i\omega \cdot DC)$.
- 1191-1203 Derivation i tidsplanet blir i transformplanet en multiplikation med $i\omega$, andraderivatan blir multiplikation med $(i\omega)^2$ osv. För att bruset (högfrekvent) inte skall bli för dominerande i de deriverade kurvorna avbryts multiplikationen vid frekvensen $\omega_{\max}/20$ och därefter multipliceras kurvan med konstant.
- 1205-1206 Inverstransformera till tidsplanet.

```

1 PROGRAM PCONEXPA
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5
6 INTEGER N,NHALF,L,MAXNOOFITERATIONS,MAXNOOFDECAYS
7
8 PARAMETER ( N=512, NHALF=N/2, MAXNOOFITERATIONS=100 )
9 PARAMETER ( MAXNOOFDECAYS=2, L=2*MAXNOOFDECAYS+3 )
10
11 REAL Y(1:N),X(1:N),DX,DET,NORMAL,THETAEST(1:L)
12 REAL RESIDUE(1:N),SIGMA(1:N),F,THETA(1:L),JX,JY,JXL,JYL,DJY
13 REAL GMATRIX(1:L,1:L),GVECTOR(1:L),DELTATHETA(1:L)
14 REAL SLASK,DFDTH,D2FDTH2,THETAMIN(1:L),THETAMAX(1:L)
15 REAL SL1,SL2,SL3,SL4,SL5,TIME,STEP,REDCHISQUARED
16 REAL PO(1:N),P(1:N),PPRIM(1:N),PBIS(1:N)
17 REAL PCONVE(1:N,1:MAXNOOFDECAYS)
18 REAL PCONVXE(1:N,1:MAXNOOFDECAYS)
19 REAL PCONVX2E(1:N,1:MAXNOOFDECAYS)
20 REAL PPRIMCONVE(1:N,1:MAXNOOFDECAYS)
21 REAL PPRIMCONVXE(1:N,1:MAXNOOFDECAYS)
22 REAL PBISCONVE(1:N,1:MAXNOOFDECAYS),PULSEFWHM
23
24 COMPLEX FTPO(1:N),FTE(1:N,1:MAXNOOFDECAYS)
25 COMPLEX FTEX(1:N,1:MAXNOOFDECAYS),FTEX2(1:N,1:MAXNOOFDECAYS)
26 COMPLEX S1(1:N),S2(1:N),W1(1:NHALF),W2(1:NHALF)
27
28 INTEGER R,K,I,NCH,NCH2,DNCH2,NOOFITERATIONS,J,NOOFDECAYS,LR
29 INTEGER NOOFVARIABLEPAR
30
31 INTEGER*2 HO,MI,SE,HN,ANS,KBDCHK,KBDINC,YEAR,MONTH,DAY
32
33 CHARACTER SV*1,LOCKEDORFREE(1:L)*6,JETTXT*50
34
35 LOGICAL SING,ERRORBAR,LINEARIZED,LOCKED(1:L),ADDNOISE,INV
36 LOGICAL LOCKTOLINEARIZED,LOCKWEIGHT,AVOIDCIRCCONV,SLLOCKED(1:L)
37 LOGICAL ONEOVERFWIGHT,ADDNOISETOPULSE
38
39 EXTERNAL F
40
41 COMMON /BLOCK1/ P,PCONVE
42 COMMON /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
43 COMMON /BLOCK3/ PCONVX2E,PPRIMCONVXE,PBIS,PBISCONVE
44 COMMON /BLOCK4/ PO,FTPO,FTE,FTEX,FTEX2,S1,S2,W1,W2
45 COMMON /BLOCK5/ NCH,NCH2,DNCH2,NOOFDECAYS,DX
46
47 10 WRITE(*,1000) 'No of decays (1..',MAXNOOFDECAYS,'):'
48 1000 FORMAT(' ',A,12,A)
49 READ(*,*,ERR=10) NOOFDECAYS
50 IF (NOOFDECAYS.LT.1.OR.NOOFDECAYS.GT.MAXNOOFDECAYS) THEN
51 GOTO 10
52 ENDIF
53 LR=2*NOOFDECAYS+3
54
55 20 WRITE(*,1100) 'No of channels (' ,LR,'..' ,N,'):'
56 1100 FORMAT(' ',A,12,A,14,A)
57 READ(*,*,ERR=20) NCH
    
```

```

58 IF (NCH.LT.LR.OR.NCH.GT.N) THEN
59 GOTO 20
60 ENDIF
61 DX=1.0/(NCH-1)
62
63 30 WRITE(*,*) 'Stray-light intensity:'
64 READ(*,*,ERR=30) THETA(1)
65 IF (THETA(1).LT.0.0) THEN
66 GOTO 30
67 ENDIF
68
69 40 WRITE(*,*) 'Pulse position (channels):'
70 READ(*,*,ERR=40) THETA(2)
71 THETA(2)=THETA(2)*DX
72
73 DO 70 I=1,NOOFDECAYS
74 50 WRITE(*,1200) 'Intensity, decay no ',I,': '
75 1200 FORMAT(' ',A,11,A)
76 READ(*,*,ERR=50) THETA(I+2)
77 IF (THETA(I+2).LE.0.0) THEN
78 GOTO 50
79 ENDIF
80 60 WRITE(*,1200) 'Decay rate, decay no ',I,': '
81 READ(*,*,ERR=60) THETA(I+NOOFDECAYS+2)
82 IF (THETA(I+NOOFDECAYS+2).LE.0.0) THEN
83 GOTO 60
84 ENDIF
85 70 CONTINUE
86
87 80 WRITE(*,*) 'Background: '
88 READ(*,*,ERR=80) THETA(LR)
89
90 WRITE(*,*) 'Add noise to data? (Y/*): '
91 READ(*,1300) SV
92 1300 FORMAT(A)
93 ADDNOISE=SV.EQ.'Y'.OR.SV.EQ.'y'
94 ERRORBAR=ADDNOISE
95 IF (ADDNOISE) THEN
96 WRITE(*,*) 'Plot errorbars? (Y/*): '
97 READ(*,1300) SV
98 ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'
99 ENDIF
100
101 AVOIDCIRCCONV=.FALSE.
102
103 NCH2 = LOG(NCH-0.5)/LOG(2.0)
104 NCH2 = 2**(NCH2+1)
105 DNCH2 = 2*NCH2
106 IF (.NOT.AVOIDCIRCCONV) THEN
107 DNCH2=NCH2
108 ENDIF
109
110 DO 90 I=1,LR
111 THETAEST(I) = THETA(I)
112 LOCKED(I) = .FALSE.
113 SLLOCKED(I) = .TRUE.
114 LOCKEDORFREE(I) = 'Free '
115 90 CONTINUE
116
117 LOCKED(1) = .TRUE.
    
```

```

118 LOCKED(2) = .TRUE.
119 LOCKEDORFREE(1) = 'Locked'
120 LOCKEDORFREE(2) = 'Locked'
121 LOCKTOLINEARIZED = .FALSE.
122 LINEARIZED = .TRUE.
123 ONEOVERFWEIGHT = .TRUE.
124
125 100 WRITE(*,*) 'Pulse FWHM (channels):'
126 READ(*,*,ERR=100) PULSEFWHM
127 IF (PULSEFWHM.LT.0.1.OR.PULSEFWHM.GT.NCH*0.5) THEN
128     GOTO 100
129 ENDIF
130
131 WRITE(*,*) 'Add noise to pulse? (Y/*):'
132 READ(*,1300) SV
133 ADDNOISETOPULSE=SV.EQ.'Y'.OR.SV.EQ.'y'
134
135 DO 110 I=1,NCH
136     PO(I)=EXP(-MIN((LOG(16.0)/PULSEFWHM**2)*(I-NCH/7.0)**2,87.0))
137     PO(I)=1000.0*PO(I)
138     IF (ADDNOISETOPULSE) THEN
139         PO(I)=PO(I)+NORMAL(0.0,10.0)
140     ENDIF
141     FTPO(I)=CMPLX(PO(I),0.0)
142 110 CONTINUE
143 DO 120 I=NCH+1,NCH2
144     FTPO(I)=CMPLX(0.0,0.0)
145 120 CONTINUE
146
147 CALL CALCW(W1,NCH2)
148 CALL CALCW(W2,DNCH2)
149 INV=.FALSE.
150 CALL FFT(FTPO,W1,NCH2,INV)
151 CALL CALCCONV(THETA,SLLOCKED,LINEARIZED)
152
153 DO 130 I=1,NCH
154     X(I)=(I-1)*DX
155     SLASK=F(THETA,NOOFDECAYS,X(I))
156     IF (ADDNOISE) THEN
157         Y(I)=NORMAL(SLASK,SQRT(ABS(SLASK)))
158         SIGMA(I)=SQRT(ABS(Y(I)))
159     ELSE
160         Y(I)=SLASK
161         SIGMA(I)=0.0
162     ENDIF
163 130 CONTINUE
164
165 140 CALL GMODE
166 CALL TMODE
167
168 WRITE(*,1400) 'Toggle Free/Locked with <shift>+Function key.'
169 1400 FORMAT(////,T10,A,/)
170
171 WRITE(*,1500) THETAEST(1),LOCKEDORFREE(1)
172 1500 FORMAT(T10,'F1 : Stray-light intensity',T45,F12.4,',',T60,A)
173
174 WRITE(*,1600) THETAEST(2)/DX,LOCKEDORFREE(2)
175 1600 FORMAT(T10,'F2 : Pulse position',T45,G12.4,',',T60,A)
176
177 DO 150 I=1,NOOFDECAYS

```

```

178 WRITE(*,1700) I+2,I,THETAEST(I+2),LOCKEDORFREE(I+2)
179 1700 FORMAT(T10,'F',I1,' : Intensity , decay no ',
180 f      I1,T45,F12.4,',',T60,A)
181 150 CONTINUE
182 DO 160 I=1,NOOFDECAYS
183     WRITE(*,1800) I+NOOFDECAYS+2,I,THETAEST(I+NOOFDECAYS+2),
184     f      LOCKEDORFREE(I+NOOFDECAYS+2)
185 1800 FORMAT(T10,'F',I1,' : Decay rate, decay no ',
186 f      I1,T45,F12.4,',',T60,A)
187 160 CONTINUE
188 WRITE(*,1900) LR,THETAEST(LR),LOCKEDORFREE(LR)
189 1900 FORMAT(T10,'F',I1,' : Background ',T45,F12.4,',',T60,A)
190 WRITE(*,2000) 'F10: Calculate.'
191 2000 FORMAT(T10,A)
192 WRITE(*,2100) '<Alt>+F10: Preview.'
193 2100 FORMAT(T4,A)
194 IF (LOCKTOLINEARIZED) THEN
195     WRITE(*,2200) '<Shift>+F10: G-matrix = linearized.'
196 ELSE
197     WRITE(*,2200) '<Shift>+F10: G-matrix = not linearized.'
198 ENDIF
199 2200 FORMAT(T2,A)
200 IF (AVOIDCIRCONV) THEN
201     WRITE(*,2300) '<Ctrl>+F10: Avoid circular convolution = True.'
202 ELSE
203     WRITE(*,2300) '<Ctrl>+F10: Avoid circular convolution = False.'
204 ENDIF
205 IF (ONEOVERFWEIGHT) THEN
206     WRITE(*,2100) '<Alt>+F1 : Weight = 1/F.'
207 ELSE
208     WRITE(*,2100) '<Alt>+F1 : Weight = 1.'
209 ENDIF
210 2300 FORMAT(T3,A)
211 WRITE(*,2300) ' '
212
213 170 ANS=KBDCHK()
214 IF (ANS.NE.0) THEN
215     GOTO 170
216 ENDIF
217 ANS=KBDINC()
218
219 IF (ANS.GE.1084.AND.ANS.LE.1083+LR) THEN
220     I=ANS-1083
221     LOCKED(I)=.NOT.LOCKED(I)
222     LOCKEDORFREE(I)='Free '
223     IF (LOCKED(I)) THEN
224         LOCKEDORFREE(I)='Locked'
225     ENDIF
226     GOTO 140
227 ELSEIF (ANS.EQ.1113) THEN
228     WRITE(*,2000) 'Preview.'
229     CALL CALCW(W2,DNCH2)
230     CALL CALCCONV(THETAEST,SLLOCKED,LINEARIZED)
231     CALL SCRIPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
232     f      THETAEST,NOOFDECAYS,ERRORBAR)
233     GOTO 140
234 ELSEIF (ANS.EQ.1068) THEN
235     Calculate.
236     CALL CALCW(W2,DNCH2)
237     GOTO 230

```

```

238 ELSEIF (ANS.EQ.1093) THEN
239 LOCKTOLINEARIZED=.NOT.LOCKTOLINEARIZED
240 GOTO 140
241 ELSEIF (ANS.EQ.1103) THEN
242 AVOIDCIRCCONV=.NOT.AVOIDCIRCCONV
243 IF (NCH.GT.NHALF) THEN
244 AVOIDCIRCCONV=.FALSE.
245 ENDIF
246 DNCH2=NCH2
247 IF (AVOIDCIRCCONV) THEN
248 DNCH2=2*NCH2
249 ENDIF
250 GOTO 140
251 ELSEIF (ANS.EQ.1104) THEN
252 ONEOVERFWEIGHT=.NOT.ONEOVERFWEIGHT
253 GOTO 140
254 ELSEIF (ANS.GE.1059.AND.ANS.LE.1058+LR) THEN
255 I=ANS-1058
256 IF (I.EQ.1) THEN
257 180 WRITE(*,2400) 'New stray-light intensity:'
258 2400 FORMAT(T10,A,I1,A)
259 READ(*,*,ERR=180) THETAEST(1)
260 IF (THETAEST(1).LT.0.0) THEN
261 GOTO 180
262 ENDIF
263 GOTO 140
264 ELSEIF (I.EQ.2) THEN
265 190 WRITE(*,2400) 'New pulse position (channels):'
266 READ(*,*,ERR=190) THETAEST(2)
267 THETAEST(2)=THETAEST(2)*DX
268 GOTO 140
269 ELSEIF (I.GT.2.AND.I.LE.NOOFDECAYS+2) THEN
270 200 WRITE(*,2400) 'New intensity, decay no ',I-2,': '
271 READ(*,*,ERR=200) THETAEST(1)
272 IF (THETAEST(1).LE.0.0) THEN
273 GOTO 200
274 ENDIF
275 GOTO 140
276 ELSEIF (I.GT.NOOFDECAYS+2.AND.I.LT.LR) THEN
277 210 WRITE(*,2400) 'New decay rate, decay no ',I-NOOFDECAYS-2,': '
278 READ(*,*,ERR=210) THETAEST(1)
279 IF (THETAEST(1).LE.0.0) THEN
280 GOTO 210
281 ENDIF
282 GOTO 140
283 ELSEIF (I.EQ.LR) THEN
284 220 WRITE(*,2400) 'New background: '
285 READ(*,*,ERR=220) THETAEST(LR)
286 GOTO 140
287 ENDIF
288 GOTO 170
289 ENDIF
290 GOTO 170
291
292 C *** Calculation ***
293 230 CONTINUE
294
295 WRITE(*,*) 'Press Q to quit iteration.'
296 WRITE(*,2800)
297 DO 240 I=1,2*NOOFDECAYS+2

298 THETAMIN(I)=THETAEST(I)/10.0
299 THETAMAX(I)=THETAEST(I)*10.0
300 240 CONTINUE
301 IF (THETAEST(1).LT.1.0) THEN
302 THETAMAX(1)=1.0
303 THETAMIN(1)=-1.0
304 ENDIF
305 THETAMAX(2)=THETAEST(2)+0.3*NCH*DX
306 THETAMIN(2)=THETAEST(2)-0.3*NCH*DX
307
308 THETAMAX(LR)=ABS(THETAEST(LR)*10.0)
309 THETAMIN(LR)=-ABS(THETAEST(LR)*10.0)
310
311 CALL GETTIM(HO,MI,SE,HN)
312 TIME=HO*3600.0+MI*60.0+SE+HN*0.01
313
314 WRITE(*,2500) 'It:',0
315 2500 FORMAT(' ',49(' '),/,',',A,14,A,G13.6)
316 IF (.NOT.LOCKED(1)) THEN
317 WRITE(*,2600) 'Stray-light int.:',THETAEST(1)
318 ENDIF
319 IF (.NOT.LOCKED(2)) THEN
320 WRITE(*,2600) 'Pulse position :',THETAEST(2)/DX
321 ENDIF
322 2600 FORMAT(' ',A,G12.5)
323 DO 250 R=1,NOOFDECAYS
324 WRITE(*,2700) 'Intensity(',R,') :',THETAEST(R+2),
325 f ' ',Decay rate(',R,'):',THETAEST(R+NOOFDECAYS+2)
326 2700 FORMAT(' ',A,I1,A,G12.5,A,I1,A,G12.5)
327 250 CONTINUE
328 IF (.NOT.LOCKED(LR)) THEN
329 WRITE(*,2600) 'Background :',THETAEST(LR)
330 ENDIF
331
332 LINEARIZED=.TRUE.
333 LOCKWEIGHT=.TRUE.
334
335 DO 330 J=1,MAXNOOFITERATIONS
336 CALL ZEROMAT(GMATRIX,LR,LR,L,L)
337 CALL ZEROMAT(GVECTOR,LR,1,L,1)
338 CALL CALCCONV(THETAEST,LOCKED,LINERIZED)
339 DO 280 I=1,NCH
340 IF (ONEOVERFWEIGHT) THEN
341 IF (LOCKWEIGHT) THEN
342 SL1=-2.0/MAX(Y(I),1.0)
343 ELSE
344 SL1=-2.0/F(THETAEST,NOOFDECAYS,X(I))
345 ENDIF
346 ELSE
347 SL1=-2.0
348 ENDIF
349 SL2=Y(I)-F(THETAEST,NOOFDECAYS,X(I))
350 DO 270 R=1,LR
351 IF (.NOT.LOCKED(R)) THEN
352 SL3=DFDTH(R,THETAEST,NOOFDECAYS,X(I))
353 GVECTOR(R)=GVECTOR(R)+SL1*SL2*SL3
354 DO 260 K=R,LR
355 IF (.NOT.LOCKED(K)) THEN
356 SL4=DFDTH(K,THETAEST,NOOFDECAYS,X(I))
357 GMATRIX(R,K)=GMATRIX(R,K)+SL1*(-SL3*SL4)

```

```

358         IF (.NOT.LINEARIZED) THEN
359             SL5=D2FDTH2(R,K,THETAEST,NOOFDECAYS,X(1))
360             GMATRIX(R,K)=GMATRIX(R,K)+SL1*SL2*SL5
361         ENDIF
362         GMATRIX(K,R)=GMATRIX(R,K)
363     ENDIF
364 260     CONTINUE
365     ENDIF
366 270     CONTINUE
367 280     CONTINUE
368
369     DO 290 R=1,LR
370         IF (LOCKED(R)) THEN
371             GMATRIX(R,R)=1.0
372         ENDIF
373 290     CONTINUE
374
375     CALL INVMAT(GMATRIX,LR,DET,SING,L,L)
376     IF (SING) THEN
377         WRITE(*,*) 'Sing. G-matrix.'
378         GOTO 450
379     ENDIF
380
381     CALL MULTCMAT(GMATRIX,-1.0,LR,LR,L,L)
382     CALL MULTMAT(GMATRIX,GVECTOR,DELTATHETA,LR,LR,1,L,L,1,L,1)
383
384     DO 300 I=1,2*NOOFDECAYS+3
385         IF (THETAEST(I)+DELTATHETA(I).LT.THETAMIN(I)) THEN
386             DELTATHETA(I)=THETAMIN(I)-THETAEST(I)
387         ELSEIF (THETAEST(I)+DELTATHETA(I).GT.THETAMAX(I)) THEN
388             DELTATHETA(I)=THETAMAX(I)-THETAEST(I)
389         ENDIF
390 300     CONTINUE
391
392     CALL ADDMAT(THETAEST,DELTATHETA,THETAEST,LR,1,L,1,L,1,L,1)
393
394     STEP=0.0
395     DO 310 I=1,LR
396         IF (ABS(THETAEST(I)).GT.ABS(DELTATHETA(I))) THEN
397             STEP=STEP+ABS(DELTATHETA(I)/THETAEST(I))
398         ELSE
399             STEP=STEP+ABS(DELTATHETA(I))
400         ENDIF
401 310     CONTINUE
402
403     WRITE(*,2500) 'It:',J,', STEP:',STEP
404     IF (.NOT.LOCKED(1)) THEN
405         WRITE(*,2600) 'Stray-light int.:',THETAEST(1)
406     ENDIF
407     IF (.NOT.LOCKED(2)) THEN
408         WRITE(*,2600) 'Pulse position  :',THETAEST(2)/DX
409     ENDIF
410     DO 320 R=1,NOOFDECAYS
411         WRITE(*,2700) 'Intensity('',R,'') :',THETAEST(R+2),
412             ' ,Decay rate('',R,''):',THETAEST(R+NOOFDECAYS+2)
413 320     CONTINUE
414     IF (.NOT.LOCKED(LR)) THEN
415         WRITE(*,2600) 'Background  :',THETAEST(LR)
416     ENDIF
417

```

```

418     IF (STEP.LE.0.5E-4.AND.(.NOT.LINEARIZED)) THEN
419         NOOFITERATIONS=J
420         GOTO 340
421     ENDIF
422     IF (STEP.LE.0.5E-2.AND.LINEARIZED) THEN
423         LINEARIZED=.FALSE.
424     ENDIF
425     IF (LOCKTOLINEARIZED) THEN
426         LINEARIZED=.TRUE.
427     ENDIF
428     IF (STEP.LE.0.5E-4.AND.LOCKTOLINEARIZED) THEN
429         NOOFITERATIONS=J
430         GOTO 340
431     ENDIF
432     IF (LOCKWEIGHT.AND.STEP.LE.0.5E-2) THEN
433         LOCKWEIGHT=.FALSE.
434     ENDIF
435
436     ANS=KBDCHK()
437     IF (ANS.NE.0) THEN
438         ANS=KBDINC()
439         IF (CHAR(ANS).EQ.'q'.OR.CHAR(ANS).EQ.'q') THEN
440             NOOFITERATIONS=J
441             GOTO 340
442         ENDIF
443     ENDIF
444
445 330     CONTINUE
446     NOOFITERATIONS=MAXNOOFITERATIONS
447 340     CONTINUE
448
449     CALL CALCCONV(THETAEST,SLLOCKED,.TRUE.)
450     CALL MULTCMAT(GMATRIX,-2.0,LR,LR,L,L)
451
452     CALL GETTIM(HO,MI,SE,HN)
453     TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
454
455     WRITE(*,*) 'Press any key to continue.'
456 350     ANS=KBDCHK()
457     IF (ANS.EQ.0) THEN
458         GOTO 350
459     ENDIF
460     ANS=KBDINC()
461
462     DO 360 I=1,NCH
463         SIGMA(I)=SQRT(ABS(F(THETAEST,NOOFDECAYS,X(I))))
464 360     CONTINUE
465
466     CALL SCRPLLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
467         f NOOFDECAYS,ERRORBAR)
468
469 370     WRITE(*,*) 'Plot on Roland plotter? (Y/*): '
470     READ(*,1300) SV
471     IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
472         CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
473         f NOOFDECAYS,ERRORBAR)
474     GOTO 370
475     ENDIF
476
477     REDCHISQUARED=0.0

```

```

478 DO 380 I=1,NCH
479   REDCHISQUARED=REDCHISQUARED+RESIDUE(I)**2
480 380 CONTINUE
481   NOOFVARIABLEPAR=0
482   DO 390 I=1,LR
483     IF (.NOT.LOCKED(I)) THEN
484       NOOFVARIABLEPAR=NOOFVARIABLEPAR+1
485     ENDIF
486 390 CONTINUE
487   IF (NCH.NE.NOOFVARIABLEPAR) THEN
488     REDCHISQUARED=REDCHISQUARED/(NCH-NOOFVARIABLEPAR)
489   ENDIF
490
491 400 WRITE(*,*) 'Plot on HP-Laserjet? (Y/*): '
492   READ(*,1300) SV
493   IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
494     CALL HPSTART('L')
495     CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
496   f      NOOFDECAYS,ERRORBAR)
497
498     JX=800
499     JXL=1119-JX
500     JYL=(13+1+NOOFDECAYS*2)*13
501     JY=719-JYL
502     CALL HPBOX(JX,JY,JXL,JYL)
503     JX=800
504     JY=719-16
505     DJY=-13
506
507     IF (LOCKTOLINEARIZED) THEN
508       WRITE(JETTXT,2900)
509   f      'G-matrix           : linearized.'
510     ELSE
511       WRITE(JETTXT,2900)
512   f      'G-matrix           : not linearized.'
513     ENDIF
514     CALL HPPRINT(JX,JY,50,JETTXT)
515     JY=JY+DJY
516
517     IF (AVOIDCIRCONV) THEN
518       WRITE(JETTXT,2900)
519   f      'Avoid circular convolution : True.'
520     ELSE
521       WRITE(JETTXT,2900)
522   f      'Avoid circular convolution : False.'
523     ENDIF
524     CALL HPPRINT(JX,JY,50,JETTXT)
525     JY=JY+DJY
526
527     IF (ONEOVERFWEIGHT) THEN
528       WRITE(JETTXT,2900)
529   f      'Weight           : 1/F.'
530     ELSE
531       WRITE(JETTXT,2900)
532   f      'Weight           : 1.'
533     ENDIF
534     CALL HPPRINT(JX,JY,50,JETTXT)
535     JY=JY+DJY
536
537     WRITE(JETTXT,2900)

```

```

538   f      'Number of channels           :',NCH
539   CALL HPPRINT(JX,JY,50,JETTXT)
540   JY=JY+DJY
541
542   WRITE(JETTXT,2900)
543   f      'Number of iterations           :',NOOFITERATIONS
544   CALL HPPRINT(JX,JY,50,JETTXT)
545   JY=JY+DJY
546
547   WRITE(JETTXT,3000)
548   f      'Run time           :',INT(TIME/60.0),':',
549   f      TIME-60.0*INT(TIME/60.0)
550   CALL HPPRINT(JX,JY,50,JETTXT)
551   JY=JY+DJY
552
553   WRITE(JETTXT,3100)
554   f      'Pulse FWHM           :',PULSEFWHM,' ch.'
555   CALL HPPRINT(JX,JY,50,JETTXT)
556   JY=JY+DJY
557
558   IF (ADDNOISETOPULSE) THEN
559     WRITE(JETTXT,2900)
560   f      'Add noise to pulse           : True.'
561   ELSE
562     WRITE(JETTXT,2900)
563   f      'Add noise to pulse           : False.'
564   ENDIF
565   CALL HPPRINT(JX,JY,50,JETTXT)
566   JY=JY+DJY
567
568   WRITE(JETTXT,3200)
569   f      'Reduced chi-square           :',REDCHISQUARED
570   CALL HPPRINT(JX,JY,50,JETTXT)
571   JY=JY+DJY
572
573   WRITE(JETTXT,1300) ' '
574   CALL HPPRINT(JX,JY,50,JETTXT)
575   JY=JY+DJY
576
577   IF ((.NOT.LOCKED(1)).AND.ADDNOISE.AND.
578   f      (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
579     WRITE(JETTXT,3200) 'Stray-light intensity :',
580   f      THETAEST(1),' +/- ',SQRT(ABS(GMATRIX(1,1)))
581   ELSEIF (LOCKED(1)) THEN
582     WRITE(JETTXT,3200) 'Stray-light intensity :',
583   f      THETAEST(1),' Locked'
584   ELSEIF ((.NOT.LOCKED(1)).AND.
585   f      ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
586   f      (.NOT.ONEOVERFWEIGHT))) THEN
587     WRITE(JETTXT,3200) 'Stray-light intensity :',
588   f      THETAEST(1)
589   ENDIF
590   CALL HPPRINT(JX,JY,50,JETTXT)
591   JY=JY+DJY
592
593   IF ((.NOT.LOCKED(2)).AND.ADDNOISE.AND.
594   f      (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
595     WRITE(JETTXT,3200) 'Pulse position           :',
596   f      THETAEST(2)/DX,' +/- ',SQRT(ABS(GMATRIX(2,2)))/DX
597   ELSEIF (LOCKED(2)) THEN

```

```

598      WRITE(JETTXT,3200) 'Pulse position      :',
599      f      THETAEST(2)/DX, ' Locked'
600      ELSEIF ((.NOT.LOCKED(2)).AND.
601      f      ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
602      f      (.NOT.ONEOVERFWEIGHT))) THEN
603      WRITE(JETTXT,3200) 'Pulse position      :',
604      f      THETAEST(2)/DX
605      ENDIF
606      CALL HPPRINT(JX,JY,50,JETTXT)
607      JY=JY+DJY
608
609      DO 410 I=1,NOOFDECAYS
610      IF ((.NOT.LOCKED(I+2)).AND.ADDNOISE.AND.
611      f      (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
612      WRITE(JETTXT,3300) 'Intensity, decay no ',I,':',
613      f      THETAEST(I+2),'+/- ',SQRT(ABS(GMATRIX(I+2,I+2)))
614      ELSEIF (LOCKED(I+2)) THEN
615      WRITE(JETTXT,3300) 'Intensity, decay no ',I,':',
616      f      THETAEST(I+2), ' Locked'
617      ELSEIF ((.NOT.LOCKED(I+2)).AND.
618      f      ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
619      f      (.NOT.ONEOVERFWEIGHT))) THEN
620      WRITE(JETTXT,3300) 'Intensity, decay no ',I,':',
621      f      THETAEST(I+2)
622      ENDIF
623      CALL HPPRINT(JX,JY,50,JETTXT)
624      JY=JY+DJY
625 410  CONTINUE
626
627      DO 420 I=1,NOOFDECAYS
628      IF ((.NOT.LOCKED(I+2+NOOFDECAYS)).AND.ADDNOISE.AND.
629      f      (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
630      WRITE(JETTXT,3300) 'Decay rate, decay no ',I,':',
631      f      THETAEST(I+2+NOOFDECAYS),'+/- ',
632      f      SQRT(ABS(GMATRIX(I+2+NOOFDECAYS,I+2+NOOFDECAYS)))
633      ELSEIF (LOCKED(I+2+NOOFDECAYS)) THEN
634      WRITE(JETTXT,3300) 'Decay rate, decay no ',I,':',
635      f      THETAEST(I+2+NOOFDECAYS), ' Locked'
636      ELSEIF ((.NOT.LOCKED(I+2+NOOFDECAYS)).AND.
637      f      ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
638      f      (.NOT.ONEOVERFWEIGHT))) THEN
639      WRITE(JETTXT,3300) 'Decay rate, decay no ',I,':',
640      f      THETAEST(I+2+NOOFDECAYS)
641      ENDIF
642      CALL HPPRINT(JX,JY,50,JETTXT)
643      JY=JY+DJY
644 420  CONTINUE
645
646      IF ((.NOT.LOCKED(LR)).AND.ADDNOISE.AND.
647      f      (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
648      WRITE(JETTXT,3200) 'Background          :',
649      f      THETAEST(LR),'+/- ',SQRT(ABS(GMATRIX(LR,LR)))
650      ELSEIF (LOCKED(LR)) THEN
651      WRITE(JETTXT,3200) 'Background          :',
652      f      THETAEST(LR), ' Locked'
653      ELSEIF ((.NOT.LOCKED(LR)).AND.
654      f      ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
655      f      (.NOT.ONEOVERFWEIGHT))) THEN
656      WRITE(JETTXT,3200) 'Background          :',
657      f      THETAEST(LR)

```

```

658      ENDIF
659      CALL HPPRINT(JX,JY,50,JETTXT)
660
661      CALL GETTIM(HO,MI,SE,HN)
662      CALL GETDAT(YEAR,MONTH,DAY)
663      WRITE(JETTXT,3500)
664      f      'PCONEXPA AP88, ',YEAR,'-',MONTH,'-',DAY,' ',HO,':',MI,':',SE
665 3500  FORMAT(A,14.4,5(A,12.2))
666      CALL HPTBOX(897.0,0.0,35,JETTXT)
667
668      CALL HPEJECT
669      GOTO 400
670      ENDIF
671
672      WRITE(*,2800)
673 2800  FORMAT(/)
674
675      IF (LOCKTOLINEARIZED) THEN
676      WRITE(*,2900) 'G-matrix          : linearized.'
677      ELSE
678      WRITE(*,2900) 'G-matrix          : not linearized.'
679      ENDIF
680
681      IF (AVOIDCIRCONV) THEN
682      WRITE(*,2900) 'Avoid circular convolution : True.'
683      ELSE
684      WRITE(*,2900) 'Avoid circular convolution : False.'
685      ENDIF
686
687      IF (ONEOVERFWEIGHT) THEN
688      WRITE(*,2900) 'Weight          : 1/F.'
689      ELSE
690      WRITE(*,2900) 'Weight          : 1.'
691      ENDIF
692
693      WRITE(*,2900) 'Number of channels      : ',NCH
694
695      WRITE(*,2900) 'Number of iterations    : ',NOOFITERATIONS
696 2900  FORMAT(' ',A,14)
697
698      WRITE(*,3000) 'Run time          : ',INT(TIME/60.0),':',
699      f      TIME-60.0*INT(TIME/60.0)
700 3000  FORMAT(' ',A,13,A,F5.2)
701
702      WRITE(*,3100) 'Pulse FWHM          : ',PULSEFWHM, ' ch.'
703 3100  FORMAT(' ',A,F8.2,A)
704
705      IF (ADDNOISETOPULSE) THEN
706      WRITE(*,2900) 'Add noise to pulse      : True.'
707      ELSE
708      WRITE(*,2900) 'Add noise to pulse      : False.'
709      ENDIF
710
711      WRITE(*,3200) 'Reduced chi-square      : ',REDCHISQUARED
712
713      WRITE(*,2800)
714
715      IF ((.NOT.LOCKED(1)).AND.ADDNOISE.AND.
716      f      (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
717      WRITE(*,3200) 'Stray-light intensity :',

```

```

718 f          THETAEST(1),' +/- ',SQRT(ABS(GMATRIX(1,1)))
719 3200 FORMAT(' ',A,F12.5,A,F9.4)
720 ELSEIF (LOCKED(1)) THEN
721 WRITE(*,3200) 'Stray-light intensity :',
722 f          THETAEST(1),' Locked'
723 ELSEIF ((.NOT.LOCKED(1)).AND.
724 f          ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
725 f          (.NOT.ONEOVERFWEIGHT))) THEN
726 WRITE(*,3200) 'Stray-light intensity :',
727 f          THETAEST(1)
728 ENDIF
729
730 IF ((.NOT.LOCKED(2)).AND.ADDNOISE.AND.
731 f          (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
732 WRITE(*,3200) 'Pulse position      :',
733 f          THETAEST(2)/DX,' +/- ',SQRT(ABS(GMATRIX(2,2)))/DX
734 ELSEIF (LOCKED(2)) THEN
735 WRITE(*,3200) 'Pulse position      :',
736 f          THETAEST(2)/DX,' Locked'
737 ELSEIF ((.NOT.LOCKED(2)).AND.
738 f          ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
739 f          (.NOT.ONEOVERFWEIGHT))) THEN
740 WRITE(*,3200) 'Pulse position      :',
741 f          THETAEST(2)/DX
742 ENDIF
743
744 DO 430 I=1,NOOFDECAYS
745 IF ((.NOT.LOCKED(I+2)).AND.ADDNOISE.AND.
746 f          (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
747 WRITE(*,3300) 'Intensity, decay no ',I,' :',
748 f          THETAEST(I+2),' +/- ',SQRT(ABS(GMATRIX(I+2,I+2)))
749 3300 FORMAT(' ',A,I1,A,F12.5,A,F9.4)
750 ELSEIF (LOCKED(I+2)) THEN
751 WRITE(*,3300) 'Intensity, decay no ',I,' :',
752 f          THETAEST(I+2),' Locked'
753 ELSEIF ((.NOT.LOCKED(I+2)).AND.
754 f          ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
755 f          (.NOT.ONEOVERFWEIGHT))) THEN
756 WRITE(*,3300) 'Intensity, decay no ',I,' :',
757 f          THETAEST(I+2)
758 ENDIF
759 430 CONTINUE
760
761 DO 440 I=1,NOOFDECAYS
762 IF ((.NOT.LOCKED(I+2+NOOFDECAYS)).AND.ADDNOISE.AND.
763 f          (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
764 WRITE(*,3300) 'Decay rate, decay no ',I,' :',
765 f          THETAEST(I+2+NOOFDECAYS),' +/- ',
766 f          SQRT(ABS(GMATRIX(I+2+NOOFDECAYS,I+2+NOOFDECAYS)))
767 ELSEIF (LOCKED(I+2+NOOFDECAYS)) THEN
768 WRITE(*,3300) 'Decay rate, decay no ',I,' :',
769 f          THETAEST(I+2+NOOFDECAYS),' Locked'
770 ELSEIF ((.NOT.LOCKED(I+2+NOOFDECAYS)).AND.
771 f          ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
772 f          (.NOT.ONEOVERFWEIGHT))) THEN
773 WRITE(*,3300) 'Decay rate, decay no ',I,' :',
774 f          THETAEST(I+2+NOOFDECAYS)
775 ENDIF
776 440 CONTINUE
777

```

```

778 IF ((.NOT.LOCKED(LR)).AND.ADDNOISE.AND.
779 f          (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
780 WRITE(*,3200) 'Background      :',
781 f          THETAEST(LR),' +/- ',SQRT(ABS(GMATRIX(LR,LR)))
782 ELSEIF (LOCKED(LR)) THEN
783 WRITE(*,3200) 'Background      :',
784 f          THETAEST(LR),' Locked'
785 ELSEIF ((.NOT.LOCKED(LR)).AND.
786 f          ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
787 f          (.NOT.ONEOVERFWEIGHT))) THEN
788 WRITE(*,3200) 'Background      :',
789 f          THETAEST(LR)
790 ENDIF
791
792 WRITE(*,2800)
793
794 450 WRITE(*,*) 'New iteration? (Y/*):'
795 READ(*,1300) SV
796 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
797 GOTO 140
798 ENDIF
799
800 WRITE(*,3400) 'New calculation? (Y/*):'
801 3400 FORMAT('/ ',A)
802 READ(*,1300) SV
803 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
804 GOTO 10
805 ELSE
806 STOP
807 ENDIF
808 END
809
810 REAL FUNCTION F(THETA,M,X)
811 C Anders Persson, 1988
812
813 IMPLICIT LOGICAL (A-Z)
814
815 INTEGER N,L,MAXNOOFDECAYS
816
817 PARAMETER ( N=512 )
818 PARAMETER ( MAXNOOFDECAYS=2, L=2*MAXNOOFDECAYS+3 )
819
820 REAL P(1:N),PCONVE(1:N,1:MAXNOOFDECAYS),THETA(1:*),DX
821 REAL X,SLASK1,SLASK2,SLX
822
823 INTEGER M,I,NCH,NCH2,DNCH2,NOOFDECAYS,IX1,IX2
824
825 COMMON /BLOCK1/ P,PCONVE
826 COMMON /BLOCK5/ NCH,NCH2,DNCH2,NOOFDECAYS,DX
827
828 SLX=X/DX+1.0
829 IX1=INT(SLX)
830 IX1=MIN(IX1,NCH-1)
831 IX1=MAX(IX1,1)
832 IX2=IX1+1
833 SLX=MIN(SLX,FLOAT(NCH))
834 SLX=MAX(SLX,1.0)
835
836 SLASK1=THETA(1)*P(IX1)+THETA(2*M+3)
837 SLASK2=THETA(1)*P(IX2)+THETA(2*M+3)

```



```

838 DO 10 I=1,M
839     SLASK1=SLASK1+THETA(I+2)*PCONVE(IX1,I)
840     SLASK2=SLASK2+THETA(I+2)*PCONVE(IX2,I)
841 10 CONTINUE
842 F=SLASK1+(SLASK2-SLASK1)*(SLX-IX1)
843 RETURN
844 END
845
846 REAL FUNCTION DFDTH(J,THETA,M,X)
847 C Anders Persson, 1988
848 IMPLICIT LOGICAL (A-Z)
849
850 INTEGER N,L,MAXNOOFDECAYS
851
852 PARAMETER ( N=512 )
853 PARAMETER ( MAXNOOFDECAYS=2, L=2*MAXNOOFDECAYS+3 )
854
855 REAL P(1:N),PPRIM(1:N),PCONVE(1:N,1:MAXNOOFDECAYS)
856 REAL THETA(1:*),DX,X,PCONVXE(1:N,1:MAXNOOFDECAYS)
857 REAL PPRIMCONVE(1:N,1:MAXNOOFDECAYS),SLASK
858
859 INTEGER I,NCH,NCH2,DNCH2,J,K,M,NOOFDECAYS,M,IX
860
861 COMMON /BLOCK1/ P,PCONVE
862 COMMON /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
863 COMMON /BLOCK5/ NCH,NCH2,DNCH2,NOOFDECAYS,DX
864
865 IX=NINT(X/DX+1.0)
866 IX=MIN(IX,NCH)
867 IX=MAX(IX,1)
868
869 IF (J.EQ.1) THEN
870     DFDTH=P(IX)
871 ELSEIF (J.EQ.2) THEN
872     SLASK=-THETA(1)*PPRIM(IX)
873     DO 10 I=1,M
874         SLASK=SLASK-THETA(I+2)*PPRIMCONVE(IX,I)
875 10 CONTINUE
876     DFDTH=SLASK/DX
877 ELSEIF (J.GE.3.AND.J.LE.M+2) THEN
878     DFDTH=PCONVE(IX,J-2)
879 ELSEIF (J.GE.M+3.AND.J.LE.2*M+2) THEN
880     DFDTH=-THETA(J-M)*PCONVXE(IX,J-M-2)
881 ELSEIF (J.EQ.2*M+3) THEN
882     DFDTH=1.0
883 ENDIF
884
885 RETURN
886 END
887
888 REAL FUNCTION D2FDTH2(J,K,THETA,M,X)
889 C Anders Persson, 1988
890
891 IMPLICIT LOGICAL (A-Z)
892
893 INTEGER N,L,MAXNOOFDECAYS
894
895 PARAMETER ( N=512 )
896 PARAMETER ( MAXNOOFDECAYS=2, L=2*MAXNOOFDECAYS+3 )
897

```

```

898 REAL P(1:N),PPRIM(1:N),PBIS(1:N),X,SLASK
899 REAL PCONVE(1:N,1:MAXNOOFDECAYS)
900 REAL PCONVXE(1:N,1:MAXNOOFDECAYS)
901 REAL PCONVX2E(1:N,1:MAXNOOFDECAYS)
902 REAL PPRIMCONVE(1:N,1:MAXNOOFDECAYS)
903 REAL PPRIMCONVXE(1:N,1:MAXNOOFDECAYS)
904 REAL PBISCONVE(1:N,1:MAXNOOFDECAYS),DX,THETA(1:*)
905
906 INTEGER I,NCH,NCH2,DNCH2,J,K,M,NOOFDECAYS,IX
907
908 LOGICAL JGTK
909
910 COMMON /BLOCK1/ P,PCONVE
911 COMMON /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
912 COMMON /BLOCK3/ PCONVX2E,PPRIMCONVXE,PBIS,PBISCONVE
913 COMMON /BLOCK5/ NCH,NCH2,DNCH2,NOOFDECAYS,DX
914
915 IX=NINT(X/DX+1.0)
916 IX=MIN(IX,NCH)
917 IX=MAX(IX,1)
918
919 JGTK=J.GT.K
920 IF (JGTK) THEN
921     I=J
922     J=K
923     K=I
924 ENDIF
925 D2FDTH2=0.0
926
927 IF (J.EQ.1) THEN
928     IF (K.EQ.2) THEN
929         D2FDTH2=-PPRIM(IX)/DX
930     ENDIF
931 ELSEIF (J.EQ.2) THEN
932     IF (K.EQ.2) THEN
933         SLASK=THETA(1)*PBIS(IX)
934         DO 10 I=1,M
935             SLASK=SLASK+THETA(I+2)*PBISCONVE(IX,I)
936 10 CONTINUE
937         D2FDTH2=SLASK/(DX**2)
938     ELSEIF (K.GE.3.AND.K.LE.M+2) THEN
939         D2FDTH2=-PPRIMCONVE(IX,K-2)/DX
940     ELSEIF (K.GE.M+3.AND.K.LE.2*M+2) THEN
941         D2FDTH2=PPRIMCONVXE(IX,K-M-2)*THETA(K-M)/DX
942     ENDIF
943 ELSEIF (J.GE.3.AND.J.LE.M+2) THEN
944     IF (K.EQ.M+J) THEN
945         D2FDTH2=-PCONVXE(IX,J-2)
946     ENDIF
947 ELSEIF (J.GE.M+3.AND.J.LE.2*M+2) THEN
948     IF (K.EQ.J) THEN
949         D2FDTH2=THETA(J-M)*PCONVX2E(IX,J-M-2)
950     ENDIF
951 ENDIF
952
953 IF (JGTK) THEN
954     I=J
955     J=K
956     K=I
957 ENDIF

```

```

958 RETURN
959 END
960
961
962 SUBROUTINE CALCCONV(THETA,LOCKED,LINEARIZED)
963 C Anders Persson, 1988
964
965 IMPLICIT LOGICAL (A-Z)
966
967 INTEGER N,NHALF,L,MAXNOOFDECAYS
968
969 PARAMETER ( N=512, NHALF=N/2 )
970 PARAMETER ( MAXNOOFDECAYS=2, L=2*MAXNOOFDECAYS+3 )
971
972 REAL PO(1:N),P(1:N),PPRIM(1:N),PBIS(1:N)
973 REAL PCONVE(1:N,1:MAXNOOFDECAYS)
974 REAL PCONVXE(1:N,1:MAXNOOFDECAYS)
975 REAL PCONVX2E(1:N,1:MAXNOOFDECAYS)
976 REAL PPRIMCONVE(1:N,1:MAXNOOFDECAYS)
977 REAL PPRIMCONVXE(1:N,1:MAXNOOFDECAYS)
978 REAL PBISCONVE(1:N,1:MAXNOOFDECAYS),DX,THETA(1:*)
979
980 COMPLEX FTPO(1:N),FTE(1:N,1:MAXNOOFDECAYS)
981 COMPLEX FTEX(1:N,1:MAXNOOFDECAYS),FTEX2(1:N,1:MAXNOOFDECAYS)
982 COMPLEX SL1(1:N),SL2(1:N),W1(1:NHALF),W2(1:NHALF)
983
984 INTEGER I,NCH,NCH2,DNCH2,NDIFF,J,NOOFDECAYS
985
986 LOGICAL LINEARIZED,LOCKED(1:*),INV
987
988 COMMON /BLOCK1/ P,PCONVE
989 COMMON /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
990 COMMON /BLOCK3/ PCONVX2E,PPRIMCONVXE,PBIS,PBISCONVE
991 COMMON /BLOCK4/ PO,FTPO,FTE,FTEX,FTEX2,SL1,SL2,W1,W2
992 COMMON /BLOCK5/ NCH,NCH2,DNCH2,NOOFDECAYS,DX
993
994 IF (THETA(2).EQ.0.0) THEN
995 DO 10 I=1,NCH
996 P(I)=PO(I)
997 CONTINUE
998 ELSE
999 DO 20 I=1,NCH2
1000 SL1(I)=FTPO(I)
1001 CONTINUE
1002 NDIFF=0
1003 CALL MOVEDIFF(SL1,W1,NCH2,THETA(2)/DX,NDIFF)
1004 DO 30 I=1,NCH
1005 P(I)=REAL(SL1(I))
1006 CONTINUE
1007 30
1008 ENDIF
1009
1010 IF (.NOT.LOCKED(2)) THEN
1011 DO 40 I=1,NCH2
1012 SL1(I)=FTPO(I)
1013 CONTINUE
1014 NDIFF=1
1015 CALL MOVEDIFF(SL1,W1,NCH2,THETA(2)/DX,NDIFF)
1016 DO 50 I=1,NCH
1017 PPRIM(I)=REAL(SL1(I))
1018 CONTINUE

```

```

1018 IF (.NOT.LINEARIZED) THEN
1019 DO 60 I=1,NCH2
1020 SL1(I)=FTPO(I)
1021 60
1022 CONTINUE
1023 NDIFF=2
1024 CALL MOVEDIFF(SL1,W1,NCH2,THETA(2)/DX,NDIFF)
1025 DO 70 I=1,NCH
1026 PBIS(I)=REAL(SL1(I))
1027 CONTINUE
1028 70
1029 ENDIF
1030 ENDIF
1031
1032 DO 100 J=1,NOOFDECAYS
1033 DO 80 I=1,NCH2
1034 FTE(I,J)=
1035 f CMLPX(EXP(-MIN(THETA(NOOFDECAYS+J+2)*(I-1)*DX,87.0)),0.0)
1036 80
1037 CONTINUE
1038 DO 90 I=NCH2+1,DNCH2
1039 FTE(I,J)=CMLPX(0.0,0.0)
1040 90
1041 CONTINUE
1042 100
1043 CONTINUE
1044
1045 DO 130 J=1,NOOFDECAYS
1046 IF (.NOT.LOCKED(NOOFDECAYS+J+2)) THEN
1047 DO 110 I=1,NCH2
1048 FTEX(I,J)=FTE(I,J)*(I-1)*DX
1049 110
1050 CONTINUE
1051 DO 120 I=NCH2+1,DNCH2
1052 FTEX(I,J)=CMLPX(0.0,0.0)
1053 120
1054 CONTINUE
1055 130
1056 ENDIF
1057 CONTINUE
1058
1059 IF (.NOT.LINEARIZED) THEN
1060 DO 160 J=1,NOOFDECAYS
1061 IF (.NOT.LOCKED(NOOFDECAYS+J+2)) THEN
1062 DO 140 I=1,NCH2
1063 FTEX2(I,J)=FTEX(I,J)*(I-1)*DX
1064 140
1065 CONTINUE
1066 DO 150 I=NCH2+1,DNCH2
1067 FTEX2(I,J)=CMLPX(0.0,0.0)
1068 150
1069 CONTINUE
1070 160
1071 ENDIF
1072 CONTINUE
1073 ENDIF
1074 INV=.FALSE.
1075
1076 DO 170 J=1,NOOFDECAYS
1077 CALL FFT(FTE(1,J),W2,DNCH2,INV)
1078 IF (.NOT.LOCKED(NOOFDECAYS+J+2)) THEN
1079 CALL FFT(FTEX(1,J),W2,DNCH2,INV)
1080 IF (.NOT.LINEARIZED) THEN
1081 CALL FFT(FTEX2(1,J),W2,DNCH2,INV)
1082 ENDIF
1083 ENDIF
1084 170
1085 CONTINUE
1086
1087 DO 180 I=1,NCH
1088 SL1(I)=CMLPX(P(I),0.0)

```

```

1078 180 CONTINUE
1079 DO 190 I=NCH+1,DNCH2
1080 SL1(I)=CMPLX(0.0,0.0)
1081 190 CONTINUE
1082 INV=.FALSE.
1083 CALL FFT(SL1,W2,DNCH2,INV)
1084
1085 DO 260 J=1,NOOFDECAYS
1086 DO 200 I=1,DNCH2
1087 SL2(I)=SL1(I)*FTE(I,J)
1088 200 CONTINUE
1089 INV=.TRUE.
1090 CALL FFT(SL2,W2,DNCH2,INV)
1091 DO 210 I=1,NCH
1092 PCONVE(I,J)=REAL(SL2(I))
1093 210 CONTINUE
1094 IF (.NOT.LOCKED(NOOFDECAYS+J+2)) THEN
1095 DO 220 I=1,DNCH2
1096 SL2(I)=SL1(I)*FTEX(I,J)
1097 220 CONTINUE
1098 INV=.TRUE.
1099 CALL FFT(SL2,W2,DNCH2,INV)
1100 DO 230 I=1,NCH
1101 PCONVXE(I,J)=REAL(SL2(I))
1102 230 CONTINUE
1103 IF (.NOT.LINEARIZED) THEN
1104 DO 240 I=1,DNCH2
1105 SL2(I)=SL1(I)*FTEX2(I,J)
1106 240 CONTINUE
1107 INV=.TRUE.
1108 CALL FFT(SL2,W2,DNCH2,INV)
1109 DO 250 I=1,NCH
1110 PCONVX2E(I,J)=REAL(SL2(I))
1111 250 CONTINUE
1112 ENDDIF
1113 ENDDIF
1114 260 CONTINUE
1115
1116 IF (.NOT.LOCKED(2)) THEN
1117 DO 270 I=1,NCH
1118 SL1(I)=CMPLX(PPRIM(I),0.0)
1119 270 CONTINUE
1120 DO 280 I=NCH+1,DNCH2
1121 SL1(I)=CMPLX(0.0,0.0)
1122 280 CONTINUE
1123 INV=.FALSE.
1124 CALL FFT(SL1,W2,DNCH2,INV)
1125
1126 DO 330 J=1,NOOFDECAYS
1127 DO 290 I=1,DNCH2
1128 SL2(I)=SL1(I)*FTE(I,J)
1129 290 CONTINUE
1130 INV=.TRUE.
1131 CALL FFT(SL2,W2,DNCH2,INV)
1132 DO 300 I=1,NCH
1133 PPRIMCONVE(I,J)=REAL(SL2(I))
1134 300 CONTINUE
1135 IF ((.NOT.LINEARIZED).AND.(.NOT.LOCKED(NOOFDECAYS+J+2))) THEN
1136 DO 310 I=1,DNCH2
1137 SL2(I)=SL1(I)*FTEX(I,J)

```

```

1138 310 CONTINUE
1139 INV=.TRUE.
1140 CALL FFT(SL2,W2,DNCH2,INV)
1141 DO 320 I=1,NCH
1142 PPRIMCONVXE(I,J)=REAL(SL2(I))
1143 320 CONTINUE
1144 ENDDIF
1145 330 CONTINUE
1146 ENDDIF
1147
1148 IF ((.NOT.LINEARIZED).AND.(.NOT.LOCKED(2))) THEN
1149 DO 340 I=1,NCH
1150 SL1(I)=CMPLX(PBIS(I),0.0)
1151 340 CONTINUE
1152 DO 350 I=NCH+1,DNCH2
1153 SL1(I)=CMPLX(0.0,0.0)
1154 350 CONTINUE
1155 INV=.FALSE.
1156 CALL FFT(SL1,W2,DNCH2,INV)
1157
1158 DO 380 J=1,NOOFDECAYS
1159 DO 360 I=1,DNCH2
1160 SL2(I)=SL1(I)*FTE(I,J)
1161 360 CONTINUE
1162 INV=.TRUE.
1163 CALL FFT(SL2,W2,DNCH2,INV)
1164 DO 370 I=1,NCH
1165 PBISCONVE(I,J)=REAL(SL2(I))
1166 370 CONTINUE
1167 380 CONTINUE
1168 ENDDIF
1169
1170 RETURN
1171 END
1172
1173 SUBROUTINE MOVEDIFF(A,W,N,DC,NDIFF)
1174 C Anders Persson, 1988
1175
1176 IMPLICIT LOGICAL (A-Z)
1177 INTEGER N,K,NDIFF
1178 COMPLEX A(0:*),W(0:*)
1179 REAL DC,PHSTEP,TWOPI,SLASK1,SLASK2
1180 PARAMETER ( TWOPI=6.283185308 )
1181 LOGICAL INV
1182
1183 PHSTEP=-TWOPI*DC/FLOAT(N)
1184 DO 10 K=0,N/2
1185 A(K)=A(K)*CEXP(CMPLX(0.0,PHSTEP*FLOAT(K)))
1186 10 CONTINUE
1187 DO 20 K=N/2+1,N-1
1188 A(K)=A(K)*CEXP(CMPLX(0.0,PHSTEP*FLOAT(K-N)))
1189 20 CONTINUE
1190
1191 IF (NDIFF.GT.0) THEN
1192 PHSTEP=TWOPI/FLOAT(N)
1193 SLASK1=PHSTEP*FLOAT(N)/20.0
1194 A(0)=CMPLX(0.0,0.0)
1195 DO 30 K=1,N/2
1196 SLASK2=MIN(PHSTEP*FLOAT(K),SLASK1)
1197 A(K)=A(K)*CMPLX(0.0,SLASK2)**NDIFF

```

```
1198 30    CONTINUE
1199        DO 40 K=N/2+1,N-1
1200            SLASK2=MIN(PHSTEP*FLOAT(N-K),SLASK1)
1201            A(K)=A(K)*CMPLX(0.0,-SLASK2)**NDIFF
1202 40    CONTINUE
1203        ENDIF
1204
1205        INV=.TRUE.
1206        CALL FFT(A,W,N,INV)
1207        RETURN
1208        END
```

PCONEXP

Programmet utför exakt samma beräkningar som programmet PCONEXPA men det finns ingen möjlighet att undvika cirkulär faltning. Detta är ingen allvarlig restriktion (se kommentar till PCONEXPA-programmet) men det innebär en halvering av dataareornas storlek och en minskning av exekveringstiden. Dessutom beräknar programmet faltningar och derivationer/translationer på ett effektivare sätt. T.ex. används det analytiska uttrycket för den diskreta Fouriertransformen av en exponential istället för att använda en FFT-rutin, detta ger en 10 ggr snabbare beräkning av dessa vektorer. Enligt [13] skulle en minskning i exekveringstid kunna åstadkommas genom att använda Hartley transformen [14] istället för en FFT rutin men det är troligen inte sant [15]. För verkligt stora besparingar i exekveringstid vore det mycket intressant att skriva om Fast Fourier transformrutinen och matrisräkningsbiblioteket för den array-processor från Data Translation (model DT 7020) man kan montera i en PC AT. Denna lär göra en 1024 punkters komplex Fouriertransform på ≈ 25 millisekunder. Detta skall jämföras med de 5-10 sekunder det tar med Fortran-rutinen.

Utskrift från testkörning av programmet PCONEXP:

```
No of decays (1.. 3)      : 1
No of channels ( 5.. 512) : 128
Stray-light intensity    : 0
Pulse position (channels) : 0
Intensity, decay no 1    : 0.1
Decay rate, decay no 1   : 10
Background               : 20
Add noise to data? (Y/*) : Y
Plot errorbars? (Y/*)   : Y
Pulse FWHM (channels)    : 10
Add noise to pulse? (Y/*) : N
```

Toggle Free/Locked with <shift>+Function key.

```
F1 : Stray-light intensity      0.0000, Locked
F2 : Pulse position            0.0000E+00, Locked
F3 : Intensity , decay no 1    0.1000, Free
F4 : Decay rate, decay no 1    10.0000, Free
F5 : Background                20.0000, Free
F10 : Calculate.
<Alt>+ F10 : Preview.
<Shift>+ F10 : G-matrix = not linearized.
<Alt>+ F1 : Weight = 1/F.

<<shift>+F2>
```

Toggle Free/Locked with <shift>+Function key.

F1	: Stray-light intensity	0.0000,	Locked
F2	: Pulse position	0.0000E+00,	Free
F3	: Intensity , decay no 1	0.1000,	Free
F4	: Decay rate, decay no 1	10.0000,	Free
F5	: Background	20.0000,	Free
F10	: Calculate.		

<Alt>+ F10 : Preview.
<Shift>+ F10 : G-matrix = not linearized.
<Alt>+ F1 : Weight = 1/F.

<F10>

Press Q to quit iteration.

It: 0
Pulse position : 0.00000E+00
Intensity(1) : 0.10000 ,Decay rate(1): 10.000
Background : 20.000

It: 1, STEP: 0.106111
Pulse position : 0.67656E-01
Intensity(1) : 0.98918E-01 ,Decay rate(1): 9.8808
Background : 18.474

It: 2, STEP: 0.213576E-01
Pulse position : 0.69095E-01
Intensity(1) : 0.98936E-01 ,Decay rate(1): 9.8824
Background : 18.471

It: 3, STEP: 0.122852E-03
Pulse position : 0.69104E-01
Intensity(1) : 0.98936E-01 ,Decay rate(1): 9.8824
Background : 18.471

It: 4, STEP: 0.354614
Pulse position : 0.93627E-01
Intensity(1) : 0.99551E-01 ,Decay rate(1): 9.9838
Background : 19.998

It: 5, STEP: 0.414929E-02
Pulse position : 0.93325E-01
Intensity(1) : 0.99531E-01 ,Decay rate(1): 9.9806
Background : 19.990

It: 6, STEP: 0.113032E-03
Pulse position : 0.93315E-01
Intensity(1) : 0.99530E-01 ,Decay rate(1): 9.9806
Background : 19.990

It: 7, STEP: 0.238581E-05
Pulse position : 0.93314E-01
Intensity(1) : 0.99530E-01 ,Decay rate(1): 9.9806
Background : 19.990

Press any key to continue. <Return>
Plot on Roland plotter? (Y/*) : N
Plot on HP-Laserjet? (Y/*) : Y

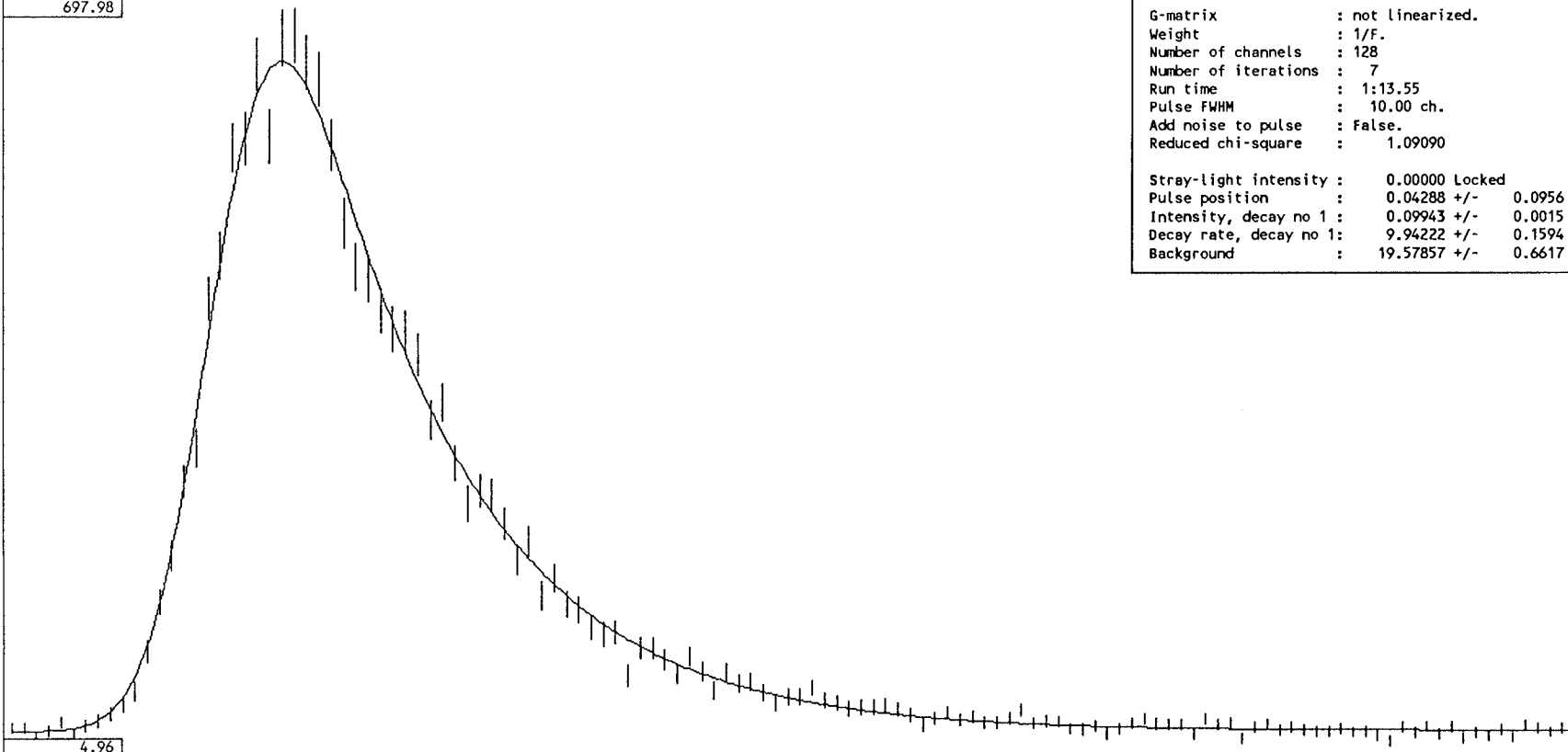
G-matrix : not linearized.
Weight : 1/F.
Number of channels : 128
Number of iterations : 7
Run time : 1:20.25
Pulse FWHM : 10.00 ch.
Add noise to pulse : False.
Reduced chi-square : 1.01089

Stray-light intensity : 0.00000 Locked
Pulse position : 0.09331 +/- 0.0976
Intensity, decay no 1 : 0.09953 +/- 0.0015
Decay rate, decay no 1 : 9.98060 +/- 0.1580
Background : 19.99007 +/- 0.6627

New iteration? (Y/*) : N

New calculation? (Y/*): N

697.98

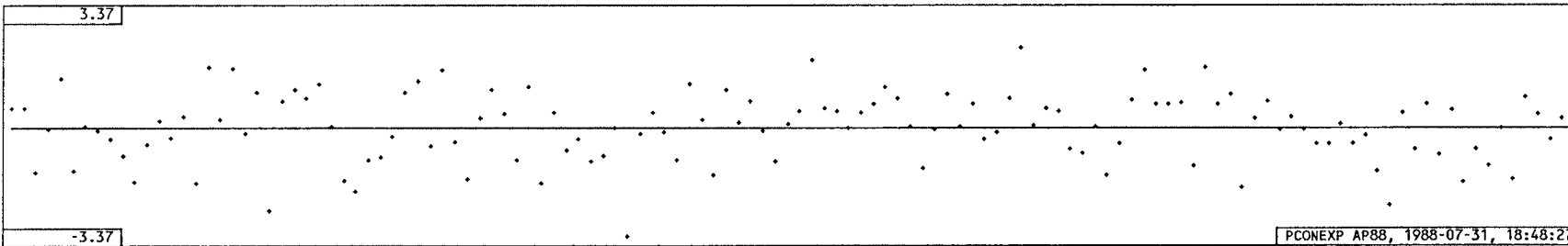


G-matrix : not linearized.
Weight : 1/F.
Number of channels : 128
Number of iterations : 7
Run time : 1:13.55
Pulse FWHM : 10.00 ch.
Add noise to pulse : False.
Reduced chi-square : 1.09090

Stray-light intensity : 0.00000 Locked
Pulse position : 0.04288 +/- 0.0956
Intensity, decay no 1 : 0.09943 +/- 0.0015
Decay rate, decay no 1: 9.94222 +/- 0.1594
Background : 19.57857 +/- 0.6617

4.96

3.37



-3.37

PCONEXP AP88, 1988-07-31, 18:48:27

Kommentarer till programlistningen:

Rad	Kommentar
1-939	Huvudprogrammet, funktionsberäkningsrutinerna och derivatarutinerna är i stort sett samma som för PCONEXPA, se därför programkommentarerna till det programmet.
941-1115	Rutinen CALCCONV beräknar samma saker som i programmet PCONEXPA men gör ingenting i onödan samt utför beräkningarna på ett effektivare sätt.
1117-1164	Även rutinen MOVEDIFF är ändrad från PVONEXPA-programmet. Rutinen är uppsnabbad genom att utnyttja symmetrier i beräkningarna. Se kommentarer i anslutning till testprogrammet TESTMD.
1166-1233	Rutinen beräknar Fouriertransformen av: $\exp(-\theta x_i)$, $x_i \cdot \exp(-\theta x_i)$ och $x_i^2 \cdot \exp(-\theta x_i)$ från analytiska uttryck för dessa. För ytterligare information se kommentarer i anslutning till testprogrammet TFEXP.

```
1 PROGRAM PCONEXP
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5
6 INTEGER N,NHALF,L,MAXNOOFITERATIONS,MAXNOOFDECAYS
7
8 PARAMETER ( N=512, NHALF=N/2, MAXNOOFITERATIONS=100 )
9 PARAMETER ( MAXNOOFDECAYS=3, L=2*MAXNOOFDECAYS+3 )
10
11 REAL Y(1:N),X(1:N),DX,DET,NORMAL,THETAEST(1:L)
12 REAL RESIDUE(1:N),SIGMA(1:N),F,PULSEFWHM,JX,JY,JXL,JYL,DJY
13 REAL GMATRIX(1:L,1:L),GVECTOR(1:L),DELTATHETA(1:L)
14 REAL SLASK,DFDTH,D2FDTH2,THETAMIN(1:L),THETAMAX(1:L)
15 REAL SL1,SL2,SL3,SL4,SL5,TIME,STEP,REDCHISQUARED
16 REAL PO(1:N),P(1:N),PPRIM(1:N),PBIS(1:N)
17 REAL PCONVE(1:N,1:MAXNOOFDECAYS)
18 REAL PCONVXE(1:N,1:MAXNOOFDECAYS)
19 REAL PCONVX2E(1:N,1:MAXNOOFDECAYS)
20 REAL PPRIMCONVE(1:N,1:MAXNOOFDECAYS)
21 REAL PPRIMCONVXE(1:N,1:MAXNOOFDECAYS)
22 REAL PBISCONVE(1:N,1:MAXNOOFDECAYS)
23
24 COMPLEX FTP0(1:N),S1(1:N),S2(1:N),W(1:NHALF)
25
26 INTEGER R,K,I,NCH,NCH2,NOOFITERATIONS,J,NOOFDECAYS,LR
27 INTEGER NOOFVARIABLEPAR,NDIFF
28
29 INTEGER*2 HO,MI,SE,HN,ANS,KBDCHK,KBDINC,YEAR,MONTH,DAY
30
31 CHARACTER SV*1,LOCKEDORFREE(1:L)*6,JETTXT*50
32
33 LOGICAL SING,ERRORBAR,LINEARIZED,LOCKED(1:L),ADDNOISE,INV
34 LOGICAL LOCKTOLINEARIZED,LOCKWEIGHT,SLLOCKED(1:L)
35 LOGICAL ONEOVERFWEIGHT,ADDNOISETOPULSE
36
37 EXTERNAL F
38
39 COMMON /BLOCK1/ P,PCONVE
40 COMMON /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
41 COMMON /BLOCK3/ PCONVX2E,PPRIMCONVXE,PBIS,PBISCONVE
42 COMMON /BLOCK4/ PO,FTP0,S1,S2,W
43 COMMON /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX
44
45 10 WRITE(*,1000) 'No of decays (1..',MAXNOOFDECAYS,'):'
46 1000 FORMAT(' ',A,12,A)
47 READ(*,*,ERR=10) NOOFDECAYS
48 IF (NOOFDECAYS.LT.1.OR.NOOFDECAYS.GT.MAXNOOFDECAYS) THEN
49 GOTO 10
50 ENDIF
51 LR=2*NOOFDECAYS+3
52
53 20 WRITE(*,1100) 'No of channels (' ,LR,'..',N,'):'
54 1100 FORMAT(' ',A,12,A,14,A)
55 READ(*,*,ERR=20) NCH
56 IF (NCH.LT.LR.OR.NCH.GT.N) THEN
57 GOTO 20
```

```
58 ENDIF
59 DX=1.0/(NCH-1)
60
61 30 WRITE(*,*) 'Stray-light intensity:'
62 READ(*,*,ERR=30) THETAEST(1)
63 IF (THETAEST(1).LT.0.0) THEN
64 GOTO 30
65 ENDIF
66
67 40 WRITE(*,*) 'Pulse position (channels):'
68 READ(*,*,ERR=40) THETAEST(2)
69 THETAEST(2)=THETAEST(2)*DX
70
71 DO 70 I=1,NOOFDECAYS
72 50 WRITE(*,1200) 'Intensity, decay no ',I,': '
73 1200 FORMAT(' ',A,I1,A)
74 READ(*,*,ERR=50) THETAEST(I+2)
75 IF (THETAEST(I+2).LE.0.0) THEN
76 GOTO 50
77 ENDIF
78 60 WRITE(*,1200) 'Decay rate, decay no ',I,': '
79 READ(*,*,ERR=60) THETAEST(I+NOOFDECAYS+2)
80 IF (THETAEST(I+NOOFDECAYS+2).LE.0.0) THEN
81 GOTO 60
82 ENDIF
83 70 CONTINUE
84
85 80 WRITE(*,*) 'Background: '
86 READ(*,*,ERR=80) THETAEST(LR)
87
88 WRITE(*,*) 'Add noise to data? (Y/*): '
89 READ(*,1300) SV
90 1300 FORMAT(A)
91 ADDNOISE=SV.EQ.'Y'.OR.SV.EQ.'y'
92 ERRORBAR=ADDNOISE
93 IF (ADDNOISE) THEN
94 WRITE(*,*) 'Plot errorbars? (Y/*): '
95 READ(*,1300) SV
96 ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'
97 ENDIF
98
99 NCH2=LOG(NCH-0.5)/LOG(2.0)
100 NCH2=2**(NCH2+1)
101
102 DO 90 I=1,LR
103 LOCKED(I) =.FALSE.
104 SLLOCKED(I) =.TRUE.
105 LOCKEDORFREE(I)='Free '
106 90 CONTINUE
107
108 LOCKED(1) =.TRUE.
109 LOCKED(2) =.TRUE.
110 LOCKEDORFREE(1) = 'Locked'
111 LOCKEDORFREE(2) = 'Locked'
112 LOCKTOLINEARIZED =.FALSE.
113 LINEARIZED =.TRUE.
114 ONEOVERFWEIGHT =.TRUE.
115
116 100 WRITE(*,*) 'Pulse FWHM (channels):'
117 READ(*,*,ERR=100) PULSEFWHM
```

```

118 IF (PULSEFWHM.LT.0.1.OR.PULSEFWHM.GT.NCH*0.5) THEN
119 GOTO 100
120 ENDIF
121
122 WRITE(*,*) 'Add noise to pulse? (Y/*):'
123 READ(*,1300) SV
124 ADDNOISETOPULSE=SV.EQ.'Y'.OR.SV.EQ.'y'
125
126 DO 110 I=1,NCH
127 PO(I)=EXP(-MIN((LOG(16.0)/PULSEFWHM**2)*(I-NCH/7.0)**2,87.0))
128 PO(I)=1000.0*PO(I)
129 IF (ADDNOISETOPULSE) THEN
130 PO(I)=PO(I)+NORMAL(0.0,10.0)
131 ENDIF
132 P(I)=PO(I)
133 FTPO(I)=CMPLX(PO(I),0.0)
134 110 CONTINUE
135 DO 120 I=NCH+1,NCH2
136 FTPO(I)=CMPLX(0.0,0.0)
137 120 CONTINUE
138
139 CALL CALCW(W,NCH2)
140 INV=.FALSE.
141 CALL FFT(FTPO,W,NCH2,INV)
142 IF (THETAEST(1).NE.0.0) THEN
143 DO 130 I=1,NCH2
144 S1(I)=FTPO(I)
145 130 CONTINUE
146 NDIFF=0
147 CALL MOVEDIFF(S1,NCH2,THETAEST(2)/DX,NDIFF)
148 INV=.TRUE.
149 CALL FFT(S1,W,NCH2,INV)
150 DO 140 I=1,NCH
151 P(I)=REAL(S1(I))
152 140 CONTINUE
153 ENDIF
154 CALL CALCCONV(THETAEST,SLLOCKED,LINEARIZED)
155
156 DO 150 I=1,NCH
157 X(I)=(I-1)*DX
158 SLASK=F(THETAEST,NOOFDECAYS,X(I))
159 IF (ADDNOISE) THEN
160 Y(I)=NORMAL(SLASK,SQRT(ABS(SLASK)))
161 SIGMA(I)=SQRT(ABS(Y(I)))
162 ELSE
163 Y(I)=SLASK
164 SIGMA(I)=0.0
165 ENDIF
166 150 CONTINUE
167
168 160 CALL GMODE
169 CALL TMODE
170
171 WRITE(*,1400) 'Toggle Free/Locked with <shift>+Function key.'
172 1400 FORMAT(////,T10,A,/)
173
174 WRITE(*,1500) THETAEST(1),LOCKEDORFREE(1)
175 1500 FORMAT(T10,'F1 : Stray-light intensity',T45,F12.4,',',',T60,A)
176
177 WRITE(*,1600) THETAEST(2)/DX,LOCKEDORFREE(2)

```

```

178 1600 FORMAT(T10,'F2 : Pulse position',T45,G12.4,',',',T60,A)
179
180 DO 170 I=1,NOOFDECAYS
181 WRITE(*,1700) I+2,I,THETAEST(I+2),LOCKEDORFREE(I+2)
182 1700 FORMAT(T10,'F',I1,' : Intensity , decay no ',
183 f I1,T45,F12.4,',',',T60,A)
184 170 CONTINUE
185 DO 180 I=1,NOOFDECAYS
186 WRITE(*,1800) I+NOOFDECAYS+2,I,THETAEST(I+NOOFDECAYS+2),
187 f LOCKEDORFREE(I+NOOFDECAYS+2)
188 1800 FORMAT(T10,'F',I1,' : Decay rate, decay no ',
189 f I1,T45,F12.4,',',',T60,A)
190 180 CONTINUE
191 WRITE(*,1900) LR,THETAEST(LR),LOCKEDORFREE(LR)
192 1900 FORMAT(T10,'F',I1,' : Background ',T45,F12.4,',',',T60,A)
193 WRITE(*,2000) 'F10: Calculate.'
194 2000 FORMAT(T10,A)
195 WRITE(*,2100) '<Alt>+F10: Preview.'
196 2100 FORMAT(T4,A)
197 IF (LOCKTOLINEARIZED) THEN
198 WRITE(*,2200) '<Shift>+F10: G-matrix = linearized.'
199 ELSE
200 WRITE(*,2200) '<Shift>+F10: G-matrix = not linearized.'
201 ENDIF
202 2200 FORMAT(T2,A)
203 IF (ONEOVERWEIGHT) THEN
204 WRITE(*,2100) '<Alt>+F1 : Weight = 1/F.'
205 ELSE
206 WRITE(*,2100) '<Alt>+F1 : Weight = 1.'
207 ENDIF
208 WRITE(*,2300) ' '
209 2300 FORMAT(T3,A)
210
211 190 ANS=KBDCHK()
212 IF (ANS.NE.0) THEN
213 GOTO 190
214 ENDIF
215 ANS=KBDINC()
216
217 IF (ANS.GE.1084.AND.ANS.LE.1083+LR) THEN
218 I=ANS-1083
219 LOCKED(I)=.NOT.LOCKED(I)
220 LOCKEDORFREE(I)='Free '
221 IF (LOCKED(I)) THEN
222 LOCKEDORFREE(I)='Locked'
223 ENDIF
224 GOTO 160
225 ELSEIF (ANS.EQ.1113) THEN
226 WRITE(*,2000) 'Preview.'
227 IF (THETAEST(1).NE.0.0) THEN
228 DO 200 I=1,NCH2
229 S1(I)=FTPO(I)
230 200 CONTINUE
231 NDIFF=0
232 CALL MOVEDIFF(S1,NCH2,THETAEST(2)/DX,NDIFF)
233 INV=.TRUE.
234 CALL FFT(S1,W,NCH2,INV)
235 DO 210 I=1,NCH
236 P(I)=REAL(S1(I))
237 210 CONTINUE

```

```

238     ENDIF
239     CALL CALCONV(THETAEST,SLLOCKED,LINEARIZED)
240     CALL SCRPLLOT(Y,SIGMA,X,RESIDUE,NCH,F,
241     f      THETAEST,NOOFDECAYS,ERRORBAR)
242     GOTO 160
243     ELSEIF (ANS.EQ.1068) THEN
244     C      Calculate.
245     IF (LOCKED(2)) THEN
246     DO 220 I=1,NCH2
247     S1(I)=FTPO(I)
248     220    CONTINUE
249     NDIFF=0
250     CALL MOVEDIFF(S1,NCH2,THETAEST(2)/DX,NDIFF)
251     INV=.TRUE.
252     CALL FFT(S1,W,NCH2,INV)
253     DO 230 I=1,NCH
254     P(I)=REAL(S1(I))
255     230    CONTINUE
256     ENDIF
257     GOTO 290
258     ELSEIF (ANS.EQ.1093) THEN
259     LOCKTOLINEARIZED=.NOT.LOCKTOLINEARIZED
260     GOTO 160
261     ELSEIF (ANS.EQ.1104) THEN
262     ONEOVERFWEIGHT=.NOT.ONEOVERFWEIGHT
263     GOTO 160
264     ELSEIF (ANS.GE.1059.AND.ANS.LE.1058+LR) THEN
265     I=ANS-1058
266     IF (I.EQ.1) THEN
267     240    WRITE(*,2400) 'New stray-light intensity:'
268     2400    FORMAT(T10,A,11,A)
269     READ(*,*,ERR=240) THETAEST(1)
270     IF (THETAEST(1).LT.0.0) THEN
271     GOTO 240
272     ENDIF
273     GOTO 160
274     ELSEIF (I.EQ.2) THEN
275     250    WRITE(*,2400) 'New pulse position (channels):'
276     READ(*,*,ERR=250) THETAEST(2)
277     THETAEST(2)=THETAEST(2)*DX
278     GOTO 160
279     ELSEIF (I.GT.2.AND.I.LE.NOOFDECAYS+2) THEN
280     260    WRITE(*,2400) 'New intensity, decay no ',I-2,': '
281     READ(*,*,ERR=260) THETAEST(1)
282     IF (THETAEST(1).LE.0.0) THEN
283     GOTO 260
284     ENDIF
285     GOTO 160
286     ELSEIF (I.GT.NOOFDECAYS+2.AND.I.LT.LR) THEN
287     270    WRITE(*,2400) 'New decay rate, decay no ',I-NOOFDECAYS-2,': '
288     READ(*,*,ERR=270) THETAEST(1)
289     IF (THETAEST(1).LE.0.0) THEN
290     GOTO 270
291     ENDIF
292     GOTO 160
293     ELSEIF (I.EQ.LR) THEN
294     280    WRITE(*,2400) 'New background: '
295     READ(*,*,ERR=280) THETAEST(LR)
296     GOTO 160
297     ENDIF

```

```

298     GOTO 190
299     ENDIF
300     GOTO 190
301
302     C      *** Calculation ***
303     290    CONTINUE
304
305     WRITE(*,*) 'Press Q to quit iteration.'
306     WRITE(*,2500)
307     2500    FORMAT(/)
308     DO 300 I=1,2*NOOFDECAYS+2
309     THETAMIN(I)=THETAEST(I)/10.0
310     THETAMAX(I)=THETAEST(I)*10.0
311     300    CONTINUE
312     IF (THETAEST(1).LT.1.0) THEN
313     THETAMAX(1)=1.0
314     THETAMIN(1)=-1.0
315     ENDIF
316     THETAMAX(2)=THETAEST(2)+0.3*NCH*DX
317     THETAMIN(2)=THETAEST(2)-0.3*NCH*DX
318
319     THETAMAX(LR)=ABS(THETAEST(LR)*10.0)
320     THETAMIN(LR)=-ABS(THETAEST(LR)*10.0)
321
322     CALL GETTIM(HO,M1,SE,HN)
323     TIME=HO*3600.0+M1*60.0+SE+HN*0.01
324
325     WRITE(*,2600) 'It:',0
326     2600    FORMAT(' ',49(' '),/,',',A,14,A,G13.6)
327     IF (.NOT.LOCKED(1)) THEN
328     WRITE(*,2700) 'Stray-light int.:',THETAEST(1)
329     ENDIF
330     IF (.NOT.LOCKED(2)) THEN
331     WRITE(*,2700) 'Pulse position :',THETAEST(2)/DX
332     ENDIF
333     2700    FORMAT(' ',A,G12.5)
334     DO 310 R=1,NOOFDECAYS
335     WRITE(*,2800) 'Intensity(',R,') :',THETAEST(R+2),
336     f      ', Decay rate(',R,'):',THETAEST(R+NOOFDECAYS+2)
337     2800    FORMAT(' ',A,11,A,G12.5,A,11,A,G12.5)
338     310    CONTINUE
339     IF (.NOT.LOCKED(LR)) THEN
340     WRITE(*,2700) 'Background :',THETAEST(LR)
341     ENDIF
342
343     LINEARIZED=.TRUE.
344     LOCKWEIGHT=.TRUE.
345
346     DO 400 J=1,MAXNOOFITERATIONS
347     CALL ZEROMAT(GMATRIX,LR,LR,L,L)
348     CALL ZEROMAT(GVECTOR,LR,1,L,1)
349     CALL CALCONV(THETAEST,LOCKED,LINEARIZED)
350     DO 340 I=1,NCH
351     IF (ONEOVERFWEIGHT) THEN
352     IF (LOCKWEIGHT) THEN
353     SL1=-2.0/MAX(Y(1),1.0)
354     ELSE
355     SL1=-2.0/F(THETAEST,NOOFDECAYS,X(I))
356     ENDIF
357     ELSE

```

```

358     SL1=-2.0
359     ENDF
360     SL2=Y(I)-F(THETAEST,NOOFDECAYS,X(I))
361     DO 330 R=1,LR
362         IF (.NOT.LOCKED(R)) THEN
363             SL3=DFDTH(R,THETAEST,NOOFDECAYS,X(I))
364             GVECTOR(R)=GVECTOR(R)+SL1*SL2*SL3
365             DO 320 K=R,LR
366                 IF (.NOT.LOCKED(K)) THEN
367                     SL4=DFDTH(K,THETAEST,NOOFDECAYS,X(I))
368                     GMATRIX(R,K)=GMATRIX(R,K)+SL1*(-SL3*SL4)
369                     IF (.NOT.LINEARIZED) THEN
370                         SL5=D2FDTH2(R,K,THETAEST,NOOFDECAYS,X(I))
371                         GMATRIX(R,K)=GMATRIX(R,K)+SL1*SL2*SL5
372                     ENDF
373                     GMATRIX(K,R)=GMATRIX(R,K)
374                 ENDF
375             CONTINUE
376         ENDF
377     CONTINUE
378 340 CONTINUE
379
380     DO 350 R=1,LR
381         IF (LOCKED(R)) THEN
382             GMATRIX(R,R)=1.0
383         ENDF
384 350 CONTINUE
385
386     CALL INVMAT(GMATRIX,LR,DET,SING,L,L)
387     IF (SING) THEN
388         WRITE(*,*) 'Sing. G-matrix.'
389         GOTO 520
390     ENDF
391
392     CALL MULTCMAT(GMATRIX,-1.0,LR,LR,L,L)
393     CALL MULTMAT(GMATRIX,GVECTOR,DELTATHETA,LR,LR,1,L,L,1,L,1)
394
395     DO 360 I=1,2*NOOFDECAYS+3
396         IF (THETAEST(I)+DELTATHETA(I).LT.THETAMIN(I)) THEN
397             DELTATHETA(I)=THETAMIN(I)-THETAEST(I)
398         ELSEIF (THETAEST(I)+DELTATHETA(I).GT.THETAMAX(I)) THEN
399             DELTATHETA(I)=THETAMAX(I)-THETAEST(I)
400         ENDF
401 360 CONTINUE
402
403     CALL ADDMAT(THETAEST,DELTATHETA,THETAEST,LR,1,L,1,L,1,L,1)
404
405     STEP=0.0
406     DO 370 I=1,LR
407         IF (ABS(THETAEST(I)).GT.ABS(DELTATHETA(I))) THEN
408             STEP=STEP+ABS(DELTATHETA(I)/THETAEST(I))
409         ELSE
410             STEP=STEP+ABS(DELTATHETA(I))
411         ENDF
412 370 CONTINUE
413
414     WRITE(*,2600) 'It:',J,', STEP:',STEP
415     IF (.NOT.LOCKED(1)) THEN
416         WRITE(*,2700) 'Stray-light int.:',THETAEST(1)
417     ENDF

```

```

418     IF (.NOT.LOCKED(2)) THEN
419         WRITE(*,2700) 'Pulse position :',THETAEST(2)/DX
420     ENDF
421     DO 380 R=1,NOOFDECAYS
422         WRITE(*,2800) 'Intensity(',R,') :',THETAEST(R+2),
423             f ' ,Decay rate(',R,'):',THETAEST(R+NOOFDECAYS+2)
424 380 CONTINUE
425     IF (.NOT.LOCKED(LR)) THEN
426         WRITE(*,2700) 'Background :',THETAEST(LR)
427     ENDF
428
429     IF (STEP.LE.0.5E-4.AND.(.NOT.LINEARIZED)) THEN
430         NOOFITERATIONS=J
431         GOTO 410
432     ENDF
433     IF (STEP.LE.0.5E-2.AND.LINEARIZED) THEN
434         LINEARIZED=.FALSE.
435     ENDF
436     IF (LOCKTOLINEARIZED) THEN
437         LINEARIZED=.TRUE.
438     ENDF
439     IF (STEP.LE.0.5E-4.AND.LOCKTOLINEARIZED) THEN
440         NOOFITERATIONS=J
441         GOTO 410
442     ENDF
443     IF (LOCKWEIGHT.AND.STEP.LE.0.5E-2) THEN
444         LOCKWEIGHT=.FALSE.
445     ENDF
446
447 390 ANS=KBDCHK()
448     IF (ANS.NE.0) THEN
449         ANS=KBDINC()
450         IF (CHAR(ANS).EQ.'q'.OR.CHAR(ANS).EQ.'q') THEN
451             NOOFITERATIONS=J
452             GOTO 410
453         ELSE
454             GOTO 390
455         ENDF
456     ENDF
457 400 CONTINUE
458     NOOFITERATIONS=MAXNOOFITERATIONS
459 410 CONTINUE
460
461     CALL CALCCONV(THETAEST,SLLOCKED,.TRUE.)
462     CALL MULTCMAT(GMATRIX,-2.0,LR,LR,L,L)
463     CALL GETTIM(HO,MI,SE,HN)
464     TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
465     WRITE(*,*) 'Press any key to continue.'
466 420 ANS=KBDCHK()
467     IF (ANS.EQ.0) THEN
468         GOTO 420
469     ENDF
470     ANS=KBDINC()
471
472     DO 430 I=1,NCH
473         SIGMA(I)=SQRT(ABS(F(THETAEST,NOOFDECAYS,X(I))))
474 430 CONTINUE
475
476     CALL SCRIPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
477     f NOOFDECAYS,ERRORBAR)

```

```

478
479 440 WRITE(*,*) 'Plot on Roland plotter? (Y/*): '
480 READ(*,1300) SV
481 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
482   CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
483   f      NOOFDECAYS,ERRORBAR)
484   GOTO 440
485   ENDF
486
487   REDCHISQUARED=0.0
488   DO 450 I=1,NCH
489     REDCHISQUARED=REDCHISQUARED+RESIDUE(I)**2
490 450 CONTINUE
491   NOOFVARIABLEPAR=0
492   DO 460 I=1,LR
493     IF (.NOT.LOCKED(I)) THEN
494       NOOFVARIABLEPAR=NOOFVARIABLEPAR+1
495     ENDF
496 460 CONTINUE
497   IF (NCH.NE.NOOFVARIABLEPAR) THEN
498     REDCHISQUARED=REDCHISQUARED/(NCH-NOOFVARIABLEPAR)
499   ENDF
500
501 470 WRITE(*,*) 'Plot on HP-Laserjet? (Y/*): '
502 READ(*,1300) SV
503 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
504   CALL HPSTART('L')
505   CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
506   f      NOOFDECAYS,ERRORBAR)
507   JX=800
508   JXL=1119-JX
509   JYL=(13+NOOFDECAYS*2)*13
510   JY=719-JYL
511   CALL HPBOX(JX,JY,JXL,JYL)
512   JX=806
513   JY=719-16
514   DJY=-13
515
516   IF (LOCKTOLINEARIZED) THEN
517     WRITE(JETTXT,2900) 'G-matrix           : linearized.'
518   ELSE
519     WRITE(JETTXT,2900) 'G-matrix           : not linearized.'
520   ENDF
521   CALL HPPRINT(JX,JY,50,JETTXT)
522   JY=JY+DJY
523
524   IF (ONEOVERFWEIGHT) THEN
525     WRITE(JETTXT,2900) 'Weight             : 1/F.'
526   ELSE
527     WRITE(JETTXT,2900) 'Weight             : 1.'
528   ENDF
529   CALL HPPRINT(JX,JY,50,JETTXT)
530   JY=JY+DJY
531
532   WRITE(JETTXT,2900) 'Number of channels   : ',NCH
533   CALL HPPRINT(JX,JY,50,JETTXT)
534   JY=JY+DJY
535
536   WRITE(JETTXT,2900) 'Number of iterations : ',NOOFITERATIONS
537   CALL HPPRINT(JX,JY,50,JETTXT)

```

```

538   JY=JY+DJY
539
540   WRITE(JETTXT,3000) 'Run time           : ',INT(TIME/60.0),' : ',
541   f      TIME-60.0*INT(TIME/60.0)
542   CALL HPPRINT(JX,JY,50,JETTXT)
543   JY=JY+DJY
544
545   WRITE(JETTXT,3100) 'Pulse FWHM         : ',PULSEFWHM,' ch.'
546   CALL HPPRINT(JX,JY,50,JETTXT)
547   JY=JY+DJY
548
549   IF (ADDNOISETOPULSE) THEN
550     WRITE(JETTXT,2900) 'Add noise to pulse : True.'
551   ELSE
552     WRITE(JETTXT,2900) 'Add noise to pulse : False.'
553   ENDF
554   CALL HPPRINT(JX,JY,50,JETTXT)
555   JY=JY+DJY
556
557   WRITE(JETTXT,3200) 'Reduced chi-square  : ',REDCHISQUARED
558   CALL HPPRINT(JX,JY,50,JETTXT)
559   JY=JY+DJY
560
561   WRITE(JETTXT,1300) ' '
562   CALL HPPRINT(JX,JY,50,JETTXT)
563   JY=JY+DJY
564
565   IF ((.NOT.LOCKED(1)).AND.ADDNOISE.AND.
566   f      (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
567     WRITE(JETTXT,3200) 'Stray-light intensity : ',
568     f      THETAEST(1),' +/- ',SQRT(ABS(GMATRIX(1,1)))
569   ELSEIF (LOCKED(1)) THEN
570     WRITE(JETTXT,3200) 'Stray-light intensity : ',
571     f      THETAEST(1),' Locked'
572   ELSEIF ((.NOT.LOCKED(1)).AND.
573   f      ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
574   f      (.NOT.ONEOVERFWEIGHT))) THEN
575     WRITE(JETTXT,3200) 'Stray-light intensity : ',
576     f      THETAEST(1)
577   ENDF
578   CALL HPPRINT(JX,JY,50,JETTXT)
579   JY=JY+DJY
580
581   IF ((.NOT.LOCKED(2)).AND.ADDNOISE.AND.
582   f      (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
583     WRITE(JETTXT,3200) 'Pulse position      : ',
584     f      THETAEST(2)/DX,' +/- ',SQRT(ABS(GMATRIX(2,2)))/DX
585   ELSEIF (LOCKED(2)) THEN
586     WRITE(JETTXT,3200) 'Pulse position      : ',
587     f      THETAEST(2)/DX,' Locked'
588   ELSEIF ((.NOT.LOCKED(2)).AND.
589   f      ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
590   f      (.NOT.ONEOVERFWEIGHT))) THEN
591     WRITE(JETTXT,3200) 'Pulse position      : ',
592     f      THETAEST(2)/DX
593   ENDF
594   CALL HPPRINT(JX,JY,50,JETTXT)
595   JY=JY+DJY
596
597   DO 480 I=1,NOOFDECAYS

```

```

598 IF ((.NOT.LOCKED(I+2)).AND.ADDNOISE.AND.
599 (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
600 WRITE(JETTXT,3300) 'Intensity, decay no ',I,' :',
601 f THETAEST(I+2),' +/- ',SQRT(ABS(GMATRIX(I+2,I+2)))
602 ELSEIF (LOCKED(I+2)) THEN
603 WRITE(JETTXT,3300) 'Intensity, decay no ',I,' :',
604 f THETAEST(I+2),' Locked'
605 ELSEIF ((.NOT.LOCKED(I+2)).AND.
606 (.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
607 (.NOT.ONEOVERFWEIGHT)) THEN
608 WRITE(JETTXT,3300) 'Intensity, decay no ',I,' :',
609 f THETAEST(I+2)
610 ENDIF
611 CALL HPPRINT(JX,JY,50,JETTXT)
612 JY=JY+DJY
613 480 CONTINUE
614
615 DO 490 I=1,NOOFDECAYS
616 IF ((.NOT.LOCKED(I+2+NOOFDECAYS)).AND.ADDNOISE.AND.
617 (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
618 f WRITE(JETTXT,3300) 'Decay rate, decay no ',I,' :',
619 THETAEST(I+2+NOOFDECAYS),' +/- ',
620 f SQRT(ABS(GMATRIX(I+2+NOOFDECAYS,I+2+NOOFDECAYS)))
621 ELSEIF (LOCKED(I+2+NOOFDECAYS)) THEN
622 WRITE(JETTXT,3300) 'Decay rate, decay no ',I,' :',
623 f THETAEST(I+2+NOOFDECAYS),' Locked'
624 ELSEIF ((.NOT.LOCKED(I+2+NOOFDECAYS)).AND.
625 (.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
626 (.NOT.ONEOVERFWEIGHT)) THEN
627 f WRITE(JETTXT,3300) 'Decay rate, decay no ',I,' :',
628 THETAEST(I+2+NOOFDECAYS)
629 ENDIF
630 CALL HPPRINT(JX,JY,50,JETTXT)
631 JY=JY+DJY
632 490 CONTINUE
633
634 IF ((.NOT.LOCKED(LR)).AND.ADDNOISE.AND.
635 (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
636 f WRITE(JETTXT,3200) 'Background :',
637 THETAEST(LR),' +/- ',SQRT(ABS(GMATRIX(LR,LR)))
638 ELSEIF (LOCKED(LR)) THEN
639 f WRITE(JETTXT,3200) 'Background :',
640 THETAEST(LR),' Locked'
641 ELSEIF ((.NOT.LOCKED(LR)).AND.
642 (.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
643 (.NOT.ONEOVERFWEIGHT)) THEN
644 f WRITE(JETTXT,3200) 'Background :',
645 THETAEST(LR)
646 ENDIF
647 CALL HPPRINT(JX,JY,50,JETTXT)
648
649 CALL GETTIM(HO,MI,SE,HN)
650 CALL GETDAT(YEAR,MONTH,DAY)
651 WRITE(JETTXT,3500)
652 f 'PCONEXP AP88, ' YEAR, '- ',MONTH, '- ',DAY, ', ',HO, ': ',MI, ': ',SE
653 3500 FORMAT(A,14.4,5(A,12.2))
654 CALL HPTBOX(903.0,0.0,34,JETTXT)
655
656 CALL HPEJECT
657 GOTO 470

```

```

658 ENDIF
659
660 WRITE(*,2500)
661
662 IF (LOCKTOLINEARIZED) THEN
663 WRITE(*,2900) 'G-matrix : linearized.'
664 ELSE
665 WRITE(*,2900) 'G-matrix : not linearized.'
666 ENDIF
667
668 IF (ONEOVERFWEIGHT) THEN
669 WRITE(*,2900) 'Weight : 1/F.'
670 ELSE
671 WRITE(*,2900) 'Weight : 1.'
672 ENDIF
673
674 WRITE(*,2900) 'Number of channels :',NCH
675
676 WRITE(*,2900) 'Number of iterations:',NOOFITERATIONS
677 2900 FORMAT(' ',A,14)
678
679 WRITE(*,3000) 'Run time :',INT(TIME/60.0),' :',
680 f TIME-60.0*INT(TIME/60.0)
681 3000 FORMAT(' ',A,13,A,F5.2)
682
683 WRITE(*,3100) 'Pulse FWHM :',PULSEFWHM,' ch.'
684 3100 FORMAT(' ',A,F8.2,A)
685
686 IF (ADDNOISETOPULSE) THEN
687 WRITE(*,2900) 'Add noise to pulse : True.'
688 ELSE
689 WRITE(*,2900) 'Add noise to pulse : False.'
690 ENDIF
691 WRITE(*,3200) 'Reduced chi-square :',REDCHISQUARED
692 3200 FORMAT(' ',A,F12.5,A,F9.4)
693
694 WRITE(*,2500)
695
696 IF ((.NOT.LOCKED(1)).AND.ADDNOISE.AND.
697 (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
698 f WRITE(*,3200) 'Stray-light intensity :',
699 THETAEST(1),' +/- ',SQRT(ABS(GMATRIX(1,1)))
700 ELSEIF (LOCKED(1)) THEN
701 f WRITE(*,3200) 'Stray-light intensity :',
702 THETAEST(1),' Locked'
703 ELSEIF ((.NOT.LOCKED(1)).AND.
704 (.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
705 (.NOT.ONEOVERFWEIGHT)) THEN
706 f WRITE(*,3200) 'Stray-light intensity :',
707 THETAEST(1)
708 ENDIF
709
710 IF ((.NOT.LOCKED(2)).AND.ADDNOISE.AND.
711 (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
712 f WRITE(*,3200) 'Pulse position :',
713 THETAEST(2)/DX,' +/- ',SQRT(ABS(GMATRIX(2,2)))/DX
714 ELSEIF (LOCKED(2)) THEN
715 f WRITE(*,3200) 'Pulse position :',
716 THETAEST(2)/DX,' Locked'
717 ELSEIF ((.NOT.LOCKED(2)).AND.

```

```

718 f      ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
719 f      (.NOT.ONEOVERFWEIGHT))) THEN
720 WRITE(*,3200) 'Pulse position      :',
721 f      THETAEST(2)/DX
722 ENDIF
723
724 DO 500 I=1,NOOFDECAYS
725 IF ((.NOT.LOCKED(I+2)).AND.ADDNOISE.AND.
726 (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
727 WRITE(*,3300) 'Intensity, decay no ',I,' :',
728 THETAEST(I+2),' +/- ',SQRT(ABS(GMATRIX(I+2,I+2)))
729 3300 FORMAT(' ',A,I1,A,F12.5,A,F9.4)
730 ELSEIF (LOCKED(I+2)) THEN
731 WRITE(*,3300) 'Intensity, decay no ',I,' :',
732 f      THETAEST(I+2),' Locked'
733 ELSEIF ((.NOT.LOCKED(I+2)).AND.
734 (.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
735 (.NOT.ONEOVERFWEIGHT))) THEN
736 WRITE(*,3300) 'Intensity, decay no ',I,' :',
737 f      THETAEST(I+2)
738 ENDIF
739 500 CONTINUE
740
741 DO 510 I=1,NOOFDECAYS
742 IF ((.NOT.LOCKED(I+2+NOOFDECAYS)).AND.ADDNOISE.AND.
743 (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
744 WRITE(*,3300) 'Decay rate, decay no ',I,' :',
745 f      THETAEST(I+2+NOOFDECAYS),' +/- ',
746 f      SQRT(ABS(GMATRIX(I+2+NOOFDECAYS,I+2+NOOFDECAYS)))
747 ELSEIF (LOCKED(I+2+NOOFDECAYS)) THEN
748 WRITE(*,3300) 'Decay rate, decay no ',I,' :',
749 f      THETAEST(I+2+NOOFDECAYS),' Locked'
750 ELSEIF ((.NOT.LOCKED(I+2+NOOFDECAYS)).AND.
751 (.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
752 (.NOT.ONEOVERFWEIGHT))) THEN
753 WRITE(*,3300) 'Decay rate, decay no ',I,' :',
754 f      THETAEST(I+2+NOOFDECAYS)
755 ENDIF
756 510 CONTINUE
757
758 IF ((.NOT.LOCKED(LR)).AND.ADDNOISE.AND.
759 (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
760 WRITE(*,3200) 'Background      :',
761 f      THETAEST(LR),' +/- ',SQRT(ABS(GMATRIX(LR,LR)))
762 ELSEIF (LOCKED(LR)) THEN
763 WRITE(*,3200) 'Background      :',
764 f      THETAEST(LR),' Locked'
765 ELSEIF ((.NOT.LOCKED(LR)).AND.
766 (.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
767 (.NOT.ONEOVERFWEIGHT))) THEN
768 WRITE(*,3200) 'Background      :',
769 f      THETAEST(LR)
770 ENDIF
771
772 WRITE(*,2500)
773
774 520 WRITE(*,*) 'New iteration? (Y/*):'
775 READ(*,1300) SV
776 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
777 GOTO 160

```

```

778 ENDIF
779
780 WRITE(*,3400) 'New calculation? (Y/*):'
781 3400 FORMAT('/' ',A)
782 READ(*,1300) SV
783 IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
784 GOTO 10
785 ELSE
786 STOP
787 ENDIF
788 END
789
790 REAL FUNCTION F(THETA,M,X)
791 C Anders Persson, 1988
792
793 IMPLICIT LOGICAL (A-Z)
794
795 INTEGER N,L,MAXNOOFDECAYS
796
797 PARAMETER ( N=512 )
798 PARAMETER ( MAXNOOFDECAYS=3, L=2*MAXNOOFDECAYS+3 )
799
800 REAL P(1:N),PCONVE(1:N,1:MAXNOOFDECAYS),THETA(1:*),DX
801 REAL X,SLASK1,SLASK2,SLX
802
803 INTEGER M,I,NCH,NCH2,NOOFDECAYS,IX1,IX2
804
805 COMMON /BLOCK1/ P,PCONVE
806 COMMON /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX
807
808 SLX=X/DX+1.0
809 IX1=INT(SLX)
810 IX1=MIN(IX1,NCH-1)
811 IX1=MAX(IX1,1)
812 IX2=IX1+1
813 SLX=MIN(SLX,FLOAT(NCH))
814 SLX=MAX(SLX,1.0)
815
816 SLASK1=THETA(1)*P(IX1)+THETA(2*M+3)
817 SLASK2=THETA(1)*P(IX2)+THETA(2*M+3)
818 DO 10 I=1,M
819 SLASK1=SLASK1+THETA(I+2)*PCONVE(IX1,I)
820 SLASK2=SLASK2+THETA(I+2)*PCONVE(IX2,I)
821 10 CONTINUE
822 F=SLASK1+(SLASK2-SLASK1)*(SLX-IX1)
823 RETURN
824 END
825
826 REAL FUNCTION DFDTH(J,THETA,M,X)
827 C Anders Persson, 1988
828 IMPLICIT LOGICAL (A-Z)
829
830 INTEGER N,L,MAXNOOFDECAYS
831
832 PARAMETER ( N=512 )
833 PARAMETER ( MAXNOOFDECAYS=3, L=2*MAXNOOFDECAYS+3 )
834
835 REAL P(1:N),PPRIM(1:N),PCONVE(1:N,1:MAXNOOFDECAYS)
836 REAL THETA(1:*),DX,X,PCONVXE(1:N,1:MAXNOOFDECAYS)
837 REAL PPRIMCONVE(1:N,1:MAXNOOFDECAYS),SLASK

```



```

838
839 INTEGER I,NCH,NCH2,J,NOOFDECAYS,M,IX
840
841 COMMON /BLOCK1/ P,PCONVE
842 COMMON /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
843 COMMON /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX
844
845 IX=NINT(X/DX+1.0)
846 IX=MIN(IX,NCH)
847 IX=MAX(IX,1)
848
849 IF (J.EQ.1) THEN
850 DFDTH=P(IX)
851 ELSEIF (J.EQ.2) THEN
852 SLASK=-THETA(1)*PPRIM(IX)
853 DO 10 I=1,M
854 SLASK=SLASK-THETA(I+2)*PPRIMCONVE(IX,I)
855 10 CONTINUE
856 DFDTH=SLASK/DX
857 ELSEIF (J.GE.3.AND.J.LE.M+2) THEN
858 DFDTH=PCONVE(IX,J-2)
859 ELSEIF (J.GE.M+3.AND.J.LE.2*M+2) THEN
860 DFDTH=-THETA(J-M)*PCONVXE(IX,J-M-2)
861 ELSEIF (J.EQ.2*M+3) THEN
862 DFDTH=1.0
863 ENDIF
864 RETURN
865 END
866
867 REAL FUNCTION D2FDTH2(J,K,THETA,M,X)
868 C Anders Persson, 1988
869
870 IMPLICIT LOGICAL (A-Z)
871
872 INTEGER N,L,MAXNOOFDECAYS
873
874 PARAMETER ( N=512 )
875 PARAMETER ( MAXNOOFDECAYS=3, L=2*MAXNOOFDECAYS+3 )
876
877 REAL P(1:N),PPRIM(1:N),PBIS(1:N),X,SLASK
878 REAL PCONVE(1:N,1:MAXNOOFDECAYS)
879 REAL PCONVXE(1:N,1:MAXNOOFDECAYS)
880 REAL PCONVX2E(1:N,1:MAXNOOFDECAYS)
881 REAL PPRIMCONVE(1:N,1:MAXNOOFDECAYS)
882 REAL PPRIMCONVXE(1:N,1:MAXNOOFDECAYS)
883 REAL PBISCONVE(1:N,1:MAXNOOFDECAYS),DX,THETA(1:*)
884
885 INTEGER I,NCH,NCH2,J,K,M,NOOFDECAYS,IX
886
887 LOGICAL JGTK
888
889 COMMON /BLOCK1/ P,PCONVE
890 COMMON /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
891 COMMON /BLOCK3/ PCONVX2E,PPRIMCONVXE,PBIS,PBISCONVE
892 COMMON /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX
893
894 IX=NINT(X/DX+1.0)
895 IX=MIN(IX,NCH)
896 IX=MAX(IX,1)
897

```

```

898 JGTK=J.GT.K
899 IF (JGTK) THEN
900 I=J
901 J=K
902 K=I
903 ENDIF
904 D2FDTH2=0.0
905
906 IF (J.EQ.1) THEN
907 IF (K.EQ.2) THEN
908 D2FDTH2=-PPRIM(IX)/DX
909 ENDIF
910 ELSEIF (J.EQ.2) THEN
911 IF (K.EQ.2) THEN
912 SLASK=THETA(1)*PBIS(IX)
913 DO 10 I=1,M
914 SLASK=SLASK+THETA(I+2)*PBISCONVE(IX,I)
915 10 CONTINUE
916 D2FDTH2=SLASK/(DX**2)
917 ELSEIF (K.GE.3.AND.K.LE.M+2) THEN
918 D2FDTH2=-PPRIMCONVE(IX,K-2)/DX
919 ELSEIF (K.GE.M+3.AND.K.LE.2*M+2) THEN
920 D2FDTH2=PPRIMCONVXE(IX,K-M-2)*THETA(K-M)/DX
921 ENDIF
922 ELSEIF (J.GE.3.AND.J.LE.M+2) THEN
923 IF (K.EQ.M+J) THEN
924 D2FDTH2=-PCONVXE(IX,J-2)
925 ENDIF
926 ELSEIF (J.GE.M+3.AND.J.LE.2*M+2) THEN
927 IF (K.EQ.J) THEN
928 D2FDTH2=THETA(J-M)*PCONVX2E(IX,J-M-2)
929 ENDIF
930 ENDIF
931
932 IF (JGTK) THEN
933 I=J
934 J=K
935 K=I
936 ENDIF
937
938 RETURN
939 END
940
941 SUBROUTINE CALCCONV(THETA,LOCKED,LINEARIZED)
942 C Anders Persson, 1988
943
944 IMPLICIT LOGICAL (A-Z)
945
946 INTEGER N,NHALF,L,MAXNOOFDECAYS
947
948 PARAMETER ( N=512, NHALF=N/2 )
949 PARAMETER ( MAXNOOFDECAYS=3, L=2*MAXNOOFDECAYS+3 )
950
951 REAL PO(1:N),P(1:N),PPRIM(1:N),PBIS(1:N)
952 REAL PCONVE(1:N,1:MAXNOOFDECAYS)
953 REAL PCONVXE(1:N,1:MAXNOOFDECAYS)
954 REAL PCONVX2E(1:N,1:MAXNOOFDECAYS)
955 REAL PPRIMCONVE(1:N,1:MAXNOOFDECAYS)
956 REAL PPRIMCONVXE(1:N,1:MAXNOOFDECAYS)
957 REAL PBISCONVE(1:N,1:MAXNOOFDECAYS),DX,THETA(1:*)

```

```

958
959 COMPLEX FTPO(1:N),SL1(1:N),SL2(1:N),W(1:NHALF)
960
961 INTEGER NCH,NCH2,XP,NDIFF,NOOFDECAYS
962
963 INTEGER*2 I,J
964
965 LOGICAL LINEARIZED,LOCKED(1:*),INV
966
967 COMMON /BLOCK1/ P,PCONVE
968 COMMON /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
969 COMMON /BLOCK3/ PCONVX2E,PPRIMCONVXE,PBIS,PBISCONVE
970 COMMON /BLOCK4/ P0,FTPO,SL1,SL2,W
971 COMMON /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX
972
973 IF (.NOT.LOCKED(2)) THEN
974   IF (THETA(2).EQ.0.0) THEN
975     DO 10 I=1,NCH
976       P(I)=P0(I)
977 10 CONTINUE
978   ELSE
979     DO 20 I=1,NCH2
980       SL1(I)=FTPO(I)
981 20 CONTINUE
982     NDIFF=0
983     CALL MOVEDIFF(SL1,NCH2,THETA(2)/DX,NDIFF)
984     INV=.TRUE.
985     CALL FFT(SL1,W,NCH2,INV)
986     DO 30 I=1,NCH
987       P(I)=REAL(SL1(I))
988 30 CONTINUE
989   ENDIF
990 ENDIF
991
992 IF (.NOT.LOCKED(2)) THEN
993   DO 40 I=1,NCH2
994     SL1(I)=FTPO(I)
995 40 CONTINUE
996   NDIFF=1
997   CALL MOVEDIFF(SL1,NCH2,THETA(2)/DX,NDIFF)
998   INV=.TRUE.
999   CALL FFT(SL1,W,NCH2,INV)
1000   DO 50 I=1,NCH
1001     PPRIM(I)=REAL(SL1(I))
1002 50 CONTINUE
1003   IF (.NOT.LINEARIZED) THEN
1004     DO 60 I=1,NCH2
1005       SL1(I)=FTPO(I)
1006 60 CONTINUE
1007     NDIFF=2
1008     CALL MOVEDIFF(SL1,NCH2,THETA(2)/DX,NDIFF)
1009     INV=.TRUE.
1010     CALL FFT(SL1,W,NCH2,INV)
1011     DO 70 I=1,NCH
1012       PBIS(I)=REAL(SL1(I))
1013 70 CONTINUE
1014   ENDIF
1015 ENDIF
1016
1017 DO 80 I=1,NCH2

```

```

1018   SL1(I)=FTPO(I)
1019 80 CONTINUE
1020   NDIFF=0
1021   CALL MOVEDIFF(SL1,NCH2,THETA(2)/DX,NDIFF)
1022
1023 DO 150 J=1,NOOFDECAYS
1024   XP=0
1025   CALL CALCFEXP(THETA(NOOFDECAYS+J+2),XP)
1026   DO 90 I=1,NCH2
1027     SL2(I)=SL1(I)*SL2(I)
1028 90 CONTINUE
1029   INV=.TRUE.
1030   CALL FFT(SL2,W,NCH2,INV)
1031   DO 100 I=1,NCH
1032     PCONVE(I,J)=REAL(SL2(I))
1033 100 CONTINUE
1034   IF (.NOT.LOCKED(NOOFDECAYS+J+2)) THEN
1035     XP=1
1036     CALL CALCFEXP(THETA(NOOFDECAYS+J+2),XP)
1037     DO 110 I=1,NCH2
1038       SL2(I)=SL1(I)*SL2(I)
1039 110 CONTINUE
1040     INV=.TRUE.
1041     CALL FFT(SL2,W,NCH2,INV)
1042     DO 120 I=1,NCH
1043       PCONVXE(I,J)=REAL(SL2(I))
1044 120 CONTINUE
1045     IF (.NOT.LINEARIZED) THEN
1046       XP=2
1047       CALL CALCFEXP(THETA(NOOFDECAYS+J+2),XP)
1048       DO 130 I=1,NCH2
1049         SL2(I)=SL1(I)*SL2(I)
1050 130 CONTINUE
1051       INV=.TRUE.
1052       CALL FFT(SL2,W,NCH2,INV)
1053       DO 140 I=1,NCH
1054         PCONVX2E(I,J)=REAL(SL2(I))
1055 140 CONTINUE
1056     ENDIF
1057   ENDIF
1058 150 CONTINUE
1059
1060 IF (.NOT.LOCKED(2)) THEN
1061   DO 160 I=1,NCH2
1062     SL1(I)=FTPO(I)
1063 160 CONTINUE
1064   NDIFF=1
1065   CALL MOVEDIFF(SL1,NCH2,THETA(2)/DX,NDIFF)
1066
1067 DO 210 J=1,NOOFDECAYS
1068   XP=0
1069   CALL CALCFEXP(THETA(NOOFDECAYS+J+2),XP)
1070   DO 170 I=1,NCH2
1071     SL2(I)=SL1(I)*SL2(I)
1072 170 CONTINUE
1073   INV=.TRUE.
1074   CALL FFT(SL2,W,NCH2,INV)
1075   DO 180 I=1,NCH
1076     PPRIMCONVE(I,J)=REAL(SL2(I))
1077 180 CONTINUE

```

```

1078      IF ((.NOT.LINEARIZED).AND.(.NOT.LOCKED(NOOFDECAYS+J+2))) THEN
1079          XP=1
1080          CALL CALCFEXP(THETA(NOOFDECAYS+J+2),XP)
1081          DO 190 I=1,NCH2
1082              SL2(I)=SL1(I)*SL2(I)
1083 190      CONTINUE
1084          INV=.TRUE.
1085          CALL FFT(SL2,W,NCH2,INV)
1086          DO 200 I=1,NCH
1087              PPRIMCONVXE(I,J)=REAL(SL2(I))
1088 200      CONTINUE
1089      ENDIF
1090 210      CONTINUE
1091      ENDIF
1092
1093      IF ((.NOT.LINEARIZED).AND.(.NOT.LOCKED(2))) THEN
1094          DO 220 I=1,NCH2
1095              SL1(I)=FTP0(I)
1096 220      CONTINUE
1097          NDIFF=2
1098          CALL MOVEDIFF(SL1,NCH2,THETA(2)/DX,NDIFF)
1099
1100          DO 250 J=1,NOOFDECAYS
1101              XP=0
1102              CALL CALCFEXP(THETA(NOOFDECAYS+J+2),XP)
1103              DO 230 I=1,NCH2
1104                  SL2(I)=SL1(I)*SL2(I)
1105 230          CONTINUE
1106              INV=.TRUE.
1107              CALL FFT(SL2,W,NCH2,INV)
1108              DO 240 I=1,NCH
1109                  PBISCONVE(I,J)=REAL(SL2(I))
1110 240          CONTINUE
1111 250          CONTINUE
1112      ENDIF
1113
1114      RETURN
1115      END
1116
1117      SUBROUTINE MOVEDIFF(A,N,DC,NDIFF)
1118 C      Anders Persson, 1988
1119
1120      IMPLICIT LOGICAL (A-Z)
1121      INTEGER N,NDIFF
1122      INTEGER*2 K,NHALF,MAXN
1123      COMPLEX A(0:*),SL1,SL3
1124      REAL DC,PHSTEP,TWOPI,SL2
1125      PARAMETER ( TWOPI=6.283185308 )
1126      LOGICAL INV
1127
1128      NHALF=N/2
1129      IF (DC.NE.0.0) THEN
1130          SL1=CEXP(CMPLX(0.0,TWOPI*(0.5*DC-DC/N)))
1131          PHSTEP=-TWOPI*DC/N
1132          DO 10 K=0,NHALF-2
1133              SL2=PHSTEP*K
1134              SL3=CMPLX(COS(SL2),SIN(SL2))
1135              A(K)=A(K)*SL3
1136              A(K+NHALF+1)=A(K+NHALF+1)*SL1*SL3
1137 10      CONTINUE

```

```

1138      DO 20 K=NHALF-1,NHALF
1139          A(K)=A(K)*CEXP(CMPLX(0.0,PHSTEP*K))
1140 20      CONTINUE
1141      ENDIF
1142
1143      IF (NDIFF.GT.0) THEN
1144          PHSTEP=TWOPI/N
1145          MAXN=MIN(10,N/4+2)
1146
1147          A(0)=CMPLX(0.0,0.0)
1148          DO 30 K=1,MAXN
1149              SL2=PHSTEP*K
1150              A(K)=A(K)*CMPLX(0.0,SL2)**NDIFF
1151              A(N-K)=A(N-K)*CMPLX(0.0,-SL2)**NDIFF
1152 30          CONTINUE
1153          SL1=CMPLX(0.0,PHSTEP*(MAXN+1))**NDIFF
1154          DO 40 K=MAXN+1,NHALF
1155              A(K)=A(K)*SL1
1156 40          CONTINUE
1157          SL1=CMPLX(0.0,-PHSTEP*(MAXN+1))**NDIFF
1158          DO 50 K=NHALF+1,N-MAXN-1
1159              A(K)=A(K)*SL1
1160 50          CONTINUE
1161      ENDIF
1162
1163      RETURN
1164      END
1165
1166      SUBROUTINE CALCFEXP(THETA,XP)
1167 C      Anders Persson, 1988
1168
1169      IMPLICIT LOGICAL (A-Z)
1170      INTEGER N,NHALF
1171      PARAMETER ( N=512, NHALF=N/2 )
1172      REAL PO(1:N),DX,THETA
1173      REAL SL1,SL2,SL3,SL4,SL5,SL6,SL7,SL8,SL10,SL11
1174      COMPLEX FTP0(1:N),S1(1:N),S2(1:N),W(1:NHALF),Z,SL9,SL12
1175      INTEGER NCH,NCHHALF,NCHP2,NOOFDECAYS,XP,NCH2
1176      INTEGER*2 I
1177
1178      COMMON /BLOCK4/ PO,FTP0,S1,S2,W
1179      COMMON /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX
1180
1181      SL1=-DX*THETA
1182      SL2=-2.0*3.141592654/NCH2
1183      SL3=EXP(SL1)
1184      SL4=SL3**NCH2
1185      SL5=1.0-SL4
1186      SL6=DX**2
1187      SL7=NCH2**2
1188      SL8=(NCH2-1)**2
1189      SL10=NCH2
1190      SL11=NCH2-1
1191      NCHP2=NCH2+2
1192      NCHHALF=NCH2/2
1193
1194      IF (XP.EQ.0) THEN
1195          DO 10 I=1,NCHHALF
1196              Z=SL3*W(I)
1197              S2(I)=SL5/(1.0-Z)

```

```

1198 10 CONTINUE
1199 Z=SL3*CMPLX(-1.0,0.0)
1200 S2(NCHHALF+1)=SL5/(1.0-Z)
1201 DO 20 I=NCH2,NCHHALF,-1
1202 S2(I)=CONJG(S2(NCHP2-I))
1203 20 CONTINUE
1204 ELSEIF (XP.EQ.1) THEN
1205 DO 30 I=1,NCHHALF
1206 Z=SL3*W(I)
1207 SL9=1.0-Z
1208 S2(I)=DX*(Z-SL4*(SL10-SL11*Z))/(SL9**2)
1209 30 CONTINUE
1210 Z=SL3*CMPLX(-1.0,0.0)
1211 SL9=1.0-Z
1212 S2(NCHHALF+1)=DX*(Z-SL4*(SL10-SL11*Z))/(SL9**2)
1213 DO 40 I=NCH2,NCHHALF,-1
1214 S2(I)=CONJG(S2(NCHP2-I))
1215 40 CONTINUE
1216 ELSEIF (XP.EQ.2) THEN
1217 DO 50 I=1,NCHHALF
1218 Z=SL3*W(I)
1219 SL9=1.0-Z
1220 SL12=Z+Z*Z+SL4*(2.0*Z+(SL8*Z-SL7)*SL9)
1221 S2(I)=SL6*SL12/(SL9**3)
1222 50 CONTINUE
1223 Z=SL3*CMPLX(-1.0,0.0)
1224 SL9=1.0-Z
1225 SL12=Z+Z*Z+SL4*(2.0*Z+(SL8*Z-SL7)*SL9)
1226 S2(NCHHALF+1)=SL6*SL12/(SL9**3)
1227 DO 60 I=NCH2,NCHHALF,-1
1228 S2(I)=CONJG(S2(NCHP2-I))
1229 60 CONTINUE
1230 ENDIF
1231 RETURN
1232 END
1233

```

TESTMD

Programmet testar rutinen MOVEDIFF, använd i programmet PCONEXP för att translatera och derivera laserpulsen. MOVEDIFF använder reglerna för derivation och translation i transformplanet för att beräkna dessa [8,9]:

<u>Originalfunktion</u>	<u>Fouriertransform</u>
$f(t)$	$F(\omega)$
$f(t-a)$	$\exp(-ia\omega) \cdot F(\omega)$
$f'(t)$	$i\omega \cdot F(\omega)$

Dessutom använder rutinen symmetriegenskaper hos den diskreta Fouriertransformen för att minimera räknearbetet. Programmet beräknar vektorn för den flyttade pulsen och vektorerna för första och andraderivatan.

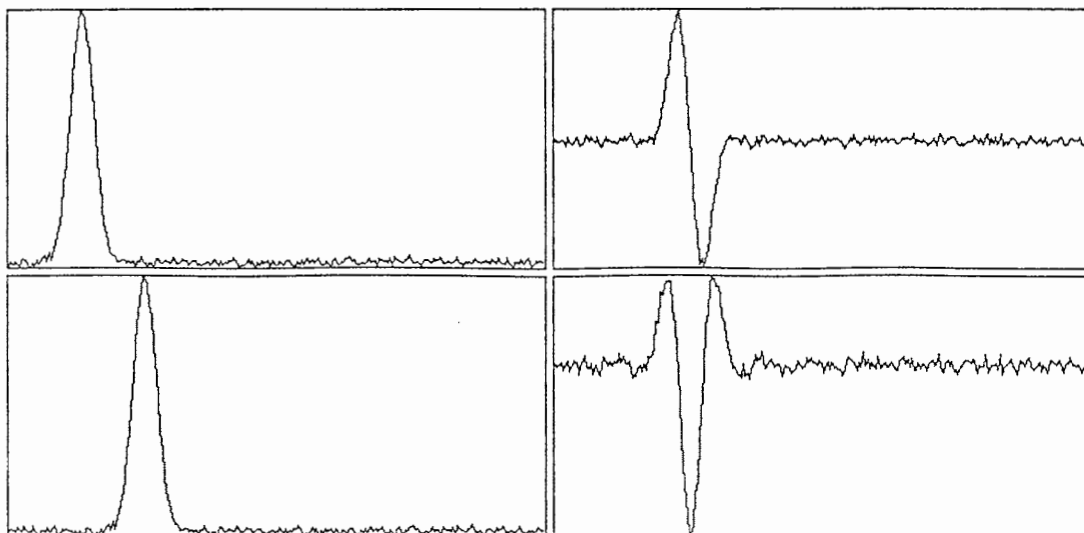
Utskrift från testkörning av programmet TESTMD:

TESTMD.RUN

MOVE PULSE (CH) : 30
NCH (16..1024, 2**K) : 256
TIME FOR MOVE : 0.28 s.
TIME FOR MOVE+1-DIFF : 0.39 s.
TIME FOR MOVE+2-DIFF : 0.38 s.

F1: PLOT
F2: NEW CALC.
F3: STOP

<F1>, <Printscreen>



<F3>

```

1      PROGRAM TESTMD
2 C    Anders Persson, 1988
3
4      INTEGER    N,NHALF,NCH,NCHHALF,NCHP2,NDIFF
5      PARAMETER ( N=1024, NHALF=N/2 )
6      REAL      PL1(1:N),PL2(1:N),PL3(1:N),PL4(1:N)
7      REAL      DX,RP1(1:2),RP2(1:2),RP3(1:2),RP4(1:N)
8      REAL      THETA,TIME,FWHM
9      COMPLEX   S1(1:N),S2(1:N),S3(1:N),S4(1:N)
10     COMPLEX   W(1:NHALF),SLASK,Z,SL9
11     INTEGER*2  ANS,KBDCHK,KBDINC,PAR(1:9),HO,MI,SE,HN,I
12     LOGICAL   INV
13
14 1    PAR(1)=0
15     PAR(2)=175
16     PAR(3)=350
17     PAR(4)=170
18     PAR(5)=1
19     PAR(6)=0
20     PAR(7)=1
21     PAR(8)=0
22     PAR(9)=1
23
24 2    WRITE(*,*) 'MOVE PULSE (CH):'
25     READ(*,*,ERR=2) THETA
26
27 3    WRITE(*,*) 'NCH (16..1024, 2**K):'
28     READ(*,*,ERR=3) NCH
29     IF (NCH.LT.16.OR.NCH.GT.1024) THEN
30         GOTO 3
31     ENDIF
32     IF (NCH.NE.16.AND.NCH.NE.32.AND.NCH.NE.64.AND.NCH.NE.128.AND.
33         f NCH.NE.256.AND.NCH.NE.512.AND.NCH.NE.1024) THEN
34         GOTO 3
35     ENDIF
36
37     NCHHALF=NCH/2
38     FWHM=NCH/20+1
39     FWHM=LOG(16.0)/FWHM**2
40
41     DO 10 I=1,NCH
42         S1(I)=CMPLX(EXP(-MIN(FWHM*(I-NCH/7.0)**2,85.0)),0.0)
43         S1(I)=1000.0*S1(I)
44         S1(I)=S1(I)+NORMAL(0.0,10.0)
45         S2(I)=S1(I)
46 10    CONTINUE
47
48     CALL CALCW(W,NCH)
49     INV=.FALSE.
50     CALL FFT(S2,W,NCH,INV)
51
52     DO 20 I=1,NCH
53         S3(I)=S2(I)
54         S4(I)=S2(I)
55 20    CONTINUE
56
57     NDIFF=0
58     CALL GETTIM(HO,MI,SE,HN)
59     TIME=HO*3600.0+MI*60.0+SE+HN*0.01
60
61     CALL MOVEDIFF(S2,NCH,THETA,NDIFF)
62
63     CALL GETTIM(HO,MI,SE,HN)
64     TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
65
66     WRITE(*,999) 'TIME FOR MOVE:',TIME
67 999  FORMAT(' ',A,F8.2)
68     PAUSE
69     INV=.TRUE.
70     CALL FFT(S2,W,NCH,INV)
71
72     NDIFF=1
73     CALL GETTIM(HO,MI,SE,HN)
74     TIME=HO*3600.0+MI*60.0+SE+HN*0.01
75
76     CALL MOVEDIFF(S3,NCH,THETA,NDIFF)
77
78     CALL GETTIM(HO,MI,SE,HN)
79     TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
80     WRITE(*,999) 'TIME FOR MOVE+1-DIFF:',TIME
81     PAUSE
82     INV=.TRUE.
83     CALL FFT(S3,W,NCH,INV)
84
85     NDIFF=2
86     CALL GETTIM(HO,MI,SE,HN)
87     TIME=HO*3600.0+MI*60.0+SE+HN*0.01
88
89     CALL MOVEDIFF(S4,NCH,THETA,NDIFF)
90
91     CALL GETTIM(HO,MI,SE,HN)
92     TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
93     WRITE(*,999) 'TIME FOR MOVE+2-DIFF:',TIME
94     PAUSE
95     INV=.TRUE.
96     CALL FFT(S4,W,NCH,INV)
97
98 70   CALL GMODE
99     CALL TMODE
100
101     WRITE(*,1000)
102 1000 FORMAT(///
103         ft10,'F1: PLOT '//
104         ft10,'F2: NEW CALC. '//
105         ft10,'F3: STOP')
106
107 80   ANS=KBDINC()
108     ANS=ANS-1058
109
110     IF (ANS.EQ.1) THEN
111         RP1(1)=REAL(S1(1))
112         RP1(1)=REAL(S1(1))
113         RP2(1)=REAL(S2(1))
114         RP2(1)=REAL(S2(1))
115         RP3(1)=REAL(S3(1))
116         RP3(1)=REAL(S3(1))
117         RP4(1)=REAL(S4(1))
    
```

```

118 RP4(1)=REAL(S4(1))
119
120 DO 90 I=1,NCH
121 PL1(I)=REAL(S1(I))
122 PL2(I)=REAL(S2(I))
123 PL3(I)=REAL(S3(I))
124 PL4(I)=REAL(S4(I))
125
126 RP1(1)=MAX(RP1(1),PL1(I))
127 RP1(2)=MIN(RP1(2),PL1(I))
128 RP2(1)=MAX(RP2(1),PL2(I))
129 RP2(2)=MIN(RP2(2),PL2(I))
130 RP3(1)=MAX(RP3(1),PL3(I))
131 RP3(2)=MIN(RP3(2),PL3(I))
132 RP4(1)=MAX(RP4(1),PL4(I))
133 RP4(2)=MIN(RP4(2),PL4(I))
134 90 CONTINUE
135 PAR(8)=0
136 PAR(1)=0
137 PAR(2)=175
138 CALL GMODE
139 CALL PVEC(PL1,NCH,PAR,RP1)
140 PAR(8)=1
141 PAR(2)=0
142 CALL PVEC(PL2,NCH,PAR,RP2)
143 PAR(1)=355
144 PAR(2)=175
145 CALL PVEC(PL3,NCH,PAR,RP3)
146 PAR(2)=0
147 CALL PVEC(PL4,NCH,PAR,RP4)
148 100 ANS=KBDCHK()
149 IF (ANS.EQ.0) THEN
150 GOTO 100
151 ENDF
152 ANS=KBDINC()
153 GOTO 70
154 ELSEIF (ANS.EQ.2) THEN
155 GOTO 1
156 ELSEIF (ANS.EQ.3) THEN
157 STOP
158 ENDF
159
160 GOTO 80
161 END
162
163 SUBROUTINE MOVEDIFF(A,N,DC,NDIFF)
164 C Anders Persson, 1988
165
166 IMPLICIT LOGICAL (A-Z)
167 INTEGER N,NDIFF
168 INTEGER*2 K,NHALF,MAXN
169 COMPLEX A(0:*),SL1,SL3
170 REAL DC,PHSTEP,TWOPI,SL2
171 PARAMETER ( TWOPI=6.283185308 )
172 LOGICAL INV
173
174 NHALF=N/2
175 IF (DC.NE.0.0) THEN
176 SL1=CEXP(CMPLX(0.0,TWOPI*(0.5*DC-DC/N)))
177 PHSTEP=-TWOPI*DC/N

```

```

178
179 DO 10 K=0,NHALF-2
180 SL2=PHSTEP*K
181 SL3=CMPLX(COS(SL2),SIN(SL2))
182 A(K)=A(K)*SL3
183 A(K+NHALF+1)=A(K+NHALF+1)*SL1*SL3
184 10 CONTINUE
185
186 DO 20 K=NHALF-1,NHALF
187 A(K)=A(K)*CEXP(CMPLX(0.0,PHSTEP*K))
188 20 CONTINUE
189 ENDF
190
191 IF (NDIFF.GT.0) THEN
192 PHSTEP=TWOPI/N
193 MAXN=MIN(10,N/4+2)
194
195 A(0)=CMPLX(0.0,0.0)
196 DO 30 K=1,MAXN
197 SL2=PHSTEP*K
198 A(K)=A(K)*CMPLX(0.0,SL2)**NDIFF
199 A(N-K)=A(N-K)*CMPLX(0.0,-SL2)**NDIFF
200 30 CONTINUE
201
202 SL1=CMPLX(0.0,PHSTEP*(MAXN+1))**NDIFF
203 DO 40 K=MAXN+1,NHALF
204 A(K)=A(K)*SL1
205 40 CONTINUE
206
207 SL1=CMPLX(0.0,-PHSTEP*(MAXN+1))**NDIFF
208 DO 50 K=NHALF+1,N-MAXN-1
209 A(K)=A(K)*SL1
210 50 CONTINUE
211
212 ENDF
213
214 RETURN
215 END

```

TFEXP

Programmet testar rutinen CALCFEXP använd i PCONEXP-programmet för att beräkna den diskreta Fouriertransformen av exponentialer från de analytiska uttrycken. Exekveringstiden jämförs med tiden för en Fouriertransform. Figurerna från testexekveringen är skärmdumpar från en test med 1024 kanaler. Vänstra delen av bilderna är real och imaginärdelen beräknade med hjälp av FFT-rutinen och till höger återfinns samma kurvor beräknade med CALCFEXP-rutinen. Rutinen använder sig av följande samband för att beräkna Fouriertransformerna:

Den diskreta fouriertransformen (DFT) för $\exp(-\theta \cdot Dx \cdot k)$ är [8].

$$\text{DFT}(\exp(-\theta \cdot Dx \cdot k)) = \sum_{k=0}^{n-1} \exp(-\theta \cdot Dx \cdot k) \cdot \exp(-i2\pi \cdot q \cdot k/n) \quad (111)$$

sätt $\exp(-\theta \cdot Dx - i2\pi \cdot q/n) = z \Rightarrow$

$$\text{DFT}(\exp(-\theta \cdot Dx \cdot k)) = \sum_{k=0}^{n-1} z^k = \frac{1-z^n}{1-z} \quad (112)$$

$$\begin{aligned} \text{DFT}(k \cdot \exp(-\theta \cdot Dx \cdot k)) &= \dots = z \cdot \frac{1-n \cdot z^{n-1} + (n-1) \cdot z^{n+1}}{(1-z)^2} = \\ &= z \cdot \frac{d}{dz} \left[\frac{1-z^n}{1-z} \right] \end{aligned} \quad (113)$$

$$\text{DFT}(k^2 \cdot \exp(-\theta \cdot Dx \cdot k)) = z \cdot \frac{d}{dz} \left[z \cdot \frac{d}{dz} \left[\frac{1-z^n}{1-z} \right] \right] \quad (114)$$

$$\text{DFT}(k^p \cdot \exp(-\theta \cdot Dx \cdot k)) = \left[z \cdot \frac{d}{dz} \right]^p \left[\frac{1-z^n}{1-z} \right] \quad (115)$$

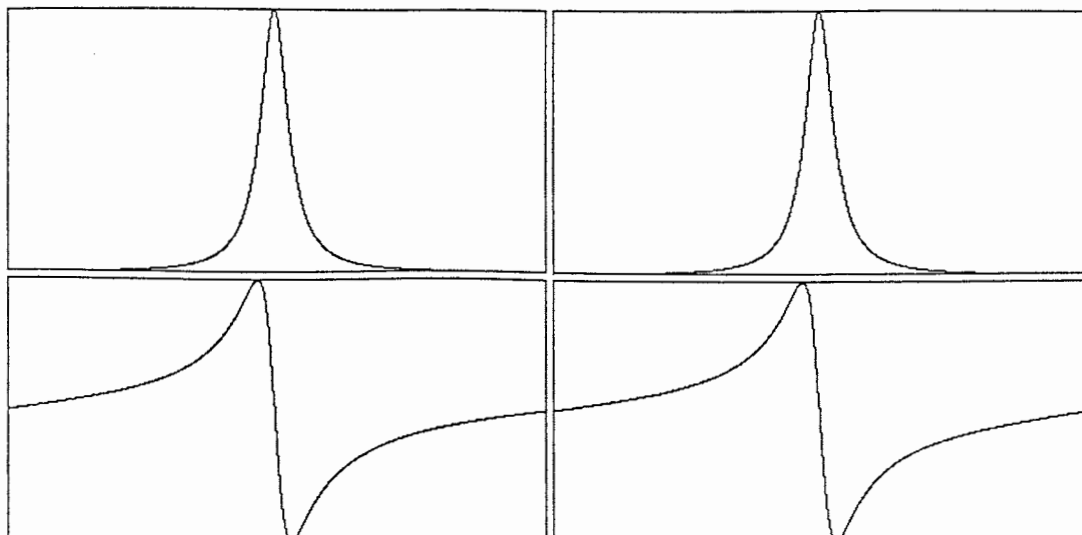
(operatorn: $z \cdot (d/dz)$ opererar p gånger på uttrycket: $(1-z^n)/(1-z)$)

Utskrift från testkörning av programmet TFEXP:

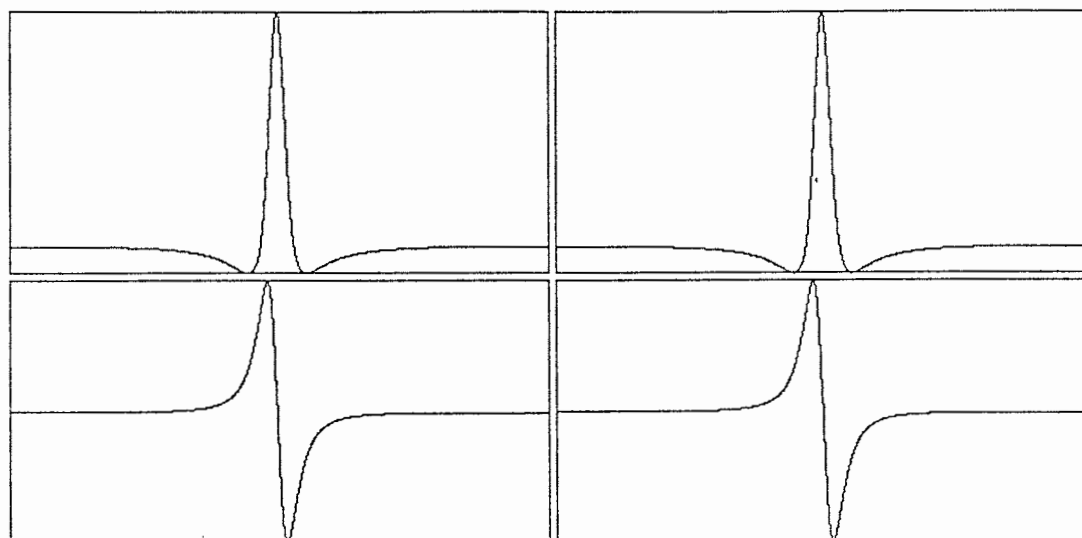
```
THETA (GT 0, LT 1000)      : 200
NCH (16..1024, 2**K)      : 1024
TIME FOR FFT               : 9.94 s.
TIME FOR CALCULATING FFT EXP : 0.55 s.
TIME FOR CALCULATING FFT X*EXP : 0.93 s.
TIME FOR CALCULATING FFT X2*EXP : 1.37 s.
```

```
F1: PLOT EXP
F2: PLOT X*EXP
F3: PLOT X**2*EXP
F4: NEW CALC.
F5: STOP
```

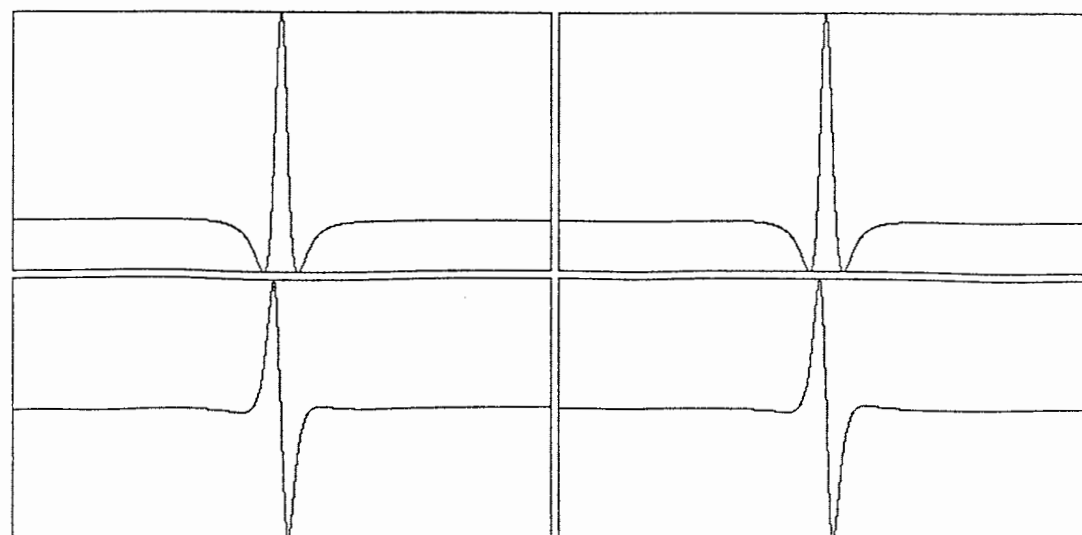

<F1>, <Printscreen>



<F2>, <Printscreen>



<F3>, <Printscreen>



<F4>

```

1 PROGRAM TFEXP
2 C Anders Persson, 1988
3
4 INTEGER N,NHALF,NCH,NCHHALF,NCHP2,NCH2,NOOFDECAYS
5 PARAMETER ( N=1024, NHALF=N/2 )
6 REAL PL1(1:N),PL2(1:N),PL3(1:N),PL4(1:N)
7 REAL DX,RPRE(1:2),RPIM(1:2),THETA,TIME
8 COMPLEX S1(1:N),S2(1:N),S3(1:N)
9 COMPLEX S4(1:N),S5(1:N),S6(1:N)
10 COMPLEX W(1:NHALF)
11 INTEGER*2 ANS,KBDCHK,KBDINC,PAR(1:9),HO,MI,SE,HN,I
12 LOGICAL INV
13 COMMON /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX,W
14
15 1 PAR(1)=0
16 PAR(2)=175
17 PAR(3)=350
18 PAR(4)=170
19 PAR(5)=1
20 PAR(6)=0
21 PAR(7)=1
22 PAR(8)=0
23 PAR(9)=1
24
25 2 WRITE(*,*) 'THETA (GT 0, LT 1000):'
26 READ(*,*,ERR=2) THETA
27 IF (THETA.LE.0.0.OR.THETA.GT.1000.0) THEN
28 GOTO 2
29 ENDIF
30
31 3 WRITE(*,*) 'NCH (16..1024, 2**K):'
32 READ(*,*,ERR=3) NCH
33 IF (NCH.LT.16.OR.NCH.GT.1024) THEN
34 GOTO 3
35 ENDIF
36 IF (NCH.NE.16.AND.NCH.NE.32.AND.NCH.NE.64.AND.NCH.NE.128.AND.
37 f NCH.NE.256.AND.NCH.NE.512.AND.NCH.NE.1024) THEN
38 GOTO 3
39 ENDIF
40
41 NCHHALF=NCH/2
42 NCH2=NCH
43
44 DX=1.0/(NCH-1)
45 DO 10 I=1,NCH
46 S1(I)=CMPLX(EXP(-MIN(THETA*(I-1)*DX,85.0)),0.0)
47 10 CONTINUE
48
49 DO 20 I=1,NCH
50 S2(I)=S1(I)*(I-1)*DX
51 20 CONTINUE
52
53 DO 30 I=1,NCH
54 S3(I)=S1(I)*((I-1)*DX)**2
55 30 CONTINUE
56
57 CALL CALCW(W,NCH)
    
```

```

58 INV=.FALSE.
59 CALL FFT(S1,W,NCH,INV)
60 CALL FFT(S2,W,NCH,INV)
61
62 CALL GETTIM(HO,MI,SE,HN)
63 TIME=HO*3600.0+MI*60.0+SE+HN*0.01
64 CALL FFT(S3,W,NCH,INV)
65 CALL GETTIM(HO,MI,SE,HN)
66 TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
67 WRITE(*,999) 'TIME FOR FFT:',TIME
68 999 FORMAT(' ',A,F8.2)
69 PAUSE
70
71 CALL GETTIM(HO,MI,SE,HN)
72 TIME=HO*3600.0+MI*60.0+SE+HN*0.01
73
74 CALL CALCFEXP(S4,THETA,0)
75
76 CALL GETTIM(HO,MI,SE,HN)
77 TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
78 WRITE(*,999) 'TIME FOR CALCULATING FFT EXP:',TIME
79 PAUSE
80
81 CALL GETTIM(HO,MI,SE,HN)
82 TIME=HO*3600.0+MI*60.0+SE+HN*0.01
83
84 CALL CALCFEXP(S5,THETA,1)
85
86 CALL GETTIM(HO,MI,SE,HN)
87 TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
88 WRITE(*,999) 'TIME FOR CALCULATING FFT X*EXP:',TIME
89 PAUSE
90
91 CALL GETTIM(HO,MI,SE,HN)
92 TIME=HO*3600.0+MI*60.0+SE+HN*0.01
93
94 CALL CALCFEXP(S6,THETA,2)
95
96 CALL GETTIM(HO,MI,SE,HN)
97 TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
98 WRITE(*,999) 'TIME FOR CALCULATING FFT X2*EXP:',TIME
99 PAUSE
100
101 70 CALL GMODE
102 CALL TMODE
103
104 WRITE(*,1000)
105 1000 FORMAT(////
106 fT10,'F1: PLOT EXP'/
107 fT10,'F2: PLOT X*EXP'/
108 fT10,'F3: PLOT X**2*EXP'/
109 fT10,'F4: NEW CALC.'/
110 fT10,'F5: STOP')
111
112 80 ANS=KBDINC()
113 ANS=ANS-1058
114
115 IF (ANS.EQ.1) THEN
116 RPRE(1)=REAL(S1(1))
117 RPIM(1)=AIMAG(S1(1))
    
```

```

118 RPRE(2)=REAL(S1(1))
119 RPIM(2)=AIMAG(S1(1))
120
121 DO 90 I=1,NCH
122 IF (I.GT.NCHHALF) THEN
123 PL1(I)=REAL(S1(I-NCHHALF))
124 PL2(I)=AIMAG(S1(I-NCHHALF))
125 PL3(I)=REAL(S4(I-NCHHALF))
126 PL4(I)=AIMAG(S4(I-NCHHALF))
127 ELSE
128 PL1(I)=REAL(S1(NCHHALF+I))
129 PL2(I)=AIMAG(S1(NCHHALF+I))
130 PL3(I)=REAL(S4(NCHHALF+I))
131 PL4(I)=AIMAG(S4(NCHHALF+I))
132 ENDIF
133 RPRE(1)=MAX(RPRE(1),PL1(I))
134 RPRE(2)=MIN(RPRE(2),PL1(I))
135 RPIM(1)=MAX(RPIM(1),PL2(I))
136 RPIM(2)=MIN(RPIM(2),PL2(I))
137 90 CONTINUE
138 PAR(8)=0
139 PAR(1)=0
140 PAR(2)=175
141 CALL GMODE
142 CALL PVEC(PL1,NCH,PAR,RPRE)
143 PAR(8)=1
144 PAR(2)=0
145 CALL PVEC(PL2,NCH,PAR,RPIM)
146 PAR(1)=355
147 PAR(2)=175
148 CALL PVEC(PL3,NCH,PAR,RPRE)
149 PAR(2)=0
150 CALL PVEC(PL4,NCH,PAR,RPIM)
151 100 ANS=KBDCHK()
152 IF (ANS.EQ.0) THEN
153 GOTO 100
154 ENDIF
155 ANS=KBDINC()
156 GOTO 70
157 ELSEIF (ANS.EQ.2) THEN
158 RPRE(1)=REAL(S2(1))
159 RPIM(1)=AIMAG(S2(1))
160 RPRE(2)=REAL(S2(1))
161 RPIM(2)=AIMAG(S2(1))
162
163 DO 110 I=1,NCH
164 IF (I.GT.NCHHALF) THEN
165 PL1(I)=REAL(S2(I-NCHHALF))
166 PL2(I)=AIMAG(S2(I-NCHHALF))
167 PL3(I)=REAL(S5(I-NCHHALF))
168 PL4(I)=AIMAG(S5(I-NCHHALF))
169 ELSE
170 PL1(I)=REAL(S2(NCHHALF+I))
171 PL2(I)=AIMAG(S2(NCHHALF+I))
172 PL3(I)=REAL(S5(NCHHALF+I))
173 PL4(I)=AIMAG(S5(NCHHALF+I))
174 ENDIF
175
176 RPRE(1)=MAX(RPRE(1),PL1(I))
177 RPRE(2)=MIN(RPRE(2),PL1(I))

```

```

178 RPIM(1)=MAX(RPIM(1),PL2(I))
179 RPIM(2)=MIN(RPIM(2),PL2(I))
180 110 CONTINUE
181 PAR(8)=0
182 PAR(1)=0
183 PAR(2)=175
184 CALL GMODE
185 CALL PVEC(PL1,NCH,PAR,RPRE)
186 PAR(8)=1
187 PAR(2)=0
188 CALL PVEC(PL2,NCH,PAR,RPIM)
189 PAR(1)=355
190 PAR(2)=175
191 CALL PVEC(PL3,NCH,PAR,RPRE)
192 PAR(2)=0
193 CALL PVEC(PL4,NCH,PAR,RPIM)
194 120 ANS=KBDCHK()
195 IF (ANS.EQ.0) THEN
196 GOTO 120
197 ENDIF
198 ANS=KBDINC()
199 GOTO 70
200 ELSEIF (ANS.EQ.3) THEN
201 RPRE(1)=REAL(S3(1))
202 RPIM(1)=AIMAG(S3(1))
203 RPRE(2)=REAL(S3(1))
204 RPIM(2)=AIMAG(S3(1))
205
206 DO 130 I=1,NCH
207 IF (I.GT.NCHHALF) THEN
208 PL1(I)=REAL(S3(I-NCHHALF))
209 PL2(I)=AIMAG(S3(I-NCHHALF))
210 PL3(I)=REAL(S6(I-NCHHALF))
211 PL4(I)=AIMAG(S6(I-NCHHALF))
212 ELSE
213 PL1(I)=REAL(S3(NCHHALF+I))
214 PL2(I)=AIMAG(S3(NCHHALF+I))
215 PL3(I)=REAL(S6(NCHHALF+I))
216 PL4(I)=AIMAG(S6(NCHHALF+I))
217 ENDIF
218
219 RPRE(1)=MAX(RPRE(1),PL1(I))
220 RPRE(2)=MIN(RPRE(2),PL1(I))
221 RPIM(1)=MAX(RPIM(1),PL2(I))
222 RPIM(2)=MIN(RPIM(2),PL2(I))
223 130 CONTINUE
224 PAR(8)=0
225 PAR(1)=0
226 PAR(2)=175
227 CALL GMODE
228 CALL PVEC(PL1,NCH,PAR,RPRE)
229 PAR(8)=1
230 PAR(2)=0
231 CALL PVEC(PL2,NCH,PAR,RPIM)
232 PAR(1)=355
233 PAR(2)=175
234 CALL PVEC(PL3,NCH,PAR,RPRE)
235 PAR(2)=0
236 CALL PVEC(PL4,NCH,PAR,RPIM)
237 140 ANS=KBDCHK()

```

```

238     IF (ANS.EQ.0) THEN
239       GOTO 140
240     ENDIF
241     ANS=KBDINC()
242     GOTO 70
243   ELSEIF (ANS.EQ.4) THEN
244     GOTO 1
245   ELSEIF (ANS.EQ.5) THEN
246     STOP
247   ENDIF
248
249   GOTO 80
250   END
251
252   SUBROUTINE CALCFEXP(A,THETA,XP)
253 C   Anders Persson, 1988
254
255   IMPLICIT LOGICAL (A-Z)
256   INTEGER N,NHALF,NCH,NCHHALF,NCHP2,NCH2,NOOFDECAYS,XP
257   PARAMETER ( N=1024, NHALF=N/2 )
258   REAL DX,THETA,SL1,SL2,SL3,SL4,SL5,SL6,SL7,SL8,SL10,SL11
259   COMPLEX A(1:*),W(1:NHALF),Z,SL9,SL12
260   INTEGER*2 I
261   COMMON /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX,W
262
263   SL1=-DX*THETA
264   SL2=-2.0*3.141592654/NCH2
265   SL3=EXP(SL1)
266   SL4=SL3**NCH2
267   SL5=1.0-SL4
268   SL6=DX**2
269   SL7=NCH2**2
270   SL8=(NCH2-1)**2
271   SL10=NCH2
272   SL11=NCH2-1
273   NCHP2=NCH2+2
274   NCHHALF=NCH2/2
275
276   IF (XP.EQ.0) THEN
277     DO 10 I=1,NCHHALF
278       Z=SL3*W(I)
279       A(I)=SL5/(1.0-Z)
280     CONTINUE
281
282     Z=SL3*CMPLX(-1.0,0.0)
283     A(NCHHALF+1)=SL5/(1.0-Z)
284     DO 20 I=NCH2,NCHHALF,-1
285       A(I)=CONJG(A(NCHP2-I))
286     CONTINUE
287
288   ELSEIF (XP.EQ.1) THEN
289     DO 30 I=1,NCHHALF
290       Z=SL3*W(I)
291       SL9=1.0-Z
292       A(I)=DX*(Z-SL4*(SL10-SL11*Z))/(SL9**2)
293     CONTINUE
294
295     Z=SL3*CMPLX(-1.0,0.0)
296     SL9=1.0-Z
297     A(NCHHALF+1)=DX*(Z-SL4*(SL10-SL11*Z))/(SL9**2)

```

```

298
299     DO 40 I=NCH2,NCHHALF,-1
300       A(I)=CONJG(A(NCHP2-I))
301     CONTINUE
302
303   ELSEIF (XP.EQ.2) THEN
304     DO 50 I=1,NCHHALF
305       Z=SL3*W(I)
306       SL9=1.0-Z
307       SL12=Z+Z*Z+SL4*(2.0*Z+(SL8*Z-SL7)*SL9)
308       A(I)=SL6*SL12/(SL9**3)
309     CONTINUE
310
311     Z=SL3*CMPLX(-1.0,0.0)
312     SL9=1.0-Z
313     SL12=Z+Z*Z+SL4*(2.0*Z+(SL8*Z-SL7)*SL9)
314     A(NCHHALF+1)=SL6*SL12/(SL9**3)
315
316     DO 60 I=NCH2,NCHHALF,-1
317       A(I)=CONJG(A(NCHP2-I))
318     CONTINUE
319
320   ENDIF
321
322   RETURN
323   END

```

Hjälprutiner:

PVEC

Rutinen PVEC används av TESTMD och TFEXP programmen för att rita en kurva på skärmen.

CALCW

Rutinen beräknar en tabell över de exponentialtermer som behövs vid beräkning av Fouriertransformen [8].

FFT

Rutinen beräknar Fouriertransformen av en komplex vektor (eller den inversa Fouriertransformen).[8]

RANF

Rutinen ger ett rektangelfördelat pseudoslumptal i intervallet 0..1. [11]

NORMAL

Rutinen ger ett approximativt normalfördelat slumptal med medelvärde M och standardavvikelse SIGMA. [10, sid. 262].

IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: PVEC.FOR Options: /LBZ 08/02/88 14:49:25

```
1 SUBROUTINE PVEC(V,N,PAR,RP)
2 C ANDERS PERSSON, 1987
3
4 IMPLICIT LOGICAL (A-Z)
5 INTEGER N,I
6 INTEGER*2 PAR(1:9),X,Y
7 REAL V(0:N-1),RP(1:2),DX
8
9 CALL GPAGE(PAR(6))
10 IF(PAR(8).EQ.0) CALL CLRSCR
11
12 X=1-PAR(5)
13 CALL LEVEL(X)
14 IF(PAR(9).EQ.0) CALL BLKFIL(PAR(1),PAR(2),PAR(3),PAR(4))
15 CALL LEVEL(PAR(5))
16
17 DX=FLOAT(PAR(3))/FLOAT(N)
18 X=PAR(1)
19 Y=((V(0)-RP(2))/(RP(1)-RP(2)))*PAR(4)+PAR(2)
20 CALL MOVE(X,Y)
21
22 DO 10 I=0,N-1
23 X=PAR(1)+I*DX
24 Y=((V(I)-RP(2))/(RP(1)-RP(2)))*PAR(4)+PAR(2)
25 IF(PAR(7).EQ.0) THEN
26 CALL PLOT(X,Y)
27 ELSE
28 CALL DLINE(X,Y)
29 ENDIF
30 10 CONTINUE
31
32 X=PAR(1)
33 Y=PAR(2)
34 CALL MOVE(X,Y)
35 X=X+PAR(3)
36 CALL DLINE(X,Y)
37 Y=Y+PAR(4)
38 CALL DLINE(X,Y)
39 X=X-PAR(3)
40 CALL DLINE(X,Y)
41 Y=Y-PAR(4)
42 CALL DLINE(X,Y)
43
44 RETURN
45 END
```

IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: CALCW.FOR Options: /LBZ 08/02/88 14:16:28

```
1 SUBROUTINE CALCW(W,N)
2 C ANDERS PERSSON, 1985
3
4 IMPLICIT LOGICAL (A-Z)
5 COMPLEX W(0:*)
6 REAL FI,TWOPI
7 PARAMETER ( TWOPI=6.283185308 )
8 INTEGER N,I
```

```
9
10 FI=-TWOPI/FLOAT(N)
11 DO 10 I=0,N/2-1
12 W(I)=CEXP(CMPLX(0.0,FI*I))
13 10 CONTINUE
14 RETURN
15 END
```

IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: FFT.FOR Options: /LBZ 08/02/88 14:24:21

```
1 SUBROUTINE FFT(G,W,N,INV)
2 C ANDERS PERSSON, 1985
3
4 IMPLICIT LOGICAL (A-Z)
5 COMPLEX G(0:*),W(0:*),SLASK
6 REAL C
7 INTEGER*2 I,J,K,L,M,P,R,S
8 INTEGER N
9 LOGICAL INV
10
11 C *** Butterfly ***
12 M=LOG(FLOAT(N))/LOG(2.0)+0.5
13 DO 10 K=0,M-1
14 DO 10 L=0,N-N/2**K,N/2**K
15 DO 10 I=0,N/2**(K+1)-1
16 R=L+1
17 S=R+N/2**(K+1)
18 P=I*2**K
19 SLASK=G(R)+G(S)
20 IF(INV) THEN
21 G(S)=(G(R)-G(S))*CONJG(W(P))
22 ELSE
23 G(S)=(G(R)-G(S))*W(P)
24 ENDIF
25 G(R)=SLASK
26 10 CONTINUE
27
28 C *** Bit reversal ***
29 DO 20 I=0,N-1
30 K=I
31 L=0
32 DO 30 J=1,M
33 L=L+MOD(K,2)*2**(M-J)
34 K=K/2
35 30 CONTINUE
36 IF(L.GT.I) THEN
37 SLASK=G(I)
38 G(I)=G(L)
39 G(L)=SLASK
40 ENDIF
41 20 CONTINUE
42
43 C *** Norm. ***
44 C=1.0/FLOAT(N)
45 IF(INV) THEN
46 DO 40 I=0,N-1
47 G(I)=G(I)*C
48 40 CONTINUE
49 ENDIF
```

```
50 RETURN
51 END
```

IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: RANF.FOR Options: /LBZ 08/02/88 14:49:38

```
1 REAL FUNCTION RANF()
2 C ANDERS PERSSON, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5 INTEGER*2 HO,MI,SE,HN
6 INTEGER X,Y,Z,SLASK
7 REAL DX,DY,DZ
8 LOGICAL FIRST
9 SAVE FIRST,X,Y,Z,DX,DY,DZ
10 DATA FIRST /.TRUE./
11
12 IF (FIRST) THEN
13 CALL GETTIM(HO,MI,SE,HN)
14 X=ABS((SE*99+HN)*5.46)
15 Y=X/3
16 Z=32767-X
17 DX=1.0/30269.0
18 DY=1.0/30307.0
19 DZ=1.0/30323.0
20 FIRST=.FALSE.
21 ENDIF
22
23 X=MOD(171*X,30269)
24 Y=MOD(172*Y,30307)
25 Z=MOD(170*Z,30323)
26 RANF=X*DX+Y*DY+Z*DZ
27 SLASK=RANF
28 RANF=RANF-SLASK
29
30 RETURN
31 END
```

IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: NRMAPROX.FOR Options: /LBZ 08/02/88 14:33:42

```
1 REAL FUNCTION NORMAL(M,SIGMA)
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5 REAL M,SIGMA,T,U,RANF
6
7 T=SIGMA*SQRT(-2.0*LOG(RANF()))
8 U=6.283185*RANF()
9 NORMAL=T*SIN(U)+M
10 RETURN
11 END
```

MATLIB, DMATLIB

Biblioteket innehåller diverse rutiner för matrisräkning. Det finns två varianter, MATLIB i enkel och DMATLIB för parametrar i dubbel precision.

Beskrivning av rutinerna.

INVMAT Inverterar matrisen A och placerar resultatet i A. Rutinen inverterar delmatrisen A(1:N,1:N). A:s deklarerade storlek ges av RA och KA. (Alla rutiner i biblioteket kan arbeta med delmatriser, men i samtliga fall måste även matrisernas deklarerade indexgränser skickas med till subrutinen.) Rutinen ger även värdet på determinanten. Om matrisen är singular har parametern SING värdet .TRUE. vid återhoppet. Maximal matrisstorlek är 100x100 men det kan lätt ändras genom att ändra i deklARATIONerna av ROW och COL vektorerna. De håller reda på hur rader och kolumner pivoteras av inverteringsalgoritmen så att matrisen kan återställas. Exekveringstiden kan beräknas med: $t(\text{sekunder}) \approx 6.635 \cdot 10^{-4} \cdot N^{2.894}$ (gäller för den AT-kopia testprogrammet kördes på) d.v.s ≈ 0.5 s för en 10x10 matris, ≈ 12 s för en 30x30 matris.

GAUSS Rutinen löser ekvationssystemet $A \cdot X = B$ med Gausselimination. A och B förstörs.

ADDMAT Rutinen adderar matriserna A och B, resultatet placeras i C.

SUBMAT $A - B \rightarrow C$

MULTMAT $A * B \rightarrow C$

TRANSMAT $A^T \rightarrow B$

ADDCMAT Addera skalären C till varje element i A.

MULTCMAT Multiplicera varje element i A med skalären C.

COPYMAT $A \rightarrow B$

UNITMAT Placera en enhetsmatris i A.

ZEROMAT Nollställ matrisen A.


```

1 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
2 C                               C
3 C   Library:  MATLIB.LIB       C
4 C                               C
5 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
6
7   SUBROUTINE INVMAT(A,N,DET,SING,RA,KA)
8 C   Anders Persson, 1988
9
10  IMPLICIT LOGICAL (A-Z)
11  INTEGER  N,R,K,I,L,ROW(1:100),COL(1:100),RA,KA
12  REAL     A(1:RA,1:KA),DET,PIVOT,INVPIV,SLASK
13  LOGICAL  SING
14
15  DET = 1.0
16  SING = .FALSE.
17
18  DO 70 I=1,N
19    PIVOT=0.0
20    DO 10 R=I,N
21      DO 10 K=1,N
22        IF (ABS(A(R,K)).GT.ABS(PIVOT)) THEN
23          PIVOT=A(R,K)
24          ROW(I)=R
25          COL(I)=K
26        ENDIF
27      10 CONTINUE
28
29    IF (PIVOT.EQ.0.0) THEN
30      SING=.TRUE.
31      DET=0.0
32      RETURN
33    ENDIF
34
35    IF (ROW(I)≠GT.I) THEN
36      DO 20 K=1,N
37        SLASK=A(I,K)
38        A(I,K)=A(ROW(I),K)
39        A(ROW(I),K)=-SLASK
40      20 CONTINUE
41    ENDIF
42
43    IF (COL(I).GT.I) THEN
44      DO 30 R=1,N
45        SLASK=A(R,I)
46        A(R,I)=A(R,COL(I))
47        A(R,COL(I))=-SLASK
48      30 CONTINUE
49    ENDIF
50
51    INVPIV=1.0/PIVOT
52    DO 40 R=1,N
53      IF(R.NE.I) A(R,I)=-A(R,I)*INVPIV
54    40 CONTINUE
55
56    DO 50 R=1,N
57      DO 50 K=1,N

```

-- 134 --

```

58      IF(R.NE.I.AND.K.NE.I) A(R,K)=A(R,K)+A(R,I)*A(I,K)
59    50 CONTINUE
60
61    DO 60 K=1,N
62      IF(K.NE.I) A(I,K)=A(I,K)*INVPIV
63    60 CONTINUE
64
65    A(I,I)=INVPIV
66    DET=DET*PIVOT
67
68  70 CONTINUE
69
70  DO 100 L=N,1,-1
71    IF (ROW(L).GT.L) THEN
72      DO 80 R=1,N
73        SLASK=A(R,L)
74        A(R,L)=-A(R,ROW(L))
75        A(R,ROW(L))=SLASK
76      80 CONTINUE
77    ENDIF
78
79    IF (COL(L).GT.L) THEN
80      DO 90 K=1,N
81        SLASK=A(L,K)
82        A(L,K)=-A(COL(L),K)
83        A(COL(L),K)=SLASK
84      90 CONTINUE
85    ENDIF
86
87  100 CONTINUE
88  RETURN
89
90  SUBROUTINE GAUSS(A,X,B,N,SING,RA,KA)
91  C   ANDERS PERSSON, 1986
92
93  IMPLICIT LOGICAL (A-Z)
94  INTEGER  RA,KA,N
95  REAL     A(1:RA,1:KA),X(1:N),B(1:N),BIG,TEMP
96  INTEGER  R,K,MAXROW,I,J
97  LOGICAL  SING
98
99
100 C *****
101 C *** Skalning ***
102 C *****
103 DO 10 R=1,N
104   BIG=ABS(B(R))
105   DO 20 K=1,N
106     BIG=MAX(BIG,ABS(A(R,K)))
107  20 CONTINUE
108   BIG=1.0/BIG
109   DO 30 K=1,N
110     A(R,K)=A(R,K)*BIG
111  30 CONTINUE
112   B(R)=B(R)*BIG
113  10 CONTINUE
114
115 DO 40 R=1,N
116 C *****
117 C *** Radpivoting ***

```

```

118 C *****
119 BIG=ABS(A(R,R))
120 MAXROW=R
121 DO 50 I=R+1,N
122 IF (ABS(A(I,R)).GT.BIG) THEN
123 MAXROW=I
124 BIG=ABS(A(I,R))
125 ENDIF
126 50 CONTINUE
127 IF (BIG.EQ.0.0) THEN
128 SING=.TRUE.
129 RETURN
130 ENDIF
131 DO 60 K=R,N
132 TEMP=A(R,K)
133 A(R,K)=A(MAXROW,K)
134 A(MAXROW,K)=TEMP
135 60 CONTINUE
136 TEMP=B(R)
137 B(R)=B(MAXROW)
138 B(MAXROW)=TEMP
139
140 C *****
141 C *** Elimination ***
142 C *****
143 TEMP=1.0/A(R,R)
144 DO 70 K=N,R,-1
145 A(R,K)=A(R,K)*TEMP
146 70 CONTINUE
147 B(R)=B(R)*TEMP
148 DO 80 I=R+1,N
149 DO 90 J=R+1,N
150 A(I,J)=A(I,J)-A(I,R)*A(R,J)
151 90 CONTINUE
152 B(I)=B(I)-B(R)*A(I,R)
153 80 CONTINUE
154 40 CONTINUE
155
156 C *****
157 C *** Substitution ***
158 C *****
159 DO 100 R=N,1,-1
160 X(R)=B(R)
161 DO 110 K=R+1,N
162 X(R)=X(R)-A(R,K)*X(K)
163 110 CONTINUE
164 100 CONTINUE
165
166 RETURN
167 END
168
169 SUBROUTINE ADDMAT(A,B,C,R,K,RA,KA,KB,RC,KC)
170 C Anders Persson, 1988
171
172 IMPLICIT LOGICAL (A-Z)
173 INTEGER R,K,I,J,RA,KA,KB,RC,KC
174 REAL A(1:RA,1:KA),B(1:RB,1:KB),C(1:RC,1:KC)
175
176 DO 10 I=1,R
177 DO 10 J=1,R

```

```

178 C(I,J)=A(I,J)+B(I,J)
179 10 CONTINUE
180
181 RETURN
182 END
183
184 SUBROUTINE SUBMAT(A,B,C,R,K,RA,KA,KB,RC,KC)
185 C Anders Persson, 1988
186
187 IMPLICIT LOGICAL (A-Z)
188 INTEGER R,K,I,J,RA,KA,KB,RC,KC
189 REAL A(1:RA,1:KA),B(1:RB,1:KB),C(1:RC,1:KC)
190
191 DO 10 I=1,R
192 DO 10 J=1,R
193 C(I,J)=A(I,J)-B(I,J)
194 10 CONTINUE
195
196 RETURN
197 END
198
199 SUBROUTINE MULTMAT(A,B,C,AR,AK,BK,RA,KA,KB,RC,KC)
200 C Anders Persson, 1988
201
202 IMPLICIT LOGICAL (A-Z)
203 INTEGER I,J,K,AR,AK,BK,RA,KA,KB,RC,KC
204 REAL A(1:RA,1:KA),B(1:RB,1:KB),C(1:RC,1:KC)
205
206 CALL ZEROMAT(C,AR,BK,RC,KC)
207 DO 10 I=1,AR
208 DO 10 J=1,BK
209 DO 10 K=1,AK
210 C(I,J)=C(I,J)+A(I,K)*B(K,J)
211 10 CONTINUE
212
213 RETURN
214 END
215
216 SUBROUTINE TRANSMAT(A,B,R,K,RA,KA,KB,KB)
217 C Anders Persson, 1988
218
219 IMPLICIT LOGICAL (A-Z)
220 INTEGER R,K,I,J,RA,KA,KB,KB
221 REAL A(1:RA,1:KA),B(1:RB,1:KB)
222
223 DO 10 I=1,R
224 DO 10 J=1,K
225 B(J,I)=A(I,J)
226 10 CONTINUE
227
228 RETURN
229 END
230
231 SUBROUTINE ADDCMAT(A,C,R,K,RA,KA)
232 C Anders Persson, 1988
233
234 IMPLICIT LOGICAL (A-Z)
235 INTEGER R,K,I,J,RA,KA
236 REAL A(1:RA,1:KA),C
237

```

```

238 DO 10 I=1,R
239 DO 10 J=1,K
240 A(I,J)=A(I,J)+C
241 10 CONTINUE
242
243 RETURN
244 END
245
246 SUBROUTINE MULTCMAT(A,C,R,K,RA,KA)
247 C Anders Persson, 1988
248
249 IMPLICIT LOGICAL (A-Z)
250 INTEGER R,K,I,J,RA,KA
251 REAL A(1:RA,1:KA),C
252
253 DO 10 I=1,R
254 DO 10 J=1,K
255 A(I,J)=A(I,J)*C
256 10 CONTINUE
257
258 RETURN
259 END
260
261 SUBROUTINE COPYMAT(A,B,R,K,RA,KA,KB,KB)
262 C Anders Persson, 1988
263
264 IMPLICIT LOGICAL (A-Z)
265 INTEGER R,K,I,J,RA,KA,KB,KB
266 REAL A(1:RA,1:KA),B(1:RB,1:KB)
267
268 DO 10 I=1,R
269 DO 10 J=1,K
270 B(I,J)=A(I,J)
271 10 CONTINUE
272
273 RETURN
274 END
275
276 SUBROUTINE UNITMAT(A,N,RA,KA)
277 C Anders Persson, 1988
278
279 IMPLICIT LOGICAL (A-Z)
280 INTEGER N,R,K,RA,KA
281 REAL A(1:RA,1:KA)
282
283 CALL ZEROMAT(A,N,N,RA,KA)
284 DO 10 R=1,N
285 A(R,R)=1.0
286 10 CONTINUE
287
288 RETURN
289 END
290
291 SUBROUTINE ZEROMAT(A,R,K,RA,KA)
292 C Anders Persson, 1988
293
294 IMPLICIT LOGICAL (A-Z)
295 INTEGER R,K,I,J,RA,KA
296 REAL A(1:RA,1:KA)
297

```

```

298 DO 10 I=1,R
299 DO 10 J=1,K
300 A(I,J)=0.0
301 10 CONTINUE
302
303 RETURN
304 END

```

```

1 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
2 C                               C
3 C   Library:  DMATLIB.LIB      C
4 C                               C
5 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
6
7   SUBROUTINE INVMAT(A,N,DET,SING,RA,KA)
8 C   Anders Persson, 1988
9
10  IMPLICIT   LOGICAL (A-Z)
11  INTEGER    N,R,K,I,L,ROW(1:100),COL(1:100),RA,KA
12  REAL*8     A(1:RA,1:KA),DET,PIVOT,INVPIV,SLASK
13  LOGICAL    SING
14
15  DET = 1.000
16  SING = .FALSE.
17
18  DO 70 I=1,N
19    PIVOT=0.000
20    DO 10 R=I,N
21      DO 10 K=I,N
22        IF (ABS(A(R,K)).GT.ABS(PIVOT)) THEN
23          PIVOT=A(R,K)
24          ROW(I)=R
25          COL(I)=K
26        ENDIF
27      10 CONTINUE
28
29    IF (PIVOT.EQ.0.000) THEN
30      SING=.TRUE.
31      DET=0.000
32      RETURN
33    ENDIF
34
35    IF (ROW(I).GT.I) THEN
36      DO 20 K=1,N
37        SLASK=A(I,K)
38        A(I,K)=A(ROW(I),K)
39        A(ROW(I),K)=-SLASK
40      20 CONTINUE
41    ENDIF
42
43    IF (COL(I).GT.I) THEN
44      DO 30 R=1,N
45        SLASK=A(R,I)
46        A(R,I)=A(R,COL(I))
47        A(R,COL(I))=-SLASK
48      30 CONTINUE
49    ENDIF
50
51    INVPIV=1.000/PIVOT
52    DO 40 R=1,N
53      IF(R.NE.I) A(R,I)=-A(R,I)*INVPIV
54    40 CONTINUE
55
56    DO 50 R=1,N
57      DO 50 K=1,N

```

```

58      IF(R.NE.I.AND.K.NE.I) A(R,K)=A(R,K)+A(R,I)*A(I,K)
59    50 CONTINUE
60
61    DO 60 K=1,N
62      IF(K.NE.I) A(I,K)=A(I,K)*INVPIV
63    60 CONTINUE
64
65    A(I,I)=INVPIV
66    DET=DET*PIVOT
67
68  70 CONTINUE
69
70  DO 100 L=N,1,-1
71    IF (ROW(L).GT.L) THEN
72      DO 80 R=1,N
73        SLASK=A(R,L)
74        A(R,L)=-A(R,ROW(L))
75        A(R,ROW(L))=SLASK
76      80 CONTINUE
77    ENDIF
78
79    IF (COL(L).GT.L) THEN
80      DO 90 K=1,N
81        SLASK=A(L,K)
82        A(L,K)=-A(COL(L),K)
83        A(COL(L),K)=SLASK
84      90 CONTINUE
85    ENDIF
86
87  100 CONTINUE
88
89  RETURN
90  END
91
92  SUBROUTINE GAUSS(A,X,B,N,SING,RA,KA)
93 C  ANDERS PERSSON, 1986
94
95  IMPLICIT   LOGICAL (A-Z)
96  INTEGER    RA,KA,N
97  REAL*8     A(1:RA,1:KA),X(1:N),B(1:N),BIG,TEMP
98  INTEGER    R,K,MAXROW,I,J
99  LOGICAL    SING
100
101 C *****
102 C *** Skalning ***
103 C *****
104  DO 10 R=1,N
105    BIG=ABS(B(R))
106    DO 20 K=1,N
107      BIG=MAX(BIG,ABS(A(R,K)))
108    20 CONTINUE
109    BIG=1.000/BIG
110    DO 30 K=1,N
111      A(R,K)=A(R,K)*BIG
112    30 CONTINUE
113    B(R)=B(R)*BIG
114  10 CONTINUE
115
116  DO 40 R=1,N
117 C *****

```

```

118 C   *** Radpivoting ***
119 C   *****
120     BIG=ABS(A(R,R))
121     MAXROW=R
122     DO 50 I=R+1,N
123       IF (ABS(A(I,R)).GT.BIG) THEN
124         MAXROW=I
125         BIG=ABS(A(I,R))
126       ENDIF
127 50   CONTINUE
128
129     IF (BIG.EQ.0.0D0) THEN
130       SING=.TRUE.
131       RETURN
132     ENDIF
133
134     DO 60 K=R,N
135       TEMP=A(R,K)
136       A(R,K)=A(MAXROW,K)
137       A(MAXROW,K)=TEMP
138 60   CONTINUE
139
140     TEMP=B(R)
141     B(R)=B(MAXROW)
142     B(MAXROW)=TEMP
143 C   *****
144 C   *** Elimination ***
145 C   *****
146
147     TEMP=1.0D0/A(R,R)
148     DO 70 K=N,R,-1
149       A(R,K)=A(R,K)*TEMP
150 70   CONTINUE
151
152     B(R)=B(R)*TEMP
153     DO 80 I=R+1,N
154       DO 90 J=R+1,N
155         A(I,J)=A(I,J)-A(I,R)*A(R,J)
156 90   CONTINUE
157     B(I)=B(I)-B(R)*A(I,R)
158 80   CONTINUE
159
160 40   CONTINUE
161
162 C   *****
163 C   *** Substitution ***
164 C   *****
165     DO 100 R=N,1,-1
166       X(R)=B(R)
167       DO 110 K=R+1,N
168         X(R)=X(R)-A(R,K)*X(K)
169 110   CONTINUE
170 100   CONTINUE
171
172     RETURN
173     END
174
175     SUBROUTINE ADDMAT(A,B,C,R,K,RA,KA,KB,RC,KC)
176 C   Anders Persson, 1988
177

```

```

178     IMPLICIT LOGICAL (A-Z)
179     INTEGER R,K,I,J,RA,KA,KB,RC,KC
180     REAL*8 A(1:RA,1:KA),B(1:RB,1:KB),C(1:RC,1:KC)
181
182     DO 10 I=1,R
183     DO 10 J=1,R
184       C(I,J)=A(I,J)+B(I,J)
185 10   CONTINUE
186
187     RETURN
188     END
189
190     SUBROUTINE SUBMAT(A,B,C,R,K,RA,KA,KB,RC,KC)
191 C   Anders Persson, 1988
192
193     IMPLICIT LOGICAL (A-Z)
194     INTEGER R,K,I,J,RA,KA,KB,RC,KC
195     REAL*8 A(1:RA,1:KA),B(1:RB,1:KB),C(1:RC,1:KC)
196
197     DO 10 I=1,R
198     DO 10 J=1,R
199       C(I,J)=A(I,J)-B(I,J)
200 10   CONTINUE
201
202     RETURN
203     END
204
205     SUBROUTINE MULTMAT(A,B,C,AR,AK,BK,RA,KA,KB,RC,KC)
206 C   Anders Persson, 1988
207
208     IMPLICIT LOGICAL (A-Z)
209     INTEGER I,J,K,AR,AK,BK,RA,KA,KB,RC,KC
210     REAL*8 A(1:RA,1:KA),B(1:RB,1:KB),C(1:RC,1:KC)
211
212     CALL ZEROMAT(C,AR,BK,RC,KC)
213     DO 10 I=1,AR
214     DO 10 J=1,BK
215     DO 10 K=1,AK
216       C(I,J)=C(I,J)+A(I,K)*B(K,J)
217 10   CONTINUE
218
219     RETURN
220     END
221
222     SUBROUTINE TRANSMAT(A,B,R,K,RA,KA,KB,KB)
223 C   Anders Persson, 1988
224
225     IMPLICIT LOGICAL (A-Z)
226     INTEGER R,K,I,J,RA,KA,KB,KB
227     REAL*8 A(1:RA,1:KA),B(1:RB,1:KB)
228
229     DO 10 I=1,R
230     DO 10 J=1,K
231       B(J,I)=A(I,J)
232 10   CONTINUE
233
234     RETURN
235     END
236
237     SUBROUTINE ADDCMAT(A,C,R,K,RA,KA)

```

```

238 C   Anders Persson, 1988
239
240     IMPLICIT   LOGICAL (A-Z)
241     INTEGER    R,K,I,J,RA,KA
242     REAL*8     A(1:RA,1:KA),C
243
244     DO 10 I=1,R
245     DO 10 J=1,K
246         A(I,J)=A(I,J)+C
247 10   CONTINUE
248
249     RETURN
250     END
251
252     SUBROUTINE MULTCMAT(A,C,R,K,RA,KA)
253 C   Anders Persson, 1988
254
255     IMPLICIT   LOGICAL (A-Z)
256     INTEGER    R,K,I,J,RA,KA
257     REAL*8     A(1:RA,1:KA),C
258
259     DO 10 I=1,R
260     DO 10 J=1,K
261         A(I,J)=A(I,J)*C
262 10   CONTINUE
263
264     RETURN
265     END
266
267     SUBROUTINE COPYMAT(A,B,R,K,RA,KA,RB,KB)
268 C   Anders Persson, 1988
269
270     IMPLICIT   LOGICAL (A-Z)
271     INTEGER    R,K,I,J,RA,KA,RB,KB
272     REAL*8     A(1:RA,1:KA),B(1:RB,1:KB)
273
274     DO 10 I=1,R
275     DO 10 J=1,K
276         B(I,J)=A(I,J)
277 10   CONTINUE
278
279     RETURN
280     END
281
282     SUBROUTINE UNITMAT(A,N,RA,KA)
283 C   Anders Persson, 1988
284
285     IMPLICIT   LOGICAL (A-Z)
286     INTEGER    N,R,K,RA,KA
287     REAL*8     A(1:RA,1:KA)
288
289     CALL ZEROMAT(A,N,N,RA,KA)
290     DO 10 R=1,N
291         A(R,R)=1.0D0
292 10   CONTINUE
293
294     RETURN
295     END
296
297     SUBROUTINE ZEROMAT(A,R,K,RA,KA)

```

```

298 C   Anders Persson, 1988
299
300     IMPLICIT   LOGICAL (A-Z)
301     INTEGER    R,K,I,J,RA,KA
302     REAL*8     A(1:RA,1:KA)
303
304     DO 10 I=1,R
305     DO 10 J=1,K
306         A(I,J)=0.0D0
307 10   CONTINUE
308
309     RETURN
310     END

```

Ritrutiner:

SCRLOT, (D)SCRLOT, ROLPLOT, (D)ROLPLOT, JETPLOT, (D)JETPLOT

Dessa rutiner gör i stort sett samma sak, de ritar punktföljden (x_i, y_i) (eventuellt med felgränser om variabeln ERRORBAR är sann) samt ritar den anpassade funktionen $f(\theta; x)$ på samma bild. Därefter beräknas och ritas residualvektorn (48). Rutinerna anropas på samma sätt med samma parametrar, SCRLOT ritar på datorns skärm, ROLPLOT ritar på en Roland pen-plotter och JETPLOT ritar på en HP laserskrivare. Alla rutiner finns i två versioner, en för vektorer i enkel och en för vektorer i dubbel precision (D). Filnamnen med rutiner i dubbel precision har ett D adderat till namnet. Med ROLPLOT-rutinen kan man rita på standardformaten A3,A4 eller i valfri storlek. JETPLOT ritar bara på A4.

```
1 SUBROUTINE SCRPL0T(Y,SIGMA,X,R,N,F,THETA,L,ERRORBAR)
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5 INTEGER PLXMIN,PLXMAX,PLYMIN,PLYMAX,FVECRANGE
6 PARAMETER ( FVECRANGE=500 )
7 INTEGER I,N,L,X0,Y0,XL,YL
8 REAL Y(1:N),SIGMA(1:N),X(1:N),R(1:N),F,THETA(1:L),XMIN,XMAX
9 REAL FVEC(1:FVECRANGE),SLASK,DX,XI,XP,YP,YMIN,YMAX,A
10 INTEGER*2 PX,PY,ANS,KBDCHK,KBDINC
11 LOGICAL ERRORBAR
12
13 XP(A)=NINT(((A-XMIN)/(XMAX-XMIN))*XL+X0)
14 YP(A)=NINT(((A-YMIN)/(YMAX-YMIN))*YL+Y0)
15 C
16 PLXMIN=0
17 PLXMAX=719
18 PLYMIN=0
19 PLYMAX=347
20 C
21 X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
22 Y0=NINT(PLYMIN+0.26*(PLYMAX-PLYMIN))
23 XL=NINT(0.99*(PLXMAX-PLXMIN))
24 YL=NINT(0.73*(PLYMAX-PLYMIN))
25
26 XMIN=X(1)
27 XMAX=X(N)
28 YMIN=Y(1)
29 YMAX=Y(1)
30
31 DX=(XMAX-XMIN)/(FVECRANGE-1)
32 DO 10 I=1,FVECRANGE
33 XI=(I-1)*DX+XMIN
34 FVEC(I)=F(THETA,L,XI)
35 YMIN=MIN(YMIN,FVEC(I))
36 YMAX=MAX(YMAX,FVEC(I))
37 10 CONTINUE
38
39 DO 20 I=1,N
40 IF (ERRORBAR) THEN
41 YMIN=MIN(YMIN,Y(I)-SIGMA(I))
42 YMAX=MAX(YMAX,Y(I)+SIGMA(I))
43 ELSE
44 YMIN=MIN(YMIN,Y(I))
45 YMAX=MAX(YMAX,Y(I))
46 ENDIF
47 20 CONTINUE
48
49 IF (YMAX-YMIN.LT.1.0E-20) THEN
50 YMAX=YMAX+0.5
51 YMIN=YMIN-0.5
52 ENDIF
53
54 CALL GMODE
55 ANS=1
56 CALL GPAGE(ANS)
57 CALL CLRSCR
```

```
58 CALL DISP(ANS)
59
60 DO 30 I=1,N
61 PX=XP(X(I))
62 IF (ERRORBAR) THEN
63 PY=YP(Y(I)+SIGMA(I))
64 ELSE
65 PY=YP(Y(I))
66 ENDIF
67 CALL MOVE(PX,PY)
68 IF (ERRORBAR) THEN
69 PY=YP(Y(I)-SIGMA(I))
70 CALL DLINE(PX,PY)
71 ELSE
72 CALL PLOT(PX,PY)
73 ENDIF
74 30 CONTINUE
75
76 PY=YP(FVEC(1))
77 PX=XP(XMIN)
78 CALL MOVE(PX,PY)
79 DO 40 I=1,FVECRANGE
80 XI=(I-1)*DX+XMIN
81 PX=XP(XI)
82 PY=YP(FVEC(I))
83 CALL DLINE(PX,PY)
84 40 CONTINUE
85
86 X0=PLXMIN
87 Y0=NINT(PLYMIN+0.25*(PLYMAX-PLYMIN))
88 XL=PLXMAX-PLXMIN
89 YL=NINT(0.75*(PLYMAX-PLYMIN))
90
91 PX=X0
92 PY=Y0
93 CALL MOVE(PX,PY)
94 PX=X0+XL
95 CALL DLINE(PX,PY)
96 PY=Y0+YL
97 CALL DLINE(PX,PY)
98 PX=X0
99 CALL DLINE(PX,PY)
100 PY=Y0
101 CALL DLINE(PX,PY)
102
103 DO 50 I=1,N
104 R(I)=0.0
105 IF (SIGMA(I).NE.0.0) THEN
106 R(I)=(Y(I)-F(THETA,L,X(I)))/SIGMA(I)
107 ENDIF
108 50 CONTINUE
109
110 YMAX=R(1)
111 YMIN=R(1)
112 DO 60 I=1,N
113 YMAX=MAX(YMAX,R(I))
114 YMIN=MIN(YMIN,R(I))
115 60 CONTINUE
116
117 YMAX=ABS(YMAX)
```



```

118 YMAX=MAX(YMAX,ABS(YMIN))
119 YMIN=-YMAX
120 IF (YMAX-YMIN.LT.1.0E-20) THEN
121   YMAX=YMAX+0.5
122   YMIN=YMIN-0.5
123 ENDIF
124
125 X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
126 Y0=NINT(PLYMIN+0.01*(PLYMAX-PLYMIN))
127 XL=NINT(0.99*(PLXMAX-PLXMIN))
128 YL=NINT(0.22*(PLYMAX-PLYMIN))
129
130 DO 70 I=1,N
131   IF (R(I).NE.0.0) THEN
132     PX=XP(X(I))
133     PY=YP(R(I))
134     CALL PLOT(PX,PY)
135   ENDIF
136 70 CONTINUE
137
138 PX=XP(X(1))
139 PY=YP(0.0)
140 CALL MOVE(PX,PY)
141 PX=XP(X(N))
142 CALL DLINE(PX,PY)
143
144 X0=PLXMIN
145 Y0=PLYMIN
146 XL=PLXMAX-PLXMIN
147 YL=NINT(0.24*(PLYMAX-PLYMIN))
148
149 PX=X0
150 PY=Y0
151 CALL MOVE(PX,PY)
152 PX=X0+XL
153 CALL DLINE(PX,PY)
154 PY=Y0+YL
155 CALL DLINE(PX,PY)
156 PX=X0
157 CALL DLINE(PX,PY)
158 PY=Y0
159 CALL DLINE(PX,PY)
160
161 80 ANS=KBDCHK()
162 IF (ANS.EQ.0) GOTO 80
163 ANS=KBDINC()
164 CALL TMODE
165
166 RETURN
167 END

```

```

7 INTEGER I,N,L,X0,Y0,XL,YL
8 REAL*8 Y(1:N),SIGMA(1:N),X(1:N),R(1:N),F,THETA(1:L),XMIN,XMAX
9 REAL*8 FVEC(1:FVECRANGE),SLASK,DX,XI,XP,YP,YMIN,YMAX,A
10 INTEGER*2 PX,PY,ANS,KBDCHK,KBDINC
11 LOGICAL ERRORBAR
12
13 XP(A)=NINT(((A-XMIN)/(XMAX-XMIN))*XL+X0)
14 YP(A)=NINT(((A-YMIN)/(YMAX-YMIN))*YL+Y0)
15
16 PLXMIN=0
17 PLXMAX=719
18 PLYMIN=0
19 PLYMAX=347
20
21 X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
22 Y0=NINT(PLYMIN+0.26*(PLYMAX-PLYMIN))
23 XL=NINT(0.99*(PLXMAX-PLXMIN))
24 YL=NINT(0.73*(PLYMAX-PLYMIN))
25
26 XMIN=X(1)
27 XMAX=X(N)
28 YMIN=Y(1)
29 YMAX=Y(1)
30
31 DX=(XMAX-XMIN)/(FVECRANGE-1)
32 DO 10 I=1,FVECRANGE
33   XI=(I-1)*DX+XMIN
34   FVEC(I)=F(THETA,L,XI)
35   YMIN=MIN(YMIN,FVEC(I))
36   YMAX=MAX(YMAX,FVEC(I))
37 10 CONTINUE
38
39 DO 20 I=1,N
40   IF (ERRORBAR) THEN
41     YMIN=MIN(YMIN,Y(I)-SIGMA(I))
42     YMAX=MAX(YMAX,Y(I)+SIGMA(I))
43   ELSE
44     YMIN=MIN(YMIN,Y(I))
45     YMAX=MAX(YMAX,Y(I))
46   ENDIF
47 20 CONTINUE
48
49 IF (YMAX-YMIN.LT.1.0D-20) THEN
50   YMAX=YMAX+0.5D0
51   YMIN=YMIN-0.5D0
52 ENDIF
53
54 CALL GMODE
55 ANS=1
56 CALL GPAGE(ANS)
57 CALL CLRSCR
58 CALL DISP(ANS)
59
60 DO 30 I=1,N
61   PX=XP(X(I))
62   IF (ERRORBAR) THEN
63     PY=YP(Y(I)+SIGMA(I))
64   ELSE
65     PY=YP(Y(I))
66   ENDIF

```

IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: DSCRLOT.FOR Options: /LBZ 08/02/88 14:19:55

```

1 SUBROUTINE SCRPLLOT(Y,SIGMA,X,R,N,F,THETA,L,ERRORBAR)
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5 INTEGER PLXMIN,PLXMAX,PLYMIN,PLYMAX,FVECRANGE
6 PARAMETER ( FVECRANGE=500 )

```

```

67     CALL MOVE(PX,PY)
68     IF (ERRORBAR) THEN
69         PY=YP(Y(I)-SIGMA(I))
70         CALL DLINE(PX,PY)
71     ELSE
72         CALL PLOT(PX,PY)
73     ENDIF
74 30  CONTINUE
75
76     PY=YP(FVEC(1))
77     PX=XP(XMIN)
78     CALL MOVE(PX,PY)
79     DO 40 I=1,FVECRANGE
80         XI=(I-1)*DX+XMIN
81         PX=XP(XI)
82         PY=YP(FVEC(I))
83         CALL DLINE(PX,PY)
84 40  CONTINUE
85
86     X0=PLXMIN
87     Y0=NINT(PLYMIN+0.25*(PLYMAX-PLYMIN))
88     XL=PLXMAX-PLXMIN
89     YL=NINT(0.75*(PLYMAX-PLYMIN))
90
91     PX=X0
92     PY=Y0
93     CALL MOVE(PX,PY)
94     PX=X0+XL
95     CALL DLINE(PX,PY)
96     PY=Y0+YL
97     CALL DLINE(PX,PY)
98     PX=X0
99     CALL DLINE(PX,PY)
100    PY=Y0
101    CALL DLINE(PX,PY)
102
103    DO 50 I=1,N
104        R(I)=0.000
105        IF (SIGMA(I).NE.0.000) THEN
106            R(I)=(Y(I)-F(THETA,L,X(I)))/SIGMA(I)
107        ENDIF
108 50  CONTINUE
109
110    YMAX=R(1)
111    YMIN=R(1)
112    DO 60 I=1,N
113        YMAX=MAX(YMAX,R(I))
114        YMIN=MIN(YMIN,R(I))
115 60  CONTINUE
116
117    YMAX=ABS(YMAX)
118    YMAX=MAX(YMAX,ABS(YMIN))
119    YMIN=-YMAX
120    IF (YMAX-YMIN.LT.1.0D-20) THEN
121        YMAX=YMAX+0.5D0
122        YMIN=YMIN-0.5D0
123    ENDIF
124
125    X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
126    Y0=NINT(PLYMIN+0.01*(PLYMAX-PLYMIN))

```

```

127    XL=NINT(0.99*(PLXMAX-PLXMIN))
128    YL=NINT(0.22*(PLYMAX-PLYMIN))
129
130    DO 70 I=1,N
131        IF (R(I).NE.0.000) THEN
132            PX=XP(X(I))
133            PY=YP(R(I))
134            CALL PLOT(PX,PY)
135        ENDIF
136 70  CONTINUE
137
138    PX=XP(X(1))
139    PY=YP(0.000)
140    CALL MOVE(PX,PY)
141    PX=XP(X(N))
142    CALL DLINE(PX,PY)
143
144    X0=PLXMIN
145    Y0=PLYMIN
146    XL=PLXMAX-PLXMIN
147    YL=NINT(0.24*(PLYMAX-PLYMIN))
148
149    PX=X0
150    PY=Y0
151    CALL MOVE(PX,PY)
152    PX=X0+XL
153    CALL DLINE(PX,PY)
154    PY=Y0+YL
155    CALL DLINE(PX,PY)
156    PX=X0
157    CALL DLINE(PX,PY)
158    PY=Y0
159    CALL DLINE(PX,PY)
160
161 80  ANS=KBDCHK()
162    IF (ANS.EQ.0) GOTO 80
163    ANS=KBDINC()
164    CALL TMODE
165
166    RETURN
167    END

```

IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: ROLPLOT.FOR Options: /LBZ 08/02/88 14:49:48

```

1  SUBROUTINE ROLPLOT(Y,SIGMA,X,R,N,F,THETA,L,ERRORBAR)
2 C  Anders Persson, 1988
3
4  IMPLICIT LOGICAL (A-Z)
5  INTEGER FVECRANGE
6  REAL PLXMIN,PLXMAX,PLYMIN,PLYMAX
7  PARAMETER ( FVECRANGE=500 )
8  INTEGER N,L,I,X0,Y0,XL,YL,P1,P2,P3
9  REAL Y(1:N),SIGMA(1:N),X(1:N),R(1:N),F,THETA(1:L),XMIN,XMAX
10 REAL FVEC(1:FVECRANGE),SLASK,DX,XI,XP,YP,YMIN,YMAX,A
11 REAL PX,PY,PS
12 INTEGER*2 ANS,KBDCHK,KBDINC
13 LOGICAL ERRORBAR,LINKPEN
14 CHARACTER SV*1,VS*6
15

```

```

16  XP(A)=NINT(((A-XMIN)/(XMAX-XMIN))*XL+X0)
17  YP(A)=NINT(((A-YMIN)/(YMAX-YMIN))*YL+Y0)
18
19  WRITE(*,*) 'Standard paper size? (Y/*):'
20  READ(*,100) SV
21 100  FORMAT(A)
22
23  IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
24    WRITE(*,*) 'Size A3 or A4 (3/4):'
25    READ(*,100) SV
26    IF (SV.EQ.'3') THEN
27      PLXMIN=0.0
28      PLXMAX=3600.0
29      PLYMIN=0.0
30      PLYMAX=2600.0
31    ELSE
32      PLXMIN=0.0
33      PLXMAX=2600.0
34      PLYMIN=0.0
35      PLYMAX=1800.0
36    ENDIF
37  ELSE
38 1  WRITE(*,*) 'Enter plot window (in cm.).'
39    WRITE(*,*) 'Xmin, Ymin, Xmax, Ymax:'
40    READ(*,*,ERR=1,END=1) PLXMIN,PLYMIN,PLXMAX,PLYMAX
41    IF (PLXMIN.LT.0.0.OR.PLXMIN.GE.38.0) GOTO 1
42    IF (PLYMIN.LT.0.0.OR.PLYMIN.GE.27.0) GOTO 1
43    IF (PLXMAX.LT.0.0.OR.PLXMAX.GT.38.0) GOTO 1
44    IF (PLYMAX.LT.0.0.OR.PLYMAX.GT.27.0) GOTO 1
45    IF (PLXMIN.GE.PLXMAX) GOTO 1
46    IF (PLYMIN.GE.PLYMAX) GOTO 1
47    PLXMIN=PLXMIN*100.0
48    PLXMAX=PLXMAX*100.0
49    PLYMIN=PLYMIN*100.0
50    PLYMAX=PLYMAX*100.0
51  ENDIF
52
53  P1=1
54  P2=2
55  P3=3
56
57  WRITE(*,*) 'Liquid ink pen? (Y/*):'
58  READ(*,100) SV
59  LINKPEN=SV.EQ.'Y'.OR.SV.EQ.'y'
60
61  IF (LINKPEN) THEN
62    P1=1
63    P2=1
64    P3=1
65    WRITE(*,*) 'Pen speed in cm/s (2..23):'
66    READ(*,*) PS
67    IF (PS.LT.2.0.OR.PS.GT.23.0) THEN
68      PS=8.0
69    ENDIF
70    WRITE(VS,200) 'VS ',NINT(PS),';'
71 200  FORMAT(A3,I2,A1)
72  ENDIF
73
74  IF (LINKPEN) THEN
75    WRITE(*,*) 'Put liquid ink pen in slot 1.'
76  ELSE
77    WRITE(*,*) 'Put pens in slot 1,2 and 3.'
78  ENDIF
79
80  WRITE(*,*) 'When plotter ready press any key!'
81 2  ANS=KBDCHK()
82  IF (ANS.EQ.0) THEN
83    GOTO 2
84  ENDIF
85  ANS=KBDINC()
86  C
87  IF (LINKPEN) THEN
88    CALL RRDGLC(VS,6)
89  ELSE
90    CALL RRDGLC('VS 23;',6)
91  ENDIF
92
93  X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
94  Y0=NINT(PLYMIN+0.26*(PLYMAX-PLYMIN))
95  XL=NINT(0.99*(PLXMAX-PLXMIN))
96  YL=NINT(0.73*(PLYMAX-PLYMIN))
97
98  XMIN=X(1)
99  XMAX=X(N)
100  YMIN=Y(1)
101  YMAX=Y(1)
102
103  DX=(XMAX-XMIN)/(FVECRANGE-1)
104  DO 10 I=1,FVECRANGE
105    XI=(I-1)*DX+XMIN
106    FVEC(I)=F(THETA,L,XI)
107    YMIN=MIN(YMIN,FVEC(I))
108    YMAX=MAX(YMAX,FVEC(I))
109 10  CONTINUE
110
111  DO 20 I=1,N
112    IF (ERRORBAR) THEN
113      YMIN=MIN(YMIN,Y(I)-SIGMA(I))
114      YMAX=MAX(YMAX,Y(I)+SIGMA(I))
115    ELSE
116      YMIN=MIN(YMIN,Y(I))
117      YMAX=MAX(YMAX,Y(I))
118    ENDIF
119 20  CONTINUE
120
121  IF (YMAX-YMIN.LT.1.0E-20) THEN
122    YMAX=YMAX+0.5
123    YMIN=YMIN-0.5
124  ENDIF
125
126  CALL RLNTYP(0)
127  CALL RASCAL(0)
128  CALL RMOVE(0.0,1500.0)
129  CALL RPENCH(P2)
130
131  DO 30 I=1,N
132    PX=XP(X(I))
133    IF (ERRORBAR) THEN
134      IF (MOD(I,2).EQ.0) THEN
135        PY=YP(Y(I)+SIGMA(I))

```

```

136     ELSE
137     PY=YP(Y(I)-SIGMA(I))
138     ENDF
139     ELSE
140     PY=YP(Y(I))
141     ENDF
142     CALL RMOVE(PX,PY)
143     IF (ERRORBAR) THEN
144     IF (MOD(I,2).EQ.0) THEN
145     PY=YP(Y(I)-SIGMA(I))
146     ELSE
147     PY=YP(Y(I)+SIGMA(I))
148     ENDF
149     CALL RDRAW(PX,PY)
150     ELSE
151     CALL RMARK(4)
152     ENDF
153 30 CONTINUE
154
155 PY=YP(FVEC(1))
156 PX=XP(XMIN)
157 IF (.NOT.LINKPEN) THEN
158 CALL RMOVE(0.0,1500.0)
159 CALL RPENCH(P3)
160 ENDF
161
162 CALL RMOVE(PX,PY)
163 DO 40 I=1,FVECRANGE
164 XI=(I-1)*DX+XMIN
165 PX=XP(XI)
166 PY=YP(FVEC(I))
167 CALL RDRAW(PX,PY)
168 40 CONTINUE
169
170 X0=PLXMIN
171 Y0=NINT(PLYMIN+0.25*(PLYMAX-PLYMIN))
172 XL=PLXMAX-PLXMIN
173 YL=NINT(0.75*(PLYMAX-PLYMIN))
174
175 PX=X0
176 PY=Y0
177 IF (.NOT.LINKPEN) THEN
178 CALL RMOVE(0.0,1500.0)
179 CALL RPENCH(P1)
180 ENDF
181
182 DO 50 I=1,2
183 CALL RMOVE(PX,PY)
184 PX=X0+XL
185 CALL RDRAW(PX,PY)
186 PY=Y0+YL
187 CALL RDRAW(PX,PY)
188 PX=X0
189 CALL RDRAW(PX,PY)
190 PY=Y0
191 CALL RDRAW(PX,PY)
192 50 CONTINUE
193
194 DO 60 I=1,N
195 R(I)=0.0

```

```

196     IF (SIGMA(I).NE.0.0) THEN
197     R(I)=(Y(I)-F(THETA,L,X(I)))/SIGMA(I)
198     ENDF
199 60 CONTINUE
200
201 YMAX=R(1)
202 YMIN=R(1)
203 DO 70 I=1,N
204 YMAX=MAX(YMAX,R(I))
205 YMIN=MIN(YMIN,R(I))
206 70 CONTINUE
207
208 YMAX=ABS(YMAX)
209 YMAX=MAX(YMAX,ABS(YMIN))
210 YMIN=-YMAX
211 IF (YMAX-YMIN.LT.1.0E-20) THEN
212 YMAX=YMAX+0.5
213 YMIN=YMIN-0.5
214 ENDF
215
216 X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
217 Y0=NINT(PLYMIN+0.01*(PLYMAX-PLYMIN))
218 XL=NINT(0.99*(PLXMAX-PLXMIN))
219 YL=NINT(0.22*(PLYMAX-PLYMIN))
220
221 IF (.NOT.LINKPEN) THEN
222 CALL RMOVE(0.0,1500.0)
223 CALL RPENCH(P2)
224 ENDF
225
226 DO 80 I=1,N
227 IF (R(I).NE.0.0) THEN
228 PX=XP(X(I))
229 PY=YP(R(I))
230 CALL RMOVE(PX,PY)
231 CALL RMARK(4)
232 ENDF
233 80 CONTINUE
234
235 PX=XP(X(1))
236 PY=YP(0.0)
237 IF (.NOT.LINKPEN) THEN
238 CALL RMOVE(0.0,1500.0)
239 CALL RPENCH(P3)
240 ENDF
241
242 CALL RMOVE(PX,PY)
243 PX=XP(X(N))
244 CALL RDRAW(PX,PY)
245
246 X0=PLXMIN
247 Y0=PLYMIN
248 XL=PLXMAX-PLXMIN
249 YL=NINT(0.24*(PLYMAX-PLYMIN))
250
251 PX=X0
252 PY=Y0
253 IF (.NOT.LINKPEN) THEN
254 CALL RMOVE(0.0,1500.0)
255 CALL RPENCH(P1)

```

```

256     ENDIF
257
258     DO 90 I=1,2
259         CALL RMOVE(PX,PY)
260         PX=XO+XL
261         CALL RDRAW(PX,PY)
262         PY=Y0+YL
263         CALL RDRAW(PX,PY)
264         PX=XO
265         CALL RDRAW(PX,PY)
266         PY=Y0
267         CALL RDRAW(PX,PY)
268 90    CONTINUE
269
270     CALL RRDGLC('VS 23;',6)
271     CALL RPENCH(0)
272     CALL RHOME
273
274     RETURN
275     END

```

IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: DROLLOT.FOR Options: /LBZ 08/02/88 14:18:55

```

1     SUBROUTINE ROLPLOT(Y,SIGMA,X,R,N,F,THETA,L,ERRORBAR)
2 C    Anders Persson, 1988
3
4     IMPLICIT LOGICAL (A-Z)
5     INTEGER FVECRANGE
6     REAL*8 PLXMIN,PLXMAX,PLYMIN,PLYMAX
7     PARAMETER ( FVECRANGE=500 )
8     INTEGER N,L,I,XO,YO,XL,YL,P1,P2,P3
9     REAL*8 Y(1:N),SIGMA(1:N),X(1:N),R(1:N),F,THETA(1:L),XMIN,XMAX
10    REAL*8 FVEC(1:FVECRANGE),SLASK,DX,XI,XP,YP,YMIN,YMAX,A
11    REAL PX,PY,PS
12    INTEGER*2 ANS,KBDCHK,KBDINC
13    LOGICAL ERRORBAR,LINKPEN
14    CHARACTER SV*1,VS*6
15
16    XP(A)=NINT(((A-XMIN)/(XMAX-XMIN))*XL+XO)
17    YP(A)=NINT(((A-YMIN)/(YMAX-YMIN))*YL+YO)
18
19    WRITE(*,*) 'Standard paper size? (Y/*):'
20    READ(*,100) SV
21 100  FORMAT(A)
22
23    IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
24        WRITE(*,*) 'Size A3 or A4 (3/4):'
25        READ(*,100) SV
26        IF (SV.EQ.'3') THEN
27            PLXMIN=0.0
28            PLXMAX=3600.0
29            PLYMIN=0.0
30            PLYMAX=2600.0
31        ELSE
32            PLXMIN=0.0
33            PLXMAX=2600.0
34            PLYMIN=0.0
35            PLYMAX=1800.0
36        ENDIF

```

```

37    ELSE
38 1    WRITE(*,*) 'Enter plot window (in cm.).'
39        WRITE(*,*) 'Xmin, Ymin, Xmax, Ymax:'
40        READ(*,*,ERR=1,END=1) PLXMIN,PLYMIN,PLXMAX,PLYMAX
41        IF (PLXMIN.LT.0.0.OR.PLXMIN.GE.38.0) GOTO 1
42        IF (PLYMIN.LT.0.0.OR.PLYMIN.GE.27.0) GOTO 1
43        IF (PLXMAX.LT.0.0.OR.PLXMAX.GT.38.0) GOTO 1
44        IF (PLYMAX.LT.0.0.OR.PLYMAX.GT.27.0) GOTO 1
45        IF (PLXMIN.GE.PLXMAX) GOTO 1
46        IF (PLYMIN.GE.PLYMAX) GOTO 1
47        PLXMIN=PLXMIN*100.0
48        PLXMAX=PLXMAX*100.0
49        PLYMIN=PLYMIN*100.0
50        PLYMAX=PLYMAX*100.0
51    ENDIF
52
53    P1=1
54    P2=2
55    P3=3
56
57    WRITE(*,*) 'Liquid ink pen? (Y/*):'
58    READ(*,100) SV
59    LINKPEN=SV.EQ.'Y'.OR.SV.EQ.'y'
60
61    IF (LINKPEN) THEN
62        P1=1
63        P2=1
64        P3=1
65        WRITE(*,*) 'Pen speed in cm/s (2..23):'
66        READ(*,*) PS
67        IF (PS.LT.2.0.OR.PS.GT.23.0) THEN
68            PS=8.0
69        ENDIF
70        WRITE(VS,200) 'VS ',NINT(PS),';'
71 200  FORMAT(A3,I2,A1)
72    ENDIF
73
74    IF (LINKPEN) THEN
75        WRITE(*,*) 'Put liquid ink pen in slot 1.'
76    ELSE
77        WRITE(*,*) 'Put pens in slot 1,2 and 3.'
78    ENDIF
79
80    WRITE(*,*) 'When plotter ready press any key!'
81 2    ANS=KBDCHK()
82    IF (ANS.EQ.0) THEN
83        GOTO 2
84    ENDIF
85    ANS=KBDINC()
86
87    IF (LINKPEN) THEN
88        CALL RRDGLC(VS,6)
89    ELSE
90        CALL RRDGLC('VS 23;',6)
91    ENDIF
92
93    XO=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
94    YO=NINT(PLYMIN+0.26*(PLYMAX-PLYMIN))
95    XL=NINT(0.99*(PLXMAX-PLXMIN))
96    YL=NINT(0.73*(PLYMAX-PLYMIN))

```

```

97
98 XMIN=X(1)
99 XMAX=X(N)
100 YMIN=Y(1)
101 YMAX=Y(1)
102
103 DX=(XMAX-XMIN)/(FVECRANGE-1)
104 DO 10 I=1,FVECRANGE
105   XI=(I-1)*DX+XMIN
106   FVEC(I)=F(THETA,L,XI)
107   YMIN=MIN(YMIN,FVEC(I))
108   YMAX=MAX(YMAX,FVEC(I))
109 10 CONTINUE
110
111 DO 20 I=1,N
112   IF (ERRORBAR) THEN
113     YMIN=MIN(YMIN,Y(I)-SIGMA(I))
114     YMAX=MAX(YMAX,Y(I)+SIGMA(I))
115   ELSE
116     YMIN=MIN(YMIN,Y(I))
117     YMAX=MAX(YMAX,Y(I))
118   ENDIF
119 20 CONTINUE
120
121 IF (YMAX-YMIN.LT.1.0D-20) THEN
122   YMAX=YMAX+0.5D0
123   YMIN=YMIN-0.5D0
124 ENDIF
125
126 CALL RLNTYP(0)
127 CALL RASCAL(0)
128 CALL RMOVE(0.0,1500.0)
129 CALL RPENCH(P2)
130
131 DO 30 I=1,N
132   PX=XP(X(I))
133   IF (ERRORBAR) THEN
134     IF (MOD(I,2).EQ.0) THEN
135       PY=YP(Y(I)+SIGMA(I))
136     ELSE
137       PY=YP(Y(I)-SIGMA(I))
138     ENDIF
139   ELSE
140     PY=YP(Y(I))
141   ENDIF
142   CALL RMOVE(PX,PY)
143   IF (ERRORBAR) THEN
144     IF (MOD(I,2).EQ.0) THEN
145       PY=YP(Y(I)-SIGMA(I))
146     ELSE
147       PY=YP(Y(I)+SIGMA(I))
148     ENDIF
149     CALL RDRAW(PX,PY)
150   ELSE
151     CALL RMARK(4)
152   ENDIF
153 30 CONTINUE
154
155 PY=YP(FVEC(1))
156 PX=XP(XMIN)
157 IF (.NOT.LINKPEN) THEN
158   CALL RMOVE(0.0,1500.0)
159   CALL RPENCH(P3)
160 ENDIF
161
162 CALL RMOVE(PX,PY)
163 DO 40 I=1,FVECRANGE
164   XI=(I-1)*DX+XMIN
165   PX=XP(XI)
166   PY=YP(FVEC(I))
167   CALL RDRAW(PX,PY)
168 40 CONTINUE
169
170 X0=PLXMIN
171 Y0=NINT(PLYMIN+0.25*(PLYMAX-PLYMIN))
172 XL=PLXMAX-PLXMIN
173 YL=NINT(0.75*(PLYMAX-PLYMIN))
174
175 PX=X0
176 PY=Y0
177 IF (.NOT.LINKPEN) THEN
178   CALL RMOVE(0.0,1500.0)
179   CALL RPENCH(P1)
180 ENDIF
181
182 DO 50 I=1,2
183   CALL RMOVE(PX,PY)
184   PX=X0+XL
185   CALL RDRAW(PX,PY)
186   PY=Y0+YL
187   CALL RDRAW(PX,PY)
188   PX=X0
189   CALL RDRAW(PX,PY)
190   PY=Y0
191   CALL RDRAW(PX,PY)
192 50 CONTINUE
193
194 DO 60 I=1,N
195   R(I)=0.0D0
196   IF (SIGMA(I).NE.0.0D0) THEN
197     R(I)=(Y(I)-F(THETA,L,X(I)))/SIGMA(I)
198   ENDIF
199 60 CONTINUE
200
201 YMAX=R(1)
202 YMIN=R(1)
203 DO 70 I=1,N
204   YMAX=MAX(YMAX,R(I))
205   YMIN=MIN(YMIN,R(I))
206 70 CONTINUE
207
208 YMAX=ABS(YMAX)
209 YMAX=MAX(YMAX,ABS(YMIN))
210 YMIN=-YMAX
211 IF (YMAX-YMIN.LT.1.0D-20) THEN
212   YMAX=YMAX+0.5D0
213   YMIN=YMIN-0.5D0
214 ENDIF
215
216 X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))

```

```

217 YO=NINT(PLYMIN+0.01*(PLYMAX-PLYMIN))
218 XL=NINT(0.99*(PLXMAX-PLXMIN))
219 YL=NINT(0.22*(PLYMAX-PLYMIN))
220
221 IF (.NOT.LINKPEN) THEN
222   CALL RMOVE(0.0,1500.0)
223   CALL RPENCH(P2)
224 ENDIF
225
226 DO 80 I=1,N
227   IF (R(I).NE.0.0) THEN
228     PX=XP(X(I))
229     PY=YP(R(I))
230     CALL RMOVE(PX,PY)
231     CALL RMARK(4)
232   ENDIF
233 80 CONTINUE
234
235 PX=XP(X(1))
236 PY=YP(0.000)
237 IF (.NOT.LINKPEN) THEN
238   CALL RMOVE(0.0,1500.0)
239   CALL RPENCH(P3)
240 ENDIF
241
242 CALL RMOVE(PX,PY)
243 PX=XP(X(N))
244 CALL RDRAW(PX,PY)
245
246 XO=PLXMIN
247 YO=PLYMIN
248 XL=PLXMAX-PLXMIN
249 YL=NINT(0.24*(PLYMAX-PLYMIN))
250
251 PX=XO
252 PY=YO
253 IF (.NOT.LINKPEN) THEN
254   CALL RMOVE(0.0,1500.0)
255   CALL RPENCH(P1)
256 ENDIF
257
258 DO 90 I=1,2
259   CALL RMOVE(PX,PY)
260   PX=XO+XL
261   CALL RDRAW(PX,PY)
262   PY=YO+YL
263   CALL RDRAW(PX,PY)
264   PX=XO
265   CALL RDRAW(PX,PY)
266   PY=YO
267 90 CONTINUE
268
269 CALL RDRAW(PX,PY)
270 CALL RRDGLC('VS 23;',6)
271 CALL RPENCH(0)
272 CALL RHOME
273
274 RETURN
275 END

```

IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: JETPLOT.FOR Options: /LBZ 08/02/88 14:29:26

```

1 SUBROUTINE JETPLOT(Y,SIGMA,X,R,N,F,THETA,L,ERRORBAR)
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5 INTEGER PLXMIN,PLXMAX,PLYMIN,PLYMAX,FVECRANGE
6 PARAMETER ( FVECRANGE=500 )
7 INTEGER I,N,L,XO,YO,XL,YL
8 CHARACTER MSG*12
9 REAL Y(1:N),SIGMA(1:N),X(1:N),R(1:N),F,THETA(1:L)
10 REAL FVEC(1:FVECRANGE),SLASK,DX,XI,XP,YP
11 REAL YMIN,YMAX,A,PX,PY,XMIN,XMAX
12 LOGICAL ERRORBAR,LANDSCAPE
13 INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
14 COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
15
16 XP(A)=NINT(((A-XMIN)/(XMAX-XMIN))*XL+XO)
17 YP(A)=NINT(((A-YMIN)/(YMAX-YMIN))*YL+YO)
18 C
19 PLXMIN=0
20 PLXMAX=719
21 PLYMIN=0
22 PLYMAX=1119
23 IF (LANDSCAPE) THEN
24   PLXMAX=1119
25   PLYMAX=719
26 ENDIF
27 C
28 XO=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
29 YO=NINT(PLYMIN+0.26*(PLYMAX-PLYMIN))
30 XL=NINT(0.99*(PLXMAX-PLXMIN))
31 YL=NINT(0.73*(PLYMAX-PLYMIN))
32
33 XMIN=X(1)
34 XMAX=X(N)
35 YMIN=Y(1)
36 YMAX=Y(1)
37
38 DX=(XMAX-XMIN)/(FVECRANGE-1)
39 DO 10 I=1,FVECRANGE
40   XI=(I-1)*DX+XMIN
41   FVEC(I)=F(THETA,L,XI)
42   YMIN=MIN(YMIN,FVEC(I))
43   YMAX=MAX(YMAX,FVEC(I))
44 10 CONTINUE
45
46 DO 20 I=1,N
47   IF (ERRORBAR) THEN
48     YMIN=MIN(YMIN,Y(I)-SIGMA(I))
49     YMAX=MAX(YMAX,Y(I)+SIGMA(I))
50   ELSE
51     YMIN=MIN(YMIN,Y(I))
52     YMAX=MAX(YMAX,Y(I))
53   ENDIF
54 20 CONTINUE
55
56 IF (YMAX-YMIN.LT.1.0E-20) THEN
57   YMAX=YMAX+0.5

```

```

58     YMIN=YMIN-0.5
59     ENDIF
60
61     DO 30 I=1,N
62     PX=XP(X(I))
63     IF (ERRORBAR) THEN
64     PY=YP(Y(I)+SIGMA(I))
65     ELSE
66     PY=YP(Y(I))
67     ENDIF
68     CALL HPMOVE(PX,PY)
69     IF (ERRORBAR) THEN
70     PY=YP(Y(I)-SIGMA(I))
71     CALL HPDRAW(PX,PY)
72     ELSE
73     CALL HPMARK(PX,PY)
74     ENDIF
75 30  CONTINUE
76
77     PY=YP(FVEC(1))
78     PX=XP(XMIN)
79     CALL HPMOVE(PX,PY)
80     DO 40 I=1,FVECRANGE
81     XI=(I-1)*DX+XMIN
82     PX=XP(XI)
83     PY=YP(FVEC(I))
84     CALL HPDRAW(PX,PY)
85 40  CONTINUE
86
87     X0=PLXMIN
88     Y0=NINT(PLYMIN+0.25*(PLYMAX-PLYMIN))
89     XL=PLXMAX-PLXMIN
90     YL=NINT(0.75*(PLYMAX-PLYMIN))
91
92     PX=X0
93     PY=Y0
94     CALL HPMOVE(PX,PY)
95     PX=X0+XL
96     CALL HPDRAW(PX,PY)
97     PY=Y0+YL
98     CALL HPDRAW(PX,PY)
99     PX=X0
100    CALL HPDRAW(PX,PY)
101    PY=Y0
102    CALL HPDRAW(PX,PY)
103
104    WRITE(MSG,100) YMIN
105 100  FORMAT(F12.2)
106    CALL HPTBOX(PX,PY,12,MSG)
107
108    PY=Y0+YL-13
109    WRITE(MSG,100) YMAX
110    CALL HPTBOX(PX,PY,12,MSG)
111
112    DO 50 I=1,N
113    R(I)=0.0
114    IF (SIGMA(I).NE.0.0) THEN
115    R(I)=(Y(I)-F(THETA,L,X(I)))/SIGMA(I)
116    ENDIF
117 50  CONTINUE

```

```

118
119     YMAX=R(1)
120     YMIN=R(1)
121     DO 60 I=1,N
122     YMAX=MAX(YMAX,R(I))
123     YMIN=MIN(YMIN,R(I))
124 60  CONTINUE
125
126     YMAX=ABS(YMAX)
127     YMAX=MAX(YMAX,ABS(YMIN))
128     YMIN=-YMAX
129     IF (YMAX-YMIN.LT.1.0E-20) THEN
130     YMAX=YMAX+0.5
131     YMIN=YMIN-0.5
132     ENDIF
133
134     X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
135     Y0=NINT(PLYMIN+0.01*(PLYMAX-PLYMIN))
136     XL=NINT(0.99*(PLXMAX-PLXMIN))
137     YL=NINT(0.22*(PLYMAX-PLYMIN))
138
139     DO 70 I=1,N
140     IF (R(I).NE.0.0) THEN
141     PX=XP(X(I))
142     PY=YP(R(I))
143     CALL HPMARK(PX,PY)
144     ENDIF
145 70  CONTINUE
146
147     PX=XP(X(1))
148     PY=YP(0.0)
149     CALL HPMOVE(PX,PY)
150     PX=XP(X(N))
151     CALL HPDRAW(PX,PY)
152
153     X0=PLXMIN
154     Y0=PLYMIN
155     XL=PLXMAX-PLXMIN
156     YL=NINT(0.24*(PLYMAX-PLYMIN))
157
158     PX=X0
159     PY=Y0
160     CALL HPMOVE(PX,PY)
161     PX=X0+XL
162     CALL HPDRAW(PX,PY)
163     PY=Y0+YL
164     CALL HPDRAW(PX,PY)
165     PX=X0
166     CALL HPDRAW(PX,PY)
167     PY=Y0
168     CALL HPDRAW(PX,PY)
169
170     WRITE(MSG,100) YMIN
171     CALL HPTBOX(PX,PY,12,MSG)
172
173     PY=Y0+YL-13
174     WRITE(MSG,100) YMAX
175     CALL HPTBOX(PX,PY,12,MSG)
176
177     RETURN

```


178 END

IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: DJETPLOT.FOR Options: /LBZ 08/02/88 14:16:35

```
1 SUBROUTINE JETPLOT(Y,SIGMA,X,R,N,F,THETA,L,ERRORBAR)
2 C Anders Persson, 1988
3
4 IMPLICIT LOGICAL (A-Z)
5 INTEGER PLXMIN,PLXMAX,PLYMIN,PLYMAX,FVECRANGE
6 PARAMETER ( FVECRANGE=500 )
7 INTEGER I,N,L,X0,Y0,XL,YL
8 CHARACTER MSG*12
9 REAL*8 Y(1:N),SIGMA(1:N),X(1:N),R(1:N),F,THETA(1:L)
10 REAL*8 FVEC(1:FVECRANGE),SLASK,DX,XI,XP,YP
11 REAL*8 YMIN,YMAX,A,XMIN,XMAX
12 REAL PX,PY
13 LOGICAL ERRORBAR,LANDSCAPE
14 INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
15 COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
16
17 XP(A)=NINT(((A-XMIN)/(XMAX-XMIN))*XL+X0)
18 YP(A)=NINT(((A-YMIN)/(YMAX-YMIN))*YL+Y0)
19 C
20 PLXMIN=0
21 PLXMAX=719
22 PLYMIN=0
23 PLYMAX=1119
24 IF (LANDSCAPE) THEN
25 PLXMAX=1119
26 PLYMAX=719
27 ENDIF
28 C
29 X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
30 Y0=NINT(PLYMIN+0.26*(PLYMAX-PLYMIN))
31 XL=NINT(0.99*(PLXMAX-PLXMIN))
32 YL=NINT(0.73*(PLYMAX-PLYMIN))
33
34 XMIN=X(1)
35 XMAX=X(N)
36 YMIN=Y(1)
37 YMAX=Y(1)
38
39 DX=(XMAX-XMIN)/(FVECRANGE-1)
40 DO 10 I=1,FVECRANGE
41 XI=(I-1)*DX+XMIN
42 FVEC(I)=F(THETA,L,XI)
43 YMIN=MIN(YMIN,FVEC(I))
44 YMAX=MAX(YMAX,FVEC(I))
45 10 CONTINUE
46
47 DO 20 I=1,N
48 IF (ERRORBAR) THEN
49 YMIN=MIN(YMIN,Y(I)-SIGMA(I))
50 YMAX=MAX(YMAX,Y(I)+SIGMA(I))
51 ELSE
52 YMIN=MIN(YMIN,Y(I))
53 YMAX=MAX(YMAX,Y(I))
54 ENDIF
55 20 CONTINUE
```

```
56
57 IF (YMAX-YMIN.LT.1.0E-20) THEN
58 YMAX=YMAX+0.5
59 YMIN=YMIN-0.5
60 ENDF
61
62 DO 30 I=1,N
63 PX=XP(X(I))
64 IF (ERRORBAR) THEN
65 PY=YP(Y(I)+SIGMA(I))
66 ELSE
67 PY=YP(Y(I))
68 ENDF
69 CALL HPMOVE(PX,PY)
70 IF (ERRORBAR) THEN
71 PY=YP(Y(I)-SIGMA(I))
72 CALL HPDRAW(PX,PY)
73 ELSE
74 CALL HPMARK(PX,PY)
75 ENDF
76 30 CONTINUE
77
78 PY=YP(FVEC(1))
79 PX=XP(XMIN)
80 CALL HPMOVE(PX,PY)
81 DO 40 I=1,FVECRANGE
82 XI=(I-1)*DX+XMIN
83 PX=XP(XI)
84 PY=YP(FVEC(I))
85 CALL HPDRAW(PX,PY)
86 40 CONTINUE
87
88 X0=PLXMIN
89 Y0=NINT(PLYMIN+0.25*(PLYMAX-PLYMIN))
90 XL=PLXMAX-PLXMIN
91 YL=NINT(0.75*(PLYMAX-PLYMIN))
92
93 PX=X0
94 PY=Y0
95 CALL HPMOVE(PX,PY)
96 PX=X0+XL
97 CALL HPDRAW(PX,PY)
98 PY=Y0+YL
99 CALL HPDRAW(PX,PY)
100 PX=X0
101 CALL HPDRAW(PX,PY)
102 PY=Y0
103 CALL HPDRAW(PX,PY)
104
105 WRITE(MSG,100) YMIN
106 100 FORMAT(F12.2)
107 CALL HPTBOX(PX,PY,12,MSG)
108
109 PY=Y0+YL-13
110 WRITE(MSG,100) YMAX
111 CALL HPTBOX(PX,PY,12,MSG)
112
113 DO 50 I=1,N
114 R(I)=0.0
115 IF (SIGMA(I).NE.0.0) THEN
```

```

116      R(I)=(Y(I)-F(THETA,L,X(I)))/SIGMA(I)
117      ENDIF
118 50   CONTINUE
119
120     YMAX=R(1)
121     YMIN=R(1)
122     DO 60 I=1,N
123         YMAX=MAX(YMAX,R(I))
124         YMIN=MIN(YMIN,R(I))
125 60   CONTINUE
126
127     YMAX=ABS(YMAX)
128     YMAX=MAX(YMAX,ABS(YMIN))
129     YMIN=-YMAX
130     IF (YMAX-YMIN.LT.1.0E-20) THEN
131         YMAX=YMAX+0.5
132         YMIN=YMIN-0.5
133     ENDIF
134
135     X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
136     Y0=NINT(PLYMIN+0.01*(PLYMAX-PLYMIN))
137     XL=NINT(0.99*(PLXMAX-PLXMIN))
138     YL=NINT(0.22*(PLYMAX-PLYMIN))
139
140     DO 70 I=1,N
141         IF (R(I).NE.0.0) THEN
142             PX=XP(X(I))
143             PY=YP(R(I))
144             CALL HPMARK(PX,PY)
145         ENDIF
146 70   CONTINUE
147
148     PX=XP(X(1))
149     PY=YP(0.0)
150     CALL HPMOVE(PX,PY)
151     PX=XP(X(N))
152     CALL HPDRAW(PX,PY)
153
154     X0=PLXMIN
155     Y0=PLYMIN
156     XL=PLXMAX-PLXMIN
157     YL=NINT(0.24*(PLYMAX-PLYMIN))
158
159     PX=X0
160     PY=Y0
161     CALL HPMOVE(PX,PY)
162     PX=X0+XL
163     CALL HPDRAW(PX,PY)
164     PY=Y0+YL
165     CALL HPDRAW(PX,PY)
166     PX=X0
167     CALL HPDRAW(PX,PY)
168     PY=Y0
169     CALL HPDRAW(PX,PY)
170
171     WRITE(MSG,100) YMIN
172     CALL HPTBOX(PX,PY,12,MSG)
173
174     PY=Y0+YL-13
175     WRITE(MSG,100) YMAX

```

```

176     CALL HPTBOX(PX,PY,12,MSG)
177
178     RETURN
179     END

```

HERCLIB är ett bibliotek bestående av 17 rutiner för att rita på datorskärmen. Grafikkortet skall vara Hercules-kompatibelt och man måste ladda de minnesresidenta programmen HGC.COM och INT10.COM från Hercules innan program som använder rutinerna kan köras. Skärmen har koordinaterna 0..719 i horisontell led och 0..347 i vertikal. Rutinerna använder nedre vänstra hörnet som origo (0,0). För ytterligare information se manualen till grafikkortet och manualen till GRAPH X programvaran från Hercules. Förutom ritrutinerna finns det tre rutiner (KBDCHK, KBDIN och KBDINC) som läser av tangentbordet via anrop av DOS-rutiner (Se IBM DOS Technical Reference Manual och IBM PC Technical Reference Manual).

Beskrivning av rutinerna.

ARC	Ritar ett cirkelsegment.
BLKFIL	Ritar en fylld rektangel.
CIRC	Ritar en cirkel.
CLRSCR	Suddar skärmen.
DISP	Väljer grafiksida (0 eller 1) för visning.
DLINE	Ritar en vektor.
FILL	Fyller ett område med färg.
GETPT	Tar reda på om en pixel är fylld.
GMODE	Ändrar skärmen till grafisk mod.
GPAGE	Väljer vilken grafiksida ritkommandon skall verka på.
LEVEL	Väljer vilken färg ritkommandon skall rita med (0..2).
MOVE	Flyttar cursorn utan att rita.
PLOT	Ritar en pixel.
TMODE	Sätter kortet i textmod.
HRDCPY	Dumpar en grafiksida till skrivare.
ONECHR	Skriver en bokstav på grafiksidan. (Används av TEXT).
TEXT	Skriver en teckensträng på grafiksidan.
KBDCHK	Ger värde skilt från noll om en tangent varit nedtryckt.
KBDIN	Ger koden för den tangent som tryckts ned.
KBDINC	Ger koden för den tangent som tryckts ned. Reagerar på Ctrl-C.

```
1 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
2 C                                     C
3 C      Library:  HERCLIB.LIB          C
4 C                                     C
5 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
6 C
7 C          TITLE      ARC
8 C      ; Anders Persson, 1986
9 C      ; FORTRAN:  SUBROUTINE ARC(X,Y,R,Q)
10 C      ;          INTEGER*2 X,Y,R,Q
11 C      ;
12 C  PARBLK      STRUC
13 C              PARX      DD      ?
14 C              PARY      DD      ?
15 C              PARR      DD      ?
16 C              PARQ      DD      ?
17 C  PARBLK      ENDS
18 C  LADATA      SEGMENT  'DATA'
19 C              DB        'ARC'
20 C  SP_SAVE     DW        0
21 C              DD        0
22 C  LADATA      ENDS
23 C  PCODE       SEGMENT  'CODE'
24 C              ASSUME   CS:PCODE,DS:LADATA
25 C              DW      SEG LADATA
26 C  ARC         PROC     FAR
27 C              PUBLIC   ARC
28 C              MOV     AX,LADATA
29 C              MOV     DS,AX
30 C              MOV     SP,SAVE,SP
31 C              LDS     SI,ES:PARX[BX]
32 C              MOV     DI,[SI]
33 C              LDS     SI,ES:PARY[BX]
34 C              MOV     BP,347
35 C              SUB     BP,[SI]
36 C              LDS     SI,ES:PARQ[BX]
37 C              MOV     AX,[SI]
38 C              LDS     SI,ES:PARR[BX]
39 C              MOV     BX,[SI]
40 C              MOV     AH,4CH
41 C              INT    10H
42 C              RET
43 C  ARC         ENDP
44 C  PCODE       ENDS
45 C              END
46 C      ;
47 C          TITLE      BLKFIL
48 C      ; Anders Persson, 1986
49 C      ; FORTRAN:  SUBROUTINE BLKFIL(X,Y,W,H)
50 C      ;          INTEGER*2 X,Y,W,H
51 C      ;
52 C  PARBLK      STRUC
53 C              PARX      DD      ?
54 C              PARY      DD      ?
55 C              PARW      DD      ?
56 C              PARH      DD      ?
57 C  PARBLK      ENDS
58 C  LADATA      SEGMENT  'DATA'
59 C              DB        'BLKFIL'
60 C  SP_SAVE     DW        0
```

```
61 C          DD        0
62 C  LADATA      ENDS
63 C  PCODE       SEGMENT  'CODE'
64 C              ASSUME   CS:PCODE,DS:LADATA
65 C              DW      SEG LADATA
66 C  BLKFIL     PROC     FAR
67 C              PUBLIC   BLKFIL
68 C              MOV     AX,LADATA
69 C              MOV     DS,AX
70 C              MOV     SP,SAVE,SP
71 C              LDS     SI,ES:PARX[BX]
72 C              MOV     DI,[SI]
73 C              LDS     SI,ES:PARY[BX]
74 C              MOV     BP,347
75 C              SUB     BP,[SI]
76 C              LDS     SI,ES:PARW[BX]
77 C              MOV     CX,[SI]
78 C              LDS     SI,ES:PARH[BX]
79 C              MOV     BX,[SI]
80 C              MOV     AH,4AH
81 C              INT    10H
82 C              RET
83 C  BLKFIL     ENDP
84 C  PCODE       ENDS
85 C              END
86 C      ;
87 C          TITLE      CIRC
88 C      ; Anders Persson, 1986
89 C      ; FORTRAN:  SUBROUTINE CIRC(X,Y,R)
90 C      ;          INTEGER*2 X,Y,R
91 C      ;
92 C  PARBLK      STRUC
93 C              PARX      DD      ?
94 C              PARY      DD      ?
95 C              PARR      DD      ?
96 C  PARBLK      ENDS
97 C  LADATA      SEGMENT  'DATA'
98 C              DB        'CIRC'
99 C  SP_SAVE     DW        0
100 C             DD        0
101 C  LADATA      ENDS
102 C  PCODE       SEGMENT  'CODE'
103 C              ASSUME   CS:PCODE,DS:LADATA
104 C              DW      SEG LADATA
105 C  CIRC        PROC     FAR
106 C              PUBLIC   CIRC
107 C              MOV     AX,LADATA
108 C              MOV     DS,AX
109 C              MOV     SP,SAVE,SP
110 C              LDS     SI,ES:PARX[BX]
111 C              MOV     DI,[SI]
112 C              LDS     SI,ES:PARY[BX]
113 C              MOV     BP,347
114 C              SUB     BP,[SI]
115 C              LDS     SI,ES:PARR[BX]
116 C              MOV     BX,[SI]
117 C              MOV     AH,4DH
118 C              INT    10H
119 C              RET
120 C  CIRC        ENDP
```

```

121 C PCODE ENDS
122 C END
123 C ;
124 C ; TITLE CLRSCR
125 C ; Anders Persson, 1986
126 C ; FORTRAN: SUBROUTINE CLRSCR
127 C ;
128 C LADATA SEGMENT 'DATA'
129 C DB 'CLRSCR '
130 C SP_SAVE DW 0
131 C DD 0
132 C LADATA ENDS
133 C PCODE SEGMENT 'CODE'
134 C ASSUME CS:PCODE,DS:LADATA
135 C DW SEG LADATA
136 C CLRSCR PROC FAR
137 C PUBLIC CLRSCR
138 C MOV AX,LADATA
139 C MOV DS,AX
140 C MOV SP_SAVE,SP
141 C MOV AH,42H
142 C INT 10H
143 C RET
144 C CLRSCR ENDP
145 C PCODE ENDS
146 C END
147 C ;
148 C ; TITLE DISP
149 C ; Anders Persson, 1986
150 C ; FORTRAN: SUBROUTINE DISP(P)
151 C ; INTEGER*2 P
152 C ;
153 C PARBLK STRUC
154 C PARB DD ?
155 C PARBLK ENDS
156 C LADATA SEGMENT 'DATA'
157 C DB 'DISP '
158 C SP_SAVE DW 0
159 C DD 0
160 C LADATA ENDS
161 C PCODE SEGMENT 'CODE'
162 C ASSUME CS:PCODE,DS:LADATA
163 C DW SEG LADATA
164 C DISP PROC FAR
165 C PUBLIC DISP
166 C MOV AX,LADATA
167 C MOV DS,AX
168 C MOV SP_SAVE,SP
169 C LDS SI,ES:PARB[BX]
170 C MOV AX,[SI]
171 C MOV AH,45H
172 C INT 10H
173 C RET
174 C DISP ENDP
175 C PCODE ENDS
176 C END
177 C ;
178 C ; TITLE DLINE
179 C ; Anders Persson, 1986
180 C ; FORTRAN: SUBROUTINE DLINE(X,Y)

```

```

181 C ; INTEGER*2 X,Y
182 C ;
183 C ; PARBLK STRUC
184 C PARX DD ?
185 C PARY DD ?
186 C PARBLK ENDS
187 C LADATA SEGMENT 'DATA'
188 C DB 'DLINE '
189 C SP_SAVE DW 0
190 C DD 0
191 C LADATA ENDS
192 C PCODE SEGMENT 'CODE'
193 C ASSUME CS:PCODE,DS:LADATA
194 C DW SEG LADATA
195 C DLINE PROC FAR
196 C PUBLIC DLINE
197 C MOV AX,LADATA
198 C MOV DS,AX
199 C MOV SP_SAVE,SP
200 C LDS SI,ES:PARX[BX]
201 C MOV DI,[SI]
202 C LDS SI,ES:PARY[BX]
203 C MOV BP,347
204 C SUB BP,[SI]
205 C MOV AH,49H
206 C INT 10H
207 C RET
208 C DLINE ENDP
209 C PCODE ENDS
210 C END
211 C ;
212 C ; TITLE FILL
213 C ; Anders Persson, 1986
214 C ; FORTRAN: SUBROUTINE FILL(X,Y)
215 C ; INTEGER*2 X,Y
216 C ;
217 C PARBLK STRUC
218 C PARX DD ?
219 C PARY DD ?
220 C PARBLK ENDS
221 C LADATA SEGMENT 'DATA'
222 C DB 'FILL '
223 C SP_SAVE DW 0
224 C DD 0
225 C LADATA ENDS
226 C PCODE SEGMENT 'CODE'
227 C ASSUME CS:PCODE,DS:LADATA
228 C DW SEG LADATA
229 C FILL PROC FAR
230 C PUBLIC FILL
231 C MOV AX,LADATA
232 C MOV DS,AX
233 C MOV SP_SAVE,SP
234 C LDS SI,ES:PARX[BX]
235 C MOV DI,[SI]
236 C LDS SI,ES:PARY[BX]
237 C MOV BP,347
238 C SUB BP,[SI]
239 C MOV AH,4EH
240 C INT 10H

```

```

241 C      RET
242 C      FILL  ENDP
243 C      PCODE ENDS
244 C      END
245 C      ;
246 C      TITLE      GETPT
247 C      ; Anders Persson, 1986
248 C      ; FORTRAN:  INTEGER*2 FUNCTION GETPT(X,Y)
249 C      ;           INTEGER*2 X,Y
250 C      ;
251 C      PARBLK     STRUC
252 C      PARX      DD      ?
253 C      PARY      DD      ?
254 C      PARBLK     ENDS
255 C      LADATA     SEGMENT 'DATA'
256 C      DB        'GETPT '
257 C      SP_SAVE    DW      0
258 C      DD        0
259 C      LADATA     ENDS
260 C      PCODE     SEGMENT 'CODE'
261 C      ASSUME    CS:PCODE,DS:LADATA
262 C      DW        SEG LADATA
263 C      GETPT     PROC    FAR
264 C      PUBLIC   GETPT
265 C      MOV      AX,LADATA
266 C      MOV      DS,AX
267 C      MOV      SP_SAVE,SP
268 C      LDS     SI,ES:PARX[BX]
269 C      MOV     DI,[SI]
270 C      LDS     SI,ES:PARY[BX]
271 C      MOV     BP,347
272 C      SUB     BP,[SI]
273 C      MOV     AH,47H
274 C      INT     10H
275 C      MOV     AH,0H
276 C      RET
277 C      GETPT     ENDP
278 C      PCODE     ENDS
279 C      END
280 C      ;
281 C      TITLE      GMODE
282 C      ; Anders Persson, 1986
283 C      ; FORTRAN:  SUBROUTINE GMODE
284 C      ;
285 C      LADATA     SEGMENT 'DATA'
286 C      DB        'GMODE '
287 C      SP_SAVE    DW      0
288 C      DD        0
289 C      LADATA     ENDS
290 C      PCODE     SEGMENT 'CODE'
291 C      ASSUME    CS:PCODE,DS:LADATA
292 C      DW        SEG LADATA
293 C      GMODE     PROC    FAR
294 C      PUBLIC   GMODE
295 C      MOV      AX,LADATA
296 C      MOV      DS,AX
297 C      MOV      SP_SAVE,SP
298 C      MOV      AH,40H
299 C      INT     10H
300 C      RET

```

```

301 C      GMODE     ENDP
302 C      PCODE     ENDS
303 C      END
304 C      ;
305 C      TITLE      GPAGE
306 C      ; Anders Persson, 1986
307 C      ; FORTRAN:  SUBROUTINE GPAGE(P)
308 C      ;           INTEGER*2 P
309 C      ;
310 C      PARBLK     STRUC
311 C      PARB      DD      ?
312 C      PARBLK     ENDS
313 C      LADATA     SEGMENT 'DATA'
314 C      DB        'GPAGE '
315 C      SP_SAVE    DW      0
316 C      DD        0
317 C      LADATA     ENDS
318 C      PCODE     SEGMENT 'CODE'
319 C      ASSUME    CS:PCODE,DS:LADATA
320 C      DW        SEG LADATA
321 C      GPAGE     PROC    FAR
322 C      PUBLIC   GPAGE
323 C      MOV      AX,LADATA
324 C      MOV      DS,AX
325 C      MOV      SP_SAVE,SP
326 C      LDS     SI,ES:PARB[BX]
327 C      MOV     AX,[SI]
328 C      MOV     AH,43H
329 C      INT     10H
330 C      RET
331 C      GPAGE     ENDP
332 C      PCODE     ENDS
333 C      END
334 C      ;
335 C      TITLE      LEVEL
336 C      ; Anders Persson, 1986
337 C      ; FORTRAN:  SUBROUTINE LEVEL(I)
338 C      ;           INTEGER*2 I
339 C      ;
340 C      PARBLK     STRUC
341 C      PARI      DD      ?
342 C      PARBLK     ENDS
343 C      LADATA     SEGMENT 'DATA'
344 C      DB        'LEVEL '
345 C      SP_SAVE    DW      0
346 C      DD        0
347 C      LADATA     ENDS
348 C      PCODE     SEGMENT 'CODE'
349 C      ASSUME    CS:PCODE,DS:LADATA
350 C      DW        SEG LADATA
351 C      LEVEL     PROC    FAR
352 C      PUBLIC   LEVEL
353 C      MOV      AX,LADATA
354 C      MOV      DS,AX
355 C      MOV      SP_SAVE,SP
356 C      LDS     SI,ES:PARI[BX]
357 C      MOV     AX,[SI]
358 C      MOV     AH,44H
359 C      INT     10H
360 C      RET

```

```

361 C LEVEL      ENDP
362 C PCODE      ENDS
363 C           END
364 C ;
365 C           TITLE      MOVE
366 C ; Anders Persson, 1986
367 C ; FORTRAN:  SUBROUTINE MOVE(X,Y)
368 C ;           INTEGER*2 X,Y
369 C ;
370 C PARBLK      STRUC
371 C           PARX      DD      ?
372 C           PARY      DD      ?
373 C PARBLK      ENDS
374 C LADATA      SEGMENT  'DATA'
375 C           DB        'MOVE'
376 C SP_SAVE     DW        0
377 C           DD        0
378 C LADATA      ENDS
379 C PCODE      SEGMENT  'CODE'
380 C           ASSUME    CS:PCODE,DS:LADATA
381 C           DW        SEG LADATA
382 C MOVE        PROC     FAR
383 C           PUBLIC   MOVE
384 C           MOV      AX,LADATA
385 C           MOV      DS,AX
386 C           MOV      SP_SAVE,SP
387 C           LDS     SI,ES:PARX[BX]
388 C           MOV     DI,[SI]
389 C           LDS     SI,ES:PARY[BX]
390 C           MOV     BP,347
391 C           SUB     BP,[SI]
392 C           MOV     AH,48H
393 C           INT     10H
394 C           RET
395 C MOVE        ENDP
396 C PCODE      ENDS
397 C           END
398 C ;
399 C           TITLE      PLOT
400 C ; Anders Persson, 1986
401 C ; FORTRAN:  SUBROUTINE PLOT(X,Y)
402 C ;           INTEGER*2 X,Y
403 C ;
404 C PARBLK      STRUC
405 C           PARX      DD      ?
406 C           PARY      DD      ?
407 C PARBLK      ENDS
408 C LADATA      SEGMENT  'DATA'
409 C           DB        'PLOT'
410 C SP_SAVE     DW        0
411 C           DD        0
412 C LADATA      ENDS
413 C PCODE      SEGMENT  'CODE'
414 C           ASSUME    CS:PCODE,DS:LADATA
415 C           DW        SEG LADATA
416 C PLOT        PROC     FAR
417 C           PUBLIC   PLOT
418 C           MOV      AX,LADATA
419 C           MOV      DS,AX
420 C           MOV      SP_SAVE,SP

```

```

421 C           LDS     SI,ES:PARX[BX]
422 C           MOV     DI,[SI]
423 C           LDS     SI,ES:PARY[BX]
424 C           MOV     BP,347
425 C           SUB     BP,[SI]
426 C           MOV     AH,46H
427 C           INT     10H
428 C           RET
429 C PLOT        ENDP
430 C PCODE      ENDS
431 C           END
432 C ;
433 C           TITLE      TMODE
434 C ; Anders Persson, 1986
435 C ; FORTRAN:  SUBROUTINE TMODE
436 C ;
437 C LADATA      SEGMENT  'DATA'
438 C           DB        'TMODE'
439 C SP_SAVE     DW        0
440 C           DD        0
441 C LADATA      ENDS
442 C PCODE      SEGMENT  'CODE'
443 C           ASSUME    CS:PCODE,DS:LADATA
444 C           DW        SEG LADATA
445 C TMODE       PROC     FAR
446 C           PUBLIC   TMODE
447 C           MOV      AX,LADATA
448 C           MOV      DS,AX
449 C           MOV      SP_SAVE,SP
450 C           MOV      AH,41H
451 C           INT     10H
452 C           RET
453 C TMODE       ENDP
454 C PCODE      ENDS
455 C           END
456 C ;
457 C           TITLE      HRDCPY
458 C ; Anders Persson, 1986
459 C ; FORTRAN:  SUBROUTINE HRDCPY(B)
460 C ;           INTEGER*2 B
461 C ;
462 C PARBLK      STRUC
463 C           PARB      DD      ?
464 C PARBLK      ENDS
465 C LADATA      SEGMENT  'DATA'
466 C           DB        'HRDCPY'
467 C SP_SAVE     DW        0
468 C           DD        0
469 C LADATA      ENDS
470 C PCODE      SEGMENT  'CODE'
471 C           ASSUME    CS:PCODE,DS:LADATA
472 C           DW        SEG LADATA
473 C HRDCPY      PROC     FAR
474 C           PUBLIC   HRDCPY
475 C           MOV      AX,LADATA
476 C           MOV      DS,AX
477 C           MOV      SP_SAVE,SP
478 C           LDS     SI,ES:PARB[BX]
479 C           MOV     BX,[SI]
480 C           MOV     BH,00H

```

```

481 C      XOR      AX,AX
482 C      MOV      ES,AX
483 C      MOV      BP,2+4*05H
484 C      MOV      AX,ES:[BP]
485 C      MOV      ES,AX
486 C      MOV      BP,130H
487 C      MOV      ES:[BP],BL
488 C      INT      5
489 C      RET
490 C  HRDCPY  ENDP
491 C  PCODE   ENDS
492 C      END
493 C      ;
494 C      TITLE   ONECHR
495 C      ; Anders Persson, 1986
496 C      ; FORTRAN: SUBROUTINE ONECHR(X,Y,I)
497 C      ;          INTEGER*2 X,Y,I
498 C      ;
499 C  PARBLK  STRUC
500 C          PARX      DD      ?
501 C          PARY      DD      ?
502 C          PARI      DD      ?
503 C  PARBLK  ENDS
504 C  LADATA  SEGMENT 'DATA'
505 C          DB          'ONECHR '
506 C  SP_SAVE DW      0
507 C          DD      0
508 C  LADATA  ENDS
509 C  PCODE   SEGMENT 'CODE'
510 C          ASSUME     CS:PCODE,DS:LADATA
511 C          DW          SEG LADATA
512 C  ONECHR  PROC     FAR
513 C          PUBLIC    ONECHR
514 C          MOV      AX,LADATA
515 C          MOV      DS,AX
516 C          MOV      SP_SAVE,SP
517 C          LDS     SI,ES:PARX[BX]
518 C          MOV     DI,[SI]
519 C          LDS     SI,ES:PARY[BX]
520 C          MOV     BP,347
521 C          SUB     BP,[SI]
522 C          LDS     SI,ES:PARI[BX]
523 C          MOV     AX,[SI]
524 C          MOV     AH,4BH
525 C          INT     10H
526 C          RET
527 C  ONECHR  ENDP
528 C  PCODE   ENDS
529 C          END
530 C
531 C      SUBROUTINE TEXT(X,Y,LEN,MSG)
532 C      Anders Persson, 1986
533 C
534 C      IMPLICIT LOGICAL (A-Z)
535 C      INTEGER*2 X,Y,LEN,IASC,IX,IY,I
536 C      CHARACTER*(*) MSG
537 C
538 C      DO 10 I=1,LEN
539 C          IX=X+(I-1)*9
540 C          IY=Y

```

```

541 C      IASC=ICHAR(MSG(I:I))
542 C      CALL ONECHR(IX,IY,IASC)
543 C10     CONTINUE
544 C
545 C      RETURN
546 C      END
547 C
548 C      TITLE   KBDCHK
549 C      ; Anders Persson, 1986
550 C      ; FORTRAN: INTEGER*2 FUNCTION KBDCHK()
551 C      ;
552 C  LADATA  SEGMENT 'DATA'
553 C          DB          'KBDCHK '
554 C  SP_SAVE DW      0
555 C          DD      0
556 C  LADATA  ENDS
557 C  PCODE   SEGMENT 'CODE'
558 C          ASSUME     CS:PCODE,DS:LADATA
559 C          DW          SEG LADATA
560 C  KBDCHK  PROC     FAR
561 C          PUBLIC    KBDCHK
562 C          MOV      AX,LADATA
563 C          MOV      DS,AX
564 C          MOV      SP_SAVE,SP
565 C          MOV      AH,0BH
566 C          INT     21H
567 C          MOV      AH,00H
568 C          RET
569 C  KBDCHK  ENDP
570 C  PCODE   ENDS
571 C          END
572 C      ;
573 C      TITLE   KBDIN
574 C      ; Anders Persson, 1986
575 C      ; FORTRAN: INTEGER*2 FUNCTION KBDIN()
576 C      ;
577 C  LADATA  SEGMENT 'DATA'
578 C          DB          'KBDIN '
579 C  SP_SAVE DW      0
580 C          DD      0
581 C  LADATA  ENDS
582 C  PCODE   SEGMENT 'CODE'
583 C          ASSUME     CS:PCODE,DS:LADATA
584 C          DW          SEG LADATA
585 C  KBDIN   PROC     FAR
586 C          PUBLIC    KBDIN
587 C          MOV      AX,LADATA
588 C          MOV      DS,AX
589 C          MOV      SP_SAVE,SP
590 C          MOV      AH,07H
591 C          INT     21H
592 C          MOV      AH,0H
593 C          CMP     AL,0H
594 C          JZ      AGAIN
595 C          RET
596 C  AGAIN:  MOV      AH,07H      ;EXTENDED ASCII CODE
597 C          INT     21H      ;F1..F10, CURSOR KEYS ETC.
598 C          MOV      AH,0H
599 C          ADD     AX,1000
600 C          RET

```



```

601 C   KBDIN      ENDP
602 C   PCODE      ENDS
603 C   END
604 C   ;
605 C           TITLE      KBDINC
606 C   ; Anders Persson, 1986
607 C   ; FORTRAN:  INTEGER*2 FUNCTION KBDINC()
608 C   ;
609 C   LADATA      SEGMENT  'DATA'
610 C           DB          'KBDINC '
611 C   SP_SAVE      DW          0
612 C           DD          0
613 C   LADATA      ENDS
614 C   PCODE      SEGMENT  'CODE'
615 C           ASSUME      CS:PCODE,DS:LADATA
616 C           DW          SEG LADATA
617 C   KBDINC      PROC      FAR
618 C           PUBLIC      KBDINC
619 C           MOV          AX,LADATA
620 C           MOV          DS,AX
621 C           MOV          SP_SAVE,SP
622 C           MOV          AH,08H
623 C           INT          21H
624 C           MOV          AH,0H
625 C           CMP          AL,0H
626 C           JZ          AGAIN
627 C           RET
628 C   AGAIN:      MOV          AH,08H      ;EXTENDED ASCII CODE
629 C           INT          21H          ;F1..F10, CURSOR KEYS ETC.
630 C           MOV          AH,0H
631 C           ADD          AX,1000
632 C           RET
633 C   KBDINC      ENDP
634 C   PCODE      ENDS
635 C   END

```

PLLIB

Biblioteket innehåller 36 rutiner för att rita på en Roland DXY-980 plotter eller en Watanabe DIGI-PLOT Model W4671 plotter. Alla rutiner till Watanabe-plottern börjar på bokstaven W och alla till Roland-plottern börjar med bokstaven R. För närmare beskrivningar till de båda plottrarna se respektive tillverkares manualer. (OBS Roland plottern skall användas i DXY-mode och vara kopplad till datorn med en vanlig skrivarkabel för parallellporten.)

Beskrivning av rutinerna.

WINITP	Initiera parallellporten. Skall anropas en gång av huvudprogrammet innan Watanabe-rutinerna kan användas.
WATOUT	Skickar ett ascii-tecken till parallellporten. Används endast internt av Watanabe-rutinerna.
RHOME } WHOME }	Flyttar pennan till hemma läge. Ritar inget.
RDRAW } WDRAW }	Ritar en rät linje från pennans nuvarande position till (x,y). Absoluta koordinater.
RMOVE } WMOVE }	Flyttar pennan till position (x,y) utan att rita. Absoluta koordinater.
RRDRAW } WRDRAW }	Samma som RDRAW, WDRAW men med relativa koordinater.
RRMOVE } WRMOVE }	Samma som RMOVE, WMOVE men med relativa koordinater.
RLNTYP } WLNTYP }	Väljer hur linjer skall ritas, kontinuerligt eller streckat på olika sätt.
RLNSCL } WLNSCL }	Väljer strecklängd för streckade linjer.
RAXIS } WAXIS }	Ritar en x- eller y-axel.
RPRINT } WPRINT }	Skriver en teckensträng.
RASCAL } WASCAL }	Väljer storlek på tecken.
RAROT } WAROT }	Väljer vilken riktning text skall skrivas.
RMARK } WMARK }	Ritar en markering i nuvarande position.
RPENCH	Byter penna.
RCIRC	Ritar en cirkel.
RRCIRC	Ritar en cirkel, relativa koordinater.
RCCENT	Specificerar ett cirkelcentrum.
RACIR	Ritar en cirkel med centrum enl. RCCENT specifikationen.

RAPROC Ritar segment och indikeringslinjer för cirklar med centrum enl. RRCENT specifikationen.

RHATCH Ritar streckade rektanglar.

RCURVE Ritar mjuk (spline) kurva mellan en serie punkter.

RRCURVE Samma som RCURVE men i relativa koordinater.

RRDGLC Roland RD-GL graphics command. Ger möjlighet att från DXY-mode anropa de HP-Graphics Language-kompatibla ritrutiner som Roland-plottern kan tolka. Det finns ungefär 50 kraftfulla kommandon att välja från. T.ex. ändra pennans hastighet, rita text med olika lutning i valfri riktning, ändra teckenuppsättningen, definiera egna tecken, skala om en ritning m.m. För närmare information se manualen .

```

1 C *****
2 C * Name: PLLIB.FOR, PLLIB.LIB *
3 C * Program library for Watanabe and Roland pen plotter. *
4 C * Anders Persson, 1988 *
5 C *****
6 C
7 C Assembler routines:
8 C *****
9 C Filename: WINITP.ASM
10 C FORTRAN CALL: CALL WINITP, no parameters
11 C Initialize paral. port. for Watanabe plotter
12 C must be called prior to any other plot routine if
13 C the Watanabe plotter is to be used
14 C
15 C TITLE WINITP
16 C ;Anders Persson, 1988
17 C
18 C LADATA SEGMENT 'DATA'
19 C DB 'WINITP '
20 C SP_SAVE DW 0
21 C DD 0
22 C LADATA ENDS
23 C PCODE SEGMENT 'CODE'
24 C ASSUME CS:PCODE,DS:LADATA
25 C DW SEG LADATA
26 C WINITP PROC FAR
27 C PUBLIC WINITP
28 C MOV AX,LADATA
29 C MOV DS,AX
30 C MOV SP_SAVE,SP
31 C STATPORTC = 770
32 C CTRLPORT = 771
33 C CTRLDATA = 10010001B
34 C MOV DX,CTRLPORT
35 C MOV AL,CTRLDATA
36 C OUT DX,AL ;INITIALIZE PORT
37 C MOV DX,STATPORTC
38 C MOV AL,00000000B
39 C OUT DX,AL ;ENSURE STROBE=0
40 C RET
41 C WINITP ENDP
42 C PCODE ENDS
43 C END
44 C
45 C *****
46 C Filename: WATOUT.ASM
47 C FORTRAN CALL: CALL WATOUT(D), INTEGER*2 D
48 C The routine sends an ascii character (D=ICHAR(CHARACTER)) to
49 C the parallell port, only used in Watanabe plot routines
50 C
51 C TITLE WATOUT
52 C ;Anders Persson, 1988
53 C PARBLK STRUC
54 C PAR DD ?
55 C PARBLK ENDS
56 C LADATA SEGMENT 'DATA'
57 C DB 'WATOUT '

```

```

58 C SP_SAVE DW 0
59 C DD 0
60 C LADATA ENDS
61 C PCODE SEGMENT 'CODE'
62 C ASSUME CS:PCODE,DS:LADATA
63 C DW SEG LADATA
64 C WATOUT PROC FAR
65 C PUBLIC WATOUT
66 C MOV AX,LADATA
67 C MOV DS,AX
68 C MOV SP_SAVE,SP
69 C UTPORTB = 769
70 C STATPORTC = 770
71 C MOV DX,STATPORTC ; CHECK
72 C BUSY: IN AL,DX ; PLOTTER
73 C AND AL,00001000B ; READY
74 C JNZ BUSY ; = BUSY=0
75 C ;
76 C ; MOV DX,UTPORTB ; PUT
77 C LDS SI,ES:PAR[BX] ; DATA
78 C MOV AX,[SI] ; ON
79 C AND AL,01111111B ; OUTPUT
80 C OUT DX,AL ; LINES
81 C ;
82 C ; MOV DX,STATPORTC ; SEND
83 C MOV AL,00010000B ;
84 C OUT DX,AL ; STROBE
85 C MOV AL,00000000B ;
86 C OUT DX,AL ; PULSE
87 C RET
88 C WATOUT ENDP
89 C PCODE ENDS
90 C END
91 C
92
93 SUBROUTINE RHOME
94 C Anders Persson, 1988
95
96 OPEN(9,FILE='PRN')
97 WRITE(9,100) 'H'
98 100 FORMAT(A1)
99 CLOSE(9)
100
101 RETURN
102 END
103
104 SUBROUTINE WHOME
105 C Anders Persson, 1988
106
107 IMPLICIT LOGICAL (A-Z)
108 INTEGER*2 D
109
110 D=ICHAR('H')
111 CALL WATOUT(D)
112 D=10
113 CALL WATOUT(D)
114
115 RETURN
116 END
117

```

```

118 SUBROUTINE RDRAW(X,Y)
119 C Anders Persson, 1988
120
121 IMPLICIT LOGICAL (A-Z)
122 REAL X,Y
123
124 IF (ABS(X).GT.16383.0.OR.ABS(Y).GT.16383.0) THEN
125 WRITE(*,*) 'Parameter out of range in RDRAW subroutine.'
126 STOP
127 ENDIF
128
129 OPEN(9,FILE='PRN')
130 WRITE(9,100) 'D',X,',',Y
131 100 FORMAT(A1,F11.4,A1,F11.4)
132 CLOSE(9)
133
134 RETURN
135 END
136
137 SUBROUTINE WDRAW(X,Y)
138 C Anders Persson, 1988
139
140 IMPLICIT LOGICAL (A-Z)
141 REAL X,Y
142 INTEGER*2 IX,IY,D,I
143 CHARACTER LINE*11
144
145 IF (X.GT.3600.0) THEN
146 IX=3600
147 ELSEIF(X.LT.0.0) THEN
148 IX=0
149 ELSE
150 IX=NINT(X)
151 ENDIF
152
153 IF (Y.GT.2600.0) THEN
154 IY=2600
155 ELSEIF(Y.LT.0.0) THEN
156 IY=0
157 ELSE
158 IY=NINT(Y)
159 ENDIF
160
161 WRITE(LINE,100) 'D',IX,',',IY,CHAR(10)
162 100 FORMAT(A,I4,A,I4,A)
163
164 DO 10 I=1,11
165 D=ICHAR(LINE(I:I))
166 CALL WATOUT(D)
167 10 CONTINUE
168
169 RETURN
170 END
171
172 SUBROUTINE RMOVE(X,Y)
173 C Anders Persson, 1988
174
175 IMPLICIT LOGICAL (A-Z)
176 REAL X,Y
177

```

```

178 IF (ABS(X).GT.16383.0.OR.ABS(Y).GT.16383.0) THEN
179 WRITE(*,*) 'Parameter out of range in RMOVE subroutine.'
180 STOP
181 ENDIF
182
183 OPEN(9,FILE='PRN')
184 WRITE(9,100) 'M',X,',',Y
185 100 FORMAT(A1,F11.4,A1,F11.4)
186 CLOSE(9)
187
188 RETURN
189 END
190
191 SUBROUTINE WMOVE(X,Y)
192 C Anders Persson, 1988
193
194 IMPLICIT LOGICAL (A-Z)
195 REAL X,Y
196 INTEGER*2 IX,IY,D,I
197 CHARACTER LINE*11
198
199 IF (X.GT.3600.0) THEN
200 IX=3600
201 ELSEIF(X.LT.0.0) THEN
202 IX=0
203 ELSE
204 IX=NINT(X)
205 ENDIF
206
207 IF (Y.GT.2600.0) THEN
208 IY=2600
209 ELSEIF(Y.LT.0.0) THEN
210 IY=0
211 ELSE
212 IY=NINT(Y)
213 ENDIF
214
215 WRITE(LINE,100) 'M',IX,',',IY,CHAR(10)
216 100 FORMAT(A,I4,A,I4,A)
217
218 DO 10 I=1,11
219 D=ICHAR(LINE(I:I))
220 CALL WATOUT(D)
221 10 CONTINUE
222
223 RETURN
224 END
225
226 SUBROUTINE RRDRAW(X,Y)
227 C Anders Persson, 1988
228
229 IMPLICIT LOGICAL (A-Z)
230 REAL X,Y
231
232 IF (ABS(X).GT.16383.0.OR.ABS(Y).GT.16383.0) THEN
233 WRITE(*,*) 'Parameter out of range in RRDRAW subroutine.'
234 STOP
235 ENDIF
236
237 OPEN(9,FILE='PRN')

```

```

238 WRITE(9,100) 'I',X,',',',Y
239 100 FORMAT(A1,F11.4,A1,F11.4)
240 CLOSE(9)
241
242 RETURN
243 END
244
245 SUBROUTINE WRDRAW(X,Y)
246 C Anders Persson, 1988
247
248 IMPLICIT LOGICAL (A-Z)
249 REAL X,Y
250 INTEGER*2 IX,IY,D,I
251 CHARACTER LINE*13
252
253 IF (ABS(X).GT.3600.0) THEN
254 IX=SIGN(3600.0,X)
255 ELSE
256 IX=NINT(X)
257 ENDIF
258
259 IF (ABS(Y).GT.2600.0) THEN
260 IY=SIGN(2600.0,Y)
261 ELSE
262 IY=NINT(Y)
263 ENDIF
264
265 WRITE(LINE,100) 'I',IX,',',',IY,CHAR(10)
266 100 FORMAT(A,15,A,15,A)
267
268 DO 10 I=1,13
269 D=ICHAR(LINE(I:I))
270 CALL WATOUT(D)
271 10 CONTINUE
272
273 RETURN
274 END
275
276 SUBROUTINE RRMOVE(X,Y)
277 C Anders Persson, 1988
278
279 IMPLICIT LOGICAL (A-Z)
280 REAL X,Y
281
282 IF (ABS(X).GT.16383.0.OR.ABS(Y).GT.16383.0) THEN
283 WRITE(*,*) 'Parameter out of range in RRMOVE subroutine.'
284 STOP
285 ENDIF
286
287 OPEN(9,FILE='PRN')
288 WRITE(9,100) 'R',X,',',',Y
289 100 FORMAT(A1,F11.4,A1,F11.4)
290 CLOSE(9)
291
292 RETURN
293 END
294
295 SUBROUTINE WRMOVE(X,Y)
296 C Anders Persson, 1988
297

```

```

298 IMPLICIT LOGICAL (A-Z)
299 REAL X,Y
300 INTEGER*2 IX,IY,D,I
301 CHARACTER LINE*13
302
303 IF (ABS(X).GT.3600.0) THEN
304 IX=SIGN(3600.0,X)
305 ELSE
306 IX=NINT(X)
307 ENDIF
308
309 IF (ABS(Y).GT.2600.0) THEN
310 IY=SIGN(2600.0,Y)
311 ELSE
312 IY=NINT(Y)
313 ENDIF
314
315 WRITE(LINE,100) 'R',IX,',',',IY,CHAR(10)
316 100 FORMAT(A,15,A,15,A)
317
318 DO 10 I=1,13
319 D=ICHAR(LINE(I:I))
320 CALL WATOUT(D)
321 10 CONTINUE
322
323 RETURN
324 END
325
326 SUBROUTINE RLNTYP(P)
327 C Anders Persson, 1988
328
329 IMPLICIT LOGICAL (A-Z)
330 INTEGER P,IP
331
332 IF (ABS(P).GE.6) THEN
333 IP=0
334 ELSE
335 IP=P
336 ENDIF
337
338 OPEN(9,FILE='PRN')
339 WRITE(9,100) 'L',IP
340 100 FORMAT(A1,I2)
341 CLOSE(9)
342
343 RETURN
344 END
345
346 SUBROUTINE WLNTYP(P)
347 C Anders Persson, 1988
348
349 IMPLICIT LOGICAL (A-Z)
350 INTEGER P
351 INTEGER*2 D,LF,L
352
353 LF=10
354 D=ICHAR('0')
355 IF (P.NE.0) THEN
356 D=ICHAR('1')
357 ENDIF

```

```

358
359 L=ICHAR('L')
360 CALL WATOUT(L)
361 CALL WATOUT(D)
362 CALL WATOUT(LF)
363
364 RETURN
365 END
366
367 SUBROUTINE RLNSCL(L)
368 C Anders Persson, 1988
369
370 IMPLICIT LOGICAL (A-Z)
371 INTEGER L
372
373 IF (L.LT.0.OR.L.GT.16383) THEN
374 WRITE(*,*) 'Parameter out of range in RLNSCL subroutine.'
375 STOP
376 ENDIF
377
378 OPEN(9,FILE='PRN')
379 WRITE(9,100) 'B',L
380 100 FORMAT(A1,I5)
381 CLOSE(9)
382
383 RETURN
384 END
385
386 SUBROUTINE WLNSCL(L)
387 C Anders Persson, 1988
388
389 IMPLICIT LOGICAL (A-Z)
390 INTEGER L
391 INTEGER*2 D,I
392 CHARACTER LINE*5
393
394 IF (L.GT.127) THEN
395 D=127
396 ELSEIF(L.LT.1) THEN
397 D=1
398 ELSE
399 D=L
400 ENDIF
401
402 WRITE(LINE,100) 'B',D,CHAR(10)
403 100 FORMAT(A,I3,A)
404
405 DO 10 I=1,5
406 D=ICHAR(LINE(I:I))
407 CALL WATOUT(D)
408 10 CONTINUE
409
410 RETURN
411 END
412
413 SUBROUTINE RAXIS(P,Q,R)
414 C Anders Persson, 1988
415
416 IMPLICIT LOGICAL (A-Z)
417 INTEGER P,R

```

```

418 REAL Q
419
420 IF (P.LT.0.OR.P.GT.1.OR.ABS(Q).GT.16383.0.OR.
421 f R.LT.1.OR.R.GT.16383) THEN
422 WRITE(*,*) 'Parameter out of range in RAXIS subroutine.'
423 STOP
424 ENDIF
425
426 OPEN(9,FILE='PRN')
427 WRITE(9,100) 'X',P,',',Q,',',R
428 100 FORMAT(A1,I1,A1,F11.4,A1,I5)
429 CLOSE(9)
430
431 RETURN
432 END
433
434 SUBROUTINE WAXIS(P,Q,R)
435 C Anders Persson, 1988
436
437 IMPLICIT LOGICAL (A-Z)
438 INTEGER P,R
439 REAL Q
440 INTEGER*2 IP,IQ,IR,D,I
441 CHARACTER LINE*13
442
443 IP=1
444 IF (P.EQ.0) THEN
445 IP=0
446 ENDIF
447
448 IF (R.LT.1) THEN
449 IR=1
450 ELSEIF (R.GT.1000) THEN
451 IR=1000
452 ELSE
453 IR=R
454 ENDIF
455
456 IF (IP.EQ.1) THEN
457 IF (Q*R.GT.3600.0) THEN
458 IQ=NINT(3600.0/R)
459 ELSEIF(Q*R.LT.1.0) THEN
460 IQ=1
461 ELSE
462 IQ=NINT(Q)
463 ENDIF
464 ELSE
465 IF (Q*R.GT.2600.0) THEN
466 IQ=NINT(2600.0/R)
467 ELSEIF(Q*R.LT.1.0) THEN
468 IQ=1
469 ELSE
470 IQ=NINT(Q)
471 ENDIF
472 ENDIF
473
474 WRITE(LINE,100) 'X',IP,',',IQ,',',IR,CHAR(10)
475 100 FORMAT(A,I1,A,I4,A,I4,A)
476
477 DO 10 I=1,13

```

```

478     D=ICHAR(LINE(1:1))
479     CALL WATOUT(D)
480 10  CONTINUE
481
482     RETURN
483     END
484
485     SUBROUTINE RPRINT(MSG,LEN)
486 C    Anders Persson, 1988
487
488     IMPLICIT      LOGICAL (A-Z)
489     INTEGER       LEN,I
490     CHARACTER*(*) MSG
491
492     OPEN(9,FILE='PRN')
493     WRITE(9,100) 'P',(MSG(1:1), I=1,LEN)
494 100  FORMAT(10000(A1))
495     CLOSE(9)
496
497     RETURN
498     END
499
500     SUBROUTINE WPRINT(MSG,LEN)
501 C    Anders Persson, 1988
502
503     IMPLICIT      LOGICAL (A-Z)
504     INTEGER       LEN
505     CHARACTER*(*) MSG
506     INTEGER*2     D,I
507
508     D=ICHAR('P')
509     CALL WATOUT(D)
510
511     DO 10 I=1,LEN
512         D=ICHAR(MSG(1:I))
513         CALL WATOUT(D)
514 10  CONTINUE
515
516     D=10
517     CALL WATOUT(D)
518
519     RETURN
520     END
521
522     SUBROUTINE RASCAL(N)
523 C    Anders Persson, 1988
524
525     IMPLICIT      LOGICAL (A-Z)
526     INTEGER       N,IN
527
528     IF (N.LT.0.OR.N.GT.127) THEN
529         IN=3
530     ELSE
531         IN=N
532     ENDIF
533
534     OPEN(9,FILE='PRN')
535     WRITE(9,100) 'S',IN
536 100  FORMAT(A1,I3)
537     CLOSE(9)

```

```

538     RETURN
539     END
540
541     SUBROUTINE WASCAL(N)
542 C    Anders Persson, 1988
543
544     IMPLICIT      LOGICAL (A-Z)
545     INTEGER       N
546     INTEGER*2     D,I
547     CHARACTER     LINE*4
548
549     IF (N.LT.0) THEN
550         D=0
551     ELSEIF(N.GT.15) THEN
552         D=15
553     ELSE
554         D=N
555     ENDIF
556
557     WRITE(LINE,100) 'S',D,CHAR(10)
558 100  FORMAT(A,I2,A)
559
560     DO 10 I=1,4
561         D=ICHAR(LINE(1:I))
562         CALL WATOUT(D)
563 10  CONTINUE
564
565     RETURN
566     END
567
568     SUBROUTINE RAROT(Q)
569 C    Anders Persson, 1988
570
571     IMPLICIT      LOGICAL (A-Z)
572     INTEGER       Q,IQ
573
574     IF (Q.LT.0.OR.Q.GT.3) THEN
575         IQ=0
576     ELSE
577         IQ=Q
578     ENDIF
579
580     OPEN(9,FILE='PRN')
581     WRITE(9,100) 'Q',IQ
582 100  FORMAT(A1,I1)
583     CLOSE(9)
584
585     RETURN
586     END
587
588     SUBROUTINE WAROT(Q)
589 C    Anders Persson, 1988
590
591     IMPLICIT      LOGICAL (A-Z)
592     INTEGER       Q
593     INTEGER*2     D,I
594     CHARACTER     LINE*3
595
596     IF (Q.LT.0) THEN
597

```



```

598     D=0
599     ELSEIF(Q.GT.3) THEN
600     D=3
601     ELSE
602     D=Q
603     ENDF
604
605     WRITE(LINE,100) 'Q',D,CHAR(10)
606 100  FORMAT(A,I1,A)
607
608     DO 10 I=1,3
609     D=ICHAR(LINE(I:I))
610     CALL WATOUT(D)
611 10  CONTINUE
612
613     RETURN
614     END
615
616     SUBROUTINE RMARK(N)
617 C    Anders Persson, 1988
618
619     IMPLICIT LOGICAL (A-Z)
620     INTEGER N,IN
621
622     IF (N.LT.0.OR.N.GT.15) THEN
623     IN=4
624     ELSE
625     IN=N
626     ENDF
627
628     OPEN(9,FILE='PRN')
629     WRITE(9,100) 'N',IN
630 100  FORMAT(A1,I2)
631     CLOSE(9)
632
633     RETURN
634     END
635
636     SUBROUTINE WMARK(N)
637 C    Anders Persson, 1988
638
639     IMPLICIT LOGICAL (A-Z)
640     INTEGER N
641     INTEGER*2 D,I
642     CHARACTER LINE*3
643
644     IF (N.LT.0) THEN
645     D=0
646     ELSEIF(N.GT.6) THEN
647     D=6
648     ELSE
649     D=N
650     ENDF
651
652     WRITE(LINE,100) 'N',D,CHAR(10)
653 100  FORMAT(A,I1,A)
654
655     DO 10 I=1,3
656     D=ICHAR(LINE(I:I))
657     CALL WATOUT(D)

```

```

658 10  CONTINUE
659
660     RETURN
661     END
662
663     SUBROUTINE RPENCH(J)
664 C    Anders Persson, 1988
665
666     IMPLICIT LOGICAL (A-Z)
667     INTEGER J,IJ
668
669     IF (J.LT.0.OR.J.GT.8) THEN
670     IJ=1
671     ELSE
672     IJ=J
673     ENDF
674
675     OPEN(9,FILE='PRN')
676     WRITE(9,100) 'J',IJ
677 100  FORMAT(A1,I1)
678     CLOSE(9)
679
680     RETURN
681     END
682
683     SUBROUTINE RCIRC(X,Y,R,T1,T2,DT)
684 C    Anders Persson, 1988
685
686     IMPLICIT LOGICAL (A-Z)
687     REAL X,Y,R,T1,T2,DT
688     CHARACTER K*1
689
690     K=', '
691
692     IF (ABS(X) .GT. 16383.0 .OR. ABS(Y) .GT. 16383.0 .OR.
693     f ABS(R) .GT. 16383.0 .OR. ABS(T1) .GT. 32767.0 .OR.
694     f ABS(T2) .GT. 32767.0 .OR. ABS(DT) .LT. 1.0 .OR.
695     f ABS(DT) .GT. 179.9999 ) THEN
696
697     WRITE(*,*) 'Parameter out of range in RCIRC subroutine.'
698     STOP
699     ENDF
700
701     OPEN(9,FILE='PRN')
702     IF (DT.GT.0.0) THEN
703     WRITE(9,100) 'C',X,K,Y,K,R,K,T1,K,T2,K,DT
704     ELSE
705     WRITE(9,100) 'C',X,K,Y,K,R,K,T1,K,T2
706     ENDF
707
708 100  FORMAT(5(A1,F11.4))
709     CLOSE(9)
710
711     RETURN
712     END
713
714     SUBROUTINE RRCIRC(R,T1,T2,DT)
715 C    Anders Persson, 1988
716
717     IMPLICIT LOGICAL (A-Z)

```

```

718 REAL R,T1,T2,DT
719 CHARACTER K*1
720
721 K=', '
722
723 IF (ABS(R) .GT. 16383.0 .OR. ABS(T1) .GT. 32767.0 .OR.
724 f ABS(T2) .GT. 32767.0 .OR. ABS(DT) .LT. 1.0 .OR.
725 f ABS(DT) .GT. 179.9999 ) THEN
726
727 WRITE(*,*) 'Parameter out of range in RRCIRC subroutine.'
728 STOP
729 ENDIF
730
731 OPEN(9,FILE='PRN')
732 IF (DT.GT.0.0) THEN
733 WRITE(9,100) 'E',R,K,T1,K,T2,K,DT
734 ELSE
735 WRITE(9,100) 'E',R,K,T1,K,T2
736 ENDIF
737 100 FORMAT(4(A1,F11.4))
738 CLOSE(9)
739
740 RETURN
741 END
742
743 SUBROUTINE RCCENT(X,Y)
744 C Anders Persson, 1988
745
746 IMPLICIT LOGICAL (A-Z)
747 REAL X,Y
748
749 IF (ABS(X).GT.16383.0.OR.ABS(Y).GT.16383.0) THEN
750 WRITE(*,*) 'Parameter out of range in RCCENT subroutine.'
751 STOP
752 ENDIF
753
754 OPEN(9,FILE='PRN')
755 WRITE(9,100) 'A',X,',',Y
756 100 FORMAT(A1,F11.4,A1,F11.4)
757 CLOSE(9)
758
759 RETURN
760 END
761
762 SUBROUTINE RACIR(R,T1,T2,DT)
763 C Anders Persson, 1988
764
765 IMPLICIT LOGICAL (A-Z)
766 REAL R,T1,T2,DT
767 CHARACTER K*1
768
769 K=', '
770
771 IF (ABS(R) .GT. 16383.0 .OR. ABS(T1) .GT. 32767.0 .OR.
772 f ABS(T2) .GT. 32767.0 .OR. ABS(DT) .LT. 1.0 .OR.
773 f ABS(DT) .GT. 179.9999 ) THEN
774
775 WRITE(*,*) 'Parameter out of range in RACIR subroutine.'
776 STOP
777 ENDIF

```

```

778
779 OPEN(9,FILE='PRN')
780 IF (DT.GT.0.0) THEN
781 WRITE(9,100) 'G',R,K,T1,K,T2,K,DT
782 ELSE
783 WRITE(9,100) 'G',R,K,T1,K,T2
784 ENDIF
785 100 FORMAT(4(A1,F11.4))
786 CLOSE(9)
787
788 RETURN
789 END
790
791 SUBROUTINE RAPROC(N,L1,L2)
792 C Anders Persson, 1988
793
794 IMPLICIT LOGICAL (A-Z)
795 REAL N,L1,L2
796 CHARACTER K*1
797
798 K=', '
799
800 IF (ABS(N) .GT. 9101.0 .OR. ABS(L1) .GT. 16383.0 .OR.
801 f ABS(L2) .GT. 16383.0 ) THEN
802
803 WRITE(*,*) 'Parameter out of range in RAPROC subroutine.'
804 STOP
805 ENDIF
806
807 OPEN(9,FILE='PRN')
808 WRITE(9,100) 'K',N,K,L1,K,L2
809 100 FORMAT(A1,F11.4,A1,F11.4,A1,F11.4)
810 CLOSE(9)
811
812 RETURN
813 END
814
815 SUBROUTINE RHATCH(M,X,Y,D,T)
816 C Anders Persson, 1988
817
818 IMPLICIT LOGICAL (A-Z)
819 INTEGER M,T,IM,IT
820 REAL X,Y,D
821
822 IF (ABS(X).GT.16383.0.OR.ABS(Y).GT.16383.0.OR.
823 f ABS(D).GT.16383.0 ) THEN
824
825 WRITE(*,*) 'Parameter out of range in RHATCH subroutine.'
826 STOP
827 ENDIF
828
829 IF (M.LT.0.OR.M.GT.3) THEN
830 IM=2
831 ELSE
832 IM=M
833 ENDIF
834
835 IF (T.LT.1.OR.T.GT.4) THEN
836 IT=2
837 ELSE

```

```

838      IT=T
839      ENDIF
840
841      OPEN(9, FILE='PRN')
842      WRITE(9,100) 'T',IM,'',X,'',Y,'',D,'',T
843 100  FORMAT(A1,I1,A1,F11.4,A1,F11.4,A1,F11.4,A1,I1)
844      CLOSE(9)
845
846      RETURN
847      END
848
849      SUBROUTINE RCURVE(M,X,Y,N)
850 C      Anders Persson, 1988
851
852      IMPLICIT LOGICAL (A-Z)
853      INTEGER M,IM,N,I
854      REAL X(1:*),Y(1:*)
855
856      IF (M.LT.0.OR.M.GT.3) THEN
857          IM=0
858      ELSE
859          IM=M
860      ENDIF
861
862      IF ((M.LE.1.AND.N.LE.2).OR.(M.GE.2.AND.N.LE.1)) THEN
863          WRITE(*,*) 'Parameter out of range in RCURVE subroutine.'
864          STOP
865      ENDIF
866
867      DO 10 I=1,N
868          IF (ABS(X(I)).GT.16383.0.OR.ABS(Y(I)).GT.16383.0) THEN
869              WRITE(*,*) 'Parameter out of range in RCURVE subroutine.'
870              STOP
871          ENDIF
872 10      CONTINUE
873
874      OPEN(9, FILE='PRN')
875      WRITE(9,100) 'Y',IM,'',(' ',X(I),' ',Y(I), I=1,N)
876 100  FORMAT(A1,I1,A1,10000(A1,F11.4,A1,F11.4))
877      CLOSE(9)
878
879      RETURN
880      END
881
882      SUBROUTINE RRCURV(M,X,Y,N)
883 C      Anders Persson, 1988
884
885      IMPLICIT LOGICAL (A-Z)
886      INTEGER M,IM,N,I
887      REAL X(1:*),Y(1:*)
888
889      IF (M.LT.0.OR.M.GT.1) THEN
890          IM=0
891      ELSE
892          IM=M
893      ENDIF
894
895      IF (N.LE.2) THEN
896          WRITE(*,*) 'Parameter out of range in RRCURVE subroutine.'
897          STOP

```

```

898      ENDIF
899
900      DO 10 I=1,N
901          IF (ABS(X(I)).GT.16383.0.OR.ABS(Y(I)).GT.16383.0) THEN
902              WRITE(*,*) 'Parameter out of range in RRCURVE subroutine.'
903              STOP
904          ENDIF
905 10      CONTINUE
906
907      OPEN(9, FILE='PRN')
908      WRITE(9,100) ' ',IM,'',(' ',X(I),' ',Y(I), I=1,N)
909 100  FORMAT(A1,I1,A1,10000(A1,F11.4,A1,F11.4))
910      CLOSE(9)
911
912      RETURN
913      END
914
915      SUBROUTINE RRDGLC(C,LEN)
916 C      Anders Persson, 1988
917
918      IMPLICIT LOGICAL (A-Z)
919      INTEGER LEN,I
920      CHARACTER*(*) C
921
922      OPEN(9, FILE='PRN')
923      WRITE(9,100) '^',(C(I:I), I=1,LEN)
924 100  FORMAT(10000(A1))
925      CLOSE(9)
926
927      RETURN
928      END

```

HPLIB, HPFLIB

Biblioteken HPLIB och HPFLIB innehåller diverse rutiner för att använda en HP-LaserJet Series II skrivare som en pen-plotter eller grafiskskärm. Den använda upplösningen är 100 punkter per tum vilket för A4-bilder innebär en rasterstorlek på $(0.1119) \times (0.719)$. Vilket bibliotek som används vid länkning beror på om programmet skall köras på den dator som är direktansluten till laserskrivaren. HPLIB skickar rasterbilden direkt till skrivaren medan HPFLIB genererar en fil där bilden lagras, denna fil kan sedan bäras över på en diskett till laserskrivaren och dumpas till denna med kopieringskommandot: COPY/B <filnamn> PRN:. De FORTRAN rutiner som skriver data till en fil är tyvärr långsamma, det tar betydlig längre tid att spara en bild på diskett än att skicka den direkt till laserskrivaren. De båda biblioteken adderar ungefär 100kB till program som använder dem eftersom rasterbilden är ungefär av den storleken om upplösningen 100 punkter per tum används. För närmare information om grafikmöjligheter på laserskrivaren se: Hewlett-Packard, LaserJet Printer Family, Technical Reference Manual.

HPLIB

Beskrivning av rutinerna.

- HPSTART Rutinen måste anropas en gång innan en ny bild skall ritas. Den initialiserar skrivarporten och laserskrivaren till den status som anges av kommentarraderna i rutinen. Om ORIENTATION='L' eller 'l' vid anrop sätts landskapsformat annars används porträttformatet. Eventuella data från föregående bild suddas och cursorn placeras i position (0,0), (nedre vänstra hörnet).
- HPINIT Rutinen initialiserar skrivarporten. Anropas från HPSTART och skall inte användas annat än genom denna.
- HPOUT Skickar ascii-tecknet i D till skrivarporten. Används endast internt i plot-rutinerna.
- HPRLINE Skickar en rasterlinje till laserskrivaren. Används av HPDUMP rutinen, skall inte anropas på annat sätt än genom denna.

HPDUMP Rutinen skickar över rasterbilden till laserskrivaren. Anropas av rutinen HPEJECT och bör endast användas via denna.

HPEJECT Rutinen anropar HPDUMP samt skickar ett 'form feed' till laserskrivaren. Anropas när bilden är färdigritad med text och grafik för att få ut papperet.

HPFLIB

Beskrivning av rutinerna.

HPSTART Samma som HPSTART i HPLIB men den läser in filnamnet på den fil där bilden skall lagras.

HPOUT Lagrar ascii-tecknet D på filen.

HPDUMP Lagrar rasterbilden på filen.

HPEJECT Anropar HPDUMP samt lagrar ett 'form feed' på filen. Anropas när bilden är färdigritad.

LOWBYTE } Dessa rutiner tar fram de 8 minst resp. mest signifikanta
HIGHBYTE } bitarna i D. Används av HPDUMP rutinen för att sära på rasterbildelementen vilka lagras i 16-bitars integer variabler.

HPLIB och HPFLIB, gemensamma rutiner.

HPPRINT Skriver texten i teckensträngen på bilden. Texten börjar i position (x,y), absoluta koordinater.

HPASCII Skickar ascii-strängen lagrad i MSG till laserskrivaren. Används internt av HP(F)LIB rutiner.

HPLEVEL Väljer hur pixel på bilden skall ritas i fortsättningen, svart, vit eller genom att ändra nuvarande färg på bildelementet till den andra (svart⇒vit, vit⇒svart). Observera att text inte påverkas av denna rutin, text ritas alltid svart.

HPCLEAR Sudda rasterbilden i datorn. Eventuell text redan skickad till skrivaren påverkas inte.

HPMOVE Flytta cursorn till position (x,y) utan att rita något. Absoluta koordinater.

HPCMOVE Flytta laserskrivarens textcursor till position (x,y) på papperet. Används av textritningsrutinerna och bör inte anropas direkt från FORTRAN program.

HPDRAW Ritar en rät linje från nuvarande position till (x,y). Absoluta koordinater.

HPPLOT Ritar en pixel i position (x,y). Absoluta koordinater.

HPSET Returnerar värdet `.TRUE.` om pixel `(x,y)` är svart, annars
 `.FALSE.`

HPRMOVE Flytta cursorn relativt nuvarande position.

HPRPLOT Ritar en pixel i en punkt relativt nuvarande position.

HPRDRAW Ritar en rät linje till `(x,y)` relativt nuvarande position.

HPRPRINT Skriver texten i teckensträngen på bilden. Texten börjar i
 position `(x,y)`, relativa koordinater.

HPMARK Ritar ett märke, (ett litet `+-`tecken), i position `(x,y)`.

HPBOX Suddar rektangeln med nedre vänstra hörnet i `(X,Y)` och med
 sidlängder `XL,YL`. Ritar därefter en ram kring området. (om
 ingen ram önskas kan ramen ritas 'vit' genom att anropa
 `HPLEVEL` rutinen)

HPTBOX Skriver texten i strängen `MSG` och ramar in den med hjälp av
 `HPBOX` rutinen.

HPCIRCL Ritar en cirkel med radie `R` och centrum i position `(x,y)`.

```

1 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
2 C                               C
3 C   Library: HPLIB.LIB         C
4 C                               C
5 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
6
7   SUBROUTINE HPSTART(ORIENTATION)
8 C   Anders Persson 1988
9
10  IMPLICIT LOGICAL (A-Z)
11  CHARACTER ORIENTATION*1
12  INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
13  LOGICAL LANDSCAPE
14  INTEGER*2 INITDATA(1:52),I
15  COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
16
17  DATA INITDATA/
18 f      27,38,108,48,79,
19 f      27,40, 48,83,
20 f      27,40,115,48,80,
21 f      27,40,115,49,54,46,54,54,72,
22 f      27,40,115,55,46,53,86,
23 f      27,40,115,48,83,
24 f      27,40,115,48,66,
25 f      27,40,115,48,84,
26 f      27,42,116,49,48,48,82/
27
28 C   Data corresponds to:
29 C   Orientation           = Portrait or landscape
30 C   Symbol set            = Swedish,
31 C   Spacing                = Fixed,
32 C   Pitch                  = 16.66 CPI,
33 C   Point size             = 8.5,
34 C   Style                   = Upright,
35 C   Stroke weight          = Medium,
36 C   Typeface               = Line printer,
37 C   Raster Graphics Resolution = 100 DPI.
38
39   COLOUR=1
40
41   LANDSCAPE=ORIENTATION.EQ.'L'.OR.ORIENTATION.EQ.'L'
42   IF (LANDSCAPE) THEN
43     INITDATA(4)=49
44   ELSE
45     INITDATA(4)=48
46   ENDIF
47
48   CALL HPINIT
49   DO 10 I=1,52
50     CALL HPOUT(INITDATA(I))
51 10 CONTINUE
52
53   CALL HPCLEAR
54   CALL HPMOVE(0.0,0.0)
55   RETURN
56   END
57

```

-- 172 --

```

58 C                               TITLE           HPINIT
59 C   ; Anders Persson, 1988
60 C   ; FORTRAN      SUBROUTINE HPINIT
61 C   ;
62 C   LADATA          SEGMENT          'DATA'
63 C                   DB              'HPINIT '
64 C   SP_SAVE        DW              0
65 C                   DD              0
66 C   LADATA          ENDS
67 C   PCODE           SEGMENT          'CODE'
68 C                   ASSUME          CS:PCODE,DS:LADATA
69 C                   DW              SEG LADATA
70 C   HPINIT          PROC            FAR
71 C                   PUBLIC          HPINIT
72 C                   MOV              AX,LADATA
73 C                   MOV              DS,AX
74 C                   MOV              SP_SAVE,SP
75 C   CTRLPORT        =              03BEH
76 C                   MOV              DX,CTRLPORT      ; INITIALIZE
77 C                   MOV              AL,11101100B     ; PRINTER
78 C                   OUT              DX,AL           ; PORT
79 C                   RET
80 C   HPINIT          ENDP
81 C   PCODE           ENDS
82 C                   END
83 C
84 C                               TITLE           HPOUT
85 C   ; Anders Persson, 1988
86 C   ; FORTRAN:     SUBROUTINE HPOUT(D)
87 C   ;               INTEGER*2 D
88 C   ;
89 C   PARBLK          STRUC
90 C                   PAR              DD              ?
91 C   PARBLK          ENDS
92 C   LADATA          SEGMENT          'DATA'
93 C                   DB              'HPOUT '
94 C   SP_SAVE        DW              0
95 C                   DD              0
96 C   LADATA          ENDS
97 C   PCODE           SEGMENT          'CODE'
98 C                   ASSUME          CS:PCODE,DS:LADATA
99 C                   DW              SEG LADATA
100 C   HPOUT          PROC            FAR
101 C                   PUBLIC          HPOUT
102 C                   MOV              AX,LADATA
103 C                   MOV              DS,AX
104 C                   MOV              SP_SAVE,SP
105 C   ;
106 C   STATPORT        =              03BDH
107 C   DATAPORT        =              03BCH
108 C   CTRLPORT        =              03BEH
109 C   ;
110 C                   MOV              DX,STATPORT      ; CHECK
111 C   BUSY:           IN              AL,DX             ; PRINTER
112 C                   AND              AL,10000000B     ; READY
113 C                   JZ                BUSY           ; = BUSY=0
114 C   ;
115 C                   MOV              DX,DATAPORT      ; PUT
116 C                   LDS              SI,ES:PAR[BX]    ; DATA
117 C                   MOV              AX,[SI]         ; ON OUTPUT

```

```

118 C      OUT          DX,AL          ; LINES
119 C      ;
120 C      MOV          DX,CTRLPORT    ; SEND
121 C      MOV          AL,11101101B   ;
122 C      OUT          DX,AL          ; STROBE
123 C      MOV          AL,11101100B   ;
124 C      OUT          DX,AL          ; PULSE
125 C      RET
126 C      HPOUT       ENDP
127 C      PCODE       ENDS
128 C      END
129 C
130 C      TITLE        HPRLINE
131 C      ; Anders Persson, 1988
132 C      ; FORTRAN:  SUBROUTINE HPRLINE(LEN,DMPLINE)
133 C      ;           INTEGER*2 LEN,DMPLINE(1:LEN)
134 C      ;
135 C      PARBLK       STRUC
136 C      LEN          DD          ?
137 C      ARRAY        DD          ?
138 C      PARBLK       ENDS
139 C      LADATA       SEGMENT 'DATA'
140 C      DB           'HPRLINE '
141 C      SP_SAVE      DW          0
142 C      DD           0
143 C      LADATA       ENDS
144 C      PCODE       SEGMENT 'CODE'
145 C      ASSUME       CS:PCODE,DS:LADATA
146 C      DW           SEG LADATA
147 C      HPRLINE      PROC
148 C      PUBLIC      HPRLINE
149 C      MOV          AX,LADATA
150 C      MOV          DS,AX
151 C      MOV          SP_SAVE,SP
152 C      ;
153 C      CTRLPORT    =          03BEH
154 C      DATAPORT    =          03BCH
155 C      STATPORT    =          03BDH
156 C      ;
157 C      LDS         SI,ES:LEN[BX]
158 C      MOV         CX,[SI]
159 C      MOV         AX,CX
160 C      ADD         CX,AX
161 C      LDS         SI,ES:ARRAY[BX]
162 C      MOV         BX,0
163 C      ;
164 C      AGAIN:      MOV          DX,STATPORT    ; CHECK
165 C      BUSY1:      IN           AL,DX          ; IF PRINTER
166 C      AND         AL,10000000B   ; IS READY
167 C      JZ          BUSY1          ; FOR DATA
168 C      ;
169 C      MOV         DX,DATAPORT    ; SEND
170 C      MOV         AX,[SI][BX]    ; FIRST
171 C      MOV         AL,AH          ; DATA
172 C      OUT         DX,AL          ; BYTE
173 C      ;
174 C      MOV         DX,CTRLPORT    ; SEND
175 C      MOV         AL,11101101B   ;
176 C      OUT         DX,AL          ; STROBE
177 C      MOV         AL,11101100B   ;
178 C      OUT         DX,AL          ; PULSE
179 C      ;
180 C      MOV         DX,STATPORT    ; CHECK
181 C      BUSY2:      IN           AL,DX          ; IF PRINTER
182 C      AND         AL,10000000B   ; IS READY
183 C      JZ          BUSY2          ; FOR DATA
184 C      ;
185 C      MOV         AX,[SI][BX]    ; SEND
186 C      MOV         DX,DATAPORT    ; SECOND
187 C      OUT         DX,AL          ; BYTE
188 C      ;
189 C      MOV         DX,CTRLPORT    ; SEND
190 C      MOV         AL,11101101B   ;
191 C      OUT         DX,AL          ; STROBE
192 C      MOV         AL,11101100B   ;
193 C      OUT         DX,AL          ; PULSE
194 C      ;
195 C      ADD         BX,2           ; IF MORE
196 C      CMP         BX,CX          ; DATA THEN
197 C      JNZ        AGAIN          ; GOTO AGAIN
198 C      ;
199 C      RET
200 C      HPRLINE     ENDP
201 C      PCODE       ENDS
202 C      END
203
204      SUBROUTINE HPDUMP
205 C      Anders Persson, 1988
206
207      IMPLICIT    LOGICAL (A-Z)
208      INTEGER*2  XPOS,YPOS,COLOUR,RASTER(0:44,0:1119),I,J,LEN
209      INTEGER*2  DMPLINE(1:48),ESC
210      LOGICAL    LANDSCAPE
211      COMMON     /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
212
213      ESC=27
214      LEN=48
215      DMPLINE(1)=256*27+42
216      DMPLINE(2)=256*98+57
217      DMPLINE(3)=256*48+87
218 C      DMPLINE(1..3)=Esc * b 9 0 W = Raster line header
219
220 C      Set cursor position
221      IF (LANDSCAPE) THEN
222          CALL HPCMOVE(1119.0,719.0)
223      ELSE
224          CALL HPCMOVE(0.0,1119.0)
225      ENDIF
226
227 C      Initialize raster graphics transfer
228      CALL HPOUT(ESC)
229      CALL HPASCII(4,'*r1A')
230
231      DO 20 I=0,1119
232 C      Transfer raster graphics line
233          DO 10 J=0,44
234              DMPLINE(J+4)=RASTER(J,I)
235          CONTINUE
236          CALL HPRLINE(LEN,DMPLINE)
237      CONTINUE

```



```

238
239 C   End raster graphics
240     CALL HPOUT(ESC)
241     CALL HPASCII(3,'*rB')
242     RETURN
243     END
244
245     SUBROUTINE HPEJECT
246 C   Anders Persson, 1988
247
248     IMPLICIT LOGICAL (A-Z)
249     INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119),FF
250     LOGICAL LANDSCAPE
251     COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
252
253     CALL HPDUMP
254     FF=12
255     CALL HPOUT(FF)
256     RETURN
257     END
258
259     CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
260 C   Remaining routines common to HPLIB and HPFLIB C
261 C   Anders Persson 1988 C
262 C   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
263 C   CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
264
265     SUBROUTINE HPPRINT(X,Y,LEN,MSG)
266 C   Anders Persson 1988
267
268     IMPLICIT LOGICAL (A-Z)
269     REAL X,Y
270     INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119),LEN,I,D
271     LOGICAL LANDSCAPE
272     CHARACTER MSG*(*)
273     COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
274
275     CALL HPCMOVE(X,Y)
276     DO 10 I=1,LEN
277         D=ICHAR(MSG(I:1))
278         CALL HPOUT(D)
279 10 CONTINUE
280     RETURN
281     END
282
283     SUBROUTINE HPASCII(LEN,MSG)
284 C   Anders Persson 1988
285
286     IMPLICIT LOGICAL (A-Z)
287     INTEGER LEN
288     INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119),I,D
289     LOGICAL LANDSCAPE
290     CHARACTER MSG*(*)
291     COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
292
293     DO 10 I=1,LEN
294         D=ICHAR(MSG(I:1))
295         CALL HPOUT(D)
296 10 CONTINUE
297     RETURN

```

```

298     END
299
300     SUBROUTINE HPLEVEL(I)
301 C   Anders Persson 1988
302
303     IMPLICIT LOGICAL (A-Z)
304     INTEGER I
305     INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
306     LOGICAL LANDSCAPE
307     COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
308
309 C   I=0 => White, I=1 => Black, I=2 => Change
310
311     COLOUR=1
312     IF (I.EQ.0) THEN
313         COLOUR=0
314     ELSEIF (I.EQ.2) THEN
315         COLOUR=2
316     ENDIF
317     RETURN
318     END
319
320     SUBROUTINE HPCLEAR
321 C   Anders Persson 1988
322
323     IMPLICIT LOGICAL (A-Z)
324     INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119),I,J
325     LOGICAL LANDSCAPE
326     COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
327
328     XPOS=0
329     YPOS=0
330     DO 10 I=0,1119
331         DO 10 J=0,44
332             RASTER(J,I)=0
333 10 CONTINUE
334     RETURN
335     END
336
337     SUBROUTINE HPMOVE(X,Y)
338 C   Anders Persson 1988
339 C
340
341     IMPLICIT LOGICAL (A-Z)
342     REAL X,Y
343     INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
344     LOGICAL LANDSCAPE
345     COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
346
347     IF (LANDSCAPE) THEN
348         IF (X.LT.0.0) THEN
349             XPOS=0
350         ELSEIF (X.GT.1119.0) THEN
351             XPOS=1119
352         ELSE
353             XPOS=NINT(X)
354         ENDIF
355         IF (Y.LT.0.0) THEN
356             YPOS=0
357         ELSEIF (Y.GT.719.0) THEN

```

```

358     YPOS=719
359     ELSE
360     YPOS=NINT(Y)
361     ENDIF
362     ELSE
363     IF (X.LT.0.0) THEN
364     XPOS=0
365     ELSEIF (X.GT.719.0) THEN
366     XPOS=719
367     ELSE
368     XPOS=NINT(X)
369     ENDIF
370     IF (Y.LT.0.0) THEN
371     YPOS=0
372     ELSEIF (Y.GT.1119.0) THEN
373     YPOS=1119
374     ELSE
375     YPOS=NINT(Y)
376     ENDIF
377     ENDIF
378
379     RETURN
380     END
381
382     SUBROUTINE HPCMOVE(X,Y)
383     C Anders Persson 1988
384
385     IMPLICIT LOGICAL (A-Z)
386     REAL X,Y
387     INTEGER*2 D,ESC,I,XCURS,YCURS
388     CHARACTER LINE*8
389     INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
390     LOGICAL LANDSCAPE
391     COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
392
393     ESC=27
394     IF (LANDSCAPE) THEN
395     IF (X.LT.0.0) THEN
396     XPOS=0
397     ELSEIF (X.GT.1119.0) THEN
398     XPOS=1119
399     ELSE
400     XPOS=NINT(X)
401     ENDIF
402     IF (Y.LT.0.0) THEN
403     YPOS=0
404     ELSEIF (Y.GT.719.0) THEN
405     YPOS=719
406     ELSE
407     YPOS=NINT(Y)
408     ENDIF
409     ELSE
410     IF (X.LT.0.0) THEN
411     XPOS=0
412     ELSEIF (X.GT.719.0) THEN
413     XPOS=719
414     ELSE
415     XPOS=NINT(X)
416     ENDIF
417     IF (Y.LT.0.0) THEN

```

```

418     YPOS=0
419     ELSEIF (Y.GT.1119.0) THEN
420     YPOS=1119
421     ELSE
422     YPOS=NINT(Y)
423     ENDIF
424     ENDIF
425
426     C Place cursor at (x,y)=(0,0)
427     CALL HPOUT(ESC)
428     CALL HPASCII(4,'*p0X')
429     CALL HPOUT(ESC)
430     CALL HPASCII(4,'*p0Y')
431
432     C Move cursor relative to the (0,0) position using decipoint
433     C (1/720 inch) cursor positioning
434     IF (LANDSCAPE) THEN
435     XCURS=NINT(XPOS*7.2)+77
436     YCURS=NINT((719-YPOS)*7.2)+144
437     ELSE
438     XCURS=NINT(XPOS*7.2)+391
439     YCURS=NINT((1119-YPOS)*7.2)-216
440     ENDIF
441
442     WRITE(LINE,100) ABS(XCURS)
443     100 FORMAT('&a+',I4,'H')
444     IF (XCURS.LT.0) THEN
445     LINE(3:3)='- '
446     ENDIF
447     CALL HPOUT(ESC)
448     CALL HPASCII(8,LINE)
449
450     WRITE(LINE,200) ABS(YCURS)
451     200 FORMAT('&a+',I4,'V')
452     IF (YCURS.LT.0) THEN
453     LINE(3:3)='- '
454     ENDIF
455     CALL HPOUT(ESC)
456     CALL HPASCII(8,LINE)
457     RETURN
458     END
459
460     SUBROUTINE HPDRAW(X,Y)
461     C Anders Persson 1988
462     C Bresenham's line drawing algorithm, rewritten from a
463     C Pascal routine in: J.R. Van Aken and C.R. Killebrew Jr.,
464     C "Better Bit-Mapped Lines", Byte March 1988, page 249-253.
465
466     IMPLICIT LOGICAL (A-Z)
467     REAL X,Y
468     INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
469     LOGICAL LANDSCAPE
470     INTEGER*2 XSTART,YSTART,XEND,YEND,D,A,B,DXDIAG,DYDIAG
471     INTEGER*2 DXNONDIAG,DYNONDIAG,DIAGINC,NONDIAGINC
472     INTEGER*2 SWAP,RX,RY,I,WORD,BIT,LINE
473
474     COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
475
476     XSTART = XPOS
477     YSTART = YPOS

```

```

478 CALL HPMOVE(X,Y)
479 XEND = XPOS
480 YEND = YPOS
481
482 RX = XSTART
483 RY = YSTART
484 A = XEND-XSTART
485 B = YEND-YSTART
486
487 IF (A.LT.0) THEN
488   A = -A
489   DXDIAG=-1
490 ELSE
491   DXDIAG=1
492 ENDIF
493
494 IF (B.LT.0) THEN
495   B = -B
496   DYDIAG=-1
497 ELSE
498   DYDIAG=1
499 ENDIF
500
501 IF (A.LT.B) THEN
502   SWAP = A
503   A = B
504   B = SWAP
505   DXNONDIAG = 0
506   DYNONDIAG = DYDIAG
507 ELSE
508   DXNONDIAG = DXDIAG
509   DYNONDIAG = 0
510 ENDIF
511
512 D = B+B-A
513 NONDIAGINC = B+B
514 DIAGINC = B+B-A-A
515
516 DO 10 I=0,A
517   IF (LANDSCAPE) THEN
518     WORD = (719-RY)/16
519     BIT = 15-MOD(719-RY,16)
520     LINE = 1119-RX
521   ELSE
522     WORD = RX/16
523     BIT = 15-MOD(RX,16)
524     LINE = 1119-RY
525   ENDIF
526
527   IF (COLOUR.EQ.0) THEN
528     RASTER(WORD,LINE)=IBCLR(RASTER(WORD,LINE),BIT)
529   ELSEIF (COLOUR.EQ.2) THEN
530     RASTER(WORD,LINE)=IBCHNG(RASTER(WORD,LINE),BIT)
531   ELSE
532     RASTER(WORD,LINE)=IBSET(RASTER(WORD,LINE),BIT)
533   ENDIF
534
535   IF (D.LT.0) THEN
536     RX = RX+DXNONDIAG
537     RY = RY+DYNONDIAG

```

```

538     D = D+NONDIAGINC
539   ELSE
540     RX = RX+DXDIAG
541     RY = RY+DYDIAG
542     D = D+DIAGINC
543   ENDIF
544 10 CONTINUE
545 RETURN
546 END
547
548 SUBROUTINE HPLOT(X,Y)
549 C Anders Persson 1988
550
551 IMPLICIT LOGICAL (A-Z)
552 REAL X,Y
553 INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
554 LOGICAL LANDSCAPE
555 INTEGER*2 WORD,LINE,BIT
556 COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
557
558 CALL HPMOVE(X,Y)
559 IF (LANDSCAPE) THEN
560   WORD = (719-YPOS)/16
561   BIT = 15-MOD(719-YPOS,16)
562   LINE = 1119-XPOS
563 ELSE
564   WORD = XPOS/16
565   BIT = 15-MOD(XPOS,16)
566   LINE = 1119-YPOS
567 ENDIF
568
569 IF (COLOUR.EQ.0) THEN
570   RASTER(WORD,LINE)=IBCLR(RASTER(WORD,LINE),BIT)
571 ELSEIF (COLOUR.EQ.2) THEN
572   RASTER(WORD,LINE)=IBCHNG(RASTER(WORD,LINE),BIT)
573 ELSE
574   RASTER(WORD,LINE)=IBSET(RASTER(WORD,LINE),BIT)
575 ENDIF
576 RETURN
577 END
578
579 LOGICAL FUNCTION HPSET(X,Y)
580 C Anders Persson 1988
581
582 IMPLICIT LOGICAL (A-Z)
583 REAL X,Y
584 INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
585 LOGICAL LANDSCAPE
586 INTEGER*2 WORD,LINE,BIT
587 COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
588
589 CALL HPMOVE(X,Y)
590 IF (LANDSCAPE) THEN
591   WORD = (719-YPOS)/16
592   BIT = 15-MOD(719-YPOS,16)
593   LINE = 1119-XPOS
594 ELSE
595   WORD = XPOS/16
596   BIT = 15-MOD(XPOS,16)
597   LINE = 1119-YPOS

```

```
598 ENDIF
599
600 HPSET=BTEST(RASTER(WORD,LINE),BIT)
601 RETURN
602 END
603
604 SUBROUTINE HPRMOVE(X,Y)
605 C Anders Persson, 1988
606
607 IMPLICIT LOGICAL (A-Z)
608 REAL X,Y,IX,IY
609 INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
610 LOGICAL LANDSCAPE
611 COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
612
613 IX=XPOS+X
614 IY=YPOS+Y
615 CALL HPMOVE(IX,IY)
616 RETURN
617 END
618
619 SUBROUTINE HPRPLOT(X,Y)
620 C Anders Persson, 1988
621
622 IMPLICIT LOGICAL (A-Z)
623 REAL X,Y,IX,IY
624 INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
625 LOGICAL LANDSCAPE
626 COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
627
628 IX=XPOS+X
629 IY=YPOS+Y
630 CALL HPLOT(IX,IY)
631 RETURN
632 END
633
634 SUBROUTINE HPRDRAW(X,Y)
635 C Anders Persson, 1988
636
637 IMPLICIT LOGICAL (A-Z)
638 REAL X,Y,IX,IY
639 INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
640 LOGICAL LANDSCAPE
641 COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
642
643 IX=XPOS+X
644 IY=YPOS+Y
645 CALL HPDRAW(IX,IY)
646 RETURN
647 END
648
649 SUBROUTINE HPRPRINT(X,Y,LEN,MSG)
650 C Anders Persson, 1988
651
652 IMPLICIT LOGICAL (A-Z)
653 REAL X,Y,IX,IY
654 INTEGER LEN
655 CHARACTER MSG*(*)
656 INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
657 LOGICAL LANDSCAPE
```

```
658 COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
659
660 IX=XPOS+X
661 IY=YPOS+Y
662 CALL HPRPRINT(IX,IY,LEN,MSG)
663 RETURN
664 END
665
666 SUBROUTINE HPMARK(X,Y)
667 C Anders Persson, 1988
668
669 IMPLICIT LOGICAL (A-Z)
670 REAL X,Y,IX,IY
671 INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
672 LOGICAL LANDSCAPE
673 COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
674
675 IX=X
676 IY=Y
677 CALL HPLOT(IX,IY)
678 CALL HPLOT(IX+1.0,IY)
679 CALL HPLOT(IX-1.0,IY)
680 CALL HPLOT(IX,IY+1.0)
681 CALL HPLOT(IX,IY-1.0)
682 CALL HPMOVE(IX,IY)
683 RETURN
684 END
685
686 SUBROUTINE HPBOX(X,Y,XL,YL)
687 C Anders Persson, 1988
688
689 IMPLICIT LOGICAL (A-Z)
690 REAL X,Y,XL,YL,IX
691 INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119),SAVECOLOUR
692 LOGICAL LANDSCAPE
693 COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
694
695 SAVECOLOUR=COLOUR
696 CALL HPLEVEL(0)
697 DO 10 IX=X,X+XL,1.0
698 CALL HPMOVE(IX,Y)
699 CALL HPDRAW(IX,Y+YL)
700 10 CONTINUE
701
702 COLOUR=SAVECOLOUR
703 CALL HPMOVE(X,Y)
704 CALL HPDRAW(X+XL,Y)
705 CALL HPDRAW(X+XL,Y+YL)
706 CALL HPDRAW(X,Y+YL)
707 CALL HPDRAW(X,Y)
708 RETURN
709 END
710
711 SUBROUTINE HPTBOX(X,Y,LEN,MSG)
712 C Anders Persson, 1988
713
714 IMPLICIT LOGICAL (A-Z)
715 REAL X,Y,XL,YL
716 INTEGER LEN
717 CHARACTER MSG*(*)
```

```

718 INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
719 LOGICAL LANDSCAPE
720 COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
721
722 YL=13.0
723 XL=(LEN+2)*6.0
724 CALL HPBOX(X,Y,XL,YL)
725 CALL HPRPRINT(6.0,3.0,LEN,MSG)
726 RETURN
727 END
728
729 SUBROUTINE HPCIRCL(X,Y,R)
730 C Anders Persson, 1988
731
732 IMPLICIT LOGICAL (A-Z)
733 REAL TWOPI
734 PARAMETER (TWOPI=6.283185308)
735 REAL X,Y,R,IX,IY,ARG
736 INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119),I
737 LOGICAL LANDSCAPE
738 COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
739
740 CALL HPMOVE(X+R,Y)
741 DO 10 I=1,100
742 ARG = TWOPI*0.01*I
743 IX = X+R*COS(ARG)
744 IY = Y+R*SIN(ARG)
745 CALL HPDRAW(IX,IY)
746 10 CONTINUE
747 RETURN
748 END

```

```

1 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
2 C                               C
3 C   Library: HPFLIB.LIB         C
4 C                               C
5 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
6
7   SUBROUTINE HPSTART(ORIENTATION)
8 C   Anders Persson 1988
9
10  IMPLICIT LOGICAL (A-Z)
11  CHARACTER ORIENTATION*1,FILENAME*40
12  INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
13  INTEGER RECORDNUMBER
14  LOGICAL LANDSCAPE
15  INTEGER*2 INITDATA(1:52),I
16  COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
17  COMMON /RECBLOCK/ RECORDNUMBER
18
19  DATA INITDATA/
20  f          27,38,108,48,79,
21  f          27,40, 48,83,
22  f          27,40,115,48,80,
23  f          27,40,115,49,54,46,54,54,72,
24  f          27,40,115,55,46,53,86,
25  f          27,40,115,48,83,
26  f          27,40,115,48,66,
27  f          27,40,115,48,84,
28  f          27,42,116,49,48,48,82/
29
30 C   Data corresponds to:
31 C   Orientation           = Portrait or landscape
32 C   Symbol set            = Swedish,
33 C   Spacing                = Fixed,
34 C   Pitch                  = 16.66 CPI,
35 C   Point size            = 8.5,
36 C   Style                  = Upright,
37 C   Stroke weight         = Medium,
38 C   Typeface              = Line printer,
39 C   Raster Graphics Resolution = 100 DPI.
40
41 10  WRITE(*,*) 'Graphics file name:'
42  READ(*,100,ERR=10) FILENAME
43 100  FORMAT(A)
44  OPEN(UNIT=21,FILE=FILENAME,ACCESS='DIRECT',FORM='UNFORMATTED',
45  f    RECL=1,STATUS='NEW',ERR=10)
46  RECORDNUMBER=0
47
48  COLOUR=1
49  LANDSCAPE=ORIENTATION.EQ.'L'.OR.ORIENTATION.EQ.'l'
50  IF (LANDSCAPE) THEN
51  INITDATA(4)=49
52  ELSE
53  INITDATA(4)=48
54  ENDIF
55
56  DO 20 I=1,52
57  CALL HPOUT(INITDATA(I))
    
```

```

58 20  CONTINUE
59
60  CALL HPCLEAR
61  CALL HPMOVE(0.0,0.0)
62  RETURN
63  END
64
65  SUBROUTINE HPOUT(D)
66 C   Anders Persson 1988
67
68  IMPLICIT LOGICAL (A-Z)
69  CHARACTER CH*1
70  INTEGER RECORDNUMBER
71  INTEGER*2 D,XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
72  LOGICAL LANDSCAPE
73  COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
74  COMMON /RECBLOCK/ RECORDNUMBER
75
76  CH=CHAR(D)
77  RECORDNUMBER=RECORDNUMBER+1
78  WRITE(21,REC=RECORDNUMBER) CH
79
80  RETURN
81  END
82
83  SUBROUTINE HPDUMP
84 C   Anders Persson, 1988
85
86  IMPLICIT LOGICAL (A-Z)
87  INTEGER*2 XPOS,YPOS,COLOUR,RASTER(0:44,0:1119),I,J,D
88  INTEGER*2 DMPLINE(1:3),ESC,HIGHBYTE,LOWBYTE
89  LOGICAL LANDSCAPE
90  COMMON /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER
91
92  ESC=27
93  DMPLINE(1)=256*27+42
94  DMPLINE(2)=256*98+57
95  DMPLINE(3)=256*48+87
96 C  DMPLINE(1..3)=Esc * b 9 0 W = Raster line header
97
98 C  Set cursor position
99  IF (LANDSCAPE) THEN
100  CALL HPCMOVE(1119.0,719.0)
101  ELSE
102  CALL HPCMOVE(0.0,1119.0)
103  ENDIF
104 C  Initialize raster graphics transfer
105  CALL HPOUT(ESC)
106  CALL HPASCII(4,'*r1A')
107
108  DO 30 I=0,1119
109 C  Transfer raster graphics line
110  DO 10 J=1,3
111  D=HIGHBYTE(DMPLINE(J))
112  CALL HPOUT(D)
113  D=LOWBYTE(DMPLINE(J))
114  CALL HPOUT(D)
115 10  CONTINUE
116  DO 20 J=0,44
117  D=HIGHBYTE(RASTER(J,I))
    
```


Länkningslista:

LINK POL1+SCRPLOT+ROLPLOT+JETPLOT+RANF+NRMAPROX, , , HERCLIB+PLLIB+HPLIB+MATLIB
LINK POL2+DSCRLOT+DROLPLOT+DJETPLOT+RANF+NRMAPROX, , , HERCLIB+PLLIB+HPLIB+DMATLIB
LINK POL3+DSCRLOT+DROLPLOT+DJETPLOT+RANF+NRMAPROX, , , HERCLIB+PLLIB+HPLIB+DMATLIB
LINK CALCPOLC, , , ,
LINK ARCCROSS+DSCRLOT+DROLPLOT+DJETPLOT, , , HERCLIB+PLLIB+HPLIB+DMATLIB
LINK EXP1+SCRPLOT+ROLPLOT+JETPLOT+RANF+NRMAPROX, , , HERCLIB+PLLIB+HPLIB+MATLIB
LINK EXPN+SCRPLOT+ROLPLOT+JETPLOT+RANF+NRMAPROX, , , HERCLIB+PLLIB+HPLIB+MATLIB
LINK GAUSSN+SCRPLOT+ROLPLOT+JETPLOT+RANF+NRMAPROX, , , HERCLIB+PLLIB+HPLIB+MATLIB
LINK LORN+SCRPLOT+ROLPLOT+JETPLOT+RANF+NRMAPROX, , , HERCLIB+PLLIB+HPLIB+MATLIB
LINK PCONEXPA+SCRPLOT+ROLPLOT+JETPLOT+CALCW+FFT+RANF+NRMAPROX, , , HERCLIB+PLLIB+HPLIB+MATLIB
LINK PCONEXP+SCRPLOT+ROLPLOT+JETPLOT+CALCW+FFT+RANF+NRMAPROX, , , HERCLIB+PLLIB+HPLIB+MATLIB
LINK TESTMD+CALCW+FFT+RANF+NRMAPROX+PVEC, , , HERCLIB
LINK TFEXP+CALCW+FFT+RANF+NRMAPROX+PVEC, , , HERCLIB

Program och subrutinregister:

1:	POL1	15	59:	(MATLIB) INVMAT	134
2:	F	16	60:	(MATLIB) GAUSS	134
3:	POL2	20	61:	(MATLIB) ADDMAT	135
4:	F	21	62:	(MATLIB) SUBMAT	135
5:	POL3	24	63:	(MATLIB) MULTMAT	135
6:	F	25	64:	(MATLIB) TRANSMAT	135
7:	POL	25	65:	(MATLIB) ADDCMAT	135
8:	READPC	25	66:	(MATLIB) MULTCMAT	136
9:	CALCPOLC	32	67:	(MATLIB) COPYMAT	136
10:	ZEROVEC	33	68:	(MATLIB) UNITMAT	136
11:	ONEVEC	34	69:	(MATLIB) ZEROMAT	136
12:	ADDVEC	34	70:	(DMATLIB) INVMAT	137
13:	SUBVEC	35	71:	(DMATLIB) GAUSS	137
14:	MULTVEC	35	72:	(DMATLIB) ADDMAT	138
15:	DIVVEC	35	73:	(DMATLIB) SUBMAT	138
16:	CONVDP	35	74:	(DMATLIB) MULTMAT	138
17:	COPYVEC	36	75:	(DMATLIB) TRANSMAT	138
18:	ARCROSS	42	76:	(DMATLIB) ADDCMAT	138
19:	F	43	77:	(DMATLIB) MULTCMAT	139
20:	FITPOL	43	78:	(DMATLIB) COPYMAT	139
21:	EXP1	51	79:	(DMATLIB) UNITMAT	139
22:	F	53	80:	(DMATLIB) ZEROMAT	139
23:	DFDTH	53	81:	SCRLOT	141
24:	D2FDTH2	53	82:	(D) SCRLOT	142
25:	EXPN	63	83:	ROLLOT	143
26:	F	67	84:	(D) ROLLOT	146
27:	DFDTH	67	85:	JETLOT	148
28:	D2FDTH2	67	86:	(D) JETLOT	150
29:	GAUSSN	73	87:	(HERCLIB) ARC	153
30:	F	75	88:	(HERCLIB) BLKFIL	153
31:	DFDTH	76	89:	(HERCLIB) CIRC	153
32:	D2FDTH2	76	90:	(HERCLIB) CLRSCR	154
33:	LORN	82	91:	(HERCLIB) DISP	154
34:	F	84	92:	(HERCLIB) DLINE	154
35:	DFDTH	84	93:	(HERCLIB) FILL	154
36:	D2FDTH2	85	94:	(HERCLIB) GETPT	155
37:	PCONEXPA	95	95:	(HERCLIB) GMODE	155
38:	F	101	96:	(HERCLIB) GPAGE	155
39:	DFDTH	102	97:	(HERCLIB) LEVEL	155
40:	D2FDTH2	102	98:	(HERCLIB) MOVE	156
41:	CALCCONV	103	99:	(HERCLIB) PLOT	156
42:	MOVEDIFF	104	100:	(HERCLIB) TMODE	156
43:	PCONEXP	111	101:	(HERCLIB) HRDCPY	156
44:	F	117	102:	(HERCLIB) ONECHR	157
45:	DFDTH	117	103:	(HERCLIB) TEXT	157
46:	D2FDTH2	118	104:	(HERCLIB) KBDCHK	157
47:	CALCCONV	118	105:	(HERCLIB) KBDIN	157
48:	MOVEDIFF	120	106:	(HERCLIB) KBDINC	158
49:	CALCFEXP	120	107:	(PLLIB) WINITP	161
50:	TESTMD	123	108:	(PLLIB) WATOUT	161
51:	MOVEDIFF	124	109:	(PLLIB) RHOME	161
52:	TFEXP	127	110:	(PLLIB) WHOME	161
53:	CALCFEXP	129	111:	(PLLIB) RDRAW	162
54:	PVEC	131	112:	(PLLIB) WDRAW	162
55:	CALCW	131	113:	(PLLIB) RMOVE	162
56:	FFT	131	114:	(PLLIB) WMOVE	162
57:	RANF	132	115:	(PLLIB) RRDRAW	162
58:	NORMAL	132	116:	(PLLIB) WRDRAW	163

117:	(PLLIB)RRMOVE	163
118:	(PLLIB)WRMOVE	163
119:	(PLLIB)RLNTYP	163
120:	(PLLIB)WLNTYP	163
121:	(PLLIB)RLNSCL	164
122:	(PLLIB)WLNSCL	164
123:	(PLLIB)RAXIS	164
124:	(PLLIB)WAXIS	164
125:	(PLLIB)RPRINT	165
126:	(PLLIB)RASCAL	165
127:	(PLLIB)WASCAL	165
128:	(PLLIB)RAROT	165
129:	(PLLIB)WAROT	165
130:	(PLLIB)RMARK	166
131:	(PLLIB)WMARK	166
132:	(PLLIB)RPENCH	166
133:	(PLLIB)RCIRC	166
134:	(PLLIB)RRCIRC	166
135:	(PLLIB)RRCENT	167
136:	(PLLIB)RACIR	167
137:	(PLLIB)RAPROC	167
138:	(PLLIB)RHATCH	167
139:	(PLLIB)RCURVE	168
140:	(PLLIB)RRCURVE	168
141:	(PLLIB)RRDGLC	168
142:	(HPLIB)HPSTART	172
143:	(HPLIB)HPINIT	172
144:	(HPLIB)HPOUT	172
145:	(HPLIB)HPRLINE	173
146:	(HPLIB)HPDUMP	173
147:	(HPLIB)HPEJECT	174
148:	(HPLIB/HPFLIB)HPPRINT	174
149:	(HPLIB/HPFLIB)HPASCII	174
150:	(HPLIB/HPFLIB)HPLEVEL	174
151:	(HPLIB/HPFLIB)HPCLEAR	174
152:	(HPLIB/HPFLIB)HPMOVE	174
153:	(HPLIB/HPFLIB)HPCMOVE	175
154:	(HPLIB/HPFLIB)HPDRAW	175
155:	(HPLIB/HPFLIB)HPLOT	176
156:	(HPLIB/HPFLIB)HPSET	176
157:	(HPLIB/HPFLIB)HPRMOVE	177
158:	(HPLIB/HPFLIB)HPRPLOT	177
159:	(HPLIB/HPFLIB)HPRDRAW	177
160:	(HPLIB/HPFLIB)HPPRINT	177
161:	(HPLIB/HPFLIB)HPMARK	177
162:	(HPLIB/HPFLIB)HPBOX	177
163:	(HPLIB/HPFLIB)HPTBOX	177
164:	(HPLIB/HPFLIB)HPCIRCL	178
165:	(HPFLIB)HPSTART	179
166:	(HPFLIB)HPOUT	179
167:	(HPFLIB)HPDUMP	179
168:	(HPFLIB)HPEJECT	180
169:	(HPFLIB)LOWBYTE	180
170:	(HPFLIB)HIGHBYTE	180