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/

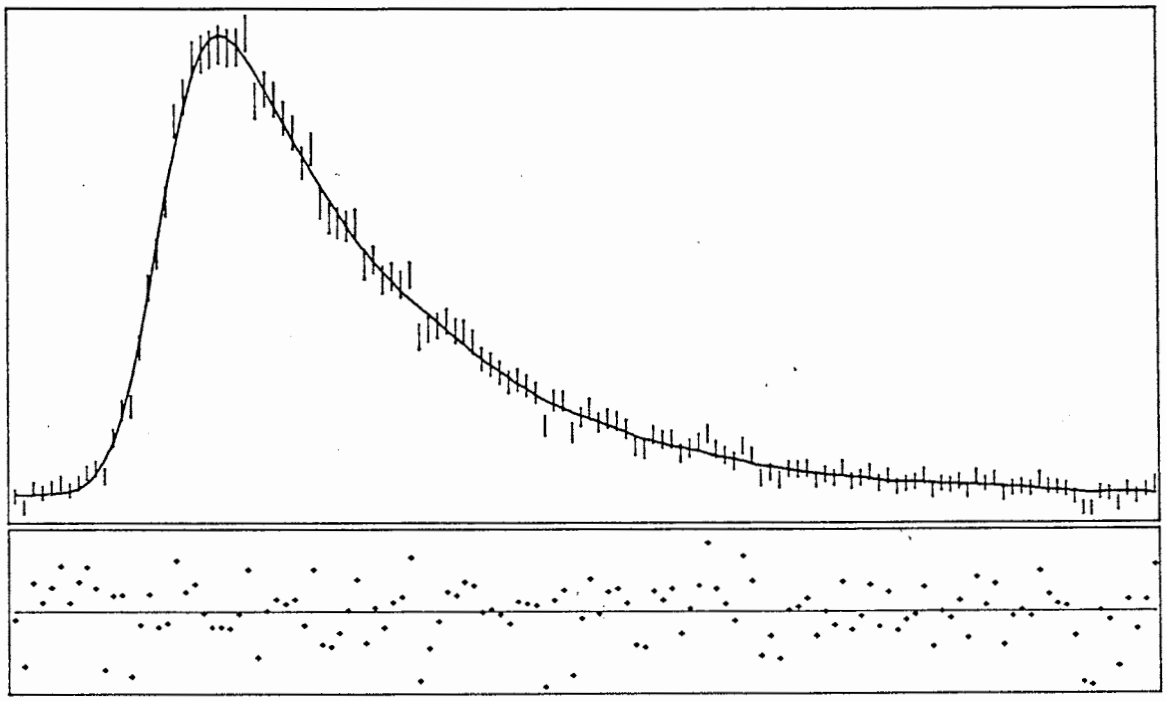
Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

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

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 grafikskärm). 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 $x_1, x_2, \ldots, x_N$ har mätvärden $y_1, y_2, \ldots, y_N$. Dessa mätvärden är behäftade med statistiska osäkerheter $\sigma_1, \sigma_2, \ldots, \sigma_N$ där σ_i är standardavvikelsen för mätvärdet y_i (x_i 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.

$$f_i = f(\theta_1, \theta_2, \ldots, \theta_L; x_i) = f(\underline{\theta}; x_i) \quad . \tag{1}$$

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

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

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

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

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

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

Ex. 1

Om y_i är Poissonfördelade är $\sigma_i = \sqrt{y_i}$ ⇒

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

Några av MK metodens fördelar är att man inte behöver känna fördelningsfunktionen för y_i (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_i är normalfördelade är ML och MK ekvivalenta.

Linjärt parameterberoende

Ex. 2

Funktionen linjär i parametrarna. Antag f_i=θ=konstant, y_i har medelvärde m och varians σ^2 (samma varians för alla punkter).

Minimera:

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

Minpunkt (eller max eller sadel) då

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

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

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

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 $1/\sigma^2=1$.

$$f_i = \theta_1 + \theta_2 \cdot x_i \tag{8}$$

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

Vi får ett ekvationssystem med två obekanta:

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

$\Leftrightarrow$

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

vars lösning är:

$$\begin{cases} \theta_1 = \dfrac{\Sigma x_i^2 \cdot \Sigma y_i - \Sigma x_i y_i \cdot \Sigma x_i}{N \cdot \Sigma x_i^2 - (\Sigma x_i)^2} \\ \theta_2 = \dfrac{N \cdot \Sigma x_i y_i - \Sigma x_i \cdot \Sigma y_i}{N \cdot \Sigma x_i^2 - (\Sigma x_i)^2} \end{cases} \qquad (12)$$

Det allmänna fallet där f är ett polynom av grad p

$$f_i = \theta_1 + \theta_2 x_i + \theta_3 x_i^2 + \cdots + \theta_{p+1} x_i^p \qquad (13)$$

ger ekvationssystemet:

$$\left(\sum_{i=1}^{N} x_i^{r+k-2} \right) \cdot \underline{\theta} = \left(\sum_{i=1}^{N} y_i x_i^{r-1} \right) \quad r,k=1,2,\ldots,p+1 \qquad (14)$$

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 polynomanpassning.

Ex. 4

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

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

$f(x)=A/(B+x^2)$ -- 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) $y_1 \pm \sigma_1$, $y_2 \pm \sigma_2, \ldots, y_N \pm \sigma_N$ i punkterna $x_1, x_2, \ldots, x_N$. Sätt:

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

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

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

döp om $a_j(x_i)$ till α_{ij}

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

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

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

detta ekvationssystem kan skrivas på formen:

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

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

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

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

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

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:

$$\mathbf{A} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \cdots & \alpha_{1L} \\ \alpha_{21} & \cdots & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \alpha_{N1} & \cdots & \cdots & \cdots & \alpha_{NL} \end{bmatrix} \qquad (22)$$

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

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

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

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

$$\nabla_{\theta}\chi^2 = -2(A^TV^{-1}\underline{y} - A^TV^{-1}A\underline{\theta}) = \underline{0} \qquad (25)$$

$\Leftrightarrow$ (19)

$$(A^TV^{-1}A)^{-1}\underline{\theta} = A^TV^{-1}\underline{y} \qquad (26)$$

Detta är lösbart om $A^TV^{-1}A$ ej är singulär och lösningen blir

$$\underline{\theta} = (A^TV^{-1}A)^{-1}A^TV^{-1}\underline{y} \qquad (27)$$

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:

$$V(\underline{\theta}) = (A^TV^{-1}A)^{-1} \qquad (28)$$

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

$$\underline{\theta} = V(\underline{\theta})A^TV^{-1}\underline{y} \qquad (29)$$

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

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

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

Ex. 5

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

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

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

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

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

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

blir A^TA enhetsmatrisen och lösningen $\underline{\theta}$ kan skrivas: $\underline{\theta} = A^T\underline{y}$. Kovariansmatrisen $V(\underline{\theta})$ blir också enkel, $V(\underline{\theta})=A^TV(\underline{y})A=\sigma^2A^TA=\sigma^2\underline{I}$ $\Rightarrow$ parametrarna är okorrelerade.

Ex. 6

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

Icke linjärt parameterberoende

Om funktionen $f(\underline{\theta};x)$ 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:

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

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

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

$$\underline{\theta}^m = [\theta_1^m, \theta_2^m, \ldots, \theta_L^m]^T \tag{36}$$

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

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

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

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

iteration m+1 skall ge

$$\underline{\theta}^{m+1}=\underline{\theta}^m+\Delta\underline{\theta}^m \tag{39}$$

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

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

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

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

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

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

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

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).

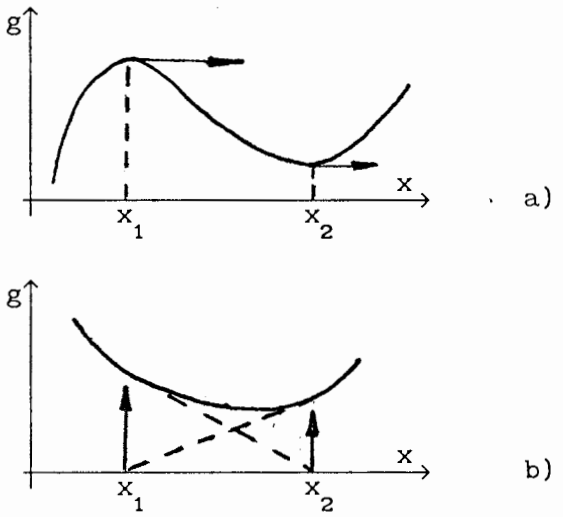


Fig. 1: $x^{k+1}=x^k-g(x^k)/g'(x^k)$. *Problem uppstår i punkterna* x_1 *och* x_2. *a) derivatan = 0, b)* x^{k+1} *hoppar mellan punkterna* x_1 *och* x_2.

Feluppskattning

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

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

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

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

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^2_{min} upp till $\chi^2_{min}+a$ 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 linearisera G-matrisen.

$$G_{kj} \approx \sum_{i=1}^{N} \left[\frac{2}{\sigma_i^2}\right] \frac{\partial f}{\partial \theta}^i_k \cdot \frac{\partial f}{\partial \theta}^i_j \qquad (46)$$

Denna är alltid positivt definit om derivatorna existerar och om den ej är singulär. 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 felen 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-Fetcher-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

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

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], $r(x_i)$

$$r(x_i) = (y_i - f_i)\cdot\sqrt{w_i} = \frac{y_i - f_i}{\sigma_i} \qquad (48)$$

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]

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

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

Medelvärdet och standardavvikelsen för de viktade resterna

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

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 HartleyTransform, JOSA 73, 1832,(1983).

[15] J. W. Hartwell: Is Hartley Really Faster (Letter), Byte July 1988 p. 26.

POL1

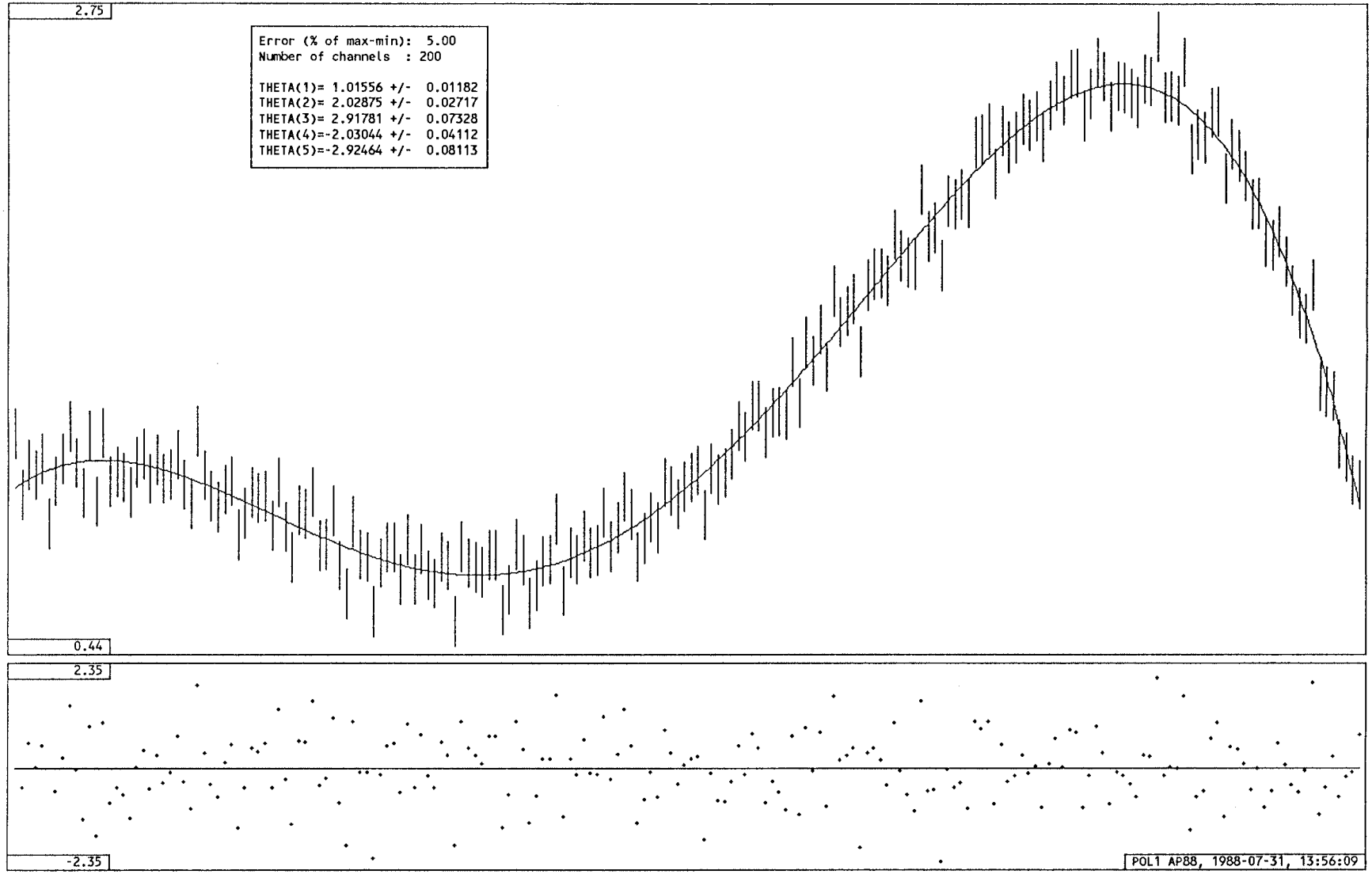
Programmet anpassar ett fjärdegradspolynom till valbart antal punkter, anpassningsfunktionen är:

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

Test kurvan beräknas i programmet genom att de exakta värdena beräknas och därefter adderas ett aproximativt 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
```


2.75
Error (% of max-min): 5.00
Number of channels : 200
THETA(1)= 1.01556 +/- 0.01182
THETA(2)= 2.02875 +/- 0.02717
THETA(3)= 2.91781 +/- 0.07328
THETA(4)=-2.03044 +/- 0.04112
THETA(5)=-2.92464 +/- 0.08113
0.44
2.35
-2.35
POL1 AP88, 1988-07-31, 13:56:09

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_i,y_i), $x\in[0,2]$, $y_i=N(f_i,\sigma_i)$.
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	Multiplicera 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 $f(\underline{\theta};x)$.

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: POL1.FOR    Options: /LBZ         08/02/88 14:47:04

      PROGRAM POL1
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    N,L
      PARAMETER  (N=1000, L=5)
      REAL       Y(1:N),X(1:N),DX,DET,RANF,THETAESTIMATE(1:L)
      REAL       SIGIN,RESIDUE(1:N),SIGMA(1:N),F,THETA(1:L),NORMAL
      REAL       B(1:L),A(1:L,1:L),JX,JY,JXL,JYL,DJY
      INTEGER    R,K,I,NCH
      INTEGER*2  HO,MI,SE,HN,YEAR,MONTH,DAY
      CHARACTER  SV*1,JETTXT*40
      LOGICAL    SING,ERRORBAR
      EXTERNAL   F

      THETA(1)= 1.0
      THETA(2)= 2.0
      THETA(3)= 3.0
      THETA(4)=-2.0
      THETA(5)=-3.0

10    WRITE(*,*) 'Sigma(Y) ( % of ymax-ymin ):'
      READ(*,*,ERR=10) SIGIN
      SIGIN=SIGIN*0.0178319

20    WRITE(*,1000) 'No of channels (',L,'..',N,'):'
1000  FORMAT(' ',A,I2,A,I4,A)
      READ(*,*,ERR=20) NCH
      IF (NCH.LT.L.OR.NCH.GT.N) GOTO 20

      WRITE(*,*) 'Plot errorbars? (Y/*):'
      READ(*,1100) SV
1100  FORMAT(A)
      ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'

      DX=2.0/(NCH-1)
      DO 30 I=1,NCH
        X(I)=(I-NCH/2)*DX
        Y(I)=NORMAL(F(THETA,L,X(I)),SIGIN)
30    CONTINUE

      CALL ZEROMAT(A,L,L,L,L)
      CALL ZEROMAT(B,L,1,L,1)
      CALL ZEROMAT(SIGMA,NCH,1,N,1)
      CALL ADDCMAT(SIGMA,SIGIN,NCH,1,N,1)

      DO 60 I=1,NCH
        B(1)=B(1)+Y(I)
        DO 40 R=2,L
          B(R)=B(R)+Y(I)*X(I)**(R-1)
40      CONTINUE
        A(1,1)=A(1,1)+1.0
        DO 50 R=1,L
        DO 50 K=R,L
          IF (R+K.NE.2) THEN
            A(R,K)=A(R,K)+X(I)**(R+K-2)
          ENDIF
          A(K,R)=A(R,K)
50      CONTINUE
60    CONTINUE

      IF (SIGIN.NE.0.0) THEN
        CALL MULTCMAT(A,1.0/SIGIN**2,L,L,L,L)
        CALL MULTCMAT(B,1.0/SIGIN**2,L,1,L,L)
      ENDIF

      CALL INVMAT(A,L,DET,SING,L,L)
      IF (SING) THEN
        WRITE(*,*) 'The A-matrix is singular.'
        GOTO 120
      ENDIF

      CALL MULTMAT(A,B,THETAESTIMATE,L,L,1,L,L,L,1,L,1)

      CALL SCRPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAESTIMATE,L,ERRORBAR)

70    WRITE(*,*) 'Plot on Roland plotter? (Y/*):'
      READ(*,1100) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAESTIMATE,L,ERRORBAR)
        GOTO 70
      ENDIF

80    WRITE(*,*) 'Plot on HP-Laserjet? (Y/*)   :'
      READ(*,1100) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CALL HPSTART('L')
        CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAESTIMATE,L,ERRORBAR)
        JX=200
        JXL=394-JX
        JYL=(3+1+L)*13
        JY=700-JYL
        CALL HPBOX(JX,JY,JXL,JYL)
        JY=700-16
        DJY=-13

        WRITE(JETTXT,1600) 'Error (% of max-min):',SIGIN/0.0178319
1600    FORMAT(' ',A,F6.2)
        CALL HPPRINT(JX,JY,40,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,1700) 'Number of channels  :',NCH
1700    FORMAT(' ',A,I4)
        CALL HPPRINT(JX,JY,40,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,1800) ' '
1800    FORMAT(' ',A)
        CALL HPPRINT(JX,JY,40,JETTXT)
        JY=JY+DJY

        DO 90 R=1,L
          IF (SIGIN.EQ.0.0) THEN
            WRITE(JETTXT,1200) 'THETA(',R,')=',THETAESTIMATE(R)
          ELSE
            WRITE(JETTXT,1200) 'THETA(',R,')=',THETAESTIMATE(R),' +/- ',
     f                  SQRT(ABS(A(R,R)))
```

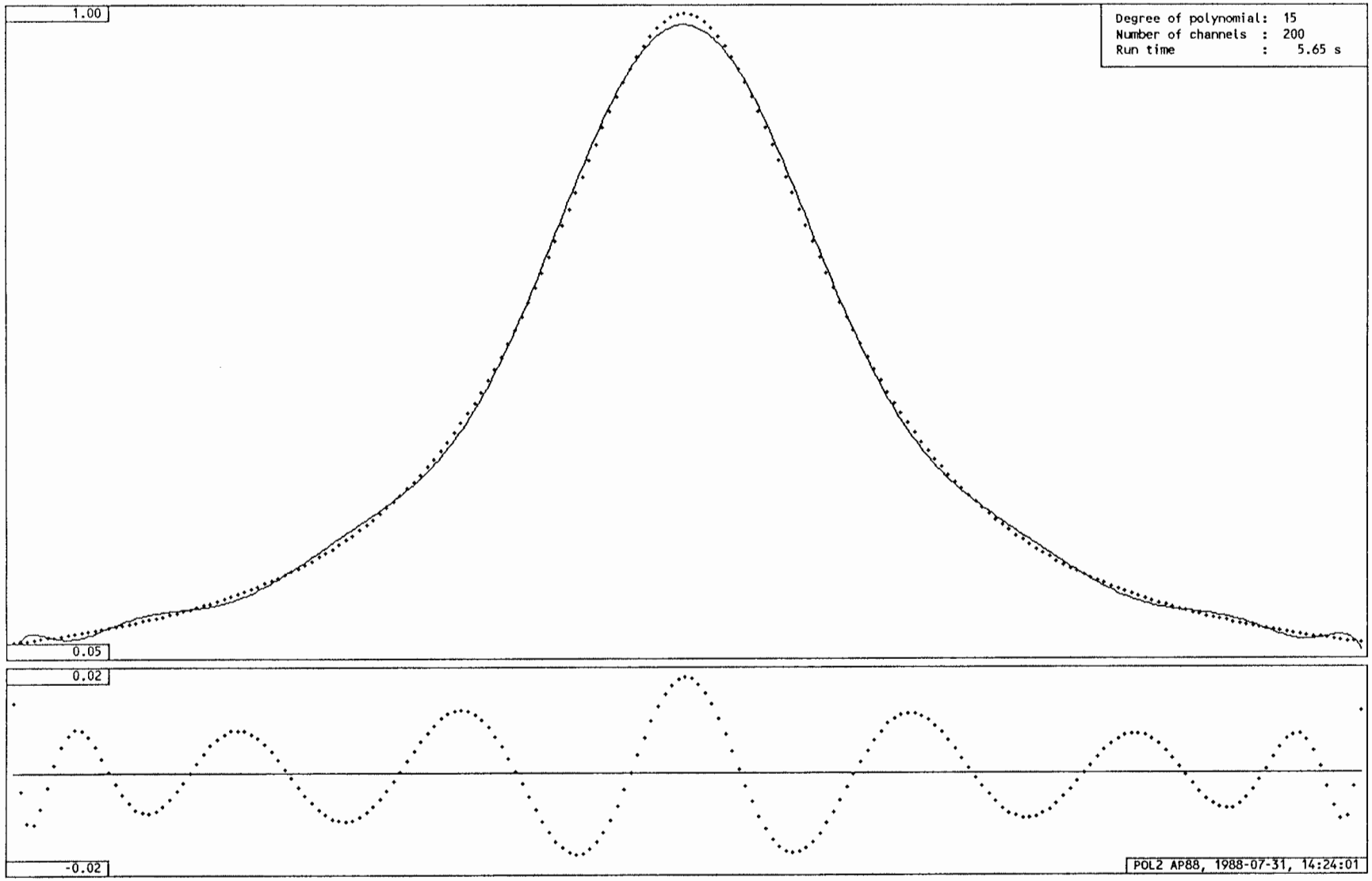
```
118           ENDIF
119           CALL HPPRINT(JX,JY,40,JETTXT)
120           JY=JY+DJY
121 90      CONTINUE
122
123         CALL GETTIM(HO,MI,SE,HN)
124         CALL GETDAT(YEAR,MONTH,DAY)
125         WRITE(JETTXT,1900)
126       f  'POL1 AP88, ',YEAR,'-',MONTH,'-',DAY,', ',HO,':',MI,':',SE
127 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
```

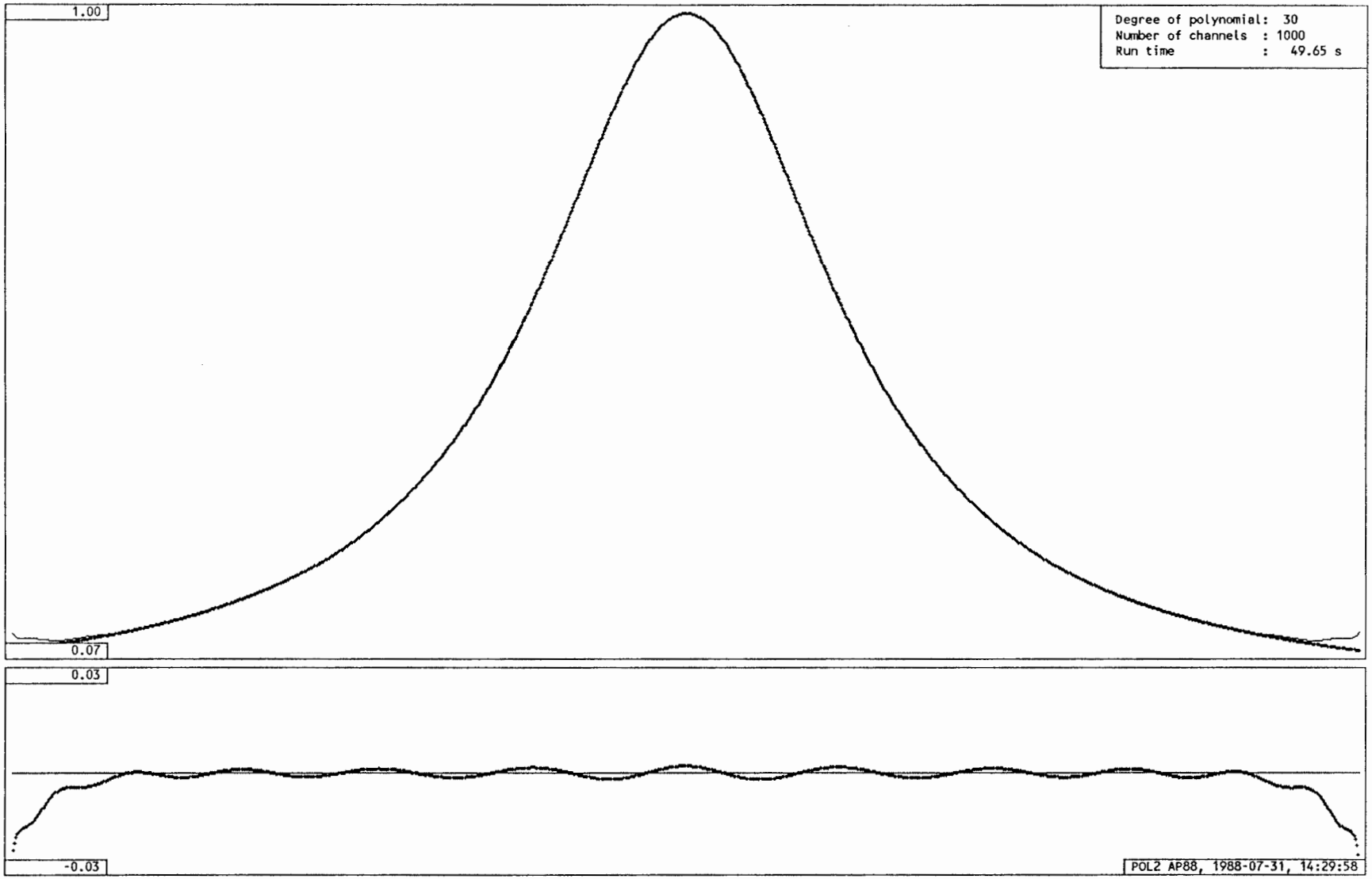
POL2

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
```


1.00
0.05
Degree of polynomial: 15
Number of channels : 200
Run time : 5.65 s
0.02
-0.02
POL2 AP88, 1988-07-31, 14:24:01


1.00
0.07
Degree of polynomial: 30
Number of channels : 1000
Run time : 49.65 s
0.03
-0.03
POL2 AP88, 1988-07-31, 14:29:58

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: POL2.FOR    Options: /LBZ         08/02/88 14:47:48

      PROGRAM POL2
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    N,L,ASLASKRANGE
      PARAMETER  (N=1000, L=100, ASLASKRANGE=L+L-2)
      REAL*8     Y(1:N),X(1:N),DX,DET,TIME,THETAESTIMATE(1:L),A(1:L,1:L)
      REAL*8     B(1:L),SIGIN,RESIDUE(1:N),SIGMA(1:N),F
      REAL*8     ASLASK(0:ASLASKRANGE),SL1,SL2,SL3
      REAL       JX,JY,JXL,JYL,DJY
      INTEGER    R,K,I,NCH,POLDEGREE,LRANGE
      INTEGER*2  HO,MI,SE,HN,ANS,KBDCHK,KBDINC
      INTEGER*2  YEAR,MONTH,DAY
      CHARACTER  SV*1,JETTXT*40
      LOGICAL    SING,ERRORBAR
      EXTERNAL   F

      SIGIN=1.0D0
      ERRORBAR=.FALSE.
      CALL ZEROMAT(SIGMA,N,1,N,1)
      CALL ADDCMAT(SIGMA,1.0D0,N,1,N,1)

10    WRITE(*,1000) 'Degree of polynomial (0..',L-1,'):'
1000  FORMAT(' ',A,I2,A)
      READ(*,*,ERR=10) POLDEGREE
      IF (POLDEGREE.LT.0.OR.POLDEGREE.GT.L-1) THEN
        GOTO 10
      ENDIF
      LRANGE=POLDEGREE+1

20    WRITE(*,1100) 'No of channels (',LRANGE,'..',N,'):'
1100  FORMAT(' ',A,I3,A,I4,A)
      READ(*,*,ERR=20) NCH
      IF (NCH.LT.LRANGE.OR.NCH.GT.N) THEN
        GOTO 20
      ENDIF

      DX=1.5D0/(NCH-1)
      DO 30 I=1,NCH
        X(I)=(I-NCH*0.5D0)*DX
        Y(I)=1.0D0/(1.0D0+25.0D0*X(I)*X(I))
30    CONTINUE

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0D0+MI*60.0D0+SE+HN*0.01D0

      CALL ZEROMAT(A,LRANGE,LRANGE,L,L)
      CALL ZEROMAT(B,LRANGE,1,L,1)
      CALL ZEROMAT(ASLASK,LRANGE+LRANGE-1,1,ASLASKRANGE+1,1)

      DO 40 I=1,NCH
        ASLASK(0)=ASLASK(0)+1.0D0
        DO 40 R=1,LRANGE+LRANGE-2
          ASLASK(R)=ASLASK(R)+X(I)**R
40    CONTINUE

      DO 50 R=1,LRANGE
        DO 50 K=R,LRANGE
          A(R,K)=ASLASK(R+K-2)
          A(K,R)=A(R,K)
50    CONTINUE

      DO 70 I=1,NCH
        SL1=Y(I)
        SL2=X(I)
        SL3=SL2
        B(1)=B(1)+SL1
        DO 60 R=2,LRANGE
          B(R)=B(R)+SL1*SL2
          SL2=SL2*SL3
60      CONTINUE
70    CONTINUE

      CALL INVMAT(A,LRANGE,DET,SING,L,L)
      IF (SING) THEN
        WRITE(*,*) 'The A-matrix is singular.'
        GOTO 110
      ENDIF
      CALL MULTMAT(A,B,THETAESTIMATE,LRANGE,LRANGE,1,L,L,L,1,L,1)

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0D0+MI*60.0D0+SE+HN*0.01D0-TIME

      WRITE(*,1200) 'Run time:',TIME,' s.'
1200  FORMAT(' ',A,F8.2,A)
      WRITE(*,*) 'Press any key to continue.'
80    ANS=KBDCHK()
      IF (ANS.EQ.0) THEN
        GOTO 80
      ENDIF
      ANS=KBDINC()

      CALL SCRPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
     f             THETAESTIMATE,LRANGE,ERRORBAR)

90    WRITE(*,*) 'Plot on Roland plotter? (Y/*):'
      READ(*,1300) SV
1300  FORMAT(A)
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
     f               THETAESTIMATE,LRANGE,ERRORBAR)
        GOTO 90
      ENDIF

100   WRITE(*,*) 'Plot on HP-Laserjet? (Y/*): '
      READ(*,1300) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CALL HPSTART('L')
        CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
     f               THETAESTIMATE,LRANGE,ERRORBAR)
        JX=900
        JXL=1119-JX
        JYL=(3+1)*13
        JY=719-JYL
        CALL HPBOX(JX,JY,JXL,JYL)
        JX=906
        JY=719-16
```

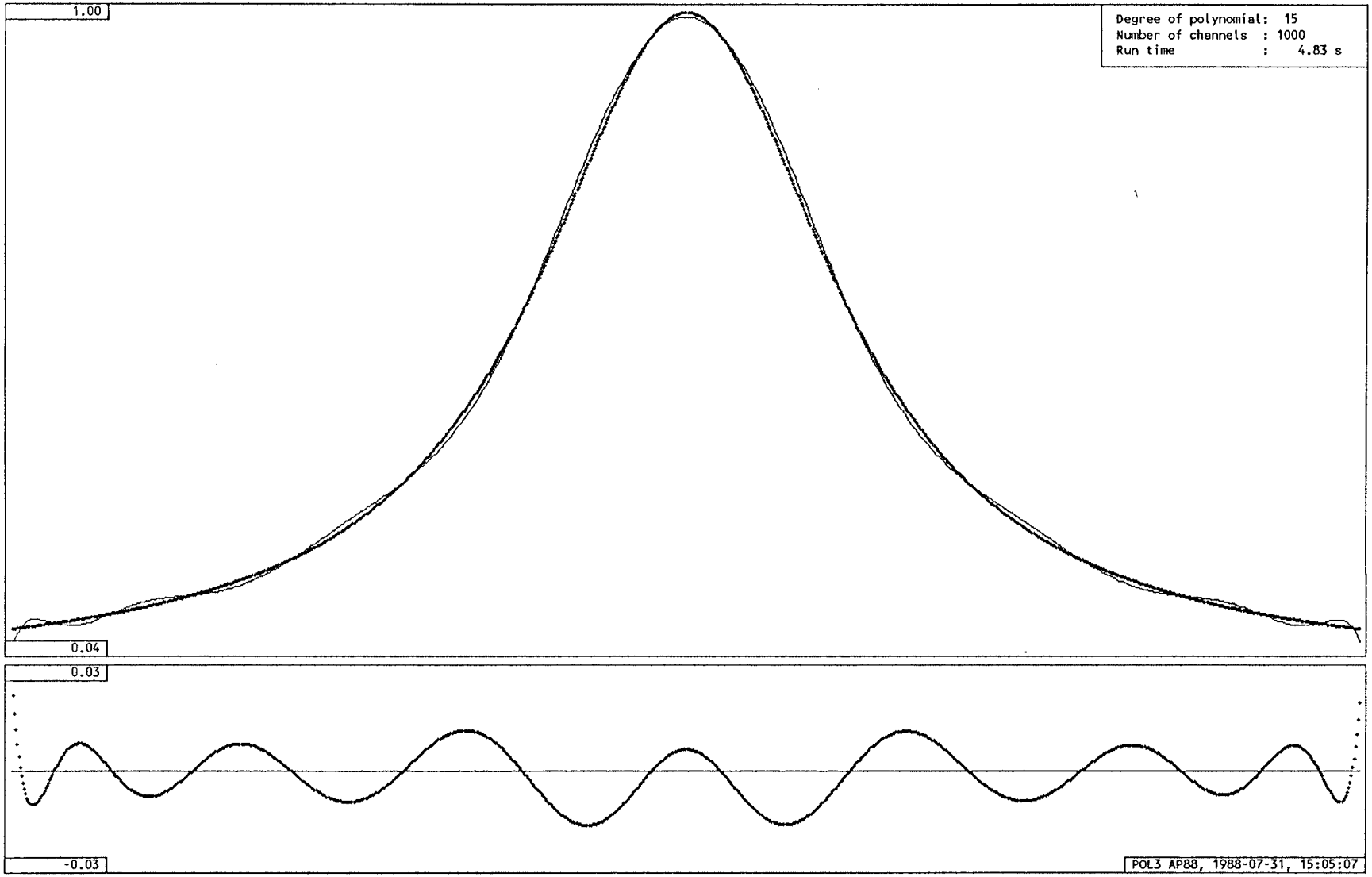
```
      DJY=-13

      WRITE(JETTXT,1400) 'Degree of polynomial:',POLDEGREE
1400  FORMAT(' ',A,I4)
      CALL HPPRINT(JX,JY,32,JETTXT)
      JY=JY+DJY

      WRITE(JETTXT,1500) 'Number of channels  : ',NCH
1500  FORMAT(' ',A,I4)
      CALL HPPRINT(JX,JY,32,JETTXT)
      JY=JY+DJY

      WRITE(JETTXT,1600) 'Run time            :',TIME,' s.'
1600  FORMAT(' ',A,F8.2,A)
      CALL HPPRINT(JX,JY,32,JETTXT)
      JY=JY+DJY

      CALL GETTIM(HO,MI,SE,HN)
      CALL GETDAT(YEAR,MONTH,DAY)
      WRITE(JETTXT,1800)
     f 'POL2 AP88, ',YEAR,'-',MONTH,'-',DAY,', ',HO,':',MI,':',SE
1800  FORMAT(A,I4.4,5(A,I2.2))
      CALL HPTBOX(921.0,0.0,31,JETTXT)

      CALL HPEJECT
      GOTO 100
    ENDIF

110 WRITE(*,1700) 'New calculation? (Y/*):'
1700 FORMAT(/' ',A)
    READ(*,1300) SV
    IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
      GOTO 10
    ELSE
      STOP
    ENDIF
    END

    REAL*8 FUNCTION F(THETA,L,X)
C   Anders Persson, 1988

    IMPLICIT   LOGICAL (A-Z)
    INTEGER    L,I
    REAL*8     THETA(1:L),X,SLASK

    SLASK=THETA(1)
    IF (X.NE.0.0D0) THEN
      DO 10 I=2,L
        SLASK=SLASK+THETA(I)*X**(I-1)
10    CONTINUE
    ENDIF
    F=SLASK

    RETURN
    END
```

POL3

Programmet utför exakt samma beräkning som POL2 men genpm 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
```


1.00
Degree of polynomial: 15
Number of channels : 1000
Run time : 4.83 s
0.04
0.03
-0.03
POL3 AP88, 1988-07-31, 15:05:07

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: POL3.FOR    Options: /LBZ         08/02/88 14:48:31

      PROGRAM POL3
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    N,L,POLVECRANGE,CRANGE
      PARAMETER  ( N=1000, L=50, POLVECRANGE=L+L-2 )
      PARAMETER  ( CRANGE=((L+L)*(L+L-1))/2 )
      REAL*8     Y(1:N),X(1:N),DX,DET,TIME,THETAESTIMATE(1:L),A(1:L,1:L)
      REAL*8     B(1:L),SIGIN,RESIDUE(1:N),SIGMA(1:N),F,SL1,SL2,SL3
      REAL*8     POL,POLVEC(0:POLVECRANGE),C(1:CRANGE)
      REAL       JX,JY,JXL,JYL,DJY
      INTEGER    R,K,I,NCH,POLDEGREE,LRANGE,S
      INTEGER*2  HO,MI,SE,HN,ANS,KBDINC,KBDCHK
      INTEGER*2  YEAR,MONTH,DAY
      CHARACTER  SV*1,JETTXT*40
      LOGICAL    SING,ERRORBAR
      EXTERNAL   F

      S       =CRANGE
      SIGIN   =1.0D0
      ERRORBAR=.FALSE.

      CALL READPC(C,S)
      CALL ZEROMAT(SIGMA,N,1,N,1)
      CALL ADDCMAT(SIGMA,1.0D0,N,1,N,1)

10    WRITE(*,1000) 'Degree of polynomial (0..',L-1,'):'
1000  FORMAT(' ',A,I2,A)
      READ(*,*,ERR=10) POLDEGREE
      IF (POLDEGREE.LT.0.OR.POLDEGREE.GT.L-1) THEN
        GOTO 10
      ENDIF
      LRANGE=POLDEGREE+1

20    WRITE(*,1100) 'No of channels (',LRANGE,'..',N,'):'
1100  FORMAT(' ',A,I3,A,I4,A)
      READ(*,*,ERR=20) NCH
      IF (NCH.LT.LRANGE.OR.NCH.GT.N) THEN
        GOTO 20
      ENDIF

      DX=1.5D0/(NCH-1)
      DO 30 I=1,NCH
        X(I)=(I-NCH*0.5D0)*DX
        Y(I)=1.0D0/(1.0D0+25.0D0*X(I)*X(I))
30    CONTINUE

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0D0+MI*60.0D0+SE+HN*0.01D0
      CALL ZEROMAT(B,LRANGE,1,L,1)

      DO 40 I=0,LRANGE+LRANGE-2
        IF (MOD(I,2).EQ.0) THEN
          POLVEC(I)=2.0D0*POL(I,NCH/2,C)*(DX**(I))
        ELSE
          POLVEC(I)=0.0D0
        ENDIF
40    CONTINUE

      POLVEC(0)=NCH
      DO 50 R=1,LRANGE
      DO 50 K=R,LRANGE
        A(R,K)=POLVEC(R+K-2)
        A(K,R)=POLVEC(R+K-2)
50    CONTINUE

      DO 70 I=1,NCH
        SL1=Y(I)
        SL2=X(I)
        SL3=SL2
        B(1)=B(1)+SL1
        DO 60 R=2,LRANGE
          B(R)=B(R)+SL1*SL2
          SL2=SL2*SL3
60      CONTINUE
70    CONTINUE

      CALL INVMAT(A,LRANGE,DET,SING,L,L)
      IF (SING) THEN
        WRITE(*,*) 'The A-matrix is singular.'
        GOTO 110
      ENDIF

      CALL MULTMAT(A,B,THETAESTIMATE,LRANGE,LRANGE,1,L,L,L,1,L,1)

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0D0+MI*60.0D0+SE+HN*0.01D0-TIME

      WRITE(*,1200) 'Run time:',TIME,'s.'
1200  FORMAT(' ',A,F8.2,A)

      WRITE(*,*) 'Press any key to continue.'
80    ANS=KBDCHK()
      IF (ANS.EQ.0) THEN
        GOTO 80
      ENDIF
      ANS=KBDINC()

      CALL SCRPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
     f             THETAESTIMATE,LRANGE,ERRORBAR)

90    WRITE(*,*) 'Plot on Roland plotter? (Y/*):'
      READ(*,1300) SV
1300  FORMAT(A)
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
     f               THETAESTIMATE,LRANGE,ERRORBAR)
        GOTO 90
      ENDIF

100   WRITE(*,*) 'Plot on HP-Laserjet? (Y/*): '
      READ(*,1300) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CALL HPSTART('L')
        CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
     f               THETAESTIMATE,LRANGE,ERRORBAR)
        JX=900
```

```
      JXL=1119-JX
      JYL=(3+1)*13
      JY=719-JYL
      CALL HPBOX(JX,JY,JXL,JYL)
      JX=906
      JY=719-16
      DJY=-13

      WRITE(JETTXT,1400) 'Degree of polynomial:',POLDEGREE
1400  FORMAT(' ',A,I4)
      CALL HPPRINT(JX,JY,32,JETTXT)
      JY=JY+DJY

      WRITE(JETTXT,1500) 'Number of channels  : ',NCH
1500  FORMAT(' ',A,I4)
      CALL HPPRINT(JX,JY,32,JETTXT)
      JY=JY+DJY

      WRITE(JETTXT,1600) 'Run time            :',TIME,' s.'
1600  FORMAT(' ',A,F8.2,A)
      CALL HPPRINT(JX,JY,32,JETTXT)

      CALL GETTIM(HO,MI,SE,HN)
      CALL GETDAT(YEAR,MONTH,DAY)
      WRITE(JETTXT,1800)
     f  'POL3 AP88, ',YEAR,'-',MONTH,'-',DAY,', ',HO,':',MI,':',SE
1800  FORMAT(A,I4.4,5(A,I2.2))
      CALL HPTBOX(921.0,0.0,31,JETTXT)

      CALL HPEJECT
      GOTO 100
    ENDIF

110 WRITE(*,1700) 'New calculation? (Y/*):'
1700 FORMAT(/' ',A)
    READ(*,1300) SV
    IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
      GOTO 10
    ELSE
      STOP
    ENDIF
    END

    REAL*8 FUNCTION F(THETA,L,X)
C   Anders Persson, 1988

    IMPLICIT   LOGICAL (A-Z)
    INTEGER    L,I
    REAL*8     THETA(1:L),X,SLASK

    SLASK=THETA(1)
    IF (X.NE.0.0D0) THEN
      DO 10 I=2,L
        SLASK=SLASK+THETA(I)*X**(I-1)
10    CONTINUE
    ENDIF
    F=SLASK

    RETURN
    END

    DOUBLE PRECISION FUNCTION POL(Q,N,C)
C   Anders Persson, 1987

    IMPLICIT   LOGICAL (A-Z)
    REAL*8     C(1:*),PROD,SUM
    INTEGER    Q,N,I,J,OFFS

    OFFS=(Q*(Q+1))/2
    SUM=1.0D0
    DO 10 I=1,Q+1
      PROD=1.0D0
      DO 20 J=1,I
        PROD=PROD*(N-J)
20    CONTINUE
      SUM=SUM+PROD*C(OFFS+I)
10  CONTINUE
    POL=SUM

    RETURN
    END

    SUBROUTINE READPC(C,S)
C   Anders Persson, 1988

    IMPLICIT   LOGICAL (A-Z)
    REAL*8     C(1:*)
    INTEGER    S,I
    LOGICAL    EX

    INQUIRE(FILE='CCOEFF.DAT',EXIST=EX)

    IF (.NOT.EX) THEN
      WRITE(*,*)
    f'The unformatted file: CCOEFF:DAT containing the Cq,i values is '
      WRITE(*,*)
    f'missing. You must supply one or create one by running the '
      WRITE(*,*)
    f'program CALCPOLC.'
      STOP
    ENDIF

    OPEN(10,FILE='CCOEFF.DAT',ACCESS='DIRECT',
    f     FORM='UNFORMATTED',RECL=8)

    DO 10 I=1,S
      READ(10,REC=I) C(I)
10  CONTINUE

    RETURN
    END
```

CALCPOLC

Summan

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

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

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

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

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

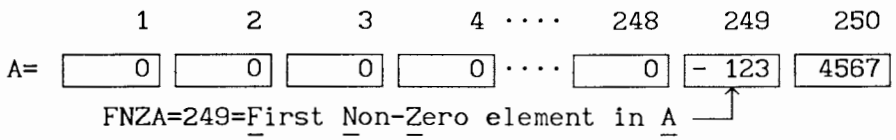
$$C_{q,i} = \frac{1}{i} \cdot \frac{1}{(i-1)!} \sum_{K=0}^{i-1} \binom{i-1}{k} (-1)^k (i+1-k)^q = \frac{1}{i} \sum_{k=0}^{i-1} \frac{(-1)^k (i+1-k)^q}{k! \cdot (i-1-k)!} \qquad (54)$$

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:

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

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 utrycken för C-kefficienterna. 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.



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

C(0,1)= 1/1

C(1,1)= 2/1
C(1,2)= 1/2

C(2,1)= 4/1
C(2,2)= 5/2
C(2,3)= 1/3

C(3,1)= 8/1
C(3,2)= 19/2
C(3,3)= 9/3
C(3,4)= 1/4

C(4,1)= 16/1
C(4,2)= 65/2
C(4,3)= 55/3
C(4,4)= 14/4
C(4,5)= 1/5

C(5,1)= 32/1
C(5,2)= 211/2
C(5,3)= 285/3
C(5,4)= 125/4
C(5,5)= 20/5
C(5,6)= 1/6

C(6,1)= 64/1
C(6,2)= 665/2
C(6,3)= 1351/3
C(6,4)= 910/4
C(6,5)= 245/5
C(6,6)= 27/6
C(6,7)= 1/7

C(7,1)= 128/1
C(7,2)= 2059/2
C(7,3)= 6069/3
C(7,4)= 5901/4
C(7,5)= 2380/5
C(7,6)= 434/6
C(7,7)= 35/7
C(7,8)= 1/8

C(8,1)= 256/1
C(8,2)= 6305/2
C(8,3)= 26335/3
C(8,4)= 35574/4
C(8,5)= 20181/5
C(8,6)= 5418/6
C(8,7)= 714/7
C(8,8)= 44/8
C(8,9)= 1/9

C(9,1)= 512/1
C(9,2)= 19171/2
C(9,3)= 111645/3
C(9,4)= 204205/4
C(9,5)= 156660/5
C(9,6)= 58107/6
C(9,7)= 11130/7
C(9,8)= 1110/8
C(9,9)= 54/9
C(9,10)= 1/10

C(10,1)= 1024/1
C(10,2)= 58025/2
C(10,3)= 465751/3
C(10,4)= 1132670/4
C(10,5)= 1144165/5
C(10,6)= 563409/6
C(10,7)= 147147/7
C(10,8)= 21120/8
C(10,9)= 1650/9
C(10,10)= 65/10
C(10,11)= 1/11

C(11,1)= 2048/1
C(11,2)= 175099/2
C(11,3)= 1921029/3
C(11,4)= 6129101/4
C(11,5)= 7997660/5
C(11,6)= 5088028/6
C(11,7)= 1740585/7
C(11,8)= 337227/8
C(11,9)= 37620/9
C(11,10)= 2365/10
C(11,11)= 77/11
C(11,12)= 1/12

C(12,1)= 4096/1
C(12,2)= 527345/2
C(12,3)= 7859215/3
C(12,4)= 3256534/4
C(12,5)= 54115061/5
C(12,6)= 43613856/6
C(12,7)= 19012708/7
C(12,8)= 4775628/8
C(12,9)= 713427/9
C(12,10)= 63635/10
C(12,11)= 3289/11
C(12,12)= 90/12
C(12,13)= 1/13

C(13,1)= 8192/1
C(13,2)= 1586131/2
C(13,3)= 31964205/3
C(13,4)= 170691885/4
C(13,5)= 357256900/5
C(13,6)= 359412053/6
C(13,7)= 195715520/7
C(13,8)= 61993360/8
C(13,9)= 11909898/9
C(13,10)= 1413412/10
C(13,11)= 103103/11
C(13,12)= 4459/12
C(13,13)= 104/13
C(13,14)= 1/14

C(14,1)= 16384/1
C(14,2)= 4766585/2
C(14,3)= 129442951/3
C(14,4)= 885423630/4
C(14,5)= 2314233285/5
C(14,6)= 2873141271/6
C(14,7)= 1925136213/7
C(14,8)= 753655760/8
C(14,9)= 181092340/9
C(14,10)= 27457430/10
C(14,11)= 2650648/11
C(14,12)= 161070/12
C(14,13)= 5915/13
C(14,14)= 119/14
C(14,15)= 1/15

C(15,1)= 32768/1
C(15,2)= 14316139/2
C(15,3)= 522538389/3
C(15,4)= 4556561101/4
C(15,5)= 14770823340/5
C(15,6)= 22426222182/6
C(15,7)= 18274230975/7
C(15,8)= 8708038053/8
C(15,9)= 2564579160/9
C(15,10)= 483124070/10
C(15,11)= 59265206/11
C(15,12)= 4744558/12
C(15,13)= 243880/13
C(15,14)= 7700/14
C(15,15)= 135/15
C(15,16)= 1/16

C(16,1)= 65536/1
C(16,2)= 42981185/2
C(16,3)= 2104469695/3
C(16,4)= 23305343894/4
C(16,5)= 93181501141/5
C(16,6)= 1717543786l4/6
C(16,7)= 168620069982/7
C(16,8)= 96646573452/8
C(16,9)= 34353829653/9
C(16,10)= 7878943930/10
C(16,11)= 1194306542/11
C(16,12)= 120944460/12
C(16,13)= 8158878/13
C(16,14)= 359380/14
C(16,15)= 9860/15
C(16,16)= 152/16
C(16,17)= 1/17

C(17,1)= 131072/1
C(17,2)= 129009091/2
C(17,3)= 8460859965/3
C(17,4)= 118631189165/4
C(17,5)= 582394350740/5
C(17,6)= 1295462151439/6
C(17,7)= 1520714938470/7
C(17,8)= 1038439231050/8
C(17,9)= 440184869982/9
C(17,10)= 121022212883/10
C(17,11)= 22210622434/11
C(17,12)= 2766584522/12
C(17,13)= 235168752/13
C(17,14)= 13549578/14
C(17,15)= 517140/15
C(17,16)= 12444/16
C(17,17)= 170/17
C(17,18)= 1/18

C(18,1)= 262144/1
C(18,2)= 387158345/2
C(18,3)= 33972448951/3
C(18,4)= 601616805790/4
C(18,5)= 3612997293605/5
C(18,6)= 9650629410813/6
C(18,7)= 13461181659199/7
C(18,8)= 10866668017920/8
C(18,9)= 5440287930870/9
C(18,10)= 1771429211695/10
C(18,11)= 387549682091/11
C(18,12)= 58176221220/12
C(18,13)= 6058947050/13
C(18,14)= 438412422/14
C(18,15)= 21823818/15
C(18,16)= 728688/16
C(18,17)= 15504/17
C(18,18)= 189/18
C(18,19)= 1/19

C(19,1)= 524288/1
C(19,2)= 1161737179/2
C(19,3)= 136276954149/3
C(19,4)= 30420564779O1/4
C(19,5)= 22279600567420/5
C(19,6)= 71167403169296/6
C(19,7)= 117340082684405/7
C(19,8)= 11126119382O479/8
C(19,9)= 65269547326620/9
C(19,10)= 24926009259515/10
C(19,11)= 6422025396787/11
C(19,12)= 1143840557951/12
C(19,13)= 143001479920/13
C(19,14)= 12635133380/14
C(19,15)= 787593510/15
C(19,16)= 34211514/16
C(19,17)= 1007760/17
C(19,18)= 19095/18
C(19,19)= 209/19
C(19,20)= 1/20

C(20,1)= 1048576/1
C(20,2)= 3485735825/2
C(20,3)= 546269553775/3
C(20,4)= 15346559343654/4
C(20,5)= 136719659882421/5
C(20,6)= 520451422752492/6
C(20,7)= 1009880864644536/7
C(20,8)= 1118690827068716/8
C(20,9)= 763956667086679/9
C(20,10)= 339455649181285/10
C(20,11)= 1019903140209591/11
C(20,12)= 21291952650150/12
C(20,13)= 3145861276831/13
C(20,14)= 332528480620/14
C(20,15)= 25236629540/15
C(20,16)= 1369189248/16
C(20,17)= 52351194/17
C(20,18)= 1370565/18
C(20,19)= 23275/19
C(20,20)= 230/20
C(20,21)= 1/21

C(21,1)= 2097152/1
C(21,2)= 10458256051/2
C(21,3)= 2188563950925/3
C(21,4)= 772790662720451/4
C(21,5)= 835664518638180/5
C(21,6)= 3779879619149865/6
C(21,7)= 8599555939908780/7
C(21,8)= 11078105508262980/8
C(21,9)= 8758257497935506/9
C(21,10)= 44979688080808l4/10
C(21,11)= 1563339417432793/11
C(21,12)= 378785698472909/12
C(21,13)= 65334010525784/13
C(21,14)= 81337884861311/14
C(21,15)= 736314553260/15
C(21,16)= 48512846756/16
C(21,17)= 2311510740/17
C(21,18)= 78391929/18
C(21,19)= 1836065/19
C(21,20)= 28105/20
C(21,21)= 252/21
C(21,22)= 1/22

C(22,1)= 4194304/1
C(22,2)= 31376865305/2
C(22,3)= 8764714059751/3
C(22,4)= 388583895311150/4
C(22,5)= 5091266178101125/5
C(22,6)= 2729482185268723l/6
C(22,7)= 725763271384201O5/7
C(22,8)= 108302505514275600/8
C(22,9)= 98660680487618040/9
C(22,10)= 582359143868244l0/10
C(22,11)= 23258041817274330/11
C(22,12)= 6487553497580610/12
C(22,13)= 1293461845833885/13
C(22,14)= 187340837817749/14
C(22,15)= 199148213382911/15
C(22,16)= 1561032948112/16
C(22,17)= 90120040076/17
C(22,18)= 3800957391/18
C(22,19)= 115113229/19
C(22,20)= 2426270/20
C(22,21)= 33649/21
C(22,22)= 275/22
C(22,23)= 1/23

C(23,1)= 8388608/1

C(23,2)= 94134790219/2
C(23,3)= 35090233104309/3
C(23,4)= 1951684190615501/4
C(23,5)= 3093618096391790 0/5
C(23,6)= 1961550191469117 70/6
C(23,7)= 607905438960048075/7
C(23,8)= 1047298876766900505/8
C(23,9)= 1094909310390456000/9
C(23,10)= 7392557387426871 00/10
C(23,11)= 3373324161941164 20/11
C(23,12)= 1075962372858222 60/12
C(23,13)= 2459601933925500 0/13
C(23,14)= 4103574413100120/14
C(23,15)= 5059779792304 05/15
C(23,16)= 464523814561 95/16
C(23,17)= 318319366948 0/17
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)= 18756876997412290 1/5
C(24,6)= 1404021314992300 290/6
C(24,7)= 5059398530827296370/7
C(24,8)= 1003359532986215 2620/8
C(24,9)= 1199639190671460505/9
C(24,10)= 9226722436560014100/10
C(24,11)= 4787244733072084140/11
C(24,12)= 1736083500909805800/12
C(24,13)= 4519405080353922 60/13
C(24,14)= 8614963553575 6800/14
C(24,15)= 1219922208078 6600/15
C(24,16)= 1295668463985720/16
C(24,17)= 1037498675068 35/17
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)= 491080109981165 25/4
C(25,5)= 1135206131030919220/5
C(25,6)= 1001571797492022 4931/6
C(25,7)= 4187920956161067 1250/7
C(25,8)= 9536175649958666 9950/8
C(25,9)= 1299975151365767 57670/9
C(25,10)= 1134903387828316 15605/10
C(25,11)= 6667365923342502 3780/11
C(25,12)= 2735633024489955 9540/12
C(25,13)= 8063250613405297440/13
C(25,14)= 1744185041071744260/14
C(25,15)= 2813371888283424 00/15
C(25,16)= 342255859685438 40/16
C(25,17)= 316316607910875 0/17
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)= 2461021576721717 10/4
C(26,5)= 6860344797183631845/5
C(26,6)= 7124523195547249 3737/6
C(26,7)= 3450493944678055 94931/7
C(26,8)= 9001350180578907 00800/8
C(26,9)= 1395336907865354 246650/9
C(26,10)= 1378391241747724529325/10
C(26,11)= 9135742495839319 00965/11
C(26,12)= 4223059524171192 97800/12
C(26,13)= 1402418388325737 23700/13
C(26,14)= 3422602622948146 1340/14
C(26,15)= 6245580062325222660/15
C(26,16)= 8631721502935876 80/16
C(26,17)= 911625753925013 40/17
C(26,18)= 7397024399454690/18
C(26,19)= 461948117390760/19
C(26,20)= 221575774348 50/20
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)= 4140817094077396 2780/5
C(27,6)= 5055769684854910 88004/6
C(27,7)= 2831640387697917 253185/7
C(27,8)= 8446264556988821902131/8
C(27,9)= 1485350409671143 3167300/9
C(27,10)= 1655764056709032 4069225/10
C(27,11)= 1234128223675490 7340905/11
C(27,12)= 6403551631006482 772365/12
C(27,13)= 2385691696073151 429600/13
C(27,14)= 6536322322747956 43800/14
C(27,15)= 1341553072266850 23900/15
C(27,16)= 2091950661731621 3220/16
C(27,17)= 250409850735861 1800/17
C(27,18)= 231706038982140450/18
C(27,19)= 166359867472698 90/19
C(27,20)= 9272572435226 10/20
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)= 3600592036072673 5/3
C(28,4)= 6172799805435714374/4
C(28,5)= 2496817856909860 46581/5
C(28,6)= 3580446950339211 578808/6
C(28,7)= 2315870007006882 9113484/7
C(28,8)= 7884802140059731 4372364/8
C(28,9)= 1569813055241031 53575131/9
C(28,10)= 1969875503347049 97928775/10
C(28,11)= 1646530274081492 12160085/11
C(28,12)= 9558745343983918 3381650/12
C(28,13)= 3980323537603060 2786765/13
C(28,14)= 1219017518019508 6086600/14
C(28,15)= 2800117147901756 026200/15
C(28,16)= 4897869197210606 48640/16
C(28,17)= 6599327974977122 5620/17
C(28,18)= 690651324801928 0350/18
C(28,19)= 5644257739275382 50/19
C(28,20)= 3610838886124 4700/20
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)= 309000049475392986 05/4
C(29,5)= 150426351395135199 3860/5
C(29,6)= 253128104380654670 98237/6
C(29,7)= 188850047510889844 486680/7
C(29,8)= 732790892675444658464760/8
C(29,9)= 164866107664162885 0123674/9
C(29,10)= 232384435920585813 0791656/10
C(29,11)= 217282387923249554 3849795/11
C(29,12)= 140728992212605859 6121535/12
C(29,13)= 652832748704267622 396360/13
C(29,14)= 222655863078956894 085765/14
C(29,15)= 569920495466231825 05800/15
C(29,16)= 111264947831597870 53080/16
C(29,17)= 167766595521694270 9800/17
C(29,18)= 197217031462137552270/18
C(29,19)= 181950287265700453 50/19
C(29,20)= 132270194001367695 0/20
C(29,21)= 75866679154145880/21
C(29,22)= 34281434595021 90/22
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)= 5762548611708199751/3
C(30,4)= 154644071295663419470/4
C(30,5)= 905648108865565126 1765/5
C(30,6)= 178693936580409621681519/6
C(30,7)= 153611319052518422299 1677/7
C(30,8)= 678396808158989177 0669520/8
C(30,9)= 172194016590917331597 01500/9
C(30,10)= 272109490279060682888 31890/10
C(30,11)= 283977309099958046569 89196/11
C(30,12)= 204675928668712572934 29750/12
C(30,13)= 105469484039858053096 70575/13
C(30,14)= 399267069488862103368 2835/14
C(30,15)= 113452865582492781417 8565/15
C(30,16)= 246142460860339562408160/16
C(30,17)= 413244819770647558 29480/17
C(30,18)= 542478955299755620 2930/18
C(30,19)= 561117605993538459270/19
C(30,20)= 459717694668572613 00/20
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 koefficienterna skall beräknas. q≈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 koefficienterna skall lagras.
65-69	Lagra talen $2,3,4,\ldots,q_{max}+2$ i hjälpvektorn SL.
71-206	Beräkna och lagra koefficienterna.
72-76	Om q≥2 multipliceras talen i hjälpvektorn SL med $2,3,4,\ldots,q_{max}+2$. Resultatet blir att SL innehåller talen $2^q,3^q,4^q,\ldots,(q_{max}+2)^q$ dessa används senare i beräkningen av koefficienterna.
78-80	Skriv ut på skärmen (utan radframmatning) vilken C-koefficient 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

      PROGRAM CALCPOLC
C     Anders Persson, 1988
C     Must be compiled with compiler option /Z (optimization = off)
C
      IMPLICIT   LOGICAL (A-Z)
      INTEGER    SLASK(1:250),C(1:250),SL(1:250,2:252),FNZ(2:252)
      INTEGER    FIRSTNZ,NPRINT,QMAX,Q,ARRPOS,NSIFF,N,I,K,J,SLM,SLD,M
      INTEGER    FNZC,FNZS,LOW1,UP1,LOW2,UP2
      REAL*8     DC
      CHARACTER  FMT*36,SLASK3*3,SV*1,FILECEX*40,FILECDPASCII*40
      CHARACTER  FILECDPUNFORM*40
      LOGICAL    CEX,DPASC,DPUNF

      FMT    = '(T2,A,Ia,A,Ib,A,T13,Ic,pppI4.4,A,Ib)'
      N      = 250
      FNZC   = 1
      FNZS   = 1
      CALL   ZEROVEC(SLASK,N,FNZS)
      CALL   ZEROVEC(C,N,FNZC)
      NSIFF  = 4
      M      = 10**NSIFF

10    WRITE(*,*) 'Qmax:'
      READ(*,*,ERR=10) QMAX
      IF (QMAX.LT.0.OR.QMAX.GT.250) THEN
        GOTO 10
      ENDIF

      CEX    = .FALSE.
      DPASC  = .FALSE.
      DPUNF  = .FALSE.

      WRITE(*,*) 'Save file with the exact Cq,i values? (Y/*):'
      READ(*,1000) SV
1000  FORMAT(A)
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CEX=.TRUE.
        WRITE(*,*) 'Filename:'
        READ(*,1000) FILECEX
        OPEN(UNIT=10,FILE=FILECEX)
      ENDIF

      WRITE(*,*) 'Save file with the formatted double precision Cq,i'
      WRITE(*,*) 'values? (Y/*):'
      READ(*,1000) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        DPASC=.TRUE.
        WRITE(*,*) 'Filename:'
        READ(*,1000) FILECDPASCII
        OPEN(UNIT=11,FILE=FILECDPASCII)
      ENDIF

      WRITE(*,*) 'Save file with the unformatted double precision Cq,i'
      WRITE(*,*) 'values? (Y/*):'
      READ(*,1000) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        DPUNF=.TRUE.
        WRITE(*,*) 'Filename:'
        READ(*,1000) FILECDPUNFORM
        OPEN(UNIT=12,FILE=FILECDPUNFORM,ACCESS='DIRECT',
     f       FORM='UNFORMATTED',RECL=8)
      ENDIF

      DO 20 I=2,QMAX+2
        FNZ(I)=1
        CALL ZEROVEC(SL(1,I),N,FNZ(I))
        SL(N,I)=I
20    CONTINUE

      DO 90 Q=0,QMAX
        IF (Q.GE.2) THEN
          DO 30 I=2,QMAX+2
            CALL MULTVEC(SL(1,I),N,NSIFF,I,FNZ(I))
30        CONTINUE
        ENDIF

        DO 80 I=1,Q+1
          WRITE(*,1100) 'C(',Q,',',I,')'
1100      FORMAT('+',A,I3,A,I3,A)

          CALL ZEROVEC(C,N,FNZC)
          DO 60 K=0,I-1
            IF (Q.NE.0) THEN
              CALL COPYVEC(SL(1,I+1-K),SLASK,N,FNZ(I+1-K),FNZS)
            ELSE
              CALL ONEVEC(SLASK,N,FNZS)
            ENDIF

            IF (K.GT.I-1-K) THEN
              LOW1  = K+1
              UP1   = I-1
              LOW2  = 2
              UP2   = I-1-K
            ELSE
              LOW1  = I-K
              UP1   = I-1
              LOW2  = 2
              UP2   = K
            ENDIF

            SLM=1
            DO 40 J=LOW1,UP1
              SLM=SLM*J
              IF (SLM*(J+1).GE.M.OR.J.EQ.UP1) THEN
                CALL MULTVEC(SLASK,N,NSIFF,SLM,FNZS)
                SLM=1
              ENDIF
40          CONTINUE

            SLD=1
            DO 50 J=LOW2,UP2
              SLD=SLD*J
              IF (SLD*(J+1).GE.M.OR.J.EQ.UP2) THEN
                CALL DIVVEC(SLASK,N,NSIFF,SLD,FNZS)
                SLD=1
              ENDIF
```

```
50          CONTINUE

            IF (MOD(K,2).EQ.0) THEN
              CALL ADDVEC(C,SLASK,N,NSIFF,FNZC,FNZS)
            ELSE
              CALL SUBVEC(C,SLASK,N,NSIFF,FNZC,FNZS)
            ENDIF

60        CONTINUE

          SLD=1
          DO 70 J=2,I-1
            SLD=SLD*J
            IF (SLD*(J+1).GE.M.OR.J.EQ.I-1) THEN
              CALL DIVVEC(C,N,NSIFF,SLD,FNZC)
              SLD=1
            ENDIF
70        CONTINUE

          IF (DPASC.OR.DPUNF) THEN
            CALL CONVDP(C,N,NSIFF,DC,FNZC)
            DC    = DC/I
            ARRPOS= Q*(Q+1)/2+I
          ENDIF

          IF (CEX) THEN
            FIRSTNZ=FNZC
            NPRINT=N-FNZC

            IF (Q.LE.9) THEN
              FMT(8:8)='1'
            ELSEIF (Q.LE.99) THEN
              FMT(8:8)='2'
            ELSE
              FMT(8:8)='3'
            ENDIF

            IF (I.LE.9) THEN
              FMT(13:13)='1'
              FMT(35:35)='1'
            ELSEIF (I.LE.99) THEN
              FMT(13:13)='2'
              FMT(35:35)='2'
            ELSE
              FMT(13:13)='3'
              FMT(35:35)='3'
            ENDIF

            IF (C(FIRSTNZ).LE.9) THEN
              FMT(22:22)='1'
            ELSEIF (C(FIRSTNZ).LE.99) THEN
              FMT(22:22)='2'
            ELSEIF (C(FIRSTNZ).LE.999) THEN
              FMT(22:22)='3'
            ELSE
              FMT(22:22)='4'
            ENDIF

            WRITE(SLASK3,1200) NPRINT
1200        FORMAT(I3)
            FMT(24:26)=SLASK3

            IF (NPRINT.EQ.0) THEN
             FMT(24:31)='        '
            ELSE
              FMT(27:31)='I4.4,'
            ENDIF

            WRITE(10,FMT) 'C(',Q,',',I,')=',C(FIRSTNZ),
     f      (C(J), J=FIRSTNZ+1,N),'/',I
          ENDIF

          IF (DPASC) THEN
            WRITE(11,1300) 'C(',ARRPOS,')=',DC
1300        FORMAT(' ',A,I5,A,D28.20)
          ENDIF

          IF (DPUNF) THEN
            WRITE(12,REC=ARRPOS) DC
          ENDIF

80      CONTINUE

          IF (CEX) THEN
            WRITE(10,1400) '  '
1400        FORMAT(' ',A)
          ENDIF

90      CONTINUE

        IF (CEX) THEN
          CLOSE(10)
        ENDIF

        IF (DPASC) THEN
          CLOSE(11)
        ENDIF

        IF (DPUNF) THEN
          CLOSE(12)
        ENDIF

        STOP
        END

        SUBROUTINE ZEROVEC(A,N,FNZ)
C       Anders Persson, 1988

        IMPLICIT  LOGICAL (A-Z)
        INTEGER   A(1:*),N,I,FNZ

        DO 10 I=FNZ,N
          A(I)=0
10      CONTINUE
        FNZ=N
        RETURN
        END

        SUBROUTINE ONEVEC(A,N,FNZ)
C       Anders Persson, 1988
```

```
      IMPLICIT  LOGICAL (A-Z)
      INTEGER   A(1:*),N,I,FNZ

      DO 10 I=FNZ,N
        A(I)=0
10    CONTINUE
      A(N)=1
      FNZ=N
      RETURN
      END

      SUBROUTINE ADDVEC(A,B,N,NSIFF,FNZA,FNZB)
C     Anders Persson, 1988

      IMPLICIT  LOGICAL (A-Z)
      INTEGER   A(1:*),B(1:*),N,NSIFF,M,I,K,FNZA,FNZB,FNZ
      LOGICAL   NEG,DIFFSIGN,BLTZERO,ABSBLTABSA

      FNZ = MIN(FNZA,FNZB)
      M   = 10**NSIFF

      IF (A(FNZA).GE.0.AND.B(FNZB).GE.0) THEN
        NEG      = .FALSE.
        DIFFSIGN = .FALSE.
      ELSEIF (A(FNZA).LE.0.AND.B(FNZB).LE.0) THEN
        NEG      = .TRUE.
        DIFFSIGN = .FALSE.
        A(FNZA)  = -A(FNZA)
        B(FNZB)  = -B(FNZB)
      ELSE
        DIFFSIGN = .TRUE.
      ENDIF

      IF (.NOT.DIFFSIGN) THEN
        DO 10 K=N,FNZ,-1
          I=A(K)+B(K)
          IF (I.GE.M) THEN
            I     = I-M
            A(K-1)= A(K-1)+1
          ENDIF
          A(K)=I
10      CONTINUE

        A(FNZ-1)=A(FNZ-1)+B(FNZ-1)
        FNZA=N
        DO 20 K=FNZ-1,N
          IF (A(K).NE.0) THEN
            FNZA=K
            GOTO 21
          ENDIF
20      CONTINUE
21      CONTINUE

        IF (NEG) THEN
          A(FNZA)=-A(FNZA)
        ENDIF
        RETURN
      ENDIF

      BLTZERO=B(FNZB).LT.0
      IF (BLTZERO) THEN
        B(FNZB)=-B(FNZB)
      ELSE
        A(FNZA)=-A(FNZA)
      ENDIF

      ABSBLTABSA=.TRUE.
      DO 30 K=FNZ,N
        IF (B(K).LT.A(K)) THEN
          ABSBLTABSA=.TRUE.
          GOTO 31
        ELSEIF (B(K).GT.A(K)) THEN
          ABSBLTABSA=.FALSE.
          GOTO 31
        ENDIF
30    CONTINUE
31    CONTINUE

      IF (.NOT.ABSBLTABSA) THEN
        DO 40 K=FNZ,N
          I    = A(K)
          A(K) = B(K)
          B(K) = I
40      CONTINUE

        I    = FNZA
        FNZA = FNZB
        FNZB = I
      ENDIF

      DO 50 K=N,FNZ,-1
        I=A(K)-B(K)
        IF (I.LT.0) THEN
          I     = I+M
          A(K-1)= A(K-1)-1
        ENDIF
        A(K)=I
50    CONTINUE

      A(FNZ-1)=A(FNZ-1)-B(FNZ-1)
      FNZA=N
      DO 60 K=FNZ-1,N
        IF (A(K).NE.0) THEN
          FNZA=K
          GOTO 61
        ENDIF
60    CONTINUE
61    CONTINUE

      IF (BLTZERO.AND.(.NOT.ABSBLTABSA)) THEN
        A(FNZA)=-A(FNZA)
      ELSEIF ((.NOT.BLTZERO).AND.ABSBLTABSA) THEN
        A(FNZA)=-A(FNZA)
      ENDIF

      RETURN
      END

      SUBROUTINE SUBVEC(A,B,N,NSIFF,FNZA,FNZB)
```

```
C     Anders Persson, 1988

      IMPLICIT  LOGICAL (A-Z)
      INTEGER   A(1:*),B(1:*),N,NSIFF,FNZA,FNZB

      B(FNZB)=-B(FNZB)
      CALL ADDVEC(A,B,N,NSIFF,FNZA,FNZB)
      RETURN
      END

      SUBROUTINE MULTVEC(A,N,NSIFF,MULT,FNZA)
C     Anders Persson, 1988

      IMPLICIT  LOGICAL (A-Z)
      INTEGER   A(1:*),N,NSIFF,MULT,K,I,M,PLUS,FNZA,SAVEMULT
      LOGICAL   NEG

      SAVEMULT = MULT
      NEG      = .FALSE.

      IF (A(FNZA).LT.0.AND.MULT.GT.0) THEN
        NEG = .TRUE.
      ELSEIF (A(FNZA).GT.0.AND.MULT.LT.0) THEN
        NEG = .TRUE.
      ENDIF

      A(FNZA) = ABS(A(FNZA))
      MULT    = ABS(MULT)
      M       = 10**NSIFF
      PLUS    = 0

      DO 10 K=N,FNZA,-1
        I=A(K)*MULT+PLUS
        IF (I.GE.M) THEN
          PLUS = I/M
          I    = I-PLUS*M
        ELSE
          PLUS = 0
        ENDIF
        A(K)=I
10    CONTINUE

      A(FNZA-1)=A(FNZA-1)*MULT+PLUS
      IF (A(FNZA-1).NE.0) THEN
        FNZA=FNZA-1
      ENDIF

      IF (NEG) THEN
        A(FNZA)=-A(FNZA)
      ENDIF
      MULT=SAVEMULT

      RETURN
      END

      SUBROUTINE DIVVEC(A,N,NSIFF,DIV,FNZA)
C     Anders Persson, 1988

      IMPLICIT  LOGICAL (A-Z)
      INTEGER   A(1:*),N,NSIFF,DIV,K,I,M,MD,FNZA,SAVEDIV
      LOGICAL   NEG

      SAVEDIV  = DIV
      NEG      = .FALSE.

      IF (A(FNZA).LT.0.AND.DIV.GT.0) THEN
        NEG = .TRUE.
      ELSEIF (A(FNZA).GT.0.AND.DIV.LT.0) THEN
        NEG = .TRUE.
      ENDIF

      A(FNZA)  = ABS(A(FNZA))
      DIV      = ABS(DIV)
      M        = 10**NSIFF

      DO 10 K=FNZA,N-1
        I      = A(K)/DIV
        MD     = A(K)-I*DIV
        A(K+1) = A(K+1)+MD*M
        A(K)   = I
10    CONTINUE

      A(N)=A(N)/DIV
      IF (A(FNZA).EQ.0) THEN
        FNZA=FNZA+1
      ENDIF

      IF (NEG) THEN
        A(FNZA)=-A(FNZA)
      ENDIF
      DIV=SAVEDIV

      RETURN
      END

      SUBROUTINE CONVDP(A,N,NSIFF,C,FNZA)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      REAL*8     C
      INTEGER    A(1:*),N,NSIFF,M,K,FNZA,LAST
      LOGICAL    NEG

      NEG     = A(FNZA).LT.0
      A(FNZA) = ABS(A(FNZA))
      LAST    = MIN(N,FNZA+25/NSIFF)
      C       = 0.0D0

      DO 10 K=FNZA,LAST
        IF (A(K).NE.0) THEN
          M=(N-K)*NSIFF
          C=C+DBLE(A(K))*(10.0D0**M)
        ENDIF
10    CONTINUE

      IF (NEG) THEN
        C=-C
      ENDIF

      RETURN
```

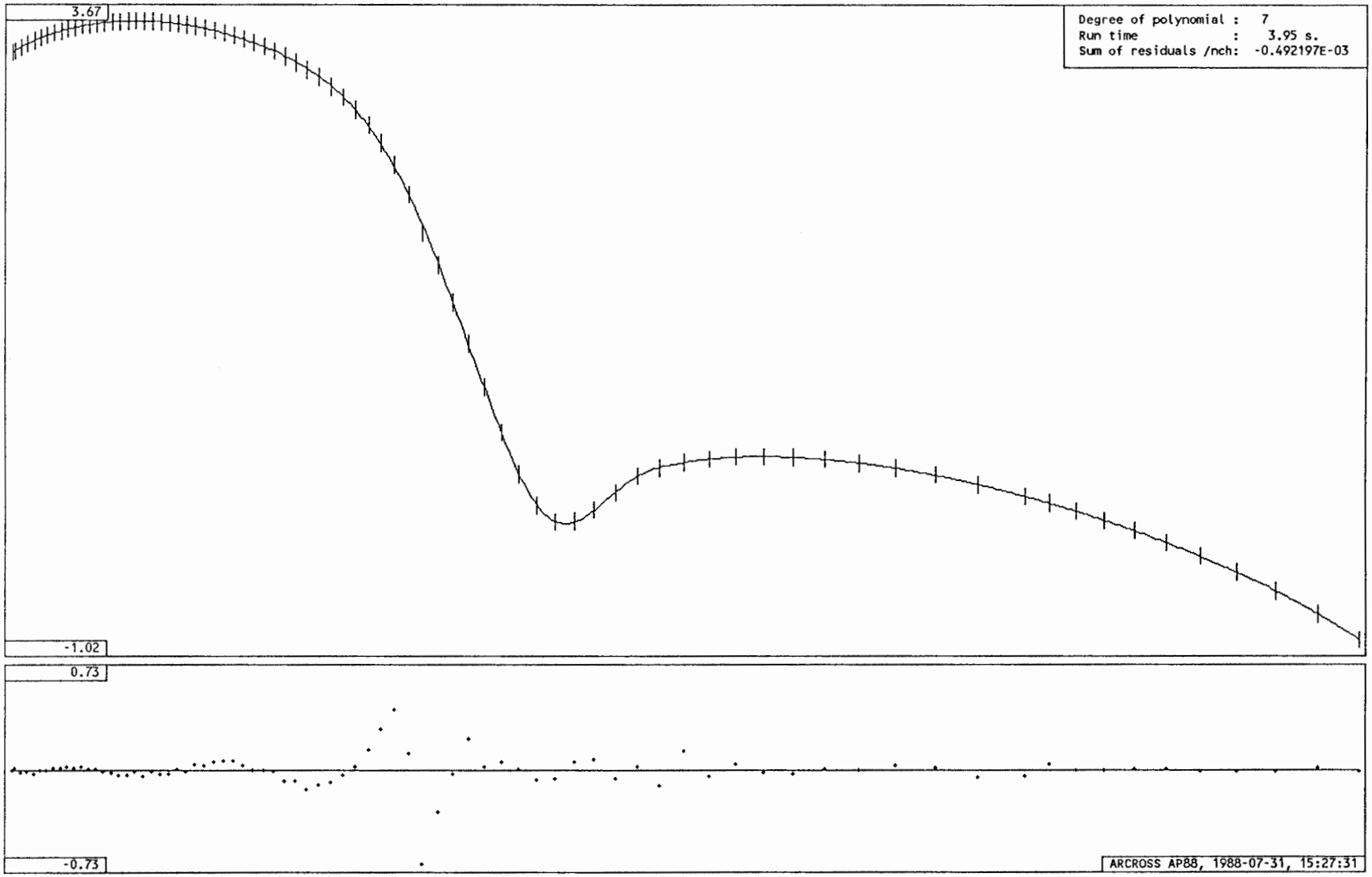
```
      END

      SUBROUTINE COPYVEC(A,B,N,FNZA,FNZB)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    A(1:*),B(1:*),N,FNZA,FNZB,I

      DO 10 I=FNZB,FNZA
        B(I)=0
10    CONTINUE

      DO 20 I=FNZA,N
        B(I)=A(I)
20    CONTINUE
      FNZB=FNZA

      RETURN
      END
```

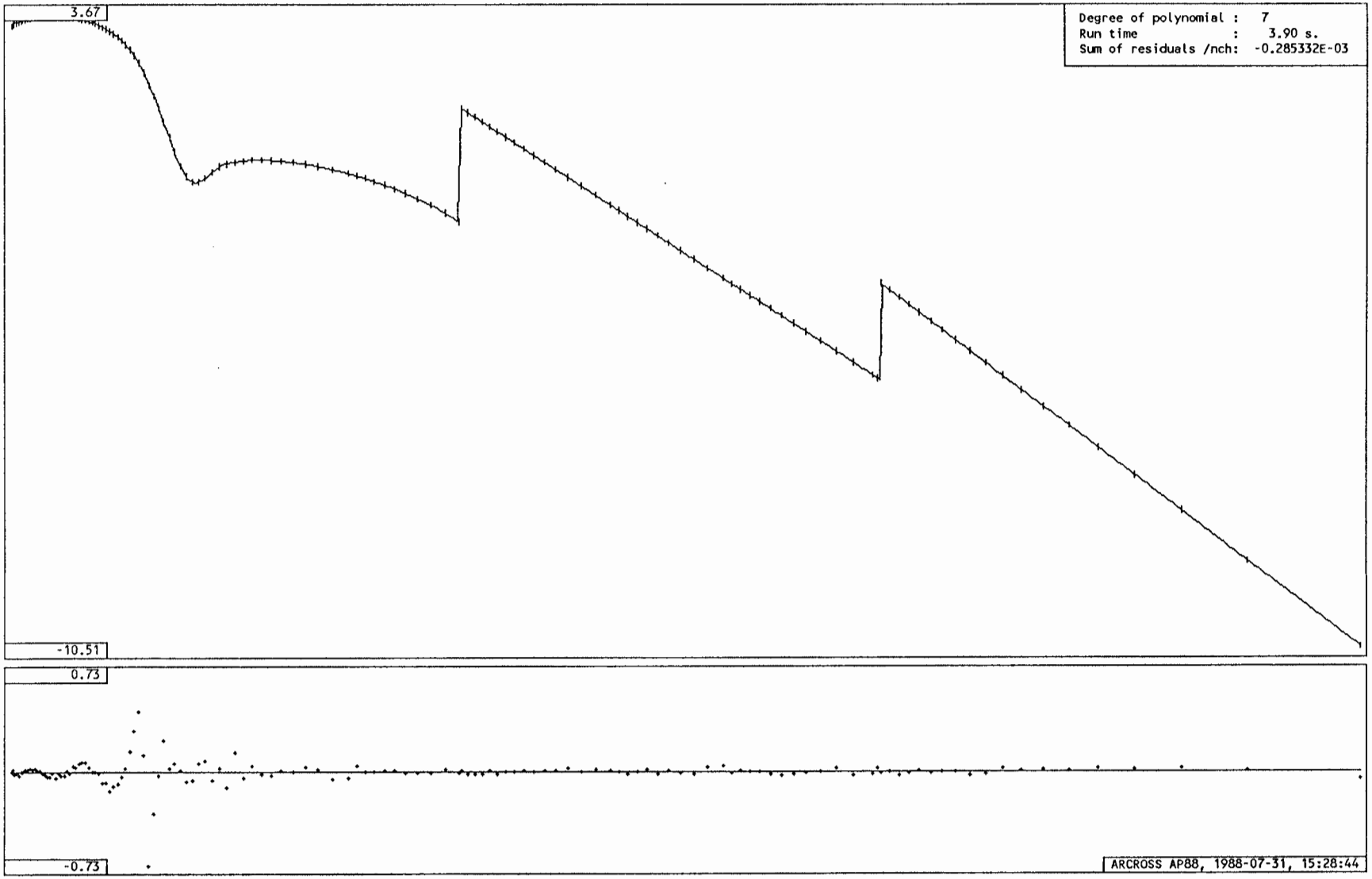
ARCROSS

Programmet anpassar polynom till mätdata från en mätning av fotojonisationstvärsnittet 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 övrlappa 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
```


3.67
-1.02
Degree of polynomial : 7
Run time : 3.95 s.
Sum of residuals /nch: -0.492197E-03
0.73
-0.73
ARCROSS AP88, 1988-07-31, 15:27:31


3.67
-10.51
Degree of polynomial : 7
Run time : 3.90 s.
Sum of residuals /nch: -0.285332E-03
0.73
-0.73
ARCROSS AP88, 1988-07-31, 15:28:44

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 residualsumman 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 $f(\underline{\theta};x)$. 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 förstagradspolynom.
206-256	Rutinen anpassar ett polynom av valbart gradtal till punkterna (x_i,y_i), i=FI,FI+1,...,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.

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: ARCROSS.FOR Options: /LBZ          08/02/88 14:13:32

      PROGRAM ARCROSS
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    N,L,MR
      PARAMETER  (N=138, L=16, MR=3*L+4)
      REAL*8     Y(1:N),X(1:N),THETAEST(1:MR),THETASLASK(1:L)
      REAL*8     RESIDUE(1:N),SIGMA(1:N),F,THETAERR(1:MR),ERRSLASK(1:L)
      REAL*8     TIME,SLASK,XM(1:5),SUMRESDIVNCH
      REAL       JX,JY,JXL,JYL,DJY
      INTEGER    R,K,I,NCH,POLDEG,LR,J
      INTEGER*2  HO,MI,SE,HN,YEAR,MONTH,DAY
      CHARACTER  SV*1,JETTXT*40
      LOGICAL    ERRORBAR
      EXTERNAL   F
      COMMON     XM
C
      NCH=138
      OPEN(UNIT=10,FILE='ARGON.DAT')
      SLASK=LOG((1.0+0.03)/(1.0-0.03))

      DO 10 I=1,138
        READ(10,*) X(I),Y(I)
        X(I)=LOG(X(I))
        Y(I)=LOG(Y(I))
        SIGMA(I)=SLASK
10    CONTINUE
      CLOSE(10)

20    WRITE(*,1000) 'Degree of polynomial (0..',L-1,'):'
1000  FORMAT(' ',A,I2,A)
      READ(*,*,ERR=20) POLDEG
      IF (POLDEG.LT.0.OR.POLDEG.GT.L-1) THEN
        GOTO 20
      ENDIF

      WRITE(*,*) 'Plot errorbars? (Y/*):'
      READ(*,1100) SV
1100  FORMAT(A)
      ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01

      LR=POLDEG+1
      CALL FITPOL(THETASLASK,ERRSLASK,X,Y,SIGMA,XM(3),1,51,POLDEG)
      DO 30 I=1,LR
        THETAEST(I+4)=THETASLASK(I)
        THETAERR(I+4)=ERRSLASK(I)
30    CONTINUE

      CALL FITPOL(THETASLASK,ERRSLASK,X,Y,SIGMA,XM(4),48,62,POLDEG)
      DO 40 I=1,LR
        THETAEST(LR+I+4)=THETASLASK(I)
        THETAERR(LR+I+4)=ERRSLASK(I)
40    CONTINUE

      CALL FITPOL(THETASLASK,ERRSLASK,X,Y,SIGMA,XM(5),59,80,POLDEG)
      DO 50 I=1,LR
        THETAEST(2*LR+I+4)=THETASLASK(I)
        THETAERR(2*LR+I+4)=ERRSLASK(I)
50    CONTINUE

      LR=2
      CALL FITPOL(THETASLASK,ERRSLASK,X,Y,SIGMA,XM(1),81,119,1)
      DO 60 I=1,LR
        THETAEST(I)=THETASLASK(I)
        THETAERR(I)=ERRSLASK(I)
60    CONTINUE

      CALL FITPOL(THETASLASK,ERRSLASK,X,Y,SIGMA,XM(2),120,138,1)
      DO 70 I=1,LR
        THETAEST(I+2)=THETASLASK(I)
        THETAERR(I+2)=ERRSLASK(I)
70    CONTINUE

      LR=3*(POLDEG+1)+4

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME

      WRITE(*,*) 'Plot only up to the first edge? (Y/*):'
      READ(*,1100) SV
      NCH=138
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        NCH=80
      ENDIF

      CALL SCRPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,POLDEG+1,ERRORBAR)

      SUMRESDIVNCH=0.0D0
      DO 80 I=1,NCH
        SUMRESDIVNCH=SUMRESDIVNCH + RESIDUE(I)
80    CONTINUE
      SUMRESDIVNCH=SUMRESDIVNCH/NCH

90    WRITE(*,*) 'Plot on Roland plotter? (Y/*):'
      READ(*,1100) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,POLDEG+1,ERRORBAR)
        GOTO 90
      ENDIF

100   WRITE(*,*) 'Plot on HP-Laserjet? (Y/*): '
      READ(*,1100) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CALL HPSTART('L')
        CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,POLDEG+1,ERRORBAR)
        JX=870
        JXL=1119-JX
        JYL=(3+1)*13
        JY=719-JYL
        CALL HPBOX(JX,JY,JXL,JYL)
        JX=876
        JY=719-16
        DJY=-13

```

```
      WRITE(JETTXT,1300) 'Degree of polynomial :',POLDEG
      CALL HPPRINT(JX,JY,40,JETTXT)
      JY=JY+DJY

      WRITE(JETTXT,1400) 'Run time             :',TIME,' s.'
      CALL HPPRINT(JX,JY,40,JETTXT)
      JY=JY+DJY

      WRITE(JETTXT,1700) 'Sum of residuals /nch:',SUMRESDIVNCH
      CALL HPPRINT(JX,JY,40,JETTXT)

      CALL GETTIM(HO,MI,SE,HN)
      CALL GETDAT(YEAR,MONTH,DAY)
      WRITE(JETTXT,1800)
     f 'ARCROSS AP88, ',YEAR,'-',MONTH,'-',DAY,', ',HO,':',MI,':',SE
1800  FORMAT(A,I4.4,5(A,I2.2))
      CALL HPTBOX(903.0,0.0,34,JETTXT)

      CALL HPEJECT
      GOTO 100
    ENDIF

    WRITE(*,1200)
1200 FORMAT(/)
    WRITE(*,1300) 'Degree of polynomial:',POLDEG
1300 FORMAT(' ',A,I4)
    WRITE(*,1400) 'Run time:',TIME,' s.'
1400 FORMAT(' ',A,F8.2,A)
    WRITE(*,1200)

    DO 110 R=1,LR
      WRITE(*,1500) 'THETA(',R,')=',THETAEST(R),' +/- ',THETAERR(R)
      IF (R.EQ.2.OR.R.EQ.4.OR.MOD(R-4,POLDEG+1).EQ.0) THEN
        WRITE(*,1600) ' '
      ENDIF
110  CONTINUE
1500 FORMAT(' ',A,I2,A,G15.5,A,G15.5)
1600 FORMAT(' ',A)

    WRITE(*,1700) 'Sum of residuals /nch:',SUMRESDIVNCH
1700 FORMAT(' ',A,G15.6)

    WRITE(*,1600) 'Quit? (Y/*):'
    READ(*,1100) SV
    IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
      STOP
    ELSE
      GOTO 20
    ENDIF
    END

    REAL*8 FUNCTION F(THETA,L,X)
C   Anders Persson, 1988

    IMPLICIT   LOGICAL (A-Z)
    INTEGER    L,I
    REAL*8     THETA(1:*),X,XM(1:5),SLASK,LOG41,LOG62,LOG248,LOG3260
    PARAMETER  (LOG41= 3.721588629, LOG62=  4.126973082)
    PARAMETER  (LOG248=5.513428746, LOG3260=8.089482474)
    COMMON     XM

    IF (X.GE.LOG248.AND.X.LT.LOG3260) THEN
      F=THETA(1)+THETA(2)*(X-XM(1))
    ELSEIF (X.GE.LOG3260) THEN
      F=THETA(3)+THETA(4)*(X-XM(2))
    ELSEIF (X.LT.LOG41) THEN
      SLASK=0.0D0
      DO 10 I=1,L
        SLASK=SLASK+THETA(I+4)*((X-XM(3))**(I-1))
10    CONTINUE
      F=SLASK
    ELSEIF (X.GE.LOG41.AND.X.LT.LOG62) THEN
      SLASK=0.0D0
      DO 20 I=1,L
        SLASK=SLASK+THETA(L+I+4)*((X-XM(4))**(I-1))
20    CONTINUE
      F=SLASK
    ELSEIF (X.GE.LOG62.AND.X.LT.LOG248) THEN
      SLASK=0.0D0
      DO 30 I=1,L
        SLASK=SLASK+THETA(2*L+I+4)*((X-XM(5))**(I-1))
30    CONTINUE
      F=SLASK
    ENDIF

    RETURN
    END

    SUBROUTINE FITPOL(THETA,ERR,X,Y,SIGMA,XM,FI,LA,POLDEG)
C   Anders Persson, 1988

    IMPLICIT   LOGICAL  (A-Z)
    INTEGER    L
    PARAMETER  ( L=16 )
    INTEGER    FI,LA,POLDEG,I,LR,R,K
    REAL*8     THETA(1:*),ERR(1:*),X(1:*),Y(1:*),SIGMA(1:*),XM
    REAL*8     DET,A(1:L,1:L),B(1:L),C(1:L)
    LOGICAL    SING

    XM=0.0D0
    DO 10 I=FI,LA
      XM=XM+X(I)
10  CONTINUE
    XM=XM/(LA-FI+1)
    LR=POLDEG+1

    CALL ZEROMAT(A,L,L,L,L)
    CALL ZEROMAT(B,L,1,L,1)
    CALL ZEROMAT(C,L,1,L,1)

    DO 40 I=FI,LA
      B(1)=B(1)+Y(I)/SIGMA(I)**2
      DO 20 R=2,LR
        B(R)=B(R)+Y(I)*(X(I)-XM)**(R-1)/SIGMA(I)**2
20    CONTINUE
      A(1,1)=A(1,1)+1.0D0/SIGMA(I)**2
      DO 30 R=1,LR
        IF (R+K.NE.2) THEN
          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,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:

$$f(\underline{\theta};x) = \theta_1 \cdot \exp(-\theta_2 \cdot x) + \theta_3 \tag{56}$$

Förstaderivator:

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

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

$$\frac{\partial f}{\partial \theta_3} = 1 \tag{59}$$

Andraderivator:

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

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

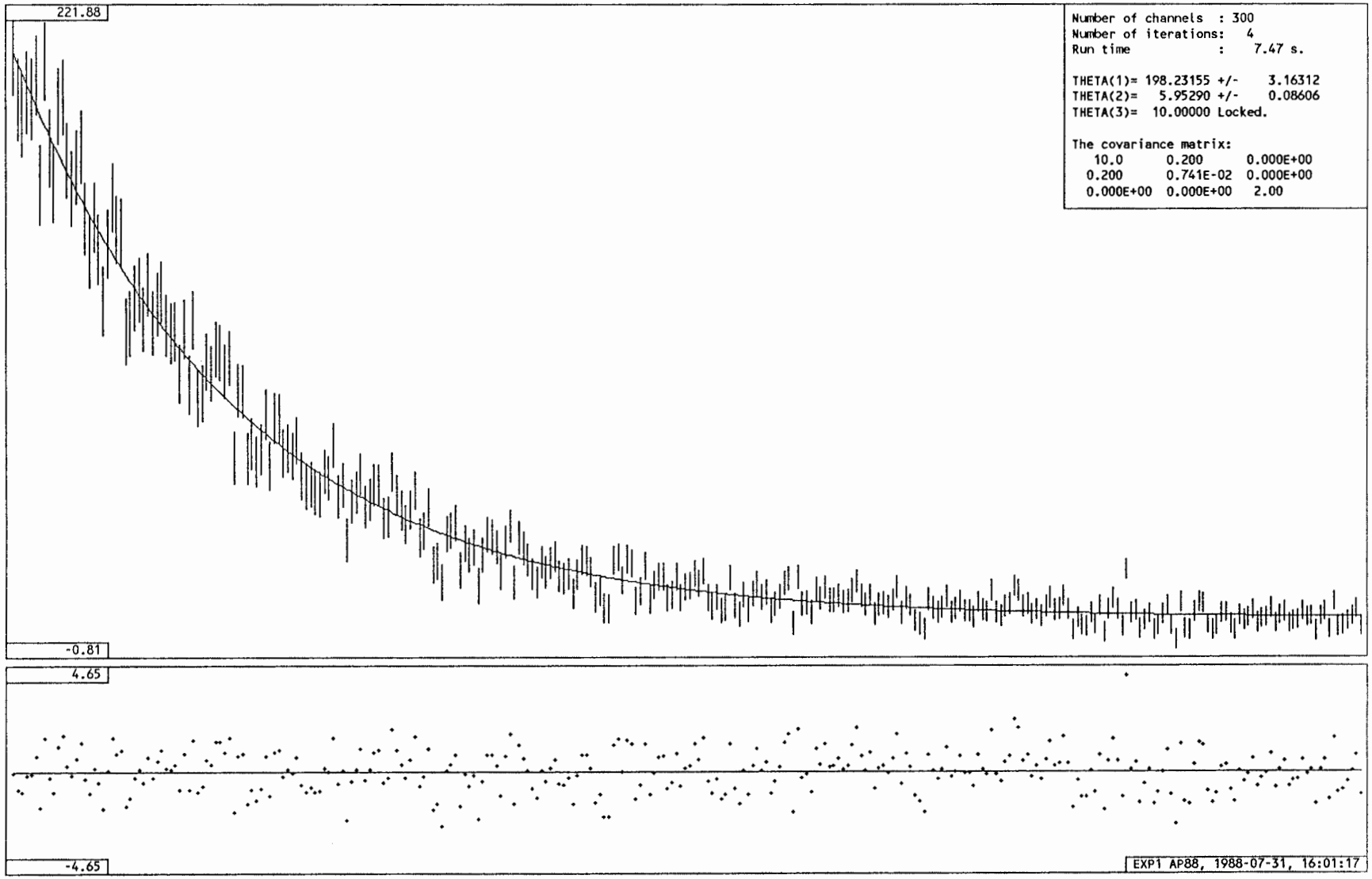
$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} = \frac{\partial^2 f}{\partial \theta_k \partial \theta_j} \tag{62}$$

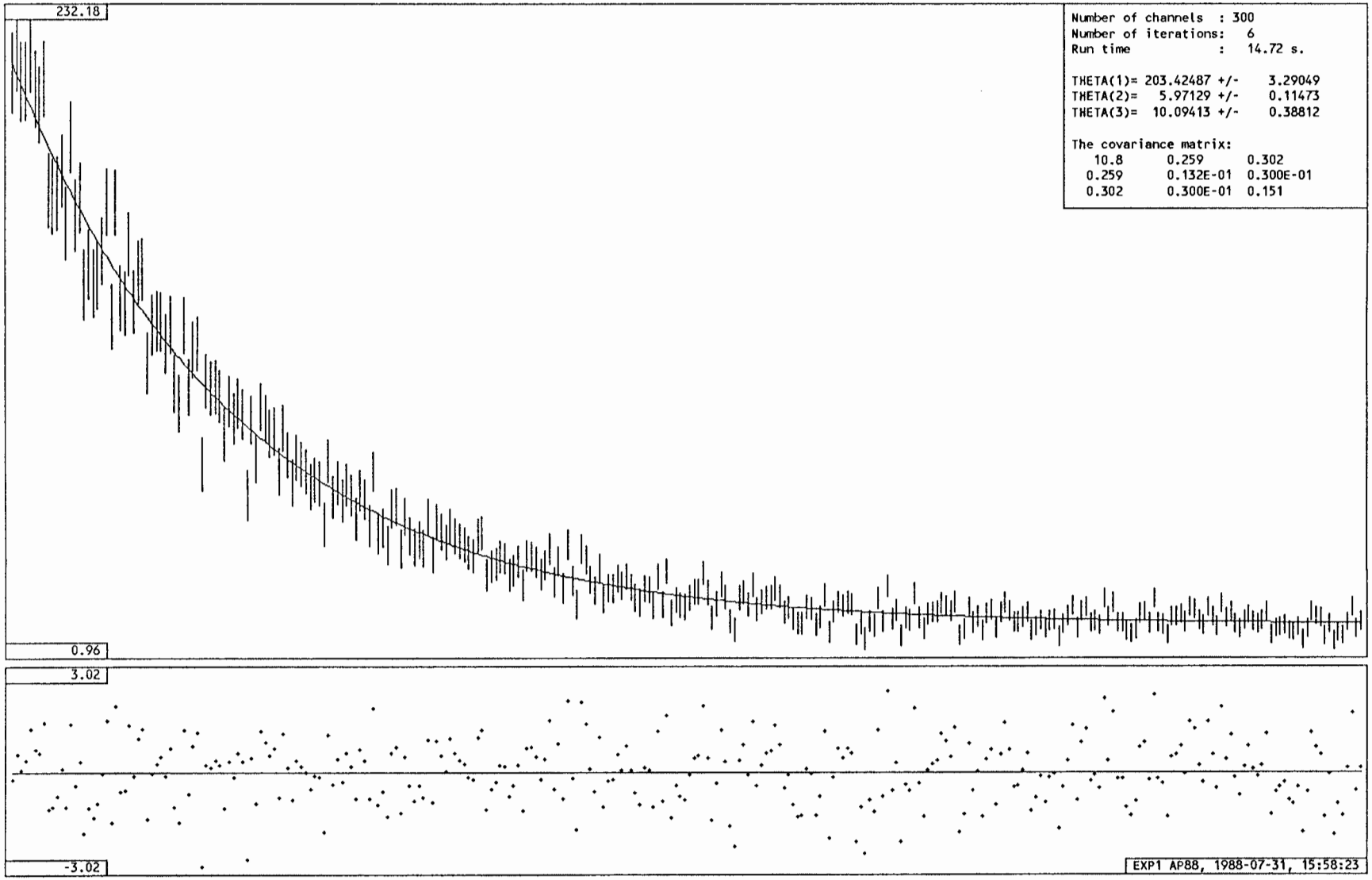
alla övriga andraderivator är noll.

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
```


221.88
-0.81
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
4.65
-4.65
EXP1 AP88, 1988-07-31, 16:01:17


232.18
0.96
3.02
-3.02
Number of channels : 300
Number of iterations: 6
Run time : 14.72 s.
THETA(1)= 203.42487 +/- 3.29049
THETA(2)= 5.97129 +/- 0.11473
THETA(3)= 10.09413 +/- 0.38812
The covariance matrix:
10.8 0.259 0.302
0.259 0.132E-01 0.300E-01
0.302 0.300E-01 0.151
EXP1 AP88, 1988-07-31, 15:58:23

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 kanalinnehållet vara Poissonfördelat (kan diskuteras se [7] sid. 181) men om kanalinnehållet 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 singulär 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 iterationsslingan. 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).

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: EXP1.FOR    Options: /LBZ         08/02/88 14:20:35

      PROGRAM EXP1
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    N,L,MAXITER
      PARAMETER  (N=1024, L=3, MAXITER=1000)
      REAL       Y(1:N),X(1:N),DX,DET,NORMAL,THETAEST(1:L)
      REAL       RESIDUE(1:N),SIGMA(1:N),F,THETA(1:L)
      REAL       GMATRIX(1:L,1:L),GVECTOR(1:L),DELTATHETA(1:L)
      REAL       BG,SX,SY,SXY,SX2,NOOFPOINTS,SLASK,DFDTH,D2FDTH2
      REAL       SL1,SL2,SL3,SL4,SL5,TIME
      REAL       JX,JY,JXL,JYL,DJY
      INTEGER    R,K,I,NCH,NOOFIT,J
      INTEGER*2  HO,MI,SE,HN,ANS,KBDCHK,KBDINC
      INTEGER*2  YEAR,MONTH,DAY
      CHARACTER  SV*1,JETTXT*40
      LOGICAL    SING,ERRORBAR,LINEARIZED,LOCKED(1:L)
      EXTERNAL   F

10    WRITE(*,1000) 'No of channels (',L,'..',N,'):'
1000  FORMAT(' ',A,I2,A,I4,A)
      READ(*,*,ERR=10) NCH
      IF (NCH.LT.L.OR.NCH.GT.N) THEN
        GOTO 10
      ENDIF

      WRITE(*,*) 'Plot errorbars? (Y/*):'
      READ(*,1100) SV
1100  FORMAT(A)
      ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'

      WRITE(*,*) 'Parameter values.'
      WRITE(*,*) 'Theta 1 (Intensity):'
      READ(*,*) THETA(1)

      WRITE(*,*) 'Theta 2 (Decay constant):'
      READ(*,*) THETA(2)

      WRITE(*,*) 'Theta 3 (Background):'
      READ(*,*) THETA(3)

      DX=1.0/(NCH-1)
      DO 20 I=1,NCH
        X(I)=(I-1)*DX
        SLASK=F(THETA,L,X(I))
        Y(I)=NORMAL(SLASK,SQRT(SLASK))
        SIGMA(I)=SQRT(SLASK)
20    CONTINUE

      DO 30 I=1,L
        LOCKED(I)=.FALSE.
30    CONTINUE

      WRITE(*,*) 'Calculate start values from data? (Y/*):'
      READ(*,1100) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        R =NCH/10+2
        K =NCH-R+1
        BG=0.0

        DO 40 I=K,NCH
          BG=BG+Y(I)
40      CONTINUE

        BG        =BG/R
        SX        =0.0
        SY        =0.0
        SXY       =0.0
        SX2       =0.0
        NOOFPOINTS=0.0

        DO 50 I=1,NCH
          IF (Y(I)-BG.LE.(Y(1)-BG)*0.1) THEN
            GOTO 60
          ENDIF
          NOOFPOINTS=NOOFPOINTS+1.0
          SX=SX+X(I)
          SY=SY+LOG(Y(I))
          SXY=SXY+X(I)*LOG(Y(I))
          SX2=SX2+X(I)**2
50      CONTINUE
60      CONTINUE

        THETAEST(3)=BG
        THETAEST(2)=(NOOFPOINTS*SX2-SX**2)/(SX*SY-NOOFPOINTS*SXY)
        THETAEST(1)=(SY*THETAEST(2)+SX)/(THETAEST(2)*NOOFPOINTS)
        THETAEST(1)=EXP(THETAEST(1))
        THETAEST(2)=1.0/THETAEST(2)
      ELSE
        WRITE(*,*) 'Enter start values for theta.'

        WRITE(*,*) 'Theta(1)='
        READ(*,*) THETAEST(1)
        WRITE(*,*) 'Lock parameter Theta(1)? (Y/*):'
        READ(*,1100) SV
        LOCKED(1)=SV.EQ.'Y'.OR.SV.EQ.'y'

        WRITE(*,*) 'Theta(2)='
        READ(*,*) THETAEST(2)
        WRITE(*,*) 'Lock parameter Theta(2)? (Y/*):'
        READ(*,1100) SV
        LOCKED(2)=SV.EQ.'Y'.OR.SV.EQ.'y'

        WRITE(*,*) 'Theta(3)='
        READ(*,*) THETAEST(3)
        WRITE(*,*) 'Lock parameter Theta(3)? (Y/*):'
        READ(*,1100) SV
        LOCKED(3)=SV.EQ.'Y'.OR.SV.EQ.'y'

      ENDIF

      WRITE(*,*) ' '
      WRITE(*,*) 'Press Q to quit iteration.'
      WRITE(*,*) ' '

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01
```

```
118
119       WRITE(*,1200) 'It:',0,',TH1:',THETAEST(1),',TH2:',THETAEST(2),
120      f     ',TH3:',THETAEST(3)
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(I)-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,I3,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 SCRPLOT(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
```

```
      CALL HPPRINT(JX,JY,40,JETTXT)
      JY=JY+DJY

      WRITE(JETTXT,1500) 'Run time            :',TIME,' s.'
      CALL HPPRINT(JX,JY,40,JETTXT)
      JY=JY+DJY

      WRITE(JETTXT,1100) ' '
      CALL HPPRINT(JX,JY,40,JETTXT)
      JY=JY+DJY

      DO 170 R=1,L
        IF (.NOT.LOCKED(R)) THEN
          WRITE(JETTXT,1600) 'THETA(',R,')=',THETAEST(R),' +/- ',
     f                    SQRT(ABS(GMATRIX(R,R)))
        ELSE
          WRITE(JETTXT,1600) 'THETA(',R,')=',THETAEST(R),' Locked.'
        ENDIF
        CALL HPPRINT(JX,JY,40,JETTXT)
        JY=JY+DJY
170   CONTINUE

      WRITE(JETTXT,1100) ' '
      CALL HPPRINT(JX,JY,40,JETTXT)
      JY=JY+DJY

      WRITE(JETTXT,1100) ' The covariance matrix:'
      CALL HPPRINT(JX,JY,40,JETTXT)
      JY=JY+DJY

      DO 180 K=1,L
        WRITE(JETTXT,1800) (GMATRIX(R,K), R=1,L)
        CALL HPPRINT(JX,JY,40,JETTXT)
        JY=JY+DJY
180   CONTINUE

      CALL GETTIM(HO,MI,SE,HN)
      CALL GETDAT(YEAR,MONTH,DAY)
      WRITE(JETTXT,2000)
     f  'EXP1 AP88, ',YEAR,'-',MONTH,'-',DAY,', ',HO,':',MI,':',SE
2000  FORMAT(A,I4.4,5(A,I2.2))
      CALL HPTBOX(921.0,0.0,31,JETTXT)

      CALL HPEJECT
      GOTO 160
    ENDIF

    WRITE(*,1300)
1300 FORMAT(/)
    WRITE(*,1400) 'Number of channels:',NCH
    WRITE(*,1400) 'Number of iterations:',NOOFIT
1400 FORMAT(' ',A,I4)
    WRITE(*,1500) 'Run time:',TIME,' s.'
1500 FORMAT(' ',A,F8.2,A)
    WRITE(*,1300)

    DO 190 R=1,L
      IF (.NOT.LOCKED(R)) THEN
        WRITE(*,1600) 'THETA(',R,')=',THETAEST(R),' +/- ',
     f                SQRT(ABS(GMATRIX(R,R)))
      ELSE
        WRITE(*,1600) 'THETA(',R,')=',THETAEST(R),' Locked.'
      ENDIF
1600  FORMAT(' ',A,I1,A,F10.5,A,F10.5)
190 CONTINUE

    WRITE(*,1700) 'The covariance matrix for theta:'
1700 FORMAT(/' ',A)
    DO 200 K=1,L
      WRITE(*,1800) (GMATRIX(R,K), R=1,L)
1800  FORMAT(' ',10G11.3)
200 CONTINUE

210 WRITE(*,1900) 'New calculation? (Y/*):'
1900 FORMAT(/' ',A)
    READ(*,1100) SV
    IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
      GOTO 10
    ELSE
      STOP
    ENDIF
    END

    REAL FUNCTION F(THETA,L,X)
C   Anders Persson, 1988

    IMPLICIT   LOGICAL (A-Z)
    INTEGER    L
    REAL       THETA(1:L),X

    F=THETA(1)*EXP(-THETA(2)*X)+THETA(3)

    RETURN
    END

    REAL FUNCTION DFDTH(J,THETA,L,X)
C   Anders Persson, 1988

    IMPLICIT   LOGICAL (A-Z)
    INTEGER    J,L
    REAL       THETA(1:L),X

    IF (J.EQ.1) THEN
      DFDTH=EXP(-THETA(2)*X)
    ELSEIF (J.EQ.2) THEN
      DFDTH=-X*THETA(1)*EXP(-THETA(2)*X)
    ELSE
      DFDTH=1.0
    ENDIF

    RETURN
    END

    REAL FUNCTION D2FDTH2(I,J,THETA,L,X)
C   Anders Persson, 1988

    IMPLICIT   LOGICAL (A-Z)
    INTEGER    I,J,L
    REAL       THETA(1:L),X
```

```
      IF (I+J.EQ.3) THEN
        D2FDTH2=-X*EXP(-THETA(2)*X)
      ELSEIF (I.EQ.2.AND.J.EQ.2) THEN
        D2FDTH2=X*X*THETA(1)*EXP(-THETA(2)*X)
      ELSE
        D2FDTH2=0.0
      ENDIF

      RETURN
      END
```

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 perametrar 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:

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

Förstaderivator:

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

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

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

Andraderivator:

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

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

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} = \frac{\partial^2 f}{\partial \theta_k \partial \theta_j} \tag{69}$$

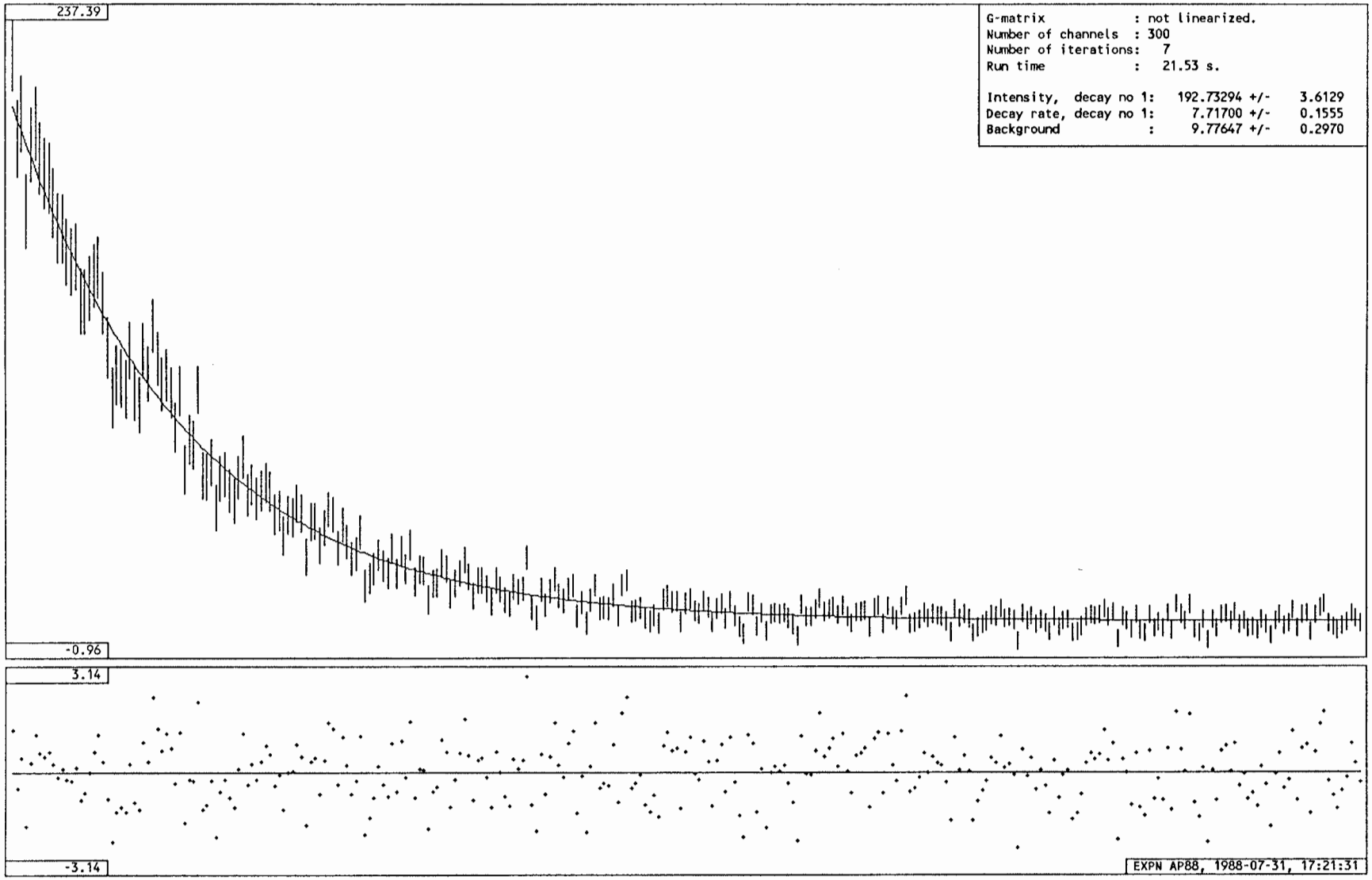
alla övriga andraderivator är noll.

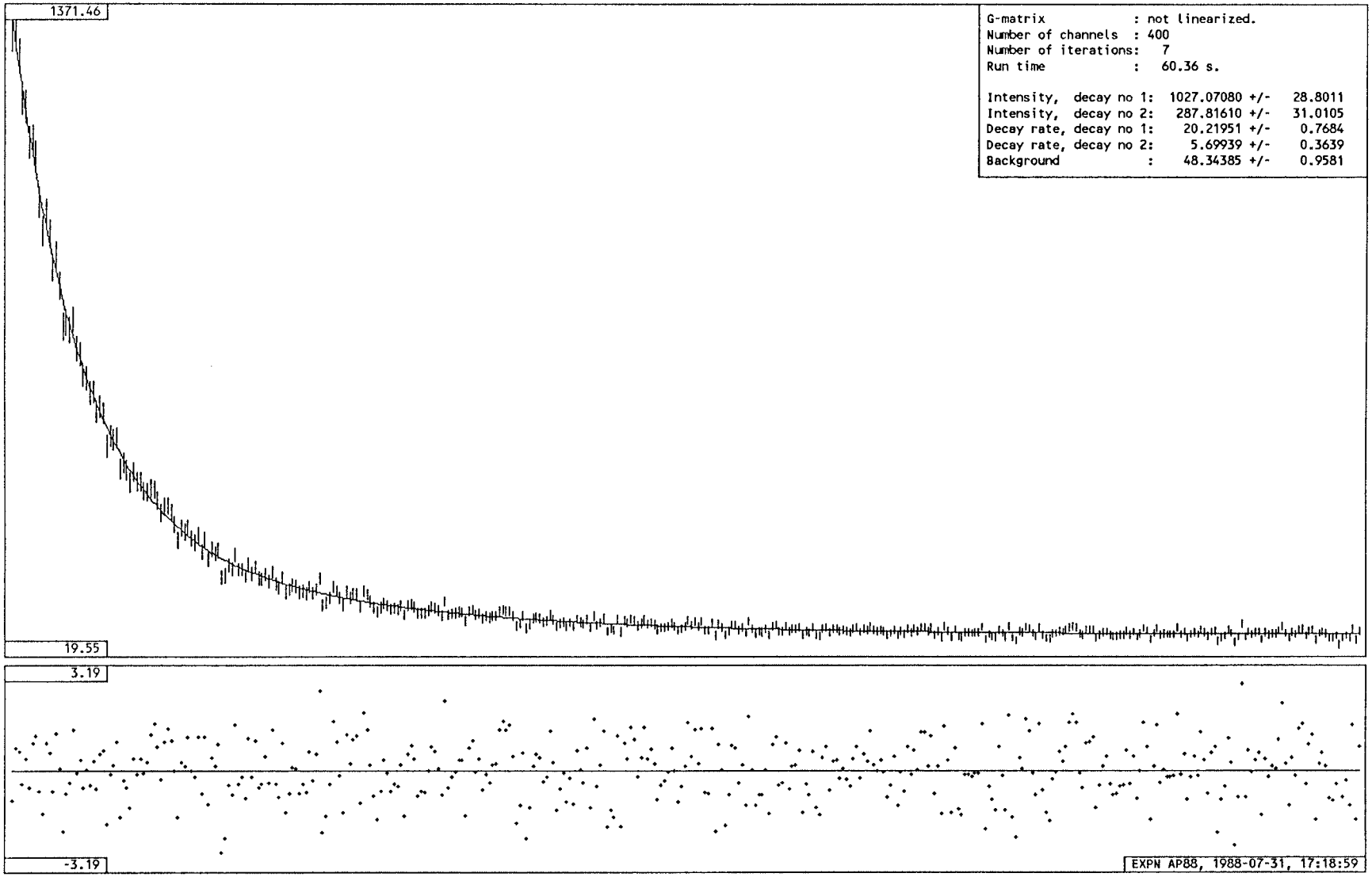
Utskrift från testkörning av programmet EXPN:

```
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
```


237.39
-0.96
G-matrix : not linearized.
Number of channels : 300
Number of iterations: 7
Run time : 21.53 s.
Intensity, decay no 1: 192.73294 +/- 3.6129
Decay rate, decay no 1: 7.71700 +/- 0.1555
Background : 9.77647 +/- 0.2970
3.14
-3.14
EXPN AP88, 1988-07-31, 17:21:31


1371.46
19.55
3.19
-3.19
EXPN AP88, 1988-07-31, 17:18:59
G-matrix : not linearized.
Number of channels : 400
Number of iterations: 7
Run time : 60.36 s.
Intensity, decay no 1: 1027.07080 +/- 28.8011
Intensity, decay no 2: 287.81610 +/- 31.0105
Decay rate, decay no 1: 20.21951 +/- 0.7684
Decay rate, decay no 2: 5.69939 +/- 0.3639
Background : 48.34385 +/- 0.9581

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 exponentialer 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, parameterns värde hamnar på min eller maxgränsen.

247 Beräkna den nya parametervektorn enl. (39).

249-254 Beräkna steglängden, $(\Sigma|\Delta\theta_i/\theta_i|)$, ett mått på hur nära minpunkten programmet nått.

256-261 Skriv ut de nya parametervärdena på skärmen.

263-266 Om steglängden är mindre än $0.5\cdot10^{-4}$ och G-matrisen ej är lineariserad är iterationen klar och slingan bryts.

267-269 Om steget är mindre än $0.5\cdot10^{-4}$ 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 $0.5\cdot10^{-4}$ är iterationen klar och slingan bryts.

277-279 Om steglängden är mindre än $0.5\cdot10^{-2}$ 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 iterationsslingan. Om programmet hamnar på rad 290 har maximalt antal iterationer utförts utan att minpunkten uppnåtts.

293 I iterationsslingan multipliceras G^{-1}-matrisen med -1 inför beräkningen av $\underline{\Delta\theta}$ (43), den måste nu multipliceras med -2 för att vara en skattning av kovariansmatrisen för $\underline{\theta}$ (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).

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: EXPN.FOR    Options: /LBZ         08/02/88 14:22:02

      PROGRAM EXPN
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    N,L,MAXITER,MAXNOOFDECAYS
      PARAMETER  (N=1024, MAXITER=1000, MAXNOOFDECAYS=3)
      PARAMETER  (L=2*MAXNOOFDECAYS+1)
      REAL       Y(1:N),X(1:N),DX,DET,NORMAL,THETAEST(1:L)
      REAL       RESIDUE(1:N),SIGMA(1:N),F,THETA(1:L)
      REAL       GMATRIX(1:L,1:L),GVECTOR(1:L),DELTATHETA(1:L)
      REAL       SLASK,DFDTH,D2FDTH2,THETAMIN(1:L),THETAMAX(1:L)
      REAL       SL1,SL2,SL3,SL4,SL5,TIME,STEP
      REAL       JX,JY,JXL,JYL,DJY
      INTEGER    R,K,I,NCH,NOOFIT,J,NOOFDECAYS,LR
      INTEGER*2  HO,MI,SE,HN,ANS,KBDCHK,KBDINC
      INTEGER*2  YEAR,MONTH,DAY
      CHARACTER  SV*1,LOCKEDORFREE(1:L)*6,JETTXT*50
      LOGICAL    SING,ERRORBAR,LINEARIZED,LOCKED(1:L),ADDNOISE
      LOGICAL    LOCKTOLINEARIZED,LOCKWEIGHT
      EXTERNAL   F

10    WRITE(*,1000) 'No of decays (1..',MAXNOOFDECAYS,'):'
1000  FORMAT(' ',A,I2,A)
      READ(*,*,ERR=10) NOOFDECAYS
      IF (NOOFDECAYS.LT.1.OR.NOOFDECAYS.GT.MAXNOOFDECAYS) THEN
        GOTO 10
      ENDIF
      LR=2*NOOFDECAYS+1

20    WRITE(*,1100) 'No of channels (',LR,'..',N,'):'
1100  FORMAT(' ',A,I2,A,I4,A)
      READ(*,*,ERR=20) NCH
      IF (NCH.LT.L.OR.NCH.GT.N) THEN
        GOTO 20
      ENDIF

      DO 50 I=1,NOOFDECAYS
30      WRITE(*,1200) 'Intensity, decay no ',I,': '
1200    FORMAT(' ',A,I1,A)
        READ(*,*,ERR=30) THETA(I)
        IF (THETA(I).LE.0.0) THEN
          GOTO 30
        ENDIF
40      WRITE(*,1200) 'Decay rate, decay no ',I,': '
        READ(*,*,ERR=40) THETA(I+NOOFDECAYS)
        IF (THETA(I+NOOFDECAYS).LE.0.0) THEN
          GOTO 40
        ENDIF
50    CONTINUE
60    WRITE(*,*) 'Background: '
      READ(*,*,ERR=60) THETA(LR)

      WRITE(*,*) 'Add noise to data? (Y/*): '
      READ(*,1300) SV
1300  FORMAT(A)
      ADDNOISE=SV.EQ.'Y'.OR.SV.EQ.'y'
      ERRORBAR=ADDNOISE
      IF (ADDNOISE) THEN
        WRITE(*,*) 'Plot errorbars? (Y/*): '
        READ(*,1300) SV
        ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'
      ENDIF

      DO 70 I=1,LR
        LOCKED(I)=.FALSE.
        LOCKEDORFREE(I)='Free  '
70    CONTINUE

      DX=1.0/(NCH-1)
      DO 80 I=1,NCH
        X(I)=(I-1)*DX
        SLASK=F(THETA,NOOFDECAYS,X(I))
        IF (ADDNOISE) THEN
          Y(I)=NORMAL(SLASK,SQRT(SLASK))
          SIGMA(I)=SQRT(SLASK)
        ELSE
          Y(I)=SLASK
          SIGMA(I)=0.0
        ENDIF
80    CONTINUE

      DO 90 I=1,LR
        THETAEST(I)=THETA(I)
90    CONTINUE

      LOCKTOLINEARIZED=.FALSE.

100   CALL GMODE
      CALL TMODE

      WRITE(*,1400) 'Toggle Free/Locked with <shift>+Function key.'
1400  FORMAT(//////,T10,A,/)
      DO 110 I=1,NOOFDECAYS
        WRITE(*,1500) I,I,THETAEST(I),LOCKEDORFREE(I)
1500    FORMAT(T10,'F',I1,' : Intensity , decay no ',
     f         I1,T45,F12.4,',',T60,A)
110   CONTINUE
      DO 120 I=1,NOOFDECAYS
        WRITE(*,1600) I+NOOFDECAYS,I,THETAEST(I+NOOFDECAYS),
     f                LOCKEDORFREE(I+NOOFDECAYS)
1600    FORMAT(T10,'F',I1,' : Decay rate, decay no ',
     f         I1,T45,F12.4,',',T60,A)
120   CONTINUE
      WRITE(*,1700) LR,THETAEST(LR),LOCKEDORFREE(LR)
1700  FORMAT(T10,'F',I1,' : Background ',T45,F12.4,',',T60,A)
      WRITE(*,1800) 'F8 : Preview.'
1800  FORMAT(T10,A)
      WRITE(*,1800) 'F9 : Calculate.'
      IF (LOCKTOLINEARIZED) THEN
        WRITE(*,1800) 'F10: G-matrix = linearized.'
      ELSE
        WRITE(*,1800) 'F10: G-matrix = not linearized.'
      ENDIF
      WRITE(*,1800) ' '

130   ANS=KBDCHK()
      IF (ANS.NE.0) THEN
```

```
        GOTO 130
      ENDIF
      ANS=KBDINC()

      IF (ANS.GE.1084.AND.ANS.LE.1083+LR) THEN
        I=ANS-1083
        LOCKED(I)=.NOT.LOCKED(I)
        LOCKEDORFREE(I)='Free  '
        IF (LOCKED(I)) THEN
          LOCKEDORFREE(I)='Locked'
        ENDIF
        GOTO 100
      ELSEIF (ANS.EQ.1066) THEN
        CALL SCRPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
     f              THETAEST,NOOFDECAYS,ERRORBAR)
        GOTO 100
      ELSEIF (ANS.EQ.1067) THEN
C       Calculate.
        GOTO 170
      ELSEIF (ANS.EQ.1068) THEN
        LOCKTOLINEARIZED=.NOT.LOCKTOLINEARIZED
        GOTO 100
      ELSEIF (ANS.GE.1059.AND.ANS.LE.1058+LR) THEN
        I=ANS-1058
        IF (I.LE.NOOFDECAYS) THEN
140       WRITE(*,1900) 'New intensity, decay no ',I,': '
1900      FORMAT(T10,A,I1,A)
          READ(*,*,ERR=140) THETAEST(I)
          IF (THETAEST(I).LE.0.0) THEN
            GOTO 140
          ENDIF
          GOTO 100
        ELSEIF (I.GT.NOOFDECAYS.AND.I.LT.LR) THEN
150       WRITE(*,1900) 'New decay rate, decay no ',I-NOOFDECAYS,': '
          READ(*,*,ERR=150) THETAEST(I)
          IF (THETAEST(I).LE.0.0) THEN
            GOTO 150
          ENDIF
          GOTO 100
        ELSEIF (I.EQ.LR) THEN
160       WRITE(*,1900) 'New background: '
          READ(*,*,ERR=160) THETAEST(LR)
          GOTO 100
        ENDIF
        GOTO 130
      ENDIF
      GOTO 130

C     *** Calculation ***
170   CONTINUE
      WRITE(*,*) ' '
      WRITE(*,*) 'Press Q to quit iteration.'
      WRITE(*,*) ' '

      DO 180 I=1,2*NOOFDECAYS
        THETAMIN(I)=THETAEST(I)/5.0
        THETAMAX(I)=THETAEST(I)*5.0
180   CONTINUE
      THETAMAX(LR)= THETAEST(LR)*5.0
      THETAMIN(LR)=-THETAMAX(LR)

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01

      WRITE(*,2000) 'It:',0
2000  FORMAT(' ',49('-'),/,' ',A,I4,A,G13.6)
      DO 190 R=1,NOOFDECAYS
        WRITE(*,2100) 'Intensity(',R,') :',THETAEST(R),
     f                'Decay rate(',R,'):',THETAEST(R+NOOFDECAYS)
2100    FORMAT(' ',A,I1,A,G12.5,A,I1,A,G12.5)
190   CONTINUE
      WRITE(*,2200) 'Background   :',THETAEST(LR)
2200  FORMAT(' ',A,G12.5)

      LINEARIZED=.TRUE.
      LOCKWEIGHT=.TRUE.

      DO 270 J=1,MAXITER
        CALL ZEROMAT(GMATRIX,LR,LR,L,L)
        CALL ZEROMAT(GVECTOR,LR,1,L,1)
        DO 220 I=1,NCH
          IF (LOCKWEIGHT) THEN
            SL1=-2.0/MAX(Y(I),1.0)
          ELSE
            SL1=-2.0/F(THETAEST,NOOFDECAYS,X(I))
          ENDIF
          SL2=Y(I)-F(THETAEST,NOOFDECAYS,X(I))
          DO 210 R=1,LR
            IF (.NOT.LOCKED(R)) THEN
              SL3=DFDTH(R,THETAEST,NOOFDECAYS,X(I))
              GVECTOR(R)=GVECTOR(R)+SL1*SL2*SL3
              DO 200 K=R,LR
                IF (.NOT.LOCKED(K)) THEN
                  SL4=DFDTH(K,THETAEST,NOOFDECAYS,X(I))
                  GMATRIX(R,K)=GMATRIX(R,K)+SL1*(-SL3*SL4)
                  IF (.NOT.LINEARIZED) THEN
                    SL5=D2FDTH2(R,K,THETAEST,NOOFDECAYS,X(I))
                    GMATRIX(R,K)=GMATRIX(R,K)+SL1*SL2*SL5
                  ENDIF
                  GMATRIX(K,R)=GMATRIX(R,K)
                ENDIF
200           CONTINUE
            ENDIF
210       CONTINUE
220     CONTINUE

        DO 230 R=1,LR
          IF (LOCKED(R)) THEN
            GMATRIX(R,R)=1.0
          ENDIF
230     CONTINUE

        CALL INVMAT(GMATRIX,LR,DET,SING,L,L)
        IF (SING) THEN
          WRITE(*,*) 'G-matrix singular.'
          GOTO 360
        ENDIF

        CALL MULTCMAT(GMATRIX,-1.0,LR,LR,L,L)
        CALL MULTMAT(GMATRIX,GVECTOR,DELTATHETA,LR,LR,1,L,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 SCRPLOT(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.LOCKED(I)).AND.ADDNOISE.AND.
356      f                    (.NOT.LOCKTOLINEARIZED)) THEN
357             WRITE(JETTXT,2600) 'Intensity,  decay no ',I,':',
```

```
     f                    THETAEST(I),' +/- ',SQRT(ABS(GMATRIX(I,I)))
          ELSEIF (LOCKED(I)) THEN
            WRITE(JETTXT,2600) 'Intensity,  decay no ',I,':',
     f                    THETAEST(I),' Locked'
          ELSEIF ((.NOT.LOCKED(I)).AND.
     f           ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED)) THEN
            WRITE(JETTXT,2600) 'Intensity,  decay no ',I,':',
     f                    THETAEST(I)
          ENDIF
          CALL HPPRINT(JX,JY,50,JETTXT)
          JY=JY+DJY
 320    CONTINUE

        DO 330 I=1,NOOFDECAYS
          IF ((.NOT.LOCKED(I+NOOFDECAYS)).AND.ADDNOISE.AND.
     f                             (.NOT.LOCKTOLINEARIZED)) THEN
            WRITE(JETTXT,2600) 'Decay rate, decay no ',I,':',
     f                   THETAEST(I+NOOFDECAYS),' +/- ',
     f                   SQRT(ABS(GMATRIX(I+NOOFDECAYS,I+NOOFDECAYS)))
          ELSEIF (LOCKED(I+NOOFDECAYS)) THEN
            WRITE(JETTXT,2600) 'Decay rate, decay no ',I,':',
     f                    THETAEST(I+NOOFDECAYS),' Locked'
          ELSEIF ((.NOT.LOCKED(I+NOOFDECAYS)).AND.
     f           ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED)) THEN
            WRITE(JETTXT,2600) 'Decay rate, decay no ',I,':',
     f                    THETAEST(I+NOOFDECAYS)
          ENDIF
          CALL HPPRINT(JX,JY,50,JETTXT)
          JY=JY+DJY
 330    CONTINUE

        IF ((.NOT.LOCKED(LR)).AND.ADDNOISE.AND.
     f                  (.NOT.LOCKTOLINEARIZED)) THEN
          WRITE(JETTXT,2700) 'Background            :',
     f                  THETAEST(LR),' +/- ',SQRT(ABS(GMATRIX(LR,LR)))
        ELSEIF (LOCKED(LR)) THEN
          WRITE(JETTXT,2700) 'Background            :',
     f                  THETAEST(LR),' Locked'
        ELSEIF ((.NOT.LOCKED(LR)).AND.
     f         ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED)) THEN
          WRITE(JETTXT,2700) 'Background            :',
     f                   THETAEST(LR)
        ENDIF
        CALL HPPRINT(JX,JY,50,JETTXT)

        CALL GETTIM(HO,MI,SE,HN)
        CALL GETDAT(YEAR,MONTH,DAY)
        WRITE(JETTXT,2900)
     f  'EXPN AP88, ',YEAR,'-',MONTH,'-',DAY,', ',HO,':',MI,':',SE
 2900   FORMAT(A,I4.4,5(A,I2.2))
        CALL HPTBOX(921.0,0.0,31,JETTXT)

        CALL HPEJECT
        GOTO 310
      ENDIF

      WRITE(*,2300)
 2300 FORMAT(/)

      IF (LOCKTOLINEARIZED) THEN
        WRITE(*,2400) 'G-matrix             : linearized.'
      ELSE
        WRITE(*,2400) 'G-matrix             : not linearized.'
      ENDIF

      WRITE(*,2400)   'Number of points     :',NCH

      WRITE(*,2400)   'Number of iterations :',NOOFIT
 2400 FORMAT(' ',A,I4)

      WRITE(*,2500) 'Run time:',TIME,' s.'
 2500 FORMAT(' ',A,F8.2,A)
      WRITE(*,2300)

      DO 340 I=1,NOOFDECAYS
        IF ((.NOT.LOCKED(I)).AND.ADDNOISE.AND.
     f                  (.NOT.LOCKTOLINEARIZED)) THEN
          WRITE(*,2600) 'Intensity, decay no ',I,' :',
     f                   THETAEST(I),' +/- ',SQRT(ABS(GMATRIX(I,I)))
 2600     FORMAT(' ',A,I1,A,F12.5,A,F9.4)
        ELSEIF (LOCKED(I)) THEN
          WRITE(*,2600) 'Intensity, decay no ',I,' :',
     f                   THETAEST(I),' Locked'
        ELSEIF ((.NOT.LOCKED(I)).AND.
     f         ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED)) THEN
          WRITE(*,2600) 'Intensity, decay no ',I,' :',
     f                   THETAEST(I)
        ENDIF
 340    CONTINUE

        DO 350 I=1,NOOFDECAYS
          IF ((.NOT.LOCKED(I+NOOFDECAYS)).AND.ADDNOISE.AND.
     f                             (.NOT.LOCKTOLINEARIZED)) THEN
            WRITE(*,2600) 'Decay rate, decay no ',I,':',
     f                   THETAEST(I+NOOFDECAYS),' +/- ',
     f                   SQRT(ABS(GMATRIX(I+NOOFDECAYS,I+NOOFDECAYS)))
          ELSEIF (LOCKED(I+NOOFDECAYS)) THEN
            WRITE(*,2600) 'Decay rate, decay no ',I,':',
     f                    THETAEST(I+NOOFDECAYS),' Locked'
          ELSEIF ((.NOT.LOCKED(I+NOOFDECAYS)).AND.
     f           ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED)) THEN
            WRITE(*,2600) 'Decay rate, decay no ',I,':',
     f                    THETAEST(I+NOOFDECAYS)
          ENDIF
 350    CONTINUE

        IF ((.NOT.LOCKED(LR)).AND.ADDNOISE.AND.
     f                  (.NOT.LOCKTOLINEARIZED)) THEN
          WRITE(*,2700) 'Background            :',
     f                  THETAEST(LR),' +/- ',SQRT(ABS(GMATRIX(LR,LR)))
 2700     FORMAT(' ',A,F12.5,A,F9.4)
        ELSEIF (LOCKED(LR)) THEN
          WRITE(*,2700) 'Background            :',
     f                  THETAEST(LR),' Locked'
        ELSEIF ((.NOT.LOCKED(LR)).AND.
     f         ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED)) THEN
          WRITE(*,2700) 'Background            :',
     f                   THETAEST(LR)
        ENDIF

```

```
      WRITE(*,2300)

360   WRITE(*,*) 'New iteration on the data? (Y/*):'
      READ(*,1300) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        GOTO 100
      ENDIF

      WRITE(*,2800) 'New calculation? (Y/*):'
2800  FORMAT(/' ',A)
      READ(*,1300) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        GOTO 10
      ELSE
        STOP
      ENDIF
      END

      REAL FUNCTION F(THETA,L,X)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    L,I
      REAL       THETA(1:*),X,SLASK

      SLASK=THETA(2*L+1)
      DO 10 I=1,L
        SLASK=SLASK+THETA(I)*EXP(-MIN(THETA(L+I)*X,80.0))
10    CONTINUE
      F=SLASK
      RETURN
      END

      REAL FUNCTION DFDTH(J,THETA,L,X)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    J,L
      REAL       THETA(1:*),X

      IF (J.LE.L) THEN
        DFDTH=EXP(-MIN(THETA(L+J)*X,80.0))
      ELSEIF (J.GT.L.AND.J.LE.2*L) THEN
        DFDTH=-X*THETA(J-L)*EXP(-MIN(THETA(J)*X,80.0))
      ELSE
        DFDTH=1.0
      ENDIF
      RETURN
      END

      REAL FUNCTION D2FDTH2(J,K,THETA,L,X)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    I,J,K,L
      REAL       THETA(1:*),X
      LOGICAL    JGTK

      JGTK=J.GT.K
      IF (JGTK) THEN
        I=J
        J=K
        K=I
      ENDIF
      D2FDTH2=0.0

      IF (J.LE.L.AND.K.EQ.L+J) THEN
        D2FDTH2=-X*EXP(-MIN(THETA(K)*X,80.0))
      ELSEIF ((J.GT.L.AND.J.LE.2*L).AND.(K.EQ.J)) THEN
        D2FDTH2=X*X*THETA(J-L)*EXP(-MIN(THETA(J)*X,80.0))
      ENDIF

      IF (JGTK) THEN
        I=J
        J=K
        K=I
      ENDIF

      RETURN
      END
```

GAUSSN

Programmet anpassar en summa av Gauss-toppar med bakgrund till indata. Programmet liknar EXP1 och EXPN. Dock kan parametervärden ej låsas 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:

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

Position, topp nr.i: θ_{L+i} (71)

Intensitet, topp nr.i: θ_i (72)

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

Förstaderivator:

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

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

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

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

Andraderivator:

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

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

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

$$+4\theta_{j-L} \cdot \theta_{2L+1}^2 \cdot (x-\theta_j)^2 \cdot \exp(-\theta_{2L+1} \cdot (x-\theta_j)^2) \tag{80}$$

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

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

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

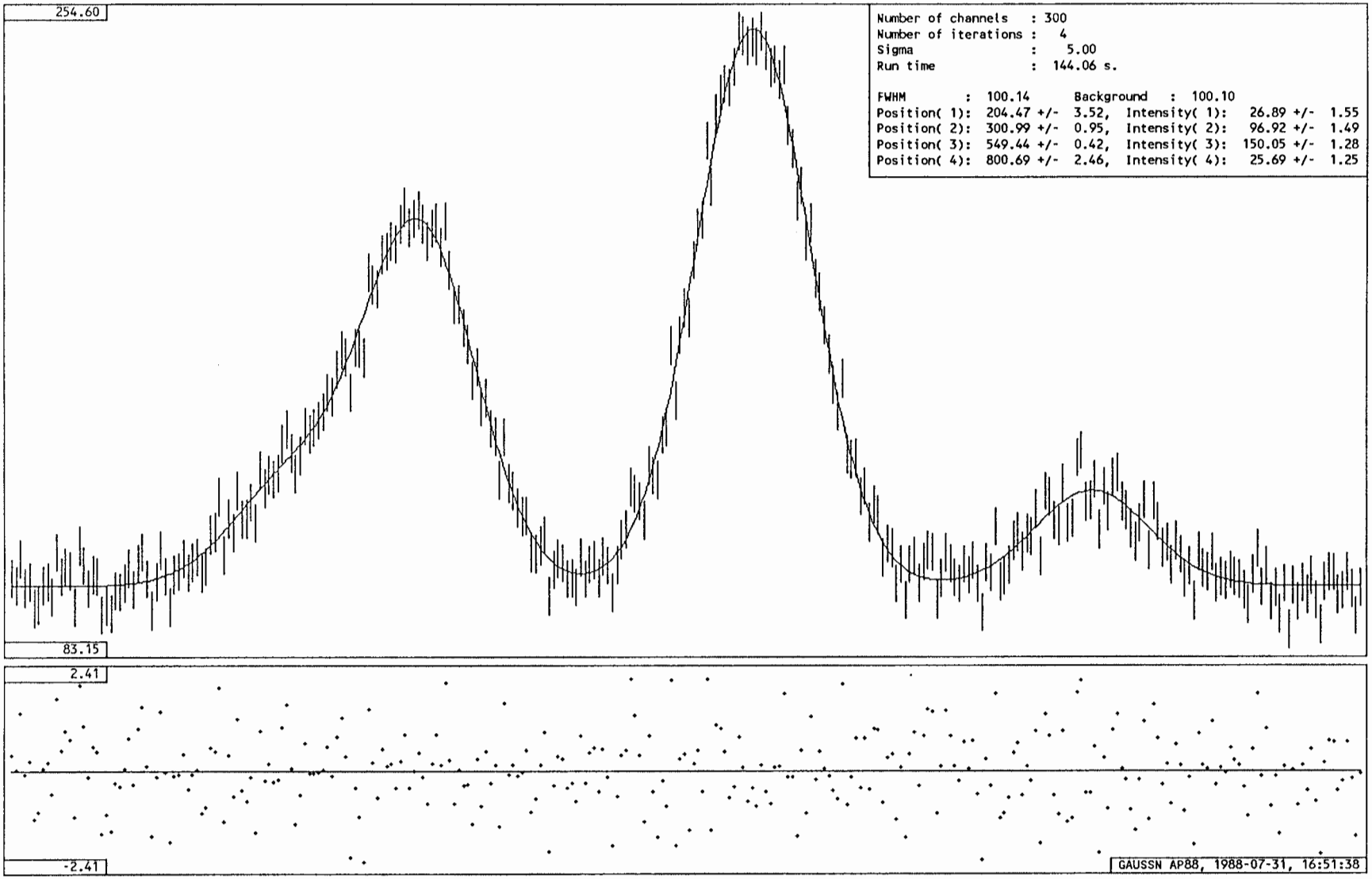
$$\frac{\partial^2 f}{\partial\theta_j \partial\theta_k} = \frac{\partial^2 f}{\partial\theta_k \partial\theta_j} \qquad (83)$$

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
```


254.60
83.15
2.41
-2.41
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 $f(\underline{\theta};x)$, (70).
363-388	Rutinen beräknar förstaderivatorna av f, (74)..(77).
390-442	Rutinen beräknar andraderivatorna av f, (78)..(83).

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: GAUSSN.FOR  Options: /LBZ          08/02/88 14:24:40

      PROGRAM GAUSSN
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    N,L,MAXITER,MAXNOOFPEAKS
      PARAMETER  (N=1000, MAXNOOFPEAKS=5, MAXITER=1000)
      PARAMETER  (L=2*MAXNOOFPEAKS+2)
      REAL       Y(1:N),X(1:N),DX,DET,RANF,NORMAL,THETAEST(1:L),SLASK
      REAL       RESIDUE(1:N),SIGMA(1:N),F,THETA(1:L),INVSIGMA2(1:N)
      REAL       GMATRIX(1:L,1:L),GVECTOR(1:L),DELTATHETA(1:L),D2FDTH2
      REAL       DFDTH,SIGMAIN,FWHM,POSITION(1:MAXNOOFPEAKS)
      REAL       INTENSITY(1:MAXNOOFPEAKS),SL1,SL2,SL3,SL4,SL5,TIME
      REAL       ERRPOS(1:MAXNOOFPEAKS),ERRINT(1:MAXNOOFPEAKS)
      REAL       JX,JY,JXL,JYL,DJY
      INTEGER    R,K,I,NCH,NOOFIT,NOOFPEAKS,LR,J
      INTEGER*2  HO,MI,SE,HN,ANS,KBDCHK,KBDINC
      INTEGER*2  YEAR,MONTH,DAY
      CHARACTER  SV*1,JETTXT*70
      LOGICAL    SING,ERRORBAR,LINEARIZED
      EXTERNAL   F

10    WRITE(*,1000) 'No of Gauss-peaks (1..',MAXNOOFPEAKS,')      :'
1000  FORMAT(' ',A,I2,A)
      READ(*,*,ERR=10) NOOFPEAKS
      IF (NOOFPEAKS.LT.1.OR.NOOFPEAKS.GT.MAXNOOFPEAKS) THEN
        GOTO 10
      ENDIF
      LR=2*NOOFPEAKS+2

20    WRITE(*,1100)   'No of channels (',LR,'..',N,')      :'
1100  FORMAT('+',A,I2,A,I4,A)
      READ(*,*,ERR=20) NCH
      IF (NCH.LT.LR.OR.NCH.GT.N) GOTO 20

      DO 30 I=1,NOOFPEAKS
        WRITE(*,1200) 'Position, peak no  ',I,' (0..1000):'
1200    FORMAT('+',A,I2,A)
        READ(*,*) POSITION(I)
        WRITE(*,1300) 'Intensity, peak no ',I,'          :'
1300    FORMAT('+',A,I2,A)
        READ(*,*) INTENSITY(I)
30    CONTINUE

      WRITE(*,1200)   'FWHM                            :'
      READ(*,*) FWHM
      FWHM=ABS(FWHM)

      WRITE(*,1200)   'Background                      :'
      READ(*,*) THETA(LR)

      WRITE(*,1200)   'Sigma(Y)                        :'
      READ(*,*) SIGMAIN
      SIGMAIN=ABS(SIGMAIN)

      ERRORBAR=.FALSE.
      IF (SIGMAIN.NE.0.0) THEN
        WRITE(*,1200) 'Plot errorbars? (Y/*)           :'
        READ(*,1400) SV
1400    FORMAT(A)
        ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'
      ENDIF

      THETA(LR-1)=LOG(2.0)/(0.5*FWHM)**2

      DO 40 I=1,NOOFPEAKS
        THETA(NOOFPEAKS+I)=POSITION(I)
        THETA(I)=INTENSITY(I)
40    CONTINUE

      DX=1000.0/(NCH-1)
      DO 50 I=1,NCH
        X(I)=(I-1)*DX
        SLASK=F(THETA,NOOFPEAKS,X(I))
        Y(I)=NORMAL(SLASK,SIGMAIN)
        SIGMA(I)=SIGMAIN
        IF (SIGMAIN.NE.0.0) THEN
          INVSIGMA2(I)=-2.0/SIGMAIN**2
        ELSE
          INVSIGMA2(I)=-2.0
        ENDIF
50    CONTINUE

      WRITE(*,*) 'Enter start values for intensity,'
      WRITE(*,*) 'position, FWHM and background.   '
      WRITE(*,*) '---------------------------------'
      WRITE(*,*) ' '

      DO 60 I=1,NOOFPEAKS
        WRITE(*,1200) 'Position, peak no  ',I,' (0..1000):'
        READ(*,*) POSITION(I)
        WRITE(*,1300) 'Intensity, peak no ',I,'          :'
        READ(*,*) INTENSITY(I)
60    CONTINUE

      WRITE(*,1200)   'FWHM                            :'
      READ(*,*) FWHM
      FWHM=ABS(FWHM)

      WRITE(*,1200)   'Background                      :'
      READ(*,*) THETAEST(LR)
      THETAEST(LR-1)=LOG(2.0)/(0.5*FWHM)**2

      DO 70 I=1,NOOFPEAKS
        THETAEST(NOOFPEAKS+I)=POSITION(I)
        THETAEST(I)=INTENSITY(I)
70    CONTINUE

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01

      WRITE(*,*) ' '
      WRITE(*,*) 'Press Q to quit iteration.'
      WRITE(*,*) ' '

      FWHM=2.0*SQRT(ABS(LOG(2.0)/THETAEST(LR-1)))
      DO 80 I=1,NOOFPEAKS
        POSITION(I)=THETAEST(NOOFPEAKS+I)
```

```
        INTENSITY(I)=THETAEST(I)
80    CONTINUE

      WRITE(*,1500) 'IT:',0
      WRITE(*,1600) 'FWHM',':',FWHM,'Background   :',THETAEST(LR)
      DO 90 I=1,NOOFPEAKS
        WRITE(*,1700) 'Position(',I,'):',POSITION(I),
     f           '  Intensity(',I,'):',INTENSITY(I)
90    CONTINUE

      LINEARIZED=.TRUE.
      DO 170 J=1,MAXITER
        CALL ZEROMAT(GMATRIX,LR,LR,L,L)
        CALL ZEROMAT(GVECTOR,LR,1,L,1)
        DO 130 I=1,NCH
          SL1=INVSIGMA2(I)
          SL2=Y(I)-F(THETAEST,NOOFPEAKS,X(I))
          DO 120 R=1,LR
            SL3=DFDTH(R,THETAEST,NOOFPEAKS,X(I))
            GVECTOR(R)=GVECTOR(R)+SL1*SL2*SL3
            DO 110 K=R,LR
              SL4=DFDTH(K,THETAEST,NOOFPEAKS,X(I))
              GMATRIX(R,K)=GMATRIX(R,K)+SL1*(-SL3*SL4)
              IF (.NOT.LINEARIZED) THEN
                SL5=D2FDTH2(R,K,THETAEST,NOOFPEAKS,X(I))
                GMATRIX(R,K)=GMATRIX(R,K)+SL1*SL2*SL5
              ENDIF
              GMATRIX(K,R)=GMATRIX(R,K)
110         CONTINUE
120       CONTINUE
130     CONTINUE

        CALL INVMAT(GMATRIX,LR,DET,SING,L,L)
        IF (SING) THEN
          WRITE(*,*) 'The G-matrix is singular.'
          GOTO 260
        ENDIF

        CALL MULTCMAT(GMATRIX,-1.0,LR,LR,L,L)
        CALL MULTMAT(GMATRIX,GVECTOR,DELTATHETA,LR,LR,1,L,L,L,1,L,1)
        CALL ADDMAT(THETAEST,DELTATHETA,THETAEST,LR,1,L,1,L,1,L,1)

        SLASK=0.0
        DO 140 I=1,LR
          IF (THETAEST(I).NE.0.0) THEN
            SLASK=SLASK+ABS(DELTATHETA(I)/THETAEST(I))
          ENDIF
140     CONTINUE

        FWHM=2.0*SQRT(ABS(LOG(2.0)/THETAEST(LR-1)))
        DO 150 I=1,NOOFPEAKS
          POSITION(I)=THETAEST(NOOFPEAKS+I)
          INTENSITY(I)=THETAEST(I)
150     CONTINUE

        WRITE(*,1500) 'IT:',J,' ,STEP:',SLASK
1500    FORMAT(' ',49('-'),/,' ',A,I3,A,G13.6)
        WRITE(*,1600) 'FWHM',':',FWHM,'Background   :',THETAEST(LR)
1600    FORMAT(' ',A,T14,A,G12.5,T29,A,G12.5)

        DO 160 I=1,NOOFPEAKS
          WRITE(*,1700) 'Position(',I,'):',POSITION(I),
     f                  '  Intensity(',I,'):',INTENSITY(I)
1700      FORMAT(' ',A,I2,A,G12.5,A,I2,A,G12.5)
160     CONTINUE

        IF (SLASK.LE.0.5E-2.AND.LINEARIZED) THEN
          LINEARIZED=.FALSE.
        ENDIF
        IF (SLASK.LE.0.5E-4.AND.(.NOT.LINEARIZED)) THEN
          NOOFIT=J
          GOTO 180
        ENDIF

        ANS=KBDCHK()
        IF (ANS.NE.0) THEN
          ANS=KBDINC()
          IF (CHAR(ANS).EQ.'Q'.OR.CHAR(ANS).EQ.'q') THEN
            NOOFIT=J
            GOTO 180
          ENDIF
        ENDIF

170   CONTINUE
      NOOFIT=MAXITER
180   CONTINUE
      CALL MULTCMAT(GMATRIX,-2.0,LR,LR,L,L)

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
      WRITE(*,1400) ' '
      WRITE(*,*) 'Press any key to continue.'
190   ANS=KBDCHK()
      IF (ANS.EQ.0) THEN
        GOTO 190
      ENDIF
      ANS=KBDINC()

      CALL SCRPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,NOOFPEAKS,ERRORBAR)

200   WRITE(*,*) 'Plot on Roland plotter? (Y/*):'
      READ(*,1400) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
       CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,NOOFPEAKS,ERRORBAR)
       GOTO 200
      ENDIF

210   WRITE(*,*) 'Plot on HP-Laserjet?    (Y/*):'
      READ(*,1400) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CALL HPSTART('L')
        CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
     f               NOOFPEAKS,ERRORBAR)
        JX=710
        JXL=1119-JX
        JYL=(6+1+NOOFPEAKS)*13
        JY=719-JYL
        CALL HPBOX(JX,JY,JXL,JYL)
        JY=719-16
        DJY=-13
```

```
238
239         WRITE(JETTXT,1900) 'Number of channels   :',NCH
240         CALL HPPRINT(JX,JY,70,JETTXT)
241         JY=JY+DJY
242
243         WRITE(JETTXT,1900) 'Number of iterations :',NOOFIT
244         CALL HPPRINT(JX,JY,70,JETTXT)
245         JY=JY+DJY
246
247         WRITE(JETTXT,2000) 'Sigma                :',SIGMAIN
248         CALL HPPRINT(JX,JY,70,JETTXT)
249         JY=JY+DJY
250
251         WRITE(JETTXT,2000) 'Run time             :',TIME,' s.'
252         CALL HPPRINT(JX,JY,70,JETTXT)
253         JY=JY+DJY
254
255         WRITE(JETTXT,1400) ' '
256         CALL HPPRINT(JX,JY,70,JETTXT)
257         JY=JY+DJY
258
259         FWHM=2.0*SQRT(ABS(LOG(2.0)/THETAEST(LR-1)))
260
261         DO 220 I=1,NOOFPEAKS
262           POSITION(I)=THETAEST(NOOFPEAKS+I)
263           ERRPOS(I)=SQRT(GMATRIX(NOOFPEAKS+I,NOOFPEAKS+I))
264           IF (SIGMAIN.EQ.0.0) THEN
265             ERRPOS(I)=0.0
266           ENDIF
267           INTENSITY(I)=THETAEST(I)
268           ERRINT(I)=SQRT(GMATRIX(I,I))
269           IF (SIGMAIN.EQ.0.0) THEN
270             ERRINT(I)=0.0
271           ENDIF
272 220     CONTINUE
273
274         WRITE(JETTXT,1600)
275      f  'FWHM',':',FWHM,'Background   :',THETAEST(LR)
276         CALL HPPRINT(JX,JY,70,JETTXT)
277         JY=JY+DJY
278
279         DO 230 I=1,NOOFPEAKS
280           WRITE(JETTXT,2100) 'Position(',I,'):',POSITION(I),
281      f                       ' +/- ',ERRPOS(I),
282      f                       ',  Intensity(',I,'):',INTENSITY(I),
283      f                       ' +/- ',ERRINT(I)
284           CALL HPPRINT(JX,JY,70,JETTXT)
285           JY=JY+DJY
286 230     CONTINUE
287
288         CALL GETTIM(HO,MI,SE,HN)
289         CALL GETDAT(YEAR,MONTH,DAY)
290         WRITE(JETTXT,2300)
291      f  'GAUSSN AP88, ',YEAR,'-',MONTH,'-',DAY,', ',HO,':',MI,':',SE
292 2300    FORMAT(A,I4.4,5(A,I2.2))
293         CALL HPTBOX(909.0,0.0,33,JETTXT)
294
295         CALL HPEJECT
296         GOTO 210
297       ENDIF
298
299       WRITE(*,1800)
300 1800  FORMAT(/)
301       WRITE(*,1900) 'Number of channels  :',NCH
302       WRITE(*,1900) 'Number of iterations:',NOOFIT
303 1900  FORMAT(' ',A,I4)
304       write(*,2000) 'Sigma               :',SIGMAIN
305       WRITE(*,2000) 'Run time            :',TIME,' s.'
306 2000  FORMAT(' ',A,F8.2,A)
307       WRITE(*,1800)
308
309       FWHM=2.0*SQRT(ABS(LOG(2.0)/THETAEST(LR-1)))
310
311       DO 240 I=1,NOOFPEAKS
312         POSITION(I)=THETAEST(NOOFPEAKS+I)
313         ERRPOS(I)=SQRT(GMATRIX(NOOFPEAKS+I,NOOFPEAKS+I))
314         IF (SIGMAIN.EQ.0.0) THEN
315           ERRPOS(I)=0.0
316         ENDIF
317         INTENSITY(I)=THETAEST(I)
318         ERRINT(I)=SQRT(GMATRIX(I,I))
319         IF (SIGMAIN.EQ.0.0) THEN
320           ERRINT(I)=0.0
321         ENDIF
322 240   CONTINUE
323
324       WRITE(*,1600) 'FWHM',':',FWHM,'Background   :',THETAEST(LR)
325
326       DO 250 I=1,NOOFPEAKS
327         WRITE(*,2100) 'Position(',I,'):',POSITION(I),
328      f                ' +/- ',ERRPOS(I),
329      f                ',  Intensity(',I,'):',INTENSITY(I),
330      f                ' +/- ',ERRINT(I)
331 250   CONTINUE
332 2100  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
```

```
      F=SLASK
      RETURN

      END

      REAL FUNCTION DFDTH(J,THETA,L,X)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    J,L,I,M
      REAL       THETA(1:*),X,SLASK,SL

      SL(M)=EXP(-MIN(THETA(2*L+1)*(X-THETA(M))**2,80.0))

      DFDTH=0.0
      IF (J.LE.L) THEN
        DFDTH=SL(L+J)
      ELSEIF (J.GT.L.AND.J.LE.2*L) THEN
        DFDTH=2.0*THETA(J-L)*THETA(2*L+1)*(X-THETA(J))*SL(J)
      ELSEIF (J.EQ.2*L+1) THEN
        SLASK=0.0
        DO 10 I=1,L
          SLASK=SLASK-THETA(I)*((X-THETA(L+I))**2)*SL(L+I)
10      CONTINUE
        DFDTH=SLASK
      ELSEIF (J.EQ.2*L+2) THEN
        DFDTH=1.0
      ENDIF

      RETURN
      END

      REAL FUNCTION D2FDTH2(J,K,THETA,L,X)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    J,K,L,I,M
      REAL       THETA(1:*),X,SL,SLASK
      LOGICAL    JGTK

      SL(M)=EXP(-MIN(THETA(2*L+1)*(X-THETA(M))**2,80.0))

      JGTK=J.GT.K
      IF (JGTK) THEN
        I=J
        J=K
        K=I
      ENDIF
      D2FDTH2=0.0

      IF (J.LE.L) THEN
        IF (K.EQ.L+J) THEN
          D2FDTH2=2.0*THETA(2*L+1)*(X-THETA(L+J))*SL(L+J)
        ELSEIF (K.EQ.2*L+1) THEN
          D2FDTH2=-((X-THETA(L+J))**2)*SL(L+J)
        ENDIF
      ELSEIF (J.GT.L.AND.J.LE.2*L) THEN
        IF (K.EQ.J) THEN
          SLASK=THETA(J-L)*THETA(2*L+1)*SL(J)
          D2FDTH2=SLASK*(4.0*THETA(2*L+1)*(X-THETA(J))**2-2.0)
        ELSEIF (K.EQ.2*L+1) THEN
          SLASK=2.0*THETA(J-L)*(X-THETA(J))*SL(J)
          D2FDTH2=SLASK*(1.0-THETA(2*L+1)*(X-THETA(J))**2)
        ENDIF
      ELSEIF (J.EQ.2*L+1) THEN
        IF (K.GT.L.AND.K.LE.2*L) THEN
          SLASK=2.0*THETA(K-L)*(X-THETA(K))*SL(K)
          D2FDTH2=SLASK*(1.0-THETA(2*L+1)*(X-THETA(K))**2)
        ELSEIF (K.EQ.2*L+1) THEN
          SLASK=0.0
          DO 10 I=1,L
            SLASK=SLASK+THETA(I)*((X-THETA(L+I))**4)*SL(L+I)
10        CONTINUE
          D2FDTH2=SLASK
        ENDIF
      ENDIF

      IF (JGTK) THEN
        I=J
        J=K
        K=I
      ENDIF

      RETURN
      END
```

LORN

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

Anpassad funktion:

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

Position, topp nr.i: θ_{L+i} (85)

Intensitet, topp nr.i: θ_i/θ_{2L+1} (86)

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

Förstaderivator:

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

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

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

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

Andraderivator:

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

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

$$\frac{\partial^2 f}{\partial\theta_j\partial\theta_k} \begin{bmatrix} j=L+1..2L \\ k=j \end{bmatrix} = 8\theta_{j-L}(x-\theta_j)^2\cdot[(x-\theta_j)^2+\theta_{2L+1}]^{-3} -$$

$$-2\theta_{j-L}\cdot[(x-\theta_j)^2+\theta_{2L+1}]^{-2} \tag{94}$$

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

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

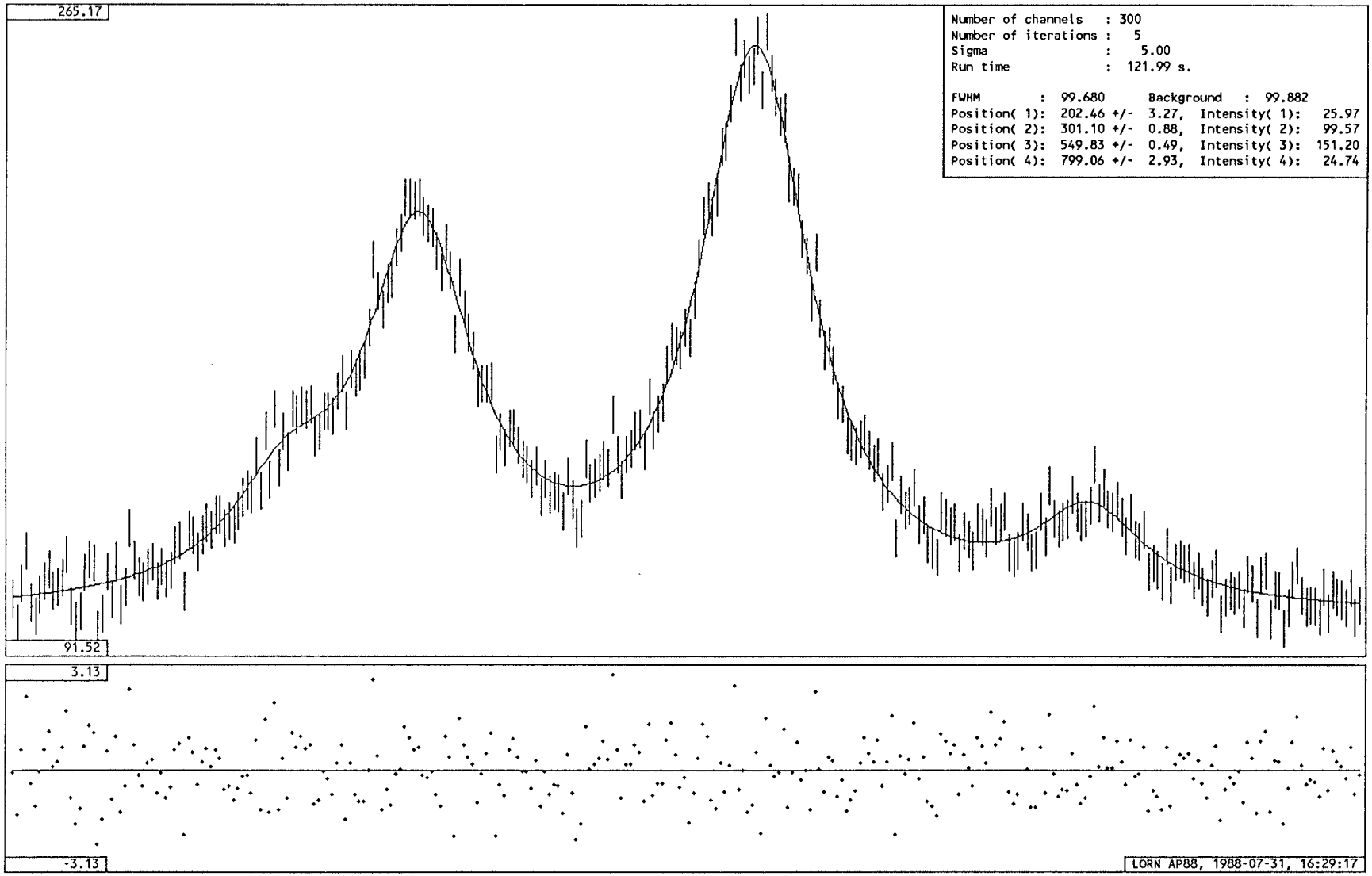
$$\frac{\partial^2 f}{\partial\theta_j\partial\theta_k} = \frac{\partial^2 f}{\partial\theta_k\partial\theta_j} \tag{97}$$

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
```


265.17
91.52
Number of channels : 300
Number of iterations : 5
Sigma : 5.00
Run time : 121.99 s.
FWHM : 99.680 Background : 99.882
Position(1): 202.46 +/- 3.27, Intensity(1): 25.97
Position(2): 301.10 +/- 0.88, Intensity(2): 99.57
Position(3): 549.83 +/- 0.49, Intensity(3): 151.20
Position(4): 799.06 +/- 2.93, Intensity(4): 24.74
3.13
-3.13
LORN AP88, 1988-07-31, 16:29:17

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 $f(\underline{\theta};x)$, (84).
348-373	Rutinen beräknar förstaderivatorna av f, (88)..(91).
375-420	Rutinen beräknar andraderivatorna av f, (92)..(97).

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

      PROGRAM LORN
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    N,L,MAXITER,MAXNOOFPEAKS
      PARAMETER  (N=1000, MAXNOOFPEAKS=5, MAXITER=1000)
      PARAMETER  (L=2*MAXNOOFPEAKS+2)
      REAL       Y(1:N),X(1:N),DX,DET,RANF,NORMAL,THETAEST(1:L),SLASK
      REAL       RESIDUE(1:N),SIGMA(1:N),F,THETA(1:L),INVSIGMA2(1:N)
      REAL       GMATRIX(1:L,1:L),GVECTOR(1:L),DELTATHETA(1:L),D2FDTH2
      REAL       DFDTH,SIGMAIN,FWHM,POSITION(1:MAXNOOFPEAKS)
      REAL       INTENSITY(1:MAXNOOFPEAKS),SL1,SL2,SL3,SL4,SL5,TIME
      REAL       ERRPOS(1:MAXNOOFPEAKS),JX,JY,JXL,JYL,DJY
      INTEGER    R,K,I,NCH,NOOFIT,NOOFPEAKS,LR,J
      INTEGER*2  HO,MI,SE,HN,ANS,KBDCHK,KBDINC
      INTEGER*2  YEAR,MONTH,DAY
      CHARACTER  SV*1,JETTXT*60
      LOGICAL    SING,ERRORBAR,LINEARIZED
      EXTERNAL   F

10    WRITE(*,1000) 'No of Lorentz-peaks (1..',MAXNOOFPEAKS,')    :'
1000  FORMAT(' ',A,I2,A)
      READ(*,*,ERR=10) NOOFPEAKS
      IF (NOOFPEAKS.LT.1.OR.NOOFPEAKS.GT.MAXNOOFPEAKS) THEN
        GOTO 10
      ENDIF
      LR=2*NOOFPEAKS+2

20    WRITE(*,1100)   'No of channels (',LR,'..',N,')      :'
1100  FORMAT('+',A,I2,A,I4,A)
      READ(*,*,ERR=20) NCH
      IF (NCH.LT.LR.OR.NCH.GT.N) GOTO 20

      DO 30 I=1,NOOFPEAKS
        WRITE(*,1200) 'Position, peak no  ',I,' (0..1000):'
1200    FORMAT('+',A,I2,A)
        READ(*,*) POSITION(I)
        WRITE(*,1300) 'Intensity, peak no ',I,'          :'
1300    FORMAT('+',A,I2,A)
        READ(*,*) INTENSITY(I)
30    CONTINUE

      WRITE(*,1200)   'FWHM                            :'
      READ(*,*) FWHM
      FWHM=ABS(FWHM)

      WRITE(*,1200)   'Background                      :'
      READ(*,*) THETA(LR)

      WRITE(*,1200)   'Sigma(Y)                        :'
      READ(*,*) SIGMAIN
      SIGMAIN=ABS(SIGMAIN)

      ERRORBAR=.FALSE.
      IF (SIGMAIN.NE.0.0) THEN
        WRITE(*,1200) 'Plot errorbars? (Y/*)           :'
        READ(*,1400) SV
1400    FORMAT(A)
        ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'
      ENDIF

      THETA(LR-1)=(0.5*FWHM)**2

      DO 40 I=1,NOOFPEAKS
        THETA(NOOFPEAKS+I)=POSITION(I)
        THETA(I)=INTENSITY(I)*THETA(LR-1)
40    CONTINUE

      DX=1000.0/(NCH-1)
      DO 50 I=1,NCH
        X(I)=(I-1)*DX
        SLASK=F(THETA,NOOFPEAKS,X(I))
        Y(I)=NORMAL(SLASK,SIGMAIN)
        SIGMA(I)=SIGMAIN
        IF (SIGMAIN.NE.0.0) THEN
          INVSIGMA2(I)=-2.0/SIGMAIN**2
        ELSE
          INVSIGMA2(I)=-2.0
        ENDIF
50    CONTINUE

      WRITE(*,*) 'Enter start values for intensity,'
      WRITE(*,*) 'position, FWHM and background.'
      WRITE(*,*) '--------------------------------'
      WRITE(*,*) ' '

      DO 60 I=1,NOOFPEAKS
        WRITE(*,1200) 'Position, peak no  ',I,' (0..1000):'
        READ(*,*) POSITION(I)
        WRITE(*,1300) 'Intensity, peak no ',I,'          :'
        READ(*,*) INTENSITY(I)
60    CONTINUE

      WRITE(*,1200)   'FWHM                            :'
      READ(*,*) FWHM
      FWHM=ABS(FWHM)

      WRITE(*,1200)   'Background                      :'
      READ(*,*) THETAEST(LR)

      THETAEST(LR-1)=(0.5*FWHM)**2
      DO 70 I=1,NOOFPEAKS
        THETAEST(NOOFPEAKS+I)=POSITION(I)
        THETAEST(I)=INTENSITY(I)*THETAEST(LR-1)
70    CONTINUE

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01

      WRITE(*,*) ' '
      WRITE(*,*) 'Press Q to quit iteration.'
      WRITE(*,*) ' '

      FWHM=2.0*SQRT(ABS(THETAEST(LR-1)))
      DO 80 I=1,NOOFPEAKS
        POSITION(I)=THETAEST(NOOFPEAKS+I)
        INTENSITY(I)=THETAEST(I)/THETAEST(LR-1)
```

```
80    CONTINUE

      WRITE(*,1500) 'IT:',0
      WRITE(*,1600) 'FWHM',':',FWHM,'Background   :',THETAEST(LR)
      DO 90 I=1,NOOFPEAKS
        WRITE(*,1700) 'Position(',I,'):',POSITION(I),
     f           '  Intensity(',I,'):',INTENSITY(I)
90    CONTINUE

      LINEARIZED=.TRUE.
      DO 170 J=1,MAXITER
        CALL ZEROMAT(GMATRIX,LR,LR,L,L)
        CALL ZEROMAT(GVECTOR,LR,1,L,1)
        DO 130 I=1,NCH
          SL1=INVSIGMA2(I)
          SL2=Y(I)-F(THETAEST,NOOFPEAKS,X(I))
          DO 120 R=1,LR
            SL3=DFDTH(R,THETAEST,NOOFPEAKS,X(I))
            GVECTOR(R)=GVECTOR(R)+SL1*SL2*SL3
            DO 110 K=R,LR
              SL4=DFDTH(K,THETAEST,NOOFPEAKS,X(I))
              GMATRIX(R,K)=GMATRIX(R,K)+SL1*(-SL3*SL4)
              IF (.NOT.LINEARIZED) THEN
                SL5=D2FDTH2(R,K,THETAEST,NOOFPEAKS,X(I))
                GMATRIX(R,K)=GMATRIX(R,K)+SL1*SL2*SL5
              ENDIF
              GMATRIX(K,R)=GMATRIX(R,K)
110         CONTINUE
120       CONTINUE
130     CONTINUE

        CALL INVMAT(GMATRIX,LR,DET,SING,L,L)
        IF (SING) THEN
          WRITE(*,*) 'The G-matrix is singular.'
          GOTO 260
        ENDIF

        CALL MULTCMAT(GMATRIX,-1.0,LR,LR,L,L)
        CALL MULTMAT(GMATRIX,GVECTOR,DELTATHETA,LR,LR,1,L,L,L,1,L,1)
        CALL ADDMAT(THETAEST,DELTATHETA,THETAEST,LR,1,L,1,L,1,L,1)

        SLASK=0.0
        DO 140 I=1,LR
          IF (THETAEST(I).NE.0.0) THEN
            SLASK=SLASK+ABS(DELTATHETA(I)/THETAEST(I))
          ENDIF
140     CONTINUE

        FWHM=2.0*SQRT(ABS(THETAEST(LR-1)))
        DO 150 I=1,NOOFPEAKS
          POSITION(I)=THETAEST(NOOFPEAKS+I)
          INTENSITY(I)=THETAEST(I)/THETAEST(LR-1)
150     CONTINUE

        WRITE(*,1500) 'IT:',J,' ,STEP:',SLASK
1500    FORMAT(' ',49('-'),/,' ',A,I3,A,G13.6)
        WRITE(*,1600) 'FWHM',':',FWHM,'Background   :',THETAEST(LR)
1600    FORMAT(' ',A,T14,A,G12.5,T29,A,G12.5)

        DO 160 I=1,NOOFPEAKS
          WRITE(*,1700) 'Position(',I,'):',POSITION(I),
     f                  '  Intensity(',I,'):',INTENSITY(I)
1700      FORMAT(' ',A,I2,A,G12.5,A,I2,A,G12.5)
160     CONTINUE

        IF (SLASK.LE.0.5E-2.AND.LINEARIZED) THEN
          LINEARIZED=.FALSE.
        ENDIF

        IF (SLASK.LE.0.5E-4.AND.(.NOT.LINEARIZED)) THEN
          NOOFIT=J
          GOTO 180
        ENDIF

        ANS=KBDCHK()
        IF (ANS.NE.0) THEN
          ANS=KBDINC()
          IF (CHAR(ANS).EQ.'Q'.OR.CHAR(ANS).EQ.'q') THEN
            NOOFIT=J
            GOTO 180
          ENDIF
        ENDIF

170   CONTINUE
      NOOFIT=MAXITER
180   CONTINUE
      CALL MULTCMAT(GMATRIX,-2.0,LR,LR,L,L)

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
      WRITE(*,1400) ' '
      WRITE(*,*) 'Press any key to continue.'
190   ANS=KBDCHK()
      IF (ANS.EQ.0) THEN
        GOTO 190
      ENDIF
      ANS=KBDINC()

      CALL SCRPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,NOOFPEAKS,ERRORBAR)

200   WRITE(*,*) 'Plot on Roland plotter? (Y/*):'
      READ(*,1400) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
       CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,NOOFPEAKS,ERRORBAR)
       GOTO 200
      ENDIF

210   WRITE(*,*) 'Plot on HP-Laserjet?    (Y/*):'
      READ(*,1400) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CALL HPSTART('L')
        CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
     f               NOOFPEAKS,ERRORBAR)
        JX=770
        JXL=1119-JX
        JYL=(6+1+NOOFPEAKS)*13
        JY=719-JYL
        CALL HPBOX(JX,JY,JXL,JYL)
        JY=719-16
        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,I4.4,5(A,I2.2))
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,I4)
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,I2,A,F8.2,A,F5.2,A,I2,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+I))**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
```

```
        SLASK1=2.0*THETA(J-L)*(X-THETA(J))
        SLASK2=((X-THETA(J))**2+THETA(2*L+1))**2
        DFDTH=SLASK1/SLASK2
      ELSEIF (J.EQ.2*L+1) THEN
        SLASK1=0.0
        DO 10 I=1,L
          SLASK2=((X-THETA(L+I))**2+THETA(2*L+1))**2
          SLASK1=SLASK1-THETA(I)/SLASK2
10      CONTINUE
        DFDTH=SLASK1
      ELSEIF (J.EQ.2*L+2) THEN
        DFDTH=1.0
      ENDIF

      RETURN
      END

      REAL FUNCTION D2FDTH2(J,K,THETA,L,X)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    J,K,L,I,M
      REAL       THETA(1:*),X,SL,SLASK
      LOGICAL    JGTK

      SL(M)=1.0/((X-THETA(M))**2+THETA(2*L+1))
      JGTK=J.GT.K
      IF (JGTK) THEN
        I=J
        J=K
        K=I
      ENDIF
      D2FDTH2=0.0

      IF (J.LE.L) THEN
        IF (K.EQ.L+J) THEN
          D2FDTH2=2.0*(X-THETA(K))*SL(K)**2
        ELSEIF (K.EQ.2*L+1) THEN
          D2FDTH2=-(SL(L+J)**2)
        ENDIF
      ELSEIF (J.GT.L.AND.J.LE.2*L) THEN
        IF (K.EQ.J) THEN
          SLASK=8.0*THETA(J-L)*((X-THETA(J))**2)*SL(J)**3
          D2FDTH2=SLASK-2.0*THETA(J-L)*SL(J)**2
        ELSEIF (K.EQ.2*L+1) THEN
          D2FDTH2=-4.0*THETA(J-L)*(X-THETA(J))*SL(J)**3
        ENDIF
      ELSEIF (J.EQ.2*L+1.AND.K.EQ.2*L+1) THEN
        SLASK=0.0
        DO 10 I=1,L
          SLASK=SLASK+2.0*THETA(I)*SL(L+I)**3
10      CONTINUE
        D2FDTH2=SLASK
      ENDIF

      IF (JGTK) THEN
        I=J
        J=K
        K=I
      ENDIF

      RETURN
      END
```

PCONEXPA

Programmet anpassar en summa av exponentialer 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 exponentialer och derivator av laserpulsen m.m. Dessa vektorer beräknas genom att Fouriertransformera puls och exponentialer 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 laserpulsen) 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)

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

Förstaderivator:

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

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

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

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

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

Andraderivator:

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k}\begin{pmatrix} j=1 \\ k=2 \end{pmatrix} = -\dot{P}(x-\theta_2) \tag{104}$$

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

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k}\begin{pmatrix} j=2 \\ k=3..L+2 \end{pmatrix} = -\dot{P}(x-\theta_2) \otimes \exp(-\theta_{L+k} \cdot x) \tag{106}$$

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

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k}\begin{pmatrix} j=3..L+2 \\ k=L+j \end{pmatrix} = -P(x-\theta_2) \otimes [x \cdot \exp(-\theta_k \cdot x)] \tag{108}$$

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

$$\frac{\partial^2 f}{\partial \theta_j \partial \theta_k} = \frac{\partial^2 f}{\partial \theta_k \partial \theta_j} \tag{110}$$

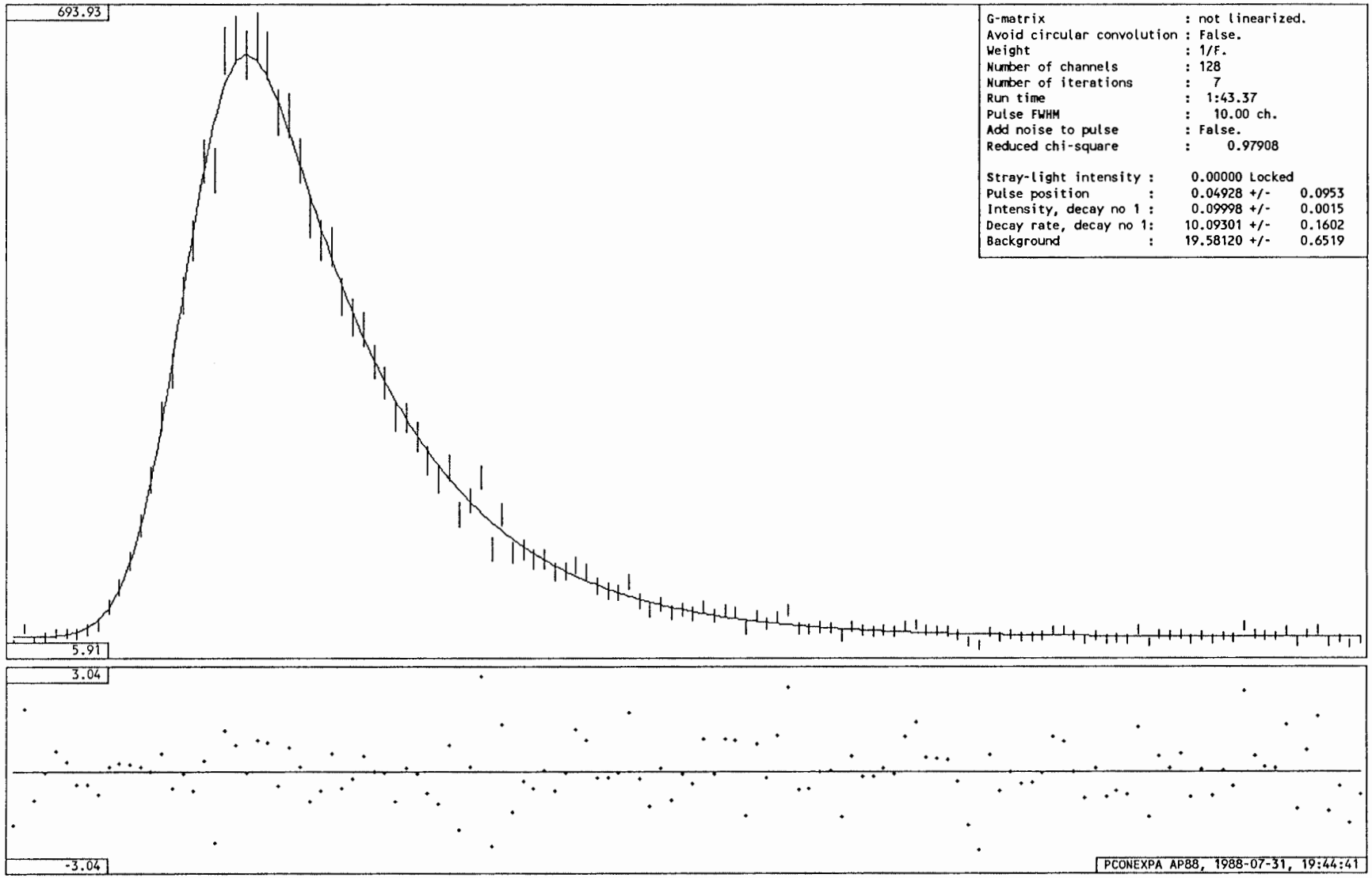
alla övriga andraderivator är noll.

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
```


693.93
5.91
3.04
-3.04
G-matrix : not linearized.
Avoid circular convolution : False.
Weight : 1/F.
Number of channels : 128
Number of iterations : 7
Run time : 1:43.37
Pulse FWHM : 10.00 ch.
Add noise to pulse : False.
Reduced chi-square : 0.97908
Stray-light intensity : 0.00000 Locked
Pulse position : 0.04928 +/- 0.0953
Intensity, decay no 1 : 0.09998 +/- 0.0015
Decay rate, decay no 1: 10.09301 +/- 0.1602
Background : 19.58120 +/- 0.6519
PCONEXPA AP88, 1988-07-31, 19:44:41

Kommentarer till programlistningen:

Rad	Kommentar
1-45	Deklarationer. Användning av vektorer och matriser: PO - laserpulsen. P - den translaterade laserpulsen. PPRIM - förstaderivatan av laserpulsen. PBIS - andraderivatan av laserpulsen. PCONVE - pulsen faltad med exp. PCONVXE - pulsen faltad med x·exp. PCONVX2E - pulsen faltad med x^2·exp. PPRIMCONVE - förstaderivatan faltad med exp. PPRIMCONVXE - förstaderivatan faltad med x·exp. PBISCONVE - andraderivatan faltad med exp. FTPO - Fouriertransformen av laserpulsen. FTE - Fouriertransformen av exp. FTEX - Fouriertransformen av x·exp. FTEX2 - Fouriertransformen av x^2·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 laserpulsen. Laserpulsen är en Gausstopp centrerad i position: x_0=antal kanaler/7.0.
135-145	Beräkna laserpulsen.
147-148	Beräkna sinus-cosinustabeller till Fouriertransformrutinen.

149-150 Beräkna laserpulsens Fouriertransform.

151 Beräkna alla derivator och faltningar av puls och exponentialer samt flytta laserpulsen till vald position. SLLOCKED-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 laserpulsen.

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=$\Sigma|\Delta\theta_i/\theta_i|$ 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 iterationsslingan. 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 laserpulsen P0, 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.

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: PCONEXPA.FOR  Options: /LBZ       08/02/88 14:38:57

      PROGRAM PCONEXPA
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)

      INTEGER    N,NHALF,L,MAXNOOFITERATIONS,MAXNOOFDECAYS

      PARAMETER  ( N=512, NHALF=N/2, MAXNOOFITERATIONS=100 )
      PARAMETER  ( MAXNOOFDECAYS=2, L=2*MAXNOOFDECAYS+3 )

      REAL       Y(1:N),X(1:N),DX,DET,NORMAL,THETAEST(1:L)
      REAL       RESIDUE(1:N),SIGMA(1:N),F,THETA(1:L),JX,JY,JXL,JYL,DJY
      REAL       GMATRIX(1:L,1:L),GVECTOR(1:L),DELTATHETA(1:L)
      REAL       SLASK,DFDTH,D2FDTH2,THETAMIN(1:L),THETAMAX(1:L)
      REAL       SL1,SL2,SL3,SL4,SL5,TIME,STEP,REDCHISQUARED
      REAL       P0(1:N),P(1:N),PPRIM(1:N),PBIS(1:N)
      REAL       PCONVE(1:N,1:MAXNOOFDECAYS)
      REAL       PCONVXE(1:N,1:MAXNOOFDECAYS)
      REAL       PCONVX2E(1:N,1:MAXNOOFDECAYS)
      REAL       PPRIMCONVE(1:N,1:MAXNOOFDECAYS)
      REAL       PPRIMCONVXE(1:N,1:MAXNOOFDECAYS)
      REAL       PBISCONVE(1:N,1:MAXNOOFDECAYS),PULSEFWHM

      COMPLEX    FTP0(1:N),FTE(1:N,1:MAXNOOFDECAYS)
      COMPLEX    FTEX(1:N,1:MAXNOOFDECAYS),FTEX2(1:N,1:MAXNOOFDECAYS)
      COMPLEX    S1(1:N),S2(1:N),W1(1:NHALF),W2(1:NHALF)

      INTEGER    R,K,I,NCH,NCH2,DNCH2,NOOFITERATIONS,J,NOOFDECAYS,LR
      INTEGER    NOOFVARIABLEPAR

      INTEGER*2  HO,MI,SE,HN,ANS,KBDCHK,KBDINC,YEAR,MONTH,DAY

      CHARACTER  SV*1,LOCKEDORFREE(1:L)*6,JETTXT*50

      LOGICAL    SING,ERRORBAR,LINEARIZED,LOCKED(1:L),ADDNOISE,INV
      LOGICAL    LOCKTOLINEARIZED,LOCKWEIGHT,AVOIDCIRCCONV,SLLOCKED(1:L)
      LOGICAL    ONEOVERFWEIGHT,ADDNOISETOPULSE

      EXTERNAL   F

      COMMON     /BLOCK1/ P,PCONVE
      COMMON     /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
      COMMON     /BLOCK3/ PCONVX2E,PPRIMCONVXE,PBIS,PBISCONVE
      COMMON     /BLOCK4/ P0,FTP0,FTE,FTEX,FTEX2,S1,S2,W1,W2
      COMMON     /BLOCK5/ NCH,NCH2,DNCH2,NOOFDECAYS,DX

10    WRITE(*,1000) 'No of decays (1..',MAXNOOFDECAYS,'):'
1000  FORMAT(' ',A,I2,A)
      READ(*,*,ERR=10) NOOFDECAYS
      IF (NOOFDECAYS.LT.1.OR.NOOFDECAYS.GT.MAXNOOFDECAYS) THEN
        GOTO 10
      ENDIF
      LR=2*NOOFDECAYS+3

20    WRITE(*,1100) 'No of channels (',LR,'..',N,'):'
1100  FORMAT(' ',A,I2,A,I4,A)
      READ(*,*,ERR=20) NCH
      IF (NCH.LT.LR.OR.NCH.GT.N) THEN
        GOTO 20
      ENDIF
      DX=1.0/(NCH-1)

30    WRITE(*,*) 'Stray-light intensity:'
      READ(*,*,ERR=30) THETA(1)
      IF (THETA(1).LT.0.0) THEN
        GOTO 30
      ENDIF

40    WRITE(*,*) 'Pulse position (channels):'
      READ(*,*,ERR=40) THETA(2)
      THETA(2)=THETA(2)*DX

      DO 70 I=1,NOOFDECAYS
50      WRITE(*,1200) 'Intensity, decay no ',I,': '
1200    FORMAT(' ',A,I1,A)
        READ(*,*,ERR=50) THETA(I+2)
        IF (THETA(I+2).LE.0.0) THEN
          GOTO 50
        ENDIF
60      WRITE(*,1200) 'Decay rate, decay no ',I,': '
        READ(*,*,ERR=60) THETA(I+NOOFDECAYS+2)
        IF (THETA(I+NOOFDECAYS+2).LE.0.0) THEN
          GOTO 60
        ENDIF
70    CONTINUE

80    WRITE(*,*) 'Background: '
      READ(*,*,ERR=80) THETA(LR)

      WRITE(*,*) 'Add noise to data? (Y/*): '
      READ(*,1300) SV
1300  FORMAT(A)
      ADDNOISE=SV.EQ.'Y'.OR.SV.EQ.'y'
      ERRORBAR=ADDNOISE
      IF (ADDNOISE) THEN
        WRITE(*,*) 'Plot errorbars? (Y/*): '
        READ(*,1300) SV
        ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'
      ENDIF

      AVOIDCIRCCONV=.FALSE.

      NCH2  = LOG(NCH-0.5)/LOG(2.0)
      NCH2  = 2**(NCH2+1)
      DNCH2 = 2*NCH2
      IF (.NOT.AVOIDCIRCCONV) THEN
        DNCH2=NCH2
      ENDIF

      DO 90 I=1,LR
        THETAEST(I)     = THETA(I)
        LOCKED(I)       = .FALSE.
        SLLOCKED(I)     = .TRUE.
        LOCKEDORFREE(I) = 'Free  '
90    CONTINUE

      LOCKED(1)         = .TRUE.
```

```
      LOCKED(2)         = .TRUE.
      LOCKEDORFREE(1)   = 'Locked'
      LOCKEDORFREE(2)   = 'Locked'
      LOCKTOLINEARIZED  = .FALSE.
      LINEARIZED        = .TRUE.
      ONEOVERFWEIGHT    = .TRUE.

100   WRITE(*,*) 'Pulse FWHM (channels):'
      READ(*,*,ERR=100) PULSEFWHM
      IF (PULSEFWHM.LT.0.1.OR.PULSEFWHM.GT.NCH*0.5) THEN
        GOTO 100
      ENDIF

      WRITE(*,*) 'Add noise to pulse? (Y/*):'
      READ(*,1300) SV
      ADDNOISETOPULSE=SV.EQ.'Y'.OR.SV.EQ.'y'

      DO 110 I=1,NCH
        PO(I)=EXP(-MIN((LOG(16.0)/PULSEFWHM**2)*(I-NCH/7.0)**2,87.0))
        PO(I)=1000.0*PO(I)
        IF (ADDNOISETOPULSE) THEN
          PO(I)=PO(I)+NORMAL(0.0,10.0)
        ENDIF
        FTPO(I)=CMPLX(PO(I),0.0)
110   CONTINUE
      DO 120 I=NCH+1,NCH2
        FTPO(I)=CMPLX(0.0,0.0)
120   CONTINUE

      CALL CALCW(W1,NCH2)
      CALL CALCW(W2,DNCH2)
      INV=.FALSE.
      CALL FFT(FTPO,W1,NCH2,INV)
      CALL CALCCONV(THETA,SLLOCKED,LINEARIZED)

      DO 130 I=1,NCH
        X(I)=(I-1)*DX
        SLASK=F(THETA,NOOFDECAYS,X(I))
        IF (ADDNOISE) THEN
          Y(I)=NORMAL(SLASK,SQRT(ABS(SLASK)))
          SIGMA(I)=SQRT(ABS(Y(I)))
        ELSE
          Y(I)=SLASK
          SIGMA(I)=0.0
        ENDIF
130   CONTINUE

140   CALL GMODE
      CALL TMODE

      WRITE(*,1400) 'Toggle Free/Locked with <shift>+Function key.'
1400  FORMAT(////,T10,A,/)

      WRITE(*,1500) THETAEST(1),LOCKEDORFREE(1)
1500  FORMAT(T10,'F1 : Stray-light intensity',T45,F12.4,',',T60,A)

      WRITE(*,1600) THETAEST(2)/DX,LOCKEDORFREE(2)
1600  FORMAT(T10,'F2 : Pulse position',T45,G12.4,',',T60,A)

      DO 150 I=1,NOOFDECAYS
        WRITE(*,1700) I+2,I,THETAEST(I+2),LOCKEDORFREE(I+2)
1700    FORMAT(T10,'F',I1,' : Intensity , decay no ',
     f         I1,T45,F12.4,',',T60,A)
150   CONTINUE
      DO 160 I=1,NOOFDECAYS
        WRITE(*,1800) I+NOOFDECAYS+2,I,THETAEST(I+NOOFDECAYS+2),
     f                LOCKEDORFREE(I+NOOFDECAYS+2)
1800    FORMAT(T10,'F',I1,' : Decay rate, decay no ',
     f         I1,T45,F12.4,',',T60,A)
160   CONTINUE
      WRITE(*,1900) LR,THETAEST(LR),LOCKEDORFREE(LR)
1900  FORMAT(T10,'F',I1,' : Background ',T45,F12.4,',',T60,A)
      WRITE(*,2000) 'F10: Calculate.'
2000  FORMAT(T10,A)
      WRITE(*,2100) '<Alt>+F10: Preview.'
2100  FORMAT(T4,A)
      IF (LOCKTOLINEARIZED) THEN
        WRITE(*,2200) '<Shift>+F10: G-matrix = linearized.'
      ELSE
        WRITE(*,2200) '<Shift>+F10: G-matrix = not linearized.'
      ENDIF
2200  FORMAT(T2,A)
      IF (AVOIDCIRCCONV) THEN
        WRITE(*,2300) '<Ctrl>+F10: Avoid circular convolution = True.'
      ELSE
        WRITE(*,2300) '<Ctrl>+F10: Avoid circular convolution = False.'
      ENDIF
      IF (ONEOVERFWEIGHT) THEN
        WRITE(*,2100) '<Alt>+F1 : Weight = 1/F.'
      ELSE
        WRITE(*,2100) '<Alt>+F1 : Weight = 1.'
      ENDIF
2300  FORMAT(T3,A)
      WRITE(*,2300) ' '

170   ANS=KBDCHK()
      IF (ANS.NE.0) THEN
        GOTO 170
      ENDIF
      ANS=KBDINC()

      IF (ANS.GE.1084.AND.ANS.LE.1083+LR) THEN
        I=ANS-1083
        LOCKED(I)=.NOT.LOCKED(I)
        LOCKEDORFREE(I)='Free  '
        IF (LOCKED(I)) THEN
          LOCKEDORFREE(I)='Locked'
        ENDIF
        GOTO 140
      ELSEIF (ANS.EQ.1113) THEN
        WRITE(*,2000) 'Preview.'
        CALL CALCW(W2,DNCH2)
        CALL CALCCONV(THETAEST,SLLOCKED,LINEARIZED)
        CALL SCRPLOT(Y,SIGMA,X,RESIDUE,NCH,F,
     f               THETAEST,NOOFDECAYS,ERRORBAR)
        GOTO 140
      ELSEIF (ANS.EQ.1068) THEN
C       Calculate.
        CALL CALCW(W2,DNCH2)
        GOTO 230
```

```
      ELSEIF (ANS.EQ.1093) THEN
        LOCKTOLINEARIZED=.NOT.LOCKTOLINEARIZED
        GOTO 140
      ELSEIF (ANS.EQ.1103) THEN
        AVOIDCIRCCONV=.NOT.AVOIDCIRCCONV
        IF (NCH.GT.NHALF) THEN
          AVOIDCIRCCONV=.FALSE.
        ENDIF
        DNCH2=NCH2
        IF (AVOIDCIRCCONV) THEN
          DNCH2=2*NCH2
        ENDIF
        GOTO 140
      ELSEIF (ANS.EQ.1104) THEN
        ONEOVERFWEIGHT=.NOT.ONEOVERFWEIGHT
        GOTO 140
      ELSEIF (ANS.GE.1059.AND.ANS.LE.1058+LR) THEN
        I=ANS-1058
        IF (I.EQ.1) THEN
180       WRITE(*,2400) 'New stray-light intensity:'
2400      FORMAT(T10,A,I1,A)
          READ(*,*,ERR=180) THETAEST(1)
          IF (THETAEST(1).LT.0.0) THEN
            GOTO 180
          ENDIF
          GOTO 140
        ELSEIF (I.EQ.2) THEN
190       WRITE(*,2400) 'New pulse position (channels):'
          READ(*,*,ERR=190) THETAEST(2)
          THETAEST(2)=THETAEST(2)*DX
          GOTO 140
        ELSEIF (I.GT.2.AND.I.LE.NOOFDECAYS+2) THEN
200       WRITE(*,2400) 'New intensity, decay no ',I-2,': '
          READ(*,*,ERR=200) THETAEST(I)
          IF (THETAEST(I).LE.0.0) THEN
            GOTO 200
          ENDIF
          GOTO 140
        ELSEIF (I.GT.NOOFDECAYS+2.AND.I.LT.LR) THEN
210       WRITE(*,2400) 'New decay rate, decay no ',I-NOOFDECAYS-2,': '
          READ(*,*,ERR=210) THETAEST(I)
          IF (THETAEST(I).LE.0.0) THEN
            GOTO 210
          ENDIF
          GOTO 140
        ELSEIF (I.EQ.LR) THEN
220       WRITE(*,2400) 'New background: '
          READ(*,*,ERR=220) THETAEST(LR)
          GOTO 140
        ENDIF
        GOTO 170
      ENDIF
      GOTO 170

C     *** Calculation ***
230   CONTINUE

      WRITE(*,*) 'Press Q to quit iteration.'
      WRITE(*,2800)
      DO 240 I=1,2*NOOFDECAYS+2
        THETAMIN(I)=THETAEST(I)/10.0
        THETAMAX(I)=THETAEST(I)*10.0
240   CONTINUE
      IF (THETAEST(1).LT.1.0) THEN
        THETAMAX(1)=1.0
        THETAMIN(1)=-1.0
      ENDIF
      THETAMAX(2)=THETAEST(2)+0.3*NCH*DX
      THETAMIN(2)=THETAEST(2)-0.3*NCH*DX

      THETAMAX(LR)= ABS(THETAEST(LR)*10.0)
      THETAMIN(LR)=-ABS(THETAEST(LR)*10.0)

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01

      WRITE(*,2500) 'It:',0
2500  FORMAT(' ',49('-'),/,' ',A,I4,A,G13.6)
      IF (.NOT.LOCKED(1)) THEN
        WRITE(*,2600) 'Stray-light int.:',THETAEST(1)
      ENDIF
      IF (.NOT.LOCKED(2)) THEN
        WRITE(*,2600) 'Pulse position  :',THETAEST(2)/DX
      ENDIF
2600  FORMAT(' ',A,G12.5)
      DO 250 R=1,NOOFDECAYS
        WRITE(*,2700) 'Intensity(',R,') :',THETAEST(R+2),
     f                ' ,Decay rate(',R,'):',THETAEST(R+NOOFDECAYS+2)
2700    FORMAT(' ',A,I1,A,G12.5,A,I1,A,G12.5)
250   CONTINUE
      IF (.NOT.LOCKED(LR)) THEN
        WRITE(*,2600) 'Background   :',THETAEST(LR)
      ENDIF

      LINEARIZED=.TRUE.
      LOCKWEIGHT=.TRUE.

      DO 330 J=1,MAXNOOFITERATIONS
        CALL ZEROMAT(GMATRIX,LR,LR,L,L)
        CALL ZEROMAT(GVECTOR,LR,1,L,1)
        CALL CALCCONV(THETAEST,LOCKED,LINEARIZED)
        DO 280 I=1,NCH
          IF (ONEOVERFWEIGHT) THEN
            IF (LOCKWEIGHT) THEN
              SL1=-2.0/MAX(Y(I),1.0)
            ELSE
              SL1=-2.0/F(THETAEST,NOOFDECAYS,X(I))
            ENDIF
          ELSE
            SL1=-2.0
          ENDIF
          SL2=Y(I)-F(THETAEST,NOOFDECAYS,X(I))
          DO 270 R=1,LR
            IF (.NOT.LOCKED(R)) THEN
              SL3=DFDTH(R,THETAEST,NOOFDECAYS,X(I))
              GVECTOR(R)=GVECTOR(R)+SL1*SL2*SL3
              DO 260 K=R,LR
                IF (.NOT.LOCKED(K)) THEN
                  SL4=DFDTH(K,THETAEST,NOOFDECAYS,X(I))
                  GMATRIX(R,K)=GMATRIX(R,K)+SL1*(-SL3*SL4)
```

```
358                 IF (.NOT.LINEARIZED) THEN
359                   SL5=D2FDTH2(R,K,THETAEST,NOOFDECAYS,X(I))
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,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     f                 ' ,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 SCRPLOT(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
```

```
      DO 380 I=1,NCH
        REDCHISQUARED=REDCHISQUARED+RESIDUE(I)**2
380   CONTINUE
      NOOFVARIABLEPAR=0
      DO 390 I=1,LR
        IF (.NOT.LOCKED(I)) THEN
          NOOFVARIABLEPAR=NOOFVARIABLEPAR+1
        ENDIF
390   CONTINUE
      IF (NCH.NE.NOOFVARIABLEPAR) THEN
        REDCHISQUARED=REDCHISQUARED/(NCH-NOOFVARIABLEPAR)
      ENDIF

400   WRITE(*,*) 'Plot on HP-Laserjet? (Y/*): '
      READ(*,1300) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CALL HPSTART('L')
        CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
     f               NOOFDECAYS,ERRORBAR)

        JX=800
        JXL=1119-JX
        JYL=(13+1+NOOFDECAYS*2)*13
        JY=719-JYL
        CALL HPBOX(JX,JY,JXL,JYL)
        JX=800
        JY=719-16
        DJY=-13

        IF (LOCKTOLINEARIZED) THEN
          WRITE(JETTXT,2900)
     f    'G-matrix                    : linearized.'
        ELSE
          WRITE(JETTXT,2900)
     f    'G-matrix                    : not linearized.'
        ENDIF
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        IF (AVOIDCIRCCONV) THEN
          WRITE(JETTXT,2900)
     f    'Avoid circular convolution : True.'
        ELSE
          WRITE(JETTXT,2900)
     f    'Avoid circular convolution : False.'
        ENDIF
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        IF (ONEOVERFWEIGHT) THEN
          WRITE(JETTXT,2900)
     f    'Weight                      : 1/F.'
        ELSE
          WRITE(JETTXT,2900)
     f    'Weight                      : 1.'
        ENDIF
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,2900)
     f  'Number of channels          :',NCH
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,2900)
     f  'Number of iterations        :',NOOFITERATIONS
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,3000)
     f  'Run time                    :',INT(TIME/60.0),':',
     f  TIME-60.0*INT(TIME/60.0)
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,3100)
     f  'Pulse FWHM                  :',PULSEFWHM,' ch.'
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        IF (ADDNOISETOPULSE) THEN
          WRITE(JETTXT,2900)
     f    'Add noise to pulse          : True.'
        ELSE
          WRITE(JETTXT,2900)
     f    'Add noise to pulse          : False.'
        ENDIF
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,3200)
     f  'Reduced chi-square          :',REDCHISQUARED
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,1300) ' '
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        IF ((.NOT.LOCKED(1)).AND.ADDNOISE.AND.
     f     (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
          WRITE(JETTXT,3200) 'Stray-light intensity :',
     f                 THETAEST(1),' +/- ',SQRT(ABS(GMATRIX(1,1)))
        ELSEIF (LOCKED(1)) THEN
          WRITE(JETTXT,3200) 'Stray-light intensity :',
     f                 THETAEST(1),' Locked'
        ELSEIF ((.NOT.LOCKED(1)).AND.
     f          ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
     f          (.NOT.ONEOVERFWEIGHT))) THEN
          WRITE(JETTXT,3200) 'Stray-light intensity :',
     f                 THETAEST(1)
        ENDIF
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        IF ((.NOT.LOCKED(2)).AND.ADDNOISE.AND.
     f     (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
          WRITE(JETTXT,3200) 'Pulse position        :',
     f               THETAEST(2)/DX,' +/- ',SQRT(ABS(GMATRIX(2,2)))/DX
        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,I4.4,5(A,I2.2))
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 (AVOIDCIRCCONV) 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,I4)
697
698         WRITE(*,3000) 'Run time                    :',INT(TIME/60.0),':',
699        f              TIME-60.0*INT(TIME/60.0)
700  3000   FORMAT(' ',A,I3,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 :',
```

```
     f                THETAEST(1),' +/- ',SQRT(ABS(GMATRIX(1,1)))
3200   FORMAT(' ',A,F12.5,A,F9.4)
      ELSEIF (LOCKED(1)) THEN
        WRITE(*,3200) 'Stray-light intensity :',
     f                THETAEST(1),' Locked'
      ELSEIF ((.NOT.LOCKED(1)).AND.
     f       ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
     f       (.NOT.ONEOVERFWEIGHT))) THEN
        WRITE(*,3200) 'Stray-light intensity :',
     f                THETAEST(1)
      ENDIF

      IF ((.NOT.LOCKED(2)).AND.ADDNOISE.AND.
     f   (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
        WRITE(*,3200) 'Pulse position        :',
     f               THETAEST(2)/DX,' +/- ',SQRT(ABS(GMATRIX(2,2)))/DX
      ELSEIF (LOCKED(2)) THEN
        WRITE(*,3200) 'Pulse position        :',
     f                THETAEST(2)/DX,' Locked'
      ELSEIF ((.NOT.LOCKED(2)).AND.
     f       ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
     f       (.NOT.ONEOVERFWEIGHT))) THEN
        WRITE(*,3200) 'Pulse position        :',
     f                THETAEST(2)/DX
      ENDIF

      DO 430 I=1,NOOFDECAYS
        IF ((.NOT.LOCKED(I+2)).AND.ADDNOISE.AND.
     f     (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
          WRITE(*,3300) 'Intensity, decay no ',I,' :',
     f    THETAEST(I+2),' +/- ',SQRT(ABS(GMATRIX(I+2,I+2)))
3300      FORMAT(' ',A,I1,A,F12.5,A,F9.4)
        ELSEIF (LOCKED(I+2)) THEN
          WRITE(*,3300) 'Intensity, decay no ',I,' :',
     f                  THETAEST(I+2),' Locked'
        ELSEIF ((.NOT.LOCKED(I+2)).AND.
     f         ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
     f         (.NOT.ONEOVERFWEIGHT))) THEN
          WRITE(*,3300) 'Intensity, decay no ',I,' :',
     f                  THETAEST(I+2)
        ENDIF
430   CONTINUE

      DO 440 I=1,NOOFDECAYS
        IF ((.NOT.LOCKED(I+2+NOOFDECAYS)).AND.ADDNOISE.AND.
     f     (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
          WRITE(*,3300) 'Decay rate, decay no ',I,':',
     f                THETAEST(I+2+NOOFDECAYS),' +/- ',
     f                SQRT(ABS(GMATRIX(I+2+NOOFDECAYS,I+2+NOOFDECAYS)))
        ELSEIF (LOCKED(I+2+NOOFDECAYS)) THEN
          WRITE(*,3300) 'Decay rate, decay no ',I,':',
     f                  THETAEST(I+2+NOOFDECAYS),' Locked'
        ELSEIF ((.NOT.LOCKED(I+2+NOOFDECAYS)).AND.
     f         ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
     f         (.NOT.ONEOVERFWEIGHT))) THEN
          WRITE(*,3300) 'Decay rate, decay no ',I,':',
     f                  THETAEST(I+2+NOOFDECAYS)
        ENDIF
440   CONTINUE

      IF ((.NOT.LOCKED(LR)).AND.ADDNOISE.AND.
     f   (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
        WRITE(*,3200) 'Background            :',
     f                THETAEST(LR),' +/- ',SQRT(ABS(GMATRIX(LR,LR)))
      ELSEIF (LOCKED(LR)) THEN
        WRITE(*,3200) 'Background            :',
     f                THETAEST(LR),' Locked'
      ELSEIF ((.NOT.LOCKED(LR)).AND.
     f       ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
     f       (.NOT.ONEOVERFWEIGHT))) THEN
        WRITE(*,3200) 'Background            :',
     f                THETAEST(LR)
      ENDIF

      WRITE(*,2800)

450   WRITE(*,*) 'New iteration? (Y/*):'
      READ(*,1300) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        GOTO 140
      ENDIF

      WRITE(*,3400) 'New calculation? (Y/*):'
3400  FORMAT(/' ',A)
      READ(*,1300) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        GOTO 10
      ELSE
        STOP
      ENDIF
      END

      REAL FUNCTION F(THETA,M,X)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)

      INTEGER    N,L,MAXNOOFDECAYS

      PARAMETER  ( N=512 )
      PARAMETER  ( MAXNOOFDECAYS=2, L=2*MAXNOOFDECAYS+3 )

      REAL       P(1:N),PCONVE(1:N,1:MAXNOOFDECAYS),THETA(1:*),DX
      REAL       X,SLASK1,SLASK2,SLX

      INTEGER    M,I,NCH,NCH2,DNCH2,NOOFDECAYS,IX1,IX2

      COMMON     /BLOCK1/ P,PCONVE
      COMMON     /BLOCK5/ NCH,NCH2,DNCH2,NOOFDECAYS,DX

      SLX=X/DX+1.0
      IX1=INT(SLX)
      IX1=MIN(IX1,NCH-1)
      IX1=MAX(IX1,1)
      IX2=IX1+1
      SLX=MIN(SLX,FLOAT(NCH))
      SLX=MAX(SLX,1.0)

      SLASK1=THETA(1)*P(IX1)+THETA(2*M+3)
      SLASK2=THETA(1)*P(IX2)+THETA(2*M+3)
```

```
      DO 10 I=1,M
        SLASK1=SLASK1+THETA(I+2)*PCONVE(IX1,I)
        SLASK2=SLASK2+THETA(I+2)*PCONVE(IX2,I)
10    CONTINUE
      F=SLASK1+(SLASK2-SLASK1)*(SLX-IX1)
      RETURN
      END

      REAL FUNCTION DFDTH(J,THETA,M,X)
C     Anders Persson, 1988
      IMPLICIT   LOGICAL (A-Z)

      INTEGER    N,L,MAXNOOFDECAYS

      PARAMETER  ( N=512 )
      PARAMETER  ( MAXNOOFDECAYS=2, L=2*MAXNOOFDECAYS+3 )

      REAL       P(1:N),PPRIM(1:N),PCONVE(1:N,1:MAXNOOFDECAYS)
      REAL       THETA(1:*),DX,X,PCONVXE(1:N,1:MAXNOOFDECAYS)
      REAL       PPRIMCONVE(1:N,1:MAXNOOFDECAYS),SLASK

      INTEGER    I,NCH,NCH2,DNCH2,J,NOOFDECAYS,M,IX

      COMMON     /BLOCK1/ P,PCONVE
      COMMON     /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
      COMMON     /BLOCK5/ NCH,NCH2,DNCH2,NOOFDECAYS,DX

      IX=NINT(X/DX+1.0)
      IX=MIN(IX,NCH)
      IX=MAX(IX,1)

      IF (J.EQ.1) THEN
        DFDTH=P(IX)
      ELSEIF (J.EQ.2) THEN
        SLASK=-THETA(1)*PPRIM(IX)
        DO 10 I=1,M
          SLASK=SLASK-THETA(I+2)*PPRIMCONVE(IX,I)
10      CONTINUE
        DFDTH=SLASK/DX
      ELSEIF (J.GE.3.AND.J.LE.M+2) THEN
        DFDTH=PCONVE(IX,J-2)
      ELSEIF (J.GE.M+3.AND.J.LE.2*M+2) THEN
        DFDTH=-THETA(J-M)*PCONVXE(IX,J-M-2)
      ELSEIF (J.EQ.2*M+3) THEN
        DFDTH=1.0
      ENDIF

      RETURN
      END

      REAL FUNCTION D2FDTH2(J,K,THETA,M,X)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)

      INTEGER    N,L,MAXNOOFDECAYS

      PARAMETER  ( N=512 )
      PARAMETER  ( MAXNOOFDECAYS=2, L=2*MAXNOOFDECAYS+3 )

      REAL       P(1:N),PPRIM(1:N),PBIS(1:N),X,SLASK
      REAL       PCONVE(1:N,1:MAXNOOFDECAYS)
      REAL       PCONVXE(1:N,1:MAXNOOFDECAYS)
      REAL       PCONVX2E(1:N,1:MAXNOOFDECAYS)
      REAL       PPRIMCONVE(1:N,1:MAXNOOFDECAYS)
      REAL       PPRIMCONVXE(1:N,1:MAXNOOFDECAYS)
      REAL       PBISCONVE(1:N,1:MAXNOOFDECAYS),DX,THETA(1:*)

      INTEGER    I,NCH,NCH2,DNCH2,J,K,M,NOOFDECAYS,IX

      LOGICAL    JGTK

      COMMON     /BLOCK1/ P,PCONVE
      COMMON     /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
      COMMON     /BLOCK3/ PCONVX2E,PPRIMCONVXE,PBIS,PBISCONVE
      COMMON     /BLOCK5/ NCH,NCH2,DNCH2,NOOFDECAYS,DX

      IX=NINT(X/DX+1.0)
      IX=MIN(IX,NCH)
      IX=MAX(IX,1)

      JGTK=J.GT.K
      IF (JGTK) THEN
        I=J
        J=K
        K=I
      ENDIF
      D2FDTH2=0.0

      IF (J.EQ.1) THEN
        IF (K.EQ.2) THEN
          D2FDTH2=-PPRIM(IX)/DX
        ENDIF
      ELSEIF (J.EQ.2) THEN
        IF (K.EQ.2) THEN
          SLASK=THETA(1)*PBIS(IX)
          DO 10 I=1,M
            SLASK=SLASK+THETA(I+2)*PBISCONVE(IX,I)
10        CONTINUE
          D2FDTH2=SLASK/(DX**2)
        ELSEIF (K.GE.3.AND.K.LE.M+2) THEN
          D2FDTH2=-PPRIMCONVE(IX,K-2)/DX
        ELSEIF (K.GE.M+3.AND.K.LE.2*M+2) THEN
          D2FDTH2=PPRIMCONVXE(IX,K-M-2)*THETA(K-M)/DX
        ENDIF
      ELSEIF (J.GE.3.AND.J.LE.M+2) THEN
        IF (K.EQ.M+J) THEN
          D2FDTH2=-PCONVXE(IX,J-2)
        ENDIF
      ELSEIF (J.GE.M+3.AND.J.LE.2*M+2) THEN
        IF (K.EQ.J) THEN
          D2FDTH2=THETA(J-M)*PCONVX2E(IX,J-M-2)
        ENDIF
      ENDIF

      IF (JGTK) THEN
        I=J
        J=K
        K=I
      ENDIF
```

```
958
959       RETURN
960       END
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       P0(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    FTP0(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/ P0,FTP0,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)=P0(I)
997 10      CONTINUE
998       ELSE
999         DO 20 I=1,NCH2
1000          SL1(I)=FTP0(I)
1001 20      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 30      CONTINUE
1007      ENDIF
1008
1009      IF (.NOT.LOCKED(2)) THEN
1010        DO 40 I=1,NCH2
1011          SL1(I)=FTP0(I)
1012 40      CONTINUE
1013        NDIFF=1
1014        CALL MOVEDIFF(SL1,W1,NCH2,THETA(2)/DX,NDIFF)
1015        DO 50 I=1,NCH
1016          PPRIM(I)=REAL(SL1(I))
1017 50      CONTINUE
1018        IF (.NOT.LINEARIZED) THEN
1019          DO 60 I=1,NCH2
1020            SL1(I)=FTP0(I)
1021 60        CONTINUE
1022          NDIFF=2
1023          CALL MOVEDIFF(SL1,W1,NCH2,THETA(2)/DX,NDIFF)
1024          DO 70 I=1,NCH
1025            PBIS(I)=REAL(SL1(I))
1026 70        CONTINUE
1027        ENDIF
1028      ENDIF
1029
1030      DO 100 J=1,NOOFDECAYS
1031        DO 80 I=1,NCH2
1032          FTE(I,J)=
1033     f      CMPLX(EXP(-MIN(THETA(NOOFDECAYS+J+2)*(I-1)*DX,87.0)),0.0)
1034 80      CONTINUE
1035        DO 90 I=NCH2+1,DNCH2
1036          FTE(I,J)=CMPLX(0.0,0.0)
1037 90      CONTINUE
1038 100   CONTINUE
1039
1040      DO 130 J=1,NOOFDECAYS
1041        IF (.NOT.LOCKED(NOOFDECAYS+J+2)) THEN
1042          DO 110 I=1,NCH2
1043            FTEX(I,J)=FTE(I,J)*(I-1)*DX
1044 110        CONTINUE
1045          DO 120 I=NCH2+1,DNCH2
1046            FTEX(I,J)=CMPLX(0.0,0.0)
1047 120        CONTINUE
1048        ENDIF
1049 130   CONTINUE
1050
1051      IF (.NOT.LINEARIZED) THEN
1052        DO 160 J=1,NOOFDECAYS
1053          IF (.NOT.LOCKED(NOOFDECAYS+J+2)) THEN
1054            DO 140 I=1,NCH2
1055              FTEX2(I,J)=FTEX(I,J)*(I-1)*DX
1056 140          CONTINUE
1057            DO 150 I=NCH2+1,DNCH2
1058              FTEX2(I,J)=CMPLX(0.0,0.0)
1059 150          CONTINUE
1060          ENDIF
1061 160     CONTINUE
1062      ENDIF
1063
1064      INV=.FALSE.
1065
1066      DO 170 J=1,NOOFDECAYS
1067        CALL FFT(FTE(1,J),W2,DNCH2,INV)
1068        IF (.NOT.LOCKED(NOOFDECAYS+J+2)) THEN
1069          CALL FFT(FTEX(1,J),W2,DNCH2,INV)
1070          IF (.NOT.LINEARIZED) THEN
1071            CALL FFT(FTEX2(1,J),W2,DNCH2,INV)
1072          ENDIF
1073        ENDIF
1074 170   CONTINUE
1075
1076      DO 180 I=1,NCH
1077        SL1(I)=CMPLX(P(I),0.0)
```

```
180   CONTINUE
      DO 190 I=NCH+1,DNCH2
        SL1(I)=CMPLX(0.0,0.0)
190   CONTINUE
      INV=.FALSE.
      CALL FFT(SL1,W2,DNCH2,INV)

      DO 260 J=1,NOOFDECAYS
        DO 200 I=1,DNCH2
          SL2(I)=SL1(I)*FTE(I,J)
200     CONTINUE
        INV=.TRUE.
        CALL FFT(SL2,W2,DNCH2,INV)
        DO 210 I=1,NCH
          PCONVE(I,J)=REAL(SL2(I))
210     CONTINUE
        IF (.NOT.LOCKED(NOOFDECAYS+J+2)) THEN
          DO 220 I=1,DNCH2
            SL2(I)=SL1(I)*FTEX(I,J)
220       CONTINUE
          INV=.TRUE.
          CALL FFT(SL2,W2,DNCH2,INV)
          DO 230 I=1,NCH
            PCONVXE(I,J)=REAL(SL2(I))
230       CONTINUE
          IF (.NOT.LINEARIZED) THEN
            DO 240 I=1,DNCH2
              SL2(I)=SL1(I)*FTEX2(I,J)
240         CONTINUE
            INV=.TRUE.
            CALL FFT(SL2,W2,DNCH2,INV)
            DO 250 I=1,NCH
              PCONVX2E(I,J)=REAL(SL2(I))
250         CONTINUE
          ENDIF
        ENDIF
260   CONTINUE

      IF (.NOT.LOCKED(2)) THEN
        DO 270 I=1,NCH
          SL1(I)=CMPLX(PPRIM(I),0.0)
270     CONTINUE
        DO 280 I=NCH+1,DNCH2
          SL1(I)=CMPLX(0.0,0.0)
280     CONTINUE
        INV=.FALSE.
        CALL FFT(SL1,W2,DNCH2,INV)

        DO 330 J=1,NOOFDECAYS
          DO 290 I=1,DNCH2
            SL2(I)=SL1(I)*FTE(I,J)
290       CONTINUE
          INV=.TRUE.
          CALL FFT(SL2,W2,DNCH2,INV)
          DO 300 I=1,NCH
            PPRIMCONVE(I,J)=REAL(SL2(I))
300       CONTINUE
          IF ((.NOT.LINEARIZED).AND.(.NOT.LOCKED(NOOFDECAYS+J+2))) THEN
            DO 310 I=1,DNCH2
              SL2(I)=SL1(I)*FTEX(I,J)
310         CONTINUE
            INV=.TRUE.
            CALL FFT(SL2,W2,DNCH2,INV)
            DO 320 I=1,NCH
              PPRIMCONVXE(I,J)=REAL(SL2(I))
320         CONTINUE
          ENDIF
330     CONTINUE
      ENDIF

      IF ((.NOT.LINEARIZED).AND.(.NOT.LOCKED(2))) THEN
        DO 340 I=1,NCH
          SL1(I)=CMPLX(PBIS(I),0.0)
340     CONTINUE
        DO 350 I=NCH+1,DNCH2
          SL1(I)=CMPLX(0.0,0.0)
350     CONTINUE
        INV=.FALSE.
        CALL FFT(SL1,W2,DNCH2,INV)

        DO 380 J=1,NOOFDECAYS
          DO 360 I=1,DNCH2
            SL2(I)=SL1(I)*FTE(I,J)
360       CONTINUE
          INV=.TRUE.
          CALL FFT(SL2,W2,DNCH2,INV)
          DO 370 I=1,NCH
            PBISCONVE(I,J)=REAL(SL2(I))
370       CONTINUE
380     CONTINUE
      ENDIF

      RETURN
      END

      SUBROUTINE MOVEDIFF(A,W,N,DC,NDIFF)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    N,K,NDIFF
      COMPLEX    A(0:*),W(0:*)
      REAL       DC,PHSTEP,TWOPI,SLASK1,SLASK2
      PARAMETER  ( TWOPI=6.283185308 )
      LOGICAL    INV

      PHSTEP=-TWOPI*DC/FLOAT(N)
      DO 10 K=0,N/2
        A(K)=A(K)*CEXP(CMPLX(0.0,PHSTEP*FLOAT(K)))
10    CONTINUE
      DO 20 K=N/2+1,N-1
        A(K)=A(K)*CEXP(CMPLX(0.0,PHSTEP*FLOAT(K-N)))
20    CONTINUE

      IF (NDIFF.GT.0) THEN
        PHSTEP=TWOPI/FLOAT(N)
        SLASK1=PHSTEP*FLOAT(N)/20.0
        A(0)=CMPLX(0.0,0.0)
        DO 30 K=1,N/2
          SLASK2=MIN(PHSTEP*FLOAT(K),SLASK1)
          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

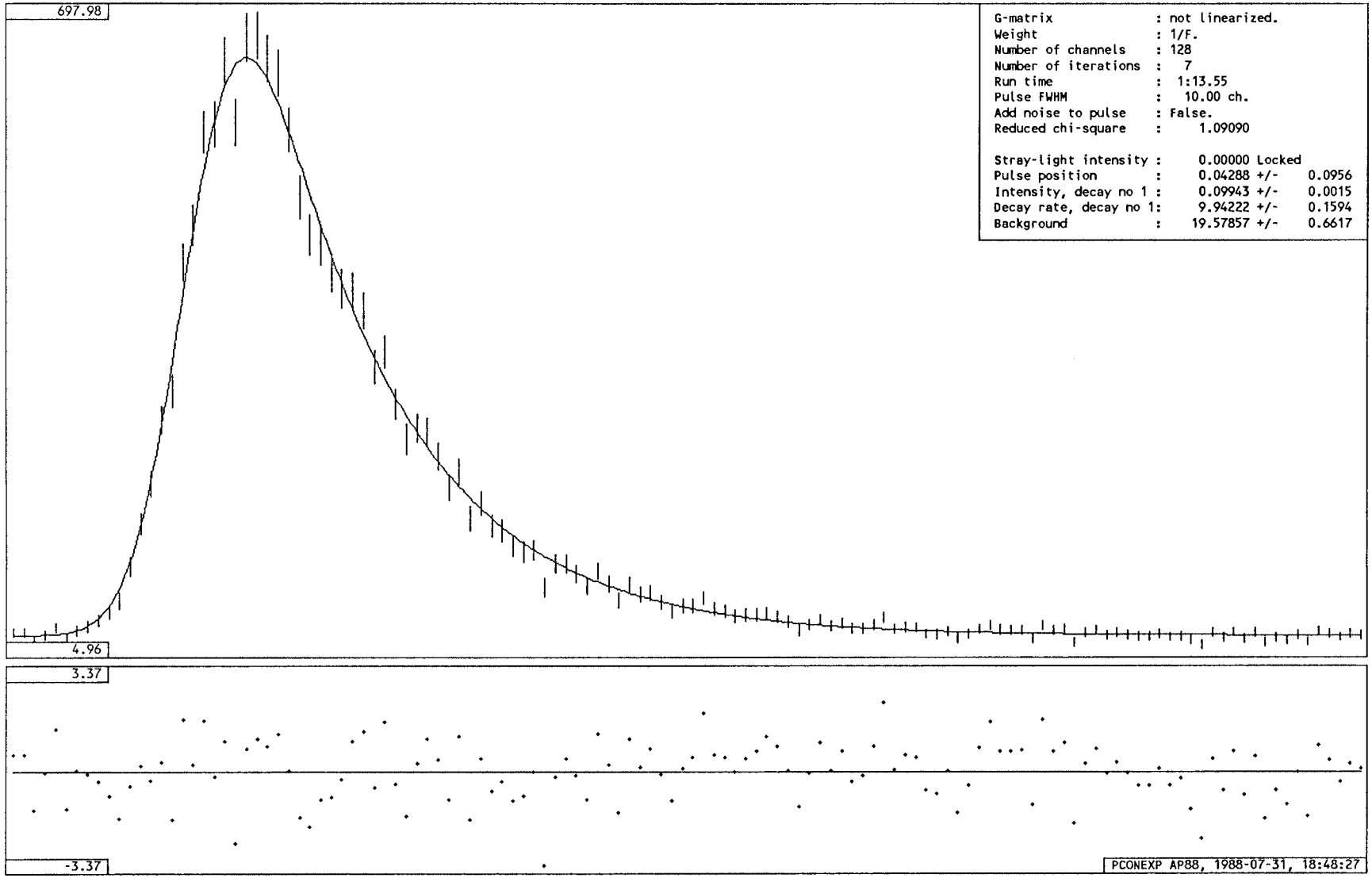
Programmet utför exakt samma beräkningar som programmet PCONEXPA men det finns ingen möjlighet att undvika cirkulär faltning. Detta är ingen allvarlig restriktion (se kommentar till PCONEXPA-programmet) men det innebär en halvering av dataareornas storlek och en minskning av exekveringstiden. Dessutom 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 ien 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
4.96
3.37
-3.37
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
PCONEXP AP88, 1988-07-31, 18:48:27

Kommentarer till programlistningen:

Rad	Kommentar
1-939	Huvudprogrammet, funktionsberäkningsrutinerna och derivata-rutinerna 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.

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: PCONEXP.FOR Options: /LBZ          08/02/88 14:33:48

      PROGRAM PCONEXP
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)

      INTEGER    N,NHALF,L,MAXNOOFITERATIONS,MAXNOOFDECAYS

      PARAMETER  ( N=512, NHALF=N/2, MAXNOOFITERATIONS=100 )
      PARAMETER  ( MAXNOOFDECAYS=3, L=2*MAXNOOFDECAYS+3 )

      REAL       Y(1:N),X(1:N),DX,DET,NORMAL,THETAEST(1:L)
      REAL       RESIDUE(1:N),SIGMA(1:N),F,PULSEFWHM,JX,JY,JXL,JYL,DJY
      REAL       GMATRIX(1:L,1:L),GVECTOR(1:L),DELTATHETA(1:L)
      REAL       SLASK,DFDTH,D2FDTH2,THETAMIN(1:L),THETAMAX(1:L)
      REAL       SL1,SL2,SL3,SL4,SL5,TIME,STEP,REDCHISQUARED
      REAL       PO(1:N),P(1:N),PPRIM(1:N),PBIS(1:N)
      REAL       PCONVE(1:N,1:MAXNOOFDECAYS)
      REAL       PCONVXE(1:N,1:MAXNOOFDECAYS)
      REAL       PCONVX2E(1:N,1:MAXNOOFDECAYS)
      REAL       PPRIMCONVE(1:N,1:MAXNOOFDECAYS)
      REAL       PPRIMCONVXE(1:N,1:MAXNOOFDECAYS)
      REAL       PBISCONVE(1:N,1:MAXNOOFDECAYS)

      COMPLEX    FTPO(1:N),S1(1:N),S2(1:N),W(1:NHALF)

      INTEGER    R,K,I,NCH,NCH2,NOOFITERATIONS,J,NOOFDECAYS,LR
      INTEGER    NOOFVARIABLEPAR,NDIFF

      INTEGER*2  HO,MI,SE,HN,ANS,KBDCHK,KBDINC,YEAR,MONTH,DAY

      CHARACTER  SV*1,LOCKEDORFREE(1:L)*6,JETTXT*50

      LOGICAL    SING,ERRORBAR,LINEARIZED,LOCKED(1:L),ADDNOISE,INV
      LOGICAL    LOCKTOLINEARIZED,LOCKWEIGHT,SLLOCKED(1:L)
      LOGICAL    ONEOVERFWEIGHT,ADDNOISETOPULSE

      EXTERNAL   F

      COMMON     /BLOCK1/ P,PCONVE
      COMMON     /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
      COMMON     /BLOCK3/ PCONVX2E,PPRIMCONVXE,PBIS,PBISCONVE
      COMMON     /BLOCK4/ PO,FTPO,S1,S2,W
      COMMON     /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX

10    WRITE(*,1000) 'No of decays (1..',MAXNOOFDECAYS,'):'
1000  FORMAT(' ',A,I2,A)
      READ(*,*,ERR=10) NOOFDECAYS
      IF (NOOFDECAYS.LT.1.OR.NOOFDECAYS.GT.MAXNOOFDECAYS) THEN
        GOTO 10
      ENDIF
      LR=2*NOOFDECAYS+3

20    WRITE(*,1100) 'No of channels (',LR,'..',N,'):'
1100  FORMAT(' ',A,I2,A,I4,A)
      READ(*,*,ERR=20) NCH
      IF (NCH.LT.LR.OR.NCH.GT.N) THEN
        GOTO 20
      ENDIF
      DX=1.0/(NCH-1)

30    WRITE(*,*) 'Stray-light intensity:'
      READ(*,*,ERR=30) THETAEST(1)
      IF (THETAEST(1).LT.0.0) THEN
        GOTO 30
      ENDIF

40    WRITE(*,*) 'Pulse position (channels):'
      READ(*,*,ERR=40) THETAEST(2)
      THETAEST(2)=THETAEST(2)*DX

      DO 70 I=1,NOOFDECAYS
50      WRITE(*,1200) 'Intensity, decay no ',I,': '
1200    FORMAT(' ',A,I1,A)
        READ(*,*,ERR=50) THETAEST(I+2)
        IF (THETAEST(I+2).LE.0.0) THEN
          GOTO 50
        ENDIF
60      WRITE(*,1200) 'Decay rate, decay no ',I,': '
        READ(*,*,ERR=60) THETAEST(I+NOOFDECAYS+2)
        IF (THETAEST(I+NOOFDECAYS+2).LE.0.0) THEN
          GOTO 60
        ENDIF
70    CONTINUE

80    WRITE(*,*) 'Background: '
      READ(*,*,ERR=80) THETAEST(LR)

      WRITE(*,*) 'Add noise to data? (Y/*): '
      READ(*,1300) SV
1300  FORMAT(A)
      ADDNOISE=SV.EQ.'Y'.OR.SV.EQ.'y'
      ERRORBAR=ADDNOISE
      IF (ADDNOISE) THEN
        WRITE(*,*) 'Plot errorbars? (Y/*): '
        READ(*,1300) SV
        ERRORBAR=SV.EQ.'Y'.OR.SV.EQ.'y'
      ENDIF

      NCH2=LOG(NCH-0.5)/LOG(2.0)
      NCH2=2**(NCH2+1)

      DO 90 I=1,LR
        LOCKED(I)       =.FALSE.
        SLLOCKED(I)     =.TRUE.
        LOCKEDORFREE(I)='Free  '
90    CONTINUE

      LOCKED(1)           =.TRUE.
      LOCKED(2)           =.TRUE.
      LOCKEDORFREE(1)     ='Locked'
      LOCKEDORFREE(2)     ='Locked'
      LOCKTOLINEARIZED    =.FALSE.
      LINEARIZED          =.TRUE.
      ONEOVERFWEIGHT      =.TRUE.

100   WRITE(*,*) 'Pulse FWHM (channels):'
      READ(*,*,ERR=100) PULSEFWHM
```

```
      IF (PULSEFWHM.LT.0.1.OR.PULSEFWHM.GT.NCH*0.5) THEN
        GOTO 100
      ENDIF

      WRITE(*,*) 'Add noise to pulse? (Y/*):'
      READ(*,1300) SV
      ADDNOISETOPULSE=SV.EQ.'Y'.OR.SV.EQ.'y'

      DO 110 I=1,NCH
        PO(I)=EXP(-MIN((LOG(16.0)/PULSEFWHM**2)*(I-NCH/7.0)**2,87.0))
        PO(I)=1000.0*PO(I)
        IF (ADDNOISETOPULSE) THEN
          PO(I)=PO(I)+NORMAL(0.0,10.0)
        ENDIF
        P(I)=PO(I)
        FTPO(I)=CMPLX(PO(I),0.0)
110   CONTINUE
      DO 120 I=NCH+1,NCH2
        FTPO(I)=CMPLX(0.0,0.0)
120   CONTINUE

      CALL CALCW(W,NCH2)
      INV=.FALSE.
      CALL FFT(FTPO,W,NCH2,INV)
      IF (THETAEST(1).NE.0.0) THEN
        DO 130 I=1,NCH2
          S1(I)=FTPO(I)
130     CONTINUE
        NDIFF=0
        CALL MOVEDIFF(S1,NCH2,THETAEST(2)/DX,NDIFF)
        INV=.TRUE.
        CALL FFT(S1,W,NCH2,INV)
        DO 140 I=1,NCH
          P(I)=REAL(S1(I))
140     CONTINUE
      ENDIF
      CALL CALCCONV(THETAEST,SLLOCKED,LINEARIZED)

      DO 150 I=1,NCH
        X(I)=(I-1)*DX
        SLASK=F(THETAEST,NOOFDECAYS,X(I))
        IF (ADDNOISE) THEN
          Y(I)=NORMAL(SLASK,SQRT(ABS(SLASK)))
          SIGMA(I)=SQRT(ABS(Y(I)))
        ELSE
          Y(I)=SLASK
          SIGMA(I)=0.0
        ENDIF
150   CONTINUE

160   CALL GMODE
      CALL TMODE

      WRITE(*,1400) 'Toggle Free/Locked with <shift>+Function key.'
1400  FORMAT(////,T10,A,/)

      WRITE(*,1500) THETAEST(1),LOCKEDORFREE(1)
1500  FORMAT(T10,'F1 : Stray-light intensity',T45,F12.4,',',T60,A)

      WRITE(*,1600) THETAEST(2)/DX,LOCKEDORFREE(2)
1600  FORMAT(T10,'F2 : Pulse position',T45,G12.4,',',T60,A)

      DO 170 I=1,NOOFDECAYS
        WRITE(*,1700) I+2,I,THETAEST(I+2),LOCKEDORFREE(I+2)
1700    FORMAT(T10,'F',I1,' : Intensity , decay no ',
     f          I1,T45,F12.4,',',T60,A)
170   CONTINUE
      DO 180 I=1,NOOFDECAYS
        WRITE(*,1800) I+NOOFDECAYS+2,I,THETAEST(I+NOOFDECAYS+2),
     f                LOCKEDORFREE(I+NOOFDECAYS+2)
1800    FORMAT(T10,'F',I1,' : Decay rate, decay no ',
     f          I1,T45,F12.4,',',T60,A)
180   CONTINUE
      WRITE(*,1900) LR,THETAEST(LR),LOCKEDORFREE(LR)
1900  FORMAT(T10,'F',I1,' : Background ',T45,F12.4,',',T60,A)
      WRITE(*,2000) 'F10: Calculate.'
2000  FORMAT(T10,A)
      WRITE(*,2100) '<Alt>+F10: Preview.'
2100  FORMAT(T4,A)
      IF (LOCKTOLINEARIZED) THEN
        WRITE(*,2200) '<Shift>+F10: G-matrix = linearized.'
      ELSE
        WRITE(*,2200) '<Shift>+F10: G-matrix = not linearized.'
      ENDIF
2200  FORMAT(T2,A)
      IF (ONEOVERFWEIGHT) THEN
        WRITE(*,2100) '<Alt>+F1 : Weight = 1/F.'
      ELSE
        WRITE(*,2100) '<Alt>+F1 : Weight = 1.'
      ENDIF
      WRITE(*,2300) ' '
2300  FORMAT(T3,A)

190   ANS=KBDCHK()
      IF (ANS.NE.0) THEN
        GOTO 190
      ENDIF
      ANS=KBDINC()

      IF (ANS.GE.1084.AND.ANS.LE.1083+LR) THEN
        I=ANS-1083
        LOCKED(I)=.NOT.LOCKED(I)
        LOCKEDORFREE(I)='Free  '
        IF (LOCKED(I)) THEN
          LOCKEDORFREE(I)='Locked'
        ENDIF
        GOTO 160
      ELSEIF (ANS.EQ.1113) THEN
        WRITE(*,2000) 'Preview.'
        IF (THETAEST(1).NE.0.0) THEN
          DO 200 I=1,NCH2
            S1(I)=FTPO(I)
200       CONTINUE
          NDIFF=0
          CALL MOVEDIFF(S1,NCH2,THETAEST(2)/DX,NDIFF)
          INV=.TRUE.
          CALL FFT(S1,W,NCH2,INV)
          DO 210 I=1,NCH
            P(I)=REAL(S1(I))
210       CONTINUE
```

```
238         ENDIF
239         CALL CALCCONV(THETAEST,SLLOCKED,LINEARIZED)
240         CALL SCRPLOT(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,I1,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(I)
282           IF (THETAEST(I).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(I)
289           IF (THETAEST(I).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,MI,SE,HN)
323       TIME=HO*3600.0+MI*60.0+SE+HN*0.01
324
325       WRITE(*,2600) 'It:',0
326 2600  FORMAT(' ',49('-'),/,' ',A,I4,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,I1,A,G12.5,A,I1,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 CALCCONV(THETAEST,LOCKED,LINEARIZED)
350         DO 340 I=1,NCH
351           IF (ONEOVERFWEIGHT) THEN
352             IF (LOCKWEIGHT) THEN
353               SL1=-2.0/MAX(Y(I),1.0)
354             ELSE
355               SL1=-2.0/F(THETAEST,NOOFDECAYS,X(I))
356             ENDIF
357           ELSE
```

```
358             SL1=-2.0
359           ENDIF
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                   ENDIF
373                   GMATRIX(K,R)=GMATRIX(R,K)
374                 ENDIF
375 320           CONTINUE
376             ENDIF
377 330       CONTINUE
378 340     CONTINUE
379
380       DO 350 R=1,LR
381         IF (LOCKED(R)) THEN
382           GMATRIX(R,R)=1.0
383         ENDIF
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       ENDIF
391
392       CALL MULTCMAT(GMATRIX,-1.0,LR,LR,L,L)
393       CALL MULTMAT(GMATRIX,GVECTOR,DELTATHETA,LR,LR,1,L,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         ENDIF
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         ENDIF
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       ENDIF
```

```
418       IF (.NOT.LOCKED(2)) THEN
419         WRITE(*,2700) 'Pulse position  :',THETAEST(2)/DX
420       ENDIF
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       ENDIF
428
429       IF (STEP.LE.0.5E-4.AND.(.NOT.LINEARIZED)) THEN
430         NOOFITERATIONS=J
431         GOTO 410
432       ENDIF
433       IF (STEP.LE.0.5E-2.AND.LINEARIZED) THEN
434         LINEARIZED=.FALSE.
435       ENDIF
436       IF (LOCKTOLINEARIZED) THEN
437         LINEARIZED=.TRUE.
438       ENDIF
439       IF (STEP.LE.0.5E-4.AND.LOCKTOLINEARIZED) THEN
440         NOOFITERATIONS=J
441         GOTO 410
442       ENDIF
443       IF (LOCKWEIGHT.AND.STEP.LE.0.5E-2) THEN
444         LOCKWEIGHT=.FALSE.
445       ENDIF
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         ENDIF
456       ENDIF
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       ENDIF
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 SCRPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
477     f              NOOFDECAYS,ERRORBAR)
```

```
440   WRITE(*,*) 'Plot on Roland plotter? (Y/*): '
      READ(*,1300) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CALL ROLPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
     f               NOOFDECAYS,ERRORBAR)
        GOTO 440
      ENDIF

      REDCHISQUARED=0.0
      DO 450 I=1,NCH
        REDCHISQUARED=REDCHISQUARED+RESIDUE(I)**2
450   CONTINUE
      NOOFVARIABLEPAR=0
      DO 460 I=1,LR
        IF (.NOT.LOCKED(I)) THEN
          NOOFVARIABLEPAR=NOOFVARIABLEPAR+1
        ENDIF
460   CONTINUE
      IF (NCH.NE.NOOFVARIABLEPAR) THEN
        REDCHISQUARED=REDCHISQUARED/(NCH-NOOFVARIABLEPAR)
      ENDIF

470   WRITE(*,*) 'Plot on HP-Laserjet? (Y/*): '
      READ(*,1300) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        CALL HPSTART('L')
        CALL JETPLOT(Y,SIGMA,X,RESIDUE,NCH,F,THETAEST,
     f               NOOFDECAYS,ERRORBAR)
        JX=800
        JXL=1119-JX
        JYL=(13+NOOFDECAYS*2)*13
        JY=719-JYL
        CALL HPBOX(JX,JY,JXL,JYL)
        JX=806
        JY=719-16
        DJY=-13

        IF (LOCKTOLINEARIZED) THEN
          WRITE(JETTXT,2900) 'G-matrix              : linearized.'
        ELSE
          WRITE(JETTXT,2900) 'G-matrix              : not linearized.'
        ENDIF
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        IF (ONEOVERFWEIGHT) THEN
          WRITE(JETTXT,2900) 'Weight                : 1/F.'
        ELSE
          WRITE(JETTXT,2900) 'Weight                : 1.'
        ENDIF
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,2900) 'Number of channels    :',NCH
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,2900) 'Number of iterations  :',NOOFITERATIONS
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,3000) 'Run time              :',INT(TIME/60.0),':',
     f                     TIME-60.0*INT(TIME/60.0)
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,3100) 'Pulse FWHM            :',PULSEFWHM,' ch.'
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        IF (ADDNOISETOPULSE) THEN
          WRITE(JETTXT,2900) 'Add noise to pulse    : True.'
        ELSE
          WRITE(JETTXT,2900) 'Add noise to pulse    : False.'
        ENDIF
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,3200) 'Reduced chi-square    :',REDCHISQUARED
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        WRITE(JETTXT,1300) '  '
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        IF ((.NOT.LOCKED(1)).AND.ADDNOISE.AND.
     f    (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
          WRITE(JETTXT,3200) 'Stray-light intensity :',
     f                       THETAEST(1),' +/- ',SQRT(ABS(GMATRIX(1,1)))
        ELSEIF (LOCKED(1)) THEN
          WRITE(JETTXT,3200) 'Stray-light intensity :',
     f                       THETAEST(1),' Locked'
        ELSEIF ((.NOT.LOCKED(1)).AND.
     f          ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
     f          (.NOT.ONEOVERFWEIGHT))) THEN
          WRITE(JETTXT,3200) 'Stray-light intensity :',
     f                       THETAEST(1)
        ENDIF
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        IF ((.NOT.LOCKED(2)).AND.ADDNOISE.AND.
     f      (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
          WRITE(JETTXT,3200) 'Pulse position        :',
     f         THETAEST(2)/DX,' +/- ',SQRT(ABS(GMATRIX(2,2)))/DX
        ELSEIF (LOCKED(2)) THEN
          WRITE(JETTXT,3200) 'Pulse position        :',
     f                       THETAEST(2)/DX,' Locked'
        ELSEIF ((.NOT.LOCKED(2)).AND.
     f          ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
     f          (.NOT.ONEOVERFWEIGHT))) THEN
          WRITE(JETTXT,3200) 'Pulse position        :',
     f                       THETAEST(2)/DX
        ENDIF
        CALL HPPRINT(JX,JY,50,JETTXT)
        JY=JY+DJY

        DO 480 I=1,NOOFDECAYS
```

```
598           IF ((.NOT.LOCKED(I+2)).AND.ADDNOISE.AND.
599      f       (.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      f           ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
607      f           (.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      f         (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
618             WRITE(JETTXT,3300) 'Decay rate, decay no ',I,':',
619      f                  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      f           ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
626      f           (.NOT.ONEOVERFWEIGHT))) THEN
627             WRITE(JETTXT,3300) 'Decay rate, decay no ',I,':',
628      f                    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      f       (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
636             WRITE(JETTXT,3200) 'Background             :',
637      f                    THETAEST(LR),' +/- ',SQRT(ABS(GMATRIX(LR,LR)))
638           ELSEIF (LOCKED(LR)) THEN
639             WRITE(JETTXT,3200) 'Background             :',
640      f                    THETAEST(LR),' Locked'
641           ELSEIF ((.NOT.LOCKED(LR)).AND.
642      f           ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
643      f           (.NOT.ONEOVERFWEIGHT))) THEN
644             WRITE(JETTXT,3200) 'Background             :',
645      f                    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,I4.4,5(A,I2.2))
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,I4)
678
679         WRITE(*,3000) 'Run time            :',INT(TIME/60.0),':',
680      f              TIME-60.0*INT(TIME/60.0)
681  3000   FORMAT(' ',A,I3,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      f    (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
698           WRITE(*,3200) 'Stray-light intensity :',
699      f                  THETAEST(1),' +/- ',SQRT(ABS(GMATRIX(1,1)))
700         ELSEIF (LOCKED(1)) THEN
701           WRITE(*,3200) 'Stray-light intensity :',
702      f                  THETAEST(1),' Locked'
703         ELSEIF ((.NOT.LOCKED(1)).AND.
704      f       ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
705      f       (.NOT.ONEOVERFWEIGHT))) THEN
706           WRITE(*,3200) 'Stray-light intensity :',
707      f                  THETAEST(1)
708         ENDIF
709
710         IF ((.NOT.LOCKED(2)).AND.ADDNOISE.AND.
711      f    (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
712           WRITE(*,3200) 'Pulse position      :',
713      f                 THETAEST(2)/DX,' +/- ',SQRT(ABS(GMATRIX(2,2)))/DX
714         ELSEIF (LOCKED(2)) THEN
715           WRITE(*,3200) 'Pulse position      :',
716      f                  THETAEST(2)/DX,' Locked'
717         ELSEIF ((.NOT.LOCKED(2)).AND.
```

```
     f       ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
     f       (.NOT.ONEOVERFWEIGHT))) THEN
        WRITE(*,3200) 'Pulse position        :',
     f                THETAEST(2)/DX
      ENDIF

      DO 500 I=1,NOOFDECAYS
        IF ((.NOT.LOCKED(I+2)).AND.ADDNOISE.AND.
     f     (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
          WRITE(*,3300) 'Intensity, decay no ',I,' :',
     f    THETAEST(I+2),' +/- ',SQRT(ABS(GMATRIX(I+2,I+2)))
3300      FORMAT(' ',A,I1,A,F12.5,A,F9.4)
        ELSEIF (LOCKED(I+2)) THEN
          WRITE(*,3300) 'Intensity, decay no ',I,' :',
     f                  THETAEST(I+2),' Locked'
        ELSEIF ((.NOT.LOCKED(I+2)).AND.
     f         ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
     f         (.NOT.ONEOVERFWEIGHT))) THEN
          WRITE(*,3300) 'Intensity, decay no ',I,' :',
     f                  THETAEST(I+2)
        ENDIF
500   CONTINUE

      DO 510 I=1,NOOFDECAYS
        IF ((.NOT.LOCKED(I+2+NOOFDECAYS)).AND.ADDNOISE.AND.
     f     (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
          WRITE(*,3300) 'Decay rate, decay no ',I,':',
     f               THETAEST(I+2+NOOFDECAYS),' +/- ',
     f               SQRT(ABS(GMATRIX(I+2+NOOFDECAYS,I+2+NOOFDECAYS)))
        ELSEIF (LOCKED(I+2+NOOFDECAYS)) THEN
          WRITE(*,3300) 'Decay rate, decay no ',I,':',
     f                  THETAEST(I+2+NOOFDECAYS),' Locked'
        ELSEIF ((.NOT.LOCKED(I+2+NOOFDECAYS)).AND.
     f         ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
     f         (.NOT.ONEOVERFWEIGHT))) THEN
          WRITE(*,3300) 'Decay rate, decay no ',I,':',
     f                  THETAEST(I+2+NOOFDECAYS)
        ENDIF
510   CONTINUE

      IF ((.NOT.LOCKED(LR)).AND.ADDNOISE.AND.
     f   (.NOT.LOCKTOLINEARIZED).AND.ONEOVERFWEIGHT) THEN
        WRITE(*,3200) 'Background            :',
     f                THETAEST(LR),' +/- ',SQRT(ABS(GMATRIX(LR,LR)))
      ELSEIF (LOCKED(LR)) THEN
        WRITE(*,3200) 'Background            :',
     f                THETAEST(LR),' Locked'
      ELSEIF ((.NOT.LOCKED(LR)).AND.
     f       ((.NOT.ADDNOISE).OR.LOCKTOLINEARIZED.OR.
     f       (.NOT.ONEOVERFWEIGHT))) THEN
        WRITE(*,3200) 'Background            :',
     f                THETAEST(LR)
      ENDIF

      WRITE(*,2500)

520   WRITE(*,*) 'New iteration? (Y/*):'
      READ(*,1300) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        GOTO 160
      ENDIF

      WRITE(*,3400) 'New calculation? (Y/*):'
3400  FORMAT(/' ',A)
      READ(*,1300) SV
      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        GOTO 10
      ELSE
        STOP
      ENDIF
      END

      REAL FUNCTION F(THETA,M,X)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)

      INTEGER    N,L,MAXNOOFDECAYS

      PARAMETER  ( N=512 )
      PARAMETER  ( MAXNOOFDECAYS=3, L=2*MAXNOOFDECAYS+3 )

      REAL       P(1:N),PCONVE(1:N,1:MAXNOOFDECAYS),THETA(1:*),DX
      REAL       X,SLASK1,SLASK2,SLX

      INTEGER    M,I,NCH,NCH2,NOOFDECAYS,IX1,IX2

      COMMON     /BLOCK1/ P,PCONVE
      COMMON     /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX

      SLX=X/DX+1.0
      IX1=INT(SLX)
      IX1=MIN(IX1,NCH-1)
      IX1=MAX(IX1,1)
      IX2=IX1+1
      SLX=MIN(SLX,FLOAT(NCH))
      SLX=MAX(SLX,1.0)

      SLASK1=THETA(1)*P(IX1)+THETA(2*M+3)
      SLASK2=THETA(1)*P(IX2)+THETA(2*M+3)
      DO 10 I=1,M
        SLASK1=SLASK1+THETA(I+2)*PCONVE(IX1,I)
        SLASK2=SLASK2+THETA(I+2)*PCONVE(IX2,I)
10    CONTINUE
      F=SLASK1+(SLASK2-SLASK1)*(SLX-IX1)
      RETURN
      END

      REAL FUNCTION DFDTH(J,THETA,M,X)
C     Anders Persson, 1988
      IMPLICIT   LOGICAL (A-Z)

      INTEGER    N,L,MAXNOOFDECAYS

      PARAMETER  ( N=512 )
      PARAMETER  ( MAXNOOFDECAYS=3, L=2*MAXNOOFDECAYS+3 )

      REAL       P(1:N),PPRIM(1:N),PCONVE(1:N,1:MAXNOOFDECAYS)
      REAL       THETA(1:*),DX,X,PCONVXE(1:N,1:MAXNOOFDECAYS)
      REAL       PPRIMCONVE(1:N,1:MAXNOOFDECAYS),SLASK
```

```
      INTEGER    I,NCH,NCH2,J,NOOFDECAYS,M,IX

      COMMON     /BLOCK1/ P,PCONVE
      COMMON     /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
      COMMON     /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX

      IX=NINT(X/DX+1.0)
      IX=MIN(IX,NCH)
      IX=MAX(IX,1)

      IF (J.EQ.1) THEN
        DFDTH=P(IX)
      ELSEIF (J.EQ.2) THEN
        SLASK=-THETA(1)*PPRIM(IX)
        DO 10 I=1,M
          SLASK=SLASK-THETA(I+2)*PPRIMCONVE(IX,I)
10      CONTINUE
        DFDTH=SLASK/DX
      ELSEIF (J.GE.3.AND.J.LE.M+2) THEN
        DFDTH=PCONVE(IX,J-2)
      ELSEIF (J.GE.M+3.AND.J.LE.2*M+2) THEN
        DFDTH=-THETA(J-M)*PCONVXE(IX,J-M-2)
      ELSEIF (J.EQ.2*M+3) THEN
        DFDTH=1.0
      ENDIF
      RETURN
      END

      REAL FUNCTION D2FDTH2(J,K,THETA,M,X)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)

      INTEGER    N,L,MAXNOOFDECAYS

      PARAMETER  ( N=512 )
      PARAMETER  ( MAXNOOFDECAYS=3, L=2*MAXNOOFDECAYS+3 )

      REAL       P(1:N),PPRIM(1:N),PBIS(1:N),X,SLASK
      REAL       PCONVE(1:N,1:MAXNOOFDECAYS)
      REAL       PCONVXE(1:N,1:MAXNOOFDECAYS)
      REAL       PCONVX2E(1:N,1:MAXNOOFDECAYS)
      REAL       PPRIMCONVE(1:N,1:MAXNOOFDECAYS)
      REAL       PPRIMCONVXE(1:N,1:MAXNOOFDECAYS)
      REAL       PBISCONVE(1:N,1:MAXNOOFDECAYS),DX,THETA(1:*)

      INTEGER    I,NCH,NCH2,J,K,M,NOOFDECAYS,IX

      LOGICAL    JGTK

      COMMON     /BLOCK1/ P,PCONVE
      COMMON     /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
      COMMON     /BLOCK3/ PCONVX2E,PPRIMCONVXE,PBIS,PBISCONVE
      COMMON     /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX

      IX=NINT(X/DX+1.0)
      IX=MIN(IX,NCH)
      IX=MAX(IX,1)

      JGTK=J.GT.K
      IF (JGTK) THEN
        I=J
        J=K
        K=I
      ENDIF
      D2FDTH2=0.0

      IF (J.EQ.1) THEN
        IF (K.EQ.2) THEN
          D2FDTH2=-PPRIM(IX)/DX
        ENDIF
      ELSEIF (J.EQ.2) THEN
        IF (K.EQ.2) THEN
          SLASK=THETA(1)*PBIS(IX)
          DO 10 I=1,M
            SLASK=SLASK+THETA(I+2)*PBISCONVE(IX,I)
10        CONTINUE
          D2FDTH2=SLASK/(DX**2)
        ELSEIF (K.GE.3.AND.K.LE.M+2) THEN
          D2FDTH2=-PPRIMCONVE(IX,K-2)/DX
        ELSEIF (K.GE.M+3.AND.K.LE.2*M+2) THEN
          D2FDTH2=PPRIMCONVXE(IX,K-M-2)*THETA(K-M)/DX
        ENDIF
      ELSEIF (J.GE.3.AND.J.LE.M+2) THEN
        IF (K.EQ.M+J) THEN
          D2FDTH2=-PCONVXE(IX,J-2)
        ENDIF
      ELSEIF (J.GE.M+3.AND.J.LE.2*M+2) THEN
        IF (K.EQ.J) THEN
          D2FDTH2=THETA(J-M)*PCONVX2E(IX,J-M-2)
        ENDIF
      ENDIF

      IF (JGTK) THEN
        I=J
        J=K
        K=I
      ENDIF

      RETURN
      END

      SUBROUTINE CALCCONV(THETA,LOCKED,LINEARIZED)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)

      INTEGER    N,NHALF,L,MAXNOOFDECAYS

      PARAMETER  ( N=512, NHALF=N/2 )
      PARAMETER  ( MAXNOOFDECAYS=3, L=2*MAXNOOFDECAYS+3 )

      REAL       P0(1:N),P(1:N),PPRIM(1:N),PBIS(1:N)
      REAL       PCONVE(1:N,1:MAXNOOFDECAYS)
      REAL       PCONVXE(1:N,1:MAXNOOFDECAYS)
      REAL       PCONVX2E(1:N,1:MAXNOOFDECAYS)
      REAL       PPRIMCONVE(1:N,1:MAXNOOFDECAYS)
      REAL       PPRIMCONVXE(1:N,1:MAXNOOFDECAYS)
      REAL       PBISCONVE(1:N,1:MAXNOOFDECAYS),DX,THETA(1:*)
```

```
      COMPLEX    FTP0(1:N),SL1(1:N),SL2(1:N),W(1:NHALF)

      INTEGER    NCH,NCH2,XP,NDIFF,NOOFDECAYS

      INTEGER*2  I,J

      LOGICAL    LINEARIZED,LOCKED(1:*),INV

      COMMON     /BLOCK1/ P,PCONVE
      COMMON     /BLOCK2/ PCONVXE,PPRIM,PPRIMCONVE
      COMMON     /BLOCK3/ PCONVX2E,PPRIMCONVXE,PBIS,PBISCONVE
      COMMON     /BLOCK4/ P0,FTP0,SL1,SL2,W
      COMMON     /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX

      IF (.NOT.LOCKED(2)) THEN
        IF (THETA(2).EQ.0.0) THEN
          DO 10 I=1,NCH
            P(I)=P0(I)
10        CONTINUE
        ELSE
          DO 20 I=1,NCH2
           SL1(I)=FTP0(I)
20        CONTINUE
          NDIFF=0
          CALL MOVEDIFF(SL1,NCH2,THETA(2)/DX,NDIFF)
          INV=.TRUE.
          CALL FFT(SL1,W,NCH2,INV)
          DO 30 I=1,NCH
            P(I)=REAL(SL1(I))
30        CONTINUE
        ENDIF
      ENDIF

      IF (.NOT.LOCKED(2)) THEN
        DO 40 I=1,NCH2
          SL1(I)=FTP0(I)
40      CONTINUE
        NDIFF=1
        CALL MOVEDIFF(SL1,NCH2,THETA(2)/DX,NDIFF)
        INV=.TRUE.
        CALL FFT(SL1,W,NCH2,INV)
        DO 50 I=1,NCH
          PPRIM(I)=REAL(SL1(I))
50      CONTINUE
        IF (.NOT.LINEARIZED) THEN
          DO 60 I=1,NCH2
            SL1(I)=FTP0(I)
60        CONTINUE
          NDIFF=2
          CALL MOVEDIFF(SL1,NCH2,THETA(2)/DX,NDIFF)
          INV=.TRUE.
          CALL FFT(SL1,W,NCH2,INV)
          DO 70 I=1,NCH
            PBIS(I)=REAL(SL1(I))
70        CONTINUE
        ENDIF
      ENDIF

      DO 80 I=1,NCH2
        SL1(I)=FTP0(I)
80    CONTINUE
      NDIFF=0
      CALL MOVEDIFF(SL1,NCH2,THETA(2)/DX,NDIFF)

      DO 150 J=1,NOOFDECAYS
        XP=0
        CALL CALCFEXP(THETA(NOOFDECAYS+J+2),XP)
        DO 90 I=1,NCH2
          SL2(I)=SL1(I)*SL2(I)
90      CONTINUE
        INV=.TRUE.
        CALL FFT(SL2,W,NCH2,INV)
        DO 100 I=1,NCH
          PCONVE(I,J)=REAL(SL2(I))
100     CONTINUE
        IF (.NOT.LOCKED(NOOFDECAYS+J+2)) THEN
          XP=1
          CALL CALCFEXP(THETA(NOOFDECAYS+J+2),XP)
          DO 110 I=1,NCH2
            SL2(I)=SL1(I)*SL2(I)
110       CONTINUE
          INV=.TRUE.
          CALL FFT(SL2,W,NCH2,INV)
          DO 120 I=1,NCH
            PCONVXE(I,J)=REAL(SL2(I))
120       CONTINUE
          IF (.NOT.LINEARIZED) THEN
            XP=2
            CALL CALCFEXP(THETA(NOOFDECAYS+J+2),XP)
            DO 130 I=1,NCH2
              SL2(I)=SL1(I)*SL2(I)
130         CONTINUE
            INV=.TRUE.
            CALL FFT(SL2,W,NCH2,INV)
            DO 140 I=1,NCH
              PCONVX2E(I,J)=REAL(SL2(I))
140         CONTINUE
          ENDIF
        ENDIF
150   CONTINUE

      IF (.NOT.LOCKED(2)) THEN
        DO 160 I=1,NCH2
          SL1(I)=FTP0(I)
160     CONTINUE
        NDIFF=1
        CALL MOVEDIFF(SL1,NCH2,THETA(2)/DX,NDIFF)

        DO 210 J=1,NOOFDECAYS
          XP=0
          CALL CALCFEXP(THETA(NOOFDECAYS+J+2),XP)
          DO 170 I=1,NCH2
            SL2(I)=SL1(I)*SL2(I)
170       CONTINUE
          INV=.TRUE.
          CALL FFT(SL2,W,NCH2,INV)
          DO 180 I=1,NCH
            PPRIMCONVE(I,J)=REAL(SL2(I))
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       P0(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/ P0,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)
```

```
10      CONTINUE
        Z=SL3*CMPLX(-1.0,0.0)
        S2(NCHHALF+1)=SL5/(1.0-Z)
        DO 20 I=NCH2,NCHHALF,-1
          S2(I)=CONJG(S2(NCHP2-I))
20      CONTINUE
      ELSEIF (XP.EQ.1) THEN
        DO 30 I=1,NCHHALF
          Z=SL3*W(I)
          SL9=1.0-Z
          S2(I)=DX*(Z-SL4*(SL10-SL11*Z))/(SL9**2)
30      CONTINUE
        Z=SL3*CMPLX(-1.0,0.0)
        SL9=1.0-Z
        S2(NCHHALF+1)=DX*(Z-SL4*(SL10-SL11*Z))/(SL9**2)
        DO 40 I=NCH2,NCHHALF,-1
          S2(I)=CONJG(S2(NCHP2-I))
40      CONTINUE
      ELSEIF (XP.EQ.2) THEN
        DO 50 I=1,NCHHALF
          Z=SL3*W(I)
          SL9=1.0-Z
          SL12=Z+Z*Z+SL4*(2.0*Z+(SL8*Z-SL7)*SL9)
          S2(I)=SL6*SL12/(SL9**3)
50      CONTINUE
        Z=SL3*CMPLX(-1.0,0.0)
        SL9=1.0-Z
        SL12=Z+Z*Z+SL4*(2.0*Z+(SL8*Z-SL7)*SL9)
        S2(NCHHALF+1)=SL6*SL12/(SL9**3)
        DO 60 I=NCH2,NCHHALF,-1
          S2(I)=CONJG(S2(NCHP2-I))
60      CONTINUE
      ENDIF

      RETURN
      END
```

TESTMD

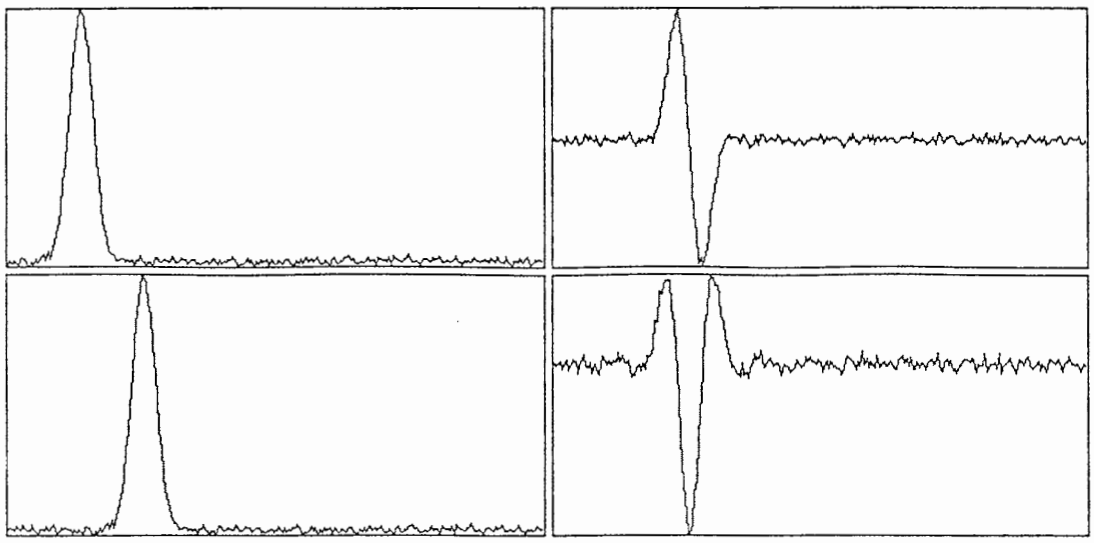
Programmet testar rutinen MOVEDIFF, använd i programmet PCONEXP för att translatera och derivera laserpulsen. MOVEDIFF använder reglerna för derivation och translation i transformplanet för att beräkna dessa [8,9]:

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

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>
```

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: TESTMD.FOR  Options: /LBZ          08/17/88 20:24:35

      PROGRAM TESTMD
C     Anders Persson, 1988

      INTEGER    N,NHALF,NCH,NCHHALF,NCHP2,NDIFF
      PARAMETER  ( N=1024, NHALF=N/2 )
      REAL       PL1(1:N),PL2(1:N),PL3(1:N),PL4(1:N)
      REAL       DX,RP1(1:2),RP2(1:2),RP3(1:2),RP4(1:N)
      REAL       THETA,TIME,FWHM
      COMPLEX    S1(1:N),S2(1:N),S3(1:N),S4(1:N)
      COMPLEX    W(1:NHALF),SLASK,Z,SL9
      INTEGER*2  ANS,KBDCHK,KBDINC,PAR(1:9),HO,MI,SE,HN,I
      LOGICAL    INV

1     PAR(1)=0
      PAR(2)=175
      PAR(3)=350
      PAR(4)=170
      PAR(5)=1
      PAR(6)=0
      PAR(7)=1
      PAR(8)=0
      PAR(9)=1

2     WRITE(*,*) 'MOVE PULSE (CH):'
      READ(*,*,ERR=2) THETA

3     WRITE(*,*) 'NCH (16..1024, 2**K):'
      READ(*,*,ERR=3) NCH
      IF (NCH.LT.16.OR.NCH.GT.1024) THEN
        GOTO 3
      ENDIF
      IF (NCH.NE.16.AND.NCH.NE.32.AND.NCH.NE.64.AND.NCH.NE.128.AND.
     f    NCH.NE.256.AND.NCH.NE.512.AND.NCH.NE.1024) THEN
        GOTO 3
      ENDIF

      NCHHALF=NCH/2
      FWHM=NCH/20+1
      FWHM=LOG(16.0)/FWHM**2

      DO 10 I=1,NCH
        S1(I)=CMPLX(EXP(-MIN(FWHM*(I-NCH/7.0)**2,85.0)),0.0)
        S1(I)=1000.0*S1(I)
        S1(I)=S1(I)+NORMAL(0.0,10.0)
        S2(I)=S1(I)
10    CONTINUE

      CALL CALCW(W,NCH)
      INV=.FALSE.
      CALL FFT(S2,W,NCH,INV)

      DO 20 I=1,NCH
        S3(I)=S2(I)
        S4(I)=S2(I)
20    CONTINUE

      NDIFF=0
      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01

      CALL MOVEDIFF(S2,NCH,THETA,NDIFF)

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME

      WRITE(*,999) 'TIME FOR MOVE:',TIME
999   FORMAT(' ',A,F8.2)
      PAUSE
      INV=.TRUE.
      CALL FFT(S2,W,NCH,INV)

      NDIFF=1
      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01

      CALL MOVEDIFF(S3,NCH,THETA,NDIFF)

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
      WRITE(*,999) 'TIME FOR MOVE+1-DIFF:',TIME
      PAUSE
      INV=.TRUE.
      CALL FFT(S3,W,NCH,INV)

      NDIFF=2
      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01

      CALL MOVEDIFF(S4,NCH,THETA,NDIFF)

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
      WRITE(*,999) 'TIME FOR MOVE+2-DIFF:',TIME
      PAUSE
      INV=.TRUE.
      CALL FFT(S4,W,NCH,INV)

70    CALL GMODE
      CALL TMODE

      WRITE(*,1000)
1000  FORMAT(////
     fT10,'F1: PLOT '/
     fT10,'F2: NEW CALC.'/
     fT10,'F3: STOP')

80    ANS=KBDINC()
      ANS=ANS-1058

      IF (ANS.EQ.1) THEN
        RP1(1)=REAL(S1(1))
        RP1(1)=REAL(S1(1))
        RP2(1)=REAL(S2(1))
        RP2(1)=REAL(S2(1))
        RP3(1)=REAL(S3(1))
        RP3(1)=REAL(S3(1))
        RP4(1)=REAL(S4(1))
```

```
      RP4(1)=REAL(S4(1))

      DO 90 I=1,NCH
        PL1(I)=REAL(S1(I))
        PL2(I)=REAL(S2(I))
        PL3(I)=REAL(S3(I))
        PL4(I)=REAL(S4(I))

        RP1(1)=MAX(RP1(1),PL1(I))
        RP1(2)=MIN(RP1(2),PL1(I))
        RP2(1)=MAX(RP2(1),PL2(I))
        RP2(2)=MIN(RP2(2),PL2(I))
        RP3(1)=MAX(RP3(1),PL3(I))
        RP3(2)=MIN(RP3(2),PL3(I))
        RP4(1)=MAX(RP4(1),PL4(I))
        RP4(2)=MIN(RP4(2),PL4(I))
90    CONTINUE
      PAR(8)=0
      PAR(1)=0
      PAR(2)=175
      CALL GMODE
      CALL PVEC(PL1,NCH,PAR,RP1)
      PAR(8)=1
      PAR(2)=0
      CALL PVEC(PL2,NCH,PAR,RP2)
      PAR(1)=355
      PAR(2)=175
      CALL PVEC(PL3,NCH,PAR,RP3)
      PAR(2)=0
      CALL PVEC(PL4,NCH,PAR,RP4)
100   ANS=KBDCHK()
      IF (ANS.EQ.0) THEN
        GOTO 100
      ENDIF
      ANS=KBDINC()
      GOTO 70
    ELSEIF (ANS.EQ.2) THEN
      GOTO 1
    ELSEIF (ANS.EQ.3) THEN
      STOP
    ENDIF

    GOTO 80
    END

    SUBROUTINE MOVEDIFF(A,N,DC,NDIFF)
C   Anders Persson, 1988

    IMPLICIT   LOGICAL (A-Z)
    INTEGER    N,NDIFF
    INTEGER*2  K,NHALF,MAXN
    COMPLEX    A(0:*),SL1,SL3
    REAL       DC,PHSTEP,TWOPI,SL2
    PARAMETER  ( TWOPI=6.283185308 )
    LOGICAL    INV

    NHALF=N/2
    IF (DC.NE.0.0) THEN
      SL1=CEXP(CMPLX(0.0,TWOPI*(0.5*DC-DC/N)))
      PHSTEP=-TWOPI*DC/N

      DO 10 K=0,NHALF-2
        SL2=PHSTEP*K
        SL3=CMPLX(COS(SL2),SIN(SL2))
        A(K)=A(K)*SL3
        A(K+NHALF+1)=A(K+NHALF+1)*SL1*SL3
10    CONTINUE

      DO 20 K=NHALF-1,NHALF
        A(K)=A(K)*CEXP(CMPLX(0.0,PHSTEP*K))
20    CONTINUE
    ENDIF

    IF (NDIFF.GT.0) THEN
      PHSTEP=TWOPI/N
      MAXN=MIN(10,N/4+2)

      A(0)=CMPLX(0.0,0.0)
      DO 30 K=1,MAXN
        SL2=PHSTEP*K
        A(K)=A(K)*CMPLX(0.0,SL2)**NDIFF
        A(N-K)=A(N-K)*CMPLX(0.0,-SL2)**NDIFF
30    CONTINUE

      SL1=CMPLX(0.0,PHSTEP*(MAXN+1))**NDIFF
      DO 40 K=MAXN+1,NHALF
        A(K)=A(K)*SL1
40    CONTINUE

      SL1=CMPLX(0.0,-PHSTEP*(MAXN+1))**NDIFF
      DO 50 K=NHALF+1,N-MAXN-1
        A(K)=A(K)*SL1
50    CONTINUE

    ENDIF

    RETURN
    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].

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

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

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

$$DFT(k \cdot \exp(-\theta \cdot Dx \cdot k)) = \cdots = z \cdot \frac{1-n \cdot z^{n-1} + (n-1) \cdot z^{n+1}}{(1-z)^2} =$$

$$= z \cdot \frac{d}{dz}\left(\frac{1-z^n}{1-z}\right) \qquad (113)$$

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

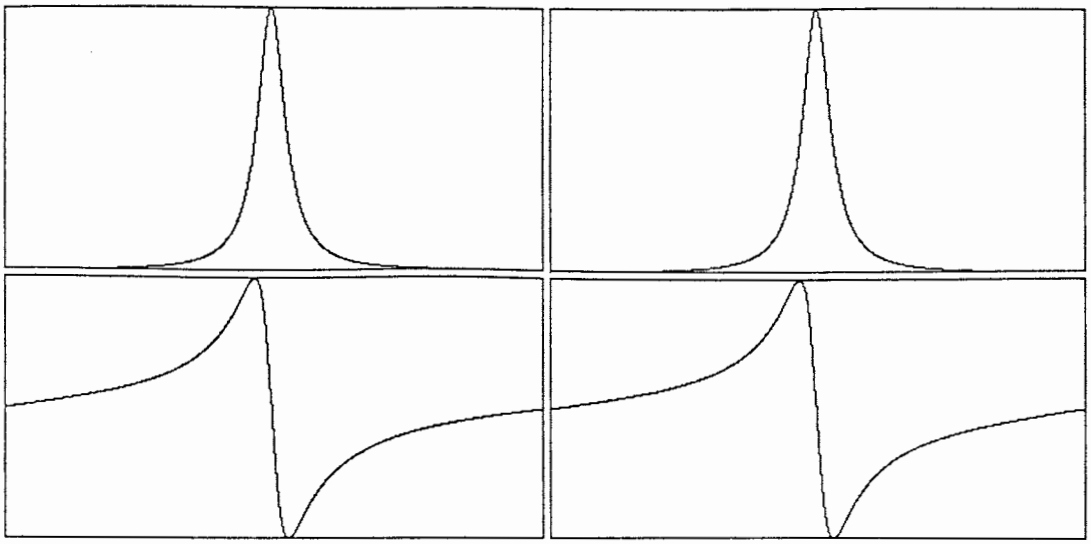
$$DFT(k^p \cdot \exp(-\theta \cdot Dx \cdot k)) = \left[z \cdot \frac{d}{dz}\right]^p \left(\frac{1-z^n}{1-z}\right) \qquad (115)$$

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

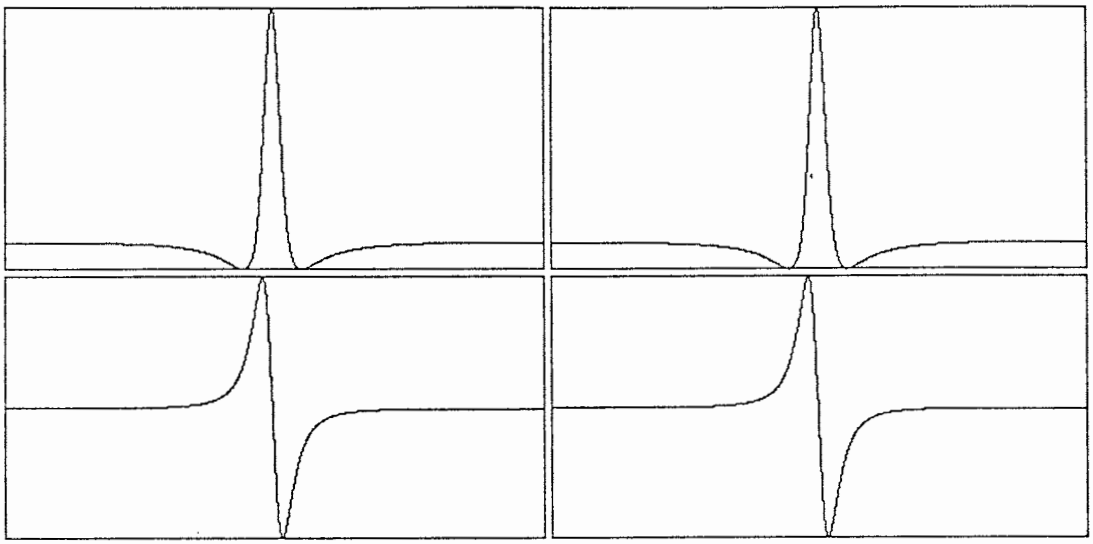
Utskrift från testkörning av programmet TFEXP:

```
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
```

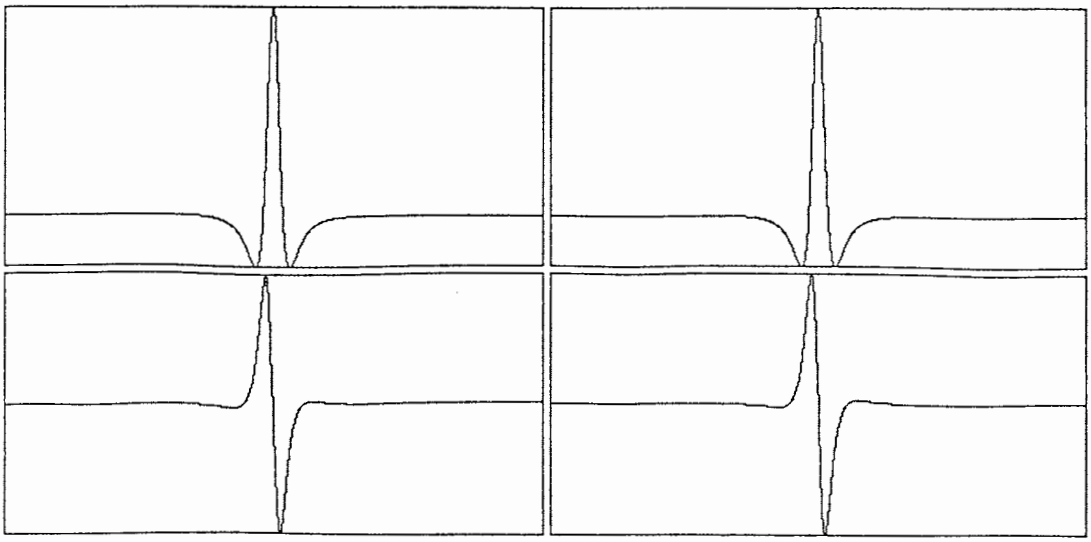
<F1>, <Printscreen>



<F2>, <Printscreen>



<F3>, <Printscreen>



<F4>

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: TFEXP.FOR   Options: /LBZ         08/17/88 20:25:24

      PROGRAM TFEXP
C     Anders Persson, 1988

      INTEGER    N,NHALF,NCH,NCHHALF,NCHP2,NCH2,NOOFDECAYS
      PARAMETER  ( N=1024, NHALF=N/2 )
      REAL       PL1(1:N),PL2(1:N),PL3(1:N),PL4(1:N)
      REAL       DX,RPRE(1:2),RPIM(1:2),THETA,TIME
      COMPLEX    S1(1:N),S2(1:N),S3(1:N)
      COMPLEX    S4(1:N),S5(1:N),S6(1:N)
      COMPLEX    W(1:NHALF)
      INTEGER*2  ANS,KBDCHK,KBDINC,PAR(1:9),HO,MI,SE,HN,I
      LOGICAL    INV
      COMMON     /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX,W

1     PAR(1)=0
      PAR(2)=175
      PAR(3)=350
      PAR(4)=170
      PAR(5)=1
      PAR(6)=0
      PAR(7)=1
      PAR(8)=0
      PAR(9)=1

2     WRITE(*,*) 'THETA (GT 0, LT 1000):'
      READ(*,*,ERR=2) THETA
      IF (THETA.LE.0.0.OR.THETA.GT.1000.0) THEN
        GOTO 2
      ENDIF

3     WRITE(*,*) 'NCH (16..1024, 2**K):'
      READ(*,*,ERR=3) NCH
      IF (NCH.LT.16.OR.NCH.GT.1024) THEN
        GOTO 3
      ENDIF
      IF (NCH.NE.16.AND.NCH.NE.32.AND.NCH.NE.64.AND.NCH.NE.128.AND.
     f    NCH.NE.256.AND.NCH.NE.512.AND.NCH.NE.1024) THEN
        GOTO 3
      ENDIF

      NCHHALF=NCH/2
      NCH2=NCH

      DX=1.0/(NCH-1)
      DO 10 I=1,NCH
        S1(I)=CMPLX(EXP(-MIN(THETA*(I-1)*DX,85.0)),0.0)
10    CONTINUE

      DO 20 I=1,NCH
        S2(I)=S1(I)*(I-1)*DX
20    CONTINUE

      DO 30 I=1,NCH
        S3(I)=S1(I)*((I-1)*DX)**2
30    CONTINUE

      CALL CALCW(W,NCH)
      INV=.FALSE.
      CALL FFT(S1,W,NCH,INV)
      CALL FFT(S2,W,NCH,INV)

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01
      CALL FFT(S3,W,NCH,INV)
      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
      WRITE(*,999) 'TIME FOR FFT:',TIME
999   FORMAT(' ',A,F8.2)
      PAUSE

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01

      CALL CALCFEXP(S4,THETA,0)

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
      WRITE(*,999) 'TIME FOR CALCULATING FFT EXP:',TIME
      PAUSE

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01

      CALL CALCFEXP(S5,THETA,1)

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
      WRITE(*,999) 'TIME FOR CALCULATING FFT X*EXP:',TIME
      PAUSE

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01

      CALL CALCFEXP(S6,THETA,2)

      CALL GETTIM(HO,MI,SE,HN)
      TIME=HO*3600.0+MI*60.0+SE+HN*0.01-TIME
      WRITE(*,999) 'TIME FOR CALCULATING FFT X2*EXP:',TIME
      PAUSE

70    CALL GMODE
      CALL TMODE

      WRITE(*,1000)
1000  FORMAT(////
     fT10,'F1: PLOT EXP'/
     fT10,'F2: PLOT X*EXP'/
     fT10,'F3: PLOT X**2*EXP'/
     fT10,'F4: NEW CALC.'/
     fT10,'F5: STOP')

80    ANS=KBDINC()
      ANS=ANS-1058

      IF (ANS.EQ.1) THEN
        RPRE(1)=REAL(S1(1))
        RPIM(1)=AIMAG(S1(1))
```

```
      RPRE(2)=REAL(S1(1))
      RPIM(2)=AIMAG(S1(1))

      DO 90 I=1,NCH
        IF (I.GT.NCHHALF) THEN
          PL1(I)=REAL(S1(I-NCHHALF))
          PL2(I)=AIMAG(S1(I-NCHHALF))
          PL3(I)=REAL(S4(I-NCHHALF))
          PL4(I)=AIMAG(S4(I-NCHHALF))
        ELSE
          PL1(I)=REAL(S1(NCHHALF+I))
          PL2(I)=AIMAG(S1(NCHHALF+I))
          PL3(I)=REAL(S4(NCHHALF+I))
          PL4(I)=AIMAG(S4(NCHHALF+I))
        ENDIF
        RPRE(1)=MAX(RPRE(1),PL1(I))
        RPRE(2)=MIN(RPRE(2),PL1(I))
        RPIM(1)=MAX(RPIM(1),PL2(I))
        RPIM(2)=MIN(RPIM(2),PL2(I))
90    CONTINUE
      PAR(8)=0
      PAR(1)=0
      PAR(2)=175
      CALL GMODE
      CALL PVEC(PL1,NCH,PAR,RPRE)
      PAR(8)=1
      PAR(2)=0
      CALL PVEC(PL2,NCH,PAR,RPIM)
      PAR(1)=355
      PAR(2)=175
      CALL PVEC(PL3,NCH,PAR,RPRE)
      PAR(2)=0
      CALL PVEC(PL4,NCH,PAR,RPIM)
100   ANS=KBDCHK()
      IF (ANS.EQ.0) THEN
        GOTO 100
      ENDIF
      ANS=KBDINC()
      GOTO 70
    ELSEIF (ANS.EQ.2) THEN
      RPRE(1)=REAL(S2(1))
      RPIM(1)=AIMAG(S2(1))
      RPRE(2)=REAL(S2(1))
      RPIM(2)=AIMAG(S2(1))

      DO 110 I=1,NCH
        IF (I.GT.NCHHALF) THEN
          PL1(I)=REAL(S2(I-NCHHALF))
          PL2(I)=AIMAG(S2(I-NCHHALF))
          PL3(I)=REAL(S5(I-NCHHALF))
          PL4(I)=AIMAG(S5(I-NCHHALF))
        ELSE
          PL1(I)=REAL(S2(NCHHALF+I))
          PL2(I)=AIMAG(S2(NCHHALF+I))
          PL3(I)=REAL(S5(NCHHALF+I))
          PL4(I)=AIMAG(S5(NCHHALF+I))
        ENDIF

        RPRE(1)=MAX(RPRE(1),PL1(I))
        RPRE(2)=MIN(RPRE(2),PL1(I))
        RPIM(1)=MAX(RPIM(1),PL2(I))
        RPIM(2)=MIN(RPIM(2),PL2(I))
110   CONTINUE
      PAR(8)=0
      PAR(1)=0
      PAR(2)=175
      CALL GMODE
      CALL PVEC(PL1,NCH,PAR,RPRE)
      PAR(8)=1
      PAR(2)=0
      CALL PVEC(PL2,NCH,PAR,RPIM)
      PAR(1)=355
      PAR(2)=175
      CALL PVEC(PL3,NCH,PAR,RPRE)
      PAR(2)=0
      CALL PVEC(PL4,NCH,PAR,RPIM)
120   ANS=KBDCHK()
      IF (ANS.EQ.0) THEN
        GOTO 120
      ENDIF
      ANS=KBDINC()
      GOTO 70
    ELSEIF (ANS.EQ.3) THEN
      RPRE(1)=REAL(S3(1))
      RPIM(1)=AIMAG(S3(1))
      RPRE(2)=REAL(S3(1))
      RPIM(2)=AIMAG(S3(1))

      DO 130 I=1,NCH
        IF (I.GT.NCHHALF) THEN
          PL1(I)=REAL(S3(I-NCHHALF))
          PL2(I)=AIMAG(S3(I-NCHHALF))
          PL3(I)=REAL(S6(I-NCHHALF))
          PL4(I)=AIMAG(S6(I-NCHHALF))
        ELSE
          PL1(I)=REAL(S3(NCHHALF+I))
          PL2(I)=AIMAG(S3(NCHHALF+I))
          PL3(I)=REAL(S6(NCHHALF+I))
          PL4(I)=AIMAG(S6(NCHHALF+I))
        ENDIF

        RPRE(1)=MAX(RPRE(1),PL1(I))
        RPRE(2)=MIN(RPRE(2),PL1(I))
        RPIM(1)=MAX(RPIM(1),PL2(I))
        RPIM(2)=MIN(RPIM(2),PL2(I))
130   CONTINUE
      PAR(8)=0
      PAR(1)=0
      PAR(2)=175
      CALL GMODE
      CALL PVEC(PL1,NCH,PAR,RPRE)
      PAR(8)=1
      PAR(2)=0
      CALL PVEC(PL2,NCH,PAR,RPIM)
      PAR(1)=355
      PAR(2)=175
      CALL PVEC(PL3,NCH,PAR,RPRE)
      PAR(2)=0
      CALL PVEC(PL4,NCH,PAR,RPIM)
140   ANS=KBDCHK()
```

```
        IF (ANS.EQ.0) THEN
          GOTO 140
        ENDIF
        ANS=KBDINC()
        GOTO 70
      ELSEIF (ANS.EQ.4) THEN
        GOTO 1
      ELSEIF (ANS.EQ.5) THEN
        STOP
      ENDIF

      GOTO 80
      END

      SUBROUTINE CALCFEXP(A,THETA,XP)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    N,NHALF,NCH,NCHHALF,NCHP2,NCH2,NOOFDECAYS,XP
      PARAMETER  ( N=1024, NHALF=N/2 )
      REAL       DX,THETA,SL1,SL2,SL3,SL4,SL5,SL6,SL7,SL8,SL10,SL11
      COMPLEX    A(1:*),W(1:NHALF),Z,SL9,SL12
      INTEGER*2  I
      COMMON     /BLOCK5/ NCH,NCH2,NOOFDECAYS,DX,W

      SL1=-DX*THETA
      SL2=-2.0*3.141592654/NCH2
      SL3=EXP(SL1)
      SL4=SL3**NCH2
      SL5=1.0-SL4
      SL6=DX**2
      SL7=NCH2**2
      SL8=(NCH2-1)**2
      SL10=NCH2
      SL11=NCH2-1
      NCHP2=NCH2+2
      NCHHALF=NCH2/2

      IF (XP.EQ.0) THEN
        DO 10 I=1,NCHHALF
          Z=SL3*W(I)
          A(I)=SL5/(1.0-Z)
10      CONTINUE

        Z=SL3*CMPLX(-1.0,0.0)
        A(NCHHALF+1)=SL5/(1.0-Z)
        DO 20 I=NCH2,NCHHALF,-1
          A(I)=CONJG(A(NCHP2-I))
20      CONTINUE

      ELSEIF (XP.EQ.1) THEN
        DO 30 I=1,NCHHALF
          Z=SL3*W(I)
          SL9=1.0-Z
          A(I)=DX*(Z-SL4*(SL10-SL11*Z))/(SL9**2)
30      CONTINUE

        Z=SL3*CMPLX(-1.0,0.0)
        SL9=1.0-Z
        A(NCHHALF+1)=DX*(Z-SL4*(SL10-SL11*Z))/(SL9**2)

        DO 40 I=NCH2,NCHHALF,-1
          A(I)=CONJG(A(NCHP2-I))
40      CONTINUE

      ELSEIF (XP.EQ.2) THEN
        DO 50 I=1,NCHHALF
          Z=SL3*W(I)
          SL9=1.0-Z
          SL12=Z+Z*Z+SL4*(2.0*Z+(SL8*Z-SL7)*SL9)
          A(I)=SL6*SL12/(SL9**3)
50      CONTINUE

        Z=SL3*CMPLX(-1.0,0.0)
        SL9=1.0-Z
        SL12=Z+Z*Z+SL4*(2.0*Z+(SL8*Z-SL7)*SL9)
        A(NCHHALF+1)=SL6*SL12/(SL9**3)

        DO 60 I=NCH2,NCHHALF,-1
          A(I)=CONJG(A(NCHP2-I))
60      CONTINUE

      ENDIF

      RETURN
      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

      SUBROUTINE PVEC(V,N,PAR,RP)
C     ANDERS PERSSON, 1987

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     N,I
      INTEGER*2   PAR(1:9),X,Y
      REAL        V(0:N-1),RP(1:2),DX

      CALL GPAGE(PAR(6))
      IF(PAR(8).EQ.0) CALL CLRSCR

      X=1-PAR(5)
      CALL LEVEL(X)
      IF(PAR(9).EQ.0) CALL BLKFIL(PAR(1),PAR(2),PAR(3),PAR(4))
      CALL LEVEL(PAR(5))

      DX=FLOAT(PAR(3))/FLOAT(N)
      X=PAR(1)
      Y=((V(0)-RP(2))/(RP(1)-RP(2)))*PAR(4)+PAR(2)
      CALL MOVE(X,Y)

      DO 10 I=0,N-1
        X=PAR(1)+I*DX
        Y=((V(I)-RP(2))/(RP(1)-RP(2)))*PAR(4)+PAR(2)
        IF(PAR(7).EQ.0) THEN
          CALL PLOT(X,Y)
        ELSE
          CALL DLINE(X,Y)
        ENDIF
10    CONTINUE

      X=PAR(1)
      Y=PAR(2)
      CALL MOVE(X,Y)
      X=X+PAR(3)
      CALL DLINE(X,Y)
      Y=Y+PAR(4)
      CALL DLINE(X,Y)
      X=X-PAR(3)
      CALL DLINE(X,Y)
      Y=Y-PAR(4)
      CALL DLINE(X,Y)

      RETURN
      END

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

      SUBROUTINE CALCW(W,N)
C     ANDERS PERSSON, 1985

      IMPLICIT   LOGICAL (A-Z)
      COMPLEX    W(0:*)
      REAL       FI,TWOPI
      PARAMETER  ( TWOPI=6.283185308 )
      INTEGER    N,I

      FI=-TWOPI/FLOAT(N)
      DO 10 I=0,N/2-1
        W(I)=CEXP(CMPLX(0.0,FI*I))
10    CONTINUE
      RETURN
      END

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

      SUBROUTINE FFT(G,W,N,INV)
C     ANDERS PERSSON, 1985

      IMPLICIT   LOGICAL (A-Z)
      COMPLEX    G(0:*),W(0:*),SLASK
      REAL       C
      INTEGER*2  I,J,K,L,M,P,R,S
      INTEGER    N
      LOGICAL    INV

C     *** Butterfly ***
      M=LOG(FLOAT(N))/LOG(2.0)+0.5
      DO 10 K=0,M-1
      DO 10 L=0,N-N/2**K,N/2**K
      DO 10 I=0,N/2**(K+1)-1
        R=L+I
        S=R+N/2**(K+1)
        P=I*2**K
        SLASK=G(R)+G(S)
        IF(INV) THEN
          G(S)=(G(R)-G(S))*CONJG(W(P))
        ELSE
          G(S)=(G(R)-G(S))*W(P)
        ENDIF
        G(R)=SLASK
10    CONTINUE

C     *** Bit reversal ***
      DO 20 I=0,N-1
        K=I
        L=0
        DO 30 J=1,M
          L=L+MOD(K,2)*2**(M-J)
          K=K/2
30      CONTINUE
        IF(L.GT.I) THEN
          SLASK=G(I)
          G(I)=G(L)
          G(L)=SLASK
        ENDIF
20    CONTINUE

C     *** Norm. ***
      C=1.0/FLOAT(N)
      IF(INV) THEN
        DO 40 I=0,N-1
          G(I)=G(I)*C
40      CONTINUE
      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 singulär 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(sekunder)≈ $6.635 \cdot 10^{-4} \cdot N^{2.894}$ (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·X=B med Gausselimination. A och B förstörs.

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

SUBMAT A-B→C

MULTMAT A*B→C

TRANSMAT A^T→B

ADDCMAT Addera skalären C till varje element i A.

MULTCMAT Multiplicera varje element i A med skalären C.

COPYMAT A→B

UNITMAT Placera en enhetsmatris i A.

ZEROMAT Nollställ matrisen A.

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: MATLIB.FOR  Options: /LBZ          08/02/88 14:32:01

   1 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
   2 C                                     C
   3 C     Library:  MATLIB.LIB            C
   4 C                                     C
   5 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
   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=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.0.0) THEN
  30           SING=.TRUE.
  31           DET=0.0
  32           RETURN
  33         ENDIF
  34
  35         IF (ROW(I)ºGT.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
  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       END
  90
  91       SUBROUTINE GAUSS(A,X,B,N,SING,RA,KA)
  92 C     ANDERS PERSSON, 1986
  93
  94       IMPLICIT    LOGICAL (A-Z)
  95       INTEGER     RA,KA,N
  96       REAL        A(1:RA,1:KA),X(1:N),B(1:N),BIG,TEMP
  97       INTEGER     R,K,MAXROW,I,J
  98       LOGICAL     SING
  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       *** Radpivotering ***
```

```
      ********************
      BIG=ABS(A(R,R))
      MAXROW=R
      DO 50 I=R+1,N
        IF (ABS(A(I,R)).GT.BIG) THEN
          MAXROW=I
          BIG=ABS(A(I,R))
        ENDIF
50    CONTINUE
      IF (BIG.EQ.0.0) THEN
        SING=.TRUE.
        RETURN
      ENDIF
      DO 60 K=R,N
        TEMP=A(R,K)
        A(R,K)=A(MAXROW,K)
        A(MAXROW,K)=TEMP
60    CONTINUE
      TEMP=B(R)
      B(R)=B(MAXROW)
      B(MAXROW)=TEMP

C     ********************
C     *** Elimination ***
C     ********************
      TEMP=1.0/A(R,R)
      DO 70 K=N,R,-1
        A(R,K)=A(R,K)*TEMP
70    CONTINUE
      B(R)=B(R)*TEMP
      DO 80 I=R+1,N
        DO 90 J=R+1,N
          A(I,J)=A(I,J)-A(I,R)*A(R,J)
90      CONTINUE
        B(I)=B(I)-B(R)*A(I,R)
80    CONTINUE
40  CONTINUE

C   ********************
C   *** Substitution ***
C   ********************
    DO 100 R=N,1,-1
      X(R)=B(R)
      DO 110 K=R+1,N
        X(R)=X(R)-A(R,K)*X(K)
110   CONTINUE
100 CONTINUE

    RETURN
    END

    SUBROUTINE ADDMAT(A,B,C,R,K,RA,KA,RB,KB,RC,KC)
C   Anders Persson, 1988

    IMPLICIT    LOGICAL (A-Z)
    INTEGER     R,K,I,J,RA,KA,RB,KB,RC,KC
    REAL        A(1:RA,1:KA),B(1:RB,1:KB),C(1:RC,1:KC)

    DO 10 I=1,R
    DO 10 J=1,R
      C(I,J)=A(I,J)+B(I,J)
10  CONTINUE

    RETURN
    END

    SUBROUTINE SUBMAT(A,B,C,R,K,RA,KA,RB,KB,RC,KC)
C   Anders Persson, 1988

    IMPLICIT    LOGICAL (A-Z)
    INTEGER     R,K,I,J,RA,KA,RB,KB,RC,KC
    REAL        A(1:RA,1:KA),B(1:RB,1:KB),C(1:RC,1:KC)

    DO 10 I=1,R
    DO 10 J=1,R
      C(I,J)=A(I,J)-B(I,J)
10  CONTINUE

    RETURN
    END

    SUBROUTINE MULTMAT(A,B,C,AR,AK,BK,RA,KA,RB,KB,RC,KC)
C   Anders Persson, 1988

    IMPLICIT    LOGICAL (A-Z)
    INTEGER     I,J,K,AR,AK,BK,RA,KA,RB,KB,RC,KC
    REAL        A(1:RA,1:KA),B(1:RB,1:KB),C(1:RC,1:KC)

    CALL ZEROMAT(C,AR,BK,RC,KC)
    DO 10 I=1,AR
    DO 10 J=1,BK
    DO 10 K=1,AK
      C(I,J)=C(I,J)+A(I,K)*B(K,J)
10  CONTINUE

    RETURN
    END

    SUBROUTINE TRANSMAT(A,B,R,K,RA,KA,RB,KB)
C   Anders Persson, 1988

    IMPLICIT    LOGICAL (A-Z)
    INTEGER     R,K,I,J,RA,KA,RB,KB
    REAL        A(1:RA,1:KA),B(1:RB,1:KB)

    DO 10 I=1,R
    DO 10 J=1,K
      B(J,I)=A(I,J)
10  CONTINUE

    RETURN
    END

    SUBROUTINE ADDCMAT(A,C,R,K,RA,KA)
C   Anders Persson, 1988

    IMPLICIT    LOGICAL (A-Z)
    INTEGER     R,K,I,J,RA,KA
    REAL        A(1:RA,1:KA),C
```

```
      DO 10 I=1,R
      DO 10 J=1,K
        A(I,J)=A(I,J)+C
10    CONTINUE

      RETURN
      END

      SUBROUTINE MULTCMAT(A,C,R,K,RA,KA)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     R,K,I,J,RA,KA
      REAL        A(1:RA,1:KA),C

      DO 10 I=1,R
      DO 10 J=1,K
        A(I,J)=A(I,J)*C
10    CONTINUE

      RETURN
      END

      SUBROUTINE COPYMAT(A,B,R,K,RA,KA,RB,KB)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     R,K,I,J,RA,KA,RB,KB
      REAL        A(1:RA,1:KA),B(1:RB,1:KB)

      DO 10 I=1,R
      DO 10 J=1,K
        B(I,J)=A(I,J)
10    CONTINUE

      RETURN
      END

      SUBROUTINE UNITMAT(A,N,RA,KA)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     N,R,K,RA,KA
      REAL        A(1:RA,1:KA)

      CALL ZEROMAT(A,N,N,RA,KA)
      DO 10 R=1,N
        A(R,R)=1.0
10    CONTINUE

      RETURN
      END

      SUBROUTINE ZEROMAT(A,R,K,RA,KA)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     R,K,I,J,RA,KA
      REAL        A(1:RA,1:KA)

      DO 10 I=1,R
      DO 10 J=1,K
        A(I,J)=0.0
10    CONTINUE

      RETURN
      END
```

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: DMATLIB.FOR Options: /LBZ          08/02/88 14:17:18

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                  C
C     Library:  DMATLIB.LIB        C
C                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      SUBROUTINE INVMAT(A,N,DET,SING,RA,KA)
C     Anders Persson, 1988

      IMPLICIT     LOGICAL (A-Z)
      INTEGER      N,R,K,I,L,ROW(1:100),COL(1:100),RA,KA
      REAL*8       A(1:RA,1:KA),DET,PIVOT,INVPIV,SLASK
      LOGICAL      SING

      DET  = 1.0D0
      SING = .FALSE.

      DO 70 I=1,N
        PIVOT=0.0D0
        DO 10 R=I,N
        DO 10 K=I,N
          IF (ABS(A(R,K)).GT.ABS(PIVOT)) THEN
            PIVOT=A(R,K)
            ROW(I)=R
            COL(I)=K
          ENDIF
10      CONTINUE

        IF (PIVOT.EQ.0.0D0) THEN
          SING=.TRUE.
          DET=0.0D0
          RETURN
        ENDIF

        IF (ROW(I).GT.I) THEN
          DO 20 K=1,N
            SLASK=A(I,K)
            A(I,K)=A(ROW(I),K)
            A(ROW(I),K)=-SLASK
20        CONTINUE
        ENDIF

        IF (COL(I).GT.I) THEN
          DO 30 R=1,N
            SLASK=A(R,I)
            A(R,I)=A(R,COL(I))
            A(R,COL(I))=-SLASK
30        CONTINUE
        ENDIF

        INVPIV=1.0D0/PIVOT
        DO 40 R=1,N
          IF(R.NE.I) A(R,I)=-A(R,I)*INVPIV
40      CONTINUE

        DO 50 R=1,N
        DO 50 K=1,N
          IF(R.NE.I.AND.K.NE.I) A(R,K)=A(R,K)+A(R,I)*A(I,K)
50      CONTINUE

        DO 60 K=1,N
          IF(K.NE.I) A(I,K)=A(I,K)*INVPIV
60      CONTINUE

        A(I,I)=INVPIV
        DET=DET*PIVOT

70    CONTINUE

      DO 100 L=N,1,-1
        IF (ROW(L).GT.L) THEN
          DO 80 R=1,N
            SLASK=A(R,L)
            A(R,L)=-A(R,ROW(L))
            A(R,ROW(L))=SLASK
80        CONTINUE
        ENDIF

        IF (COL(L).GT.L) THEN
          DO 90 K=1,N
            SLASK=A(L,K)
            A(L,K)=-A(COL(L),K)
            A(COL(L),K)=SLASK
90        CONTINUE
        ENDIF

100   CONTINUE

      RETURN
      END

      SUBROUTINE GAUSS(A,X,B,N,SING,RA,KA)
C     ANDERS PERSSON, 1986

      IMPLICIT     LOGICAL (A-Z)
      INTEGER      RA,KA,N
      REAL*8       A(1:RA,1:KA),X(1:N),B(1:N),BIG,TEMP
      INTEGER      R,K,MAXROW,I,J
      LOGICAL      SING

C     ****************
C     *** Skalning ***
C     ****************
      DO 10 R=1,N
        BIG=ABS(B(R))
        DO 20 K=1,N
          BIG=MAX(BIG,ABS(A(R,K)))
20      CONTINUE
        BIG=1.0D0/BIG
        DO 30 K=1,N
          A(R,K)=A(R,K)*BIG
30      CONTINUE
        B(R)=B(R)*BIG
10    CONTINUE

      DO 40 R=1,N
C       *********************
```

```
C       *** Radpivotering ***
C       *********************
        BIG=ABS(A(R,R))
        MAXROW=R
        DO 50 I=R+1,N
          IF (ABS(A(I,R)).GT.BIG) THEN
            MAXROW=I
            BIG=ABS(A(I,R))
          ENDIF
50      CONTINUE

        IF (BIG.EQ.0.0D0) THEN
          SING=.TRUE.
          RETURN
        ENDIF

        DO 60 K=R,N
          TEMP=A(R,K)
          A(R,K)=A(MAXROW,K)
          A(MAXROW,K)=TEMP
60      CONTINUE

        TEMP=B(R)
        B(R)=B(MAXROW)
        B(MAXROW)=TEMP
C       *******************
C       *** Elimination ***
C       *******************

        TEMP=1.0D0/A(R,R)
        DO 70 K=N,R,-1
          A(R,K)=A(R,K)*TEMP
70      CONTINUE

        B(R)=B(R)*TEMP
        DO 80 I=R+1,N
          DO 90 J=R+1,N
            A(I,J)=A(I,J)-A(I,R)*A(R,J)
90        CONTINUE
          B(I)=B(I)-B(R)*A(I,R)
80      CONTINUE

40    CONTINUE

C     *********************
C     *** Substitution ***
C     *********************
      DO 100 R=N,1,-1
        X(R)=B(R)
        DO 110 K=R+1,N
          X(R)=X(R)-A(R,K)*X(K)
110     CONTINUE
100   CONTINUE

      RETURN
      END

      SUBROUTINE ADDMAT(A,B,C,R,K,RA,KA,RB,KB,RC,KC)
C     Anders Persson, 1988

      IMPLICIT     LOGICAL (A-Z)
      INTEGER      R,K,I,J,RA,KA,RB,KB,RC,KC
      REAL*8       A(1:RA,1:KA),B(1:RB,1:KB),C(1:RC,1:KC)

      DO 10 I=1,R
      DO 10 J=1,R
        C(I,J)=A(I,J)+B(I,J)
10    CONTINUE

      RETURN
      END

      SUBROUTINE SUBMAT(A,B,C,R,K,RA,KA,RB,KB,RC,KC)
C     Anders Persson, 1988

      IMPLICIT     LOGICAL (A-Z)
      INTEGER      R,K,I,J,RA,KA,RB,KB,RC,KC
      REAL*8       A(1:RA,1:KA),B(1:RB,1:KB),C(1:RC,1:KC)

      DO 10 I=1,R
      DO 10 J=1,R
        C(I,J)=A(I,J)-B(I,J)
10    CONTINUE

      RETURN
      END

      SUBROUTINE MULTMAT(A,B,C,AR,AK,BK,RA,KA,RB,KB,RC,KC)
C     Anders Persson, 1988

      IMPLICIT     LOGICAL (A-Z)
      INTEGER      I,J,K,AR,AK,BK,RA,KA,RB,KB,RC,KC
      REAL*8       A(1:RA,1:KA),B(1:RB,1:KB),C(1:RC,1:KC)

      CALL ZEROMAT(C,AR,BK,RC,KC)
      DO 10 I=1,AR
      DO 10 J=1,BK
      DO 10 K=1,AK
        C(I,J)=C(I,J)+A(I,K)*B(K,J)
10    CONTINUE

      RETURN
      END

      SUBROUTINE TRANSMAT(A,B,R,K,RA,KA,RB,KB)
C     Anders Persson, 1988

      IMPLICIT     LOGICAL (A-Z)
      INTEGER      R,K,I,J,RA,KA,RB,KB
      REAL*8       A(1:RA,1:KA),B(1:RB,1:KB)

      DO 10 I=1,R
      DO 10 J=1,K
        B(J,I)=A(I,J)
10    CONTINUE

      RETURN
      END

      SUBROUTINE ADDCMAT(A,C,R,K,RA,KA)
```

```
      Anders Persson, 1988

      IMPLICIT     LOGICAL (A-Z)
      INTEGER      R,K,I,J,RA,KA
      REAL*8       A(1:RA,1:KA),C

      DO 10 I=1,R
      DO 10 J=1,K
        A(I,J)=A(I,J)+C
10    CONTINUE

      RETURN
      END

      SUBROUTINE MULTCMAT(A,C,R,K,RA,KA)
C     Anders Persson, 1988

      IMPLICIT     LOGICAL (A-Z)
      INTEGER      R,K,I,J,RA,KA
      REAL*8       A(1:RA,1:KA),C

      DO 10 I=1,R
      DO 10 J=1,K
        A(I,J)=A(I,J)*C
10    CONTINUE

      RETURN
      END

      SUBROUTINE COPYMAT(A,B,R,K,RA,KA,RB,KB)
C     Anders Persson, 1988

      IMPLICIT     LOGICAL (A-Z)
      INTEGER      R,K,I,J,RA,KA,RB,KB
      REAL*8       A(1:RA,1:KA),B(1:RB,1:KB)

      DO 10 I=1,R
      DO 10 J=1,K
        B(I,J)=A(I,J)
10    CONTINUE

      RETURN
      END

      SUBROUTINE UNITMAT(A,N,RA,KA)
C     Anders Persson, 1988

      IMPLICIT     LOGICAL (A-Z)
      INTEGER      N,R,K,RA,KA
      REAL*8       A(1:RA,1:KA)

      CALL ZEROMAT(A,N,N,RA,KA)
      DO 10 R=1,N
        A(R,R)=1.0D0
10    CONTINUE

      RETURN
      END

      SUBROUTINE ZEROMAT(A,R,K,RA,KA)
C     Anders Persson, 1988

      IMPLICIT     LOGICAL (A-Z)
      INTEGER      R,K,I,J,RA,KA
      REAL*8       A(1:RA,1:KA)

      DO 10 I=1,R
      DO 10 J=1,K
        A(I,J)=0.0D0
10    CONTINUE

      RETURN
      END
```

Ritrutiner:

SCRPLOT, (D)SCRPLOT, 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 $f(\underline{\theta}; x)$ på samma bild. Därefter beräknas och ritas residualvektorn (48). Rutinerna anropas på samma sätt med samma parametrar, SCRPLOT 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.

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: SCRPLOT.FOR Options: /LBZ          08/02/88 14:50:47

      SUBROUTINE SCRPLOT(Y,SIGMA,X,R,N,F,THETA,L,ERRORBAR)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    PLXMIN,PLXMAX,PLYMIN,PLYMAX,FVECRANGE
      PARAMETER  ( FVECRANGE=500 )
      INTEGER    I,N,L,X0,Y0,XL,YL
      REAL       Y(1:N),SIGMA(1:N),X(1:N),R(1:N),F,THETA(1:L),XMIN,XMAX
      REAL       FVEC(1:FVECRANGE),SLASK,DX,XI,XP,YP,YMIN,YMAX,A
      INTEGER*2  PX,PY,ANS,KBDCHK,KBDINC
      LOGICAL    ERRORBAR

      XP(A)=NINT(((A-XMIN)/(XMAX-XMIN))*XL+X0)
      YP(A)=NINT(((A-YMIN)/(YMAX-YMIN))*YL+Y0)
C
      PLXMIN=0
      PLXMAX=719
      PLYMIN=0
      PLYMAX=347
C
      X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
      Y0=NINT(PLYMIN+0.26*(PLYMAX-PLYMIN))
      XL=NINT(0.99*(PLXMAX-PLXMIN))
      YL=NINT(0.73*(PLYMAX-PLYMIN))

      XMIN=X(1)
      XMAX=X(N)
      YMIN=Y(1)
      YMAX=Y(1)

      DX=(XMAX-XMIN)/(FVECRANGE-1)
      DO 10 I=1,FVECRANGE
        XI=(I-1)*DX+XMIN
        FVEC(I)=F(THETA,L,XI)
        YMIN=MIN(YMIN,FVEC(I))
        YMAX=MAX(YMAX,FVEC(I))
10    CONTINUE

      DO 20 I=1,N
        IF (ERRORBAR) THEN
          YMIN=MIN(YMIN,Y(I)-SIGMA(I))
          YMAX=MAX(YMAX,Y(I)+SIGMA(I))
        ELSE
          YMIN=MIN(YMIN,Y(I))
          YMAX=MAX(YMAX,Y(I))
        ENDIF
20    CONTINUE

      IF (YMAX-YMIN.LT.1.0E-20) THEN
        YMAX=YMAX+0.5
        YMIN=YMIN-0.5
      ENDIF

      CALL GMODE
      ANS=1
      CALL GPAGE(ANS)
      CALL CLRSCR
      CALL DISP(ANS)

      DO 30 I=1,N
        PX=XP(X(I))
        IF (ERRORBAR) THEN
          PY=YP(Y(I)+SIGMA(I))
        ELSE
          PY=YP(Y(I))
        ENDIF
        CALL MOVE(PX,PY)
        IF (ERRORBAR) THEN
          PY=YP(Y(I)-SIGMA(I))
          CALL DLINE(PX,PY)
        ELSE
          CALL PLOT(PX,PY)
        ENDIF
30    CONTINUE

      PY=YP(FVEC(1))
      PX=XP(XMIN)
      CALL MOVE(PX,PY)
      DO 40 I=1,FVECRANGE
        XI=(I-1)*DX+XMIN
        PX=XP(XI)
        PY=YP(FVEC(I))
        CALL DLINE(PX,PY)
40    CONTINUE

      X0=PLXMIN
      Y0=NINT(PLYMIN+0.25*(PLYMAX-PLYMIN))
      XL=PLXMAX-PLXMIN
      YL=NINT(0.75*(PLYMAX-PLYMIN))

      PX=X0
      PY=Y0
      CALL MOVE(PX,PY)
      PX=X0+XL
      CALL DLINE(PX,PY)
      PY=Y0+YL
      CALL DLINE(PX,PY)
      PX=X0
      CALL DLINE(PX,PY)
      PY=Y0
      CALL DLINE(PX,PY)

      DO 50 I=1,N
        R(I)=0.0
        IF (SIGMA(I).NE.0.0) THEN
          R(I)=(Y(I)-F(THETA,L,X(I)))/SIGMA(I)
        ENDIF
50    CONTINUE

      YMAX=R(1)
      YMIN=R(1)
      DO 60 I=1,N
        YMAX=MAX(YMAX,R(I))
        YMIN=MIN(YMIN,R(I))
60    CONTINUE

      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
```

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

```
  1       SUBROUTINE SCRPLOT(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*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
```

```
        CALL MOVE(PX,PY)
        IF (ERRORBAR) THEN
          PY=YP(Y(I)-SIGMA(I))
          CALL DLINE(PX,PY)
        ELSE
          CALL PLOT(PX,PY)
        ENDIF
30    CONTINUE

      PY=YP(FVEC(1))
      PX=XP(XMIN)
      CALL MOVE(PX,PY)
      DO 40 I=1,FVECRANGE
        XI=(I-1)*DX+XMIN
        PX=XP(XI)
        PY=YP(FVEC(I))
        CALL DLINE(PX,PY)
40    CONTINUE

      X0=PLXMIN
      Y0=NINT(PLYMIN+0.25*(PLYMAX-PLYMIN))
      XL=PLXMAX-PLXMIN
      YL=NINT(0.75*(PLYMAX-PLYMIN))

      PX=X0
      PY=Y0
      CALL MOVE(PX,PY)
      PX=X0+XL
      CALL DLINE(PX,PY)
      PY=Y0+YL
      CALL DLINE(PX,PY)
      PX=X0
      CALL DLINE(PX,PY)
      PY=Y0
      CALL DLINE(PX,PY)

      DO 50 I=1,N
        R(I)=0.0D0
        IF (SIGMA(I).NE.0.0D0) THEN
          R(I)=(Y(I)-F(THETA,L,X(I)))/SIGMA(I)
        ENDIF
50    CONTINUE

      YMAX=R(1)
      YMIN=R(1)
      DO 60 I=1,N
        YMAX=MAX(YMAX,R(I))
        YMIN=MIN(YMIN,R(I))
60    CONTINUE

      YMAX=ABS(YMAX)
      YMAX=MAX(YMAX,ABS(YMIN))
      YMIN=-YMAX
      IF (YMAX-YMIN.LT.1.0D-20) THEN
        YMAX=YMAX+0.5D0
        YMIN=YMIN-0.5D0
      ENDIF

      X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
      Y0=NINT(PLYMIN+0.01*(PLYMAX-PLYMIN))
      XL=NINT(0.99*(PLXMAX-PLXMIN))
      YL=NINT(0.22*(PLYMAX-PLYMIN))

      DO 70 I=1,N
        IF (R(I).NE.0.0D0) THEN
          PX=XP(X(I))
          PY=YP(R(I))
          CALL PLOT(PX,PY)
        ENDIF
70    CONTINUE

      PX=XP(X(1))
      PY=YP(0.0D0)
      CALL MOVE(PX,PY)
      PX=XP(X(N))
      CALL DLINE(PX,PY)

      X0=PLXMIN
      Y0=PLYMIN
      XL=PLXMAX-PLXMIN
      YL=NINT(0.24*(PLYMAX-PLYMIN))

      PX=X0
      PY=Y0
      CALL MOVE(PX,PY)
      PX=X0+XL
      CALL DLINE(PX,PY)
      PY=Y0+YL
      CALL DLINE(PX,PY)
      PX=X0
      CALL DLINE(PX,PY)
      PY=Y0
      CALL DLINE(PX,PY)

80    ANS=KBDCHK()
      IF (ANS.EQ.0) GOTO 80
      ANS=KBDINC()
      CALL TMODE

      RETURN
      END
```

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

```
      SUBROUTINE ROLPLOT(Y,SIGMA,X,R,N,F,THETA,L,ERRORBAR)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    FVECRANGE
      REAL       PLXMIN,PLXMAX,PLYMIN,PLYMAX
      PARAMETER  ( FVECRANGE=500 )
      INTEGER    N,L,I,X0,Y0,XL,YL,P1,P2,P3
      REAL       Y(1:N),SIGMA(1:N),X(1:N),R(1:N),F,THETA(1:L),XMIN,XMAX
      REAL       FVEC(1:FVECRANGE),SLASK,DX,XI,XP,YP,YMIN,YMAX,A
      REAL       PX,PY,PS
      INTEGER*2  ANS,KBDCHK,KBDINC
      LOGICAL    ERRORBAR,LINKPEN
      CHARACTER  SV*1,VS*6

```

```
      XP(A)=NINT(((A-XMIN)/(XMAX-XMIN))*XL+X0)
      YP(A)=NINT(((A-YMIN)/(YMAX-YMIN))*YL+Y0)

      WRITE(*,*) 'Standard paper size? (Y/*):'
      READ(*,100) SV
100   FORMAT(A)

      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        WRITE(*,*) 'Size A3 or A4 (3/4):'
        READ(*,100) SV
        IF (SV.EQ.'3') THEN
          PLXMIN=0.0
          PLXMAX=3600.0
          PLYMIN=0.0
          PLYMAX=2600.0
        ELSE
          PLXMIN=0.0
          PLXMAX=2600.0
          PLYMIN=0.0
          PLYMAX=1800.0
        ENDIF
      ELSE
1       WRITE(*,*) 'Enter plot window (in cm.).'
        WRITE(*,*) 'Xmin, Ymin, Xmax, Ymax:'
        READ(*,*,ERR=1,END=1) PLXMIN,PLYMIN,PLXMAX,PLYMAX
        IF (PLXMIN.LT.0.0.OR.PLXMIN.GE.38.0) GOTO 1
        IF (PLYMIN.LT.0.0.OR.PLYMIN.GE.27.0) GOTO 1
        IF (PLXMAX.LT.0.0.OR.PLXMAX.GT.38.0) GOTO 1
        IF (PLYMAX.LT.0.0.OR.PLYMAX.GT.27.0) GOTO 1
        IF (PLXMIN.GE.PLXMAX) GOTO 1
        IF (PLYMIN.GE.PLYMAX) GOTO 1
        PLXMIN=PLXMIN*100.0
        PLXMAX=PLXMAX*100.0
        PLYMIN=PLYMIN*100.0
        PLYMAX=PLYMAX*100.0
      ENDIF

      P1=1
      P2=2
      P3=3

      WRITE(*,*) 'Liquid ink pen? (Y/*):'
      READ(*,100) SV
      LINKPEN=SV.EQ.'Y'.OR.SV.EQ.'y'

      IF (LINKPEN) THEN
        P1=1
        P2=1
        P3=1
        WRITE(*,*) 'Pen speed in cm/s (2..23):'
        READ(*,*) PS
        IF (PS.LT.2.0.OR.PS.GT.23.0) THEN
          PS=8.0
        ENDIF
        WRITE(VS,200) 'VS ',NINT(PS),';'
200     FORMAT(A3,I2,A1)
      ENDIF

      IF (LINKPEN) THEN
        WRITE(*,*) 'Put liquid ink pen in slot 1.'
      ELSE
        WRITE(*,*) 'Put pens in slot 1,2 and 3.'
      ENDIF

      WRITE(*,*) 'When plotter ready press any key!'
2     ANS=KBDCHK()
      IF (ANS.EQ.0) THEN
        GOTO 2
      ENDIF
      ANS=KBDINC()
C
      IF (LINKPEN) THEN
        CALL RRDGLC(VS,6)
      ELSE
        CALL RRDGLC('VS 23;',6)
      ENDIF

      X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
      Y0=NINT(PLYMIN+0.26*(PLYMAX-PLYMIN))
      XL=NINT(0.99*(PLXMAX-PLXMIN))
      YL=NINT(0.73*(PLYMAX-PLYMIN))

      XMIN=X(1)
      XMAX=X(N)
      YMIN=Y(1)
      YMAX=Y(1)

      DX=(XMAX-XMIN)/(FVECRANGE-1)
      DO 10 I=1,FVECRANGE
        XI=(I-1)*DX+XMIN
        FVEC(I)=F(THETA,L,XI)
        YMIN=MIN(YMIN,FVEC(I))
        YMAX=MAX(YMAX,FVEC(I))
10    CONTINUE

      DO 20 I=1,N
        IF (ERRORBAR) THEN
          YMIN=MIN(YMIN,Y(I)-SIGMA(I))
          YMAX=MAX(YMAX,Y(I)+SIGMA(I))
        ELSE
          YMIN=MIN(YMIN,Y(I))
          YMAX=MAX(YMAX,Y(I))
        ENDIF
20    CONTINUE

      IF (YMAX-YMIN.LT.1.0E-20) THEN
        YMAX=YMAX+0.5
        YMIN=YMIN-0.5
      ENDIF

      CALL RLNTYP(0)
      CALL RASCAL(0)
      CALL RMOVE(0.0,1500.0)
      CALL RPENCH(P2)

      DO 30 I=1,N
        PX=XP(X(I))
        IF (ERRORBAR) THEN
          IF (MOD(I,2).EQ.0) THEN
            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)=0.0
196         IF (SIGMA(I).NE.0.0) 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.1.0E-20) THEN
212         YMAX=YMAX+0.5
213         YMIN=YMIN-0.5
214       ENDIF
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       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(0.0)
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       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)
```

```
      ENDIF

      DO 90 I=1,2
        CALL RMOVE(PX,PY)
        PX=X0+XL
        CALL RDRAW(PX,PY)
        PY=Y0+YL
        CALL RDRAW(PX,PY)
        PX=X0
        CALL RDRAW(PX,PY)
        PY=Y0
        CALL RDRAW(PX,PY)
90    CONTINUE

      CALL RRDGLC('VS 23;',6)
      CALL RPENCH(0)
      CALL RHOME

      RETURN
      END
```

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

```
      SUBROUTINE ROLPLOT(Y,SIGMA,X,R,N,F,THETA,L,ERRORBAR)
C     Anders Persson, 1988

      IMPLICIT   LOGICAL (A-Z)
      INTEGER    FVECRANGE
      REAL*8     PLXMIN,PLXMAX,PLYMIN,PLYMAX
      PARAMETER  ( FVECRANGE=500 )
      INTEGER    N,L,I,X0,Y0,XL,YL,P1,P2,P3
      REAL*8     Y(1:N),SIGMA(1:N),X(1:N),R(1:N),F,THETA(1:L),XMIN,XMAX
      REAL*8     FVEC(1:FVECRANGE),SLASK,DX,XI,XP,YP,YMIN,YMAX,A
      REAL       PX,PY,PS
      INTEGER*2  ANS,KBDCHK,KBDINC
      LOGICAL    ERRORBAR,LINKPEN
      CHARACTER  SV*1,VS*6

      XP(A)=NINT(((A-XMIN)/(XMAX-XMIN))*XL+X0)
      YP(A)=NINT(((A-YMIN)/(YMAX-YMIN))*YL+Y0)

      WRITE(*,*) 'Standard paper size? (Y/*):'
      READ(*,100) SV
100   FORMAT(A)

      IF (SV.EQ.'Y'.OR.SV.EQ.'y') THEN
        WRITE(*,*) 'Size A3 or A4 (3/4):'
        READ(*,100) SV
        IF (SV.EQ.'3') THEN
          PLXMIN=0.0
          PLXMAX=3600.0
          PLYMIN=0.0
          PLYMAX=2600.0
        ELSE
          PLXMIN=0.0
          PLXMAX=2600.0
          PLYMIN=0.0
          PLYMAX=1800.0
        ENDIF
      ELSE
1       WRITE(*,*) 'Enter plot window (in cm.).'
        WRITE(*,*) 'Xmin, Ymin, Xmax, Ymax:'
        READ(*,*,ERR=1,END=1) PLXMIN,PLYMIN,PLXMAX,PLYMAX
        IF (PLXMIN.LT.0.0.OR.PLXMIN.GE.38.0) GOTO 1
        IF (PLYMIN.LT.0.0.OR.PLYMIN.GE.27.0) GOTO 1
        IF (PLXMAX.LT.0.0.OR.PLXMAX.GT.38.0) GOTO 1
        IF (PLYMAX.LT.0.0.OR.PLYMAX.GT.27.0) GOTO 1
        IF (PLXMIN.GE.PLXMAX) GOTO 1
        IF (PLYMIN.GE.PLYMAX) GOTO 1
        PLXMIN=PLXMIN*100.0
        PLXMAX=PLXMAX*100.0
        PLYMIN=PLYMIN*100.0
        PLYMAX=PLYMAX*100.0
      ENDIF

      P1=1
      P2=2
      P3=3

      WRITE(*,*) 'Liquid ink pen? (Y/*):'
      READ(*,100) SV
      LINKPEN=SV.EQ.'Y'.OR.SV.EQ.'y'

      IF (LINKPEN) THEN
        P1=1
        P2=1
        P3=1
        WRITE(*,*) 'Pen speed in cm/s (2..23):'
        READ(*,*) PS
        IF (PS.LT.2.0.OR.PS.GT.23.0) THEN
          PS=8.0
        ENDIF
        WRITE(VS,200) 'VS ',NINT(PS),';'
200     FORMAT(A3,I2,A1)
      ENDIF

      IF (LINKPEN) THEN
        WRITE(*,*) 'Put liquid ink pen in slot 1.'
      ELSE
        WRITE(*,*) 'Put pens in slot 1,2 and 3.'
      ENDIF

      WRITE(*,*) 'When plotter ready press any key!'
2     ANS=KBDCHK()
      IF (ANS.EQ.0) THEN
        GOTO 2
      ENDIF
      ANS=KBDINC()

      IF (LINKPEN) THEN
        CALL RRDGLC(VS,6)
      ELSE
        CALL RRDGLC('VS 23;',6)
      ENDIF

      X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
      Y0=NINT(PLYMIN+0.26*(PLYMAX-PLYMIN))
      XL=NINT(0.99*(PLXMAX-PLXMIN))
      YL=NINT(0.73*(PLYMAX-PLYMIN))
```

```
      XMIN=X(1)
      XMAX=X(N)
      YMIN=Y(1)
      YMAX=Y(1)

      DX=(XMAX-XMIN)/(FVECRANGE-1)
      DO 10 I=1,FVECRANGE
        XI=(I-1)*DX+XMIN
        FVEC(I)=F(THETA,L,XI)
        YMIN=MIN(YMIN,FVEC(I))
        YMAX=MAX(YMAX,FVEC(I))
10    CONTINUE

      DO 20 I=1,N
        IF (ERRORBAR) THEN
          YMIN=MIN(YMIN,Y(I)-SIGMA(I))
          YMAX=MAX(YMAX,Y(I)+SIGMA(I))
        ELSE
          YMIN=MIN(YMIN,Y(I))
          YMAX=MAX(YMAX,Y(I))
        ENDIF
20    CONTINUE

      IF (YMAX-YMIN.LT.1.0D-20) THEN
        YMAX=YMAX+0.5D0
        YMIN=YMIN-0.5D0
      ENDIF

      CALL RLNTYP(0)
      CALL RASCAL(0)
      CALL RMOVE(0.0,1500.0)
      CALL RPENCH(P2)

      DO 30 I=1,N
        PX=XP(X(I))
        IF (ERRORBAR) THEN
          IF (MOD(I,2).EQ.0) THEN
            PY=YP(Y(I)+SIGMA(I))
          ELSE
            PY=YP(Y(I)-SIGMA(I))
          ENDIF
        ELSE
          PY=YP(Y(I))
        ENDIF
        CALL RMOVE(PX,PY)
        IF (ERRORBAR) THEN
          IF (MOD(I,2).EQ.0) THEN
            PY=YP(Y(I)-SIGMA(I))
          ELSE
            PY=YP(Y(I)+SIGMA(I))
          ENDIF
          CALL RDRAW(PX,PY)
        ELSE
          CALL RMARK(4)
        ENDIF
30    CONTINUE

      PY=YP(FVEC(1))
      PX=XP(XMIN)
      IF (.NOT.LINKPEN) THEN
        CALL RMOVE(0.0,1500.0)
        CALL RPENCH(P3)
      ENDIF

      CALL RMOVE(PX,PY)
      DO 40 I=1,FVECRANGE
        XI=(I-1)*DX+XMIN
        PX=XP(XI)
        PY=YP(FVEC(I))
        CALL RDRAW(PX,PY)
40    CONTINUE

      X0=PLXMIN
      Y0=NINT(PLYMIN+0.25*(PLYMAX-PLYMIN))
      XL=PLXMAX-PLXMIN
      YL=NINT(0.75*(PLYMAX-PLYMIN))

      PX=X0
      PY=Y0
      IF (.NOT.LINKPEN) THEN
        CALL RMOVE(0.0,1500.0)
        CALL RPENCH(P1)
      ENDIF

      DO 50 I=1,2
        CALL RMOVE(PX,PY)
        PX=X0+XL
        CALL RDRAW(PX,PY)
        PY=Y0+YL
        CALL RDRAW(PX,PY)
        PX=X0
        CALL RDRAW(PX,PY)
        PY=Y0
        CALL RDRAW(PX,PY)
50    CONTINUE

      DO 60 I=1,N
        R(I)=0.0D0
        IF (SIGMA(I).NE.0.0D0) THEN
          R(I)=(Y(I)-F(THETA,L,X(I)))/SIGMA(I)
        ENDIF
60    CONTINUE

      YMAX=R(1)
      YMIN=R(1)
      DO 70 I=1,N
        YMAX=MAX(YMAX,R(I))
        YMIN=MIN(YMIN,R(I))
70    CONTINUE

      YMAX=ABS(YMAX)
      YMAX=MAX(YMAX,ABS(YMIN))
      YMIN=-YMAX
      IF (YMAX-YMIN.LT.1.0D-20) THEN
        YMAX=YMAX+0.5D0
        YMIN=YMIN-0.5D0
      ENDIF

      X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
```

```
      YO=NINT(PLYMIN+0.01*(PLYMAX-PLYMIN))
      XL=NINT(0.99*(PLXMAX-PLXMIN))
      YL=NINT(0.22*(PLYMAX-PLYMIN))

      IF (.NOT.LINKPEN) THEN
        CALL RMOVE(0.0,1500.0)
        CALL RPENCH(P2)
      ENDIF

      DO 80 I=1,N
        IF (R(I).NE.0.0) THEN
          PX=XP(X(I))
          PY=YP(R(I))
          CALL RMOVE(PX,PY)
          CALL RMARK(4)
        ENDIF
80    CONTINUE

      PX=XP(X(1))
      PY=YP(0.0D0)
      IF (.NOT.LINKPEN) THEN
        CALL RMOVE(0.0,1500.0)
        CALL RPENCH(P3)
      ENDIF

      CALL RMOVE(PX,PY)
      PX=XP(X(N))
      CALL RDRAW(PX,PY)

      XO=PLXMIN
      YO=PLYMIN
      XL=PLXMAX-PLXMIN
      YL=NINT(0.24*(PLYMAX-PLYMIN))

      PX=XO
      PY=YO
      IF (.NOT.LINKPEN) THEN
        CALL RMOVE(0.0,1500.0)
        CALL RPENCH(P1)
      ENDIF

      DO 90 I=1,2
        CALL RMOVE(PX,PY)
        PX=XO+XL
        CALL RDRAW(PX,PY)
        PY=YO+YL
        CALL RDRAW(PX,PY)
        PX=XO
        CALL RDRAW(PX,PY)
        PY=YO
90    CONTINUE

      CALL RDRAW(PX,PY)
      CALL RRDGLC('VS 23;',6)
      CALL RPENCH(0)
      CALL RHOME

      RETURN
      END
```



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

      SUBROUTINE JETPLOT(Y,SIGMA,X,R,N,F,THETA,L,ERRORBAR)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     PLXMIN,PLXMAX,PLYMIN,PLYMAX,FVECRANGE
      PARAMETER   ( FVECRANGE=500 )
      INTEGER     I,N,L,XO,YO,XL,YL
      CHARACTER   MSG*12
      REAL        Y(1:N),SIGMA(1:N),X(1:N),R(1:N),F,THETA(1:L)
      REAL        FVEC(1:FVECRANGE),SLASK,DX,XI,XP,YP
      REAL        YMIN,YMAX,A,PX,PY,XMIN,XMAX
      LOGICAL     ERRORBAR,LANDSCAPE
      INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
      COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

      XP(A)=NINT(((A-XMIN)/(XMAX-XMIN))*XL+XO)
      YP(A)=NINT(((A-YMIN)/(YMAX-YMIN))*YL+YO)
C
      PLXMIN=0
      PLXMAX=719
      PLYMIN=0
      PLYMAX=1119
      IF (LANDSCAPE) THEN
        PLXMAX=1119
        PLYMAX=719
      ENDIF
C
      XO=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
      YO=NINT(PLYMIN+0.26*(PLYMAX-PLYMIN))
      XL=NINT(0.99*(PLXMAX-PLXMIN))
      YL=NINT(0.73*(PLYMAX-PLYMIN))

      XMIN=X(1)
      XMAX=X(N)
      YMIN=Y(1)
      YMAX=Y(1)

      DX=(XMAX-XMIN)/(FVECRANGE-1)
      DO 10 I=1,FVECRANGE
        XI=(I-1)*DX+XMIN
        FVEC(I)=F(THETA,L,XI)
        YMIN=MIN(YMIN,FVEC(I))
        YMAX=MAX(YMAX,FVEC(I))
10    CONTINUE

      DO 20 I=1,N
        IF (ERRORBAR) THEN
          YMIN=MIN(YMIN,Y(I)-SIGMA(I))
          YMAX=MAX(YMAX,Y(I)+SIGMA(I))
        ELSE
          YMIN=MIN(YMIN,Y(I))
          YMAX=MAX(YMAX,Y(I))
        ENDIF
20    CONTINUE

      IF (YMAX-YMIN.LT.1.0E-20) THEN
        YMAX=YMAX+0.5
```

```
      YMIN=YMIN-0.5
    ENDIF

    DO 30 I=1,N
      PX=XP(X(I))
      IF (ERRORBAR) THEN
        PY=YP(Y(I)+SIGMA(I))
      ELSE
        PY=YP(Y(I))
      ENDIF
      CALL HPMOVE(PX,PY)
      IF (ERRORBAR) THEN
        PY=YP(Y(I)-SIGMA(I))
        CALL HPDRAW(PX,PY)
      ELSE
        CALL HPMARK(PX,PY)
      ENDIF
30  CONTINUE

    PY=YP(FVEC(1))
    PX=XP(XMIN)
    CALL HPMOVE(PX,PY)
    DO 40 I=1,FVECRANGE
      XI=(I-1)*DX+XMIN
      PX=XP(XI)
      PY=YP(FVEC(I))
      CALL HPDRAW(PX,PY)
40  CONTINUE

    X0=PLXMIN
    Y0=NINT(PLYMIN+0.25*(PLYMAX-PLYMIN))
    XL=PLXMAX-PLXMIN
    YL=NINT(0.75*(PLYMAX-PLYMIN))

    PX=X0
    PY=Y0
    CALL HPMOVE(PX,PY)
    PX=X0+XL
    CALL HPDRAW(PX,PY)
    PY=Y0+YL
    CALL HPDRAW(PX,PY)
    PX=X0
    CALL HPDRAW(PX,PY)
    PY=Y0
    CALL HPDRAW(PX,PY)

    WRITE(MSG,100) YMIN
100 FORMAT(F12.2)
    CALL HPTBOX(PX,PY,12,MSG)

    PY=Y0+YL-13
    WRITE(MSG,100) YMAX
    CALL HPTBOX(PX,PY,12,MSG)

    DO 50 I=1,N
      R(I)=0.0
      IF (SIGMA(I).NE.0.0) THEN
        R(I)=(Y(I)-F(THETA,L,X(I)))/SIGMA(I)
      ENDIF
50  CONTINUE

    YMAX=R(1)
    YMIN=R(1)
    DO 60 I=1,N
      YMAX=MAX(YMAX,R(I))
      YMIN=MIN(YMIN,R(I))
60  CONTINUE

    YMAX=ABS(YMAX)
    YMAX=MAX(YMAX,ABS(YMIN))
    YMIN=-YMAX
    IF (YMAX-YMIN.LT.1.0E-20) THEN
      YMAX=YMAX+0.5
      YMIN=YMIN-0.5
    ENDIF

    X0=NINT(PLXMIN+0.005*(PLXMAX-PLXMIN))
    Y0=NINT(PLYMIN+0.01*(PLYMAX-PLYMIN))
    XL=NINT(0.99*(PLXMAX-PLXMIN))
    YL=NINT(0.22*(PLYMAX-PLYMIN))

    DO 70 I=1,N
      IF (R(I).NE.0.0) THEN
        PX=XP(X(I))
        PY=YP(R(I))
        CALL HPMARK(PX,PY)
      ENDIF
70  CONTINUE

    PX=XP(X(1))
    PY=YP(0.0)
    CALL HPMOVE(PX,PY)
    PX=XP(X(N))
    CALL HPDRAW(PX,PY)

    X0=PLXMIN
    Y0=PLYMIN
    XL=PLXMAX-PLXMIN
    YL=NINT(0.24*(PLYMAX-PLYMIN))

    PX=X0
    PY=Y0
    CALL HPMOVE(PX,PY)
    PX=X0+XL
    CALL HPDRAW(PX,PY)
    PY=Y0+YL
    CALL HPDRAW(PX,PY)
    PX=X0
    CALL HPDRAW(PX,PY)
    PY=Y0
    CALL HPDRAW(PX,PY)

    WRITE(MSG,100) YMIN
    CALL HPTBOX(PX,PY,12,MSG)

    PY=Y0+YL-13
    WRITE(MSG,100) YMAX
    CALL HPTBOX(PX,PY,12,MSG)

    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       ENDIF
  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         ENDIF
  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         ENDIF
  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 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
2 C                                      C
3 C     Library:  HERCLIB.LIB            C
4 C                                      C
5 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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
```

```
PCODE           ENDS
                END
;
                TITLE     CLRSCR
; Anders Persson, 1986
; FORTRAN:      SUBROUTINE CLRSCR
;
LADATA          SEGMENT   'DATA'
                DB        'CLRSCR  '
SP_SAVE         DW        0
                DD        0
LADATA          ENDS
PCODE           SEGMENT   'CODE'
                ASSUME    CS:PCODE,DS:LADATA
                DW        SEG  LADATA
CLRSCR          PROC      FAR
                PUBLIC    CLRSCR
                MOV       AX,LADATA
                MOV       DS,AX
                MOV       SP_SAVE,SP
                MOV       AH,42H
                INT       10H
                RET
CLRSCR          ENDP
PCODE           ENDS
                END
;
                TITLE     DISP
; Anders Persson, 1986
; FORTRAN:      SUBROUTINE DISP(P)
;               INTEGER*2 P
;
PARBLK          STRUC
                PARB          DD        ?
PARBLK          ENDS
LADATA          SEGMENT   'DATA'
                DB        'DISP    '
SP_SAVE         DW        0
                DD        0
LADATA          ENDS
PCODE           SEGMENT   'CODE'
                ASSUME    CS:PCODE,DS:LADATA
                DW        SEG  LADATA
DISP            PROC      FAR
                PUBLIC    DISP
                MOV       AX,LADATA
                MOV       DS,AX
                MOV       SP_SAVE,SP
                LDS       SI,ES:PARB[BX]
                MOV       AX,[SI]
                MOV       AH,45H
                INT       10H
                RET
DISP            ENDP
PCODE           ENDS
                END
;
                TITLE     DLINE
; Anders Persson, 1986
; FORTRAN:      SUBROUTINE DLINE(X,Y)
;               INTEGER*2 X,Y
;
PARBLK          STRUC
PARX                DD        ?
PARY                DD        ?
PARBLK          ENDS
LADATA          SEGMENT   'DATA'
                DB        'DLINE   '
SP_SAVE         DW        0
                DD        0
LADATA          ENDS
PCODE           SEGMENT   'CODE'
                ASSUME    CS:PCODE,DS:LADATA
                DW        SEG  LADATA
DLINE           PROC      FAR
                PUBLIC    DLINE
                MOV       AX,LADATA
                MOV       DS,AX
                MOV       SP_SAVE,SP
                LDS       SI,ES:PARX[BX]
                MOV       DI,[SI]
                LDS       SI,ES:PARY[BX]
                MOV       BP,347
                SUB       BP,[SI]
                MOV       AH,49H
                INT       10H
                RET
DLINE           ENDP
PCODE           ENDS
                END
;
                TITLE     FILL
; Anders Persson, 1986
; FORTRAN:      SUBROUTINE FILL(X,Y)
;               INTEGER*2 X,Y
;
PARBLK          STRUC
PARX                DD        ?
PARY                DD        ?
PARBLK          ENDS
LADATA          SEGMENT   'DATA'
                DB        'FILL    '
SP_SAVE         DW        0
                DD        0
LADATA          ENDS
PCODE           SEGMENT   'CODE'
                ASSUME    CS:PCODE,DS:LADATA
                DW        SEG  LADATA
FILL            PROC      FAR
                PUBLIC    FILL
                MOV       AX,LADATA
                MOV       DS,AX
                MOV       SP_SAVE,SP
                LDS       SI,ES:PARX[BX]
                MOV       DI,[SI]
                LDS       SI,ES:PARY[BX]
                MOV       BP,347
                SUB       BP,[SI]
                MOV       AH,4EH
                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
```

```
C     LEVEL           ENDP
C     PCODE           ENDS
C                     END
C     ;
C                     TITLE     MOVE
C     ; Anders Persson, 1986
C     ; FORTRAN:     SUBROUTINE MOVE(X,Y)
C     ;               INTEGER*2 X,Y
C     ;
C     PARBLK          STRUC
C                     PARX          DD        ?
C                     PARY          DD        ?
C     PARBLK          ENDS
C     LADATA          SEGMENT   'DATA'
C                     DB        'MOVE    '
C     SP_SAVE         DW        0
C                     DD        0
C     LADATA          ENDS
C     PCODE           SEGMENT   'CODE'
C                     ASSUME    CS:PCODE,DS:LADATA
C                     DW        SEG  LADATA
C     MOVE            PROC      FAR
C                     PUBLIC    MOVE
C                     MOV       AX,LADATA
C                     MOV       DS,AX
C                     MOV       SP_SAVE,SP
C                     LDS       SI,ES:PARX[BX]
C                     MOV       DI,[SI]
C                     LDS       SI,ES:PARY[BX]
C                     MOV       BP,347
C                     SUB       BP,[SI]
C                     MOV       AH,48H
C                     INT       10H
C                     RET
C     MOVE            ENDP
C     PCODE           ENDS
C                     END
C     ;
C                     TITLE     PLOT
C     ; Anders Persson, 1986
C     ; FORTRAN:     SUBROUTINE PLOT(X,Y)
C     ;               INTEGER*2 X,Y
C     ;
C     PARBLK          STRUC
C                     PARX          DD        ?
C                     PARY          DD        ?
C     PARBLK          ENDS
C     LADATA          SEGMENT   'DATA'
C                     DB        'PLOT    '
C     SP_SAVE         DW        0
C                     DD        0
C     LADATA          ENDS
C     PCODE           SEGMENT   'CODE'
C                     ASSUME    CS:PCODE,DS:LADATA
C                     DW        SEG  LADATA
C     PLOT            PROC      FAR
C                     PUBLIC    PLOT
C                     MOV       AX,LADATA
C                     MOV       DS,AX
C                     MOV       SP_SAVE,SP
C                     LDS       SI,ES:PARX[BX]
C                     MOV       DI,[SI]
C                     LDS       SI,ES:PARY[BX]
C                     MOV       BP,347
C                     SUB       BP,[SI]
C                     MOV       AH,46H
C                     INT       10H
C                     RET
C     PLOT            ENDP
C     PCODE           ENDS
C                     END
C     ;
C                     TITLE     TMODE
C     ; Anders Persson, 1986
C     ; FORTRAN:     SUBROUTINE TMODE
C     ;
C     LADATA          SEGMENT   'DATA'
C                     DB        'TMODE   '
C     SP_SAVE         DW        0
C                     DD        0
C     LADATA          ENDS
C     PCODE           SEGMENT   'CODE'
C                     ASSUME    CS:PCODE,DS:LADATA
C                     DW        SEG  LADATA
C     TMODE           PROC      FAR
C                     PUBLIC    TMODE
C                     MOV       AX,LADATA
C                     MOV       DS,AX
C                     MOV       SP_SAVE,SP
C                     MOV       AH,41H
C                     INT       10H
C                     RET
C     TMODE           ENDP
C     PCODE           ENDS
C                     END
C     ;
C                     TITLE     HRDCPY
C     ; Anders Persson, 1986
C     ; FORTRAN:     SUBROUTINE HRDCPY(B)
C     ;               INTEGER*2 B
C     ;
C     PARBLK          STRUC
C                     PARB          DD        ?
C     PARBLK          ENDS
C     LADATA          SEGMENT   'DATA'
C                     DB        'HRDCPY  '
C     SP_SAVE         DW        0
C                     DD        0
C     LADATA          ENDS
C     PCODE           SEGMENT   'CODE'
C                     ASSUME    CS:PCODE,DS:LADATA
C                     DW        SEG  LADATA
C     HRDCPY          PROC      FAR
C                     PUBLIC    HRDCPY
C                     MOV       AX,LADATA
C                     MOV       DS,AX
C                     MOV       SP_SAVE,SP
C                     LDS       SI,ES:PARB[BX]
C                     MOV       BX,[SI]
C                     MOV       BH,00H
```

```
C                   XOR       AX,AX
C                   MOV       ES,AX
C                   MOV       BP,2+4*05H
C                   MOV       AX,ES:[BP]
C                   MOV       ES,AX
C                   MOV       BP,130H
C                   MOV       ES:[BP],BL
C                   INT       5
C                   RET
C     HRDCPY        ENDP
C     PCODE         ENDS
C                   END
C     ;
C                   TITLE     ONECHR
C     ; Anders Persson, 1986
C     ; FORTRAN:    SUBROUTINE ONECHR(X,Y,I)
C     ;             INTEGER*2 X,Y,I
C     ;
C     PARBLK        STRUC
C                   PARX          DD        ?
C                   PARY          DD        ?
C                   PARI          DD        ?
C     PARBLK        ENDS
C     LADATA        SEGMENT   'DATA'
C                   DB        'ONECHR  '
C     SP_SAVE       DW        0
C                   DD        0
C     LADATA        ENDS
C     PCODE         SEGMENT   'CODE'
C                   ASSUME    CS:PCODE,DS:LADATA
C                   DW        SEG  LADATA
C     ONECHR        PROC      FAR
C                   PUBLIC    ONECHR
C                   MOV       AX,LADATA
C                   MOV       DS,AX
C                   MOV       SP_SAVE,SP
C                   LDS       SI,ES:PARX[BX]
C                   MOV       DI,[SI]
C                   LDS       SI,ES:PARY[BX]
C                   MOV       BP,347
C                   SUB       BP,[SI]
C                   LDS       SI,ES:PARI[BX]
C                   MOV       AX,[SI]
C                   MOV       AH,4BH
C                   INT       10H
C                   RET
C     ONECHR        ENDP
C     PCODE         ENDS
C                   END
C
C      SUBROUTINE TEXT(X,Y,LEN,MSG)
CC     Anders Persson, 1986
C
C      IMPLICIT      LOGICAL (A-Z)
C      INTEGER*2     X,Y,LEN,IASC,IX,IY,I
C      CHARACTER*(*) MSG
C
C      DO 10 I=1,LEN
C        IX=X+(I-1)*9
C        IY=Y
C        IASC=ICHAR(MSG(I:I))
C        CALL ONECHR(IX,IY,IASC)
C10    CONTINUE
C
C      RETURN
C      END
C
C                   TITLE     KBDCHK
C     ; Anders Persson, 1986
C     ; FORTRAN:    INTEGER*2 FUNCTION KBDCHK()
C     ;
C     LADATA        SEGMENT   'DATA'
C                   DB        'KBDCHK  '
C     SP_SAVE       DW        0
C                   DD        0
C     LADATA        ENDS
C     PCODE         SEGMENT   'CODE'
C                   ASSUME    CS:PCODE,DS:LADATA
C                   DW        SEG  LADATA
C     KBDCHK        PROC      FAR
C                   PUBLIC    KBDCHK
C                   MOV       AX,LADATA
C                   MOV       DS,AX
C                   MOV       SP_SAVE,SP
C                   MOV       AH,0BH
C                   INT       21H
C                   MOV       AH,00H
C                   RET
C     KBDCHK        ENDP
C     PCODE         ENDS
C                   END
C     ;
C                   TITLE     KBDIN
C     ; Anders Persson, 1986
C     ; FORTRAN:    INTEGER*2 FUNCTION KBDIN()
C     ;
C     LADATA        SEGMENT   'DATA'
C                   DB        'KBDIN   '
C     SP_SAVE       DW        0
C                   DD        0
C     LADATA        ENDS
C     PCODE         SEGMENT   'CODE'
C                   ASSUME    CS:PCODE,DS:LADATA
C                   DW        SEG  LADATA
C     KBDIN         PROC      FAR
C                   PUBLIC    KBDIN
C                   MOV       AX,LADATA
C                   MOV       DS,AX
C                   MOV       SP_SAVE,SP
C                   MOV       AH,07H
C                   INT       21H
C                   MOV       AH,0H
C                   CMP       AL,0H
C                   JZ        AGAIN
C                   RET
C     AGAIN:        MOV       AH,07H          ;EXTENDED ASCII CODE
C                   INT       21H             ;F1..F10, CURSOR KEYS ETC.
C                   MOV       AH,0H
C                   ADD       AX,1000
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 .

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: PLLIB.FOR   Options: /LBZ          08/02/88 14:43:56

C     ****************************************************************
C     *  Name: PLLIB.FOR, PLLIB.LIB                                  *
C     *  Program library for Watanabe and Roland pen plotter.        *
C     *  Anders Persson, 1988                                        *
C     ****************************************************************
C
C     Assembler routines:
C     ****************************************************************
C     Filename: WINITP.ASM
C     FORTRAN CALL: CALL WINITP, no parameters
C     Initialize paral. port. for Watanabe plotter
C     must be called prior to any other plot routine if
C     the Watanabe plotter is to be used
C
C                   TITLE           WINITP
C  ;Anders Persson, 1988
C
C   LADATA          SEGMENT         'DATA'
C                   DB              'WINITP  '
C   SP_SAVE         DW              0
C                   DD              0
C   LADATA          ENDS
C   PCODE           SEGMENT         'CODE'
C                   ASSUME          CS:PCODE,DS:LADATA
C                   DW              SEG  LADATA
C   WINITP          PROC            FAR
C                   PUBLIC          WINITP
C                   MOV             AX,LADATA
C                   MOV             DS,AX
C                   MOV             SP_SAVE,SP
C   STATPORTC       =               770
C   CTRLPORT        =               771
C   CTRLDATA        =               10010001B
C                   MOV             DX,CTRLPORT
C                   MOV             AL,CTRLDATA
C                   OUT             DX,AL           ;INITIALIZE PORT
C                   MOV             DX,STATPORTC
C                   MOV             AL,00000000B
C                   OUT             DX,AL           ;ENSURE STROBE=0
C                   RET
C   WINITP          ENDP
C   PCODE           ENDS
C                   END
C
C     ****************************************************************
C     Filename: WATOUT.ASM
C     FORTRAN CALL: CALL WATOUT(D), INTEGER*2 D
C     The routine sends an ascii character (D=ICHAR(CHARACTER)) to
C     the parallell port, only used in Watanabe plot routines
C
C                   TITLE           WATOUT
C  ;Anders Persson, 1988
C   PARBLK          STRUC
C                   PAR             DD              ?
C   PARBLK          ENDS
C   LADATA          SEGMENT         'DATA'
C                   DB              'WATOUT  '
C   SP_SAVE         DW              0
C                   DD              0
C   LADATA          ENDS
C   PCODE           SEGMENT         'CODE'
C                   ASSUME          CS:PCODE,DS:LADATA
C                   DW              SEG  LADATA
C   WATOUT          PROC            FAR
C                   PUBLIC          WATOUT
C                   MOV             AX,LADATA
C                   MOV             DS,AX
C                   MOV             SP_SAVE,SP
C   UTPORTB         =               769
C   STATPORTC       =               770
C                   MOV             DX,STATPORTC            ; CHECK
C   BUSY:           IN              AL,DX                   ; PLOTTER
C                   AND             AL,00001000B            ; READY
C                   JNZ             BUSY                    ; = BUSY=0
C  ;
C                   MOV             DX,UTPORTB              ; PUT
C                   LDS             SI,ES:PAR[BX]           ; DATA
C                   MOV             AX,[SI]                 ; ON
C                   AND             AL,01111111B            ; OUTPUT
C                   OUT             DX,AL                   ; LINES
C  ;
C                   MOV             DX,STATPORTC            ; SEND
C                   MOV             AL,00010000B            ;
C                   OUT             DX,AL                   ; STROBE
C                   MOV             AL,00000000B            ;
C                   OUT             DX,AL                   ; PULSE
C                   RET
C   WATOUT          ENDP
C   PCODE           ENDS
C                   END
C

      SUBROUTINE RHOME
C     Anders Persson, 1988

      OPEN(9,FILE='PRN')
      WRITE(9,100) 'H'
  100 FORMAT(A1)
      CLOSE(9)

      RETURN
      END

      SUBROUTINE WHOME
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER*2   D

      D=ICHAR('H')
      CALL WATOUT(D)
      D=10
      CALL WATOUT(D)

      RETURN
      END
```

```
      SUBROUTINE RDRAW(X,Y)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y

      IF (ABS(X).GT.16383.0.OR.ABS(Y).GT.16383.0) THEN
        WRITE(*,*) 'Parameter out of range in RDRAW subroutine.'
        STOP
      ENDIF

      OPEN(9,FILE='PRN')
      WRITE(9,100) 'D',X,',',Y
100   FORMAT(A1,F11.4,A1,F11.4)
      CLOSE(9)

      RETURN
      END

      SUBROUTINE WDRAW(X,Y)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y
      INTEGER*2   IX,IY,D,I
      CHARACTER   LINE*11

      IF (X.GT.3600.0) THEN
        IX=3600
      ELSEIF(X.LT.0.0) THEN
        IX=0
      ELSE
        IX=NINT(X)
      ENDIF

      IF (Y.GT.2600.0) THEN
        IY=2600
      ELSEIF(Y.LT.0.0) THEN
        IY=0
      ELSE
        IY=NINT(Y)
      ENDIF

      WRITE(LINE,100) 'D',IX,',',IY,CHAR(10)
100   FORMAT(A,I4,A,I4,A)

      DO 10 I=1,11
        D=ICHAR(LINE(I:I))
        CALL WATOUT(D)
10    CONTINUE

      RETURN
      END

      SUBROUTINE RMOVE(X,Y)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y

      IF (ABS(X).GT.16383.0.OR.ABS(Y).GT.16383.0) THEN
        WRITE(*,*) 'Parameter out of range in RMOVE subroutine.'
        STOP
      ENDIF

      OPEN(9,FILE='PRN')
      WRITE(9,100) 'M',X,',',Y
100   FORMAT(A1,F11.4,A1,F11.4)
      CLOSE(9)

      RETURN
      END

      SUBROUTINE WMOVE(X,Y)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y
      INTEGER*2   IX,IY,D,I
      CHARACTER   LINE*11

      IF (X.GT.3600.0) THEN
        IX=3600
      ELSEIF(X.LT.0.0) THEN
        IX=0
      ELSE
        IX=NINT(X)
      ENDIF

      IF (Y.GT.2600.0) THEN
        IY=2600
      ELSEIF(Y.LT.0.0) THEN
        IY=0
      ELSE
        IY=NINT(Y)
      ENDIF

      WRITE(LINE,100) 'M',IX,',',IY,CHAR(10)
100   FORMAT(A,I4,A,I4,A)

      DO 10 I=1,11
        D=ICHAR(LINE(I:I))
        CALL WATOUT(D)
10    CONTINUE

      RETURN
      END

      SUBROUTINE RRDRAW(X,Y)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y

      IF (ABS(X).GT.16383.0.OR.ABS(Y).GT.16383.0) THEN
        WRITE(*,*) 'Parameter out of range in RRDRAW subroutine.'
        STOP
      ENDIF

      OPEN(9,FILE='PRN')
```

```
      WRITE(9,100) 'I',X,',',Y
100   FORMAT(A1,F11.4,A1,F11.4)
      CLOSE(9)

      RETURN
      END

      SUBROUTINE WRDRAW(X,Y)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y
      INTEGER*2   IX,IY,D,I
      CHARACTER   LINE*13

      IF (ABS(X).GT.3600.0) THEN
        IX=SIGN(3600.0,X)
      ELSE
        IX=NINT(X)
      ENDIF

      IF (ABS(Y).GT.2600.0) THEN
        IY=SIGN(2600.0,Y)
      ELSE
        IY=NINT(Y)
      ENDIF

      WRITE(LINE,100) 'I',IX,',',IY,CHAR(10)
100   FORMAT(A,I5,A,I5,A)

      DO 10 I=1,13
        D=ICHAR(LINE(I:I))
        CALL WATOUT(D)
10    CONTINUE

      RETURN
      END

      SUBROUTINE RRMOVE(X,Y)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y

      IF (ABS(X).GT.16383.0.OR.ABS(Y).GT.16383.0) THEN
        WRITE(*,*) 'Parameter out of range in RRMOVE subroutine.'
        STOP
      ENDIF

      OPEN(9,FILE='PRN')
      WRITE(9,100) 'R',X,',',Y
100   FORMAT(A1,F11.4,A1,F11.4)
      CLOSE(9)

      RETURN
      END

      SUBROUTINE WRMOVE(X,Y)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y
      INTEGER*2   IX,IY,D,I
      CHARACTER   LINE*13

      IF (ABS(X).GT.3600.0) THEN
        IX=SIGN(3600.0,X)
      ELSE
        IX=NINT(X)
      ENDIF

      IF (ABS(Y).GT.2600.0) THEN
        IY=SIGN(2600.0,Y)
      ELSE
        IY=NINT(Y)
      ENDIF

      WRITE(LINE,100) 'R',IX,',',IY,CHAR(10)
100   FORMAT(A,I5,A,I5,A)

      DO 10 I=1,13
        D=ICHAR(LINE(I:I))
        CALL WATOUT(D)
10    CONTINUE

      RETURN
      END

      SUBROUTINE RLNTYP(P)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     P,IP

      IF (ABS(P).GE.6) THEN
        IP=0
      ELSE
        IP=P
      ENDIF

      OPEN(9,FILE='PRN')
      WRITE(9,100) 'L',IP
100   FORMAT(A1,I2)
      CLOSE(9)

      RETURN
      END

      SUBROUTINE WLNTYP(P)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     P
      INTEGER*2   D,LF,L

      LF=10
      D=ICHAR('0')
      IF (P.NE.0) THEN
        D=ICHAR('1')
      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
```

```
        D=ICHAR(LINE(I:I))
        CALL WATOUT(D)
10    CONTINUE

      RETURN
      END

      SUBROUTINE RPRINT(MSG,LEN)
C     Anders Persson, 1988

      IMPLICIT      LOGICAL (A-Z)
      INTEGER       LEN,I
      CHARACTER*(*) MSG

      OPEN(9,FILE='PRN')
      WRITE(9,100) 'P',(MSG(I:I), I=1,LEN)
100   FORMAT(10000(A1))
      CLOSE(9)

      RETURN
      END

      SUBROUTINE WPRINT(MSG,LEN)
C     Anders Persson, 1988

      IMPLICIT      LOGICAL (A-Z)
      INTEGER       LEN
      CHARACTER*(*) MSG
      INTEGER*2     D,I

      D=ICHAR('P')
      CALL WATOUT(D)

      DO 10 I=1,LEN
        D=ICHAR(MSG(I:I))
        CALL WATOUT(D)
10    CONTINUE

      D=10
      CALL WATOUT(D)

      RETURN
      END

      SUBROUTINE RASCAL(N)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     N,IN

      IF (N.LT.0.OR.N.GT.127) THEN
        IN=3
      ELSE
        IN=N
      ENDIF

      OPEN(9,FILE='PRN')
      WRITE(9,100) 'S',IN
100   FORMAT(A1,I3)
      CLOSE(9)

      RETURN
      END

      SUBROUTINE WASCAL(N)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     N
      INTEGER*2   D,I
      CHARACTER   LINE*4

      IF (N.LT.0) THEN
        D=0
      ELSEIF(N.GT.15) THEN
        D=15
      ELSE
        D=N
      ENDIF

      WRITE(LINE,100) 'S',D,CHAR(10)
100   FORMAT(A,I2,A)

      DO 10 I=1,4
        D=ICHAR(LINE(I:I))
        CALL WATOUT(D)
10    CONTINUE

      RETURN
      END

      SUBROUTINE RAROT(Q)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     Q,IQ

      IF (Q.LT.0.OR.Q.GT.3) THEN
        IQ=0
      ELSE
        IQ=Q
      ENDIF

      OPEN(9,FILE='PRN')
      WRITE(9,100) 'Q',IQ
100   FORMAT(A1,I1)
      CLOSE(9)

      RETURN
      END

      SUBROUTINE WAROT(Q)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     Q
      INTEGER*2   D,I
      CHARACTER   LINE*3

      IF (Q.LT.0) THEN
```

```
598         D=0
599       ELSEIF(Q.GT.3) THEN
600         D=3
601       ELSE
602         D=Q
603       ENDIF
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       ENDIF
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       ENDIF
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       ENDIF
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
687       IMPLICIT    LOGICAL (A-Z)
688       REAL        X,Y,R,T1,T2,DT
689       CHARACTER   K*1
690
691       K=','
692
693       IF (ABS(X)  .GT. 16383.0 .OR. ABS(Y)  .GT. 16383.0 .OR.
694      f    ABS(R)  .GT. 16383.0 .OR. ABS(T1) .GT. 32767.0 .OR.
695      f    ABS(T2) .GT. 32767.0 .OR. ABS(DT) .LT. 1.0     .OR.
696      f    ABS(DT) .GT. 179.9999                                ) THEN
697
698         WRITE(*,*) 'Parameter out of range in RCIRC subroutine.'
699         STOP
700       ENDIF
701
702       OPEN(9,FILE='PRN')
703       IF (DT.GT.0.0) THEN
704         WRITE(9,100) 'C',X,K,Y,K,R,K,T1,K,T2,K,DT
705       ELSE
706         WRITE(9,100) 'C',X,K,Y,K,R,K,T1,K,T2
707       ENDIF
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)
```

```
      REAL        R,T1,T2,DT
      CHARACTER   K*1

      K=','

      IF (ABS(R)  .GT. 16383.0 .OR. ABS(T1) .GT. 32767.0 .OR.
     f    ABS(T2) .GT. 32767.0 .OR. ABS(DT) .LT. 1.0     .OR.
     f    ABS(DT) .GT. 179.9999                              ) THEN

        WRITE(*,*) 'Parameter out of range in RRCIRC subroutine.'
        STOP
      ENDIF

      OPEN(9,FILE='PRN')
      IF (DT.GT.0.0) THEN
        WRITE(9,100) 'E',R,K,T1,K,T2,K,DT
      ELSE
        WRITE(9,100) 'E',R,K,T1,K,T2
      ENDIF
100   FORMAT(4(A1,F11.4))
      CLOSE(9)

      RETURN
      END

      SUBROUTINE RCCENT(X,Y)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y

      IF (ABS(X).GT.16383.0.OR.ABS(Y).GT.16383.0) THEN
        WRITE(*,*) 'Parameter out of range in RCCENT subroutine.'
        STOP
      ENDIF

      OPEN(9,FILE='PRN')
      WRITE(9,100) 'A',X,',',Y
100   FORMAT(A1,F11.4,A1,F11.4)
      CLOSE(9)

      RETURN
      END

      SUBROUTINE RACIR(R,T1,T2,DT)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        R,T1,T2,DT
      CHARACTER   K*1

      K=','

      IF (ABS(R)  .GT. 16383.0 .OR. ABS(T1) .GT. 32767.0 .OR.
     f    ABS(T2) .GT. 32767.0 .OR. ABS(DT) .LT. 1.0     .OR.
     f    ABS(DT) .GT. 179.9999                              ) THEN

        WRITE(*,*) 'Parameter out of range in RACIR subroutine.'
        STOP
      ENDIF

      OPEN(9,FILE='PRN')
      IF (DT.GT.0.0) THEN
        WRITE(9,100) 'G',R,K,T1,K,T2,K,DT
      ELSE
        WRITE(9,100) 'G',R,K,T1,K,T2
      ENDIF
100   FORMAT(4(A1,F11.4))
      CLOSE(9)

      RETURN
      END

      SUBROUTINE RAPROC(N,L1,L2)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        N,L1,L2
      CHARACTER   K*1

      K=','

      IF (ABS(N)  .GT. 9101.0 .OR. ABS(L1) .GT. 16383.0 .OR.
     f    ABS(L2) .GT. 16383.0                               ) THEN

        WRITE(*,*) 'Parameter out of range in RAPROC subroutine.'
        STOP
      ENDIF

      OPEN(9,FILE='PRN')
      WRITE(9,100) 'K',N,K,L1,K,L2
100   FORMAT(A1,F11.4,A1,F11.4,A1,F11.4)
      CLOSE(9)

      RETURN
      END

      SUBROUTINE RHATCH(M,X,Y,D,T)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     M,T,IM,IT
      REAL        X,Y,D

      IF (ABS(X).GT.16383.0.OR.ABS(Y).GT.16383.0.OR.
     f    ABS(D).GT.16383.0                          ) THEN

        WRITE(*,*) 'Parameter out of range in RHATCH subroutine.'
        STOP
      ENDIF

      IF (M.LT.0.OR.M.GT.3) THEN
        IM=2
      ELSE
        IM=M
      ENDIF

      IF (T.LT.1.OR.T.GT.4) THEN
        IT=2
      ELSE
```

```
        IT=T
      ENDIF

      OPEN(9,FILE='PRN')
      WRITE(9,100) 'T',IM,',',X,',',Y,',',D,',',T
100   FORMAT(A1,I1,A1,F11.4,A1,F11.4,A1,F11.4,A1,I1)
      CLOSE(9)

      RETURN
      END

      SUBROUTINE RCURVE(M,X,Y,N)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     M,IM,N,I
      REAL        X(1:*),Y(1:*)

      IF (M.LT.0.OR.M.GT.3) THEN
        IM=0
      ELSE
        IM=M
      ENDIF

      IF ((M.LE.1.AND.N.LE.2).OR.(M.GE.2.AND.N.LE.1)) THEN
        WRITE(*,*) 'Parameter out of range in RCURVE subroutine.'
        STOP
      ENDIF

      DO 10 I=1,N
        IF (ABS(X(I)).GT.16383.0.OR.ABS(Y(I)).GT.16383.0) THEN
          WRITE(*,*) 'Parameter out of range in RCURVE subroutine.'
          STOP
        ENDIF
10    CONTINUE

      OPEN(9,FILE='PRN')
      WRITE(9,100) 'Y',IM,',',(' ',X(I),' ',Y(I), I=1,N)
100   FORMAT(A1,I1,A1,10000(A1,F11.4,A1,F11.4))
      CLOSE(9)

      RETURN
      END

      SUBROUTINE RRCURV(M,X,Y,N)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      INTEGER     M,IM,N,I
      REAL        X(1:*),Y(1:*)

      IF (M.LT.0.OR.M.GT.1) THEN
        IM=0
      ELSE
        IM=M
      ENDIF

      IF (N.LE.2) THEN
        WRITE(*,*) 'Parameter out of range in RRCURVE subroutine.'
        STOP
      ENDIF

      DO 10 I=1,N
        IF (ABS(X(I)).GT.16383.0.OR.ABS(Y(I)).GT.16383.0) THEN
          WRITE(*,*) 'Parameter out of range in RRCURVE subroutine.'
          STOP
        ENDIF
10    CONTINUE

      OPEN(9,FILE='PRN')
      WRITE(9,100) ' ',IM,',',(' ',X(I),' ',Y(I), I=1,N)
100   FORMAT(A1,I1,A1,10000(A1,F11.4,A1,F11.4))
      CLOSE(9)

      RETURN
      END

      SUBROUTINE RRDGLC(C,LEN)
C     Anders Persson, 1988

      IMPLICIT      LOGICAL (A-Z)
      INTEGER       LEN,I
      CHARACTER*(*) C

      OPEN(9,FILE='PRN')
      WRITE(9,100) '^',(C(I:I), I=1,LEN)
100   FORMAT(10000(A1))
      CLOSE(9)

      RETURN
      END
```

HPLIB, HPFLIB

Biblioteken HPLIB och HPFLIB innehåller diverse rutiner för att använda en HP-LaserJet Series II skrivare som en pen-plotter eller grafikskärm. Den använda upplösningen är 100 punkter per tum vilket för A4-bilder innebär en rasterstorlek på (0..1119)*(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, HIGHBYTE Dessa rutiner tar fram de 8 minst resp. mest signifikanta bitarna i D. Används av HPDUMP rutinen för att särа 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).

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: HPLIB.FOR   Options: /LBZ          08/02/88 14:27:25

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                               C
C     Library: HPLIB.LIB        C
C                               C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      SUBROUTINE HPSTART(ORIENTATION)
C     Anders Persson 1988

      IMPLICIT    LOGICAL (A-Z)
      CHARACTER   ORIENTATION*1
      INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
      LOGICAL     LANDSCAPE
      INTEGER*2   INITDATA(1:52),I
      COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

      DATA        INITDATA/
     f                  27,38,108,48,79,
     f                  27,40, 48,83,
     f                  27,40,115,48,80,
     f                  27,40,115,49,54,46,54,54,72,
     f                  27,40,115,55,46,53,86,
     f                  27,40,115,48,83,
     f                  27,40,115,48,66,
     f                  27,40,115,48,84,
     f                  27,42,116,49,48,48,82/

C     Data corresponds to:
C     Orientation                 = Portrait or landscape
C     Symbol set                  = Swedish,
C     Spacing                     = Fixed,
C     Pitch                       = 16.66 CPI,
C     Point size                  = 8.5,
C     Style                       = Upright,
C     Stroke weight               = Medium,
C     Typeface                    = Line printer,
C     Raster Graphics Resolution  = 100 DPI.

      COLOUR=1

      LANDSCAPE=ORIENTATION.EQ.'L'.OR.ORIENTATION.EQ.'l'
      IF (LANDSCAPE) THEN
        INITDATA(4)=49
      ELSE
        INITDATA(4)=48
      ENDIF

      CALL HPINIT
      DO 10 I=1,52
         CALL HPOUT(INITDATA(I))
10    CONTINUE

      CALL HPCLEAR
      CALL HPMOVE(0.0,0.0)
      RETURN
      END

C                     TITLE           HPINIT
C     ; Anders Persson, 1988
C     ; FORTRAN     SUBROUTINE HPINIT
C     ;
C     LADATA        SEGMENT         'DATA'
C                   DB              'HPINIT  '
C     SP_SAVE       DW              0
C                   DD              0
C     LADATA        ENDS
C     PCODE         SEGMENT         'CODE'
C                   ASSUME          CS:PCODE,DS:LADATA
C                   DW              SEG  LADATA
C     HPINIT        PROC            FAR
C                   PUBLIC          HPINIT
C                   MOV             AX,LADATA
C                   MOV             DS,AX
C                   MOV             SP_SAVE,SP
C     CTRLPORT      =               03BEH
C                   MOV             DX,CTRLPORT     ;INITIALIZE
C                   MOV             AL,11101100B    ;PRINTER
C                   OUT             DX,AL           ;PORT
C                   RET
C     HPINIT        ENDP
C     PCODE         ENDS
C                   END
C
C                   TITLE           HPOUT
C     ; Anders Persson, 1988
C     ; FORTRAN:   SUBROUTINE HPOUT(D)
C     ;            INTEGER*2 D
C     ;
C     PARBLK        STRUC
C                   PAR             DD              ?
C     PARBLK        ENDS
C     LADATA        SEGMENT         'DATA'
C                   DB              'HPOUT   '
C     SP_SAVE       DW              0
C                   DD              0
C     LADATA        ENDS
C     PCODE         SEGMENT         'CODE'
C                   ASSUME          CS:PCODE,DS:LADATA
C                   DW              SEG  LADATA
C     HPOUT         PROC            FAR
C                   PUBLIC          HPOUT
C                   MOV             AX,LADATA
C                   MOV             DS,AX
C                   MOV             SP_SAVE,SP
C     ;
C     STATPORT      =               03BDH
C     DATAPORT      =               03BCH
C     CTRLPORT      =               03BEH
C     ;
C                   MOV             DX,STATPORT             ; CHECK
C     BUSY:         IN              AL,DX                   ; PRINTER
C                   AND             AL,10000000B            ; READY
C                   JZ              BUSY                    ; = BUSY=0
C     ;
C                   MOV             DX,DATAPORT             ; PUT
C                   LDS             SI,ES:PAR[BX]           ; DATA
C                   MOV             AX,[SI]                 ; ON OUTPUT
```

```
C                       OUT             DX,AL                   ; LINES
C       ;
C                       MOV             DX,CTRLPORT             ; SEND
C                       MOV             AL,11101101B            ;
C                       OUT             DX,AL                   ; STROBE
C                       MOV             AL,11101100B            ;
C                       OUT             DX,AL                   ; PULSE
C                       RET
C       HPOUT           ENDP
C       PCODE           ENDS
C                       END
C
C                       TITLE           HPRLINE
C       ; Anders Persson, 1988
C       ; FORTRAN:   SUBROUTINE HPRLINE(LEN,DMPLINE)
C       ;            INTEGER*2 LEN,DMPLINE(1:LEN)
C       ;
C       PARBLK          STRUC
C                       LEN             DD              ?
C                       ARRAY           DD              ?
C       PARBLK          ENDS
C       LADATA          SEGMENT         'DATA'
C                       DB              'HPRLINE '
C       SP_SAVE         DW              0
C                       DD              0
C       LADATA          ENDS
C       PCODE           SEGMENT         'CODE'
C                       ASSUME          CS:PCODE,DS:LADATA
C                       DW              SEG  LADATA
C       HPRLINE         PROC            FAR
C                       PUBLIC          HPRLINE
C                       MOV             AX,LADATA
C                       MOV             DS,AX
C                       MOV             SP_SAVE,SP
C       ;
C       CTRLPORT        =               03BEH
C       DATAPORT        =               03BCH
C       STATPORT        =               03BDH
C       ;
C                       LDS             SI,ES:LEN[BX]
C                       MOV             CX,[SI]
C                       MOV             AX,CX
C                       ADD             CX,AX
C                       LDS             SI,ES:ARRAY[BX]
C                       MOV             BX,0
C       ;
C       AGAIN:          MOV             DX,STATPORT     ; CHECK
C       BUSY1:          IN              AL,DX           ; IF PRINTER
C                       AND             AL,10000000B    ; IS READY
C                       JZ              BUSY1           ; FOR DATA
C       ;
C                       MOV             DX,DATAPORT     ; SEND
C                       MOV             AX,[SI][BX]     ; FIRST
C                       MOV             AL,AH           ; DATA
C                       OUT             DX,AL           ; BYTE
C       ;
C                       MOV             DX,CTRLPORT     ; SEND
C                       MOV             AL,11101101B    ;
C                       OUT             DX,AL           ; STROBE
C                       MOV             AL,11101100B    ;
C                       OUT             DX,AL           ; PULSE
C       ;
C                       MOV             DX,STATPORT     ; CHECK
C       BUSY2:          IN              AL,DX           ; IF PRINTER
C                       AND             AL,10000000B    ; IS READY
C                       JZ              BUSY2           ; FOR DATA
C       ;
C                       MOV             AX,[SI][BX]     ; SEND
C                       MOV             DX,DATAPORT     ; SECOND
C                       OUT             DX,AL           ; BYTE
C       ;
C                       MOV             DX,CTRLPORT     ; SEND
C                       MOV             AL,11101101B    ;
C                       OUT             DX,AL           ; STROBE
C                       MOV             AL,11101100B    ;
C                       OUT             DX,AL           ; PULSE
C       ;
C                       ADD             BX,2            ; IF MORE
C                       CMP             BX,CX           ; DATA THEN
C                       JNZ             AGAIN           ; GOTO AGAIN
C       ;
C                       RET
C       HPRLINE         ENDP
C       PCODE           ENDS
C                       END

        SUBROUTINE HPDUMP
C       Anders Persson, 1988

        IMPLICIT    LOGICAL (A-Z)
        INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119),I,J,LEN
        INTEGER*2   DMPLINE(1:48),ESC
        LOGICAL     LANDSCAPE
        COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

        ESC=27
        LEN=48
        DMPLINE(1)=256*27+42
        DMPLINE(2)=256*98+57
        DMPLINE(3)=256*48+87
C       DMPLINE(1..3)=Esc * b 9 0 W = Raster line header

C       Set cursor position
        IF (LANDSCAPE) THEN
          CALL HPCMOVE(1119.0,719.0)
        ELSE
          CALL HPCMOVE(0.0,1119.0)
        ENDIF

C       Initialize raster graphics transfer
        CALL HPOUT(ESC)
        CALL HPASCII(4,'*r1A')

        DO 20 I=0,1119
C         Transfer raster graphics line
          DO 10 J=0,44
            DMPLINE(J+4)=RASTER(J,I)
10        CONTINUE
          CALL HPRLINE(LEN,DMPLINE)
20      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 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
260 C                                                               C
261 C     Remaining routines common to HPLIB and HPFLIB             C
262 C                                                               C
263 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
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:I))
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:I))
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
338       SUBROUTINE HPMOVE(X,Y)
339 C     Anders Persson 1988
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
```

```
      CALL   HPMOVE(X,Y)
      XEND   = XPOS
      YEND   = YPOS

      RX     = XSTART
      RY     = YSTART
      A      = XEND-XSTART
      B      = YEND-YSTART

      IF (A.LT.0) THEN
        A     =-A
        DXDIAG=-1
      ELSE
        DXDIAG=1
      ENDIF

      IF (B.LT.0) THEN
        B     =-B
        DYDIAG=-1
      ELSE
        DYDIAG=1
      ENDIF

      IF (A.LT.B) THEN
        SWAP      = A
        A         = B
        B         = SWAP
        DXNONDIAG = 0
        DYNONDIAG = DYDIAG
      ELSE
        DXNONDIAG = DXDIAG
        DYNONDIAG = 0
      ENDIF

      D          = B+B-A
      NONDIAGINC = B+B
      DIAGINC    = B+B-A-A

      DO 10 I=0,A
        IF (LANDSCAPE) THEN
          WORD  = (719-RY)/16
          BIT   = 15-MOD(719-RY,16)
          LINE  = 1119-RX
        ELSE
          WORD  = RX/16
          BIT   = 15-MOD(RX,16)
          LINE  = 1119-RY
        ENDIF

        IF (COLOUR.EQ.0) THEN
          RASTER(WORD,LINE)=IBCLR(RASTER(WORD,LINE),BIT)
        ELSEIF (COLOUR.EQ.2) THEN
          RASTER(WORD,LINE)=IBCHNG(RASTER(WORD,LINE),BIT)
        ELSE
          RASTER(WORD,LINE)=IBSET(RASTER(WORD,LINE),BIT)
        ENDIF

        IF (D.LT.0) THEN
          RX    = RX+DXNONDIAG
          RY    = RY+DYNONDIAG
          D     = D+NONDIAGINC
        ELSE
          RX    = RX+DXDIAG
          RY    = RY+DYDIAG
          D     = D+DIAGINC
        ENDIF
   10 CONTINUE
      RETURN
      END

      SUBROUTINE HPPLOT(X,Y)
C     Anders Persson 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y
      INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
      LOGICAL     LANDSCAPE
      INTEGER*2   WORD,LINE,BIT
      COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

      CALL HPMOVE(X,Y)
      IF (LANDSCAPE) THEN
        WORD  = (719-YPOS)/16
        BIT   = 15-MOD(719-YPOS,16)
        LINE  = 1119-XPOS
      ELSE
        WORD  = XPOS/16
        BIT   = 15-MOD(XPOS,16)
        LINE  = 1119-YPOS
      ENDIF

      IF (COLOUR.EQ.0) THEN
        RASTER(WORD,LINE)=IBCLR(RASTER(WORD,LINE),BIT)
      ELSEIF (COLOUR.EQ.2) THEN
        RASTER(WORD,LINE)=IBCHNG(RASTER(WORD,LINE),BIT)
      ELSE
        RASTER(WORD,LINE)=IBSET(RASTER(WORD,LINE),BIT)
      ENDIF
      RETURN
      END

      LOGICAL FUNCTION HPSET(X,Y)
C     Anders Persson 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y
      INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
      LOGICAL     LANDSCAPE
      INTEGER*2   WORD,LINE,BIT
      COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

      CALL HPMOVE(X,Y)
      IF (LANDSCAPE) THEN
        WORD   = (719-YPOS)/16
        BIT    = 15-MOD(719-YPOS,16)
        LINE   = 1119-XPOS
      ELSE
        WORD   = XPOS/16
        BIT    = 15-MOD(XPOS,16)
        LINE   = 1119-YPOS
```

```
      ENDIF

      HPSET=BTEST(RASTER(WORD,LINE),BIT)
      RETURN
      END

      SUBROUTINE HPRMOVE(X,Y)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y,IX,IY
      INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
      LOGICAL     LANDSCAPE
      COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

      IX=XPOS+X
      IY=YPOS+Y
      CALL HPMOVE(IX,IY)
      RETURN
      END

      SUBROUTINE HPRPLOT(X,Y)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y,IX,IY
      INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
      LOGICAL     LANDSCAPE
      COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

      IX=XPOS+X
      IY=YPOS+Y
      CALL HPPLOT(IX,IY)
      RETURN
      END

      SUBROUTINE HPRDRAW(X,Y)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y,IX,IY
      INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
      LOGICAL     LANDSCAPE
      COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

      IX=XPOS+X
      IY=YPOS+Y
      CALL HPDRAW(IX,IY)
      RETURN
      END

      SUBROUTINE HPRPRINT(X,Y,LEN,MSG)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y,IX,IY
      INTEGER     LEN
      CHARACTER   MSG*(*)
      INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
      LOGICAL     LANDSCAPE
      COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

      IX=XPOS+X
      IY=YPOS+Y
      CALL HPPRINT(IX,IY,LEN,MSG)
      RETURN
      END

      SUBROUTINE HPMARK(X,Y)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y,IX,IY
      INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
      LOGICAL     LANDSCAPE
      COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

      IX=X
      IY=Y
      CALL HPPLOT(IX,IY)
      CALL HPPLOT(IX+1.0,IY)
      CALL HPPLOT(IX-1.0,IY)
      CALL HPPLOT(IX,IY+1.0)
      CALL HPPLOT(IX,IY-1.0)
      CALL HPMOVE(IX,IY)
      RETURN
      END

      SUBROUTINE HPBOX(X,Y,XL,YL)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y,XL,YL,IX
      INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119),SAVECOLOUR
      LOGICAL     LANDSCAPE
      COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

      SAVECOLOUR=COLOUR
      CALL HPLEVEL(0)
      DO 10 IX=X,X+XL,1.0
        CALL HPMOVE(IX,Y)
        CALL HPDRAW(IX,Y+YL)
10    CONTINUE

      COLOUR=SAVECOLOUR
      CALL HPMOVE(X,Y)
      CALL HPDRAW(X+XL,Y)
      CALL HPDRAW(X+XL,Y+YL)
      CALL HPDRAW(X,Y+YL)
      CALL HPDRAW(X,Y)
      RETURN
      END

      SUBROUTINE HPTBOX(X,Y,LEN,MSG)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        X,Y,XL,YL
      INTEGER     LEN
      CHARACTER   MSG*(*)
```

```
      INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119)
      LOGICAL     LANDSCAPE
      COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

      YL=13.0
      XL=(LEN+2)*6.0
      CALL HPBOX(X,Y,XL,YL)
      CALL HPRPRINT(6.0,3.0,LEN,MSG)
      RETURN
      END

      SUBROUTINE HPCIRCL(X,Y,R)
C     Anders Persson, 1988

      IMPLICIT    LOGICAL (A-Z)
      REAL        TWOPI
      PARAMETER   (TWOPI=6.283185308)
      REAL        X,Y,R,IX,IY,ARG
      INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119),I
      LOGICAL     LANDSCAPE
      COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

      CALL HPMOVE(X+R,Y)
      DO 10 I=1,100
        ARG  = TWOPI*0.01*I
        IX   = X+R*COS(ARG)
        IY   = Y+R*SIN(ARG)
        CALL HPDRAW(IX,IY)
10    CONTINUE
      RETURN
      END
```

```
IBM Professional FORTRAN Compiler (V1.30) by Ryan-McFarland Corp
Source File: HPFLIB.FOR  Options: /LBZ         08/02/88 14:26:58

   1 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
   2 C                                C
   3 C     Library: HPFLIB.LIB        C
   4 C                                C
   5 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
   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))
```

```
          CALL HPOUT(D)
          D=LOWBYTE(RASTER(J,I))
          CALL HPOUT(D)
   20   CONTINUE
   30 CONTINUE

C    End raster graphics
     CALL HPOUT(ESC)
     CALL HPASCII(3,'*rB')
     RETURN
     END

     SUBROUTINE HPEJECT
C    Anders Persson, 1988

     IMPLICIT    LOGICAL (A-Z)
     INTEGER*2   XPOS,YPOS,COLOUR,RASTER(0:44,0:1119),FF
     LOGICAL     LANDSCAPE
     COMMON      /RGRP/ XPOS,YPOS,LANDSCAPE,COLOUR,RASTER

     CALL HPDUMP
     FF=12
     CALL HPOUT(FF)
     CLOSE(21)
     RETURN
     END

C                    TITLE           LOWBYTE
C    ; Anders Persson, 1988
C    ; FORTRAN:    INTEGER*2 FUNCTION LOWBYTE(D)
C    ;             INTEGER*2 D
C    ;
C    PARBLK         STRUC
C                   PAR             DD              ?
C    PARBLK         ENDS
C    LADATA         SEGMENT         'DATA'
C                   DB              'LOWBYTE '
C    SP_SAVE        DW              0
C                   DD              0
C    LADATA         ENDS
C    PCODE          SEGMENT         'CODE'
C                   ASSUME          CS:PCODE,DS:LADATA
C                   DW              SEG  LADATA
C    LOWBYTE        PROC            FAR
C                   PUBLIC          LOWBYTE
C                   MOV             AX,LADATA
C                   MOV             DS,AX
C                   MOV             SP_SAVE,SP
C    ;
C                   LDS             SI,ES:PAR[BX]
C                   MOV             AX,[SI]
C                   MOV             AH,0
C    ;
C                   RET
C    LOWBYTE        ENDP
C    PCODE          ENDS
C                   END
C
C                   TITLE           HIGHBYTE
C    ; Anders Persson, 1988
C    ; FORTRAN:    INTEGER*2 FUNCTION HIGHBYTE(D)
C    ;             INTEGER*2 D
C    ;
C    PARBLK         STRUC
C                   PAR             DD              ?
C    PARBLK         ENDS
C    LADATA         SEGMENT         'DATA'
C                   DB              'HIGHBYTE'
C    SP_SAVE        DW              0
C                   DD              0
C    LADATA         ENDS
C    PCODE          SEGMENT         'CODE'
C                   ASSUME          CS:PCODE,DS:LADATA
C                   DW              SEG  LADATA
C    HIGHBYTE       PROC            FAR
C                   PUBLIC          HIGHBYTE
C                   MOV             AX,LADATA
C                   MOV             DS,AX
C                   MOV             SP_SAVE,SP
C    ;
C                   LDS             SI,ES:PAR[BX]
C                   MOV             AX,[SI]
C                   MOV             AL,AH
C                   MOV             AH,0
C    ;
C                   RET
C    HIGHBYTE       ENDP
C    PCODE          ENDS
C                   END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                      C
C    More routines in HPLIB.FOR        C
C                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

Länkningslista:

```
LINK POL1+SCRPLOT+ROLPLOT+JETPLOT+RANF+NRMAPROX,,,HERCLIB+PLLIB+HPLIB+MATLIB
LINK POL2+DSCRPLOT+DROLPLOT+DJETPLOT+RANF+NRMAPROX,,,HERCLIB+PLLIB+HPLIB+DMATLIB
LINK POL3+DSCRPLOT+DROLPLOT+DJETPLOT+RANF+NRMAPROX,,,HERCLIB+PLLIB+HPLIB+DMATLIB
LINK CALCPOLC,,,,
LINK ARCROSS+DSCRPLOT+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: