Comparative Studies Of Electrical Properties And Temperature Behaviour Of Inductive And Capacitive Sparks For Ignition Systems

Diploma paper
by

Tomas L Starczewski

LRAP-73 (1987)

Department of Physics, Lund Institute of Technology
P.O. Box 725, S-220 07 LUND
together with:
Mecel AB of SAAB-Scania Combitech
Box 32, S-662 00 ÅMÅL

CONTENTS

INTRODUCTION

The performance levels required of petrol engines have been rising with each year. Recently, attention has been focused on the factors improving engine driveability, in addition to exhaust emission purification and better fuel consumption economy.

In engine combustion, the first important stage of the combustion process is ignition. In many investigations attempts have been made to improve combustion by enhancing ignition performance. One particular way of obtaining this goal has been the design of new ignition systems.

Conventional ignition systems are based on the inductive priciple. Recently however designers have turned their attention towards another principle employing capacitive discharge circuits.

Capacitive ignition systems are said to have a considerable number of advantages. A few of them are:

- due to the inherent properties of capacitive systems the scattering in the time delay between the voltage pulse and the spark is strongly reduced,
- due to the shorter spark duration time lifetimes of sparking plugs are much longer,
- much better lean mixture operating conditions,
- in some capacitive systems [1] the flame kernel expands more rapidly than in conventional (inductive) systems.

The basic purpose of this diploma work was to examine the electrical properties and the temperature behaviour of inductive and capacitive sparks.

The systems investigated in this diploma work were a commercially available inductive system and a capacitive system. Comparison with yet another capacitive system - one generating ultra-short high-current sparks - was also made. Throughout this paper I will be referring to these three as the inductive, the ordinary capacitive and the ultra-fast capacitive system respectively.

The electrical properties measured were the voltage and the current of the sparks. The power and the dissipated energy were computed numerically.

The temperature behaviour of the sparks was determined by means of one-wavelength optical interferometry.

ELECTRICAL MEASUREMENTS

2.1. INTRODUCTION

All the electrical measurements were carried out under exactly the same conditions. The parameters held constant were:

- the air pressure ≅ 760 mm Hg (atmospheric)
- the air temperature = 22° C
- the power supply voltage ≅ 12 V DC
  A commercial car-battery was used

- the outer electrical properties such as cable capacitances and inductances

Two different sparking plugs were used:

1. commercial sparking plug with flat electrodes,
2. specially designed sparking plug with sharply edged electrodes of stainless steel.

In both cases the constant parameter was:

- the electrode separation = 0.90 ± 0.05 mm

The ultra-fast capacitive system used electrodes whose shapes were half spheres with a radius of 1.5 mm, and they were separated by approximately 2 mm. All electrode arrangements are shown in Fig. 2:1.

Fig. 2:1. The electrode arrangements used in the experiments.

2.2 EXPERIMENTAL DETAILS

The properties directly measured were the current and the voltage of the spark. In order to obtain the time spectra of the these parameters a fast oscilloscope with a Polaroid camera were used for the registration. The spark frequency was set to be approximately 1 Hz.

In the inductive system the symmetric square-wave trig pulse was sent to the ignition circuit generating an electrical pulse to the ignition coil, generating in turn the high voltage needed for the break-down. The diode was placed only for the protection of the transistor and had no effect on the pulse shape. The coaxial 50 Ω coaxial cable was used for the high-voltage connections.

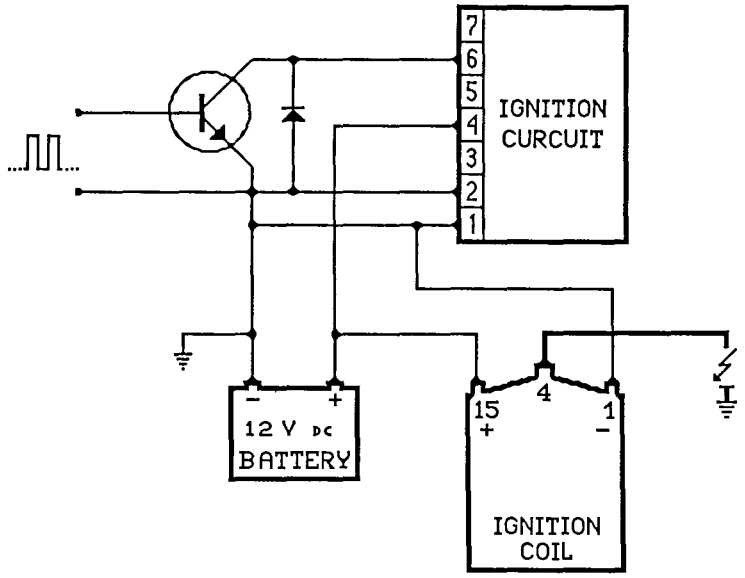


Fig. 2:2. Electrical arrangement for the inductive system.

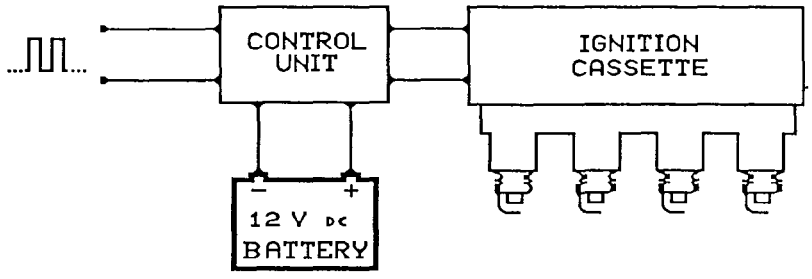


Fig. 2:3. Electrical arrangement for the ordinary capacitive system.

Fig. 2:4. Electrical arrangement for the ultra-fast capacitive system.

The current and the voltage measurements were conducted on different sparks. Reproductivity of the current and the voltage shapes were good for the systems with conventional sparking plugs as seen from the oscilloscope screen.

The spark current was measured by a resistor made of a special alloy. Special care was taken to avoid contact resistances in resistance measuring. The resistance had to be kept low for the inductive system in order not to change the system's behaviour.

The spark voltage was measured with a conventional high-voltage probe. The probe was carefully calibrated and the attenuance ratio was 1095 times.

The temporal behaviour of the power was obtained by manual multiplication of the current and the voltage pulses.

The dissipated energy in the sparks was calculated using $\int p \cdot dt = \int (u \cdot i)\, dt$.

2.3 RESULTS AND DISCUSSION

Examples of the shapes of the current and the voltage pulses together with the power and the energy of the different sparks are shown in figures 2:5 - 2:7. Comparative diagrams of the spark energies are shown in figures 2:8 and 2:9. Exact plots of the power can be found in appendices A:1 - A:4.

There is an inevitable variation in the obtained data due to the fact that the current and the voltage measurements were performed at different times and consequently on different sparks.

The different natures of the measured systems show clearly in their different electrical properties.

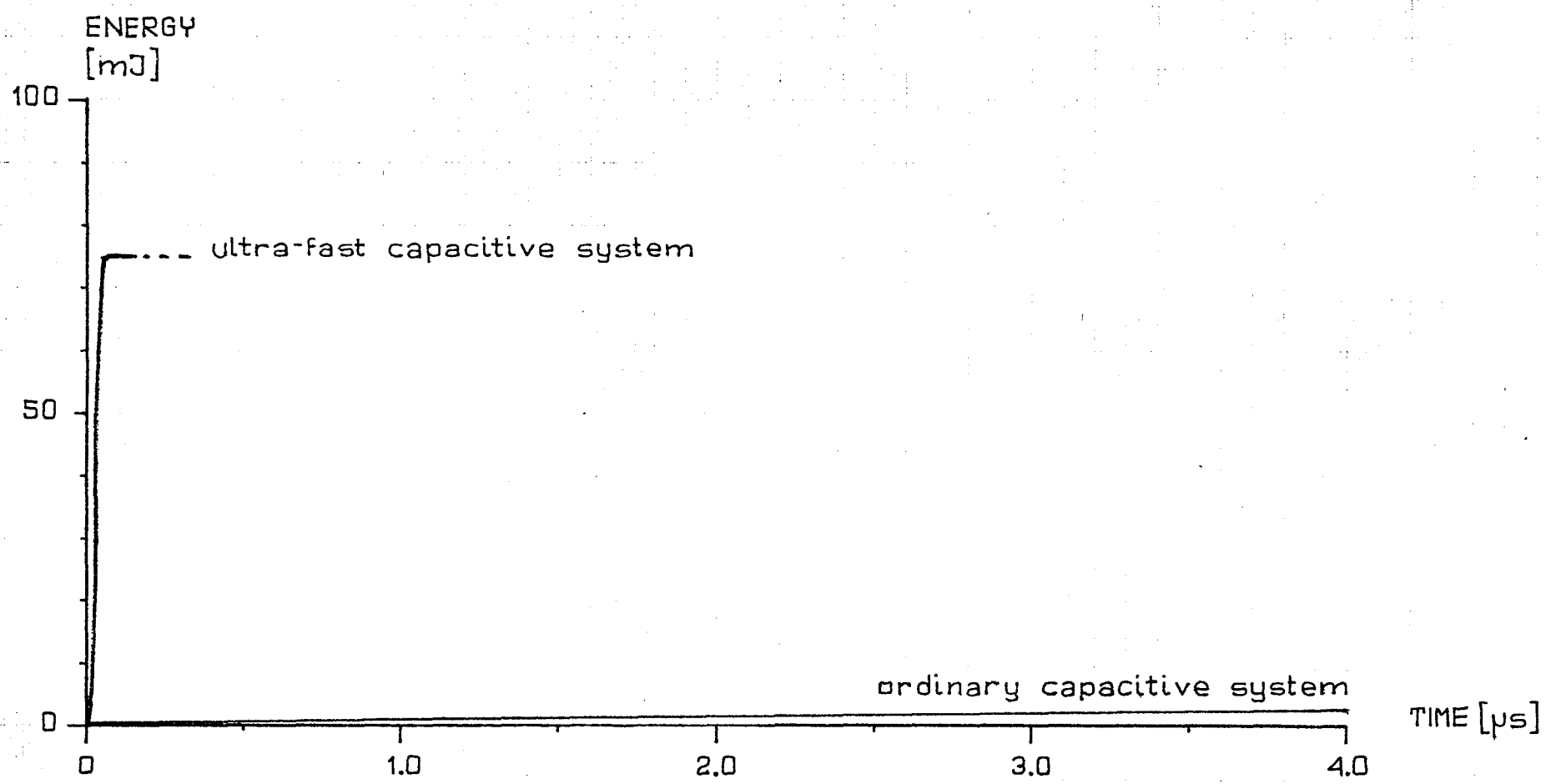
The inductive system's power quickly rises (in about 0.5 µs) to approximately 800 W only to decrease rapidly. The time domain of that decrease is 3 ms. The energy is therefore dissipated mainly during the first 2 ms. The total energy is approximately 8 mJ). The behaviour of the inductive system can be seen in figure 2:5.

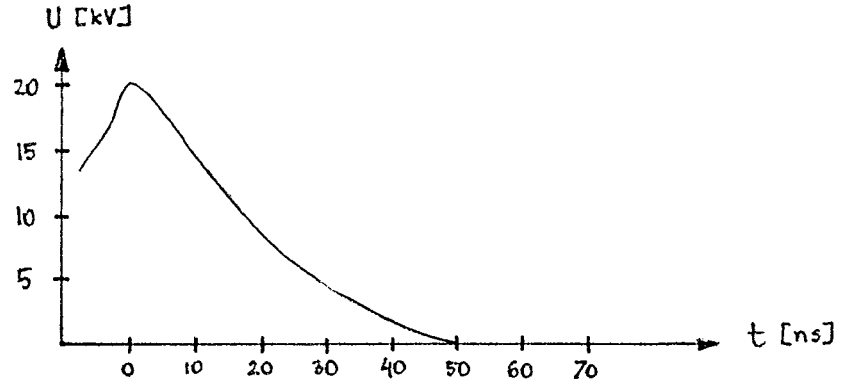
The quicker commercial capacitive system's current and voltage are of oscillatory nature. The power is therefore also a series of short energy-rich pulses. The first two pulses are highest measuring 990 W and 920 W and occuring at 5 µs and 20 µs respectively. The remaining pulses are smaller and contribute less to the total energy dissipation of approximately 35 mJ. The main portion of energy is pumped during the first 100 µs. The behaviour of the commercial capacitive system can be seen in figure 2:6.

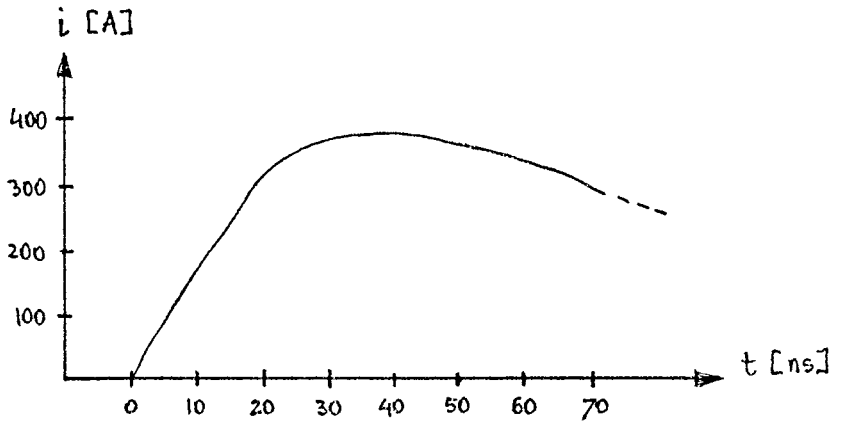
The ultra-fast capacitive system's power is a single pulse with a maximum of 3 MW at 15 ns. Almost all energy (75 mJ) is transferred in 40 ns. The behaviour of the ultra-fast capacitive system can be seen in figure 2:7.

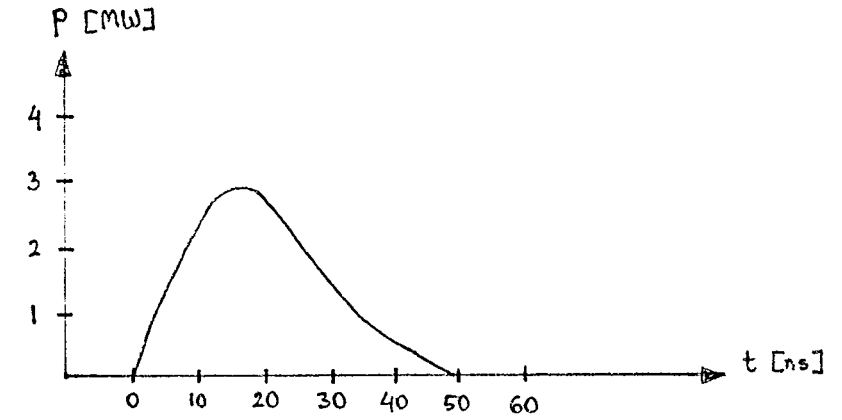
The immense time difference in the energy behaviour of the systems is shown in figures 2:8 and 2:9. In figure 2:8 we can see the difference in rapidness between the inductive and commercial capacitive systems. The difference is even greater when the commercial and the ultra-fast capacitive systems are compared in figure 2:9.

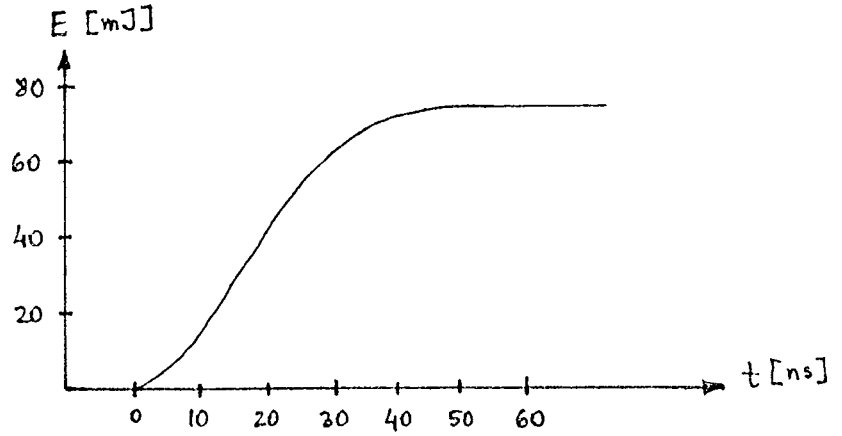
The slight differences between the standard plug and the sharp-edge electrode plug measurements can only be explained by somewhat different electrical properties of the two arrangemnets. The measuring of their inductances and capacitances has however not been able to prove that such differences exist.

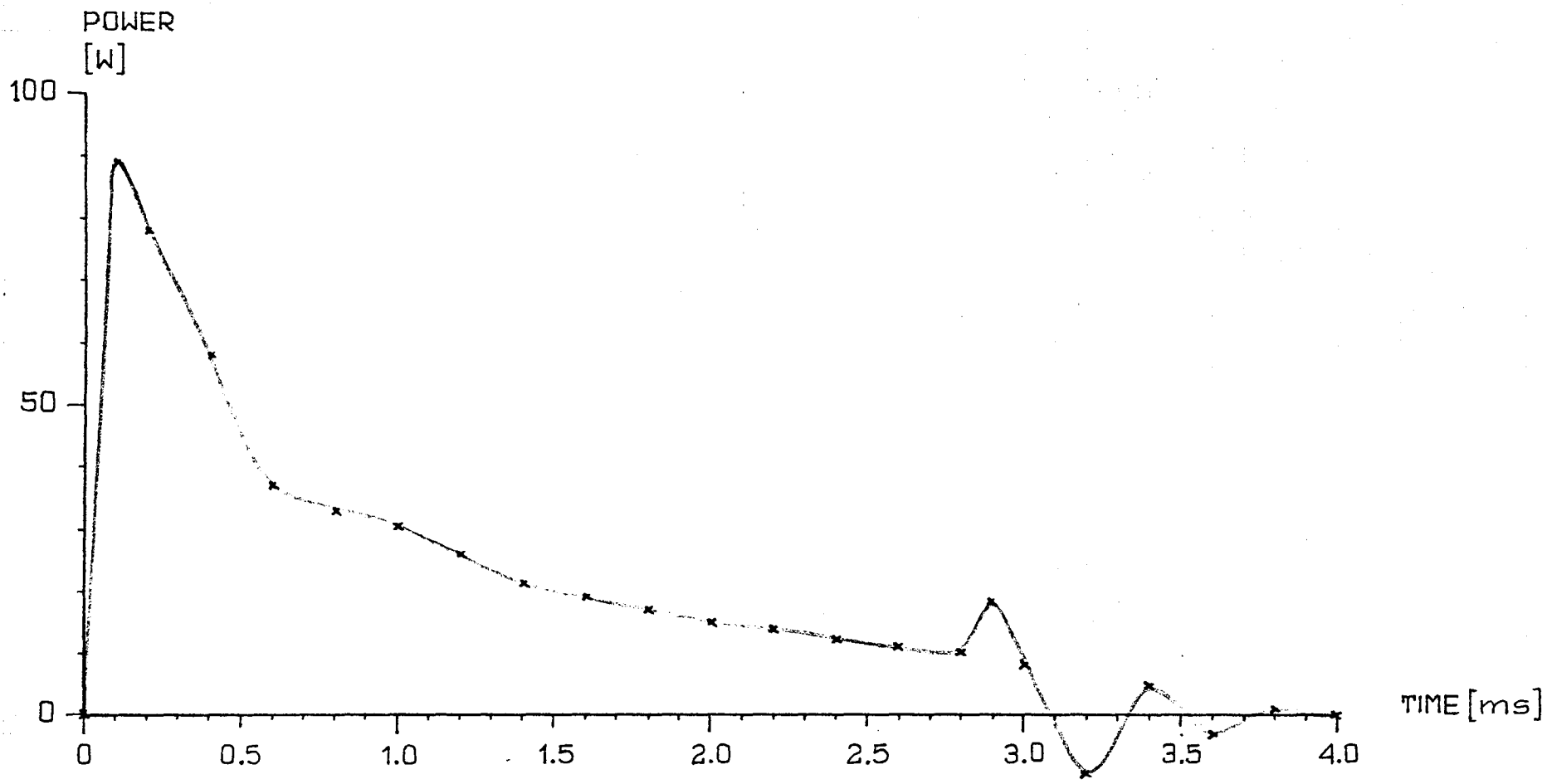

ENERGY
[mJ]
100
50
0
ultra-fast capacitive system
ordinary capacitive system
TIME [μs]
0
1.0
2.0
3.0
4.0

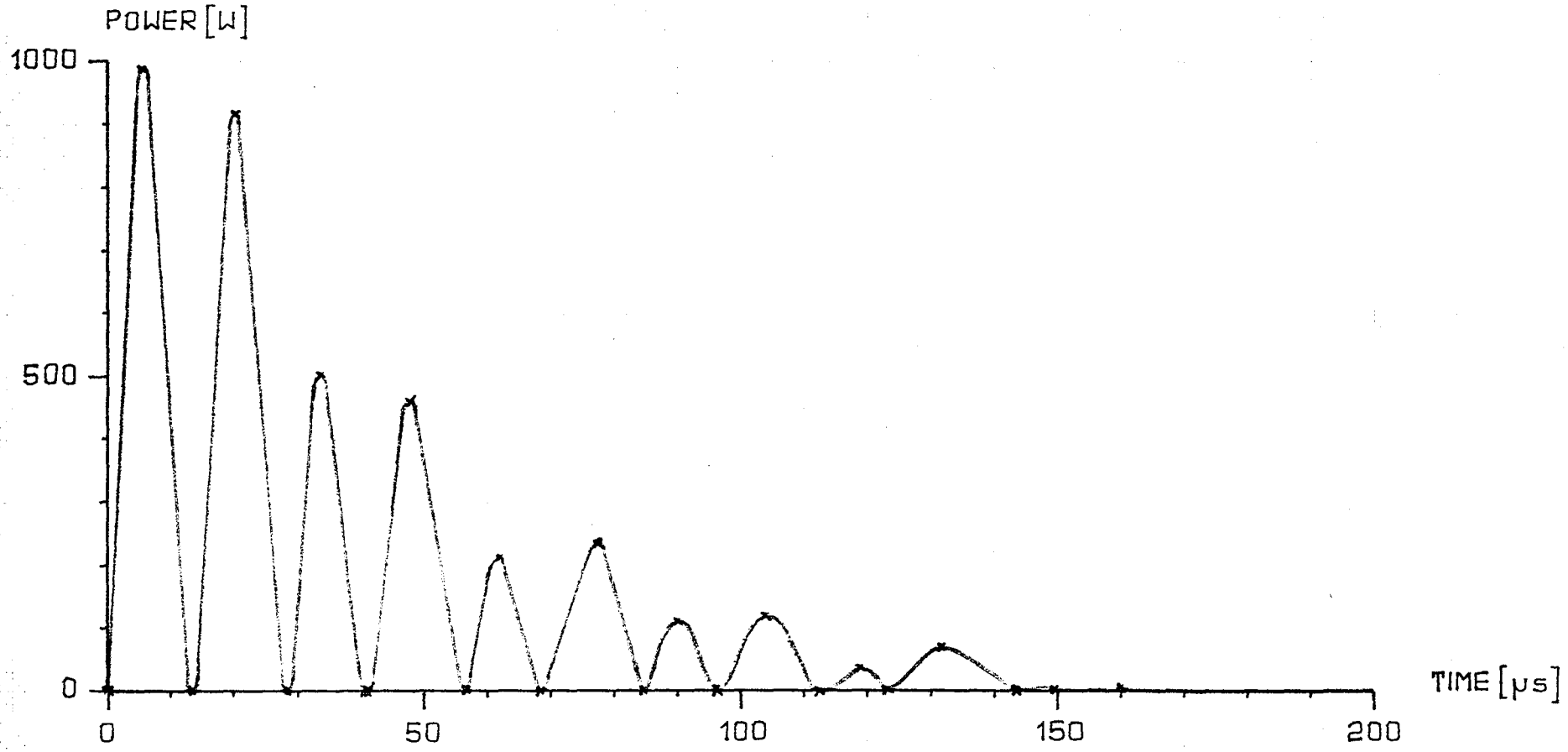

U [kV]
20
15
10
5
0
10
20
30
40
50
60
70
t [ns]

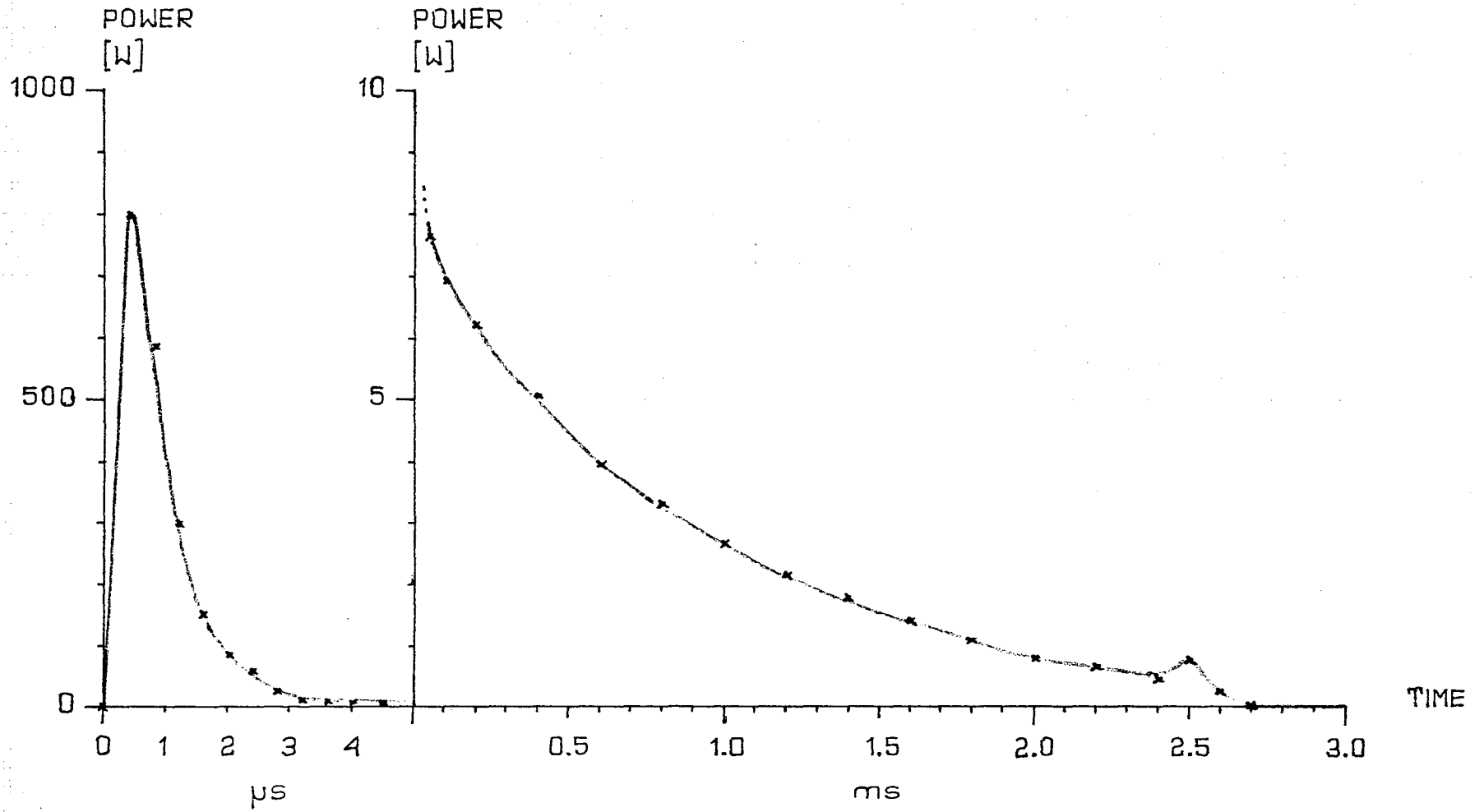

i [A]
400
300
200
100
0
10
20
30
40
50
60
70
t [ns]

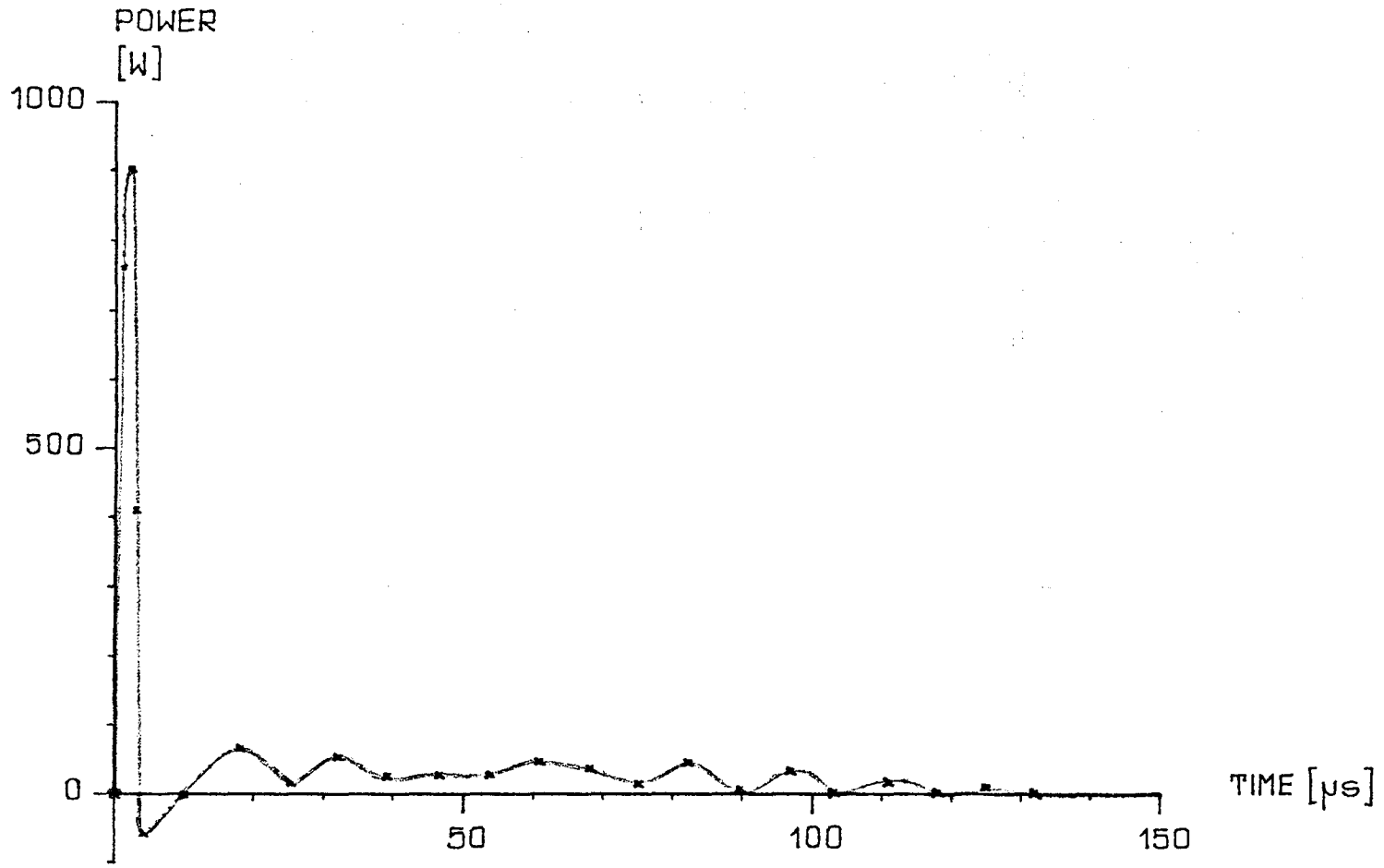

P [MW]
4
3
2
1
0
10
20
30
40
50
60
t [ns]

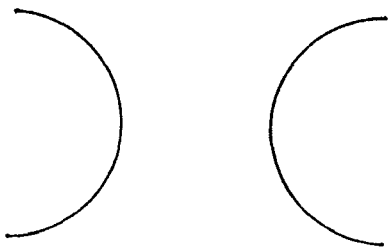

E [mJ]
80
60
40
20
0
10
20
30
40
50
60
t [ns]

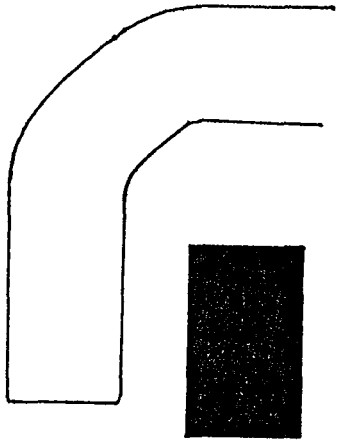

POWER
[W]
100
50
0
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
TIME [ms]

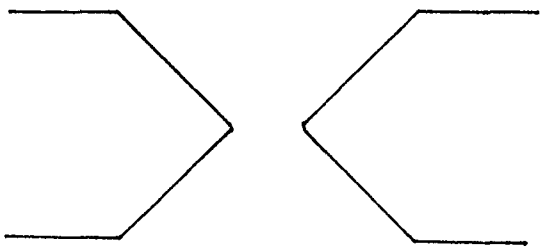

POWER [W]
1000
500
0
0
50
100
150
200
TIME [μs]

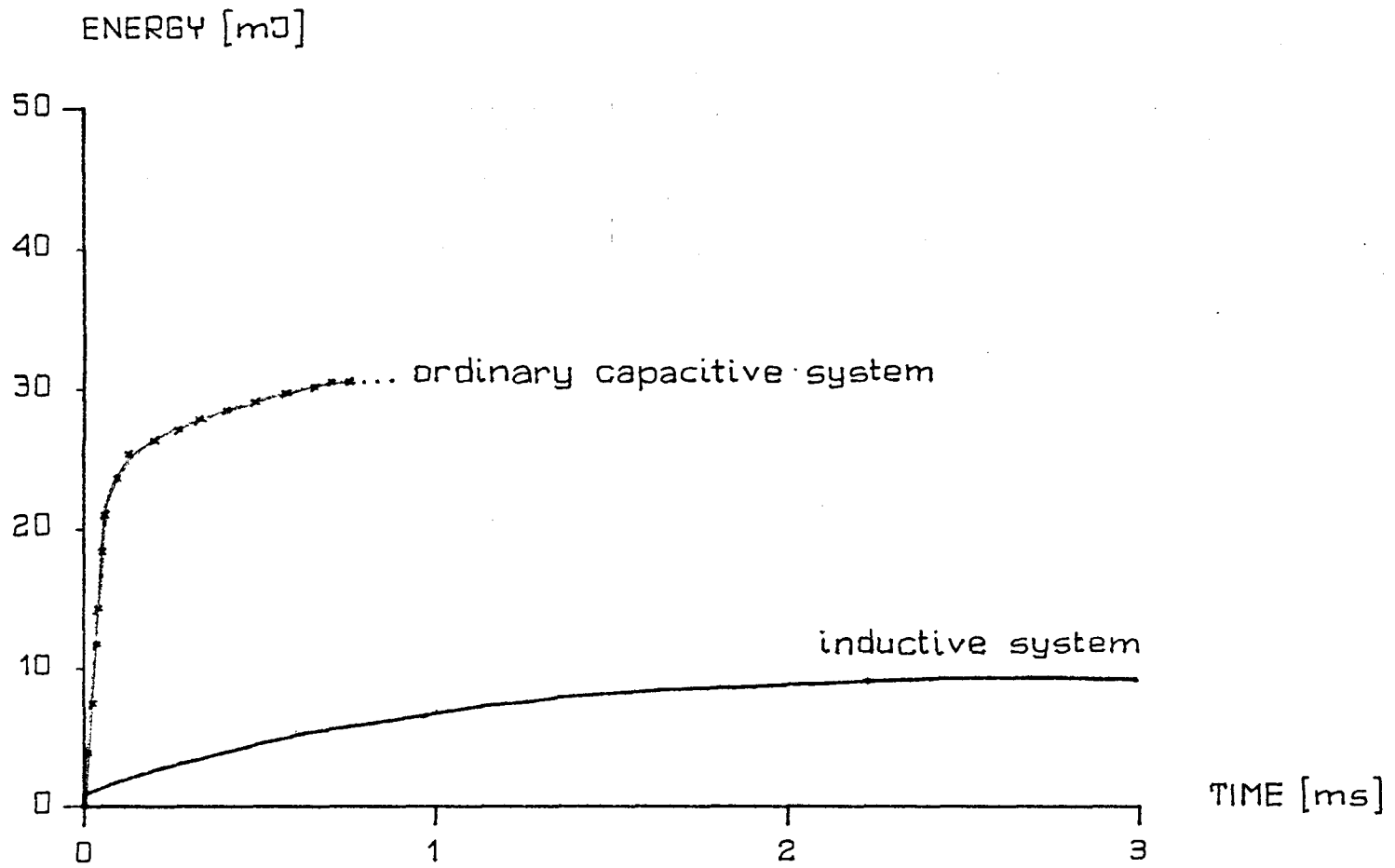

POWER
[W]
1000
500
0
0
1
2
3
4
μs
POWER
[W]
10
5
0.5
1.0
1.5
2.0
2.5
3.0
ms
TIME


POWER
[W]
1000
500
0
50
100
150
TIME [μs]








ENERGY [mJ]
50
40
30
20
10
0
... ordinary capacitive system
inductive system
0
1
2
3
TIME [ms]

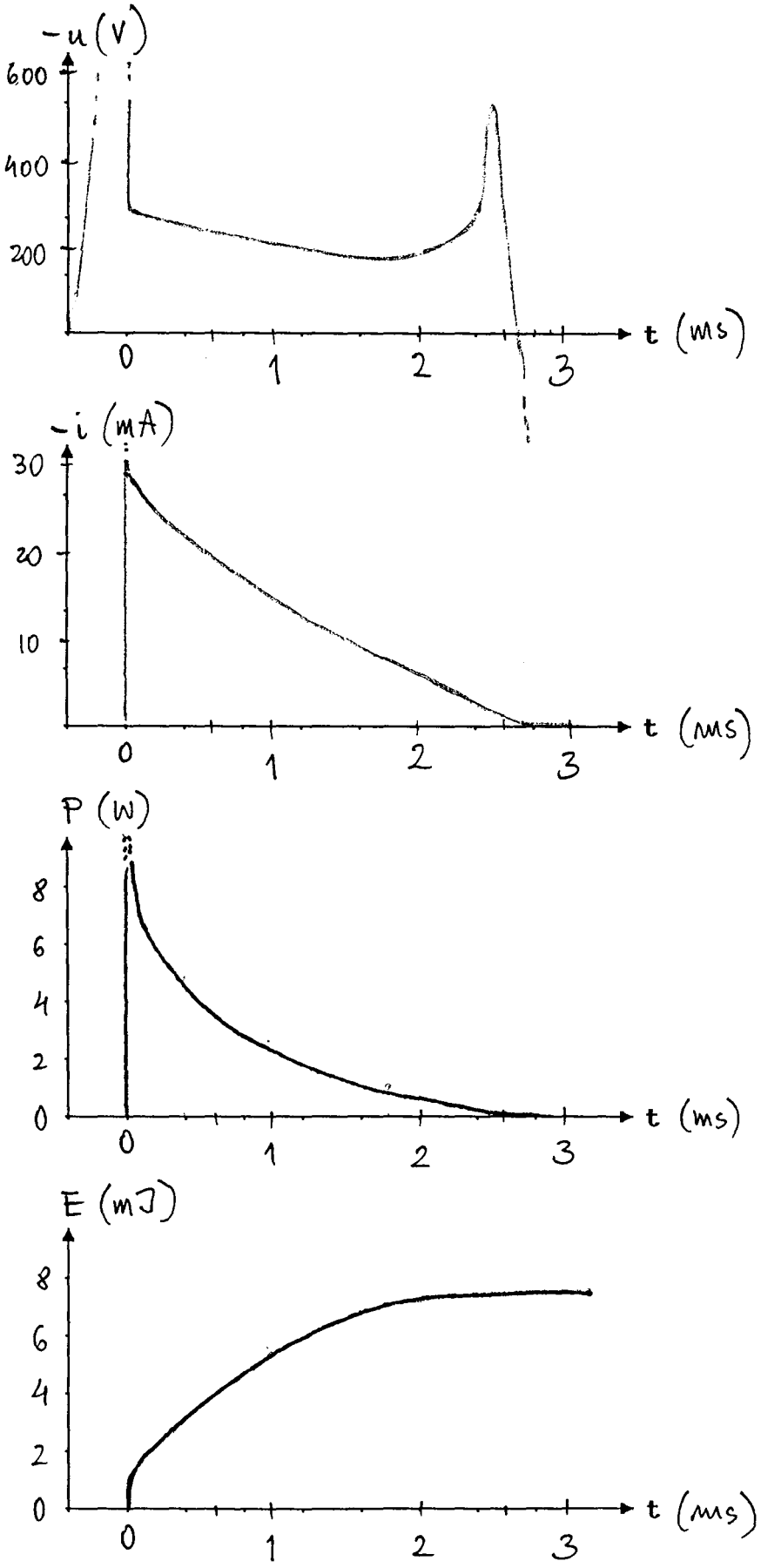


Fig.5. The voltage, the current, the power and the energy of the inductive system. (STANDARD PLUG)

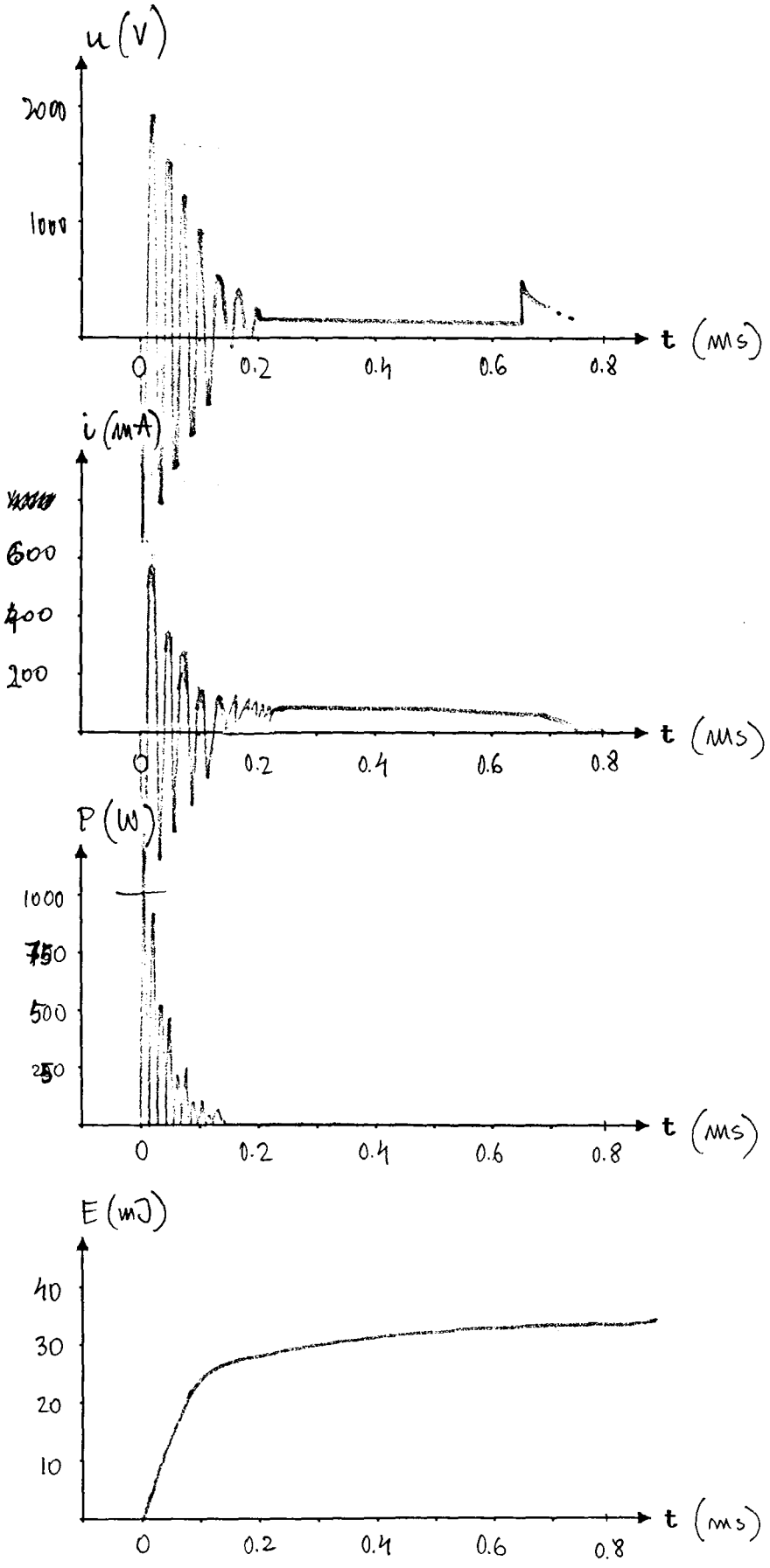
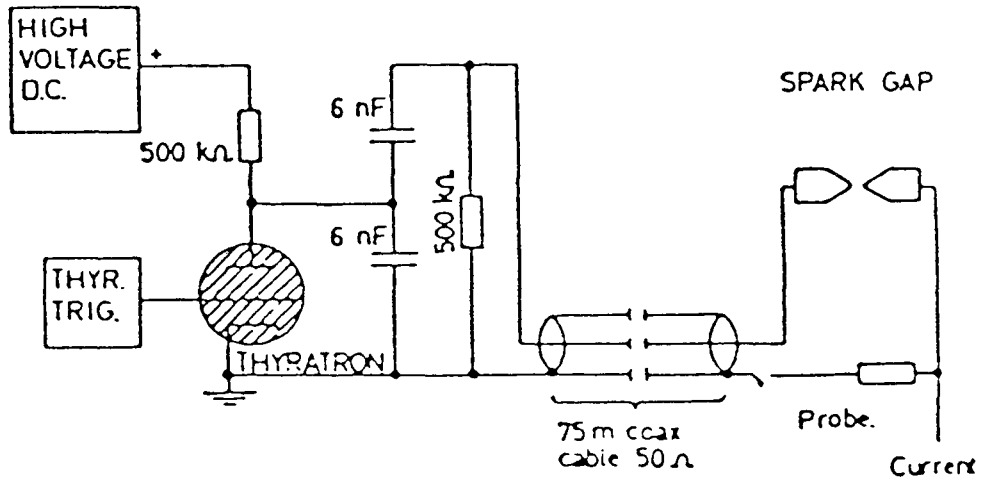


Fig.6. The voltage, the current, the power and the energy of the commercial capacitive system. (STANDARD PLUG)


HIGH
VOLTAGE
D.C.
+
500 kΩ
6 nF
6 nF
500 kΩ
THYR.
TRIG.
THYRATRON
SPARK GAP
75 m coax
cable 50 Ω
Probe.
Current

OPTICAL MEASUREMENTS

3.1 INTRODUCTION

A number of different optical devices and methods were employed to determine the temporal temperature distribution and the electron densities in the sparks.

All the experiments were carried out under similar circumstances. The parameters held constant were:

- the battery voltage - approximately 12 volts DC,
- the spark environment - nitrogen gas,
- the temperature - 22 degrees Centigrade.

The atmospheric pressure was read off continuously during the experiments.

Below follows a description of the devices used in the experiments. Also a short summary of the theory is given.

Optical interference is a basic concept of the subject of optics. According to the wave theory, light can be interpreted as an electromagnetical wave having in every point in space and in every moment in time a well-defined value of the electric-field density **E** and the magnetic-field density **B**. According to the same theory the resulting electric-field density, at a point in space where two or more light waves overlap, is equal to the vector sum of the individual contituent disturbances obeying the important principle of superposition. Briefly, optical interference may be termed as an interaction of two or more light waves yielding a resultant irradiance which deviates from the sum of the component irradiances. Optical systems taking advantage of the phenomenon of interference are called intereferometers. Monochromatic laser light is often used in interferometers.

The Mach-Zehnder interferometer is an amplitude-splitting device, that is where the incoming primary light wave is devided into two segments which travel different paths before recombining and interfering. As shown in Fig.3.1 the Mach-Zehnder interferometer consists of two beam splitters and two totally reflecting mirrors. The two waves within the apparatus travel along separate paths. A difference between the optical paths can be introduced by a slight tilt of one of the beam splitters. Since the two paths are

separated, the interferometer is relatively difficult to align. For the very same reason, however, it finds a large number of applications. The pattern on the interferograms is a series of parallell and mutually interchanging dark and light fringes.

Fig. 3.1. The Mach-Zehnder interferometer

Introducing an object with varying index of refraction, such as a hot spark plasma, in one beam alters the optical path-length difference, thereby changing the fringe pattern. The example of a fringe pattern obtained is shown in Fig. 3.2.

Fig.3.2. The fringe patterns of the hot plasma.

The dye-laser is a laser in which the active medium consists of solution of certain organic dye compounds in liquids. The laser wavelength can be tuned continuously within a large range. The dye laser used was the Lambda Physik FL 2002 and operated in the pulse mode. The pulse energy was in the milijoule region. The duration of the pulses was 15 ns and the wavelength was 633 nm. This wavelenth was chosen for the best fringe pattern visibility.

The excimer laser Lambda Physik EMG 102 was employed as a pump laser for the dye laser. Excimer lasers use excimers as the active medium. An excimer is a molecule that only exists in the excited state. It has accordingly no stable ground state. The excimer used was XeCl that produced intensive 15 ns long light pulses of 308 nm wavelength.

The camera used for the registration of the interferograms was a standard 35 mm Nikkormat camera equipped with a 300 mm lens. The magnification factor was approximately 0.8.

The Abel inversion method.

PRELIMINARY CONSIDERATIONS

When an electromagnetic wave travels through a medium with a refractive index μ, the change of the optical pathlength ΔL is giv by:

$$\Delta L = \int \Delta\mu(l)\,dl \qquad (3:1)$$

where $\Delta\mu$ is the variation of the refractive index along the path and L is the geometrical length of the medium. The components of the medium, in the general case the spark plasma consisting of molecules, atoms, ions and electrons, contribute to the refractive index, and the individual contributions are calculated from the specific refractivities K_i, and the number densities n_i, as:

$$\mu-1 = \sum K_i n_i \qquad (3:2)$$

In the case of a single species' plasma this can be written as:

$$\mu-1 = K_e n_e+K_m n_m+K_a n_a+K_1 n_1+K_2 n_2+... \qquad (3:3)$$

where the subscript e stands for electrons,
m for molecules,
a for atoms,
1 for ions with charge -1,
2 for ions with charge -2.

In the interferometric measurement the refractive index is measured with the gas at room temperature and atmospheric pressure as a reference (i.e. no molecular dissociation or ionization present). Then the variation of the refractive index becomes:

$$\Delta\mu = K_e n_e+K_m (n_m-n_o)+K_a n_a+K_1 n_1+K_2 n_2+... \qquad (3:4)$$

where the undisturbed molecular density n_o, is calculated from the ideal gas law:

$$pV = NRT$$

as
$$n_o\ [cm^{-3}] = 9.6570\cdot 10^{18} \cdot \frac{p\ [torr]}{T\ [Kelvin]} \qquad (3:5)$$

The measured quantity of the interferometric images is the number of fringe shifts due to the density variations along the path of integration which equals:

$$\Delta N \cdot \frac{\Delta L}{\lambda} = \frac{1}{\lambda}\int \Delta\mu\cdot dl \qquad (3:6)$$

ABEL INVERSION

The spark is assumed to be cylindrically symmetric. Then the number densities in equation 3:3 are functions of a radial coordinate r, with r=a at the center of the spark. Consequently, the variation of the refractive index is also a function of r. The integration path in equation 3:6 should therefore be transformed to radial coordinates as follows:

$$x^2+y^2 = r^2$$

$$X^2+y^2 = R^2$$

Fig. 3.3. Geometrical relations between variables.

From figure 3.3 we see that equation 3:6 in this case should be written

$$\Delta N(y) = \frac{1}{\lambda} \cdot \int \Delta\mu(r) \cdot dx \qquad (3:7)$$

The integration is performed along a strip of constant y, and a variable transformation from the Cartesian coordinates to the radial coordinates is done according to:

$$dx = \frac{r}{(r^2-y^2)^{1/2}} \cdot dr$$

and

$$X = (R^2-y^2)^{1/2}$$

That gives:

$$\Delta N(y) = \frac{2}{\lambda} \cdot \int \frac{\Delta\mu(r) \cdot r \cdot dr}{(r^2-y^2)^{1/2}} \qquad (3:8)$$

where the symmetry about the y-axis has been taken advantage of. The equation 3:8 is one form of the Abel integral equation. If $\Delta\mu(r)$ is assumed to be zero for r>R, it is possible to invert 3:8 analytically into:

$$\Delta\mu(r) = -\frac{\lambda}{\pi}\cdot\int \frac{\frac{d(\Delta N(y))}{dy}}{(y^2-r^2)^{1/2}}\cdot dy \qquad (3:9)$$

or, equivalently, if the $\Delta N(y)$ function is well-behaved:

$$\Delta\mu(r) = -\frac{\lambda}{\pi r}\cdot\frac{d}{dr}\cdot\int \frac{\Delta N(y)\cdot y}{(y^2-r^2)^{1/2}}\cdot dy \qquad (3:10)$$

These two last equations are the Abel inversion formulas and they follow from equation 3:8 in the case of cylindrical symmetry.

The Abel inversion method is very well suited for numerical treatment on computers.

The fringe shift values are obtained as a set of discrete numerical values from the measurements of the interferometric images. If a numerical inversion method is employed (based on equation 3:9) such as Bockasten's method [ref.8] for instance, serious complications may arise. When the numerical derivative in equation 3:9 is evaluated, the noise of the data is drastically amplified. These methods should therefore only be used when the random errors in the fringe shift values are negligible. In the second inversion formula of equation 3:10, the order of integration and differentiation is reversed and this reduces the noise amplification. An example of a method based on that relation is the method proposed by Barr [ref.9] which will now be discribed.

The formula 3:10 can be divided into two parts:

$$F(r) = 2\cdot\int \frac{\Delta N(y)\cdot y}{(y^2-r^2)^{1/2}}\cdot dy \qquad (3:11)$$

and

$$\Delta\mu(r) = \frac{\lambda}{2\pi r}\cdot\frac{dF(r)}{dr} \qquad (3:12)$$

If we now require that the fringe shift values should be measured at equidistant points, i.e. the coordinates:

$$y_n = n\cdot\Delta \qquad (3:13)$$

where n are integers in the range

$$0 \le n \le N \qquad (3:14)$$

and $$R = N\cdot\Delta \qquad (3:15)$$

Consequently, the Abel inverted fringe shift values per unit length will be calculated at the points:

$$r_n = n\cdot\Delta \qquad (3:16)$$

From the measured fringe shift values ΔN_n, a fringe shift function $\Delta N(y)$ is now constructed by assuming that that function must consist of second-order polynomials between the measured points:

$$\Delta N(y) = a_n + b_n \cdot y^2 , \quad y_n \leq y \leq y_{n+1} \tag{3:17}$$

where the coefficients are determined by requiring that the measured values and the constructed curve's values should be the same:

$$\Delta N(y_n) = \Delta N_n \tag{3:18}$$

$$\Delta N(y_{n+1}) = \Delta N_{n+1} \tag{3:19}$$

This form was chosen because it has the necessary zero-slope at y=0 for the convergence condition and it has an accuracy at least as good as the accuracy in the ΔN_n values. The integration of equation 3:17 can now be performed analytically to give the F(r) function as a set of discrete values F_k dependent on the measured fringe shifts:

$$F_k = F(r_k) = \Delta \sum_{n=k}^{N} \alpha_{kn} \Delta N_n \tag{3:20}$$

The expression for the coefficients α_{kn} can be found in Barr's article. Here it is enough to note that they are slowly varying functions of n and k, and that the F_k-values are relatively insensitive to small random errors in the ΔN_n-values.

A least-squares method is then applied to fit a polynomial F(k) into each section of the F_k versus k-curve. This is sufficient since the errors in the F_k-values are small. The F_k-values are represented at each point by a polynomial F(k) of the form:

$$F(k) = (A_k + B_k k^2 + C_k k^4) \cdot \Delta \tag{3:21}$$

where the coefficients A_k, B_k and C_k are determined in terms of α_{kn} and ΔN_n by requiring that the sum of the differences between F_k (equation 3:20) and F(k) (equation 3:21) over the five points from k-2 to k+2 should be as small as possible. The form of the polynomial 3:21 has been chosen because it gives the best fit to a Gaussian profile. For the two points k=0 and k=1, the points k=0,1,2,3,4 were used for the fit.

When the expression 3:21 is substituted into the equation 3:12 we obtain:

$$\Delta\mu(r_k) = \frac{\lambda}{2\pi\Delta^2 k} \cdot \frac{dF(k)}{dk} = -\frac{\lambda}{\pi\Delta} \cdot (B_k + 2C_k k^2) \tag{3:22}$$

The combination of two coefficients is then of the form:

$$B_k + 2C_k k^2 = \begin{cases} -\sum_{n=k-2}^{N} \beta_{kn} \Delta N_n & , k \geq 2 \\ -\sum_{n=0}^{N} \beta_{kn} \Delta N_n & , k < 2 \end{cases} \tag{3:23}$$

where the β_{kn} are functions of k and n only, and not dependent on N.

Combining the formulas 3:22 and 3:23 produces the final result - the Abel inversion formula:

$$\Delta\mu_k = \Delta\mu(r_k) = \begin{cases} \frac{\lambda}{\pi\Delta} \cdot \sum \beta_{kn} \Delta N_n & , k \geq 2 \\ \frac{\lambda}{\pi\Delta} \cdot \sum \beta_{kn} \Delta N_n & , k < 2 \end{cases} \qquad (3:24)$$

In the coefficients β_{kn} the entire process of integration, least-square fitting and final differentiation is incorporated. The values of the β_{kn}-coefficients (multiplied by a facor -10^4) are given in the program listing [Appendix]. Barr recommends that this method should be used when the noise of the input data is of the order of magnitude of one percent.

Finally it must be mentioned that because the β_{kn}-coefficients are obtained by the fitting of the F(k)-polynomials (equation 3:21) into a Gaussian profile, the final values of the inversion for small k-values are smaller than the real values and slightly higher than the real values for high k-values.

REFRACTIVITY OF THE PLASMA COMPONENTS

In the preceding section the procedure of how to calculate the variation of the refractive index as a function of the radial coordinate from the measured fringe-shift patterns, was outlined. Using equation 3:4 it is then possible to calculate the number densities of some plasma components if the refractivities are known and some additional assumptions about the state of the plasma are made.

Both classical and quantum mechanics agree on the formula for the index of refraction:

$$\mu - 1 = \frac{2 \cdot e^2}{m} \sum n_l \sum \frac{f_{lk}}{\omega_{lk} - \omega^2} \quad , \ \omega \neq \omega_{lk} \qquad (3:25)$$

where n_l is the particle density in quantum state l,
m is the electronic mass,
f_{lk} is the oscillator strength for transitions between states l and k,
ω_{lk} is the angular frequency of the line corresponding to the transition from l to k,
and ω is the frequency of the electromagnetic wave passing through the medium.

There are, however, some difficulties when using the formula. The oscillator strengths of many levels must be known, and the transitions from discrete to continuum states must be included. Moreover, expression 3:25 is strictly true only if the wavelength of

the impinging radiation is far away from any resonance wavelengths. Otherwise, imaginary damping constants have to be included in the denominator.

If the used wavelengths are far from the resonance wavelengths and only a restricted range of frequencies are allowed, it is possible to expand the formula 3:25 into a power series of λ^{-2}. In that way we arrive at the Cauchy formula:

$$\mu - 1 = A + \frac{B}{\lambda^2} = K_r n_r = \left(A_r + \frac{B_r}{\lambda^2}\right) \cdot n_r \qquad (3:26)$$

which is a good approximation for a given species' ground state in the desired range. For nitrogen molecules the numerical values of the constants at room temperature, a pressure of 1 atmosphere and in the range of the visible electromagnetic radiation gives the following formula for the refractivity:

$$K_m\ [cm^3] = 1.08 \cdot 10^{-23} + \frac{7.6 \cdot 10^{-34}}{(\lambda\ [cm])^2} \qquad (3:27)$$

The same relation is valid for nitrogen atoms and ions (with different values of the constants A and B, but since these expressions will not be needed in the subsequent treatment, they are not given here). The refractivity is also dependent on the temperature especially when the temperature is high. The contribution from excited states becomes also more dominant.

Finally, it should be mentioned that the normally dominating contribution to the plasmatic refractivity comes from the free electrons. If the incoming radiation's angular frequency is much greater than the electron plasma frequency:

$$\omega_p\ [s^{-1}] = \left(\frac{4\eta n_e e^2}{m}\right)^{1/2} = 5.64 \cdot 10^4 \cdot (n_e\ [cm^{-3}])^{1/2} \qquad (3:28)$$

the free electron refractive index is:

$$\mu_e - 1 = K_e n_e = -\frac{2\eta e^2}{m}\frac{1}{\omega^2} n_e = -4.46 \cdot 10^{-14} \cdot n_e\ [cm^{-3}] \cdot (\lambda\ [cm])^2 \qquad (3:29)$$

The frequency dependence of the electronic, atomic and ionic refractivity is given in the figure 3.4. The ratio between the two last refractivities is:

$$\frac{K_i}{K_n} \approx 0.63 \qquad (3:30)$$

Fig.3.4. K values as function of wavelength

CALCULATION OF THE DISSOCIATION EQUILIBRIUM

In order to calculate the composition of the spark plasma, a way to calculate the extent of the dissociation of nitrogen molecules has to be devised.

In the general case of a gas-phase chemical reaction:

$$\sum a_i A_i \Leftrightarrow \sum b_i B_i \qquad (3:31)$$

where A_i are the reactants, B_i are the products and a_i ,b_i are their respective stochiometric coefficients, two related equilibrium constants can be defined. The one is the pressure equilibrium constant K_p, defined in terms of the partial pressures:

$$K_p = \frac{\prod p^{b_i(B_i)}}{\prod p^{a_i(A_i)}} \qquad (3:32)$$

where K_p depends on the partition functions of the products and reactants, through:

$$\ln K_p = -\frac{\Delta E_o}{RT} + \sum b_i \cdot \ln Q_p(B_i) - \sum a_i \cdot \ln Q_p(A_i) \qquad (3:33)$$

where ΔE_o is the zero-point energy difference between the products and the reactants in their reference standard states:

$$\Delta E_o = \sum b_i \cdot E_o(B_i) - \sum a_i \cdot E_o(A_i) \qquad (3:34)$$

Q_p in formula 3:33 is the partition function of the standard state of unit pressure, and it is related to the total partition function Q by:

$$Q_p = p \cdot Q \qquad (3:35)$$

and the total partition function can be split into a translational, vibrational, rotational and electronical part:

$$Q = Q_t Q_v Q_r Q_e$$

if the different degrees of freedom are assumed to be uncoupled.

The other equilibrium constant, the concentration equilibrium constant K_c, is defined by:

$$K_c = \frac{\prod n^{b_i(B_i)}}{\prod n^{a_i(A_i)}} \qquad (3:37)$$

where $n(A_i)$ and $n(B_i)$ are the concentrations of reactants and products. For K_c there is a formula similar to equation 3:33, except for the fact that the partition of the standard state of unit pressure is replaced by the partition of the standard state of unit concentration Q_c, which is related to the total partition function by:

$$Q_c = \frac{p}{R \cdot T} \cdot Q \qquad (3:38)$$

From formulas 3:35, 3:38 and 3:33 the relation between the two equilibrium constants is derived:

$$K_c = K_p \cdot (RT)^{\sum a_i - \sum b_i} \qquad (3:39)$$

In the report of Hansen [ref.10] expressions are given that are valid for nitrogen below temperatures of 15000 K. The pressure equilibrium constant is given by:

$$\ln K_p \; (N_2 \Leftrightarrow 2N) = - \frac{113200}{T} + 2 \cdot \ln Q_p(N) - \ln Q_p(N_2) \qquad (3:40)$$

and the total partition functions are given as:

$$\ln Q(N) = \frac{5}{2} \cdot \ln T + 0.30 + \ln\left(4 + 10 \cdot \exp\left\{-\frac{27700}{T}\right\} + 6 \cdot \exp\left\{-\frac{41500}{T}\right\}\right) - \ln p \qquad (3:41)$$

and

$$\ln Q(N_2) = \frac{7}{2} \cdot \ln T - 0.42 - \ln\left(1 - \exp\left\{-\frac{3390}{T}\right\}\right) - \ln p \qquad (3:41)$$

To calculate the equilibrium composition when the presuure and the temperature of the hot gas are known, the quantity ε, describing the fraction of the molecules being dissociated into atoms, is

needed. Using ε, the expressions of the partial pressures of atoms and molecules can be written as:

$$p(N_2) = x(N_2)\cdot p = \frac{1-\varepsilon}{1+\varepsilon}\cdot p \qquad (3:43)$$

$$p(N) = x(N)\cdot p = \frac{2\varepsilon}{1+\varepsilon}\cdot p \qquad (3:44)$$

where x(A) is the mole fraction of the component A. Combining formula 3:32 in the case of nitrogen together with formulas 3:43 and 3:44 yields:

$$K_p = \frac{(p(N))^2}{p(N_2)} = \frac{4\varepsilon^2 p}{1-\varepsilon^2} \qquad (3:45)$$

and resolving ε gives:

$$\varepsilon = \left(1 + \frac{4\cdot p}{K_p} \right)^{-1/2} \qquad (3:46)$$

The relation between ε and the number densities of molecules and atoms is, by the definition of ε, found to be:

$$\frac{n_m^\circ - n_m}{n_m} = \varepsilon \qquad (3:47)$$

where n_m° is the number density of molecules before dissociation, and the obvious relation is:

$$n_m^\circ = n_m + \frac{n_a}{2} \qquad (3:48)$$

Eliminating n_m from the two relations above yields:

$$n_a = \frac{2\varepsilon}{1-\varepsilon}\cdot n_m \qquad (3:49)$$

which is valid when $\varepsilon \neq 1$.

Now, having all the necessary formulas, the calculation of the number densities and the temperature of the nitrogen spark plasma can be performed. Starting with the formula:

$$\Delta\mu_k = \Delta\mu(r_k) = \begin{cases} \frac{\lambda}{\pi\Delta}\cdot\sum \beta_{kn}\Delta N_n & ,k\geq 2 \\ \frac{\lambda}{\pi\Delta}\cdot\sum \beta_{kn}\Delta N_n & ,k<2 \end{cases} \qquad (3:24)$$

the Abel-inverted fringe shifts or the radial variation of the refractive index is calculated from the observed fringe shifts ΔN_n. This quantity is on the left hand side of relation 3:4 and becomes, under the assumption of low temperatures (no ionization, T<10000 K):

$$\Delta\mu_k = K_m(n_{m,k}-n_o) + K_a n_{a,k} \qquad (3:50)$$

The undisturbed molecular density n_o is calculated from:

$$n_o\ [cm^{-3}] = 9.6570\cdot 10^{18}\cdot \frac{p\ [torr]}{T\ [Kelvin]} \qquad (3:5)$$

Rearranging equation 3:49 gives:

$$\frac{\Delta\mu_k}{K_m} + n_o = n_{m,k} + R\cdot n_{a,k} \qquad (3:51)$$

where R is the ratio between the atomic and molecular refractivities of nitrogen:

$$R = \frac{K_a}{K_m} \approx 0.63 \qquad (3:30)$$

and K_m is approximated by A_r in formula 3:26 since the contribution of the second term is negligible. The left-hand side of formula 3:50 is known, and under certain assumptions about the condition of the spark, the two number densities can be calculated.

The assumptions mentioned above is that the spark pressure is the same as the surrounding's pressure (isobaric condition) and the plasma being in thermodynamic equilibrium with the surrounding air (self-relaxation and energy relaxation times of the order of nanoseconds, 10^{-9} s) [ref.11,12].

Setting the atomic number density n_a in equation 3:50 to zero, the molecular number density can now be calculated. From the ideal gas law, with known atmospheric pressure, the actual temperature is calculated. Knowing the temperature, the extent of the dissociation can be calculated from equation 3:46.

Combining relations 3:51 and 3:39 and eliminating the atomic number density yields:

$$n_{m,k} = \frac{1-\varepsilon}{1+\varepsilon(2R-1)} \cdot \left(\frac{\Delta\mu_k}{K_m} + n_o\right) \qquad (3:52)$$

then the atomic number density n_m is readily calculated from equation 3:49.

This iterative procedure is continued by calculating a new temperature from the ideal gas law and that in turn gives a new value of the extent of dissociation, whereupon the procedure is repeated. The whole process is terminated when the difference between two consecutive iteration values is smaller than a desired value.

The iteration process is depicted in figure 3.5.

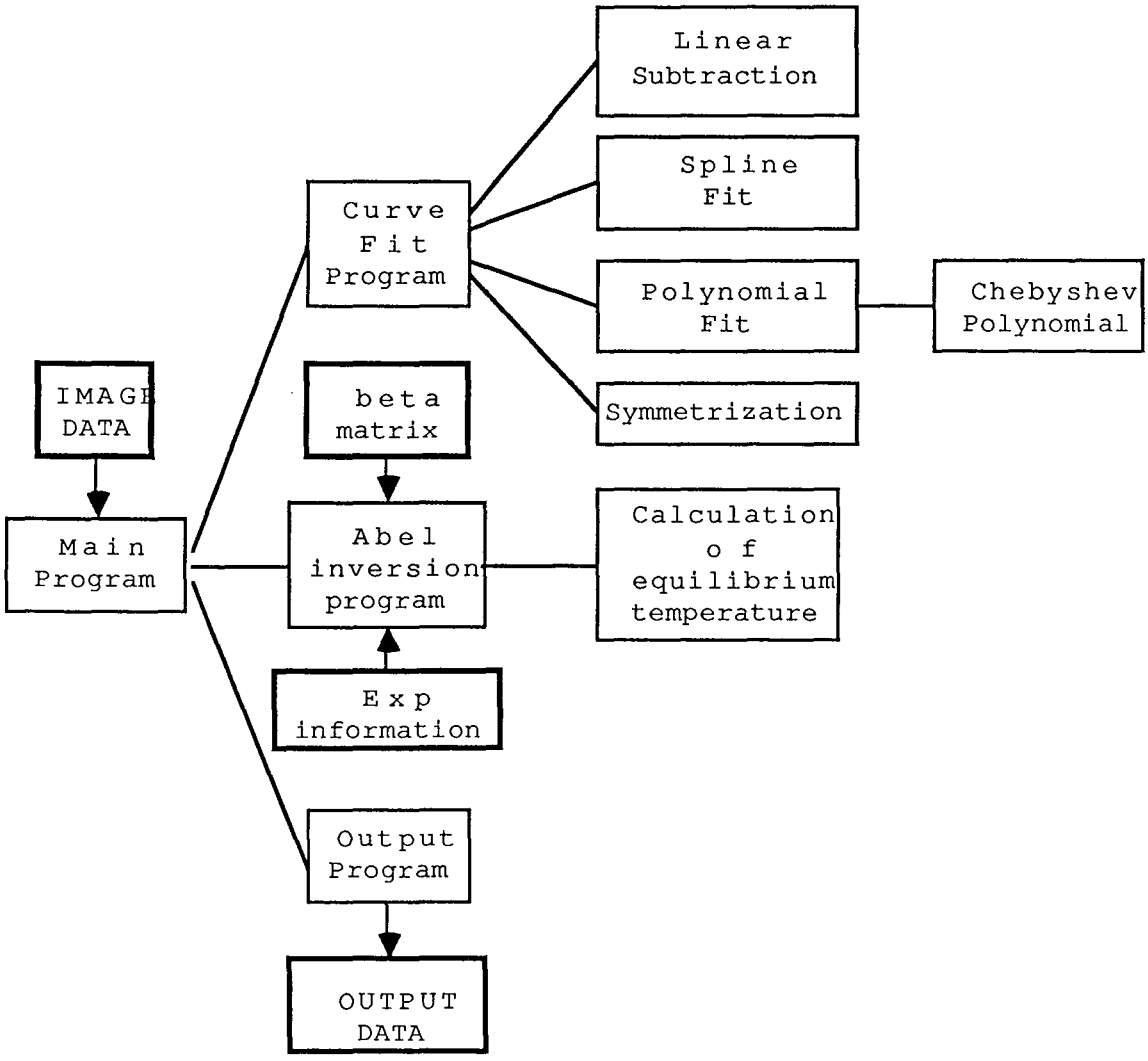


Fig. 3.5. The iteration model.

3.2 EXPERIMENTAL DETAILS

The experimental set-up is shown in Fig. 3.6. The prism behind the dye-laser deflected the thin laser beam to be expanded in the beam expander. Upon leaving the beam expander the beam was parallell and approximately 20 mm in diameter.

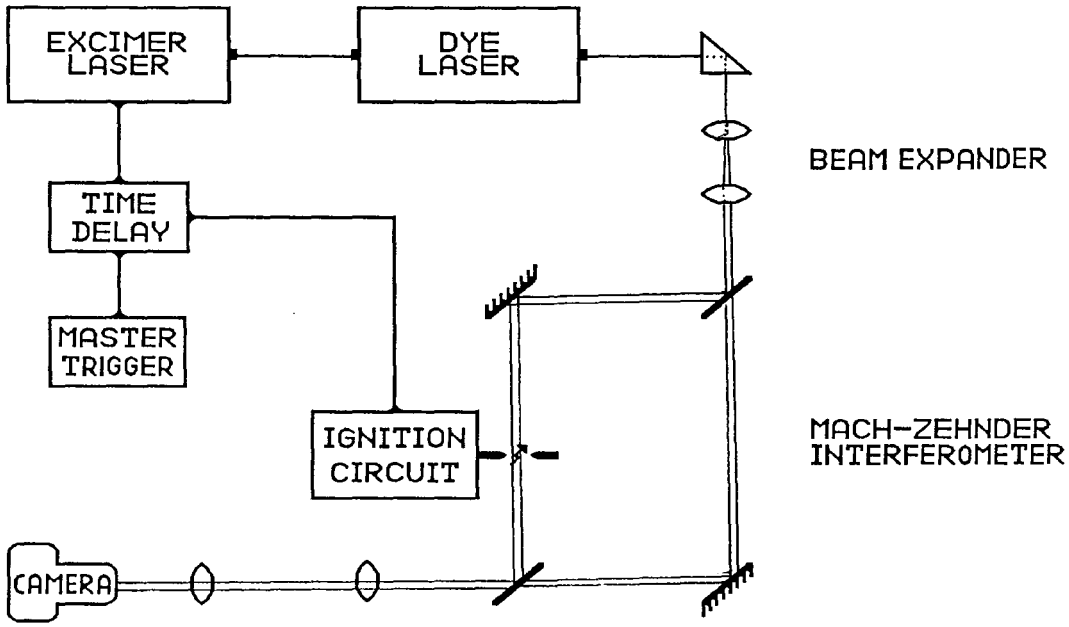


Fig. 3.6. Optical arrangement for time resolved one-wavelength interferometry

The spark plasma was centered in one beam of the interferometer causing a disturbance in the resulting fringe pattern of the interferograms due to its varying index of refraction. The interferograms were finally recorded photographically.

In order to perform measurements at different times after the spark breakthrough the laser pulse was electronically delayed compared to the triggering pulse of the ignition system. The exact delay time between the start of the spark and the laser pulse was measured by comparing the current form of the spark and a vaccum diode signal of the laser pulse on a fast oscilloscope. For the inductive system there were time fluctuations for a certain adjustment of the delay unit. In order to obtain a correct delay time continuous reading of the values had to be done for each laser shot.

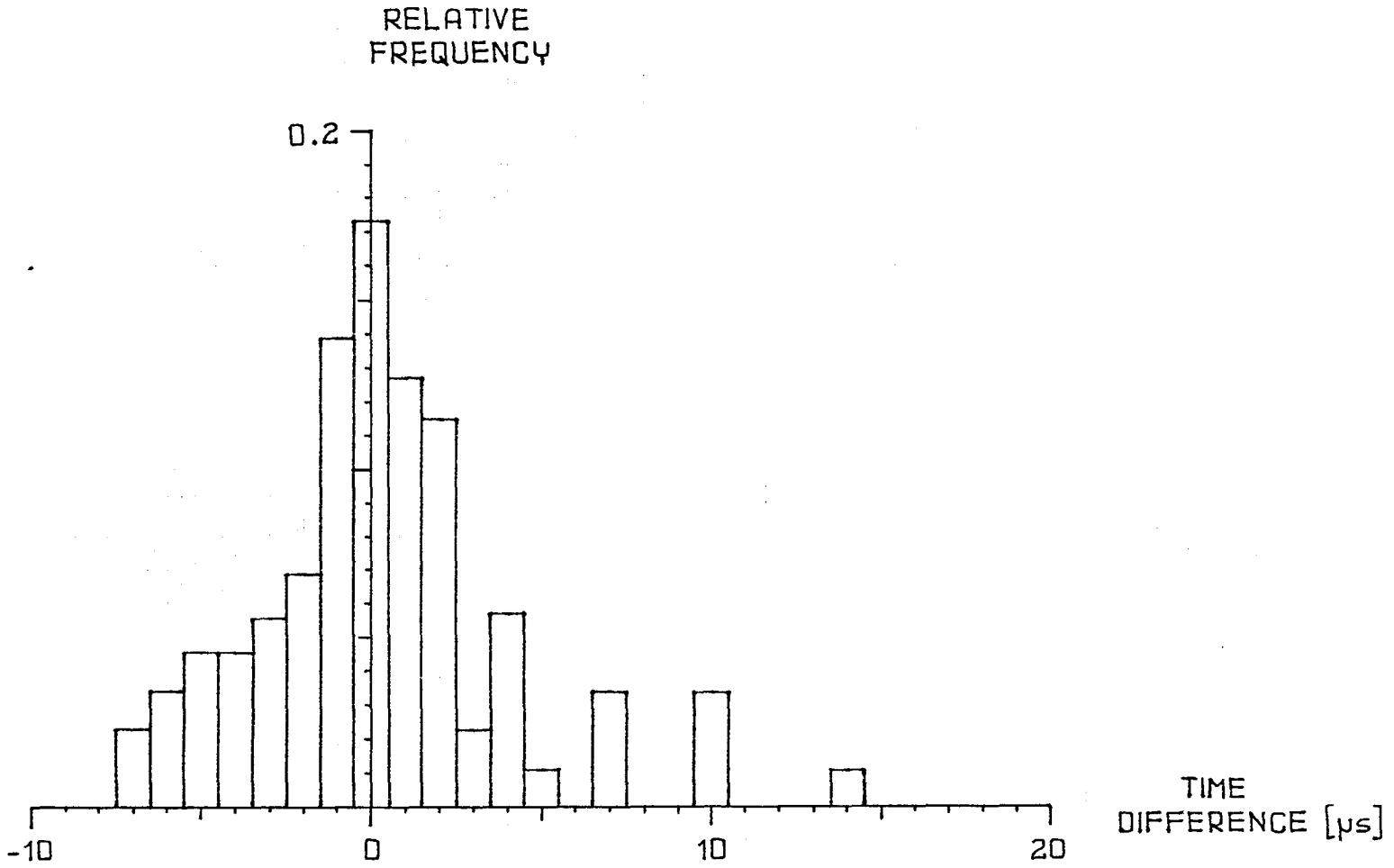
3.3 RESULTS AND DISCUSSION

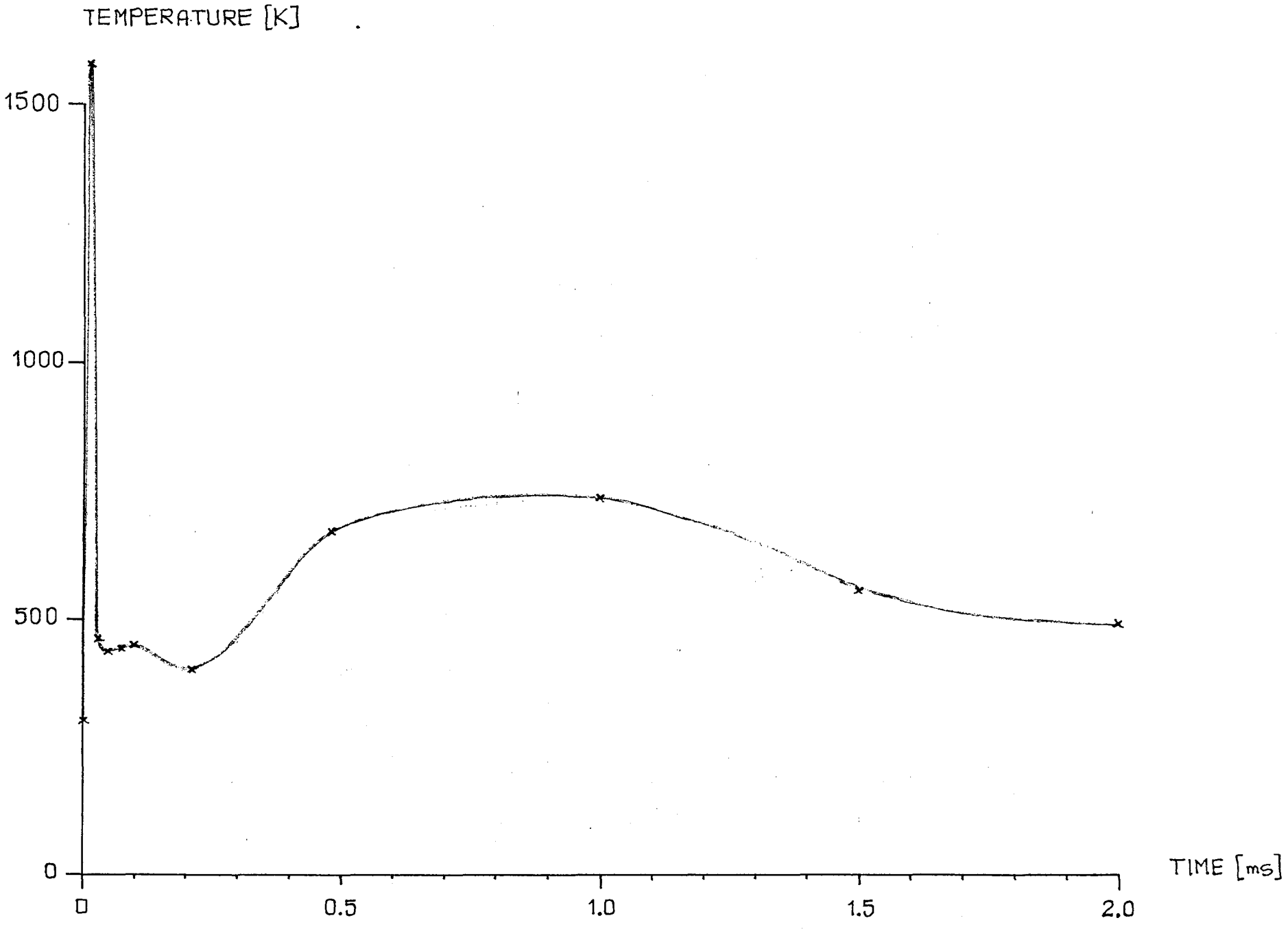
The fringe shifts of the interferograms were carefully measured in several points using an Abbe comparator.

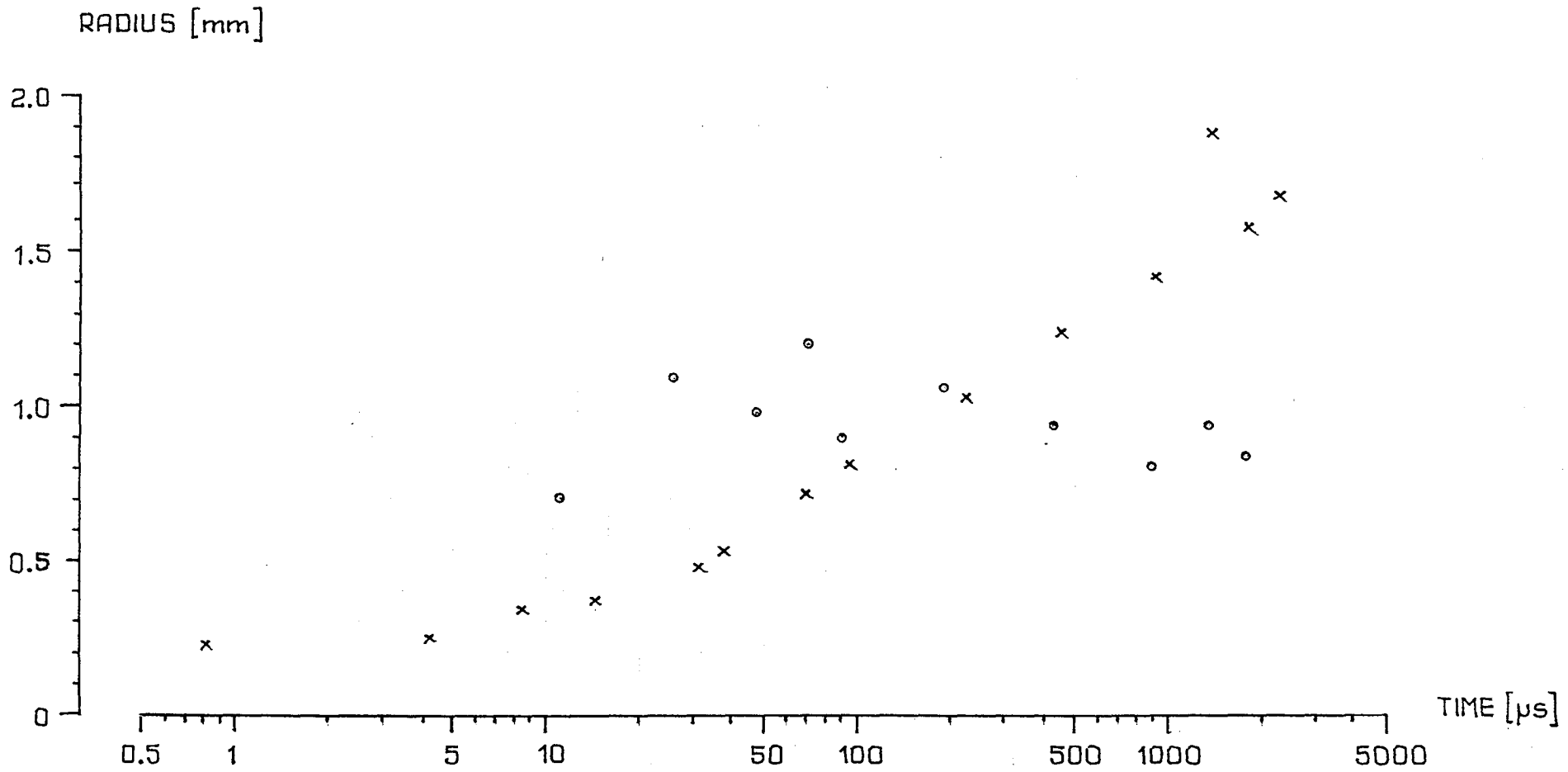
The direct information obtained from these measurements was the mean temperature distribution in time according to Alphert-White law. This can be seen in figures 3.7 - 3.11. The inductive system's mean temperature has a sharp maximum of 5800 K (for the standard plug arrangement) at 1 - 1.5 µs which coincides roughly with the power maximum. The room temperature is reached after approximately 3 ms which also is the time spectrum of the power curve. The commercial capacitive system has a sharp maximum too. It occurs at 6 µs that is slightly later than the inductive system's maximum. It is also lower - approximately 3000 K. Even here there are similarities with the time behaviour of the power - the room temperature is reached after some 100 µs and the power maxima tend to overlap the mean temperature maxima at least for the shorter times. This tendency is kept even for the ultra-fast capacitive system which leads to the conclusion that there is a strong correlation between the mean temperature and the power spectra.

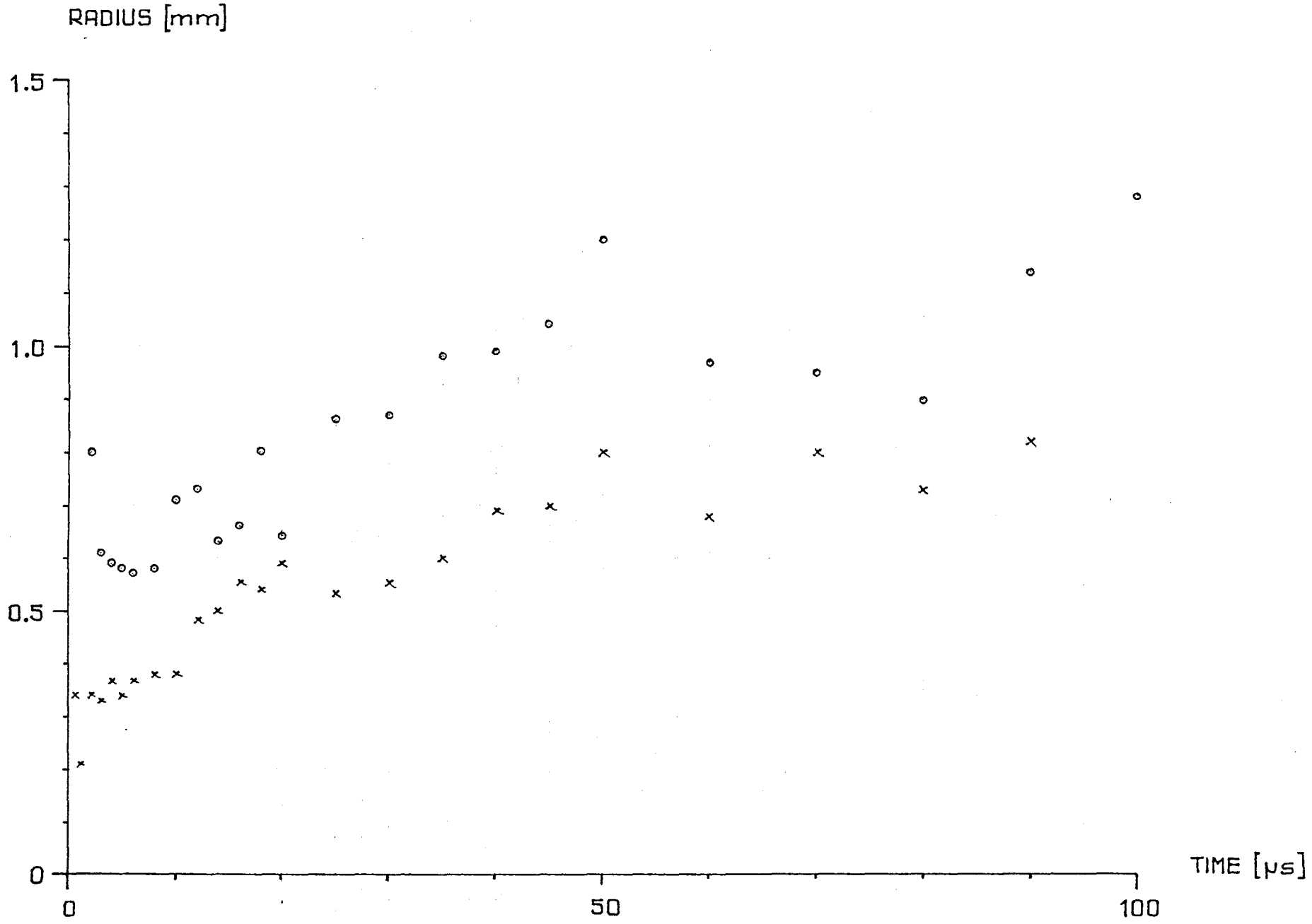
The radial expansion of the sparks is illustraded in figures 3:12 - 3:13.

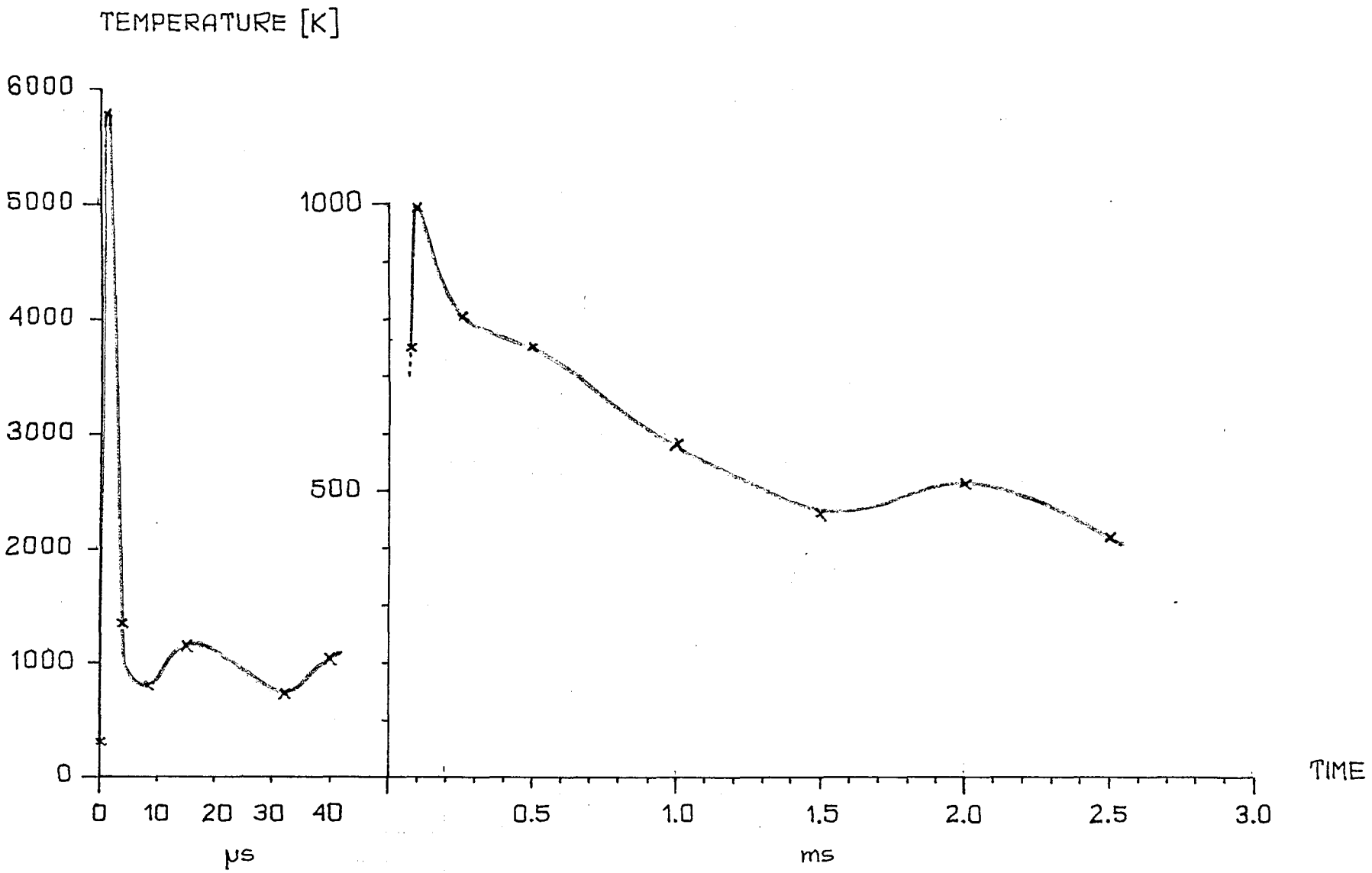
The above mentioned fluctuations in the delay time were registered and the frequency function is seen in figure 3:14. The standard deviation was found to be approximately 4 µs according to simple statistical analysis done.

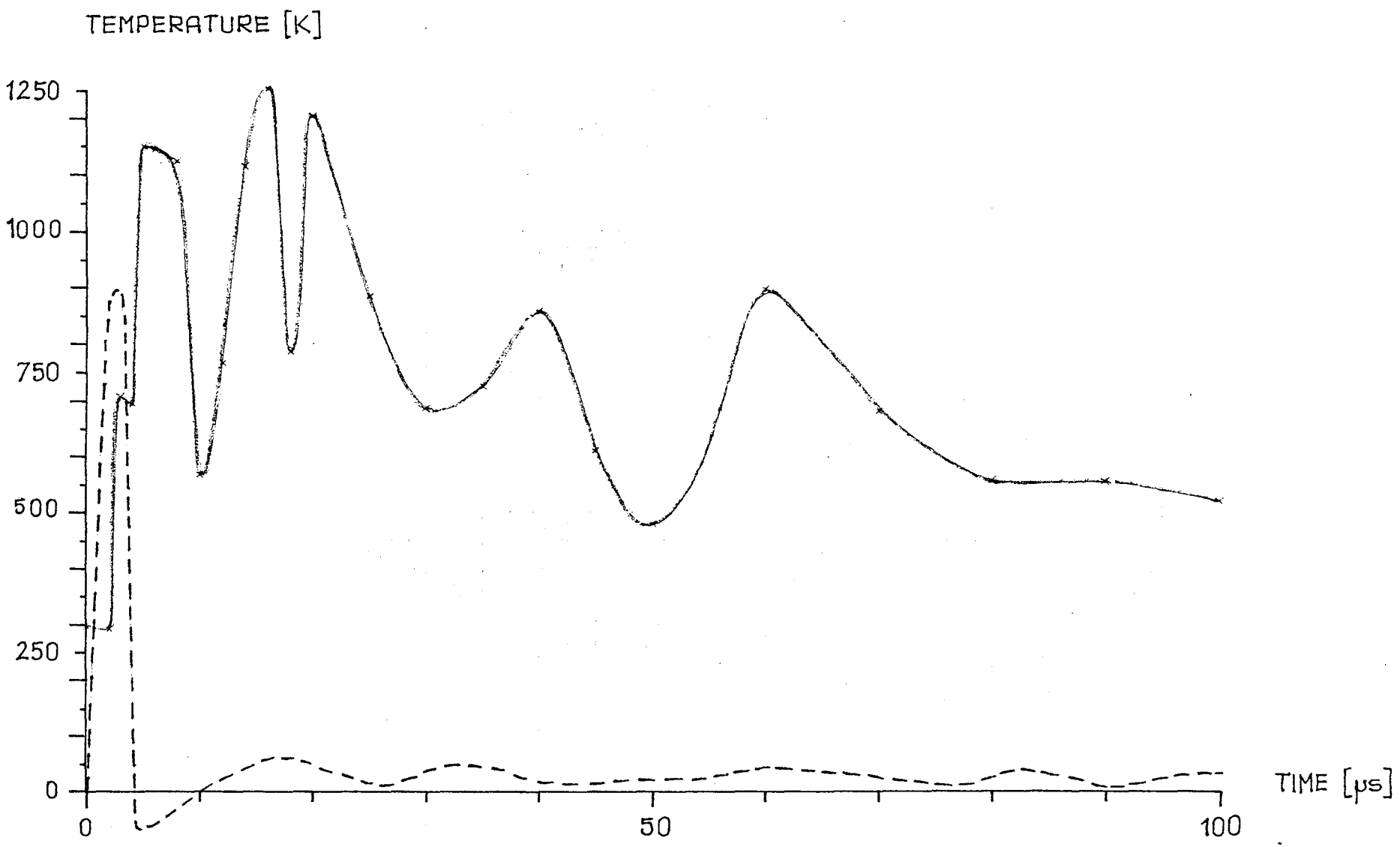

RELATIVE
FREQUENCY
0.2
-10
0
10
20
TIME
DIFFERENCE [μs]

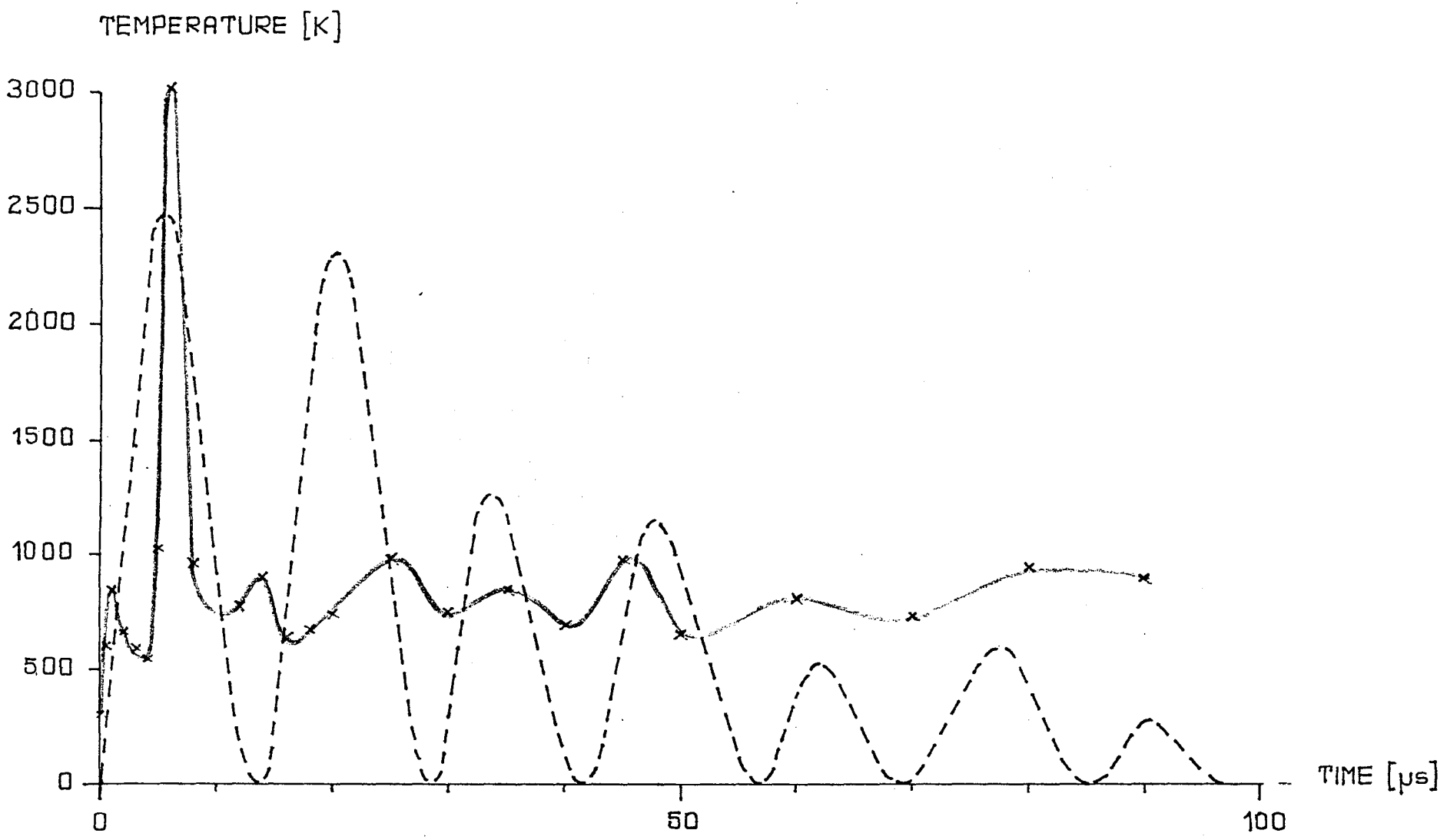

TEMPERATURE [K]
1500
1000
500
0
0
0.5
1.0
1.5
2.0
TIME [ms]


RADIUS [mm]
2.0
1.5
1.0
0.5
0
0.5
1
5
10
50
100
500
1000
5000
TIME [μs]


RADIUS [mm]
1.5
1.0
0.5
0
0
50
100
TIME [μs]


TEMPERATURE [K]
6000
5000
4000
3000
2000
1000
0
1000
500
0
10
20
30
40
μs
0.5
1.0
1.5
2.0
2.5
3.0
ms
TIME


TEMPERATURE [K]
1250
1000
750
500
250
0
0
50
100
TIME [μs]


TEMPERATURE [K]
3000
2500
2000
1500
1000
500
0
0
50
100
TIME [μs]

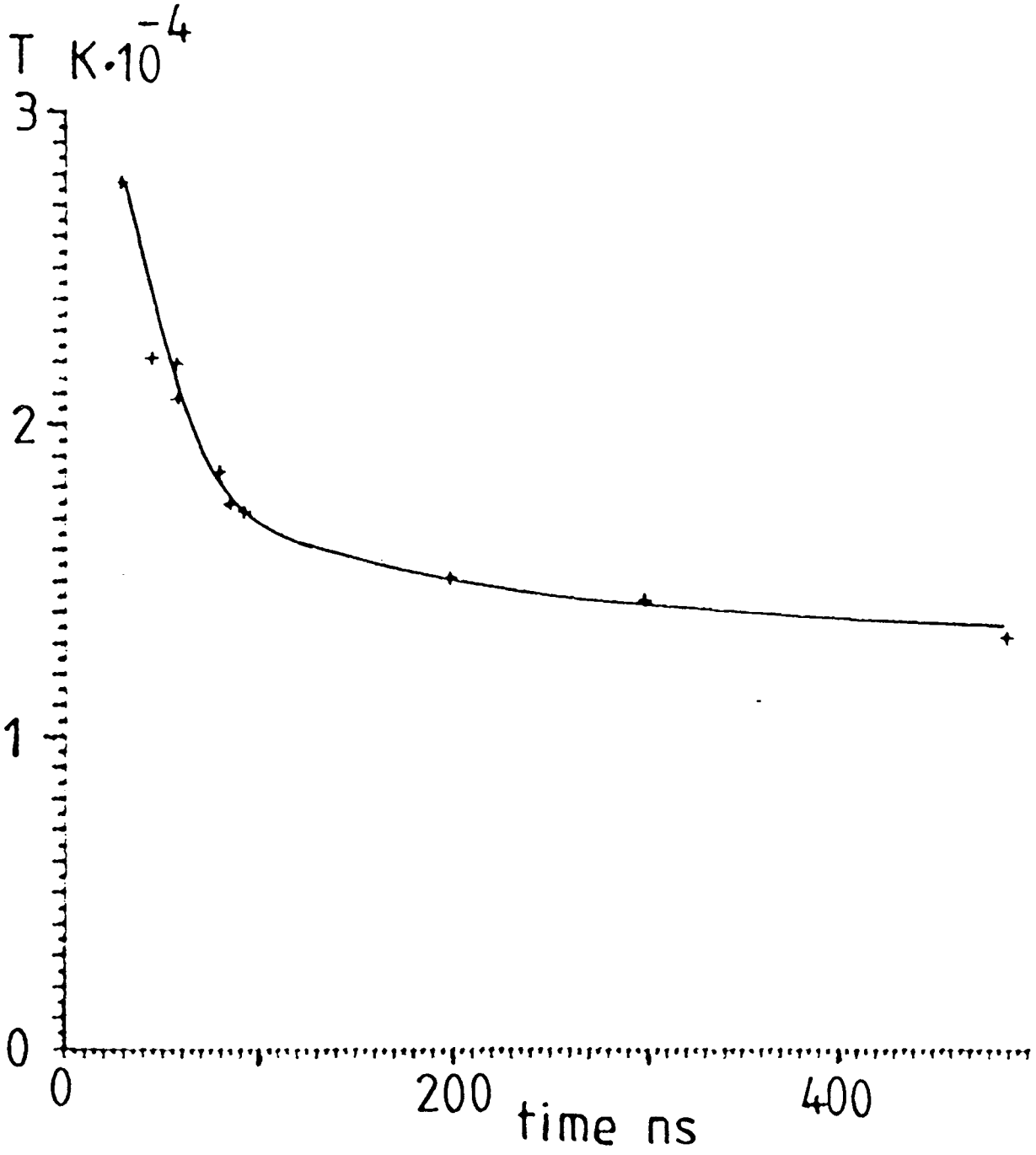
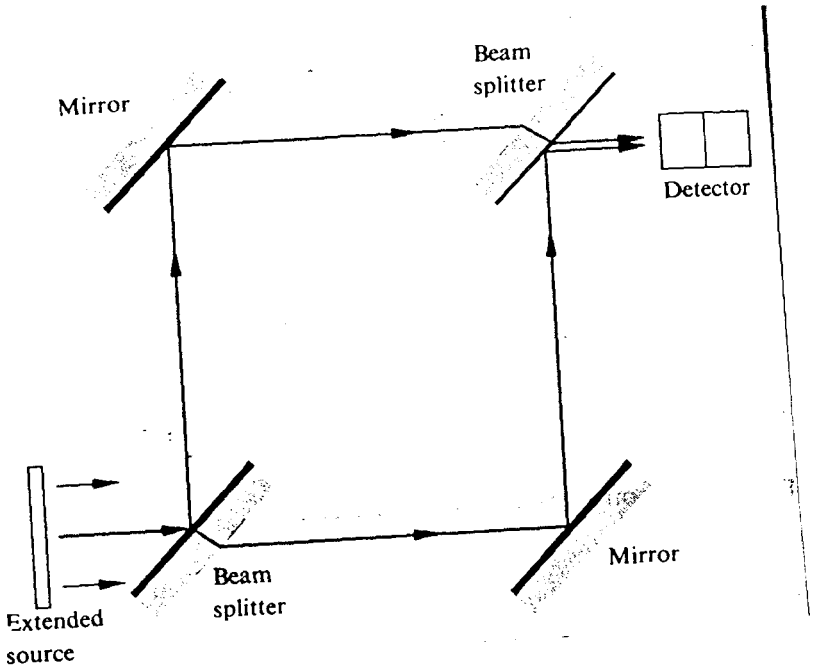
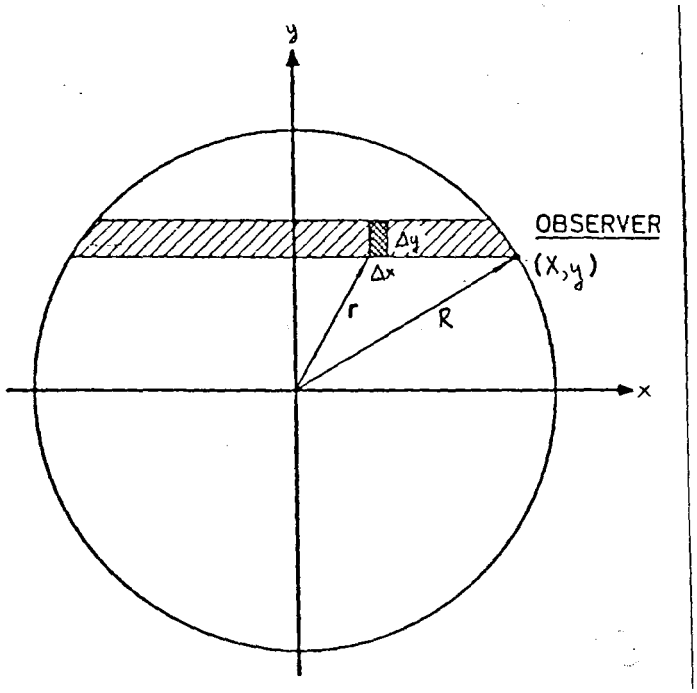
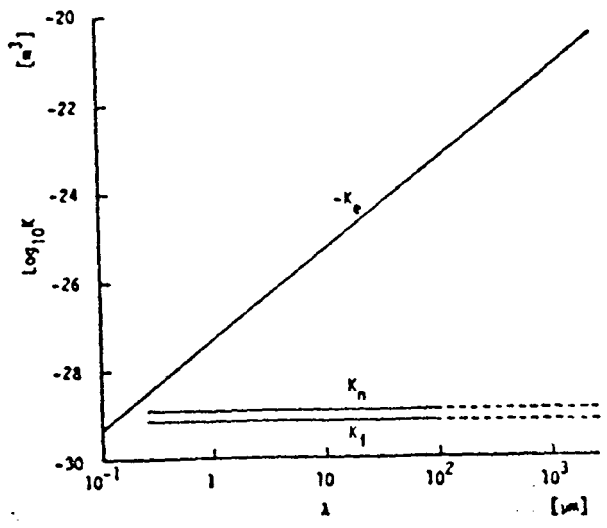
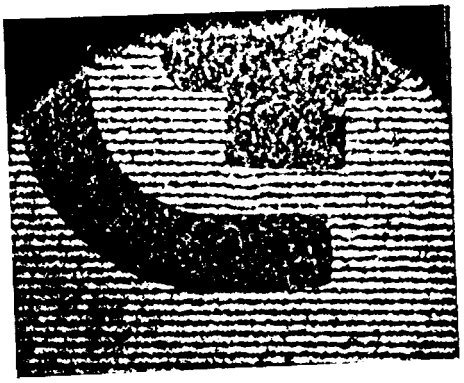


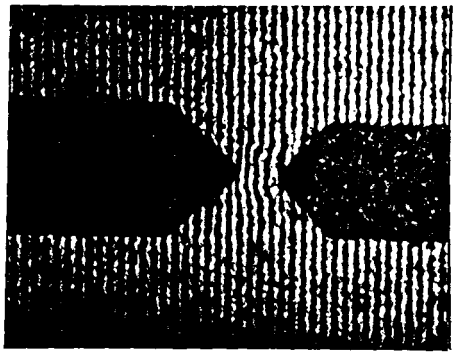
Fig. 3.7 Mean temperature as a function of time.

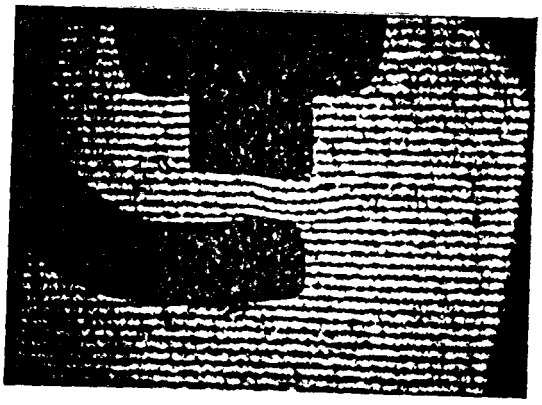

Beam
splitter
Mirror
Detector
Extended
source
Beam
splitter
Mirror


y
OBSERVER
Δy
Δx
(X,y)
r
R
x


-20
-22
-24
-26
-28
-30
$[a^3]$
$\log_{10}K$
K_e
K_n
K_1
10^{-1}
1
10
10^2
10^3
λ
[μm]







ACKNOWLEDGEMENTS

I would like to thank all the people that helped me to complete this paper. In the first place I owe a great debt to Göran S. Holmstedt, Dept. of Physics, Lund Institute of Technology and Hasse Johansson, SAAB-Scania Combitech AB without whose help and patience I couldn't have managed. Among others that deserve a note of gratitude are: Lars Martinsson, Åke Bergqvist and Thure Högberg.

REFERENCES

[1] O.Svelto "Principles Of Lasers", Plenum Press, New York & London, 1982

[2] S.Borgström "Kompendium I Laserfysik", Avd. för Atomfysik, LTH, 1985

[3] E.Hecht, A.Zajac "Optics", Addison-Wesley Publishing Company, 1974

[4] M.Alden, P.Grafström, H.M.Hertz, G.S.Holmstedt, T.Högberg, G.Russberg, S.Svanverg "Characterization Of Ultra-Short High Current Sparks For Ignition Systems", Avd. för Atomfysik, LTH, Avd. för Teoretisk Fysik, Chalmers TH, 1985

[5] M.Deutsch, I.Beniaminy "Inversion of Abel's Integral Equation for Experimental Data", J.Appl.Phys. 54 (1983) 137

[6] W.L.Howes, D.R.Buchele "Optical Interferometry of Inhomogeneous Gases", Journal of the Optical Society of America, Volume 56, Number 11, November 1966

[7] R.A.Alpher, D.R.White "Optical Refractivity of High-Temperature Gases", The Physics of Fluids, Volume 2, Number 2, March - April 1959

[8] K.Bockasten "Transformation of Observed Radiances into Radial Distribution of the Emission of a Plasma", J.Opt.Soc.Am. 51 (1961) 943

[9] W.L.Barr "Method for Computing the Radial Distribution of Emitters in a Cylindrical Source", J.Opt.Soc.Am 52 (1962) 885

[10] C.F.Hansen "Approximations for the Thermodynamic and Transport Properties of High-Temperature Air", NASA Tech.Rep. R-50, 1959

[11] P.W.Atkins "Physical Chemistry", Oxford University Press, 1984

[12] K.Muraoka, M.Hamamoto, M.Akazaki "Studies of an Impulse Spark Using Two-Wavelength Interferometry", Jpn.J.Appl.Phys., 19 (1980) L293

APPENDICES

A:1 Power as a function of time.
Inductive system.
Standard sparking plug.

A:2 Power as a function of time.
Inductive system.
Sharp edge electrode sparking plug.

A:3 Power as a function of time.
Commercial capacitive system.
Standard sparking plug.

A:4 Power as a function of time.
Commercial capacitive system.
Sharp edge electrode sparking plug.

A:5 Listing of the data processing program.

```
1*
2*            PROGRAM LMABELINV
3*      ***   i fil lm-abel-sparkinv ***
4*      ************************************************************************
5*      * MAIN PROGRAM                                                         *
6*      *                                                                      *
7*      * Input to the program is a set of experimental values from a one-     *
8*      * wavelength interferometry experiment. First a linear missalignement  *
9*      * is subtracted from the y-values and then the data are fitted to a    *
10*     * polynomial which is used in a spline-interpolation to yield smoothed *
11*     * y-values for equidistant x-values. The y-values are then input to the*
12*     * Abel-inversion routine which gives spatially resolved density- and   *
13*     * temperature-values for a nitrogen gas.                               *
14*     *                                                                      *
15*     * CALLING SEQUENCE :   CURVEFIT                                        *
16*     *                      ABELINVERSION                                   *
17*     *                      OUTPUTWRITE                                     *
18*     *                                                                      *
19*     * FILES USED :         (A-COMB)LM-ABEL-"fileid":IN                     *
20*     *                       The string "fileid" is supplied by the user.   *
21*     *                       It is assumed that an input-file by that name  *
22*     *                       exists where the necessary data is written.    *
23*     *                       The program creates an output-file (:UT) with  *
24*     *                       the same name.                                 *
25*     *                                                                      *
26*     *                                                                      *
27*     * Program written by Lars Martinsson, January 1987                     *
28*     * Program last modified by                                             *
29*     ************************************************************************
30*
31*          CHARACTER*5   FILEID            ! variable part of filename
32*          CHARACTER*30  INNAME            ! complete filename input-file
33*          CHARACTER*30  OUTNAME           ! complete filename output-file
34*          CHARACTER*28  TEXT              ! dummy-var to reach input data
35*          CHARACTER*10  INFORM(1:4)       ! experimental information
36*          REAL          PRESSURE          ! pressure in gas , Torr
37*          REAL          LABTEMP           ! temperature in gas ,Kelvin
38*          REAL          FRDIBA            ! fringe distance background mm
39*          REAL          MAGNIF            ! magnification factor
40*          INTEGER       NIN               ! number of input values ODD N:R
41*          INTEGER       NINMAX            ! maximum number of input values
42*
43*          PARAMETER(    NINMAX = 81 )     ! NOTE NINMAX MUST BE ODD
44*
45*          REAL          XIN(1:NINMAX)     ! x-values, read from input-file
46*          REAL          YIN(1:NINMAX)     ! y-values, read from input-file
47*
48*          INTEGER       NOUT              ! number of output values
49*          INTEGER       NOUTMAX           ! max length of output vector
50*
51*          PARAMETER(    NOUTMAX = 21 )    ! MAXIMUM ALLOWED 21
```

```
52*
53*           REAL          YSYM(1:NOUTMAX)   ! y-values from CURVEFIT
54*           REAL          LUNIT             ! dist between x-points
55*           REAL          AVN2CO            ! av. density N2
56*           REAL          AVNCO             ! av. density N
57*           REAL          AVTEMP            ! av. temp in spark
58*           REAL          N2CO(1:NOUTMAX)   ! density N2
59*           REAL          NCO(1:NOUTMAX)    ! density N
60*           REAL          TEMP(1:NOUTMAX)   ! temperature in spark
61*           INTEGER       I,K               ! DO-loop variables
62*
63*
64*     *** read file-name where input to program is stored
65*
66*           WRITE(*,100)'¤WHICH IMAGE DO YOU WANT TO PROCESS? :'
67*           READ(*,*)FILEID
68*           INNAME = '(A-COMB)LM-ABEL-' // FILEID // ':IN'
69*           OUTNAME = '(A-COMB)LM-ABEL-' // FILEID // ':UT'
70*
71*     *** read data from input-file
72*
73*           OPEN( 3 , FILE = INNAME , STATUS = 'OLD' , ACCESS = 'READ' )
74*
75*           DO 10 K = 1 , 4
76*               READ(3,200)TEXT,INFORM(K)
77*     10    CONTINUE
78*           READ(3,300)TEXT,PRESSURE
79*           READ(3,300)TEXT,LABTEMP
80*           READ(3,300)TEXT,FRDIBA
81*           READ(3,300)TEXT,MAGNIF
82*           READ(3,400)TEXT,NIN
83*
84*           IF( NIN . GT . NINMAX )THEN
85*               WRITE(*,*)' TOO MANY VALUES ON INPUTFILE!!'
86*               CLOSE(3)
87*               GOTO 9999
88*           ENDIF
89*
90*           DO 20 I = 1 , NIN
91*               READ(3,*) XIN(I),YIN(I)
92*     20    CONTINUE
93*
94*           CLOSE(3)
95*
96*     *** specify number of output values and call curvefit subroutine
97*
98*     30    WRITE(*,100)'¤HOW MANY OUTPUT VALUES DO YOU WANT? :'
99*           READ(*,*)NOUT
100*          IF( NOUT . GT . NOUTMAX )THEN
101*              WRITE(*,*)'YOU WANT TOO MANY, MAXIMUM 21 ALLOWED. TRY AGAIN!'
102*              GOTO 30
```

```
103*            ENDIF
104*
105*            CALL CURVEFIT( NIN,XIN,YIN,NOUT,YSYM,LUNIT )
106*
107*       *** call abel-inversion program
108*       *** LUNIT MUST BE DIVIDED BY 10 TO BE IN CM !!!!
109*
110*            LUNIT = LUNIT / 10.0
111*
112*            CALL ABELINVERSION( NOUT,YSYM,PRESSURE,LABTEMP,LUNIT,FRDIBA,
113*           &                     MAGNIF,AVN2CO,AVNCO,AVTEMP,N2CO,NCO,
114*           &                     TEMP)
115*
116*       *** call output writing subroutine
117*
118*            CALL OUTPUTWRITE( INFORM,PRESSURE,LABTEMP,LUNIT,AVN2CO,
119*           &                  AVNCO,AVTEMP,NOUT,N2CO,NCO,TEMP,
120*           &                  OUTNAME)
121*
122*
123*       9999  STOP
124*       100   FORMAT(//,A)
125*       200   FORMAT(2A)
126*       300   FORMAT(A,F11.5)
127*       400   FORMAT(A,I5,//)
128*
129*            END
```

```
130*
131*
132*            SUBROUTINE CURVEFIT(N,X,Y,NOUT,YSYM,STEP)
133*      ************************************************************************
134*      * SUBROUTINE CURVEFIT                                                   *
135*      *                                                                       *
136*      * DESCRIPTION:                                                         *
137*      * This subroutine controls the subroutines that fit a smoothed poly-   *
138*      * nomial with equidistant x-values to the experimental readings.       *
139*      *                                                                       *
140*      * CALLING SEQUENCE:    LINSUBT                                         *
141*      *                      SPLINEFIT                                       *
142*      *                      POLYFIT                                         *
143*      *                      SYMMETRIZE                                      *
144*      *                                                                       *
145*      * Program written by Lars Martinsson, March 1987                       *
146*      * Program last modified by                                             *
147*      ************************************************************************
148*
149*            INTEGER  N             ! number of inputvalues ARG
150*            REAL     X(1:N)        ! input x-values  ARG
151*            REAL     Y(1:N)        ! input y-values  ARG
152*            INTEGER  NOUT          ! number of outputvalues ARG
153*            INTEGER  NINT          ! intermediate NOUT value
154*            INTEGER  NMAX          ! maximum val of NINT
155*
156*            PARAMETER( NMAX = 41 )
157*
158*            REAL     XPF(1:NMAX)   ! output from POLYFIT
159*            REAL     YPF(1:NMAX)   ! output from POLYFIT
160*            REAL     STEP          ! dist. between adjacent x-points
161*            REAL     YSYM(1:NOUT)  ! output from SYMMETRIZE  ARG
162*
163*
164*
165*            NINT = 2 * NOUT - 1
166*
167*            IF( NINT .GT. NMAX )THEN
168*                WRITE(*,*)' NOUT TOO LARGE IN CURVEFIT CALL. '
169*            ENDIF
170*
171*            CALL LINBSUBT(N,X,Y)
172*
173*            CALL SPLINEFIT(N,X,Y,N)
174*
175*            CALL POLYFIT(N,X,Y,NINT,XPF,YPF,STEP)
176*
177*            CALL SYMMETRIZE(NINT,XPF,YPF,NOUT,YSYM)
178*
179*            RETURN
180*            END
```

```
181*
182*
183*            SUBROUTINE LINBSUBT(N,X,Y)
184*      ************************************************************************
185*      * SUBROUTINE LINBSUBT                                                  *
186*      *                                                                      *
187*      * DESCRIPTION:                                                         *
188*      * This subroutine subtracts a linear background ( y=kx+b ) from the    *
189*      * input y-vector and stores the result in the y-vector. The resulting  *
190*      * vectors are written to a file and a plotting routine is called.      *
191*      *                                                                      *
192*      * FILES USED:          (A-COMB)LM-ABEL-OUT0:DATA   OUTPUT              *
193*      *                                                                      *
194*      * SUBPROGRAMS CALLED:       PLOT     plot program                      *
195*      *                                                                      *
196*      * EXTERNAL SUBROUTINES:     none                                       *
197*      *                                                                      *
198*      * COMMON BLOCKS USED:    none                                          *
199*      *                                                                      *
200*      * Program written by Lars Martinsson November 1986                     *
201*      * Program last modified by Lars Martinsson March 1987                  *
202*      ************************************************************************
203*
204*            INTEGER   N          ! number of inputvalues  ARG
205*            REAL      X(1:N)     ! x-values input   ARG
206*            REAL      Y(1:N)     ! y-values input/output ARG
207*            REAL      X1,XN      ! value of X(I)
208*            REAL      Y1,YN      ! value of Y(I)
209*            REAL      SLOPE      ! slope of linear background
210*            INTEGER   I          ! DO-loop variable
211*            CHARACTER*16 TEXT    ! text in plot question
212*
213*
214*            X1 = X(1)
215*            XN = X(N)
216*            Y1 = Y(1)
217*            YN = Y(N)
218*            SLOPE = ( YN - Y1 ) / ( XN - X1 )
219*            DO 10 I = 1 , N
220*                Y(I) = SLOPE * ( X1 - X(I) ) + Y(I) - Y1
221*      10    CONTINUE
222*
223*      *** call plot of input data and write to a file
224*
225*            TEXT ='INPUT DATA'
226*
227*            CALL PLOT(N,X,Y,TEXT)
228*
229*            OPEN(3,FILE='(A-COMB)LM-ABEL-OUT0:DATA',STATUS='UNKNOWN',
230*           &     ACCESS='WRITE')
231*            WRITE(3,100)' INPUT DATA'
```

```
232*            WRITE(3,200)'X-KOORDINAT','Y-KOORDINAT'
233*            DO 20 I = 1 , N
234*                WRITE(3,300) X(I) , Y(I)
235*      20    CONTINUE
236*            CLOSE(3)
237*
238*            RETURN
239*      100   FORMAT(A)
240*      200   FORMAT(T5,A,TR5,A,//)
241*      300   FORMAT(T7,F8.4,TR8,F8.4)
242*            END
```

```
243*
244*
245*
246*           SUBROUTINE SPLINEFIT(N,X,Y,NOUT)
247*     ************************************************************************
248*     * SUBROUTINE SPLINEFIT                                                 *
249*     *                                                                      *
250*     * DESCRIPTION:                                                        *
251*     * Input to the subroutine is a set of (x,y) values which are used for  *
252*     * interpolation to yield the y-values for NOUT equidistant x-values    *
253*     * by using cubic spline-functions. The method is described in Dahlquist*
254*     * Bjorck: Numerical Methods, Prentice-Hall 1974, pp.131-134 and pp.166-*
255*     * 167. Note : The method assumes that the second derivates at the end- *
256*     * points are equal to zero, and the x-values must be written in in-    *
257*     * creasing order.                                                      *
258*     * This version uses the input vectors as output vectors!               *
259*     *                                                                      *
260*     * FILES USED :          (A-COMB)LM-ABEL-OUT1:DATA       OUTPUT         *
261*     *                                                                      *
262*     * SUBPROGRAMD CALLED:             PLOT          plot program           *
263*     *                                                                      *
264*     * EXTERNAL SUBROUTINES:          none                                  *
265*     *                                                                      *
266*     * COMMON BLOCKS USED:      none                                        *
267*     *                                                                      *
268*     * Program written by Lars Martinsson, November 1986                    *
269*     * Program last modified by                                             *
270*     ************************************************************************
271*
272*           INTEGER      N              ! number of input values          ARG
273*           REAL         X(1:N)         ! x-values, read from input-file  ARG
274*           REAL         Y(1:N)         ! y-values, read from input-file  ARG
275*           INTEGER      NMAX           ! maximum size of internal vectors
276*
277*           PARAMETER(   NMAX = 99 )    ! must be .GE. max of N
278*
279*           REAL         XSTEP(2:NMAX)  ! dist between X(I) and X(I-1)
280*           REAL         YSTEP(2:NMAX)  ! dist between Y(I) and Y(I-1)
281*           REAL         X1,X2,X3       ! value of X(I)
282*           REAL         H1,H2          ! = XSTEP(I)
283*           REAL         Y1,Y2,Y3       ! value of Y(I)
284*           REAL         D1,D2          ! = YSTEP(I)
285*           REAL         ALFAV(2:NMAX)  ! variables in solv eq. syst
286*           REAL         ALFA           ! variables in solv eq. syst
287*           REAL         BETA           ! variables in solv eq. syst
288*           REAL         CV(2:NMAX-1)   ! variables in solv eq. syst
289*           REAL         C              ! variables in solv eq. syst
290*           REAL         GV(2:NMAX),G   ! variables in solv eq. syst
291*           REAL         KV(2:NMAX),K   ! endpoint of equation solution
292*           REAL         STEP           ! dist between XSF(I) and XSF(I-1)
293*           INTEGER      NOUT           ! prescribed n:r of output values  ARG
```

```
294*            INTEGER      NSFM           ! max of output values from SF
295*
296*            PARAMETER(   NSFM = 99 )
297*
298*            REAL         XSF(1:NSFM)    ! equidistant x-values to splin-f
299*            REAL         YSF(1:NSFM)    ! spline-fitted y-values
300*            INTEGER      I,L            ! DO-loop variables
301*            REAL         T              ! var used in calc of new values
302*            CHARACTER*16 TEXT           ! text in plot question
303*
304*
305*      *** calculation of spline-fitted curve in two steps:
306*      ***  1. calculate coefficients to a tridiagonal equation system
307*      ***     which is solved by forward-backward substitution
308*      ***  2. the solution to the equation-syst is used as coefficients
309*      ***     in a polynomial by which the new y-values are calculated
310*      ***  the variable names used are the same as in the ref.
311*
312*      *** initiation and calculation of elements in the tridiagonal system
313*      *** concurrent with forward substitution
314*
315*
316*            IF( N . GT . NMAX )THEN
317*                WRITE(*,*)' TOO MANY INPUT VALUES TO SPLINEFIT. '
318*            ENDIF
319*
320*
321*            X1 = X(1)
322*            X2 = X(2)
323*            Y1 = Y(1)
324*            Y2 = Y(2)
325*            H1 = X2-X1
326*            XSTEP(2) = H1
327*            D1 = ( Y2 - Y1 ) / H1
328*            YSTEP(2) = D1
329*            ALFA = 2.0
330*            ALFAV(2) = ALFA
331*            C = 1.0
332*            CV(2) = C
333*            G = 3.0 * D1
334*            GV(2) = G
335*
336*            DO 10 I = 3 , N - 1
337*                X3 = X(I)
338*                Y3 = Y(I)
339*                H2 = X3 - X2
340*                D2 = ( Y3 - Y2 ) / H2
341*                BETA = H2 / ALFA
342*                ALFA = 2.0 * ( H1 + H2 ) - BETA * C
343*                G = 3.0 * ( H1 * D2 + H2 *D1 ) - BETA * G
344*                C = H1
```

```
345*                ALFAV(I) = ALFA
346*                GV(I) = G
347*                CV(I) = C
348*                XSTEP(I) = H2
349*                YSTEP(I) = D2
350*                H1 = H2
351*                D1 = D2
352*                X2 = X3
353*                Y2 = Y3
354*       10     CONTINUE
355*
356*              X3 = X(N)
357*              Y3 = Y(N)
358*              H2 = X3 - X2
359*              D2 = ( Y3 - Y2 ) / H2
360*
361*       *** initiate and start backward substitution
362*
363*              BETA = 1.0 / ALFA
364*              ALFAV(N) = 2.0 - BETA * C
365*              GV(N) = 3.0 * D2 - BETA * G
366*              XSTEP(N) = H2
367*              YSTEP(N) = D2
368*
369*              K = GV(N) / ALFAV(N)
370*              KV(N) = K
371*              DO 20 I = N - 1 , 2 ,-1
372*                  K = ( GV(I) - CV(I) * K ) / ALFAV(I)
373*                  KV(I) = K
374*       20     CONTINUE
375*
376*       *** calculate the spline-fitted x,y values
377*
378*              X1 = X(1)
379*              STEP = ( X(N) - X1 ) / ( NOUT - 1 )
380*
381*              XSF(1) = X1
382*              DO 30 I = 2 , NOUT - 1
383*                  XSF(I) = ( I-1.0 ) * STEP + X1
384*       30     CONTINUE
385*              XSF(NOUT) = X(N)
386*
387*              L = 2
388*              YSF(1) = Y(1)
389*              YSF(NOUT) = Y(N)
390*              DO 40 I = 2 , NOUT - 1
391*       50         IF( XSF(I) . GT . X(L) )THEN
392*                      L = L + 1
393*                      GOTO 50
394*                  ENDIF
395*                  T = ( XSF(I) - X(L-1) ) / XSTEP(L)
```

```
396*                YSF(I) = T * Y(L) + (1.0 - T) * Y(L-1) + XSTEP(L) *
397*           &            T * (1.0 - T) * ( ( KV(L-1) - YSTEP(L) ) *
398*           &            ( 1.0 - T ) - ( KV(L) - YSTEP(L) ) * T )
399*       40    CONTINUE
400*
401*
402*       *** call plot of spline-fitted data, write the values to a file
403*
404*
405*             TEXT = 'SPLINE-FITTED'
406*
407*             CALL PLOT(NOUT,XSF,YSF,TEXT)
408*
409*             OPEN(4,FILE='(A-COMB)LM-ABEL-OUT1:DATA',STATUS='UNKNOWN',
410*            &     ACCESS='WRITE')
411*             WRITE(4,100)' SPLINE-FITTED DATA'
412*             WRITE(4,200)'X-KOORDINAT','Y-KOORDINAT'
413*             DO 60 I = 1 , NOUT
414*                 WRITE(4,300) XSF(I) , YSF(I)
415*       60    CONTINUE
416*             CLOSE(4)
417*
418*       *** output from program is stored in input vectors
419*
420*             DO 70 I = 1,N
421*                 X(I) = XSF(I)
422*                 Y(I) = YSF(I)
423*       70    CONTINUE
424*
425*             RETURN
426*       100   FORMAT(A)
427*       200   FORMAT(T5,A,TR5,A,//)
428*       300   FORMAT(T7,F8.4,TR8,F8.4)
429*             END
```

```
430*
431*
432*
433*
434*           SUBROUTINE POLYFIT(NIN,X,Y,NOUT,XPF,YPF,STEP)
435*     ************************************************************************
436*     * SUBROUTINE POLYFIT                                                    *
437*     *                                                                       *
438*     * DESCRIPTION:                                                         *
439*     * The program fits a polynom to a set of (x,y) values, input and output*
440*     * both have equidistant x-values. The polynom are calculated by a      *
441*     * series expansion of ( orthonomal ) Chebyshev polynomials up to order *
442*     * ORDER - 1. The value of the polynomial is suplied by a subroutine    *
443*     * CHEBPOL, and the polynomial-fitted values are calculated by a sub-   *
444*     * routine CHEBEXP. NOUT values are generated and the distance between  *
445*     * them is STEP. The endpoint y-values are set to zero in the program.  *
446*     * The method is described in for example:                              *
447*     *                        Handbook of Mathematical Functions,           *
448*     *                         Abramowitz & Stegun, Dover p 791             *
449*     *                        Statistics and Experimental Design , vol I,   *
450*     *                         Johnson & Leone, Wiley pp 423-436            *
451*     *                                                                       *
452*     *                                                                       *
453*     * FILES USED:        (A-COMB)LM-ABEL-OUT2:DATA   output                *
454*     *                                                                       *
455*     * SUBROUTINES CALLED :      PLOT     plotprogram                      *
456*     *                           CHEBPOL  the subroutine calculates the    *
457*     *                                    value of Chebyshev polynomials   *
458*     *                           CHEBEXP  the subroutine calculates the    *
459*     *                                    value of n-term exp in Cheb. pol.*
460*     *                                                                       *
461*     * EXTERNAL SUBROUTINES:   none                                        *
462*     *                                                                       *
463*     * COMMON BLOCKS USED:    none                                         *
464*     *                                                                       *
465*     * Program written by Lars Martinsson April 1987                       *
466*     * Program last modified by                                            *
467*     ************************************************************************
468*
469*           INTEGER    NIN             ! number of input values ARG
470*           REAL       X(1:NIN)        ! x-values input  ARG
471*           REAL       Y(1:NIN)        ! y-values input  ARG
472*           INTEGER    NOUT            ! number of output values  ARG
473*           REAL       ARG             ! arg x-value transformed
474*           INTEGER    ORDER           ! order or polynom to be fitted
475*
476*           PARAMETER( ORDER = 8 )
477*
478*           REAL       VAL(0:ORDER)    ! result of eval. of polynomials
479*           REAL       VALL            ! spec. value of VAL(L)
480*           REAL       NUM(0:ORDER)    ! numerator in coeff
```

```
481*            REAL        DEN(0:ORDER)     ! denominator in coeff
482*            INTEGER     K,L              ! DO-loop variables
483*            REAL        X1               ! =X(1)
484*            REAL        DIST             ! dist between ARG in exp. calc.
485*            REAL        STEP             ! equidistance between x-values OUTARG
486*            REAL        COEFF(0:ORDER)   ! coefficient vector
487*            REAL        XPF(1:NOUT)      ! x-vector polyfit output
488*            REAL        YPF(1:NOUT)      ! y-vector polyfit output
489*            CHARACTER*16  TEXT           ! text in plot question
490*
491*      *** write warning if there are too few points to perform a fit
492*
493*            IF( NIN . LT . ORDER + 3 )THEN
494*                WRITE(*,*)' WARNING ! YOU HAVE TOO FEW INPUT VALUES TO GET A',
495*           &                ' GOOD SMOOTHING. '
496*            ENDIF
497*
498*      *** calculate expansion coefficients
499*
500*            DO 10 K = 1 , NIN
501*                ARG = K - 1.0
502*                CALL CHEBPOL( NIN , ORDER , ARG , VAL )
503*                DO 20 L = 0 , ORDER
504*                    VALL = VAL(L)
505*                    NUM(L) = NUM(L) + VALL * Y(K)
506*                    VALL = VALL * VALL
507*                    DEN(L) = DEN(L) + VALL
508*      20        CONTINUE
509*      10    CONTINUE
510*
511*            DO 30 K = 0 , ORDER
512*                COEFF(K) = NUM(K) / DEN(K)
513*      30    CONTINUE
514*
515*      *** calculate fitted polynomial
516*
517*            X1 = X(1)
518*
519*            DIST = ( NIN - 1.0 ) / ( NOUT - 1.0 )
520*
521*            STEP = ( X(NIN) - X1 ) / ( NOUT - 1.0 )
522*
523*            DO 40 K = 1 , NOUT
524*                ARG = ( K - 1.0 ) * DIST
525*                XPF(K) = X1 + ( K - 1.0 ) * STEP
526*                CALL CHEBEXP( NIN,ORDER,ARG,COEFF,YPF(K) )
527*      40    CONTINUE
528*
529*      *** set y-values at end-points to zero
530*
531*            YPF(1) = 0.0
```

```
532*            YPF(NOUT) = 0.0
533*
534*      *** call plot of input data and write to a file
535*
536*            TEXT = 'POLYFIT DATA'
537*
538*            CALL PLOT(NOUT,XPF,YPF,TEXT)
539*
540*            OPEN(3,FILE='(A-COMB)LM-ABEL-OUT2:DATA',STATUS='UNKNOWN',
541*           &     ACCESS='WRITE')
542*            WRITE(3,100)' POLYFIT DATA'
543*            WRITE(3,200)'X-KOORDINAT','Y-KOORDINAT'
544*            DO 50 K = 1 , NOUT
545*                WRITE(3,300) XPF(K) , YPF(K)
546*      50    CONTINUE
547*            CLOSE(3)
548*
549*            RETURN
550*      100   FORMAT(A)
551*      200   FORMAT(T5,A,TR5,A,//)
552*      300   FORMAT(T7,F8.4,TR8,F8.4)
553*            END
```

```
554*
555*
556*           SUBROUTINE CHEBPOL( N,ORD,X,Y)
557*      ************************************************************************
558*      * SUBROUTINE CHEBPOL                                                    *
559*      *                                                                       *
560*      * DESCRIPTION:                                                         *
561*      * This subroutine calculates the values of Chebyshev polynomials for   *
562*      * an argument value ARG and orders 0 up to ORD. The polynomials are    *
563*      * orthogonal for discrete integer values. Evaluation is based on a re- *
564*      * currence relation for Chebyshev polynomials ( on p. 791 in ref.).    *
565*      * Starting values are       Cheb(0,ARG) = 1.0                          *
566*      *                  and     Cheb(1,ARG) = 1.0 - 2.0 * ARG / ( N - 1 )  *
567*      * where N is the number of points to be fitted. The output is stored in*
568*      * a vector Y(0:ORD).                                                    *
569*      * Reference: Handbook of Mathematical Functions, Abramowitz & Stegun   *
570*      *                                                                       *
571*      * FILES USED:   none                                                   *
572*      *                                                                       *
573*      * SUBROUTINES CALLED:  none                                            *
574*      *                                                                       *
575*      * EXTERNAL SUBROUTINES:  none                                          *
576*      *                                                                       *
577*      * COMMON BLOCKS USED:  none                                            *
578*      *                                                                       *
579*      * Program written by Lars Martinsson, April 1987                       *
580*      * Program last modified by                                             *
581*      ************************************************************************
582*
583*
584*           INTEGER    N         ! number of points to be fitted  ARG
585*           INTEGER    ORD       ! highest order of polynomial   ARG
586*           REAL       X         ! argument to Cheb. pol.        ARG
587*           REAL       Y(0:ORD)  ! result   OUTPUT
588*           REAL       Y0,Y1,Y2  ! spec val. of Y
589*           INTEGER    K         ! DO-loop var.
590*
591*
592*           IF( ORD . LT . 1 ) RETURN
593*
594*      *** initiation
595*
596*           Y0 = 1.0
597*           Y1 = 1.0 - ( 2.0 * X ) / ( N - 1.0 )
598*           Y(0) = Y0
599*           Y(1) = Y1
600*
601*      *** iteration
602*
603*           DO 10 K = 1 , ORD - 1
604*               Y2 = ( ( 2.0 * K + 1.0 ) * ( N - 1.0 - 2.0 * X ) * Y1 - K *
```

```
605*           &           ( N + K ) * Y0 ) / ( ( K + 1.0 ) * ( N - K - 1.0 ) )
606*                Y( K + 1 ) = Y2
607*                Y0 = Y1
608*                Y1 = Y2
609*      10     CONTINUE
610*
611*            RETURN
612*            END
```

```
613*
614*
615*
616*            SUBROUTINE CHEBEXP( N,ORD,X,COEFF,RES)
617*      ************************************************************************
618*      * SUBROUTINE CHEBEXP                                                   *
619*      *                                                                      *
620*      * DESCRIPTION:                                                         *
621*      * This subroutine computes the value of an ORD-term expansion in Cheb- *
622*      * yshev polynomials with coefficient vector COEFF(0:ORD) and argument  *
623*      * ARG. The output Y = SUM COEFF(I)*Cheb(I,ARG), summation over I from  *
624*      * 0 to ORD. The Chebyshev polynomials, orthogonal for discrete integer *
625*      * arguments, are calculated by a recurrence equation , p. 791 in Hand- *
626*      * book of Mathematical Functions, Ed. by Abramowitz and Stegun.         *
627*      *                                                                      *
628*      * FILES USED:  none                                                    *
629*      *                                                                      *
630*      * SUBROUTINES CALLED:  none                                            *
631*      *                                                                      *
632*      * EXTERNAL SUBROUTINES: none                                           *
633*      *                                                                      *
634*      * COMMON BLOCKS USED: none                                             *
635*      *                                                                      *
636*      * Program written by Lars Martinsson, April 1987                       *
637*      * Program last modified by                                             *
638*      ************************************************************************
639*
640*
641*
642*            INTEGER    N             ! number of points to be fitted  ARG
643*            INTEGER    ORD           ! highest order of polynomial    ARG
644*            REAL       X             ! argument to pol.               ARG
645*            REAL       COEFF(0:ORD)  ! coeff in exp. in Cheb. pol     ARG
646*            REAL       RES           ! result of calc.       OUTPUT   ARG
647*            REAL       C0,C1         ! spec. val. of coeff.
648*            REAL       Y0,Y1,Y2      ! val. of Cheb. pol.
649*            INTEGER    K             ! DO-loop var.
650*
651*
652*            IF( ORD . LT . 1 ) RETURN
653*
654*      *** initiation
655*
656*            Y0 = 1.0
657*            Y1 = 1.0 - ( 2.0 * X ) / ( N - 1.0 )
658*            C0 = COEFF(0)
659*            C1 = COEFF(1)
660*
661*            RES = C0 * Y0 + C1 * Y1
662*
663*      *** iteration
```

```
664*
665*            DO 10 K = 1 , ORD - 1
666*                Y2 = ( ( 2.0 * K + 1.0 ) * ( N - 1.0 - 2.0 * X ) * Y1 - K *
667*           &           ( N + K ) * Y0 ) / ( ( K + 1.0 ) * ( N - K - 1.0 ) )
668*                C0 = COEFF( K + 1 )
669*                RES = RES + C0 * Y2
670*                Y0 = Y1
671*                Y1 = Y2
672*      10    CONTINUE
673*
674*            RETURN
675*            END
```

```
676*
677*
678*
679*           SUBROUTINE SYMMETRIZE(NIN,XIN,YIN,NOUT,YSYM)
680*     ************************************************************************
681*     * SUBROUTINE SYMMETRIZE                                                 *
682*     *                                                                       *
683*     * DESCRIPTION :                                                        *
684*     * This program takes the average of symmetrical y-values and stores    *
685*     * this value in YSYM(I). Equidistant x-values are required as input    *
686*     * to the program.                                                       *
687*     *                                                                       *
688*     * FILES USED :             (A-COMB)LM-ABEL-OUT3:DATA  output           *
689*     *                                                                       *
690*     * SUBPROGRAMS CALLED :              PLOT     plotprogram               *
691*     *                                                                       *
692*     * EXTERNAL SUBROUTINES: none                                           *
693*     *                                                                       *
694*     * COMMON BLOCKS USED:  none                                            *
695*     *                                                                       *
696*     * Program written by Lars Martinsson, November 1986                    *
697*     * Program last modified by Lars Martinsson, April 1987                 *
698*     ************************************************************************
699*
700*
701*           INTEGER    NIN              ! number of input values  INPUT
702*           REAL       YIN(1:NIN)       ! y-values  INPUT
703*           REAL       XIN(1:NIN)       ! r-values  INPUT
704*           INTEGER    NOUT             ! number of values to output-file ARG
705*           INTEGER    NOUTMAX          ! max. val of NOUT
706*
707*           PARAMETER( NOUTMAX = 21 )
708*
709*           REAL       XSYM(1:NOUTMAX)  ! symmetrized r-values
710*           REAL       YSYM(1:NOUT)     ! symmetrized y-values OUTPUT
711*           INTEGER    K                ! DO-loop variable
712*           CHARACTER*16  TEXT          ! text in plot question
713*
714*
715*
716*     *** calculate mean-value of y-values placed symmetric about the middle
717*     *** point
718*
719*           YSYM(1) = YIN(NOUT)
720*           XSYM(1) = XIN(NOUT)
721*           DO 10 K = 2 , NOUT
722*               XSYM(K) = XIN( NOUT - 1 + K )
723*               YSYM(K) = ( YIN(NOUT - 1 + K) + YIN(NOUT + 1 - K) ) / 2.0
724*     10    CONTINUE
725*
726*     *** call plot of symmetrsized data
```

```
*
*           TEXT = 'SYMMETRIC DATA'
*
*           CALL PLOT(NOUT,XSYM,YSYM,TEXT)
*
*     *** write output to a file
*
*           OPEN( 5 , FILE ='(A-COMB)LM-ABEL-OUT3:DATA', STATUS = 'UNKNOWN'
*          &                 , ACCESS = 'WRITE' )
*
*           WRITE(5,100)' SYMMETRISERADE DATA'
*           WRITE(5,200)'X-KOORDINAT','Y-KOORDINAT'
*           DO 20 K = 1 , NOUT
*               WRITE(5,300) XSYM(K) , YSYM(K)
*     20    CONTINUE
*           CLOSE(5)
*
*           RETURN
*     100   FORMAT(A)
*     200   FORMAT(T5,A,TR5,A,//)
*     300   FORMAT(T7,F8.4,TR8,F8.4)
*           END
```

```
749*
750*
751*
752*
753*
754*           SUBROUTINE ABELINVERSION(NMAX,Y,PRES,LABTEMP,LUNIT,FRDIBA,MAGNIF,
755*          &                         AVN2CO,AVNCO,AVTEMP,N2CO,NCO,TEMP)
756*     ************************************************************************
757*     * SUBROUTINE ABELINVERSION                                              *
758*     *                                                                       *
759*     * DESCRIPTION :                                                         *
760*     * This subroutine performs Abelinversion on a set of experimental one- *
761*     * wavelength interferometry data with equidistant x-values. The inputs *
762*     * are Y (mm), PRES (torr), LABTEMP (K), LUNIT (cm), FRDIBA (mm), MAGNIF*
763*     * (-) and NMAX. The inversion matrix and other necessary input data    *
764*     * are read from datafiles. The program calculates the molecular number *
765*     * number density (cm-3) and the temperature (K) from the relative      *
766*     * fringe shifts in a cylindrically symmetric spark. The temperature is *
767*     * calculated under the assumption of isobaric conditions in the spark. *
768*     * The temperature and density are then used as input in a calculation  *
769*     * of the densities of atoms and molecules and the associated tempera-  *
770*     * ture if thermal equilibrium conditions prevails.                     *
771*     *                                                                       *
772*     * REFERENCE: Barr, Journal of the Optical Society of America, 52       *
773*     *                                          (1962), 8, 885             *
774*     *                                                                       *
775*     *                                                                       *
776*     * FILES USED :  (A-COMB)LM-ABEL-MATRIS:DATA  inversion matrix          *
777*     *                                            from ref Barr, except     *
778*     *                                            opposite sign and a factor*
779*     *                                            10E4 times less           *
780*     *                                                                      *
781*     *               (A-COMB)LM-ABEL-IN:DATA      wavelength in micrometer, *
782*     *                                            molecular refractivity in *
783*     *                                            cm3 and (refrN2 / refrN)  *
784*     *                                                                      *
785*     * SUBPROGRAMS CALLED :  EQTEMP             equilib calc                *
786*     *                                                                      *
787*     * EXTERNAL SUBPROGRAMS : none                                          *
788*     *                                                                      *
789*     * ND-MONITOR CALLS USED:  none                                         *
790*     *                                                                      *
791*     * COMMON BLOCKS USED    :  none                                        *
792*     *                                                                      *
793*     * Program written by Peter Grafstrom                                   *
794*     * Program last modified by Lars Martinsson, October 1986               *
795*     ************************************************************************
796*
797*
798*           INTEGER          NMAX              ! number of output values
799*           INTEGER          I,K               ! DO-loop variables
```

```
800*            REAL            AMATRIS(1:21,1:21) ! inversion matrix
801*            REAL            LAMBDA             ! wavelength micometers
802*            REAL            REFRAC             ! spec. refractivity cm**3
803*            REAL            RELREF             ! rel refrac N to N2
804*            REAL            AVDEN              ! factor in calc average val
805*            REAL            FRDIBA             ! fringe dist background
806*            REAL            LUNIT              ! length unit r-direction cm
807*            REAL            MAGNIF             ! magnification factor
808*
809*            REAL            Y(1:NMAX)          ! y-values,same unit as FRDIBA
810*            REAL            PI                 ! pi
811*
812*            PARAMETER(      PI = 3.1415926 )
813*
814*            REAL            CONST1             ! prop const abelinv - density
815*            REAL            CONST2             ! prop const pressure-refdens
816*            REAL            CONST3             ! prop const density-temp
817*
818*            PARAMETER(      CONST2 = 9.6570E18 ,
819*           &                CONST3 = 9.6570E18 )
820*
821*            REAL            RELFS(1:21)        ! relative fringe shifts
822*            REAL            SUM                ! sumation variable
823*            REAL            AVRELFS            ! average fringe shift
824*            REAL            PRES               ! pressure in torr in gas
825*            REAL            LABTEMP            ! temp in Kelvin in gas
826*            REAL            NUMDEN0            ! ref number density cm-3
827*            REAL            ESAVNUDE           ! est av number density cm-3
828*            REAL            ESAVTEMP           ! estimate of temperature
829*            REAL            AVTEMP             ! equilib. av. temp
830*            REAL            AVN2CO             ! av.num.den. N2 cm-3
831*            REAL            AVNCO              ! av.num.den. N  cm-3
832*            INTEGER         ALTN               ! alt start of sumation
833*            REAL            AINVFS(1:21)       ! Abel inv fringe shifts
834*            REAL            ESNUDE             ! est. of number density cm-3
835*            REAL            ESTEMP             ! estimate of temperature
836*            REAL            TEMP(1:NMAX)       ! equili. temperature
837*            REAL            N2CO(1:NMAX)       ! equili. N2 numb. dens.
838*            REAL            NCO(1:NMAX)        ! equili. N numb. dens.
839*
840*
841*
842*      *** read inversion matrix
843*
844*             OPEN(6,FILE='(ATOM-COMB)LM-ABEL-MATRIS:DATA',STATUS='OLD',
845*            &             ACCESS='READ')
846*
847*             DO 10 I = 1 , 21
848*                 READ(6,250,ERR=999) ( AMATRIS(I,K) , K = 1 , 21 )
849*      10     CONTINUE
850*
```

```
851*            CLOSE(6)
852*      250   FORMAT(F7.0,20F6.0)
853*
854*      *** read input-data to abelinversion
855*
856*            OPEN(7,FILE='(ATOM-COMB)LM-ABEL-IN:DATA',STATUS='OLD',
857*           &             ACCESS='READ')
858*            READ(7,*) LAMBDA , REFRAC , RELREF
859*            CLOSE(7)
860*
861*      *** calculate prop factor and relative fringe shifts
862*
863*            LUNIT = LUNIT * MAGNIF
864*
865*            CONST1 = - 1.0E-8 * LAMBDA / REFRAC / PI / LUNIT
866*
867*            DO 20 I = 1 , NMAX
868*                RELFS(I) = Y(I) / FRDIBA
869*      20    CONTINUE
870*
871*      *** calculate reference number density
872*
873*            NUMDEN0 = CONST2 * PRES / LABTEMP
874*
875*      *** calculate estimate of mean number density of molecules and
876*      *** temperature
877*
878*            SUM = 0.0
879*            DO 30 I = 2 , NMAX
880*                SUM = SUM + RELFS(I)
881*      30    CONTINUE
882*            AVRELFS = RELFS(1) + 2.0 * SUM
883*            AVDEN = - 2.0E4 / ( NMAX * ( 2 * NMAX + 1 ) )
884*            ESAVNUDE = AVDEN * CONST1 * AVRELFS + NUMDEN0
885*            ESAVTEMP = CONST3 * PRES / ESAVNUDE
886*
887*      *** calculate equlibrium average-temperature and -composition
888*
889*            IF( ESAVNUDE . GT . 0.0 )THEN
890*                CALL EQTEMP( ESAVNUDE,ESAVTEMP,RELREF,AVTEMP,AVN2CO,AVNCO )
891*            ENDIF
892*
893*      *** calculate spatially resolved estimate of number density of atoms
894*      *** and estimate of temperature and equilibrium composition and eq.
895*      *** temperature
896*
897*            DO 40 I = 1 , NMAX
898*                ALTN = I - 2
899*                IF( ALTN . LT . 1 )THEN
900*                    ALTN = 1
901*                ENDIF
```

```
902*                SUM = 0.0
903*                DO 50 K = ALTN , NMAX
904*                    SUM = SUM + AMATRIS(K,I) * RELFS(K)
905*      50        CONTINUE
906*                AINVFS(I) = SUM
907*      40    CONTINUE
908*            DO 60 I = 1 , NMAX
909*                ESNUDE = CONST1 * AINVFS(I) + NUMDEN0
910*                ESTEMP = CONST3 * PRES / ESNUDE
911*                IF( ESNUDE . GT . 0.0 . AND . ESTEMP . LT . 40000.0 )THEN
912*                    CALL EQTEMP( ESNUDE,ESTEMP,RELREF,TEMP(I),N2CO(I),NCO(I) )
913*                ENDIF
914*      60    CONTINUE
915*
916*            RETURN
917*
918*      999   WRITE(*,*)' ERROR WHEN READING INVERSION MATRIX!!!!!!!!!!!!!!!!'
919*            END
```

```
920*
921*
922*
923*
924*
925*           SUBROUTINE EQTEMP( N2CONC,TEMP,REFQ,OUTTEMP,N2NEW,NNEW )
926*     ************************************************************************
927*     * SUBROUTINE EQTEMPQ                                                    *
928*     *                                                                       *
929*     * DESCRIPTION:                                                         *
930*     * This subroutine calculates the molecule and atom concentrations      *
931*     * (cm-3) and temperature (Kelvin) from a first estimate of temperature *
932*     * and molecule concentration from an Abelinversion. The quantities are *
933*     * calculated under the assumption of thermal equlibrium. The subprogram*
934*     * EPS is specific for nitrogen!                                         *
935*     *                                                                       *
936*     * REFERENCE: Hansen, NASA Technical Report R-50, 1959                   *
937*     *                                                                       *
938*     * DATA-FILES USED: none                                                 *
939*     *                                                                       *
940*     * SUBPROGRAMS CALLED:  EPS                                              *
941*     *                                                                       *
942*     * EXTERNAL SUBPROGRAMS: none                                            *
943*     *                                                                       *
944*     * ND-MONITOR CALLS USED: none                                           *
945*     *                                                                       *
946*     * COMMON BLOCKS USED: none                                              *
947*     *                                                                       *
948*     * Program written by Lars Martinsson, January 1987                      *
949*     * Program last modified by Lars Martinsson, April 1987                  *
950*     ************************************************************************
951*
952*           REAL TEMP           ! temperature Kelvin            INARG
953*           REAL EPSILON        ! fraction of N2 dissoc. to N
954*           REAL EPS            ! SUBFCN calculates EPSILON
955*           REAL ALFA           ! conc N2 / conc N
956*           REAL ATEMP          ! calc temp from eq. realations
957*           REAL N2CONC         ! abel inv, conc of N2              INARG
958*           REAL CONST          ! prop. const between conc and temp
959*
960*           PARAMETER( CONST = 7.2432E21 )
961*
962*           REAL FACTOR         ! temperature of pure N2
963*           REAL DELTATEMP      ! diff between temperatures
964*           REAL NEWTEMP        ! new approx temperature
965*           REAL OUTTEMP        ! output temperature           OUTARG
966*           REAL REFQ           ! refractivity N / refr N2     INARG
967*           REAL TEMPACC        ! accuracy in temp iteration
968*
969*           PARAMETER( TEMPACC = 5.0 )
970*
```

```
971*            REAL N2NEW          ! calculated N2 conc   OUTARG
972*            REAL NNEW           ! calculated N conc    OUTARG
973*
974*
975*            FACTOR = CONST / N2CONC
976*
977*      *** temperature iteration
978*
979*      10    EPSILON = EPS( TEMP )
980*            ALFA = 2.0 * EPSILON / ( 1.0 - EPSILON )
981*            ATEMP = FACTOR * ( 1.0 + REFQ * ALFA ) / ( 1 + ALFA )
982*            DELTATEMP = ( ATEMP - TEMP ) / 4.0
983*            NEWTEMP = TEMP + DELTATEMP
984*
985*            IF ( ABS( DELTATEMP ) . GT . TEMPACC ) THEN
986*                TEMP = NEWTEMP
987*                GOTO 10
988*            ENDIF
989*
990*      *** calculate outparameters
991*
992*            OUTTEMP = TEMP
993*            N2NEW = N2CONC / ( 1.0 + REFQ * ALFA )
994*            NNEW = ALFA * N2NEW
995*
996*
997*            RETURN
998*            END
```

```
 999*
1000*
1001*           REAL FUNCTION EPS(TEMP)
1002*      ************************************************************************
1003*      * SUBROUTINE  REAL FUNCTION EPS                                        *
1004*      *                                                                      *
1005*      * DESCRIPTION:                                                         *
1006*      * The program calculates the equilibrium coefficient for N2-2N reaction*
1007*      * as a function of temperature and pressure. From this EPS, the        *
1008*      * fraction of N2 which dissociates to N, is calculated.                *
1009*      *                                                                      *
1010*      * REFERENCE: Hansen, NASA Technical Report R-50, 1959                  *
1011*      *                                                                      *
1012*      * DATA-FILES USED:  none                                               *
1013*      *                                                                      *
1014*      * SUBPROGRAMS CALLED:  REAL FUNCTION LOGQPN    log of partition fcn N  *
1015*      *                      REAL FUNCTION LOGQPN2   log of partition fcn N2 *
1016*      *                                                                      *
1017*      * EXTERNAL SUBPROGRAMS:  none                                          *
1018*      *                                                                      *
1019*      * ND-MONITOR CALLS USED:  none                                         *
1020*      *                                                                      *
1021*      * COMMON BLOCKS USED:  none                                            *
1022*      *                                                                      *
1023*      * Program written by Lars Martinsson, January 1987                     *
1024*      * Program last modified by                                             *
1025*      ************************************************************************
1026*
1027*           REAL P          ! pressure in atm
1028*
1029*           PARAMETER( P = 1.0 )
1030*
1031*           REAL TEMP       ! temperature in Kelvin
1032*           REAL LOGKP      ! log of eq. const
1033*           REAL KP         ! eq. const
1034*           REAL LOGQPN     ! log of parti fcn N  FUNCSUB
1035*           REAL LOGQPN2    ! log of parti fcn N2  FUNCSUB
1036*
1037*
1038*
1039*           LOGKP = - 113200. / TEMP + 2.0 * LOGQPN(TEMP) - LOGQPN2(TEMP)
1040*           KP = EXP( LOGKP )
1041*
1042*           IF( KP . GT . 0.0 ) THEN
1043*               EPS = 1.0 / SQRT( 1.0 + 4.0 * P / KP )
1044*           ELSE
1045*               EPS = 0.0
1046*           ENDIF
1047*
1048*           RETURN
1049*           END
```

```
1050*
1051*
1052*
1053*
1054*           REAL FUNCTION LOGQPN(T)
1055*     ************************************************************************
1056*     * SUBROUTINE  REAL FUNCTION LOGQPN                                     *
1057*     *                                                                      *
1058*     * DESCRIPTION:                                                         *
1059*     * This function subroutine calculates the partition function for N for *
1060*     * temperatures below 15000 K.                                          *
1061*     *                                                                      *
1062*     * REFERENCE: Hansen, NASA Technical Report R-50, 1959                  *
1063*     *                                                                      *
1064*     * DATA-FILES USED:  none                                               *
1065*     *                                                                      *
1066*     * SUBPROGRAMS CALLED:  none                                            *
1067*     *                                                                      *
1068*     * EXTERNAL SUBPROGRAMS:  none                                          *
1069*     *                                                                      *
1070*     * ND-MONITOR CALLS USED:  none                                         *
1071*     *                                                                      *
1072*     * COMMON BLOCKS USED:  none                                            *
1073*     *                                                                      *
1074*     * Program written by Lars Martinsson, January 1987                     *
1075*     * Program last modified by                                             *
1076*     ************************************************************************
1077*
1078*
1079*           REAL T           ! temperature Kelvin  ARG
1080*
1081*           LOGQPN = 2.5 * LOG( T ) + 0.30 + LOG( 4.0 + 10.0 * EXP( -
1082*          &         27700. / T ) + 6.0 * EXP( - 41500. / T ) )
1083*
1084*           RETURN
1085*           END
```

```
1086*
1087*
1088*
1089*           REAL FUNCTION LOGQPN2(T)
1090*      ************************************************************************
1091*      * SUBROUTINE  REAL FUNCTION LOGQPN2                                     *
1092*      *                                                                       *
1093*      * DESCRIPTION:                                                          *
1094*      * This function subroutine calculates the partition function for N2 for*
1095*      * temperatures below 15000 K.                                           *
1096*      *                                                                       *
1097*      * REFERENCE: Hansen, NASA Technical Report R-50, 1959                   *
1098*      *                                                                       *
1099*      * DATA-FILES USED: none                                                 *
1100*      *                                                                       *
1101*      * SUBPROGRAMS CALLED:  none                                             *
1102*      *                                                                       *
1103*      * EXTERNAL SUBPROGRAMS:  none                                           *
1104*      *                                                                       *
1105*      * ND-MONITOR CALLS USED:  none                                          *
1106*      *                                                                       *
1107*      * COMMON BLOCKS USED: none                                              *
1108*      *                                                                       *
1109*      * Program written by Lars Martinsson, January 1987                      *
1110*      * Program last modified by                                              *
1111*      ************************************************************************
1112*
1113*           REAL T          ! temperature Kelvin  ARG
1114*
1115*           LOGQPN2 = 3.5 * LOG( T ) - 0.42 - LOG( 1.0 - EXP( - 3390.
1116*          &            / T ) )
1117*
1118*           RETURN
1119*           END
```

```
1120*
1121*
1122*
1123*           SUBROUTINE OUTPUTWRITE( INFORM,PRES,LABTEMP,LUNIT,AVN2CO
1124*          &                ,AVNCO,AVTEMP,NMAX,N2CO,NCO,TEMP,FILENAME)
1125*
1126*     ************************************************************************
1127*     * SUBROUTINE OUTPUTWRITE                                               *
1128*     *                                                                      *
1129*     * DESCRIPTION:                                                         *
1130*     * The output data are written to the terminal and optionally to a file *
1131*     * with the same name as the data input file, except that the file type *
1132*     * is changed to :UT . A plotting routine is also called.                *
1133*     *                                                                      *
1134*     * REFERENCE: none                                                      *
1135*     *                                                                      *
1136*     * DATA-FILES USED: (A-COMB)LM-ABEL-"fileid":UT                         *
1137*     *                                                                      *
1138*     * SUBPROGRAMS CALLED: PLOT      plotting program                       *
1139*     *                                                                      *
1140*     * EXTERNAL SUBPROGRAMS: none                                           *
1141*     *                                                                      *
1142*     * ND-MONITOR CALLS USED:  CLOCK          Get current time and date     *
1143*     *                                        from operating system         *
1144*     *                                                                      *
1145*     * COMMON BLOCKS USED: none                                             *
1146*     *                                                                      *
1147*     * Program written by Lars Martinsson, May 1987                         *
1148*     * Program last modified by                                             *
1149*     ************************************************************************
1150*
1151*
1152*           CHARACTER*10     INFORM(1:4)       ! Experiment inform
1153*           REAL             PRES              ! pressure in torr in gas
1154*           REAL             LABTEMP           ! temp in Kelvin in gas
1155*           REAL             LUNIT             ! length unit r-direction cm
1156*           REAL             AVN2CO            ! av.num.den. N2 cm-3
1157*           REAL             AVNCO             ! av.num.den. N  cm-3
1158*           REAL             AVTEMP            ! equilib. av. temp
1159*           INTEGER          NMAX              ! number of y-values on input
1160*           REAL             N2CO(1:NMAX)      ! equili. N2 numb. dens.
1161*           REAL             NCO(1:NMAX)       ! equili. N numb. dens.
1162*           REAL             TEMP(1:NMAX)      ! equili. temperature
1163*           CHARACTER*30     FILENAME          ! Total name output file
1164*           CHARACTER*9      MONTH(1:12)       ! name of months DATA
1165*           REAL             MINCONC           ! output minconc parameter
1166*           REAL             R(1:21)           ! r-vector
1167*           CHARACTER        ANSW              ! answer to Y/N question
1168*           CHARACTER*16     TEXT              ! text in plot quoestion
1169*           INTEGER          PARAMS(1:7)       ! output from CLOCK,time date
1170*           INTEGER          I                 ! DO-loop variable
```

```
1171*
1172*
1173*           DATA MONTH/'  JANUARY',' FEBRUARY','    MARCH','    APRIL',
1174*          &           '      MAY','     JUNE',     JULY','   AUGUST',
1175*          &           'SEPTEMBER','  OCTOBER',' NOVEMBER',' DECEMBER'/
1176*
1177*      *** set small concentration values equal to zero
1178*
1179*           MINCONC = 1.E5
1180*
1181*           IF( AVN2CO . LT . MINCONC ) AVN2CO = 0.0
1182*           IF( AVNCO . LT . MINCONC ) AVNCO = 0.0
1183*
1184*           DO 10 I = 1 , NMAX
1185*               IF( N2CO(I) . LT . MINCONC ) N2CO(I) = 0.0
1186*               IF( NCO(I) . LT . MINCONC ) NCO(I) = 0.0
1187*      10   CONTINUE
1188*
1189*
1190*      *** write output to the terminal
1191*
1192*           WRITE(*,*)'  IMAGE NUMBER :',INFORM(1)
1193*           WRITE(*,*)'  SYSTEM       :',INFORM(2),' SPARKPLUGG :',INFORM(3)
1194*           WRITE(*,*)'  TIME :',INFORM(4),'MICROSECONDS'
1195*           WRITE(*,900)'  PRESSURE :',PRES,' TORR','LABTEMP : ',LABTEMP,'K'
1196*
1197*           WRITE(*,200)'  MEAN TEMPERATURE :',AVTEMP,' KELVIN'
1198*           WRITE(*,*)'  MEAN NUMBER DENSITY IN  CM-3'
1199*           WRITE(*,300)'  MOLECULES : ', AVN2CO ,'ATOMS : ',AVNCO
1200*           WRITE(*,*)'   SPATIALLY RESOLVED DATA '
1201*           WRITE(*,400)' RADIAL DIST','TEMPERATURE','MOLECULES','ATOMS'
1202*           WRITE(*,500)'CM','KELVIN','  CM-3','  CM-3'
1203*           DO 20 I = 1 , NMAX
1204*               R(I) = ( I - 1.0 ) * LUNIT
1205*               WRITE(*,600) R(I) , TEMP(I) , N2CO(I) , NCO(I)
1206*
1207*               IF( MOD(I,3) . EQ. 0 )THEN
1208*                   WRITE(*,*)
1209*               ENDIF
1210*
1211*      20   CONTINUE
1212*
1213*      *** call plot of temperature
1214*
1215*           TEXT = 'TEMPERATURE'
1216*
1217*           CALL PLOT(NMAX,R,TEMP,TEXT)
1218*
1219*      *** write output-data to specified file
1220*
1221*           WRITE(*,100)'¤DO YOU WANT TO STORE OUTPUT DATA? Y/N:'
```

```
1222*            READ(*,*)ANSW
1223*            IF( ANSW . EQ . 'Y' . OR . ANSW . EQ . 'y' )THEN
1224*
1225*                WRITE(*,800)' WRITING TO FILE ',FILENAME
1226*
1227*                CALL CLOCK(PARAMS)
1228*                I = PARAMS(6)
1229*
1230*                OPEN(8,FILE=FILENAME,STATUS='UNKNOWN',ACCESS='WRITE')
1231*
1232*                WRITE(8,700) 'WRITTEN : ',PARAMS(4),PARAMS(3),PARAMS(2),
1233*           &                  MONTH(I),PARAMS(5),PARAMS(7)
1234*                WRITE(8,*)'  IMAGE NUMBER :',INFORM(1)
1235*                WRITE(8,*)'  SYSTEM       :',INFORM(2),' SPARKPLUGG :',
1236*           &                  INFORM(3)
1237*                WRITE(8,*)'  TIME :',INFORM(4),'MICROSECONDS'
1238*                WRITE(8,900)'  PRESSURE :',PRES,' TORR','LABTEMP : '
1239*           &                  ,LABTEMP,'K'
1240*
1241*
1242*                WRITE(8,200)'  MEAN TEMPERATURE :',AVTEMP,' KELVIN'
1243*                WRITE(8,*)'  MEAN NUMBER DENSITY IN  CM-3'
1244*                WRITE(8,300)'  MOLECULES : ', AVN2CO ,'ATOMS : ',AVNCO
1245*                WRITE(8,*)'   SPATIALLY RESOLVED DATA '
1246*                WRITE(8,400)' RADIAL DIST','TEMPERATURE','MOLECULES','ATOMS'
1247*                WRITE(8,500)'CM','KELVIN','  CM-3','  CM-3'
1248*                DO 30 I = 1 , NMAX
1249*                    R(I) = ( I - 1.0 ) * LUNIT
1250*                    WRITE(8,600) R(I) , TEMP(I) , N2CO(I) , NCO(I)
1251*
1252*                    IF( MOD(I,3) . EQ . 0 )THEN
1253*                        WRITE(8,*)
1254*                    ENDIF
1255*
1256*       30       CONTINUE
1257*
1258*                CLOSE(8)
1259*
1260*            ENDIF
1261*
1262*            RETURN
1263*       100  FORMAT(A)
1264*       200  FORMAT(A,F6.0,TR1,A/)
1265*       300  FORMAT(A,1PE12.5,TR3,A,E12.5///)
1266*       400  FORMAT(T3,A,TR4,A,TR4,A,TR10,A)
1267*       500  FORMAT(T8,A,TR11,A,TR7,A,TR12,A/)
1268*       600  FORMAT(T3,1PE12.5,0P,TR5,F7.0,1P,2(TR5,E12.5))
1269*       700  FORMAT(' ',A,I2,'.',I2,'.',I2,'  ',A9,' ',I2,' ',I4//)
1270*       800  FORMAT(T2,2A)
1271*       900  FORMAT(A,F5.1,TR1,A,TR3,A,F5.1,TR1,A//)
1272*            END
```

```
1273*
1274*
1275*
1276*
1277*           SUBROUTINE PLOT(N,X,Y,QUEST)
1278*      ************************************************************************
1279*      * This program calls the plot subroutine AUTOPLOT ( written by Lars    *
1280*      * Gramstad ) which runs on the ND-computer Bertil at Dep. of Physics   *
1281*      * in Lund. The subroutine NOTGRAPH must be used when the PC-PERFECT    *
1282*      * emulator is used.                                                    *
1283*      *                                                                      *
1284*      *         External subroutines:                                        *
1285*      *                      (A-LG)AUTOPLOT    plotprogram                 *
1286*      *                      (UTILITY)TCS      Tectronix subroutines       *
1287*      *                      (A-LM)LM-NOTGRAPH clears screen for PC-PERFECT *
1288*      *                                        emulator                    *
1289*      *                                                                      *
1290*      * Program written by Lars Martinsson, February 1987                   *
1291*      * Program last modified by                                            *
1292*      ************************************************************************
1293*
1294*           INTEGER       N          ! number of points to plot
1295*           REAL          X(1:N)     ! x-value in plot
1296*           REAL          Y(1:N)     ! y-value in plot
1297*           CHARACTER*16  QUEST      ! text in question
1298*           CHARACTER     ANSW       ! answer to Y/N question
1299*
1300*
1301*           WRITE(*,100) '¤DO YOU WANT TO PLOT '// QUEST //'? Y/N: '
1302*           READ(*,*)ANSW
1303*           IF( ANSW . EQ . 'Y' . OR . ANSW . EQ . 'y' )THEN
1304*               CALL AUTOPLOT(X,Y,N,0,0,0,0,0)
1305*               CALL NOTGRAPH
1306*           ENDIF
1307*
1308*           RETURN
1309*      100  FORMAT(A)
1310*           END
```

0.633 1.10E-23 0.63

Image number :28-25
System :BOSCH
Sparkplugg :SPETSELEKTROD
Time (microseconds) :2000.0
Pressure (Torr) :727.2
Temperature (Kelvin) :293.0
Fringe dist background (mm):0.1392
Magnification factor :1.5943
Number of input-values :19

Input-values in mm!!

X	Y
10.76	-59.912
10.82	-59.923
10.875	-59.931
10.935	-59.938
10.995	-59.942
11.055	-59.949
11.11	-59.953
11.17	-59.960
11.23	-59.961
11.29	-59.961
11.345	-59.964
11.405	-59.971
11.465	-59.971
11.52	-59.960
11.58	-59.945
11.64	-59.936
11.70	-59.929
11.755	-59.916
11.815	-59.909

Image number :
System :
Sparkplugg :
Time (microseconds) :
Pressure (Torr) :
Temperature (Kelvin) :
Fringe dist background (mm):
Magnification factor :
Number of input-values :
Input-values in mm!!
X Y

-2029.	-1831.	-1239.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
-4439.	-4041.	-2847.	-1324.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
-1791.	-1778.	-1740.	-1936.	-1172.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
5111.	4342.	2034.	-928.	-1523.	-1036.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
1298.	1299.	1304.	964.	-722.	-1291.	-930.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
95.	204.	531.	969.	538.	-635.	-1140.	-847.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
237.	262.	338.	501.	810.	330.	-586.	-1032.	-781.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
193.	203.	235.	292.	464.	714.	214.	-551.	-951.	-727.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
152.	157.	173.	202.	260.	431.	646.	142.	-524.	-886.	-682.	0.	0.	0.	0.	0.	0.	0.	0.	0.
122.	125.	134.	150.	179.	236.	403.	595.	95.	-501.	-834.	-644.	0.	0.	0.	0.	0.	0.	0.	0.
99.	101.	107.	117.	133.	162.	217.	380.	555.	63.	-482.	-790.	-612.	0.	0.	0.	0.	0.	0.	0.
82.	84.	87.	94.	104.	120.	148.	202.	359.	522.	39.	-465.	-752.	-583.	0.	0.	0.	0.	0.	0.
69.	70.	73.	77.	84.	94.	111.	138.	189.	342.	495.	22.	-450.	-720.	-558.	0.	0.	0.	0.	0.
59.	60.	61.	65.	69.	76.	87.	103.	129.	179.	326.	471.	9.	-437.	-691.	-536.	0.	0.	0.	0.
51.	51.	53.	55.	58.	63.	70.	81.	96.	122.	170.	313.	450.	-1.	-424.	-666.	-517.	0.	0.	0.
44.	45.	46.	47.	50.	53.	58.	65.	75.	91.	115.	162.	301.	432.	-9.	-413.	-643.	-499.	0.	0.
39.	39.	40.	41.	43.	46.	49.	54.	61.	71.	86.	110.	155.	290.	416.	-15.	-402.	-623.	-483.	0.
35.	35.	35.	36.	38.	40.	43.	46.	51.	58.	67.	82.	105.	149.	280.	402.	-20.	-393.	-604.	-468.
31.	31.	32.	32.	33.	35.	37.	40.	43.	48.	55.	64.	78.	101.	144.	271.	389.	-24.	-384.	-587.
28.	28.	28.	29.	30.	31.	33.	35.	37.	41.	46.	52.	61.	75.	97.	139.	263.	377.	-28.	-375.
25.	25.	25.	26.	27.	28.	29.	31.	33.	35.	39.	44.	50.	59.	72.	94.	135.	256.	367.	-30.