Huygens data calibration report

G. Kargl[1], W. Macher[1], A. J. Ball[2], and M. Towner[2]

$\boxed{163}$

March 2005

This report is part of the Huygens SSP flight data calibration.

[1] Space Research Institute, Austrian Academy of Sciences, Schmidlstr. 6, A-8042 Graz, Austria

[2] Planetary and Space Sciences Research Institute, The Open University, Walton Hall, Milton Keynes, MK7 6AA, United Kingdom

Purpose of the document

The purpose of the document is to report on the data calibration procedure for the Huygens SSP ACC-E and ACC-I sensors. It includes a brief introduction to filter theory and the removal of the influence of a conditioning electronics to sensor data.

The sources and type of noise and other spurious signals will be pointed out as well as strategies how to treat them accordingly.

In short it is meant as a kind of data correction handbook for the calibration of specifically SSP flight data, though the treatment of laboratory data might benefit from the algorithms described here as well.

TABLE OF CONTENTS

1 ACC-E ---------- 5

1.1 Sensor description ---------- 5
1.1.1 Electronic schematics ---------- 5
1.1.2 Filter description ---------- 6
Filter Parameters ---------- 6
Bode diagram ---------- 10
Step response ---------- 11
Impulse response ---------- 12

1.2 Data ---------- 13
1.2.1 Raw data ---------- 13
1.2.2 Discussion of spurious signals ---------- 14
Noise ---------- 14
Aliasing ---------- 15
Smoothing ---------- 15
1.2.3 Filter removal ---------- 15
Convolution and deconvolution ---------- 15
Restoration of pre-filter data ---------- 16
1.2.4 Voltage to Force conversion ---------- 16
1.2.5 Discussion ---------- 17

2 ACC-I ---------- 18

2.1 Sensor description ---------- 18
2.1.1 Electronic schematics ---------- 18
2.1.2 Filter description ---------- 19
Anti-aliasing Filter Parameters ---------- 19
Bode diagram ---------- 20
Step response ---------- 21
Impulse response ---------- 22
High pass filter ---------- 22
Step response ---------- 24
Impulse response ---------- 24

2.2 Data ---------- 26
2.2.1 Raw data ---------- 26
2.2.2 Smoothing discussion ---------- 26
2.2.3 Filter removal ---------- 27
Restoration of pre-filter data ---------- 27
2.2.4 Voltage to Acceleration conversion ---------- 28

3 REFERENCES ---------- 29

4 PROGRAM LISTINGS ---------- 30

4.1 ACC-E Flight data processing ---------- 30

4.2 ACC-I flight data processing ---------- 35

4.3 Subroutines -- 42
4.3.1 Get component values: getcomp --- 42
4.3.2 Low-Pass impulse response: irp --- 42
4.3.3 Low-pass filter function: alp2 -- 43
4.3.4 Bode plot graphic package: ibodeplot --- 43
4.3.5 High-pass impulse response: hpirp--- 45
4.3.6 Averaging routine: daver -- 45

1 ACC-E

1.1 Sensor description

The ACC-E sensor of the Huygens Surface Science Package (SSP) is a piezoelectric force transducer mounted on a pylon in front of the Huygens entry probe. It measures the resulting force during the penetration of Titans surface at the moment of the landing impact.(Zarnecki et al., 1997, Lorenz, 1994).

The actual sensor is a disc of piezoelectric ceramic PZT-5A with 14 mm diameter, mounted between a hemispherical nut and a 5 cm pylon. The nominal force range is from 5 to 2000 N which are recorded at a sampling rate of 10 kHz.

PZT's generate a charge proportional to the applied force, this charge is converted into a voltage by a charge amplifier. Subsequently the signal is pre-amplified by a pseudo-logarithmic amplifier with three different linear gain segments depending on the input voltage and prepared for AD conversion with an anti-aliasing filter.

1.1.1 Electronic schematics

A previous simulation showed that the charge amplifier and pseudo-log amplifier are non critical with respect to a frequency dependent gain correction and are therefore omitted for the moment. Their contributions will be regarded as simple linear relationship between output voltage and applied force. As the flight data indicated that only the lowest gain segment of the pseudo-logarithmic amplifier was used this should be a safe assumption for the moment.

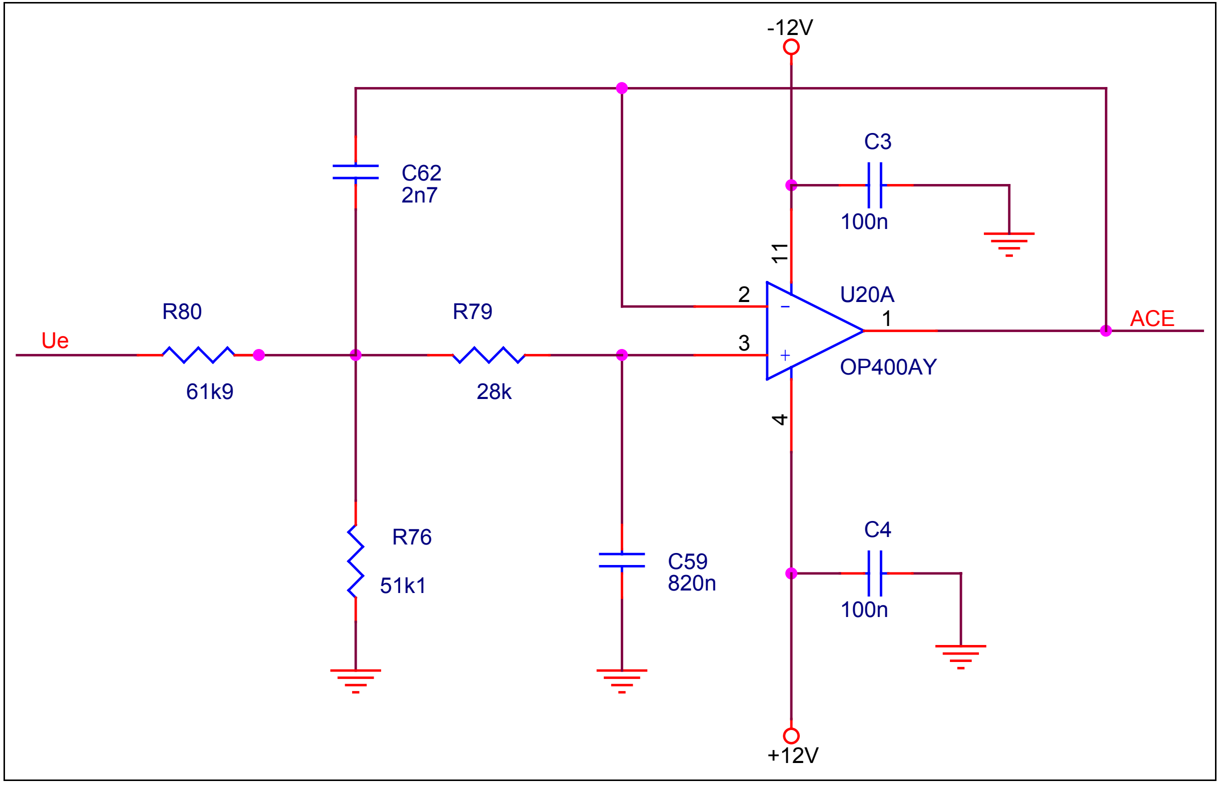

Figure. 1 ACC-E filter schematics

The anti-aliasing filter used for ACC-E is a low-pass filter of second order with single feed-back commonly known as Sallen-Key type. In this specific case the filter has in deviation to the standard scheme, an additional voltage divider in the input line. Thus the internal gain of the filter is unity and the total filter gain is controlled by the voltage divider $R_{80}//R_{76}$.

1.1.2 Filter description

Filter Parameters

The following section describes the filter parameters derived from the schematics. These parameters are essential for the characterization and will be used for the restoration of the pre-filter data.

From the sampling theory follows that only frequencies smaller than half of the sampling frequency $\mathbf{f_s}$ can be resolved

$$f_{Ny} = \frac{f_s}{2} \qquad \text{(Eq. 1)}$$

Frequencies larger than $\mathbf{f}_{Ny}$ (known as Nyquist frequency) will be mirrored at the Nyquist frequency and map themselves onto lower frequencies. Since the higher part of the spectrum is not known, this is an irreversible process and can spoil sampled data up to a total uselessness. Therefore, the optimal corner frequency of an anti-aliasing filter has carefully to be chosen such as to attenuate higher frequency components well before $\mathbf{f}_{Ny}$ by some order of magnitudes (in db[1]). Actually, the corner frequency of a low-pass filter is defined as the frequency where the filter gain is at -3dB from the nominal filter gain $\mathbf{A_0}$.

For a proper filter description two out of three parameter sets are necessary:

1. The sampling frequency $\mathbf{f_s}$ and the filter corner or cut-off frequency $\mathbf{f}_c$.
2. The filter type and shape factors a_i and b_i to be used in the filter polynomial equation.
3. The electronic components values[2] constituting the physical filter on the PCB.

Usually the design of a anti-aliasing filter starts, after getting the corner frequency, with the decision which type of filter to implement (e.g. Low - Pass Butterworth or Tschebyscheff etc.), which defines the filter shape via the polynomial factors $\mathbf{a_i}$ and $\mathbf{b_i}$. Since the type of application for anti-aliasing filters naturally requires the usage of a low-pass filter this will mostly not be explicitly mentioned in the following.

For the further design of the filter an appropriate gain and corner frequency should be chosen. For a second order low-pass filters the complex transfer function is:

$$H(P) = \frac{A_0}{1+a_1P+b_1P^2} \qquad \text{(Eq. 2)}$$

[1] db → $Y=20 \log_{10}(A/A_0)$

[2] For filters implemented in hardware components values can only be chosen from a discrete number of available parts values. The range of available capacitors is usually more restricted than that of resistors.

with P being the normalized frequency.

$$P = \mathbf{i}\,\frac{f}{f_c} \qquad \text{(Eq. 3)}$$

With a selected filter gain **A_0** and a corner frequency **f_c** the electrical components for the implemented filter follows for an Sallen-Key design implemented for ACC-E:

Given **f_c** and choosing **C_1** the condition

$$\frac{C_2}{C_1} \geq \frac{4b_1}{a_1^2} \qquad \text{(Eq. 4)}$$

must be fulfilled to get real numbers for the resistors. The resistors of the circuit can be computed as

$$R_{1,2} = \frac{a_1C_2 \pm \sqrt{,a_1^2C_2^2 - 4C_1b_1(1-A_0)}}{4\pi f_c C_1 C_2} \qquad \text{(Eq. 5)}$$

The above definition is only valid if the internal filter gain **G_i=1**, which is fulfilled for the used Sallen-Key design of ACC-E.

Internal gain:

$$G_i = +\frac{R_f}{R_g} = 1 \qquad \text{(Eq. 6)}$$

External gain:

$$G_e = \frac{1}{R_{80} + R_{76}} \qquad \text{(Eq. 7)}$$

Total filter gain

$$A_0 = G_i \cdot G_e \qquad \text{(Eq. 8)}$$

Electronic components

Given the components values for the ACC-E filter with (components labels are taken from the ACC-E flight electronics design schematics) (Fig. 1):

R80 =61.9 kΩ;

R76 =51.1 kΩ;

R79 =28.0 kΩ;

Resistors for internal gain not present in Fig. 1 but added for consistency with the above definition formulas.

Rf=0 Ω;

Rg=Inf;

Capacitors

C59=820 pF;

C62=2.7 nF;

Sampling frequency

f_s=10 kHz;

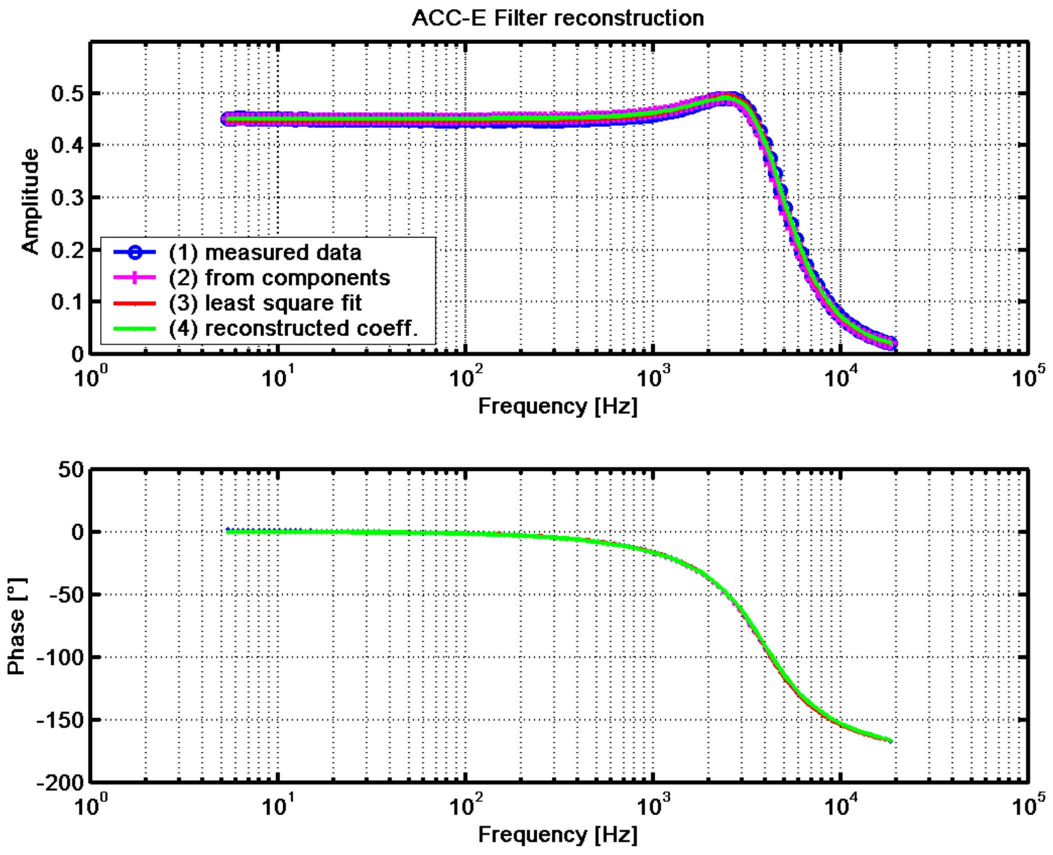

Figure. 2. ACC-E amplitude and phase filter curves computed from all filter reconstruction methods described below.

Corner frequency

The filter corner frequency is due to a lack of documentation only available by circuit simulation and breadboard measurements. However, for a proper reconstruction **f_c** is essential to know. The overall design of the ACC-E filter corresponds to a Tschebyscheff low pass type with a corrugation factor of about 0.7 db.

From measurements on a laboratory breadboard the corner frequency can be derived as.

$f_{c\_breadborad} = 4738.7065$ Hz

The filter coefficients were computed also with a least square fit on the measured data.

$f_{c\_fit} = 4823.7743$ Hz

$a_{1\_fit} = 1.3369$

$b_{1\_fit} = 1.5532$

$A_{0\_fit} = 0.4483$

For a Tschebyscheff filter with a corrugation of 0.7 db the filter factors can be reconstructed via an approximation formula and interpolation in between the tabulated values for 0.5 and 1 db corrugation filters (Tietze and Schenk, 1993).

$f_{c\_reconstr} = 4733.4231$ Hz

$a_{1\_reconstr} = 1.3332$

$b_{1\_reconstr} = 1.4622$

$A_{0\_reconstr} = 0.4522$

From the electronic components and an assumption for **f_c** from the reconstruction formulas all coefficients an be computed from the standard formulas for a single feedback low-pass filter.

$$a_1 = \omega_c \left(R_{12}\, C_5\, (1 - G_i) + C_4{*}(R_3{+}R_{12})\right) \qquad \text{(Eq. 9)}$$

$$b_1 = \omega_c^2\, R_{12}\, R_3\, C_4\, C_5 \qquad \text{(Eq. 10)}$$

Utilizing, Eqs. 9 and 10 results in the component derived filter parameters.

$f_{c\_comp} = 4733.4231$ Hz

$a_{1\_comp} = 1.3655$

$b_{1\_comp} = 1.5349$

$A_{0\_comp} = 0.4522$

From those parameter sets above the set derived from the approximation reconstruction was chosen as standard set for further data treatment. The reason for this choice was a good agreement with the measured filter data in the region of the corner frequency.

Chosen parameter set

With this parameter set the complex filter curve, the impulse response and the step response were computed, where the impulse response was used to remove the filter influence from the ACC-E penetration force data.

$f_c \;= 4733.4$ Hz

$a_1 \;= 1.3332$

$b_1 \;= 1.4622$

$A_0 \;= 0.4522$

Since the Nyquist frequency of the conditioning electronics is at 5 kHz, a corner frequency of roughly 4.7 kHz means that the filter is not exactly in optimum configuration with regard to the task to fulfil.

Bode diagram

The general behaviour of a complex filter curve in the frequency domain also called the "transfer function" can be demonstrated at best in the form of the so called Bode-diagram showing the absolute magnitude and the phase shift over the frequency range.

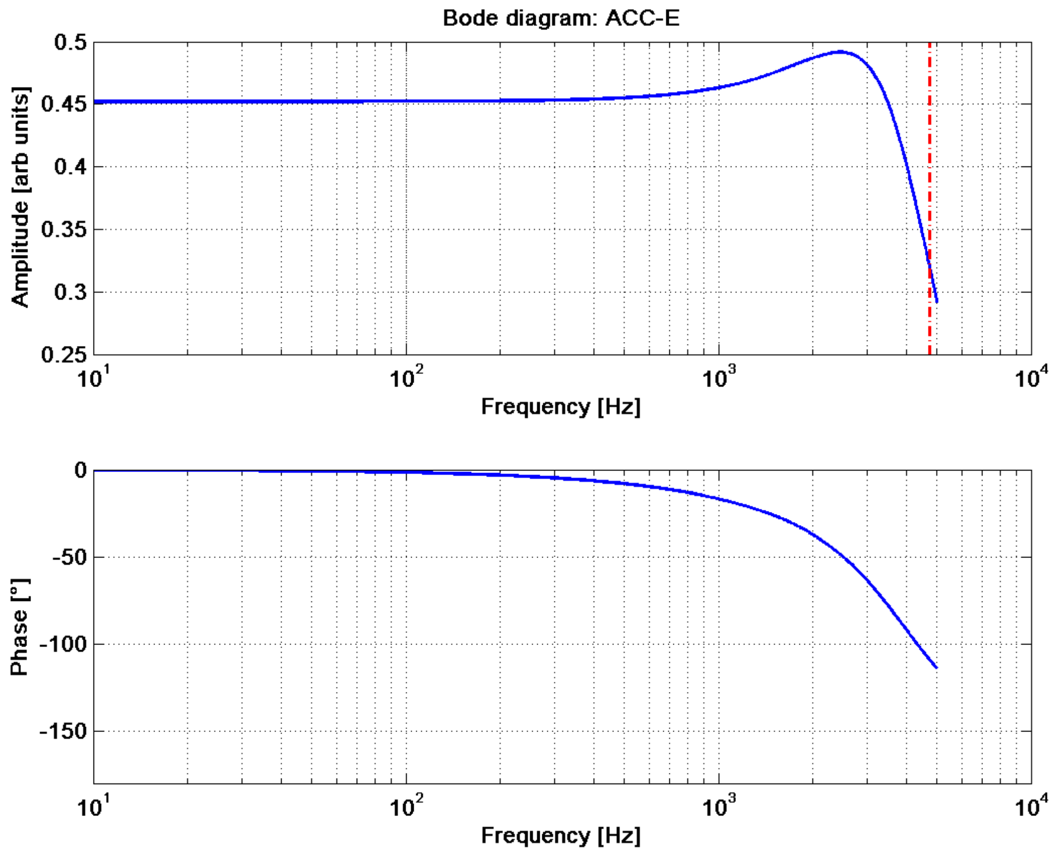

Figure. 3. Bode diagram of the chosen ACC-E parameter set. The red dashed curve indicates the location of the corner frequency which is at-3db or at A0/√2.

Step response

The output signal after the application of a step signal i.e. the steep rise or also fall from a low level up to a new signal level is called the filter step response. It is a measure of the transfer quality for rectangular wave signals. Since the typical signal from a dynamic penetrometer is essential a step function, the knowledge of the filter step response is important for the further interpretation of the sensor signal.

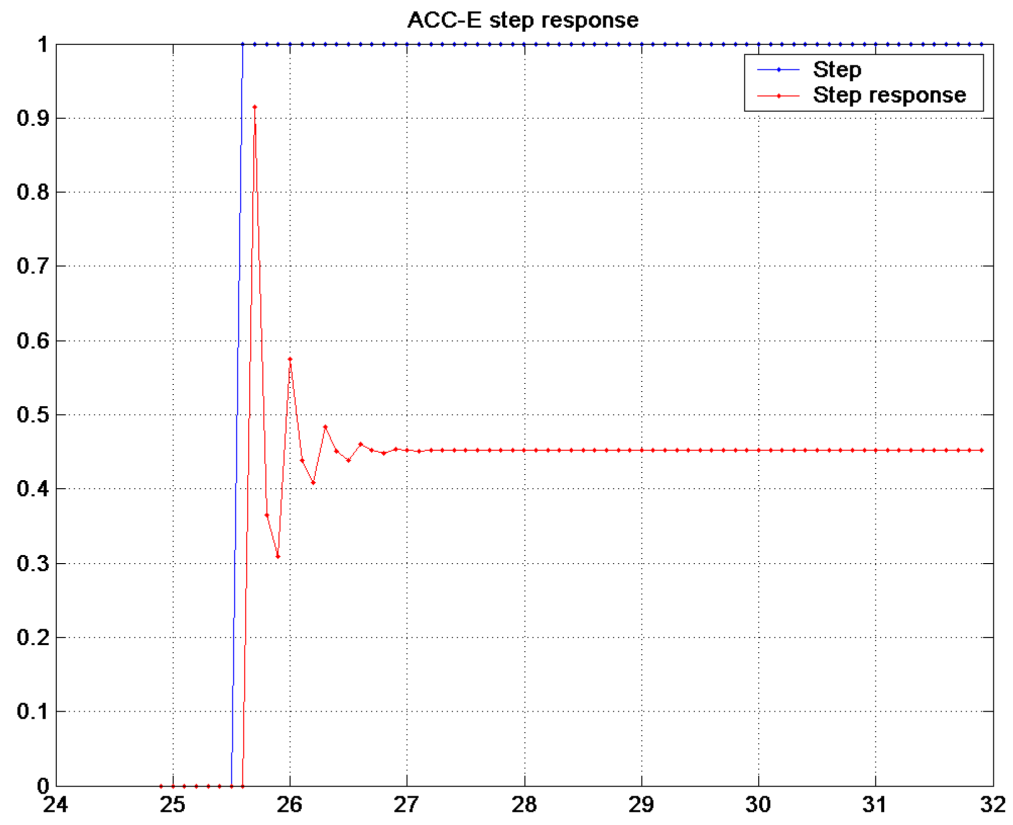

Figure. 4. The step response of the ACC-E anti-aliasing low pas filter with a filter gain of $A_0 = 0.4522$. The used filter shows a particular high level of ringing after the initial signal rise.

It should be noted that the use of a Tschebyscheff filter for signals mostly representing a step function has some particular drawbacks, even if the attenuation of higher frequency components in the signal is the most steep of all commonly used filter types.

The first and most prominent to recognize is the gain rise close to the corner frequency $\mathbf{f_c}$ which makes it necessary to apply a more sophisticated filter correction procedure. However, the major disadvantage is that such steep filter designs cause a severe ringing in the signal at the occurrence of steep flanks i.e. impulses and steps as opposed to less rippling filters as e.g. a Bessel type filter, most of the time a Butterworth filter is used as a compromise between fast attenuation and minimal rippling in the frequency plane.

The discussion above refers mostly to filters of second order, but remains valid for higher orders as well, with the distinction that corrugated filters increase the ripple amplitudes and numbers with the filter order. However, an e.g. 4th order Butterworth can give a similar fast attenuation with less ripples than a 2nd order Tschebyscheff with a high corrugation level.

Impulse response

The impulse response of a second order low-pass filter can be analytically described, and derived from the inverse Fourier transformation of Eq. 2 the transfer function. Which is the signal one would get if a single impulse signal would be applied to the filter circuit. In practise this function will be used for mathematical operations only, but is not very useful to describe a technically realised filter circuit.

$$h(t) = \frac{\omega_c\, A_0 \left(1 + \operatorname{sign}(t)\right) \sinh\left(\frac{\sqrt{a_1^2 - 4b_1}\, f_c\, \pi\, t}{b_1}\right)}{\sqrt{a_1^2 - 4b_1}\; e^{\frac{a_1 f_c \pi t}{b_1}}} \qquad \text{(Eq. 11)}$$

The overall shape is that of a sin(x)/x or *sinc(x)* function but necessarily has to be zero for times $t \leq 0$, otherwise the filter would violate the causality of the signal.

This is of course valid for analogue filters only, because for digital filters it is quite common to use a recursive algorithm whenever it is applicable and useful for the data evaluation.

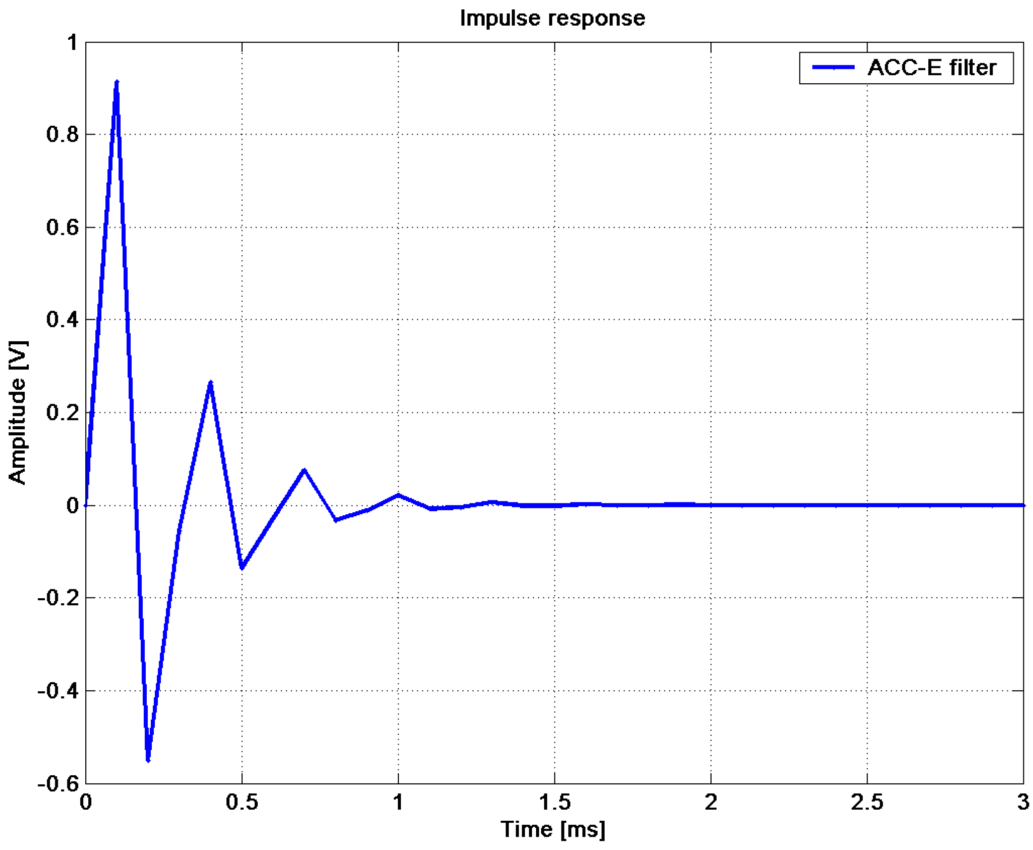

Figure. 5. ACC-E filter impulse response computed with the time resolution of a 10 kHz sampling rate. It should be noted that the period of the ringing coincides with the corner frequency of the filter.

1.2 Data

The data for the Huygens SSP sensor ACC-E were delivered from the spacecraft as time tagged data blocks in bit units of the used analogue to digital converter. To be able to use them finally as scientific data several conversion steps are necessary.

1.2.1 Raw data

The first step is the conversion from ADC-units into voltages:

ACC-E is equipped with a 8-bit AD converter with a nominal range of 4.5 V, where the ADC resolution **d** is the nominal range **r** divided by the bit resolution of the ADC (**b** is the bit length of the ADC).

$$d = \frac{r}{2^b} \qquad \text{(Eq. 12)}$$

Thus 1 bit corresponds for ACC-E to 0.0176 V. With this the ACC-E data can be converted using eq. 13

$$y = x \cdot r \qquad \text{(Eq. 13)}$$

which can be seen in Fig. 6.

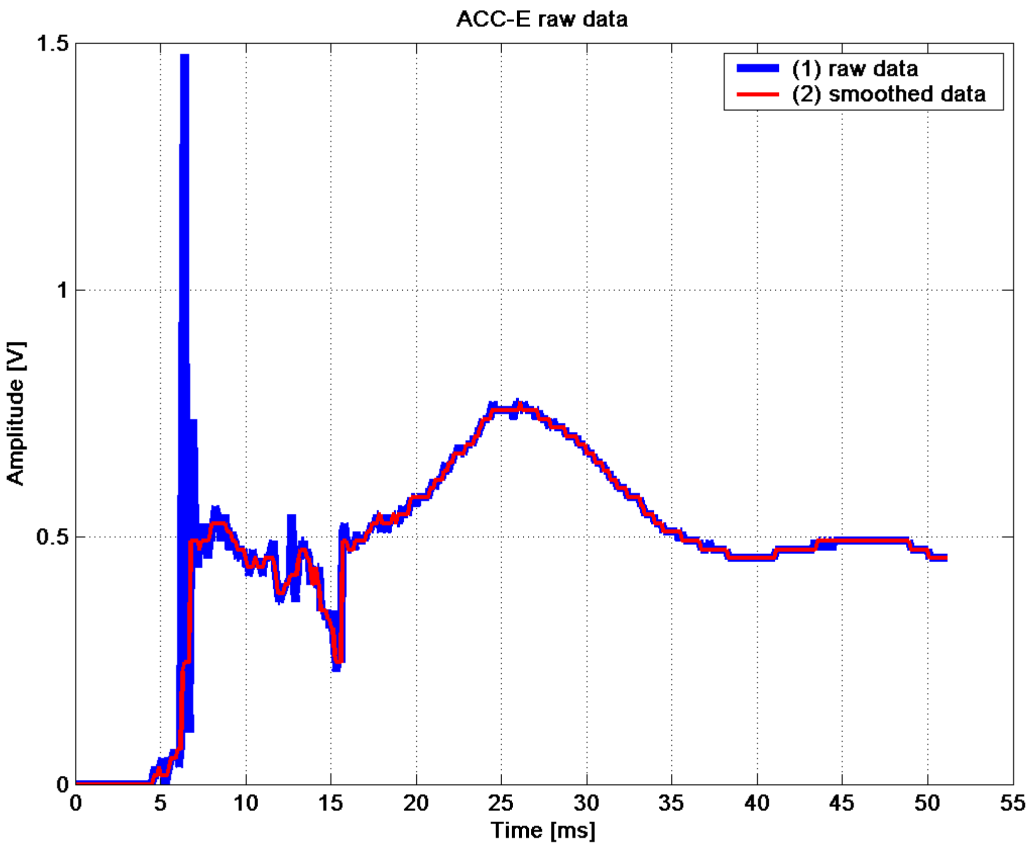

Figure. 6 Huygens ACC-E raw data after the conversion from ADC units into volts (blue line). Superimposed are data (red line) after an additional smoothing process using a five point medial filtering as discussed below.

1.2.2 Discussion of spurious signals

Generally data sampled in a measurement contain nor only the process data we are interested in but also a certain amount of noise and spurious data contributed from systematic sources. Thus it is important to judge already during the design phase of an instrument the sources of disturbances and try to minimize them as thoroughly as possible.

A sensor conditioning electronics can be characterized with a general block schematic as shown in Fig. 7 where elements are represented as functional blocks. Thus it is easy to distinct between isolated elements and their influence on the signal. At the same time the sources of spurious signals and operations to improve the data quality are shown as well.

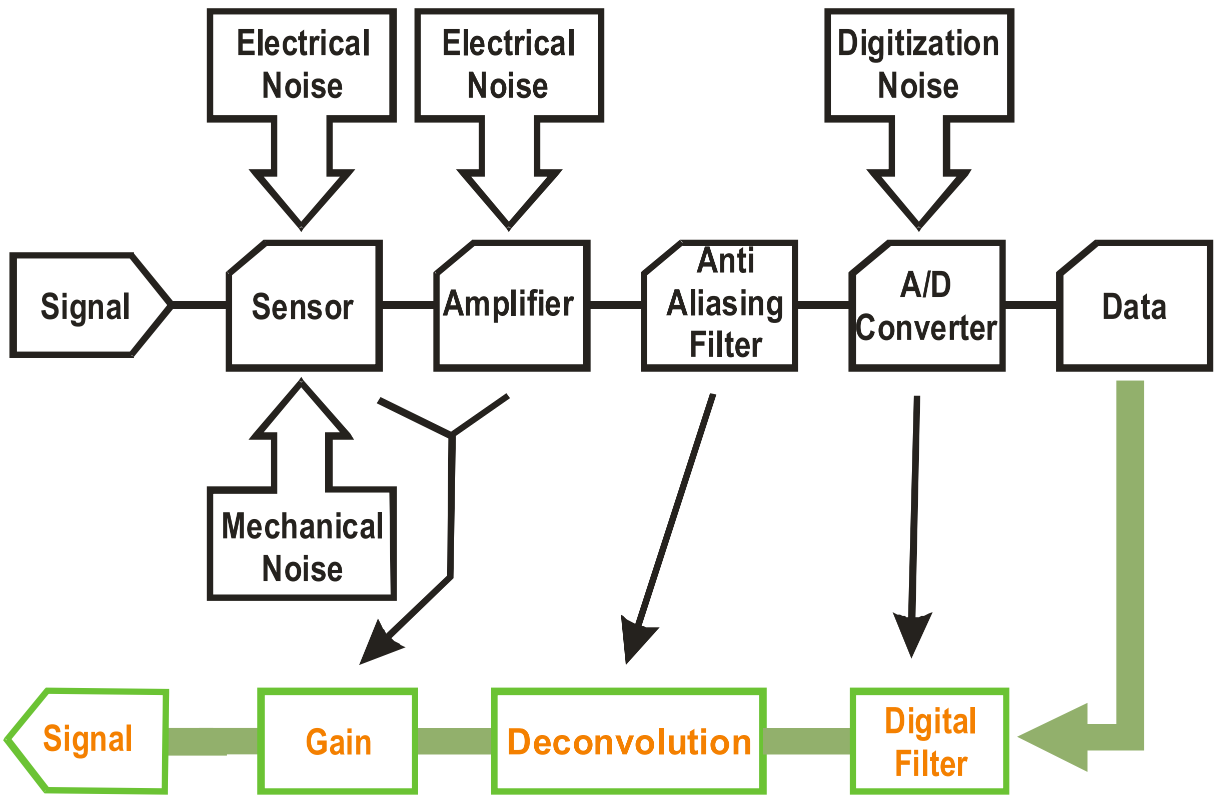

Figure. 7 ACC-E block schematic of the conditioning electronics representing the signal path including sources of spurious signals and also depicting a data processing path.

Noise

Any type of sensor is prone to the addition of ambient and system inherent noise to the measured signal. Mostly this is electrical noise picked up via induction and capacitances. Since ACC-E and ACC-I as well are basically mechanical sensitive systems mechanical noise will also be present. Another source of noise is within the charge amplifier, which by the very nature of the amplifier is quite sensitive to stray capacitances and discharging for long duration signals. The analogue to digital conversion adds white noise[3] to the data, due to the quantisation of the analogue signal into discrete digital bit levels.

[3] Noise with an uniform spectral distribution of the signal

Aliasing

As already discussed above (eq. 1) exist another source of spurious signals due to the sampling of the analogue signal with a finite sampling frequency. The so called aliasing is the superposition of high frequent signal parts upon lower frequent ones. Since, we have no way to reconstruct the signal part above the Nyquist frequency the aliasing should be suppressed as thoroughly as possible. Nevertheless, sometimes it is not avoidable to have signal contributions from aliasing sources, and it remains within the careful judgement of the data processing scientist to remove them before the interpretation.

Smoothing

The only way to reduce the noise level of a signal is a high frequency filtering most commonly referred to as smoothing and is usually a kind of sliding average in simple cases or a full scaled digital filter for more sophisticated applications.

In the case of ACC-E data, a five point median filter was chosen to remove most of the digitisation noise. A median filtering was chosen because it preserves steep flanks within a signal much better than usual sliding average filters. (Fig. 6)

For the sake of quantisation noise removal a three point median filter would be enough most of the time, however we could use a five point version also to get rid of some of the excessive peaks which can not fully accounted for by the filter inversion process. A reasonable source of this excessive ringing amplitude is probably due to an aliasing of the ACC-E pylon resonance's into the lower frequency part of the signal which are excited at the time of sharp strokes against the pylon i.e. at the first impact and probably also at a later time during the penetration.

1.2.3 Filter removal

To compensate the influence of the anti-aliasing filter of the conditioning electronics and to get closer to the sensor response the filter needs to be removed from the signal. For this purpose the filter attitude needs to be known as precise as possible as already discussed above.

For the removal of filter influences on signals are two possible procedures available.

1. Division of the Fourier transformed signal by the filter transfer function, which is rather the engineering approach and useful for long data sequences.
2. The deconvolution of the signal and the filter impulse response, which is a more theoretical or mathematical approach. The advantage of this method is, that for short signals, as for ACC-E the causality of the signal is not violated.

Convolution and deconvolution

Any electronic circuit acts as a linear time invariant system, i.e. a kernel **h(t)** relates the output **y(t)** to the input **x(t)** by

$$y(t) = h * x = \int h(t-s)\, x(s) ds \qquad \text{(Eq. 14)}$$

where * denotes convolution. As already described in eq. 11 **h(t)** is the impulse response of the conditioning electronic system. By substituting **x(t)** with the Dirac delta function **δ(t)** we obtain

$$H(f) = \int h(t)e^{-i2\pi ft}dt \qquad \text{(Eq. 15)}$$

Which is called the transfer function of a circuit. Applying eq. 15 on eq. 14 we get,

$$Y(f) = H(f) \cdot X(f) \qquad \text{(Eq. 16)}$$

a relation between input signal and transfer function in the complex Fourier plane, which is used in our approach number 1 mentioned above.

For the pre-filter signal restoration we need to divide **Y(f)** by **H(f)** and perform an inverse Fourier transformation or in other words make a deconvolution of **y(t)** and **h(t)**.

Restoration of pre-filter data

After preparing the flight data with the above mentioned smoothing algorithm we can apply a deconvolution procedure to the data (Fig. 8). For the use of a deconvolution on a set of discrete data points some normalisation operations are necessary.

Given

$$n = \text{length(Impulse ressponse data)} \qquad \text{(Eq. 17)}$$

the data have to be zero padded with

$$m = n-2 \qquad \text{(Eq. 18)}$$

m trailing zeros to get a output signal in the same length than the raw data signal. For the use in the software package the impulse response signal of eq. 11 has to be normalised by the integral over the impulse response and multiplied by the total filter gain $\mathbf{A_0}$ (eq. 8)

It should be noted that all high frequency components of the signal which were not sufficiently suppressed beforehand will be enhanced to a large extend and become more prominent in the data. This can easily be made imaginable by supposing an inversion of the amplitude graph in Fig. 3.

1.2.4 Voltage to Force conversion

After restoring the pre filter level of the signal it remains only to deal with the influence of the pseudo-log pre-amplifier and the charge amplifier. A numerical simulation of both circuits showed that the frequency dependent influence of both circuits are negligible, thus only the frequency invariant gain changes have to be considered.

Since the penetration force measured by ACC-E on Titan was so low only the small signal level branch of the three gain levels was used. The PZT flight sensor had a sensitivity of $\mathbf{C_c} = 3.715157\ 10^{-3}\ {}^{V}/_{N}$. Therefore the amplification of the signal can be considered as a linear relation

$$y(t) = \frac{C_s}{C_c \cdot C_p}\, x(t) \qquad \text{(Eq. 19)}$$

with $\mathbf{C_p} = 9.3251$ the small signal gain of the pseudo logarithmic amplifier and with **Cs** = 1.833333333 the change in sensitivity of the PZT due to low temperatures at Titan.

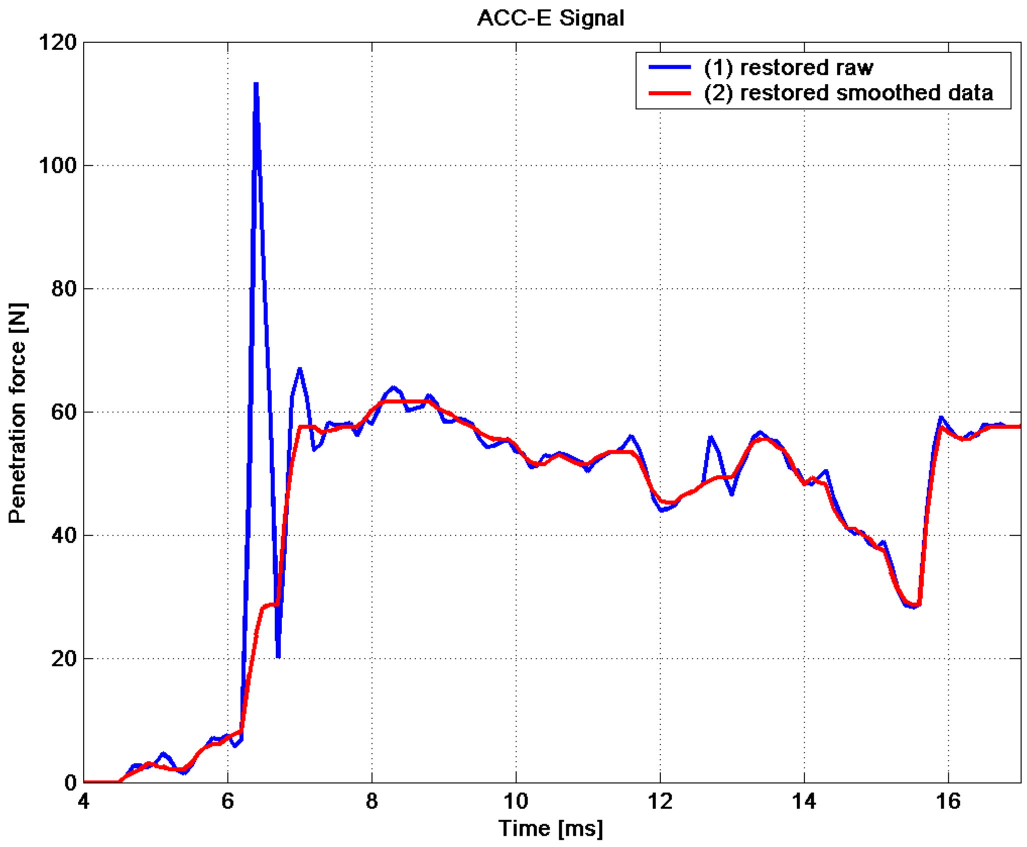

Figure. 8 ACC-E force data restored from the raw flight data. The blue line shows the data set derived if all above discussed additional noise sources were disregarded. The red line includes a smoothing that removes most of the noise and suppresses the impact ringing at the begin of the penetration process.

Applying eq. 19 to the ACC-E data after the filter inversion we get the penetration force data in Newtons as showed in Fig. 8. For a better view only the penetration part of the signal up to the impact of the base plate is shown.

1.2.5 Discussion

The next step will be the scientific interpretation of the penetration profile and the comparison with laboratory data. As can be seen in Fig. 8 the force data from the ACC-E sensor shows despite the smoothing still some internal structuring. However, the amount of those internal structures is a question of discussion.

From an engineering point of view, the strong ripples at the time of the impact have such a close resemblance with the impulse or the step response that for all practical purposes they should be discarded from a direct scientific interpretation. Eliminating them from the data could if desired allow a smaller amount of smoothing, even if the chosen 5 point median is already a minimal operation preserving most of the steep flanks in the data.

After an small rise which is probably due to the initial penetration of the pylon front half spherical tip a sharp impulse expressed in the already discussed ringing is superimposed on the signal. If this is the response to hitting a necessarily obstacle e.g. a glancing blow to a larger pebble or simply a mechanical resonance can only be explained by comparison with laboratory data. A similar but less obvious occurrence is at the time of about 13 ms where another impulse is followed by an area of enhanced resistance.

2 ACC-I

2.1 Sensor description

The ACC-I (Accelerometer-Internal) sensor comprises an Endevco 2271M20 piezoelectric accelerometer, mounted directly on the SSP electronic box inside the Huygens structure.

Its prime function is to record the Probe's deceleration history at impact, at a 500 Hz sampling rate, to infer the mechanical properties of the surface material.(Zarnecki et al., 1997)

The charge output is proportional to the acceleration, and is converted to a proportional voltage, filtered with a low pass filter and digitised at 12bits resolution. Like ACC-E, therefore it is not DC-sensitive and is unable to monitor slowly-varying accelerations (e.g. during entry). The maximum measurement range is 100 g. Like Acc-E the accelerometer signal is stored in a circular buffer, frozen when the integrated deceleration exceeds a preset threshold.

2.1.1 Electronic schematics

The anti aliasing filter part of the ACC-I signal conditioning electronics is a second order low-pass inverting filter with multiple feedback and a non-inverting pre-amplifier before the actual anti-aliasing filter.

It should be noted that the sensor charge amplifier including the stimulus-pulse input is not shown in the figure 9 below. For the total schematics refer to the flight model circuit schematics from RAL (2-KE-0107-818-01-E).

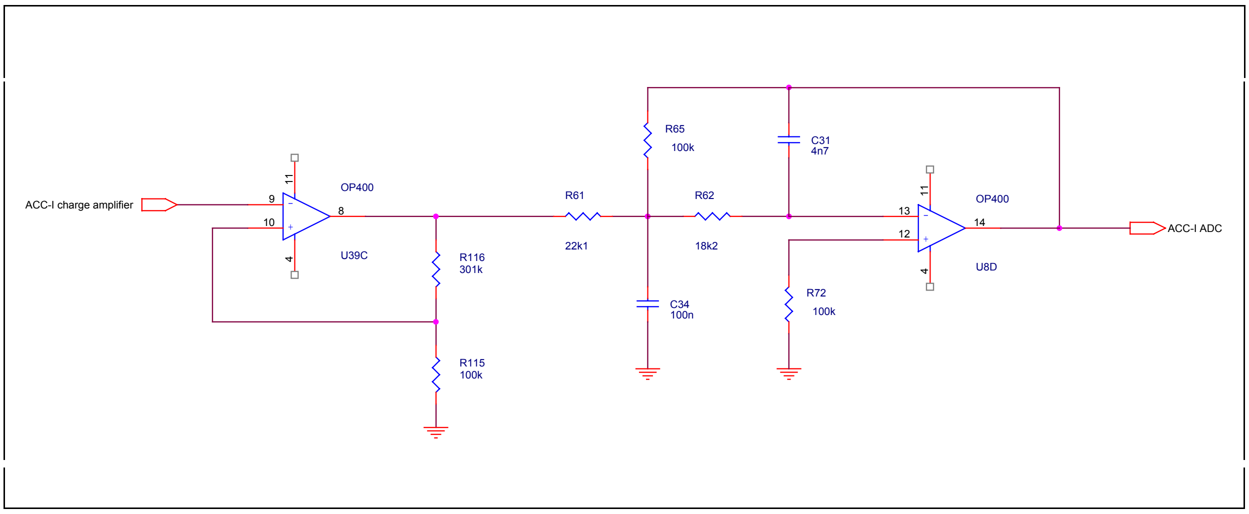

Figure. 9 ACC-I anti aliasing filter schematics including a pre-amplifier.

2.1.2 Filter description

Anti-aliasing Filter Parameters

As before for ACC-E the conditioning electronics is designed with a low-pass anti-aliasing filter, to suppress higher frequencies in the sampled acceleration data, which properties will be discussed in the following sections. However, for a total view of the ACC-I data an inherent high-pass filtering has to be considered as well in the data restoration process. The properties of the high-pass filter will be discussed separately after the anti-aliasing filter section.

As for the ACC-E filter the general filter transfer function follows the eqs. 2 and 3, with the filter parameters as:

$$a_1 = \omega_c C_1 \left(R_{65} + R_{62} + \frac{R_{65} R_{62}}{R_{61}} \right) \qquad \text{(Eq. 20)}$$

and

$$b_1 = \omega_c^2 C_1 C_2 R_{65} R_{62} \qquad \text{(Eq. 21)}$$

The filter gain is then

$$A_{0f} = -\frac{R_{65}}{R_{62}} \qquad \text{(Eq. 22)}$$

For the pre-amplifier the gain is

$$A_{0p} = 1 + \frac{R_{116}}{R_{115}} \qquad \text{(Eq. 23)}$$

and the total gain

$$A_0 = A_{0f} \cdot A_{0p} \qquad \text{(Eq. 24)}$$

Electronic components

With the components values and labels taken from the ACC-I flight electronic schematics as:

R_{61}= 22.1 kΩ;

R_{65}= 100 kΩ;

R_{62}= 18.2 kΩ;

R_{72}= 100 kΩ;

R_{116}= 301 kΩ;

R_{115}= 100 kΩ;

C_{31}= 4.7 nF

C_{34}= 100 nF

And a sampling frequency of:

f_s= 500

we can compute all necessary parameters for this filter.

Corner frequency

From a circuit simulation, we get a corner frequency of,

f_c= 217.3 Hz

which is in good agreement with the corner frequency given in the flight electronics test procedure protocol from RAL with 204 Hz and the nominal value of 215 Hz given there.

With the values above we get then the following filter parameters

Pre-amplifier gain

p_{a0}= 4.01

Filter gain

f_{A0}= -4.5249

Resulting total gain

A_0= -18.1448

And the filter parameters for a second order low-pas filter

a_1= 1.2870

b_1= 1.5946

Which indicates once again a Tschebyscheff type filter with 1.14 db corrugation as can be seen in Fig. 10.

Bode diagram

The properties of the ACC-I anti aliasing filter can best be depicted once again in the form of a Bode diagram, which shows that the price for the steep attenuation of higher frequencies at the corner frequency is a corrugation of more than 1.14 db in the range from ~40 - 150 Hz.

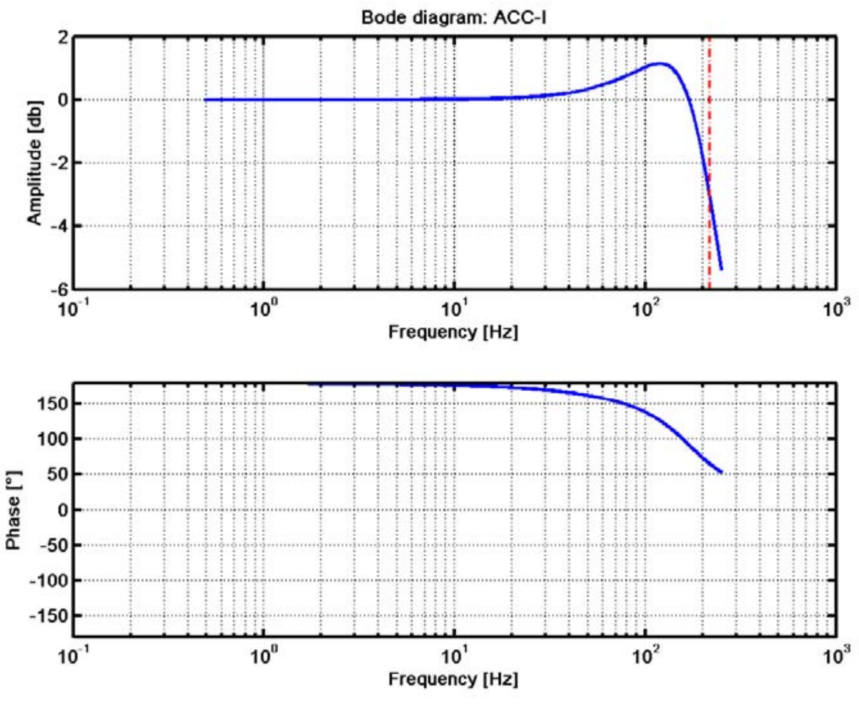

Figure. 10 Bode diagram of the ACC-I anti aliasing filter. Note that the gain showed in the upper panel is only the filter gain A_{0f} and not the total gain A_0 of the circuit in Fig. 9.

Since the power spectrum of the ACC-I data (Fig. 11)shows that most of the signal is in the frequency range below 50 Hz, the filter is nonetheless a good match to the anti aliasing task of the conditioning electronics.

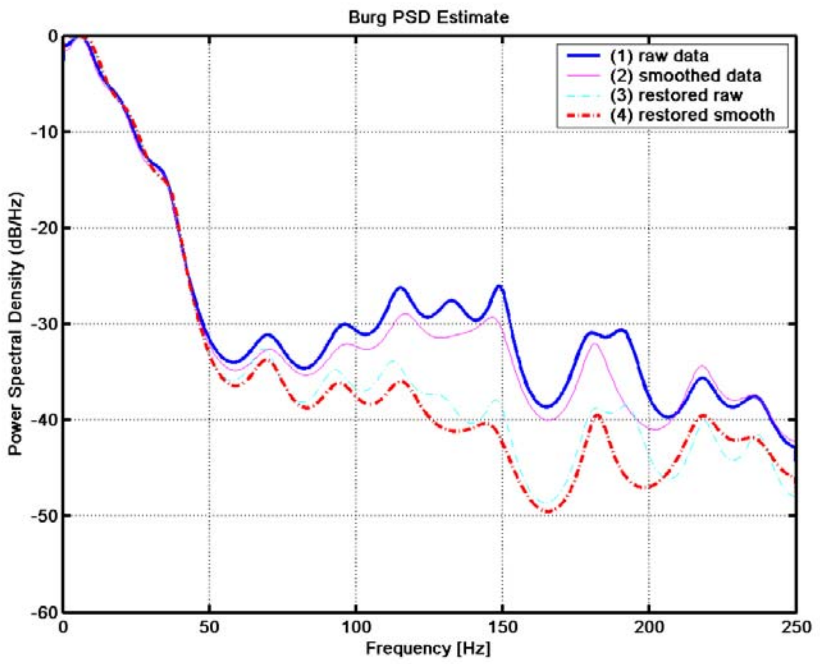

Figure. 11 Power spectrum of the ACC-I signal, derived with a Burg spectral estimation.

Step response

The step response of the ACC-I anti-aliasing filter is shown in Fig. 12. It should be noted that due to the inverting character of the filter circuit a positive step results in a negative output signal. This could also be seen in fig. 10 in the lower panel where a 180° phase shift is present for most of the frequency range as opposed to the more or less 0° phase shift of the ACC-E filter (Fig. 3)

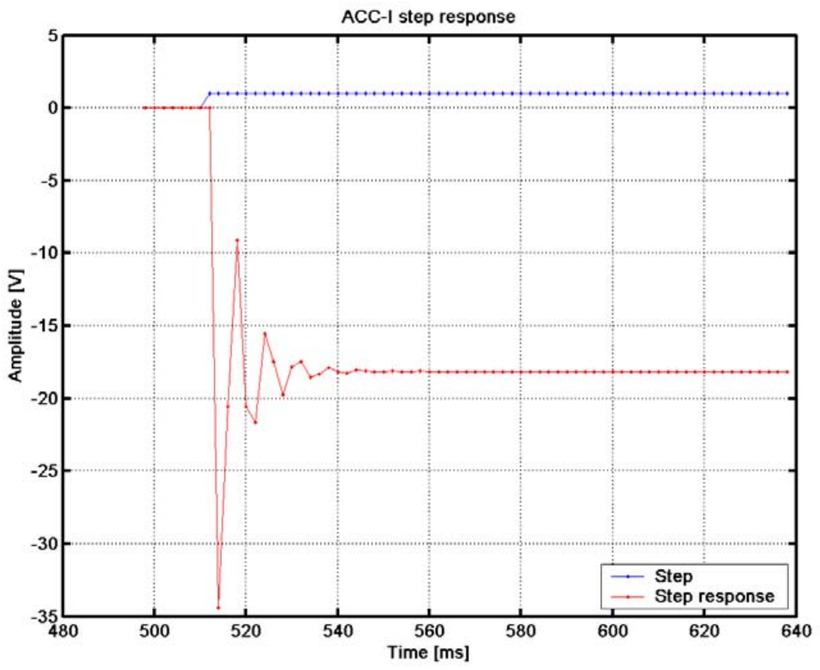

Figure. 12 Step response of the ACC-I anti-aliasing filter

Impulse response

For the restoration of the pre-filter data of the accelerometer, we once again need to know the impulse response of the low pass filter circuit computed with eq. 11 and the parameters set inferred above.

The remarkable difference for the ACC-I impulse response is that for a sampling frequency of 500 Hz and a corner frequency of ~217 Hz the duration of the ringing is much longer than for ACC-E. Also, since the filter is in a signal inverting configuration, the response to a positive impulse is resulting in a ringing starting in the negative direction.

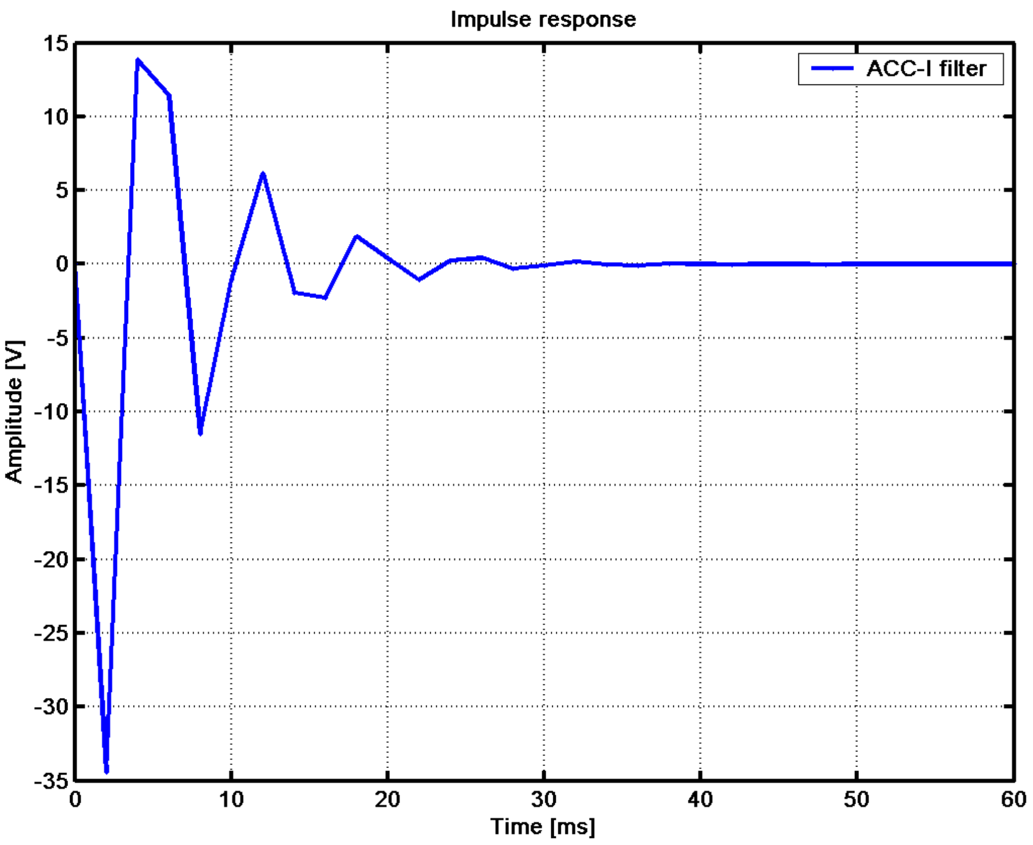

Figure. 13 ACC-I filter impulse response computed with the time resolution of a 500 Hz sampling rate.

High pass filter

The deceleration signal of ACC-I shows clear signatures of a high pass filtering with the corner frequency at ~1.7 Hz. This corner frequency has been confirmed also with tests on a shaker table and are part of the flight instrument documentation. As above the bode diagram of the high-pass filter can be seen in Fig. 14. If this high pass component of the signal remains untreated, a signal dependent decaying offset is introduced to the signal causing a wide excursion into negative accelerations which are electronics artefacts nonetheless.

For the data restoration each of this filter events can be treated separately, but could be removed in a combined action as well. Since we follow in principle still the data restoration path depicted in Fig. 7 we will however, remove each filter individually, starting with the low-pass filter and show the influence of each component on the signal.

Generally it can be stated that the influence of the anti-aliasing filter on the waveform of the ACC-I signal is much lower than for ACC-E, which is a good indication of the good balance of the whole system of anti-aliasing filter and sampling frequency.

Filter function

The high pass filter follows the equation of a first order filter

$$H_{hp} = \frac{A_\infty}{1 + \frac{a_1}{P}} \qquad \text{(Eq. 25)}$$

In the case of ACC-I we assume the following parameters:

$f_c = 1.7$ Hz

$\mathbf{A}_\infty = 1$

and

a1 = 1.

where $\mathbf{A}_\infty$ is the filter gain at $\mathbf{f} >> \mathbf{f_c}$ and a_1 is the standard filter parameter for as we have seen already in eq. 9 and further on as well.

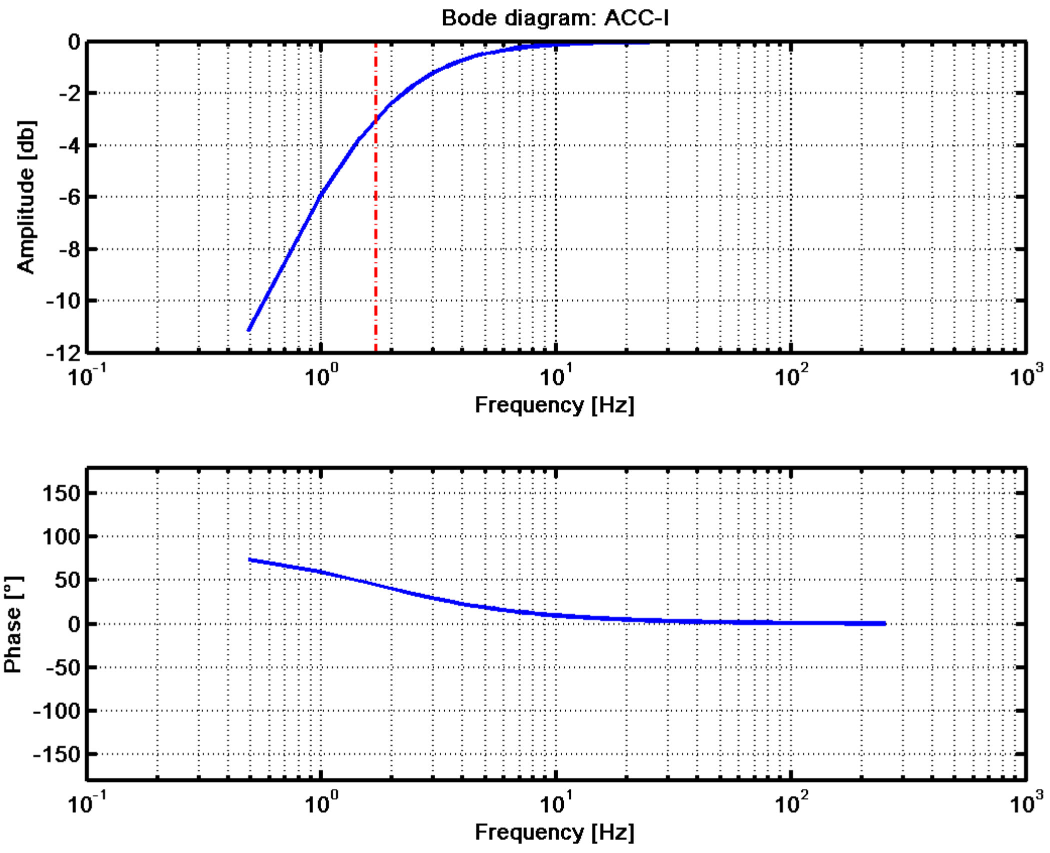

Figure. 14 Bode diagram of the high pass filter inherent in the ACC-I conditioning electronics including the accelerometer sensor

Step response

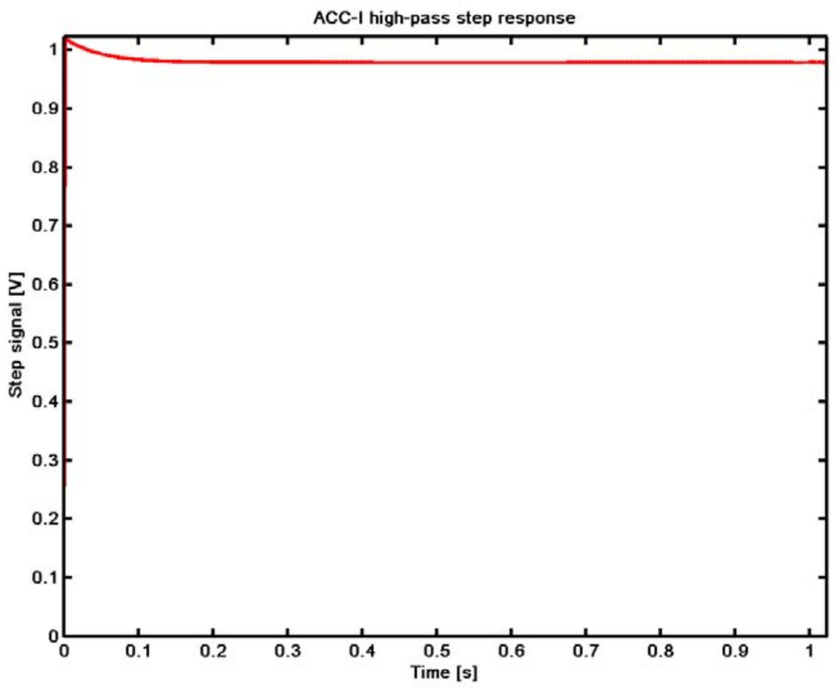

Figure. 15 Step response of the ACC-I high-pass filter.

Impulse response

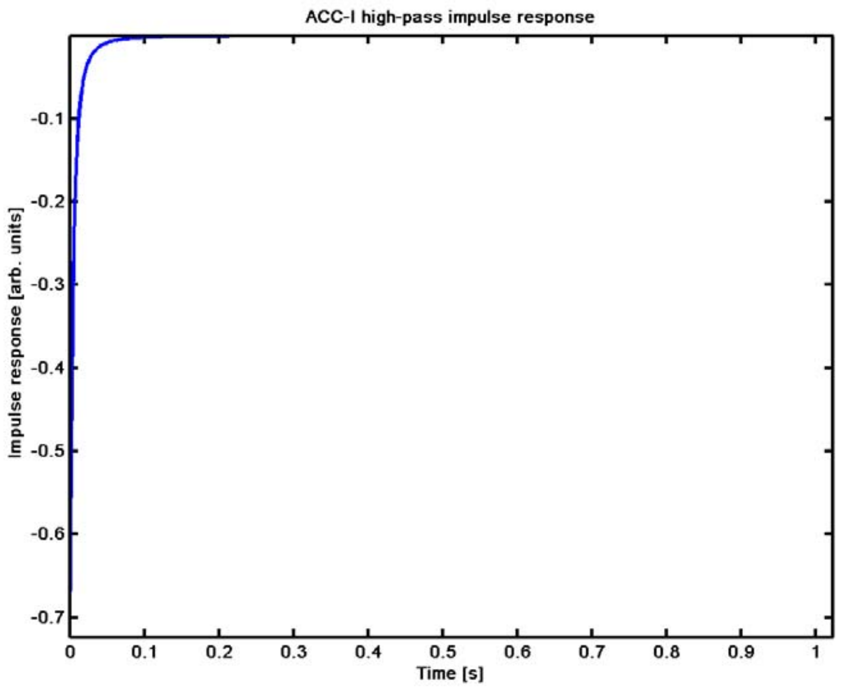

Figure. 16 Response of the ACC-I high-pass filter to a unity pulse.

The impulse as the step response of the high-pass filter present in the ACC-I data shows a delayed decay of the signal as response to a charging of the filter capacitance and subsequent slow discharging over the high impedance filter resistors.

Transfer function of the conditioning electronics

A overview of the total transfer function of the ACC-I conditioning electronics can be derived as the product of both the low- and high-pass transfer functions.

$$H_{ACC\text{-}I} = H_{LP} \bullet H_{HP} \qquad \text{(Eq. 26)}$$

The resulting transfer function in the frequency plain can be seen in Fig. 17 below.

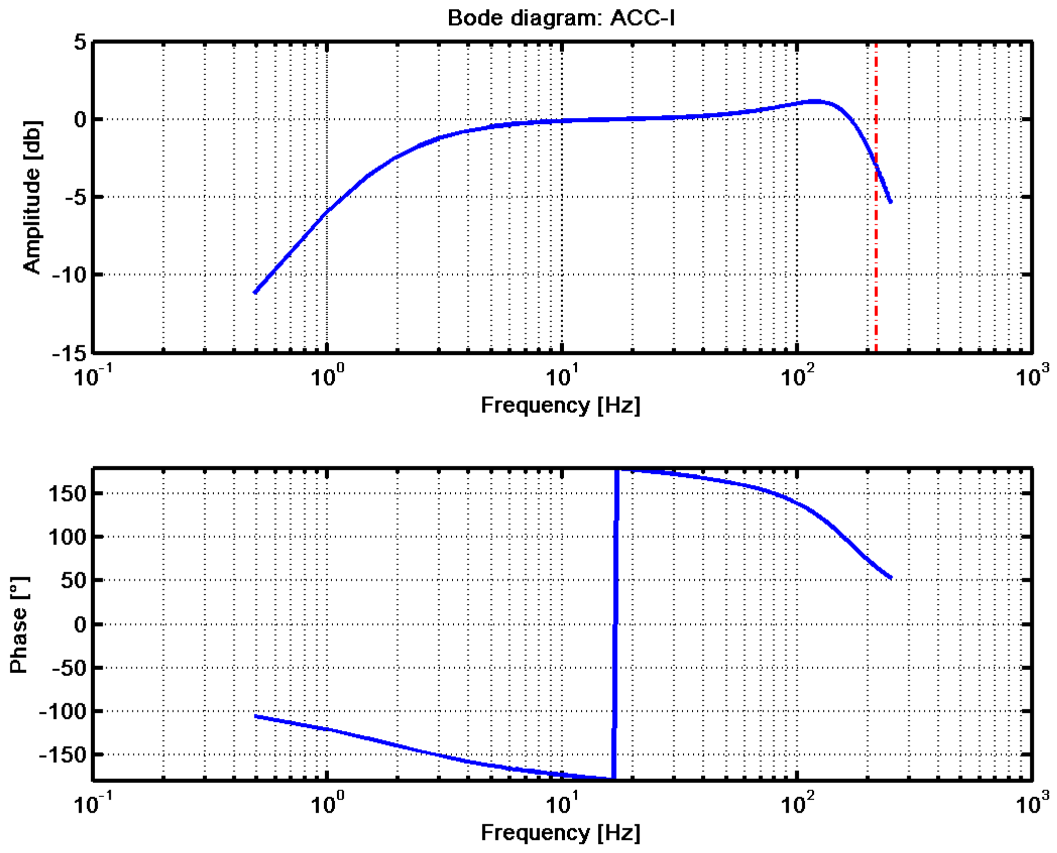

Figure. 17 Bode diagram of the resulting total transfer function of the ACC-I high- and low-pass filter.

2.2 Data

2.2.1 Raw data

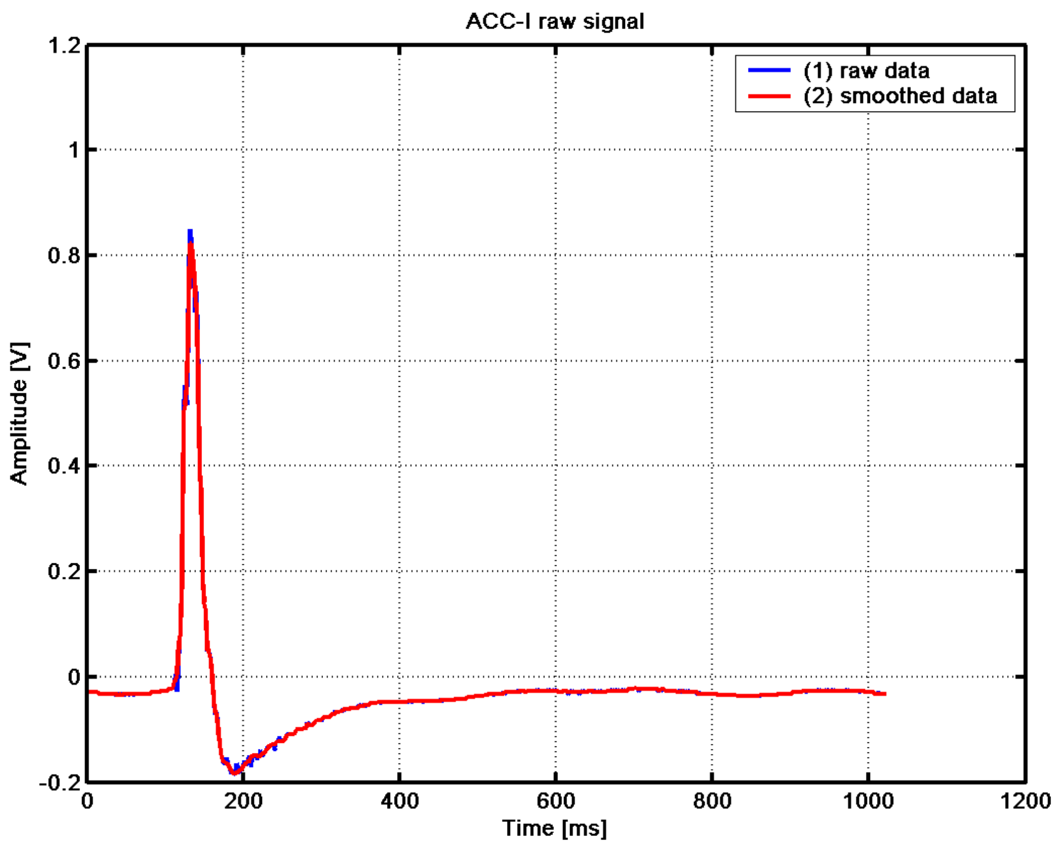

Figure. 18 ACC-I raw voltage data derived from ADC counts, the blue curve are raw untreated data, whereas the red curve is the result of a minor smoothing operation (3 point median filtering) to get rid of some ADC noise.

2.2.2 Smoothing discussion

As already discussed above also the ACC-I flight data contain a certain amount of digitisation noise, but obviously less of other "noise" than the ACC-E data. This may be due to the lower sampling rate and the different type of sensor being used in this application. How much of Huygens structural movement is or can be present in the deceleration signal is yet to be determined at later processing stages beyond this document.

What is present in the data, however are signal distortions due to the low- and high-pass filtering, especially the DC cut-off occurring in the accelerometer sensor is causing offsets and a over swing of the signal after the main deceleration at impact. This filter effects are, however already part of the data reconstruction procedure apart of a possible smoothing anyway.

The remaining noise can be easily removed with a simple three point median filtering. As can be seen in fig. 18 the raw and smoothed data do match in a more unsuspicious way than they do for the ACC-E data.

2.2.3 Filter removal

Restoration of pre-filter data

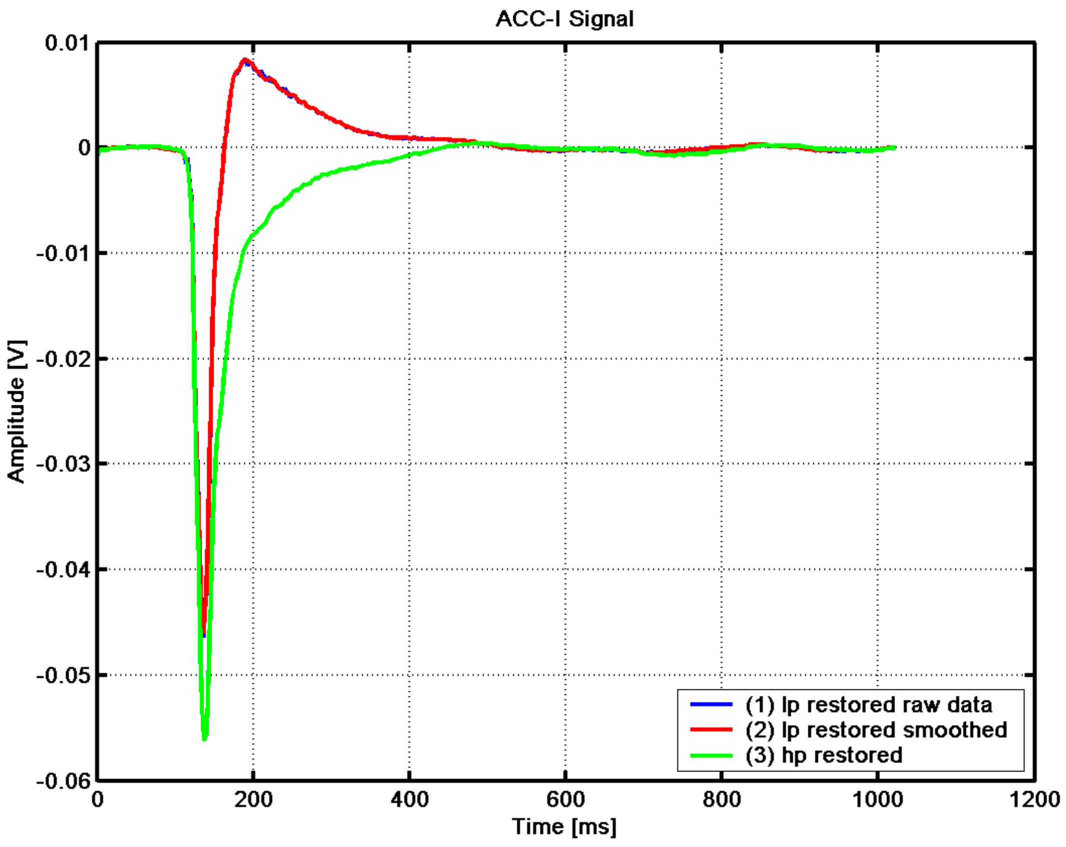

Figure. 19 ACC-I restored voltage data with both the anti-aliasing low-pass filter and the inherent high-pas filter influence removed.

The restoration of the pre-filter data is once again done by performing a deconvolution operation with the ACC-I data and the filter impulse response.

Tests on a shaker table indicated that there is possibly a high-pass filter present in the system, which is somehow problematic since it is clearly not present in the electrical design. However, we do know that the accelerometer is not DC sensitive i.e. it is not sensitive to static loads.

Test results indicate a high-pass transfer function with a cut off frequency of about 1.7 Hz, however, there are few data to confirm this since the low frequency resolution of the test facility was not sufficient for this low frequencies that would be needed.

This virtual high-pass filter was then treated as a first order high-pass filter with a corner frequency of 1.7 Hz. After the removal of this virtual filter the large rebound or over swing was more or less cancelled out completely from the signal as can be seen in Fig. 19.

The small oscillation motion remaining in the signal at signal times above 400 ms could well be some elastic motion within the Huygens structure or even an elastic rebound of Titans topsoil.

2.2.4 Voltage to Acceleration conversion

After the filter removal operation remains the conversion of the voltage signal (**y**) back into the physical acceleration/deceleration event (**a**) acting on the ACC-I accelerometer. The conversion operation is derived from the charge to voltage operation of a charge amplifier (Seifart, 2003) and the sensitivity of the accelerometer.

$$a = -9.81\, y \frac{C_i}{\kappa} \qquad \text{(Eq. 27)}$$

where

$Ci=4.7e-9$ [F]

is the integrating capacitor in the charge amplifier[4] and

$\kappa =13.e-12$ [C/g]

the typical charge sensitivity[5] of the Endevco 2271A/AM20 accelerometer, resulting in the ACC-I acceleration in m/s^2.

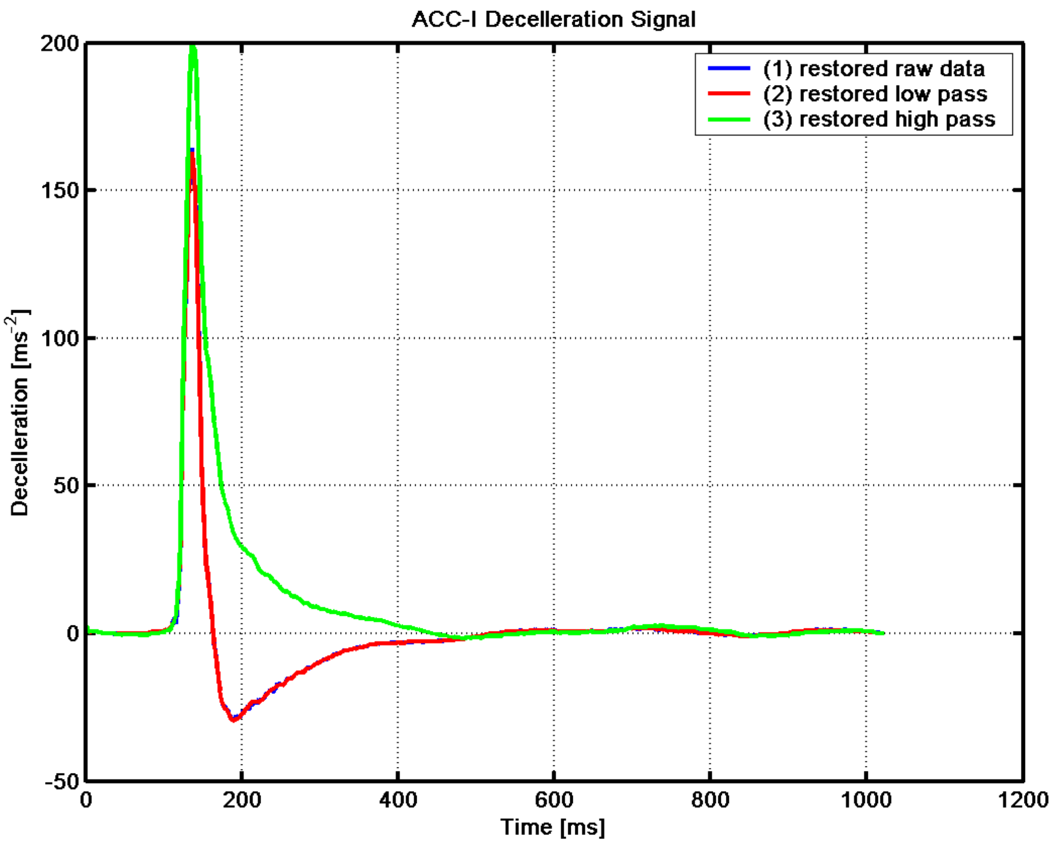

Figure. 20 The ACC-I acceleration at impact at Titans surface in various restoration levels with the green ▬ line being the fully restored impact signal.

[4] [F]=[As/V]

[5] [C/g]=9.81[As^2/m]

3 References

D.Ch. von Grünigen, Digitale Signalverarbeitung, Fachbuchverlag Leipzig (Hanser Verlag), 2004

G. Kargl, W. Macher, N. I. Kömle, M. Thiel, Ch. Rohe, A. J. Ball, Accelerometry measurements using the Rosetta Lander's anchoring harpoon: experimental set-up, data reduction and signal analysis. Planet. Space Sci. 49, 425 - 435, 2001

R. D. Lorenz, B. Hathi, M. R. Leese, J. R. Garry, and J. C. Zarnecki, The Huygens SSP penetrometer: an update, In: *Penetrometry in the Solar System*, Eds. Kömle, N. I. et al., Austrian Academy of Sciences Press, Vienna, 99-108, 2001

R. D. Lorenz, M. Bannister, P. M. Daniell, Z. Krysinski, M. R. Leese, R. J. Miller, G. Newton, P. Rabbets, D. M. Willett, and J. C. Zarnecki, An impact penetrometer for a landing spacecraft., Meas. Sci. Technol., 1033 - 1041, 1994

W. Macher, G. Kargl, and N. I. Kömle, The influence of the MUPUS-ANC-M transfer function on recorded accelerometer signals, IWF Internal Report 112, Institute for Space Sciences, Austrian Academy of Sciences, 1999

M. Seifart, Analoge Schaltungen, HUSS-Medien GmbH, Berlin, 2003

U. Tietze and Ch. Schenk (Eds.), Halbleiter-Schaltungstechnik, 10. Edition, Springer Berlin Heidelberg New York, 1993

J. Z. Zarnecki, M. Banaszkiewicz, M. Bannister, W. V. Boynton. P. Challenor, B. Clark, P. M. Daniell, J. Delderfield, M. A. English, M. Fulchignoni, J. R. C. Garry, J. E. Geake, S. F. Green, B. Hathi, S. Jaroslawski, M. R. Leese, R. D. Lorenz, J. A. M. McDonnel, H. Svedhem, R. F. Turner & M. J. Wright, The Huygens Surface Science Package Huygens: Science, Payload and Mission, Proceedings of an ESA conference. Edited by A. Wilson (1997)., p.177, 1997

4 Program Listings

4.1 ACC-E Flight data processing

```
% ACC-E flight data processing
% load flight data and and convert ADC units into voltages
% %anti aliasing filter and noise removal
% restoration of pre-filter data
% G. Kargl, IWF, 1. Mar. 2005
% Version: 1.0
% Changelog:

%----------------------------------------------------------------------------
close all;
% clear all;

% ACCEFDL                                  %load ACC-E (ADC) data from file
% ACC-E flight data loader
% load flight data and and convert ADC units into voltages
% G. Kargl, IWF Jan. 2005
%..........................................................................
% ACC-E Flight data file
acfdac_file='C:\Projekte\ACC-E\Data\Flight\ACC-E data.txt';
% acfildata=[pathname filename];
acfdac=load(acfdac_file);    %load flight data ADC units

z=acfdac(:,2);               %flight raw data ADC units in row 2
%..........................................................................
% Data are in ADC units: 8bit 4.5V range
%----------------------------------------------------------------------------
x1=DAC(z,4.5,8);             %convert ADC units into voltages
% disp('ACC-E flight data loader');
%----------------------------------------------------------------------------
% x1=x;                                    %x1 ACC-E voltage data
%..........................................................................
% ACCECOMP                           %compute anti aliasing filter components
% get components values from file
comp=getcomp( 'C:\Projekte\ACC-E\Mfiles\ACCEfil.dat' );
%..........................................................................
%..........................................................................
comp.fs=1.e4;                        %sampling frequency
%..........................................................................
% comp=salkeyfilt(comp);             %compute designed ACC-E filter paramters
comp.a1=    1.3332;
comp.b1=    1.4622;
comp.A0=    0.4522;
comp.fc=    4733; %Hz corner frequency derived by other means
%..........................................................................

fc=comp.fc;                          %corner frequency of filter
a0=comp.A0;                          %filter gain
a1=comp.a1;                          %first filter parameter
b1=comp.b1;                          %second filter parameter
fs=comp.fs;                          %sampling frequency

t0=1/fs;                             %time resolution

n=length(x1);                        %number of data points
%..........................................................................
% compute impulse response of filter
```

```
ti=[0:t0:t0*(n-1)]';          %time vector

irp7=irp(a0,a1,b1,fc,ti);     %impulse response of ACC-E filter
%.................................................................
% generate example data
      xs0=70;           %start point for first peak
      xs=70;            %end  point  of  first  peak:  xs=xs0  single  hit  else
its a crust
      xs2=130;    %end point of first plateau
      xs3=xs0+30;                   %point of second hit (impulse)

      x=zeros(1,512);   %make all points zero
      x(xs:end)=1.3;    %final plateau (foredome)
%     x(xs-15:xs)=0.025.*(log([1:16]));  %log rise simulating sphere tip
%     x(xs:xs2)=-0.3*((xs:xs2)-xs+1)./xs2+1;   %first plateau declining
      x(xs0:xs)=3.5;                                  %hight     of     first
impulse (hit)
      %bump at the center
%     x(xs3+1:xs3+11)=x(xs3+1:xs3+11)+(-0.003.*([xs3+1:xs3+11]-xs3-7).^2
+0.15)';
%     x(xs3)=1.3; %second hit
% %   x(101)=0.25;
x=[zeros(1,n/2),ones(1,n/2)]';            %step signal
%.................................................................
% zero padding
z1=[x1' zeros(1,length(irp7)-2)]';
%.................................................................
z3=conv(x,irp7);              %filtered or convoluted signal
%.................................................................
% here it happens
% it turns out that obviously a deconvolution in this case
%.................................................................
y1=(deconv(z1,irp7(2:end)));%filter restoration per deconvolution
%.................................................................
%.................................................................
% pre filtering against ADC and aliasing noise
% z2=z1;                                  %just in case we want no smooting
filn=3;
z2=daver(z1,filn,5);                %median to remove white ADC noise
% z2=daver(z2,5,4);                 %Hanning windows sliding average
x2=z2(1:n);
%.................................................................
y2=(deconv(z2,irp7(2:end)));%restore smoothed data
%.................................................................
% compute the filter transfer function just for show
%------------------------------------------------------------------
n3=length(z2);                                    %length of padded data
m3=fix( n3/2 -1/4);                               %incomprehensive
number
      ff3=comp.fs/ n3*(1: m3+2)';               %frequency vector
      P=i* ff3/comp.fc;                         %normalized frequency

      H=alp2(comp.A0,comp.a1,comp.b1,P);  %complex filter function
%.................................................................
% Voltage to Force conversion
PCT_sens=0.003715157;   %V/N sensitivity of PZT flight sensor
plhg=9.3251;                  %pseudo log amplifier high gain branch
senstdrop=1.833333333;  %PCT sensitivity drop due to low temperatures
scalfac=PCT_sens*plhg/senstdrop;
ac1=y1./scalfac;
ac2=y2./scalfac;
%------------------------------------------------------------------
% from here on graphical output only
```

```
%---------------------------------------------------------------------------
xax=ti*1000;                        %time in miliseconds
xlab='Time [ms]';            %time as x axis
tit='ACC-E';                                  %label    of    corresponding
filter
pc0=5;                                  %graph window counters
%---------------------------------------------------------------------------
% impulse response
pc=1;                               %graph window counter
ae(pc).fi=figure(pc+pc0);           %open new graph window
set(ae(pc).fi,'name','Impulse response','Position',[310 190 700 500]);

      ae(pc).p=plot(xax,irp7,'.b-');

      ae(pc).ti=title('Impulse response');
      ae(pc).le=legend('ACC-E filter',1);
      ae(pc).yl=ylabel('Amplitude [V]','fontsize',11,'Fontweight','demi');
      ae(pc).xl=xlabel(xlab,'fontsize',11,'Fontweight','demi');
            set(gca,'fontsize',11,'Fontweight','demi');
            set(gca,'linewidth',2)
            set(ae(pc).le,'fontsize',11,'Fontweight','demi');
            set(ae(pc).ti,'fontsize',11,'Fontweight','demi');
            set(ae(pc).p,'LineWidth',2.);
 set(gca,'xlim',[0.,3])

      grid on;
      zoom on
%     axis tight
%...........................................................................
% % difference between signal and restoration
pc=pc+1;
% ae(pc).fi=figure(pc+pc0);
% set(ae(pc).fi,'name','Signal differences');
%     ae(pc).p=plot(ti,x1/a0-y1,'r');
% hold on
%     ae(pc).p2=plot(ti,y1-y2,'g');
% hold off

%     ae(pc).ti=title('difference');
%     ae(pc).le=legend('x1/a0-y1','y1-y2',1);
%           set(gca,'fontsize',11,'Fontweight','demi');
%           set(ae(pc).le,'fontsize',11,'Fontweight','demi');
%           set(ae(pc).ti,'fontsize',11,'Fontweight','demi');
% %         set(ae(pc).p,'LineWidth',2.);
%           set(ae(pc).p2,'LineWidth',2.);
%
%     grid on;
%     zoom on
%     axis tight
%...........................................................................
% step response
pc=pc+1;
ae(pc).fi=figure(pc+pc0);
set(ae(pc).fi,'name','Step response','Position',[290 170 700 500]);

n2=250;
n3=320;
      ae(pc).p=plot(xax(n2:n3),x(n2:n3),'.b-');
hold on
      ae(pc).p2=plot(xax(n2:n3),z3(n2:n3),'.r-');
hold off

      ae(pc).ti=title('ACC-E step response');
      ae(pc).le=legend('Step','Step response',1);
            set(gca,'fontsize',11,'Fontweight','demi');
            set(gca,'linewidth',2)
            set(ae(pc).le,'fontsize',11,'Fontweight','demi');
            set(ae(pc).ti,'fontsize',11,'Fontweight','demi');

      grid on;
      zoom on

%.............................................................................
% bode diagram
pc=pc+1;
ae(pc).fi=figure(pc+pc0);
set(ae(pc).fi,'name','Bode diagram','Position',[270 150 700 500]);
opt.fc=comp.fc;         %set options for bode plot
opt.lincol='r';
opt.linestyle='-.';
opt.a0=comp.A0;

[mh,imh]=max(abs(H));                             %gain maximum in H
ae(pc).bd=ibodeplot(ff3,H,pc+pc0,'abs',tit,opt);     %create  bode  plot  of
filter
%.............................................................................
%power spectrum         compute and plot power spcectrum of ACC-E data
pc=pc+1;
ae(pc).fi=figure(pc+pc0);
set(ae(pc).fi,'name','Power spectrum','Position',[250 130 700 500]);

%     xff=fft(xb);      %FFT of ACC-E raw data
[pte1,fpe]=pburg(x1,25,1024*2,fs);  %spectral power spectral density
[pte2,fpe]=pburg(x2,25,1024*2,fs);  %spectral power spectral density
[pte3,fpe]=pburg(y1,25,1024*2,fs);  %spectral power spectral density
[pte4,fpe]=pburg(y2,25,1024*2,fs);  %spectral power spectral density

            ae(pc).p=plot(fpe,mkdb(pte1)/2,'b-');
      hold on
            ae(pc).p2=plot(fpe,mkdb(pte2)/2,'m-');
            ae(pc).p3=plot(fpe,mkdb(pte3)/2,'c-.');
            ae(pc).p4=plot(fpe,mkdb(pte4)/2,'r-.');
      hold off
%     ,'fontsize',11,'Fontweight','demi'
            ae(pc).ti=title('Burg PSD Estimate');
      ae(pc).le=legend('(1) raw data'...
                              ,'(2) smoothed data'...
                              ,'(3) restored raw'...
                              ,'(4) restored smooth',1);
            ae(pc).yl=ylabel('Power Spectral Density (dB/Hz)'...
                  ,'fontsize',11,'Fontweight','demi');
            ae(pc).xl=xlabel('Frequency [Hz]'...
                  ,'fontsize',11,'Fontweight','demi');
            set(gca,'fontsize',11,'Fontweight','demi');
            set(gca,'linewidth',2)
            set(ae(pc).le,'fontsize',11,'Fontweight','demi');
            set(ae(pc).ti,'fontsize',11,'Fontweight','demi');
            set(ae(pc).p,'LineWidth',2.);
            set(ae(pc).p4,'LineWidth',2.);
      grid on;
      zoom on
%.............................................................................
% raw signal
pc=pc+1;
ae(pc).fi=figure(pc+pc0);
set(ae(pc).fi,'name','ACC-E raw data','Position',[230 110 700 500]);
```

```
      ae(pc).p=plot(xax,x1,'b');
hold on
      ae(pc).p2=plot(xax,x2,'r');
hold off
      ae(pc).ti=title('ACC-E raw data');
      ae(pc).yl=ylabel('Amplitude [V]','fontsize',11,'Fontweight','demi');
      ae(pc).xl=xlabel(xlab,'fontsize',11,'Fontweight','demi');
      ae(pc).le=legend('(1) raw data'...
                              ,'(2) smoothed data',1);
      set(gca,'fontsize',11,'Fontweight','demi');
            set(ae(pc).le,'fontsize',11,'Fontweight','demi');
            set(ae(pc).ti,'fontsize',11,'Fontweight','demi');
            set(ae(pc).p,'LineWidth',4.);
            set(ae(pc).p2,'LineWidth',2.);
      grid on;
      zoom on
      set(gca,'xlim',[0 55]);
%     axis tight
%.............................................................................
% signal and restored signal
pc=pc+1;
ae(pc).fi=figure(pc+pc0);
set(ae(pc).fi,'name','ACC-E signal','Position',[210 110 700 500]);

      ae(pc).p=plot(xax,y1/scalfac,'b');
hold on
      ae(pc).p2=plot(xax,y2/scalfac,'r');
%     ae(pc).p4=plot(xax,x2/scalfac,'m');
%     ae(pc).p2=plot(xax,y1/scalfac,'c');
hold off

      ae(pc).ti=title('ACC-E Signal');
      ae(pc).le=legend('(1) restored raw'...
                              ,'(2) restored smoothed data',1);
      ae(pc).yl=ylabel('Penetration                                    force
[N]','fontsize',11,'Fontweight','demi');
      ae(pc).xl=xlabel(xlab,'fontsize',11,'Fontweight','demi');
            set(gca,'fontsize',11,'Fontweight','demi');
            set(gca,'linewidth',2)
            set(ae(pc).le,'fontsize',11,'Fontweight','demi');
            set(ae(pc).ti,'fontsize',11,'Fontweight','demi');
            set(ae(pc).p,'LineWidth',2.);
            set(ae(pc).p2,'LineWidth',2.);

      set(gca,'xlim',[4 17]);
      grid on;
      zoom on
%     axis tight
%.............................................................................
```

4.2 ACC-I flight data processing

```
% ACC-I flight data processing
% load flight data and and convert ADC units into voltages
% %anti aliasing filter and noise removal
% restoration of pre-filter data
% G. Kargl, IWF, 1. Mar. 2005
% Version: 1.0
% Changelog:

%-----------------------------------------------------------------------
% clear all
%......................................................................
%-----------------------------------------------------------------------
% ACC-I Flight data file
% acidfile='C:\Projekte\ACC-E\Data\Flight\ACC-I\acciv2.txt';
acidfile='C:\Projekte\ACC-E\Data\Flight\ACC-I\acci_voltsv2.txt';

fidi=fopen(acidfile);
      ai=fscanf(fidi,'%g %g',[2 inf])';
fclose(fidi);

tai=ai(:,1);
tai=tai-tai(1);

x1=ai(:,2);
%-----------------------------------------------------------------------
%......................................................................
% Data are in V units:
%-----------------------------------------------------------------------
%......................................................................
% get components values from file
comp=getcomp( 'C:\Projekte\ACC-E\Mfiles\ACCIfil.dat' );
% comp=salkeyfilt(comp);           %compute designed ACC-E filter paramters
%......................................................................
comp.fc =  217.3;                  %corner    frequency    asserted    from
simulation
% comp.fc =  204.3;                      %corner  frequency  asserted  from
simulation
comp.fhc =  1.7;                   %corner    frequency    asserted    from
simulation
comp.pa0=(1 + comp.R5/comp.R6);    %gain of pre-amplifier

comp.fA0=-comp.R2/comp.R1;         %filter gain w/o pre-amp
comp.A0=comp.pa0*comp.fA0;         %total gain
comp.a1=2*pi*comp.fc*comp.C1*(comp.R2+comp.R3+(comp.R2*comp.R3/comp.R1));
comp.b1=(2*pi*comp.fc)^2 *comp.C1*comp.C2*comp.R2*comp.R3;
%......................................................................
comp.fs=500;                       %sampling frequency
%......................................................................
fg=comp.fc;                        %corner frequency of filter
a0=comp.A0;                        %filter gain
a1=comp.a1;                        %first filter parameter
b1=comp.b1;                        %second filter parameter
fs=comp.fs;                        %sampling frequency

t0=1/fs;                           %time resolution

n=length(x1);                      %number of data points
xan=50;                                  %length    point    for    offset
correction
%......................................................................
```

```
% compute impulse response of filter
ti=[0:t0:t0*(n-1)]';          %time vector

irp7=irp(a0,a1,b1,fg,ti);     %impulse response of ACC-E filter
%.............................................................................
% generate example data
      xs0=70;          %start point for first peak
      xs=70;           %end  point  of  first  peak:  xs=xs0  single  hit  else
its a crust
      xs2=130;    %end point of first plateau
      xs3=xs0+30;                 %point of second hit (impulse)

      x=zeros(512,1);   %make all points zero
      x(xs:end)=1.3;    %final plateau (foredome)
%     x(xs-15:xs)=0.025.*(log([1:16]));  %log rise simulating sphere tip
%     x(xs:xs2)=-0.3*((xs:xs2)-xs+1)./xs2+1;   %first plateau declining
%     x(xs0:xs)=3.5;                                 %hight     of     first
impulse (hit)
      %bump at the center
%     x(xs3+1:xs3+11)=x(xs3+1:xs3+11)+(-0.003.*([xs3+1:xs3+11]-xs3-7).^2
+0.15)';
%     x(xs3)=1.3; %second hit
% %   x(101)=0.25;
x=[zeros(1,n/2),ones(1,n/2)]';          %step signal
%.............................................................................
% zero padding: padd with trailing zeros to have the right sample length
z1=[x1' zeros(1,length(irp7)-2)]';
%.............................................................................
z3=conv(x,irp7);              %filtered or convoluted signal
%.............................................................................
% here it happens
% it turns out that obviously a deconvolution in this case is a good
% approach. However, some "noise" has to be considered as not due to the
% filter electronics alone. Therefore educated guesses for the source are
% appropriate and should carefully be eliminated if necessary.
%.............................................................................
y1=(deconv(z1,irp7(2:end)));%filter restoration per deconvolution
%.............................................................................
% pre filtering against ADC and aliasing noise
% z2=z1;                            %just in case we want no smooting
sl=3;                               %smoothing length
% [z2,mdat,stdat,typ,aver]=daver(z1,sl,5);%median to remove white ADC noise
[x2,mdata,stdata,typ,aver]=daver(x1,sl,5);
% x2=z2(1:n);
z2=[x2' zeros(1,length(irp7)-2)]';
%.............................................................................
y2=(deconv(z2,irp7(2:end)));%restore smoothed data
%.............................................................................
% compute the low pass filter transfer function just for show
%-----------------------------------------------------------------------------
n=length(z2);                                    %length of padded data
m=fix( n/2 -1/4);                          %incomprehensive number
      ff3=comp.fs/ n*(1: m+2)';            %frequency vector
      P=i* ff3/comp.fc;                    %normalized frequency

      H=alp2(comp.fA0,comp.a1,comp.b1,P); %complex filter function
      tit='ACC-I';                               %label              of
corresponding filter
%.............................................................................
% The ACC-I data have a high pass filter characteristics included as well
% high pass filter treatment

% Hhp=A_inf./(1+(a_1./P);                 %definition formula for hp filter
[Hhp]=ahp2(1,1,0,i* ff3/comp.fhc); % complex high pass filter function

[irp8]=hpirp(1,1,comp.fhc,ti);                  %high pass impulse response
% x3=y2;
% x3=daver(y2,3,4);                %typ=4 Hanning windows sliding average
% zero padding: padd with trailing zeros to have the right sample length
% z4=[x3' zeros(1,length(irp8)-1)]';
%.........................................................................
% y4=real(ifft(fft(y2-median(y2(1:xan)))./Hhp));
% y4=ACCEFIL(Hhp,y2-median(y2(1:xan)));
y4=ACCEFIL(1./Hhp,y2);
% y3=(deconv(z4,irp8));%filter restoration per deconvolution
%.........................................................................
% convert voltage data into charge and decelleration
%Ua=-Q/Ci %output voltage = -charge/integration capacitance
%Q=-Ua*Ci
% charge sensitivity 2271A cs=11.5; %pC/g
% charge sensitivity 2271A FM cs=13;
% charge sensitivity 2271A FS cs=12.55;
Ci=4.7e-9; %integrating capacitor in charge amplifier [F]=[As/V]
cs=13.e-12; %typical charge sensitivity of Endevco 2271A [C/g]=9.81[As^2/m]
acc_scal=Ci/cs*9.81;    %voltage to ms^-2 scaling factor
ac1=-y1*acc_scal; %[ms^-2];
ac2=-y2*acc_scal; %[ms^-2];
% ac3=-y3*acc_scal; %[ms^-2];
ac4=-y4*acc_scal; %[ms^-2];

mac1=median(ac1(1:xan));      %front offset raw
mac2=median(ac2(1:xan));      %front offset smooth data
mac4=median(ac4(1:xan));      %front offset smooth data

ac1=ac1-mac1;           %remove offset
ac2=ac2-mac2;           %remove offset
ac4=ac4-mac4;           %remove offset
%.........................................................................
%-------------------------------------------------------------------------
% from here on graphical output only
%-------------------------------------------------------------------------
xax=ti*1000;                       %time in miliseconds
xlab='Time [ms]';            %time as x axis
sensn='ACC-I';
%-------------------------------------------------------------------------
% impulse response
pc0=10;                                  %graph window counters
pc=1;
ae(pc).fi=figure(pc+pc0);          %open new graph window
set(ae(pc).fi,'name','Impulse response','Position',[310 190 700 500]);

      ae(pc).p=plot(xax,irp7,'.b-');

      ae(pc).ti=title('Impulse response');
      ae(pc).le=legend('ACC-I filter',1);
      ae(pc).yl=ylabel('Amplitude [V]','fontsize',11,'Fontweight','demi');
      ae(pc).xl=xlabel(xlab,'fontsize',11,'Fontweight','demi');
            set(gca,'fontsize',11,'Fontweight','demi');
            set(gca,'linewidth',2)
            set(ae(pc).le,'fontsize',11,'Fontweight','demi');
            set(ae(pc).ti,'fontsize',11,'Fontweight','demi');
            set(ae(pc).p,'LineWidth',2.);
 set(gca,'xlim',[0.,60])

      grid on;
      zoom on
```

```
%     axis tight
%..............................................................
% hp bode diagram
pc=pc+1;
ae(pc).fi=figure(pc+pc0);
set(ae(pc).fi,'name','High  pass  bode  diagram','Position',[290  170  700
500]);

opt.fc=comp.fc;         %set options for bode plot
opt.lincol='r';
opt.linestyle='-.';
opt.a0=comp.fA0;

[mh,imh]=max(abs(H));                              %gain maximum in H
ae(pc).bd=ibodeplot(ff3,H,pc+pc0,'db',tit,opt);%create bode plot of filter
%..............................................................
% lp bode diagram
pc=pc+1;
ae(pc).fi=figure(pc+pc0);
set(ae(pc).fi,'name','Low pass bode diagram','Position',[290 170 700 500]);

opt.fc=1.7;       %set options for bode plot
opt.lincol='r';
opt.linestyle='-.';
opt.a0=1.;

[mh,imh]=max(abs(Hhp));                            %gain maximum in H
ae(pc).bd=ibodeplot(ff3,Hhp,pc+pc0,'db',tit,opt);    %create  bode  plot  of
filter
%..............................................................
% ACCI bode diagram
pc=pc+1;
ae(pc).fi=figure(pc+pc0);
set(ae(pc).fi,'name','Bode diagram','Position',[290 170 700 500]);

clear opt;
opt.fc=comp.fc;         %set options for bode plot
opt.lincol='r';
opt.linestyle='-.';
opt.a0=comp.fA0;

[mh,imh]=max(abs(Hhp));                            %gain maximum in H
ae(pc).bd=ibodeplot(ff3,H.*Hhp,pc+pc0,'db',tit,opt); %create  bode  plot  of
filter
%..............................................................
% step response
pc=pc+1;
ae(pc).fi=figure(pc+pc0);
set(ae(pc).fi,'name','Step response','Position',[270 150 700 500]);

n2=250;
n3=320;
      ae(pc).p=plot(xax(n2:n3),x(n2:n3),'.b-');
hold on
      ae(pc).p2=plot(xax(n2:n3),z3(n2:n3),'.r-');
hold off

      ae(pc).ti=title('ACC-I step response');
      ae(pc).le=legend('Step','Step response',4);
      ae(pc).yl=ylabel('Amplitude [V]','fontsize',11,'Fontweight','demi');
      ae(pc).xl=xlabel(xlab,'fontsize',11,'Fontweight','demi');
            set(gca,'fontsize',11,'Fontweight','demi');
            set(gca,'linewidth',2)
            set(ae(pc).le,'fontsize',11,'Fontweight','demi');
            set(ae(pc).ti,'fontsize',11,'Fontweight','demi');
%           set(ae(pc).p,'LineWidth',2.);
grid on;
      zoom on

%...........................................................................
%power spectrum        compute and plot power spcectrum of ACC-E data
pc=pc+1;
ae(pc).fi=figure(pc+pc0);
set(ae(pc).fi,'name','Power spectrum','Position',[250 150 700 500]);

%     xff=fft(xb);      %FFT of ACC-E raw data
[pte1,fpe]=pburg(x1,25,1024*2,fs);  %spectral power spectral density
[pte2,fpe]=pburg(x2,25,1024*2,fs);  %spectral power spectral density
[pte3,fpe]=pburg(y1,25,1024*2,fs);  %spectral power spectral density
[pte4,fpe]=pburg(y2,25,1024*2,fs);  %spectral power spectral density
[pte5,fpe]=pburg(y4,25,1024*2,fs);  %spectral power spectral density

            ae(pc).p=plot(fpe,mkdb(pte1)/2,'b-');
      hold on
            ae(pc).p2=plot(fpe,mkdb(pte2)/2,'m-');
            ae(pc).p3=plot(fpe,mkdb(pte3)/2,'c-.');
            ae(pc).p4=plot(fpe,mkdb(pte4)/2,'r-.');
            ae(pc).p5=plot(fpe,mkdb(pte5)/2,'k-.');
      hold off
%     ,'fontsize',11,'Fontweight','demi'
            ae(pc).ti=title('Burg PSD Estimate');
      ae(pc).le=legend('(1) raw data'...
                              ,'(2) smoothed data'...
                              ,'(3) restored raw'...
                              ,'(4) restored smooth',1);
            ae(pc).yl=ylabel('Power Spectral Density (dB/Hz)'...
                  ,'fontsize',11,'Fontweight','demi');
            ae(pc).xl=xlabel('Frequency [Hz]'...
                  ,'fontsize',11,'Fontweight','demi');
            set(gca,'fontsize',11,'Fontweight','demi');
            set(gca,'linewidth',2)
            set(ae(pc).le,'fontsize',11,'Fontweight','demi');
            set(ae(pc).ti,'fontsize',11,'Fontweight','demi');
            set(ae(pc).p,'LineWidth',2.);
            set(ae(pc).p4,'LineWidth',2.);
      grid on;
      zoom on
%...........................................................................
% raw and smoothed signal
pc=pc+1;
ae(pc).fi=figure(pc+pc0);
set(ae(pc).fi,'name','ACC-I raw signal','Position',[230 130 700 500]);

      ae(pc).p=plot(xax,x1,'b');
hold on
      ae(pc).p2=plot(xax,x2,'r');
hold off

      ae(pc).ti=title('ACC-I raw signal');
      ae(pc).le=legend('(1) raw data'...
                              ,'(2) smoothed data',1);
      ae(pc).yl=ylabel('Amplitude [V]','fontsize',11,'Fontweight','demi');
      ae(pc).xl=xlabel(xlab,'fontsize',11,'Fontweight','demi');
            set(gca,'fontsize',11,'Fontweight','demi');
            set(gca,'linewidth',2)
            set(ae(pc).le,'fontsize',11,'Fontweight','demi');
            set(ae(pc).ti,'fontsize',11,'Fontweight','demi');
            set(ae(pc).p,'LineWidth',2.);
            set(ae(pc).p2,'LineWidth',2.);

      grid on;
      zoom on
%     axis tight
%...........................................................................
% restored signal
pc=pc+1;
ae(pc).fi=figure(pc+pc0);
set(ae(pc).fi,'name','ACC-I signal','Position',[210 110 700 500]);

      ae(pc).p=plot(xax,y1-median(y1(1:xan)),'b');
hold on
      ae(pc).p2=plot(xax,y2-median(y2(1:xan)),'r');
%     ae(pc).p3=plot(xax,y3,'g');
      ae(pc).p4=plot(xax,y4-median(y4(1:xan)),'g');
hold off
      ae(pc).ti=title('ACC-I Signal');
      ae(pc).le=legend('(1) lp restored raw data'...
                              ,'(2) lp restored smoothed'...
                              ,'(3) hp restored',4);
      ae(pc).yl=ylabel('Amplitude [V]','fontsize',11,'Fontweight','demi');
      ae(pc).xl=xlabel(xlab,'fontsize',11,'Fontweight','demi');
            set(gca,'fontsize',11,'Fontweight','demi');
            set(gca,'linewidth',2)
            set(ae(pc).le,'fontsize',11,'Fontweight','demi');
            set(ae(pc).ti,'fontsize',11,'Fontweight','demi');
            set(ae(pc).p,'LineWidth',2.);
            set(ae(pc).p2,'LineWidth',2.);
            set(ae(pc).p4,'LineWidth',2.);

      grid on;
      zoom on
%     axis tight
%...........................................................................
% restored decelleration signal
pc=pc+1;
ae(pc).fi=figure(pc+pc0);
set(ae(pc).fi,'name','ACC-I deceleration signal','Position',[180 80 700
500]);

      ae(pc).p=plot(xax,ac1,'b');
hold on
      ae(pc).p2=plot(xax,ac2,'r');
%     ae(pc).p3=plot(xax,ac3,'g');
      ae(pc).p4=plot(xax,ac4,'g');
hold off

%     ae(pc).lc1=line([min(xax),xax(xan)],[mac1,mac1] ...
%           ,'color','c','linestyle',opt.linestyle,'LineWidth',1.5);
%
%     ae(pc).lc2=line([min(xax),xax(xan)],[mac2,mac2] ...
%           ,'color','m','linestyle',opt.linestyle,'LineWidth',1.5);

      ae(pc).ti=title('ACC-I Decelleration Signal');
      ae(pc).le=legend('(1) restored raw data'...
                              ,'(2) restored low pass'...
                              ,'(3) restored high pass',1);
      ae(pc).yl=ylabel('Decelleration [ms^-^2]',
'fontsize',11,'Fontweight','demi');
      ae(pc).xl=xlabel(xlab,'fontsize',11,'Fontweight','demi');
```

```
            set(gca,'fontsize',11,'Fontweight','demi');
            set(gca,'linewidth',2)
            set(ae(pc).le,'fontsize',11,'Fontweight','demi');
            set(ae(pc).ti,'fontsize',11,'Fontweight','demi');
            set(ae(pc).p,'LineWidth',2.);
            set(ae(pc).p2,'LineWidth',2.);
            set(ae(pc).p4,'LineWidth',2.);

      grid on;
      zoom on
%     axis tight
%..........................................................................
```

4.3 Subroutines

4.3.1 Get component values: getcomp

```
function  comp=getcomp( componentfile,echo );
% read components values from filter definiteion files
% calling sequence:
%           comp=getcomp( componentfile,echo );
%                 componentfile:    complete filename of definition file
%                 echo:             'echo' enables procedure echo display
% G. Kargl, IWF Graz, Jan. 2005
%............................................................................

%echo the selected file
if nargin==2;
      disp('Read component dat fom file:');
      disp(componentfile)
end

% load filter hardware data
% fid=fopen('C:\Projekte\ACC-E\Mfiles\ACCEfil.dat');
fid=fopen(componentfile);

fseek(fid,0,'eof');           %look for end of file
eof=ftell(fid);               %get eof position
fseek(fid,0,'bof');           %go back to begin of file
fpos=ftell(fid);        %get bof position

while fpos < eof        %read electronic components value from file
      line=fgetl(fid);
      eval(['comp.' line]);
      fpos=ftell(fid);
end
fclose(fid);                  %close component data file

%............................................................................
%display components values
if nargin==2;
      comp                    %display component value structure variable
end
```

4.3.2 Low-Pass impulse response: irp

```
function irp7=irp(a0,a1,b1,fg,ti);
%filter impulse responsefor second order filter
% G. Kargl, IWF Graz, Mar. 2005

irp1=2*a0*fg*pi*(1+sign(ti));
irp2=sqrt(a1^2 - 4*b1);
irp3=sinh(irp2*fg*pi.*ti/b1);
irp4=exp(a1*fg.*ti/b1);

irp5=irp1./irp4;
irp6=irp3./irp2;
irp7=irp5.*irp6;

irp7=irp7./sum(irp7)*a0;            %weighted impulse response of filter
```

4.3.3 Low-pass filter function: alp2

```
function [APLP]=alp2(A0,a1c,b1c,P)
%---------------------------------------------------------------------------
%computes transfer function for second order active low pass filter
%     G. Kargl, IWF 1998
% negative gain (- A0) -> (A0) removed for more a general filter function
%     G. Kargl, IWF, nov. 2004
%---------------------------------------------------------------------------
% Calling sequence [APLP]=alp2(A0,a1c,b1c,P)
%active low pass filter second order

%a1=1.4142;                  %Second order Butterworth
%b1=1.;                            %      -- " --

APLP=A0./(1+a1c*P+(b1c *P.^2));
%...........................................................................
```

4.3.4 Bode plot graphic package: ibodeplot

```
function bd=ibodeplot(f,H,fig,style,tit,opt);
% bode plot of  complex filter function
% callin sequence:
%     bd=ibodeplot(f,H,fig,style);
%     f: frequency vector
%     H: Filter function
%     fig: figure window to plot to
%     style:      'abs' absolute values
%                 'db' in dezibel units
% G. Kargl, IWF Graz, Dec. 2004

%----------------------------------------------------------------------------
if fig~=0;
      bd.hfig=figure(fig);
end

a=abs(H);
ph=angle(H)*180/pi;
if not(exist('tit'))
      tit='';
end

if not(isfield(opt,'a0'))
            opt.a0=max(a);
end

if exist('style')
      switch style
            case 'db'
                  a=mkdb(a,opt.a0);
                  ylab='db';
            otherwise
                  a=a;
                  ylab='arb units';
      end
end

if not(exist('style'));
      tit='';
      dummy='';
```

```
else
      dummy=': ';
end

ds=size(H);

%data amplitude
bd.sp1=subplot(2,1,1);

      bd.ap=semilogx(f,a,'LineWidth',2);

      title(['Bode diagram' dummy tit],'Fontweight','demi','fontsize',11);
            bd.axlab=xlabel('Frequency
[Hz]','Fontweight','demi','fontsize',11);
            bd.aylab=ylabel(['Amplitude [' ylab ']'
],'Fontweight','demi','fontsize',11);
      grid on
      zoom on
            set(gca,'fontsize',11,'Fontweight','demi')
            set(gca,'linewidth',2)

      xlim=get(bd.sp1,'xlim');
if exist('opt')
      if isfield(opt,'fc')
            ylim=get(bd.sp1,'ylim');
      end
      if isfield(opt,'lincol')
            lincol=opt.lincol;
      else
            lincol='g';
      end
      if isfield(opt,'linestyle')
            linestyle=opt.linestyle;
      else
            linestyle='-';
      end
      bd.lc=line([opt.fc,opt.fc],[ylim(1),ylim(2)] ...
            ,'color',lincol,'linestyle',linestyle,'LineWidth',1.5);
end
            %data phase
bd.sp2=subplot(2,1,2);
      bd.pp=semilogx(f,ph,'LineWidth',2);
            miph=min(ph);
            mph=max(ph);
            phv=[0;15;30;60;120;180];
            tol=[0; 1; 2; 2;  5;  5];
            maph=sign(mph)*phv(min(find((mph)<=phv)));
            miph=sign(miph)*phv(max(find(abs(miph)>=phv)));
%           axis([xlim(1),xlim(2),miph,maph]);
            axis([xlim(1),xlim(2),-180,180]);

            bd.pxlab=xlabel('Frequency
[Hz]','Fontweight','demi','fontsize',11);
            bd.pylab=ylabel(['Phase [°]'
],'Fontweight','demi','fontsize',11);
      grid on
      zoom on
            set(gca,'fontsize',11,'Fontweight','demi')
            set(gca,'linewidth',2)
```

4.3.5 High-pass impulse response: hpirp

```
function [irp8]=hpirp(a0,a1,fg,ti);
% impulse response of first order high pass filter
% caling sequence:
%     [irp8,ti0i]=hpirp(a0,a1,fg,ti);
%           a0:         filter gain
%           a1:   first filter factor
%           fg:         high pass corner frequency
%           ti:         time vector (sampling interval dependend)
%           irp8: filter impulse response
%           ti0i: index of ti > 0 points
% G. Kargl, IWF, 6. 4. 2005

ti0i=find(ti > 0);
ti=ti(ti0i);

irp1=2*a0*fg*pi*ti;
irp2=(-cosh(2*a1*fg*pi.*ti) + sinh(2*a1*fg*pi.*ti));

irp8=irp1.*irp2./ti;
% irp8=-irp8./sum(irp8);            %weighted impulse response of filter

% figure(2)
% plot(ti,irp8,'.r-')
```

4.3.6 Averaging routine: daver

```
function [adata,mdata,stdata,type,aver]=daver(data,n,sel)
%digital filtering of one dimensional data by convoluting kernel with data
%call [adata,mdata,stdata]=daver(data,n,sel)
%     adata == smothed data
%     mdata == mean of data
%     stdata == standard deviation of data
%     aver == filter function
%     data == data to be smoothed
%     n == filter broadness; n should be an odd number e.g. n=7
%     sel == select filter type
%aver is the averaging filter function
%sel=1 aver=ones(1,n);                                   %Sliding average
%sel=2 aver=cos(((0:n-1)./(n-1)*pi/2)-pi/4);   %Cosine filter
%sel=3 aver=hamming(n);                                  %Hamming filter
%sel=4 aver=hanning(n);                                  %Hanning filter
%sel=5 medfilt1(data,n);                                 %Median filter
%
%adata=conv(data(a+naver:e-naver),aver)/saver; %apply filter to signal
%
% created: 14. june. 1999
%     Günter Kargl, IWF
% revisions:
%                 median filter added, 21. 5. 2004, G.K.
%                 minimum smoothing length n=3 added, 01. 03. 2005, G.K.
%==========================================================================
==

      e=length(data);         %start and stop index of signal/frequency
if n<3
      disp('Minimum smoothung length n=3!');
      n=3;                                        %n point smoothing
end
switch sel
```

```
      case 1
            aver=ones(1,n);               %sliding average
            saver=sum(aver);              %sum of averaging factor
            type='sliding average';
            adata=conv(data,aver/saver);  %apply filter to signal /saver
      case 2
            aver=cos(((0:n-1)./(n-1)*pi/2)-pi/4);    %cosine filter
            saver=sum(aver);                    %sum of averaging factor
            type='cosine';
            adata=conv(data,aver/saver);  %apply filter to signal /saver
      case 3
            aver=hamming(n);                    %haming filter
            saver=sum(aver);                    %sum of averaging factor
            type='hamming';
            adata=conv(data,aver/saver);  %apply filter to signal /saver
      case 5                                    %median filter
            aver=1;
            adata=medfilt1(data,n);
            type='median';
      otherwise
            aver=hanning(n);                    %hanning filter
            saver=sum(aver);                    %sum of averaging factor
            type='hanning';
            adata=conv(data,aver/saver);  %apply filter to signal /saver
  end;
            naver=ceil(length(aver)/2);

adata=adata(naver:e+naver-1); %select data core (where averaging took place)
%add original unsmoothed data on fringes
adata=[data(1:naver).'     adata(naver+1:length(adata)-naver).'     data(e-
naver+1:e).'].';

mdata=mean(data);                         %mean signal
stdata=std(data);                         %standard deviation of signal
%-----------------------------------------------------------------------------
```