Chapter 6

Real Life Application

MEANWHILE, we are in the position to show some so called *real life applications* of M. We will show two examples, both using the *Inductively Coupled Plasma Optical Emission Spectroscopy* (ICP-OES). It is a very important technique for elemental analysis and besides *Titration* it is one of the first methods that was implemented in *Uncertainty Manager*. We will try to explain the principles of the ICP-OES such that non-experts will be able to understand the M-Code of the examples. There are two examples shown in the following sections.

We are grateful to Marc Salit from NIST and to Alfred Jacob from the Laboratory Spiez for providing the data of the two measurements. The former is the determination and certification of bismuth content in a reference material (Section 6.1) and the latter is the determination of calcium in spring water (Section 6.2). The determination of calcium content was done using the β-version of the software *Uncertainty Manager*.

Principle of ICP-OES

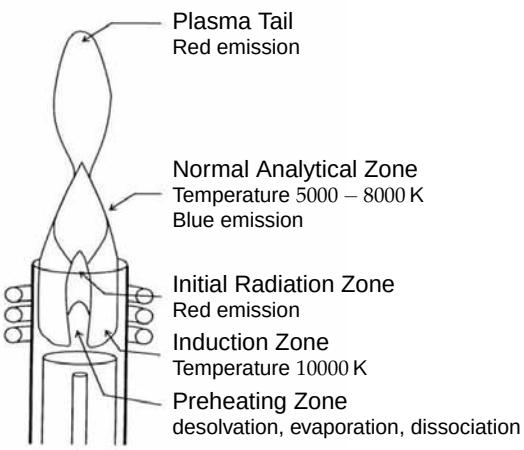
The following overview is a summary from the Web-pages at `http://www.icp-oes.com/`.

Inductively coupled plasma optical emission spectroscopy (ICP-OES or ICP-AES) is a major technique for elemental analysis. The sample to be analyzed, if solid, is normally first dissolved and then mixed with water before being fed into the plasma. Atoms in the plasma emit light (photons) with

characteristic wavelengths for each element. This light is recorded by one or more optical spectrometers and when calibrated against standards the technique provides a quantitative analysis of the original sample.

ICP instruments comprise various optical spectrometers, nebulizers, spray chambers, ICP torch, and Radio Frequency generators. When liquid is fed into an IC plasma it must be in the form of fine droplets otherwise the liquid will not fully dissolve and atomize. The first stage in forming droplets is the nebulizer. Most nebulizers are pneumatic, that is, they rely on the Venturi Effect. The device which produces the IC plasma is commonly referred to as the ICP torch. It consists of two to four Argon flows depending on the manufacturer. Figure 15 illustrates the structure of an IC torch.

Figure 15 Schematic Illustration of ICP-OES Torch (reprint with kind permission)



Nebulizer gas (inner Argon flow), at about $1\,l/min$, carries the analyte aerosol. Sheath gas is responsible for producing a laminar flow. Plasma gas, at about $12 - 16\,l/min$, sets the plasma conditions. The argon and analyte flow into a toroidal radio frequency field, usually at 40.68 MHz. The plasma is ignited by a Tesla spark. ICP's generally require $1 - 2\,kW$ maximum output at radio frequency (RF) to maintain the plasma. In other words, the RF

field induces energy into the plasma like your micro wave oven into your soup. Aerosol vapor is transported to the plasma, where vapor dissolves. Atomization occurs within the plasma, i.e. atoms get excited to atomic and ionic states. This process produces rich spectra because of presence of both atomic and ionic lines.

An optical system scatters the spectrum and a CCD-device measures the emitted lines and processes the result. Nowadays spectrometer optical systems consist of a polychromator which scatters the spectrum and isolates the analytical lines of the elements to be analyzed. This allows simultaneous measurement of emitted intensities of numerous elements.

As a matter of fact, the ICP-OES process is always subject to fluctuations during the measurement process due to torch temperature, droplet irregularities, pressure variations etc. These fluctuations are called *drift*. The possibility of being able to measure several elements' emissions simultaneously offers an option for correcting this drift: Besides observing the emitted intensity of the analyte(s) there is also an *Internal Standard* (IS) being observed. The difference between the expected intensity of the IS and the observed intensity is a good estimate for the drift on the intensity of the analyte(s).

6.1 Certification of Bismuth content in SRM 3106

6.1.1 NIST SRM Program

U.S. National Institute for Standards and Technology (NIST)[1] ensures accurate and compatible measurements by developing, certifying, and distributing *Standard Reference Materials* (SRMs). Approximately 1300 SRMs are currently available for use in:

1. industrial materials production and analysis
2. environmental analysis
3. health measurements
4. basic measurements in science and metrology

[1] `http://www.nist.gov`

The SRM Program implements policies and coordinates priorities for the development, production, and certification of SRMs and serves as the main contact point for all NIST reference materials activities interfacing with similar efforts in the private sector, other federal agencies, and other nations. This includes programs with the American Society for Testing Materials (ASTM), the International Organization for Standardization (ISO), the American Association for Clinical Chemistry, the International Union of Pure and Applied Chemistry AACC, and the European Economic Community.

In our example we show the measurement uncertainty evaluation of SRM 3106. The certificate header reads:

> This Standard Reference Material (SRM) is intended primarily for use in calibrating instruments used in atomic spectrometry, including atomic absorption spectrometry, inductively coupled plasma optical emission spectrometry, and inductively coupled plasma mass spectrometry. It can also be used in conjunction with any other analytical technique or procedure where an aqueous standard solution is required. One unit of SRM 3106 consists of five 10 ml sealed borosilicate glass ampoules of a single element solution prepared gravimetrically to contain a known amount of bismuth in an approximate nitric acid volume fraction of 10%.

6.1.2 High Performance ICP-OES

NIST Measurement protocol for High-Performance ICP-OES

The NIST Measurement protocol used is described in detail in [41]. In order to understand the M-code example 21 we will summarize the procedure very informally.

We would like to thank Marc Salit from NIST and Matthias Rösslein from EMPA for providing data and analytical explanations.

The general equation for obtaining the mass fraction in High-Performance ICP-OES is as follows

$$Massfraction = \frac{\left(I_{Analyte}/I_{IntStd}\right)_{unknown}}{\left(I_{Analyte}/I_{IntStd}\right)_{Calibrant}} \times \frac{\left(mass_{Analyte}/mass_{IntStd}\right)_{Calibrant}}{\left(mass_{sample}/mass_{IntStd}\right)_{unknown}}. \quad (6.1)$$

The applied method constructs 8 calibrant solutions from 4 parent solutions as thoroughly described in[41]. These 8 calibrant solutions are fitted against their intensities yielding an average slope for the mass fraction versus intensity fraction ratio. The average slope is then used to predict the expected mass fraction of 6 samples of the probe. And finally, the 6 expected mass fractions of the analyte are converted and averaged into concentration of bismuth in the probe.

Example 21 Input:

```
#-- masses of metal in parent solutions
m_met_cb1 = <1002.76[mg] : 0.02[mg]>;
m_met_cb2 = <1004.55[mg] : 0.02[mg]>;
m_met_tb1 = <1035.43[mg] : 0.02[mg]>;
m_met_tb2 = <998.67[mg] : 0.02[mg]>;

#-- purity of metals in parent solutions
p_met = <0.999976 : 0.000004>;

#-- total masses of parent solutions
m_solv_cb1 = <1004.53[g] : 0.004[g]>;
m_solv_cb2 = <1001.29[g] : 0.004[g]>;
m_solv_tb1 = <998.65[g] : 0.004[g]>;
m_solv_tb2 = <1003.43[g] : 0.004[g]>;

#--- aliquot weights
m_cb1_a = <34.732 [g] : 3 [mg]>;
m_cb1_b = <33.860 [g] : 3 [mg]>;
m_cb2_a = <34.197 [g] : 3 [mg]>;
m_cb2_b = <31.854 [g] : 3 [mg]>;
m_tb1_a = <35.512 [g] : 3 [mg]>;
m_tb1_b = <38.470 [g] : 3 [mg]>;
m_tb2_a = <39.335 [g] : 3 [mg]>;
m_tb2_b = <38.179 [g] : 3 [mg]>;

#-- build the mass ratios for calibrants
function m_rat(aliq, met, purity, solv, is) {
        ((aliq * met * purity) / solv) / is;
};

```

```
# -- mass of analyte in mass of solvent
m_rat_cb1_a = m_rat(m_cb1_a, m_met_cb1, p_met, m_solv_cb1, m_is_cb1_a);
m_rat_cb1_b = m_rat(m_cb1_b, m_met_cb1, p_met, m_solv_cb1, m_is_cb1_b);
m_rat_cb2_a = m_rat(m_cb2_a, m_met_cb2, p_met, m_solv_cb2, m_is_cb2_a);
m_rat_cb2_b = m_rat(m_cb2_b, m_met_cb2, p_met, m_solv_cb2, m_is_cb2_b);
m_rat_tb1_a = m_rat(m_tb1_a, m_met_tb1, p_met, m_solv_tb1, m_is_tb1_a);
m_rat_tb1_b = m_rat(m_tb1_b, m_met_tb1, p_met, m_solv_tb1, m_is_tb1_b);
m_rat_tb2_a = m_rat(m_tb2_a, m_met_tb2, p_met, m_solv_tb2, m_is_tb2_a);
m_rat_tb2_b = m_rat(m_tb2_b, m_met_tb2, p_met, m_solv_tb2, m_is_tb2_b);

# -- internal standard
m_is_cb1_a = <3.028 [g] : 3[mg]>;
m_is_cb1_b = <3.035 [g] : 3[mg]>;
m_is_cb2_a = <3.048 [g] : 3[mg]>;
m_is_cb2_b = <3.064 [g] : 3[mg]>;
m_is_tb1_a = <3.039 [g] : 3[mg]>;
m_is_tb1_b = <3.059 [g] : 3[mg]>;
m_is_tb2_a = <3.059 [g] : 3[mg]>;
m_is_tb2_b = <3.052 [g] : 3[mg]>;

# -- Intensities
i_rat_cb1_a = < 0.11509 : 0.0000542 >;
i_rat_cb1_b = < 0.11192 : 0.0000315 >;
i_rat_cb2_a = < 0.11250 : 0.0000595 >;
i_rat_cb2_b = < 0.10421 : 0.0001048 >;
i_rat_tb1_a = < 0.11717 : 0.0000414 >;
i_rat_tb1_b = < 0.12608 : 0.0000756 >;
i_rat_tb2_a = < 0.12885 : 0.0000664 >;
i_rat_tb2_b = < 0.12536 : 0.0000499 >;

yvals = (i_rat_cb1_a, i_rat_cb1_b, i_rat_cb2_a, i_rat_cb2_b,
         i_rat_tb1_a, i_rat_tb1_b, i_rat_tb2_a, i_rat_tb2_b);

xvals = (m_rat_cb1_a, m_rat_cb1_b, m_rat_cb2_a, m_rat_cb2_b,
         m_rat_tb1_a, m_rat_tb1_b, m_rat_tb2_a, m_rat_tb2_b);

# -- calibrate
cali = fit(type = 0, xvals, yvals);
f = get(inv,cali);
```

```
# eval(cali);

# -- these are the measured, drift corrected, intensity ratios and their
#    standard deviations (not std. unc)
i_rat_srmA_A = <0.10639 : 0.00010>;
i_rat_srmA_B = <0.09901 : 0.00005>;
i_rat_srmB_A = <0.09995 : 0.00007>;
i_rat_srmB_B = <0.10048 : 0.00004>;
i_rat_srmC_A = <0.10021 : 0.00006>;
i_rat_srmC_B = <0.09955 : 0.00005>;

# -- these are the masses of sample taken
m_anal_srmA_A = <3.213[g] : 3[mg] >;
m_anal_srmA_B = <2.979[g] : 3[mg] >;
m_anal_srmB_A = <3.017[g] : 3[mg] >;
m_anal_srmB_B = <3.039[g] : 3[mg] >;
m_anal_srmC_A = <3.022[g] : 3[mg] >;
m_anal_srmC_B = <3.023[g] : 3[mg] >;

# -- these are the masses of internal standard solution
m_IS_srmA_A = <3.030[g] : 3[mg] >;
m_IS_srmA_B = <3.022[g] : 3[mg] >;
m_IS_srmB_A = <3.033[g] : 3[mg] >;
m_IS_srmB_B = <3.037[g] : 3[mg] >;
m_IS_srmC_A = <3.029[g] : 3[mg] >;
m_IS_srmC_B = <3.047[g] : 3[mg] >;

# -- predicted mass ratios
m_rat_hat_srmA_A = f( i_rat_srmA_A );
m_rat_hat_srmA_B = f( i_rat_srmA_B );
m_rat_hat_srmB_A = f( i_rat_srmB_A );
m_rat_hat_srmB_B = f( i_rat_srmB_B );
m_rat_hat_srmC_A = f( i_rat_srmC_A );
m_rat_hat_srmC_B = f( i_rat_srmC_B );

#-- function to calculate the concentration from
# the calibration and the masses
function calc_conc(pred_ratio, m_IS, m_Sample ) {
        (pred_ratio * m_IS / m_Sample);
```

```
};

# -- calculate the concentrations
conc_srmA_A = calc_conc( m_rat_hat_srmA_A, m_IS_srmA_A, m_anal_srmA_A );
conc_srmA_B = calc_conc( m_rat_hat_srmA_B, m_IS_srmA_B, m_anal_srmA_B );
conc_srmB_A = calc_conc( m_rat_hat_srmB_A, m_IS_srmB_A, m_anal_srmB_A );
conc_srmB_B = calc_conc( m_rat_hat_srmB_B, m_IS_srmB_B, m_anal_srmB_B );
conc_srmC_A = calc_conc( m_rat_hat_srmC_A, m_IS_srmC_A, m_anal_srmC_A );
conc_srmC_B = calc_conc( m_rat_hat_srmC_B, m_IS_srmC_B, m_anal_srmC_B );

avg_conc = (conc_srmA_A + conc_srmA_B + conc_srmB_A +
            conc_srmB_B + conc_srmC_A + conc_srmC_B ) / 6;

# -- perform the ISO evaluation
iso(avg_conc);

# -- perform a Monte Carlo Simulation
mc(avg_conc, file="bismuth.mc", size=100000);
```

Example 21: The data in Lines 2–14 is determining the mass fraction of the calibrant.

The function `m_rat` on Lines 27ff. is used to construct the $mass_{Analyte}$ in the following block Lines 32–39. The mass fraction in Equation (6.1) is thus constructed 8 times starting from 4 calibrant solutions.

The data in Lines 42–49 comprises the $mass_{IntStd}$ and the Lines represent the means of 5 readings each. These are the intensity ratios $I_{Analyte}/I_{IntStd}$ used in the calibration process.

Lines 61–69 define and prepare the calibration function `f` which in this case is a 0-linear function, i.e. with no y-axis interception. The commented line 70 is useful for observing the calibration. If `eval(cali)` is executed, it displays the complete attribute list representing the entire calibration process. This is in general a very lengthy expression. For demonstration purpose its output is shown in the output section. The attribute `iso_p0` contains the resulting parameter by the iso-calibration, i.e. the XIP-FIT. The attribute `mc_p0` contains the result obtained by the Monte Carlo Simulator.

Lines 74–79 contain the data of the *drift corrected* intensities. Again they are averages of 5 readings each. The drift correction itself is not done in this M-script.

Lines 82–95 contain the data for $mass_{Sample}$ and $mass_{IntStd}$.

Lines 98–103 convert the observed — and drift corrected — intensities of the SRM probes to expected mass fractions yielding $(mass_{sample}/mass_{IntStd})_{unknown}$.

Function `calc_conc` on Lines 107ff. and 112–117 computes the (measured) concentration of analyte in the SRM probe.

Line 119 constructs the average of the 6 estimates.

Line 123 calls the ISO-evaluator to evaluate the average concentration and Line 126 calls the Monte Carlo Simulator to do the same (and thereby writing the generated sample to the file `bismuth.mc` for later visualization).

The following output has been slightly edited. It basically shows the output of the expression `eval(cali)` as mentioned above. Line 34 is the result of the ISO-evaluator. Lines 35 and 36 are the result of the Monte Carlo Simulation. Note that the two results agree up to the first 2 significant figures. This is less than expected with a sample size of 1′000′000 in the Monte Carlo Simulation. The difference between the two results therefore reflects the difference between Berkson's Model and the Regression done by the Monte Carlo Simulation.

Finally there is a warning in the output of this example. This is due to the fact that the TSVD-solver in the XIP-FIT procedure did not reach the expected precision. Often, this is due to bad scaling of the input calibration data. The calibration object `cali` contains two attributes `scalex` and `scaley` by which each `xval` and `yval` element might be scaled. However, in this particular example scaling will not help because this calibration data set intrinsically leads to an ill conditioned Jacobian during the XIP-FIT.

Example 21 Output:

```
...
"Regression"(type = 0,( fun=sfunction(x) {p0 * x;},
            inv=sfunction(x) {x / p0;}, iso_p0=<9.947 : 0.00392813>,
            mc_fingerprint=9324, mc_p0= <9.93685 : 0.0039901>, parnum=1,
5           scalex=0.0130389, scaley=0.12885,
            xvals=((((<34.732 : 0.003>[g] * <1.00276 : 2e-005>[g]
                            ) * <0.999976 : 4e-006>) /
                            <1.00453 : 4e-006>[kg]) / <3.028 : 0.003>[g],
                            (((<33.86 : 0.003>[g] * <1.00276 : 2e-005>[g])
10                           * <0.999976 : 4e-006>) /
                            <1.00453 : 4e-006>[kg]) / <3.035 : 0.003>[g],
                            (((<34.197 : 0.003>[g] * <1.00455 : 2e-005>[g]
                            ) * <0.999976 : 4e-006>) /
                            <1.00129 : 4e-006>[kg]) / <3.048 : 0.003>[g],
15                          (((<31.854 : 0.003>[g] * <1.00455 : 2e-005>[g]
                            ) * <0.999976 : 4e-006>) /
```

```
                                <1.00129 : 4e-006>[kg]) / <3.064 : 0.003>[g],
                                (((<35.512 : 0.003>[g] * <1.03543 : 2e-005>[g]
                                ) * <0.999976 : 4e-006>) / <998.65 : 0.004>[g]
                                ) / <3.039 : 0.003>[g], (((<38.47 : 0.003>[g]
                                 * <1.03543 : 2e-005>[g]) *
                                <0.999976 : 4e-006>) / <998.65 : 0.004>[g]) /
                                <3.059 : 0.003>[g], (((<39.335 : 0.003>[g] *
                                <998.67 : 0.02>[mg]) * <0.999976 : 4e-006>) /
                                <1.00343 : 4e-006>[kg]) / <3.059 : 0.003>[g],
                                (((<38.179 : 0.003>[g] * <998.67 : 0.02>[mg])
                                 * <0.999976 : 4e-006>) /
                                <1.00343 : 4e-006>[kg]) / <3.052 : 0.003>[g])
                    , yvals=(<0.11509 : 5.42e-005>, <0.11192 : 3.15e-005>,
                             <0.1125 : 5.95e-005>, <0.10421 : 0.0001048>,
                             <0.11717 : 4.14e-005>, <0.12608 : 7.56e-005>,
                             <0.12885 : 6.64e-005>, <0.12536 : 4.99e-005>)));
...
<0.0100943 : 6.31456e-006>;
"MonteCarloSimulation"(file="bismuth.mc", result=<0.0101047 : 7.53728e-006>,
                        size=10000);
Warning:
CEvalIso: <#508#> REGRESSION: tol=1.000000e-008 not reached.
        Try scaling your x and/or y values to improve results.
```

Figures 16–18 show the measurement report for the "Determination of Bi in SRM 3106" which was executed January 21, 2000 by Marc Salit. It attests the SRM 3106 a mass fraction of 10.015 mg/g of bismuth with an uncertainty of 0.007 mg/g. This result was obtained by yet another method of calibration: averaging the calibrants' slopes using Microsoft® Excel. The uncertainty matches nicely between the two estimates by the ISO-Evaluator and the Monte Carlo Simulation, whereas the deviation from the "recommended value" is slightly bigger than expected.

6.2 Measuring calcium content in spring water

As might have become clear in Example 21, standard measurement procedures tend to become very complex very quickly in terms of handling all data. In the course of evaluating the β-version of *Uncertainty Manager* a number of real life measurements were executed at several laboratories. The following application was done by Matthias Rösslein from EMPA together

Figure 16 NIST Report 839.01-00-518, Pages 1 and 2

U.S. Department of Commerce
National Institute of Standards and Technology
Chemical Science and Technology Laboratory
Analytical Chemistry Division
Gaithersburg, MD 20899

REPORT OF ANALYSIS

January 21, 2000

Submitted to: Willie E. May, Chief
Analytical Chemistry Division

Author: M.L. Salit and A.P. Lindstrom

Title: Determination of Bi in SRM 3106, Lot 991212

Constituent: Bi

Method: Inductively Coupled Plasma Optical Emission Spectrometry (ICP-OES).

Copies to: SRM File

Background: To value assign mass fraction levels for Bi in the single element spectrometric solution SRM for certification of this material.

Samples: Three samples of the candidate SRM were received in glass ampoules from Therese Butler. These ampoules contained clear solutions, and were labeled as "SRM-A," "SRM-B," and "SRM-C" after receipt.

Standards: NIST Primary Solutions (NPS) described in ROAs 839.01-00-515 and 839.01-00-516 were used as calibration standards. Two pre-weighed aliquots of each of four different Bi NPS standards were used.

Analytical Method: The emission signals from the samples were measured with an Inductively Coupled Plasma-Optical Emission Spectrometer (ICP-OES). The signal measured from the candidate SRM solutions was compared to the signal measured from the NPS standards. The mass fraction of analyte in the candidate SRM is inferred from this comparison.

Analytical Protocol: Two aliquots of each sample were weighed into two new LDPE bottles. Internal standard solution was weighed into each bottle immediately after addition of sample. The same internal standard solution was weighed into each of the 8 NPS bottles selected for calibration. The bottles were vigorously mixed. Effort was made to maintain a relatively constant (±10%) ratio of mass of Bi to mass of internal standard solution, to minimize possible bias due to emission signal non-linearity. The mass fraction of the internal standard solution was selected to yield an appropriate mass fraction ratio for simultaneous measurement. These solutions underwent subsequent dilution to yield samples for analysis at the appropriate mass fraction levels. These sample preparations are described in Table I.

Autosampler tubes were filled with the diluted samples, and were randomly ordered for the analysis. The experiment captures sample-to-sample variability (arising from heterogeneity), calibrant-to-calibrant calibration factor variability, and the preparation variability (arising from sample handling).

ICP-OES Measurements: The ICP-OES system employed for this analysis was a Perkin-Elmer Optima 3300 DV, which is composed of a free-running 40 MHz ICP, axially viewed, and an Echelle spectrometer with segmented-CCD array detection. ICP wavelengths and integration times are presented in Table I, while instrument-operating parameters are detailed in Table II.

ICP-OES Data Reduction: Raw intensity data (in units of counts-per-second) were transferred to a Microsoft Excel™ spreadsheet, where internal standard ratios were calculated. Short-term precision is calculated before and after internal standardization as the relative standard deviation of the separate integrations, but these integrations are averaged for continued calculations.

A drift correction procedure[1] is applied to the internally standardized data for each element. Then the data from the repeated measurements are summarized, and the mass fraction of analyte is calculated for each solution.

Results: The results of this analysis are reported in Table III. This table includes sections for the recommended value, the Type A & B uncertainty components, and the combined and expanded uncertainties.

The 6 samples prepared from 3 ampoules of the candidate SRM material were treated as independent measures of the SRM mass fraction. The different NPS standards yielded consistent calibration factors, and were treated as 8 measures of the calibration factor.

The type A uncertainties are the standard uncertainties for the mean SRM signal and the mean calibration factor. The type B uncertainty is the uncertainty associated with the purity of the

[1] M.L. Salit and G.C. Turk, "A Drift Correction Procedure," *Anal Chem* **70,** 3184-3190 (1998).

Figure 17 NIST Report 839.01-00-518, Pages 3 and 4

calibrant material. Variability in measurement appears to dominate this analysis, as reflected in the Type A uncertainty estimates. The expanded uncertainty of 0.07 % is as small as could be expected with this measurement approach.

Original data are recorded in laboratory notebook APL III, pp. 39-63. Calculations and data analysis were performed using the spreadsheet file "000103 Bi Results.xls".

Prepared and Analyzed by:

Marc L. Salit
Research Chemist

Abigail P. Lindstrom
Research Chemist

Reviewed by:

John D. Fassett
Supervisory Research Chemist

Gregory C. Turk
Research Chemist

Table I. Analytical Parameters

Analyte Signal Measurement	
Element	Bi
Nominal Concentration	10 µg/g
Wavelength	223.061 nm
Integration Time	100 ms
Internal Standard Signal Measurement	
Element	Sc
Nominal Concentration	0.5 µg/g
Wavelength	361.383 nm
Integration Time	100 ms
Sampling Scheme	
nominal mg analyte	30 mg (3 g solution)
nominal mg IS	30 mg (3 g solution)
dilution	500:1 in 2 % Volume Fraction HNO3
Analysis Run Parameters	
Total Experiment Time	278 Minutes
Number of Repeats of the Analysis Run	5

Figure 18 NIST Report 839.01-00-518, Pages 5 and 6

839.01-00-518

Table II. Instrument Operating Parameters

ICP SOURCE OPERATING PARAMETERS	
Plasma Flow	15 L min^{-1}
Auxiliary Flow	0.5 L min^{-1}
Nebulizer Flow	0.8 L min^{-1}
Power	1300 W
Sample Uptake	1 mL min^{-1}
Autosampler Probe Rinse	15 sec in 2 % Volume Fraction HNO_3
SPECTROMETER OPERATING PARAMETERS	
Signal Measurement Mode	Peak Integration, Low Resolution Readout
Background Correction	Manually selected, 2-point interpolation
Measurement Time	5 s
Replicate Measurements	5

Page 5 of 6

839.01-00-518

Table III. Values and Uncertainties as Measured by ICP-OES

RECOMMENDED VALUE	
Mass Fraction (mg/g)	10.015
TYPE A UNCERTAINTIES	
Standard Error of Replication (mg/g) (DF=5)	0.00244
Relative Standard Error of Replication	0.0243%
Average Calibration Factor $(mg/g)^{-1}$	0.0100
Standard Error of Calibration Factor $(mg/g)^{-1}$ (DF=7)	0.0000017
Relative Standard Error of Calibration Factor	0.0167%
Uncertainty due to Calibration (mg/g)	0.00167
Combined Type A (mg/g)	0.00295
TYPE B UNCERTAINTIES	
Relative Uncertainty in Calibration Material Purity	0.0004%
Uncertainty due to Calibration Material Impurity (mg/g)	0.00004
COMBINED UNCERTAINTIES	
u_c (mg/g)	0.00295
DF	9
k	2.26
U (mg/g)	0.007

Page 6 of 6

with Alfred Jacob from the Laboratory Spiez. We thank them for providing the data. Thus, in this second real life example we will show a sequence of screen shots using the software *Uncertainty Manager* which ultimately produces the M-code in Example 22. This M-code is evaluated by the calculator module and the results are visualized by the graphical user interface again. Shown here is the sparsely commented M-code.

Example 22 is a standard measurement of calcium content in spring water, once more using ICP-OES. What is special in this case is that there is but one probe of calibrant solution c_{cal} to calibrate the ICP-OES apparatus. Therefore the calibration is done by diluting the calibrant solution which induces strong and non-negligible correlations among the x-values of the calibration. The complexity of data evolving from this dilution and calibration process crosses the feasibility border when employing standard Excel techniques.

Example 22: Figures 20–22 show a few screen shots of the current β-version of *Uncertainty Manager* in order to illustrate the flavor of graphically constructing M-code.

The generated M-code is shown in the following block. It can be considered as output or input depending on your point of view. It is the output of *Uncertainty Manager* comparable to an archive or protocol of the executed calculations. It can be fed to the Calculator Module basically described in this entire text in order to recompute the results.

Figure 19 shows the dialog for selecting the type of function to be fitted to the calibration points and for selecting the number of calibration points which is in this example 4 calibration points.

Figure 20 shows the dialog for describing one calibration point. In this case it is the construction of calibrant solution $c_4 = c_{cal}\frac{V_1}{V_2}\frac{V_9}{V_{10}}$. The original calibrant solution has been diluted in two steps: add volume V_1 and take out V_2. Then add volume V_9 and take out volume V_{10}. These volumes are all defined in the same dialog. In fact the dialog for describing calibrant solution resembles very much the top view (c.f. Figure 14 on page 124) which is not too surprising. After all, both are intended to construct the description of a measured entity. The definition of the x-value for the calibration (c_4) can be found in the M file on Lines 858–961. In Line 963 one can see the definition of the corresponding y-value which constitutes a reading during the calibration process. Our example shows a 4-point calibration, therefore this entire block is basically repeated 3 times. The corresponding y-values can be found on Lines 618, 733 and 848 respectively.

Figure 19 Calibration Type Selection: A pop up list in the right top corner offers the various types of functions to be fitted to k calibration points, where k is the number of calibration points to be entered in the field below the pop up list. The function graph is for visualizing the type of function.

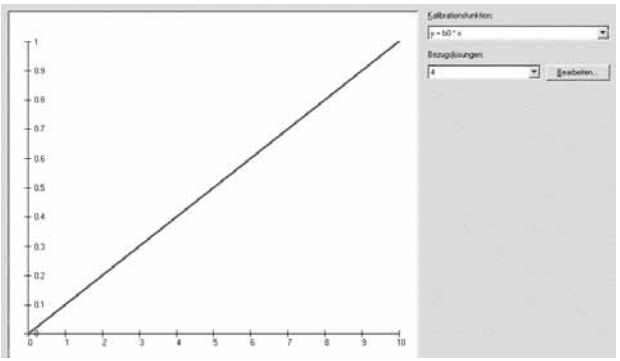
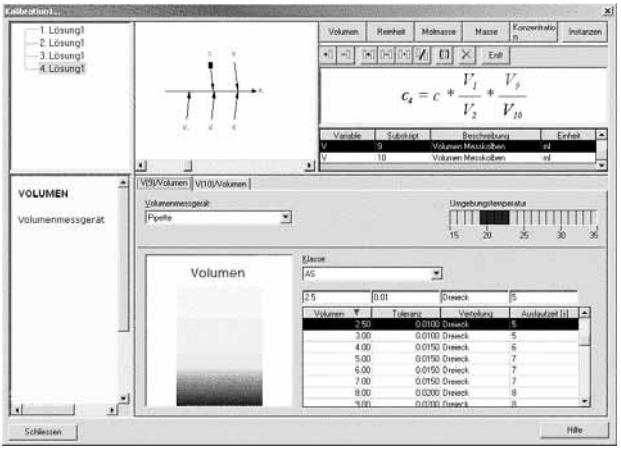


Figure 20 Dilution for One Calibration Point: The measurement, or rather the construction, of one calibrant solution is described in one dialog per calibration point, defining the model and the instruments used for the calibrant solution.

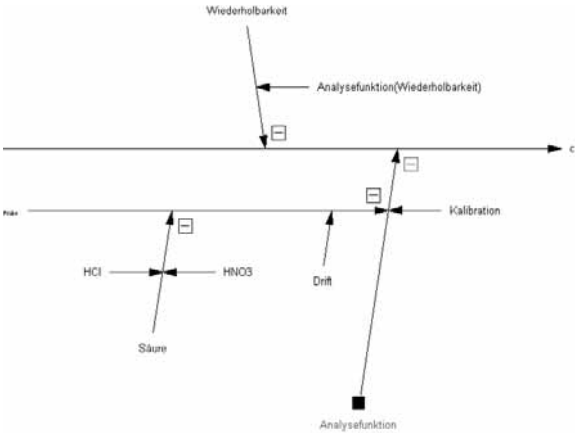


The calibration is prepared on Lines 970–973. Figure 19 is the corresponding look and feel in *Uncertainty Manager*.

The reader may find it surprising and/or interesting how complex the complete description of a relatively simple standard measurement procedure can be. In fact, the complexity of almost all standard procedures is the reason why project MUSAC was launched in the first place.

The Graphical User Interface relies on a large collection of so called "library functions" stored in a database. These contain models for each and every single measurement step. They are the condensed expert knowledge of numerous metrological experts per measurement step, so to say. Lines 9–385 are the library functions selected by the User Interface. Many of them are very simple, indeed, but they provide a transparent mechanism to enhance and elaborate the various models.

Figure 21 Resulting CED for ICP-OES. Influences on Calibration process not (yet) shown.



The last two figures (Figure 21 and Figure 22) in this example show two output results. Figure 21 shows the CED of the measurement process where the influences of the calibration process are subsumed in one arrow "Kalibration", which has 4 similar CEDs for each calibration point as shown in Figure 20.

The actual measurement using the calibrated ICP-OES apparatus is defined on Line 1040.

Example 22 Output / Input:

```
# ------------------------------------------------------------------------
# Library functions for titration
# (c) by EMPA, St. Gallen
# Version 1.0 mro
# Program - Version 1.0 mwo/18.01.2001
# Edit - mwo/03.09.2001 -> Micropipet
# ------------------------------------------------------------------------

# Library-Functions for FLASK: ---------------------------------------------
# Calibration
# Tolerance given by the manufactor of the volumetric flask
function vflask_cal_tolerance(tolerance){
        tolerance;
};

# Variability in terms of filling the volumetric flask
function vflask_cal_filling(filling){
        filling;
};

# Variablity rsp. change in the volume of the volumetric flask
# because of use ("Ageing")
function vflask_cal_aging(nom, af, time){
        # nom: nomianl value [vol_unit]
        # af: aging_factor [parts per time]
        # time: age[time]

        nom * af * time;
};

# Repeatability
function vflask_rep_rep(repeatability){
        repeatability;
};

# Temperature influence
function vflask_temp_not_corrected(nom, tau, temp){
        # nom: nominal value [vol_unit]
        # tau: expansion coefficient [parts per K]
        # temp: temperatur_variation [temperatur_unit]

        nom * tau * temp;
};
```

```
# Library-Functions for MASS: ------------------------------------------------
# Repeatability
# Repeatability of the scale
function mmass_rep_scale(repeatability){
        repeatability;
};

# Repeatability of the sample
function mmass_rep_sample(repeatability){
        repeatability;
};

# Linearity
function mmass_linearity(linearity){
        2 * linearity;
};

# Sensitivity
function mmass_sensitivity(nom, sensitivity){
        # nom: value from the weighing [mass_unit]
        # sensitivity: longtime change []

        nom * sensitivity;
};

# Drift
function mmass_drift(nom, drifttemperatur, drift){
        # nom: value from the weighing [mass_unit]
        # drifttemperatur: temperatur change causing drift [temperatur_unit]
        # drift: change due to temperature variation [parts per temperatur_unit]

        nom * drifttemperatur * drift;
};

# Library-Functions for VOLUME (TITRATION): ----------------------------------
# Calibration
# Regular testing of the delivered volume
# Limit given by the lab/norm of the volume delivery
function titration_cal_test(limit){
        limit;
};

# No regular testing of the delivered volume
# Tolerance given by the manufactor of the volume delivery
function titration_cal_tolerance(tolerance){
        tolerance;
```

```
};

# Variablity rsp. change in the volume of the volume delivery
# because of use ("Ageing")
function titration_cal_aging(nom, af, time){
        # nom: nomianl value [vol_unit]
        # af: aging_factor [parts per time]
        # time: age[time]

        nom * af * time;
};

# Repeatability of delivered volume
function titration_vol_rep(repeatability){
        repeatability;
};

# Temperature influence
function titration_vol_tempnotcorrected(nom, tau, temp){
        # nom: nominal value [vol_unit]
        # tau: expansion coefficient [parts per K]
        # temp: temperatur_variation [temperatur_unit]

        nom * tau * temp;
};

# Endpoint determination
# Repeatability of the end-point determination
function titration_end_rep(repeatability){
        repeatability;
};

# Bias range of the end-point determination
# Inflence of the endpoint detection
function titration_end_detection(c_det, u_det){
        # c: sensitivity coefficient;
        # u: uncertainty (variability of the given influence)

        c_det * u_det;
};

# Influence of the volume used for titration
function titration_end_samplevol(c_det, u_det){
        # c: sensitivity coefficient;
        # u: uncertainty (variability of the given influence)

        c_det * u_det;
```

```
};

# Influence of the stirrer used for titration
function titration_end_stirrer(c_sti, u_sti){
        # c: sensitivity coefficient;
        # u: uncertainty (variability of the given influence)

        c_sti * u_sti;
};

# Influence of the arrangement used during the titration
function titration_end_arrangement(c_arr, u_arr){
        # c: sensitivity coefficient;
        # u: uncertainty (variability of the given influence)

        c_arr * u_arr;
};

# Influence of a parameter set during the titration
function titration_end_parameter_1(c_pa1, u_pa1){
        # c: sensitivity coefficient;
        # u: uncertainty (variability of the given influence)

        c_pa1 * u_pa1;
};

# Library-Functions for PIPETTE: ----------------------------------------------
# Calibration
# Tolerance given by the manufactor of the volumetric pipet
function vvolpipet_cal_tolerance(tolerance){
        tolerance;
};

# Variability in terms of discharging the volumetric pipet
function vvolpipet_cal_discharge(discharge){
        discharge;
};

# Variablity rsp. change in the volume of the volumetric pipette
# because of use ("Ageing")
function vvolpipet_cal_aging(nom, af, time){
        # nom: nomianl value [vol_unit]
        # af: aging_factor [parts per time]
        # time: age[time]

        nom * af * time;
```

```
};

# Repeatability
function vvolpipet_rep_rep(repeatability){
        repeatability;
};

# Temperature influence
function vvolpipet_temp_not_corrected(nom, tau, temp){
        # nom: nominal value [vol_unit]
        # tau: expansion coefficient [parts per K]
        # temp: temperatur_variation [temperatur_unit]

        nom * tau * temp;
};

# Volumetric Pipet (Micropipette)
# Version 1.0 Rm/280 04.09.2001
#

# Library-Functions: ------------------------------------------------------------
#
# Calibration -------------------------------------------------------------------
#
# Tolerance given by the manufactor of the volumetric micropipet

function vmicropipet_cal_tolerance(tol){
        tol;
};

# Variability in terms of discharging the volumetric micropipet

function vmicropipet_cal_discharge(dis){
        dis;
};

# Variablity rsp. change in the volume of the volumetric micropipet
# because of use ("Ageing")

function vmicropipet_cal_aging(nom, af, time){
        # nom: nomianl value [vol_unit]
        # af: aging_factor [parts per time]
        # time: age[time]

        nom * af * time;
};
```

```
# Variability in terms of altitude above sea level

function vmicropipet_cal_location(loc){
        loc;
};

# Variability in terms of density of the solvent (organic)

function vmicropipet_cal_density(den){
        den;
};

# Repeatability ------------------------------------------------------------

function vmicropipet_rep_rep(repeatability){
        repeatability;
};

# Temperature influence ----------------------------------------------------

function vmicropipet_temp_not_corrected(nom, tau, temp){
        # nom: nominal value [vol_unit]
        # tau: expansion coefficient [parts per K]
        # temp: temperatur_variation [temperatur_unit]
        nom * tau * temp;
};

# HPLC
# 1-Punkt Kalibration
# Version 1.0 Rm/280 17.7.2001
# Version 1.5 Rm/280 4.10.2001
#

# Library-Functions: -------------------------------------------------------
#
# Injection ----------------------------------------------------------------

function hplc_injection(in){
        # in: injection []

        in;
};

#-------------------------------------------------------------------------

# Flow ---------------------------------------------------------------------

function hplc_flow(sf, uf){
```

```
        # sf: sensitifity coefficient for the flow [min/ml]
        # uf: long-term variation of the flow [ml/min]

        sf * uf;
};

#-------------------------------------------------------------------------

# Eluent ---------------------------------------------------------------

function hplc_eluent(se, ue){
        # se: sensitivity coefficient for the eluent composition
        # ue: long-term variation of the eluent composition

        se * ue;
};

#-------------------------------------------------------------------------

# Column ---------------------------------------------------------------

function hplc_column(sc, uc){
        # sc: sensitivity coefficient for the column temperatur [1/K]
        # uc: long-term variation of the column temperature [K]

        sc * uc;
};

#-------------------------------------------------------------------------

# Detection ------------------------------------------------------------

function hplc_detection(ss, ul){
        # ss: sensitivity of the spectrum [1/nm]
        # ul: uncertainty of the wavelength [nm]

        ss * ul;
};

#-------------------------------------------------------------------------

# Limit ----------------------------------------------------------------

function hplc_limit(li){
        # li: limit

        li;
};

#-------------------------------------------------------------------------
```

```
# Peak ------------------------------------------------------------------

function hplc_peak(pe){
        #pe: uncertainty in peak area

        pe;
};

# ICP-OES
# 1-Punkt Kalibration
# Version 1.0 Rm/280 25.6.2001
# Version 1.5 Rm/280 7.10.2001
#

# Library-Functions: ----------------------------------------------------
#
# Limit -----------------------------------------------------------------

function icpoes_limit(li){
        # li: limit

        li;
};
#-----------------------------------------------------------------------

# Drift -----------------------------------------------------------------

function icpoes_drift(dr, ut){
        # dr: drift
        # ut: time to drift

        dr * ut;
};
#-----------------------------------------------------------------------

# Acid influence--------------------------------------------------------

function icpoes_acid(sa, ua){
        # sa: sensitivity coefficient for an acid
        # ua: difference between the acid concentration in
        #     the sample and the reference

        sa * ua;
};
#-----------------------------------------------------------------------

# Salt influence--------------------------------------------------------

function icpoes_salt(ss, us){
```

```
        # ss: sensitivity coefficient for a salt
        # us: difference between the salt concentration
        #     in the sample and the reference

        ss * us;
};
#-------------------------------------------------------------------------

# -------------------------------------------------------------------------
# Standard solution (Konzentration des Titiermittels)
# Version 1.0 Rm/246 26.11.2000
# Program - Version 1.0 mwo, 18.01.2001
# -------------------------------------------------------------------------

# Assignments: ------------------------------------------------------------
var_sol1_c__2 = "Sub_ssol_sol1_c__2"(conc = <10 :r 0.05>[g/l]);

# Calculation of the parameter: -------------------------------------------
sol1_c__2 = get(conc, var_sol1_c__2);

# -------------------------------------------------------------------------
# Volumetric Pipet (Vollpipette)
# Version 1.0 Rm/246 26.11.2000
# Program - Version 1.0 mwo, 21.01.2002
# Program - Version 1.1 mwo, 04.10.2002
# -------------------------------------------------------------------------

# Assignments: ------------------------------------------------------------
sub_sol1_v_1_8 = "Substance_sol1_v_1_8"();

# All the assignment for the instrument measuring volume sol1_v_1_8
varsol1_v_1_8 = "Avolpipet_sol1_v_1_8"(
                tolerance = <0 :t 0.007[ml]> [ml],
                discharge = <0 :r 0.0005[ml]> [ml],
                age_factor = <0.0001 :r 20%> [1/y], age = <0 :r 5[y]> [y],
                rep = <0 :g 0.002> [ml],
                tau = <0.00021 :r 20%> [1/K], temp = <0 :r 2[K]> [K]

##              tolerance = <0 :r 0.03>[ml],
##              discharge = <0 :r 0.05%>[ml],
##              age_factor = <6e-4 :r 3e-4>[1/y], age = <1 :r 5>[y],
##              rep = <0 :g 0.03>[ml],
##              tau = <2.1e-4 :r 20%>[1/K], temp = <1:r 3> [K]
);

# Calculation of the parameter: -------------------------------------------
```

```
# Combine the different influence quanities of the volume sol1_v_1_8
tool_sol1_v_1_8 = "Tvolpipet_sol1_v_1_8"(
                calibration_sol1_v_1_8 = function(){
                        vvolpipet_cal_tolerance(get(tolerance, varsol1_v_1_8)) +
                        vvolpipet_cal_discharge(get(discharge, varsol1_v_1_8)) +
                        vvolpipet_cal_aging(value, get(age_factor, varsol1_v_1_8),
                                        get(age, varsol1_v_1_8));
                },

                rep_sol1_v_1_8 = function(){
                        vvolpipet_rep_rep(get(rep, varsol1_v_1_8));
                },

                temp_sol1_v_1_8 = function(){
                        vvolpipet_temp_not_corrected(value, get(tau, varsol1_v_1_8),
                                                get(temp, varsol1_v_1_8));
                },

                function do(){
                        value +
                        calibration_sol1_v_1_8() +
                        rep_sol1_v_1_8() +
                        temp_sol1_v_1_8();
                }
);

# Final definition of variable ------------------------------------------------
sol1_v_1_8 = 1[ml] sub_sol1_v_1_8 tool_sol1_v_1_8;

# ----------------------------------------------------------------------------
# Volumetric Flask (Messkolben)
# Version 1.2 Rm/246 23.11.2000 unique function names
# Program - Version 1.0 Mwo 20.01.2001
# Program - Version 1.1 Mwo 04.10.2001

# Assignments: ----------------------------------------------------------------
sub_sol1_v_2_12 = "Substance_sol1_v_2_12"();

# All the assignment for the instrument measuring volume sol1_v_2_12
varsol1_v_2_12 = "Aflask_sol1_v_2_12"(
                tolerance = <0 :t 0.1[ml]> [ml],
                filling = <0 :r 0.004[ml]> [ml],
                age_factor = <0.0001 :r 20%> [1/y], age = <0 :r 5[y]> [y],
                rep = <0 :g 0.006> [ml],
                tau = <0.00021 :r 20%> [1/K], temp = <0 :r 1.5[K]> [K]
);
```

```
# Calculation of the parameter: ------------------------------------------------
# Combine the different influence quanities of the volume sol1_v_2_12
tool_sol1_v_2_12 = "Tflask_sol1_v_2_12"(
        calibration_sol1_v_2_12 = function(){
                vflask_cal_tolerance(get(tolerance, varsol1_v_2_12)) +
                vflask_cal_filling(get(filling, varsol1_v_2_12)) +
                vflask_cal_aging(value, get(age_factor, varsol1_v_2_12),
                                 get(age, varsol1_v_2_12));
                },

                rep_sol1_v_2_12 = function(){
                        vflask_rep_rep(get(rep, varsol1_v_2_12));
                },


                temp_sol1_v_2_12 = function(){
                        vflask_temp_not_corrected(value, get(tau, varsol1_v_2_12),
                                                  get(temp, varsol1_v_2_12));
                },

                function do(){
                        value +
                        calibration_sol1_v_2_12() +
                                rep_sol1_v_2_12() +
                        temp_sol1_v_2_12();
                }
);

# Final definition of variable -------------------------------------------------
sol1_v_2_12 = 100[ml] sub_sol1_v_2_12 tool_sol1_v_2_12;


# ------------------------------------------------------------------------------
# Volumetric Pipet (Vollpipette)
# Version 1.0 Rm/246 26.11.2000
# Program - Version 1.0 mwo, 21.01.2002
# Program - Version 1.1 mwo, 04.10.2002
# ------------------------------------------------------------------------------

# Assignments: -----------------------------------------------------------------
sub_sol1_v_3_16 = "Substance_sol1_v_3_16"();

# All the assignment for the instrument measuring volume sol1_v_3_16
varsol1_v_3_16 = "Avolpipet_sol1_v_3_16"(
                tolerance = <0 :t 0.03[ml]> [ml],
```

```
                discharge = <0 :r 0.01[ml]> [ml],
                age_factor = <0.0001 :r 20%> [1/y], age = <0 :r 5[y]> [y],
                rep = <0 :g 0.008> [ml],
                tau = <0.00021 :r 20%> [1/K], temp = <0 :r 2[K]> [K]

##              tolerance = <0 :r 0.03>[ml],
##              discharge = <0 :r 0.05%>[ml],
##              age_factor = <6e-4 :r 3e-4>[1/y], age = <1 :r 5>[y],
##              rep = <0 :g 0.03>[ml],
##              tau = <2.1e-4 :r 20%>[1/K], temp = <1:r 3> [K]
);

# Calculation of the parameter: ------------------------------------------------
# Combine the different influence quanities of the volume sol1_v_3_16
tool_sol1_v_3_16 = "Tvolpipet_sol1_v_3_16"(
                calibration_sol1_v_3_16 = function(){
                        vvolpipet_cal_tolerance(get(tolerance, varsol1_v_3_16)) +
                        vvolpipet_cal_discharge(get(discharge, varsol1_v_3_16)) +
                        vvolpipet_cal_aging(value, get(age_factor, varsol1_v_3_16),
                                        get(age, varsol1_v_3_16));
                },

                rep_sol1_v_3_16 = function(){
                        vvolpipet_rep_rep(get(rep, varsol1_v_3_16));
                },

                temp_sol1_v_3_16 = function(){
                        vvolpipet_temp_not_corrected(value, get(tau, varsol1_v_3_16),
                                                get(temp, varsol1_v_3_16));
                },

                function do(){
                        value +
                        calibration_sol1_v_3_16() +
                        rep_sol1_v_3_16() +
                        temp_sol1_v_3_16();
                }
);

# Final definition of variable -------------------------------------------------
sol1_v_3_16 = 20[ml] sub_sol1_v_3_16 tool_sol1_v_3_16;

# ------------------------------------------------------------------------------
# Volumetric Flask (Messkolben)
# Version 1.2 Rm/246 23.11.2000 unique function names
# Program - Version 1.0 Mwo 20.01.2001
```

```
# Program - Version 1.1 Mwo 04.10.2001

# Assignments: ------------------------------------------------------------
sub_sol1_v_4_20 = "Substance_sol1_v_4_20"();

# All the assignment for the instrument measuring volume sol1_v_4_20
varsol1_v_4_20 = "Aflask_sol1_v_4_20"(
                tolerance = <0 :t 0.1[ml]> [ml],
                filling = <0 :r 0.004[ml]> [ml],
                age_factor = <0.0001 :r 20%> [1/y], age = <0 :r 5[y]> [y],
                rep = <0 :g 0.006> [ml],
                tau = <0.00021 :r 20%> [1/K], temp = <0 :r 1.5[K]> [K]
);

# Calculation of the parameter: -------------------------------------------
# Combine the different influence quanities of the volume sol1_v_4_20
tool_sol1_v_4_20 = "Tflask_sol1_v_4_20"(
        calibration_sol1_v_4_20 = function(){
                        vflask_cal_tolerance(get(tolerance, varsol1_v_4_20)) +
                        vflask_cal_filling(get(filling, varsol1_v_4_20)) +
                        vflask_cal_aging(value, get(age_factor, varsol1_v_4_20),
                                        get(age, varsol1_v_4_20));
                },

                rep_sol1_v_4_20 = function(){
                        vflask_rep_rep(get(rep, varsol1_v_4_20));
                },


                temp_sol1_v_4_20 = function(){
                        vflask_temp_not_corrected(value, get(tau, varsol1_v_4_20),
                        get(temp, varsol1_v_4_20));
                },

                function do(){
                        value +
                        calibration_sol1_v_4_20() +
                                rep_sol1_v_4_20() +
                        temp_sol1_v_4_20();
                }
);

# Final definition of variable --------------------------------------------
sol1_v_4_20 = 100[ml] sub_sol1_v_4_20 tool_sol1_v_4_20;


# ------------------------------------------------------------------------------
```

```
# Definition of formula for calibration
# Program - Version mwo 19.10.2001
# ------------------------------------------------------------------------

# Definition of formula ---------------------------------------------------
sol1_c_1_3 = sol1_c__2 * (sol1_v_1_8) / (sol1_v_2_12)
                      * (sol1_v_3_16) / (sol1_v_4_20);
# Definition of y-value ---------------------------------------------------
ysol1_c_1_3 = <248235.22 : 3267.67>;

# ------------------------------------------------------------------------
# Volumetric Pipet (Vollpipette)
# Version 1.0 Rm/246 26.11.2000
# Program - Version 1.0 mwo, 21.01.2002
# Program - Version 1.1 mwo, 04.10.2002
# ------------------------------------------------------------------------

# Assignments: ------------------------------------------------------------
sub_sol2_v_5_16 = "Substance_sol2_v_5_16"();

# All the assignment for the instrument measuring volume sol2_v_5_16
varsol2_v_5_16 = "Avolpipet_sol2_v_5_16"(
                tolerance = <0 :t 0.02[ml]> [ml],
                discharge = <0 :r 0.005[ml]> [ml],
                age_factor = <0.0001 :r 20%> [1/y], age = <0 :r 5[y]> [y],
                rep = <0 :g 0.005> [ml],
                tau = <0.00021 :r 20%> [1/K], temp = <0 :r 2[K]> [K]

##              tolerance = <0 :r 0.03>[ml],
##              discharge = <0 :r 0.05%>[ml],
##              age_factor = <6e-4 :r 3e-4>[1/y], age = <1 :r 5>[y],
##              rep = <0 :g 0.03>[ml],
##              tau = <2.1e-4 :r 20%>[1/K], temp = <1:r 3> [K]
);

# Calculation of the parameter: -------------------------------------------
# Combine the different influence quanities of the volume sol2_v_5_16
tool_sol2_v_5_16 = "Tvolpipet_sol2_v_5_16"(
                calibration_sol2_v_5_16 = function(){
                        vvolpipet_cal_tolerance(get(tolerance, varsol2_v_5_16)) +
                        vvolpipet_cal_discharge(get(discharge, varsol2_v_5_16)) +
                        vvolpipet_cal_aging(value, get(age_factor, varsol2_v_5_16),
                                        get(age, varsol2_v_5_16));
                },

                rep_sol2_v_5_16 = function(){
```

```
                        vvolpipet_rep_rep(get(rep, varsol2_v_5_16));
                },

                temp_sol2_v_5_16 = function(){
                        vvolpipet_temp_not_corrected(value, get(tau, varsol2_v_5_16),
                                                get(temp, varsol2_v_5_16));
                },

                function do(){
                        value +
                        calibration_sol2_v_5_16() +
                                rep_sol2_v_5_16() +
                        temp_sol2_v_5_16();
                }
);

# Final definition of variable ------------------------------------------------
sol2_v_5_16 = 10[ml] sub_sol2_v_5_16 tool_sol2_v_5_16;

# ------------------------------------------------------------------------------
# Volumetric Flask (Messkolben)
# Version 1.2 Rm/246 23.11.2000 unique function names
# Program - Version 1.0 Mwo 20.01.2001
# Program - Version 1.1 Mwo 04.10.2001

# Assignments: ----------------------------------------------------------------
sub_sol2_v_6_20 = "Substance_sol2_v_6_20"();

# All the assignment for the instrument measuring volume sol2_v_6_20
varsol2_v_6_20 = "Aflask_sol2_v_6_20"(
                tolerance = <0 :t 0.1[ml]> [ml],
                filling = <0 :r 0.004[ml]> [ml],
                age_factor = <0.0001 :r 20%> [1/y], age = <0 :r 5[y]> [y],
                rep = <0 :g 0.006> [ml],
                tau = <0.00021 :r 20%> [1/K], temp = <0 :r 1.5[K]> [K]
);

# Calculation of the parameter: -----------------------------------------------
# Combine the different influence quanities of the volume sol2_v_6_20
tool_sol2_v_6_20 = "Tflask_sol2_v_6_20"(
        calibration_sol2_v_6_20 = function(){
                vflask_cal_tolerance(get(tolerance, varsol2_v_6_20)) +
                vflask_cal_filling(get(filling, varsol2_v_6_20)) +
                vflask_cal_aging(value, get(age_factor, varsol2_v_6_20),
                                get(age, varsol2_v_6_20));
                },
```

```
                rep_sol2_v_6_20 = function(){
                        vflask_rep_rep(get(rep, varsol2_v_6_20));
                },

                temp_sol2_v_6_20 = function(){
                        vflask_temp_not_corrected(value, get(tau, varsol2_v_6_20),
                                                  get(temp, varsol2_v_6_20));
                },

                function do(){
                        value +
                        calibration_sol2_v_6_20() +
                                rep_sol2_v_6_20() +
                        temp_sol2_v_6_20();
                }
);

# Final definition of variable ------------------------------------------------
sol2_v_6_20 = 100[ml] sub_sol2_v_6_20 tool_sol2_v_6_20;


# ----------------------------------------------------------------------------
# Definition of formula for calibration
# Program - Version mwo 19.10.2001
# ----------------------------------------------------------------------------

# Definition of formula -------------------------------------------------------
sol2_c_2_3 = sol1_c__2 * (sol1_v_1_8) / (sol1_v_2_12)
                       * (sol2_v_5_16) / (sol2_v_6_20);
# Definition of y-value -------------------------------------------------------
ysol2_c_2_3 = <124861.33 : 1197.88>;

# ----------------------------------------------------------------------------
# Volumetric Pipet (Vollpipette)
# Version 1.0 Rm/246 26.11.2000
# Program - Version 1.0 mwo, 21.01.2002
# Program - Version 1.1 mwo, 04.10.2002
# ----------------------------------------------------------------------------

# Assignments: ----------------------------------------------------------------
sub_sol3_v_7_16 = "Substance_sol3_v_7_16"();

# All the assignment for the instrument measuring volume sol3_v_7_16
varsol3_v_7_16 = "Avolpipet_sol3_v_7_16"(
                tolerance = <0 :t 0.015[ml]> [ml],
                discharge = <0 :r 0.0025[ml]> [ml],
```

```
                age_factor = <0.0001 :r 20%> [1/y], age = <0 :r 5[y]> [y],
                rep = <0 :g 0.004> [ml],
                tau = <0.00021 :r 20%> [1/K], temp = <0 :r 2[K]> [K]

##              tolerance = <0 :r 0.03>[ml],
##              discharge = <0 :r 0.05%>[ml],
##              age_factor = <6e-4 :r 3e-4>[1/y], age = <1 :r 5>[y],
##              rep = <0 :g 0.03>[ml],
##              tau = <2.1e-4 :r 20%>[1/K], temp = <1:r 3> [K]
);

# Calculation of the parameter: ---------------------------------------------
# Combine the different influence quanities of the volume sol3_v_7_16
tool_sol3_v_7_16 = "Tvolpipet_sol3_v_7_16"(
                calibration_sol3_v_7_16 = function(){
                        vvolpipet_cal_tolerance(get(tolerance, varsol3_v_7_16)) +
                        vvolpipet_cal_discharge(get(discharge, varsol3_v_7_16)) +
                        vvolpipet_cal_aging(value, get(age_factor, varsol3_v_7_16),
                                        get(age, varsol3_v_7_16));
                },

                rep_sol3_v_7_16 = function(){
                        vvolpipet_rep_rep(get(rep, varsol3_v_7_16));
                },

                temp_sol3_v_7_16 = function(){
                        vvolpipet_temp_not_corrected(value, get(tau, varsol3_v_7_16),
                                                get(temp, varsol3_v_7_16));
                },

                function do(){
                        value +
                        calibration_sol3_v_7_16() +
                        rep_sol3_v_7_16() +
                        temp_sol3_v_7_16();
                }
);

# Final definition of variable ----------------------------------------------
sol3_v_7_16 = 5[ml] sub_sol3_v_7_16 tool_sol3_v_7_16;

# ---------------------------------------------------------------------------
# Volumetric Flask (Messkolben)
# Version 1.2 Rm/246 23.11.2000 unique function names
# Program - Version 1.0 Mwo 20.01.2001
# Program - Version 1.1 Mwo 04.10.2001
```

```
# Assignments: ----------------------------------------------------------------
sub_sol3_v_8_20 = "Substance_sol3_v_8_20"();

# All the assignment for the instrument measuring volume sol3_v_8_20
varsol3_v_8_20 = "Aflask_sol3_v_8_20"(
                tolerance = <0 :t 0.1[ml]> [ml],
                filling = <0 :r 0.004[ml]> [ml],
                age_factor = <0.0001 :r 20%> [1/y], age = <0 :r 5[y]> [y],
                rep = <0 :g 0.006> [ml],
                tau = <0.00021 :r 20%> [1/K], temp = <0 :r 1.5[K]> [K]
);

# Calculation of the parameter: -----------------------------------------------
# Combine the different influence quanities of the volume sol3_v_8_20
tool_sol3_v_8_20 = "Tflask_sol3_v_8_20"(
        calibration_sol3_v_8_20 = function(){
                vflask_cal_tolerance(get(tolerance, varsol3_v_8_20)) +
                vflask_cal_filling(get(filling, varsol3_v_8_20)) +
                vflask_cal_aging(value, get(age_factor, varsol3_v_8_20),
                                get(age, varsol3_v_8_20));
                },

                rep_sol3_v_8_20 = function(){
                        vflask_rep_rep(get(rep, varsol3_v_8_20));
                },

                temp_sol3_v_8_20 = function(){
                        vflask_temp_not_corrected(value, get(tau, varsol3_v_8_20),
                                                get(temp, varsol3_v_8_20));
                },

                function do(){
                        value +
                        calibration_sol3_v_8_20() +
                                rep_sol3_v_8_20() +
                        temp_sol3_v_8_20();
                }
);

# Final definition of variable ------------------------------------------------
sol3_v_8_20 = 100[ml] sub_sol3_v_8_20 tool_sol3_v_8_20;


# ------------------------------------------------------------------------------
# Definition of formula for calibration
# Program - Version mwo 19.10.2001
```

```
# ------------------------------------------------------------------------

# Definition of formula ---------------------------------------------------
sol3_c_3_3 = sol1_c__2 * (sol1_v_1_8) / (sol1_v_2_12)
                       * (sol3_v_7_16) / (sol3_v_8_20);
# Definition of y-value ---------------------------------------------------
ysol3_c_3_3 = <62442.44 : 622.22>;

# ------------------------------------------------------------------------
# Volumetric Pipet (Vollpipette)
# Version 1.0 Rm/246 26.11.2000
# Program - Version 1.0 mwo, 21.01.2002
# Program - Version 1.1 mwo, 04.10.2002
# ------------------------------------------------------------------------

# Assignments: ------------------------------------------------------------
sub_sol4_v_9_16 = "Substance_sol4_v_9_16"();

# All the assignment for the instrument measuring volume sol4_v_9_16
varsol4_v_9_16 = "Avolpipet_sol4_v_9_16"(
                tolerance = <0 :t 0.01[ml]> [ml],
                discharge = <0 :r 0.00125[ml]> [ml],
                age_factor = <0.0001 :r 20%> [1/y], age = <0 :r 5[y]> [y],
                rep = <0 :g 0.003> [ml],
                tau = <0.00021 :r 20%> [1/K], temp = <0 :r 2[K]> [K]

##              tolerance = <0 :r 0.03>[ml],
##              discharge = <0 :r 0.05%>[ml],
##              age_factor = <6e-4 :r 3e-4>[1/y], age = <1 :r 5>[y],
##              rep = <0 :g 0.03>[ml],
##              tau = <2.1e-4 :r 20%>[1/K], temp = <1:r 3> [K]
);

# Calculation of the parameter: -------------------------------------------
# Combine the different influence quanities of the volume sol4_v_9_16
tool_sol4_v_9_16 = "Tvolpipet_sol4_v_9_16"(
                calibration_sol4_v_9_16 = function(){
                        vvolpipet_cal_tolerance(get(tolerance, varsol4_v_9_16)) +
                        vvolpipet_cal_discharge(get(discharge, varsol4_v_9_16)) +
                        vvolpipet_cal_aging(value, get(age_factor, varsol4_v_9_16),
                                        get(age, varsol4_v_9_16));
                },

                rep_sol4_v_9_16 = function(){
                        vvolpipet_rep_rep(get(rep, varsol4_v_9_16));
                },
```

```
                temp_sol4_v_9_16 = function(){
890                     vvolpipet_temp_not_corrected(value, get(tau, varsol4_v_9_16),
                                                get(temp, varsol4_v_9_16));
                },

                function do(){
895                     value +
                        calibration_sol4_v_9_16() +
                        rep_sol4_v_9_16() +
                        temp_sol4_v_9_16();
                }
900 );

    # Final definition of variable -------------------------------------------------
    sol4_v_9_16 = 2.5[ml] sub_sol4_v_9_16 tool_sol4_v_9_16;

905 # ------------------------------------------------------------------------------
    # Volumetric Flask (Messkolben)
    # Version 1.2 Rm/246 23.11.2000 unique function names
    # Program - Version 1.0 Mwo 20.01.2001
    # Program - Version 1.1 Mwo 04.10.2001
910
    # Assignments: -----------------------------------------------------------------
    sub_sol4_v_10_20 = "Substance_sol4_v_10_20"();

    # All the assignment for the instrument measuring volume sol4_v_10_20
915 varsol4_v_10_20 = "Aflask_sol4_v_10_20"(
                tolerance = <0 :t 0.1[ml]> [ml],
                filling = <0 :r 0.004[ml]> [ml],
                age_factor = <0.0001 :r 20%> [1/y], age = <0 :r 5[y]> [y],
                rep = <0 :g 0.006> [ml],
920             tau = <0.00021 :r 20%> [1/K], temp = <0 :r 1.5[K]> [K]
    );

    # Calculation of the parameter: ------------------------------------------------
    # Combine the different influence quanities of the volume sol4_v_10_20
925 tool_sol4_v_10_20 = "Tflask_sol4_v_10_20"(
        calibration_sol4_v_10_20 = function(){
                vflask_cal_tolerance(get(tolerance, varsol4_v_10_20)) +
                vflask_cal_filling(get(filling, varsol4_v_10_20)) +
                vflask_cal_aging(value, get(age_factor, varsol4_v_10_20),
930                             get(age, varsol4_v_10_20));
                },

                rep_sol4_v_10_20 = function(){
                     vflask_rep_rep(get(rep, varsol4_v_10_20));
```

```
                },

                temp_sol4_v_10_20 = function(){
                     vflask_temp_not_corrected(value, get(tau, varsol4_v_10_20),
                                               get(temp, varsol4_v_10_20));
                },

                function do(){
                        value +
                        calibration_sol4_v_10_20() +
                                rep_sol4_v_10_20() +
                        temp_sol4_v_10_20();
                }
);

# Final definition of variable ------------------------------------------------
sol4_v_10_20 = 100[ml] sub_sol4_v_10_20 tool_sol4_v_10_20;


# ------------------------------------------------------------------------------
# Definition of formula for calibration
# Program - Version mwo 19.10.2001
# ------------------------------------------------------------------------------

# Definition of formula -------------------------------------------------------
sol4_c_4_3 = sol1_c__2 * (sol1_v_1_8) / (sol1_v_2_12)
                      * (sol4_v_9_16) / (sol4_v_10_20);
# Definition of y-value -------------------------------------------------------
ysol4_c_4_3 = <29488.22 : 347.56>;

# ------------------------------------------------------------------------------
# Definition of evaluation for calibration
# Program - Version mwo 19.10.2001
# ------------------------------------------------------------------------------

xval = (sol1_c_1_3,sol2_c_2_3,sol3_c_3_3,sol4_c_4_3);
yval = (ysol1_c_1_3,ysol2_c_2_3,ysol3_c_3_3,ysol4_c_4_3);
cal = fit(xval, yval, type=1);
myfun = get(inv, cal);


# ICP-OES
# 1-Punkt Kalibration und n-Punkt Kalibration
# Version 1.0 Rm/280 25.6.2001
# Version 1.5 Rm/280 7.10.2001
#

# Assignments: ----------------------------------------------------------------
```

```
#
# Substance assignment needed, but not used

sub_analysefunktion__5 = "Substance_analysefunktion__5"();

# All the assignment for the instrument measuring volume analysefunktion__5

varanalysefunktion__5 = "Ainstrument_analysefunktion__5"(
                limit = <0 :r 0.02>, sens_acid_1 = <0.01 :r 0.002%>,
                u_acid_1 = <0 :r 0.5>, sens_acid_2 = <0.01 :r 0.002%>,
                u_acid_2 = <0 :r 0.5>
#               limit = <0 :r 0.05>,
#               drift = <0.01 :r 20%>[1/h], u_time = <0 :r 5>[h],
#               sens_acid_1 = <1 :r 20%>, u_acid_1 = <0 :r 0.1>,
#               sens_salt_1 = <1 :r 20%>[l/mg], u_salt_1 = <0 :r 0.1>[mg/l],
);

#
#-------------------------------------------------------------------------

# Calculation of the parameter: ---------------------------------------------
#
# Combine the different influence quanities of the relation ip/iref

tool_analysefunktion__5 = "Tinstrument_analysefunktion__5"(
                limit_analysefunktion__5 = function(){
                        icpoes_limit(get(limit, varanalysefunktion__5));
                },

#               drift_analysefunktion__5 = function(){
#                       icpoes_drift(get(drift, varanalysefunktion__5),
#                               get(u_time,  varanalysefunktion__5));
#               },

                        acid_1_analysefunktion__5 = function(){
                        icpoes_acid(get(sens_acid_1,varanalysefunktion__5),
                                get(u_acid_1, varanalysefunktion__5));
                },

                acid_2_analysefunktion__5 = function(){
                        icpoes_acid(get(sens_acid_2,varanalysefunktion__5),
                        get(u_acid_2, varanalysefunktion__5));
                },

                function do(){
                        cx = myfun(<value : 700>);
                        cx * (1 +
```

```
                        limit_analysefunktion__5() +
#                       drift_analysefunktion__5()
                        acid_1_analysefunktion__5() +
                acid_2_analysefunktion__5()
                        );
                }
);

# Final definition of a ICP-OES-device

analysefunktion__5 = 65555.55 sub_analysefunktion__5 tool_analysefunktion__5;


# Repeatability (Wiederholbarkeit einer Messung)
# Version 1.0 Rm/246 5.12.2000
# Program - Version 1.0 mwo, 18.01.2001
# ------------------------------------------------------------------------

# Assignments: ----------------------------------------------------------
var_wiederholbarkeit__0 = "Repeatability"(repeatability = <1 :g 0.01>);

# Calculation of the parameter: -----------------------------------------
wiederholbarkeit__0 = get(repeatability, var_wiederholbarkeit__0);

# ------------------------------------------------------------------------
# Definition of formula
# Program - Version mwo 21.01.2001
# ------------------------------------------------------------------------

# Definition of formula -------------------------------------------------
c__3 = (  analysefunktion__5 ) * wiederholbarkeit__0;

# Evaluation of the uncertainty -----------------------------------------
eval(c__3);
ureal(c__3);
calcIso(c__3);
mc(c__3, size=100000);

# END -------------------------------- END ------------------------------ END
```

As stated above the M-Code shown here is both input *and* output. In terms of output it can be considered to be a "calculation protocol" showing the complete data used to calculate the results. The software *Uncertainty Manager* uses this M-code to (re-)produce numerous results, one of which is shown in Figure 22 representing the *uncertainty budget*. Each bar basically shows the uncertainty that it (i.e. its sub-influences) contributes to the total uncertainty of the measurement result. If this M-code is given to a stand alone MUSAC-Calculator,

Figure 22 Visualization of Uncertainty Budgets in an early prototypic stage: The length of each bar shows the amount of uncertainty reduction on the result if that particular influence's uncertainty were set to zero. The numbers in the bars state the remaining uncertainty of the result.

Lines 1063–1066 compute the results as shown in the following block. We refrain from showing the evaluation of `eval(c__3)` because this is a fairly long and unreadable chunk of M-Code rewriting the entire measurement method's definition. But the result of the `ureal`-evaluation, the `iso`-evaluation and the `mc`-simulation are shown.

Example 22 Output:

```
...
<5.32072 : 0.123572>[mg/l];
<5.32072 : 0.100633>[mg/l];
"MonteCarloSimulation"(result=<5.30539 : 0.10819>[mg/l], size=100000);
```

The three results lie close together. Not surprising the `iso`-evaluation and the deprecated `ureal`-evaluation agree on the mean of the measurement result, but they disagree on the uncertainty, because the latter ignores all kinds of correlations that arose during the measurement procedure especially those stemming from the dilution process. The Monte Carlo Simulation (`mc`-evaluation) yields a slightly different expected value for the measurement result, while the measurement uncertainty is pretty much in the same order as the uncertainty by the `iso`-evaluation. We interpret this situation as follows: since both methods `iso` and `mc` agree on the measurement uncertainty it can be assumed to be quite insensitive in the operating range of this measurement application. The difference

in the expected value of the measurement result can be an indication that either the correlations induced by the dilution process are not properly captured by the first-order approximation of the `iso`-evaluation, or that the Monte Carlo Simulation of the calibration process using ordinary least squares (c.f. Section 5.3.3 on page 120) uncovers a bias in this particular set of data.