



HAL
open science

Modelling and traceability for computationally-intensive precision engineering and metrology

J.M. Linares, G. Goch, A. Forbes, J.M. Sprauel, Clément Audibert, F. Haertig, W. Gao

► To cite this version:

J.M. Linares, G. Goch, A. Forbes, J.M. Sprauel, Clément Audibert, et al.. Modelling and traceability for computationally-intensive precision engineering and metrology. CIRP Annals - Manufacturing Technology, 2018, 67 (2), pp.815 - 838. 10.1016/j.cirp.2018.05.003 . hal-01898648

HAL Id: hal-01898648

<https://hal.science/hal-01898648>

Submitted on 18 Oct 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Modelling and traceability for computationally-intensive precision engineering and metrology

J.M. Linares(1)^a, G. Goch(1)^b, A. Forbes^c, J.M. Sprauel^a, A. Clément(1)^d, F. Haertig^e, W. Gao(1)^f

^a Aix Marseille Univ, CNRS, ISM, Marseille, France

^b University of North Carolina, 9201 University City Blvd, Charlotte, NC, United States

^c National Physical Laboratory, Hampton Road, Teddington, Middlesex, TW11 0LW, United Kingdom

^d Dassault Systèmes, France

^e Department of Coordinate Metrology, Physikalisch-Technische Bundesanstalt, D-38116 Braunschweig, Germany

^f Tohoku University, Japan

In contrast to measurements of the dimensions of machined parts realized by machine tools and characterized by CMMs, software results are not fully traceable and certified. Indeed, a computer is not a perfect machine and binary encoding of real numbers leads to rounding of successive intermediate calculations that may lead to globally false results. This is the case for poor implementations and poorly conditioned algorithms. Therefore, accurate geometric modelling and implementations will be detailed. Based on the works of National Metrology Institutes, the problem of software traceability will also be discussed. Some prospects for this complex task will finally be suggested.

Keywords: Geometric modelling, Error, Computational accuracy

1. Introduction

Humanity is facing very big challenges such as global warming, depletion of natural resources, and sustainable development, for example. At the same time, the business world tries to improve industrial competitiveness accounting for these environmental constraints. Most industrialized countries have thus launched development programs to answer these challenges (USA: Advanced Manufacturing, Manufacturing renaissance, National Network for Manufacturing Innovation, India: Make in India, Japan: Innovation 25 program, China: Intelligent Manufacturing, Made in China 2025, European countries: Horizon 2020, Factories of the Future, Industry 4.0...). All these programs are based on the development of computerization and networking in industrial systems. This is the convergence of the physical and virtual worlds to Cyberspace. The German National Academy of Science and Engineering defines this evolution as a 4th generation industrial revolution based on Cyber-Physical Systems (CPS) [84,122,143,172]. In this new environment, information technology (IT) will allow decision makers to do more over time to improve product quality and customisation, productivity and customer satisfaction. This global expansion of the use of computers in industry brings to the forefront the need for traceability and certification of industrial software. Scientific calculations have indeed become central issues in design, manufacturing, precision engineering and metrology software. The fundamental binary code together with all basic arithmetic operations were developed by Leibniz in 1697 [123]. The principle of modern programmable computers was first proposed by Alan Turing in his 1937 paper: "On Computable Numbers with an Application to the Entscheidungsproblem" [167]. The Turing machine is the first universal programmable computer. It invented the concepts of programming and program. The construction of Pilot ACE (Automatic Computing Engine) based on Turing's designs was completed by the National Physical Laboratory in the early 1950's. In 1946, the first architecture of electronic computers was proposed: the ENIAC (Electronic Numerical Integrator and Computer, using the vacuum tube

technology. Figure 1 shows a picture of this computer. The second generation of computers was based on the invention of the transistor in 1947. Despite the use of transistors and printed circuits, the computers were still bulky and only used by universities, governments and large companies. The third generation of computers (around 1959) was based on electronic chips. In 1971, Intel revealed the first commercial microprocessor, the 4004. It did not achieve more than 60,000 operations per second. Today, standard desktop computers have a much bigger processing capacity (for example, an Intel Core 2 Duo processor at 2.4 GHz can execute around 2 billion operations per second). Microprocessors include most of the computing components (except for the clock and the memory) on a single chip. The computer is now the daily companion of people both at the office and in private life. For the average user, the computer remains however a black box that just provides results from the data that were entered. These outcomes are generally considered above any suspicion. For future industrial systems, based on cybernetics, will the results supplied by computers be really traceable and validated numerically? To answer this question, technological incidents could be reconsidered that have happened in recent history and had a computer error as source.

In computing, an integer overflow is a condition that occurs when a mathematical operation produces a numerical value larger than the greatest number that can be represented by the set of bits (binary digits) of the implemented variable. Perhaps the best-known consequence of such an error is the self-destruction of the Ariane 5 rocket during its first launch on June 4th, 1996. The Inertial Reference System (IRS) of Ariane 5 was derived from that of Ariane 4 but the new launcher had high initial accelerations and a trajectory that resulted in horizontal velocities five times larger than those of the previous rocket. These high speeds were captured by the sensors of the inertial platform but exceeded the maximum value that could be processed by the navigation program. It resulted in an integer overflow exception in the IRS software and the shutdown of the computers caused by the conversion from a 64-bit real number to a 16-bit integer [125]. False flight data led then to erroneous

corrections of the launcher trajectory and finally to the self-destruction of the rocket.

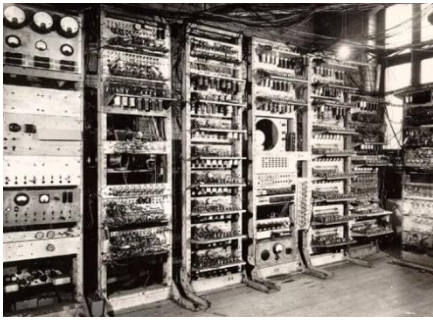


Figure 1: ENIAC (Electronic Numerical Integrator and Computer) [95]

An intrinsic feature of numerical computing is that real numbers are represented in finite precision and this means nearly all real numbers have to be rounded to be represented. The accuracy of the rounding operation can have great influence on calculated results. In 1992, at Dhahran in Saudi Arabia, a Patriot battery failed to track and to destroy a Scud missile [64]. This incident was caused by a software problem in the system's weapon control computer due to an inaccurate calculation of time and consequently of the tracking trajectory. The precision of calculations of on-board computers often depends on the number of bits of its registers. Patriot's clock system was performing some arithmetic operations using a 24-bit fixed point register. This hardware limitation led to a drift between the times elapsed since last boot, as measured by the system's internal clock, and the real delays. Table 1 shows the evolution of this time drift (inaccuracy) and the estimated shift in the range gate that Patriot tracked.

Table 1: Precision of a computer's calculations

Hours	Seconds	Calculated time (s)	Inaccuracy (s)	Shift in range gate (m)
0	0	0	0	0
1	3600	3599.9966	.0034	7
8	28800	28799.9725	.0275	55
20	72000	71999.9313	.0648	137
48	172800	172799.8352	.1648	330
72	259200	259199.7528	.2472	494
100	360000	355999.6667	.3433	687

After 100 hours of monitoring, the elapsed time calculated by Patriot's clock system drifted by approximately 0.3 s. In connection with the speed of the tracked Scud rocket, the resulting error of the calculated interception point was estimated to be about 700 m. Patriot's battery therefore failed to destroy the missile. The size of the registers of current generation of computers is now at least 64 bits, permitting calculations with greater precision but rounding effects are still inevitable.

Can such problems arise in precision engineering and metrology? Whereas there is an infrastructure to provide traceability of dimensions of machined parts realized by machine tools and characterized by CMMs, the results of software calculation are usually not fully traceable and certified. Indeed, a computer is not a perfect machine and binary encoding of real numbers leads to rounding of successive intermediate calculations that may lead to globally false results for poorly constructed calculations. To understand these calculation limits, the second section of the paper will be dedicated to the intrinsic performances of computer hardware and software. False computation results are often due to poor software implementations and badly conditioned or numerically unstable

algorithms. The third section of the paper will therefore deal with detailed smart implementations of geometric modelling. Based on the works of National Metrology Institutes, the problem of software certification and traceability will also be discussed in the fourth section. Some prospects about these different subjects will finally be suggested.

2. Intrinsic performances of computer hardware and software

As discussed in introduction, the hardware (number of bits, number of processors...) and software (conversion effects, rounding effects, cancellation effects...) of computers, have a great influence on the accuracy of the calculated results. These topics will therefore be discussed now. The logical structure and functional characteristics of computers are shown in Figure 2. A computer is built around one or more microprocessors with each microprocessor have one or more cores. The processor (named CPU for Central Processing Unit) is an electronic circuit clocked at the rate of an internal clock. A processor has internal registers of a fixed number of bits (now usually 64 bits) used to encode and manipulate the processed values. Several processor clock pulses are generally necessary to perform an elementary action called an instruction. The indicator, Cycles Per Instruction (CPI), characterizes the mean number of clock cycles required to execute a basic instruction on a microprocessor. It is about four for most current microprocessors. The CPU power can thus be characterized by the number of instructions processed per second and is often expressed in units of millions of instructions per second (MIPS) and corresponds to the frequency of the processor divided by the CPI. The CPU includes one or several Arithmetic Logic Units (ALU) that provide the basic functions of arithmetic calculations and logical operations on integers, and a Floating-Point Unit (FPU) to perform operations on floating point numbers. The processor employs cache memories (buffers) to reduce the time to exchange information between the main computer memory, its Random Access Memory (RAM), and the internal data registers.

Computer microprocessors only manipulate binary instructions and data. The encoding of such information requires two states 0 and 1. In a simplified way: either the electrical current flows through an elementary circuit or it does not. A binary machine language encodes the set of basic instructions implemented in the microprocessor hardware to perform the available elementary operations such as addition, subtraction, multiplication, division, comparison, etc.

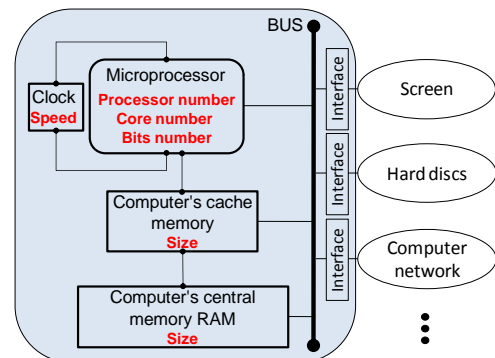


Figure 2: Logical structure and functional characteristics of a computer

The quality of numerical results in scientific calculations performed by a computer will depend on both the hardware and the software. In the remainder of this section, the hardware

aspects will be treated first, and then the software aspects will be presented.

2.1 Technical advances of computer hardware

For use in scientific calculations, the performance of a computer greatly depends on the number of processors and cores, the clock frequency, the number of bits of the CPU registers and the size of the cache memories. The considerable improvements of computers over the past decades are mainly linked to a great increase in the number of the elementary components integrated in the processors. The empirical Moore's law [1], largely verified since 1971, established that the number of transistors in a densely integrated circuit is doubled every two years (Figure 3). This law is unlikely to be satisfied in the near future because of the physical limitations of the actual silicon technology being reached quickly. In fact, due to quantum tunnelling effects, the smallest size of elementary transistors is at present limited to 20 nm [163]. Nevertheless, Intel believes that this size may be reduced to 7 nm by 2020, even if it will perhaps require using materials other than silicon, such as Indium Gallium Arsenide [89]. The computing performance also depends on the length of the data words manipulated by the computer. The size of the internal registers and of the data bus of the first Intel 4004 processor was limited to 4 bits. It has grown since the 1970's to reach 64 bits on current processors.

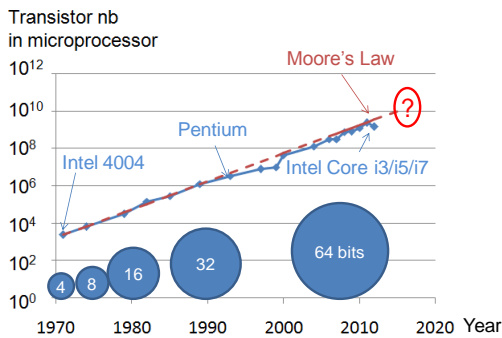


Figure 3: Moore's law

Other technologies are also being developed to replace current hardware based on electronic transistors. In 1982, Richard Feynman suggested that simple controllable quantum systems could be used to simulate the quantum dynamics of problems that cannot be modelled by a conventional computer [56]. In 2012, the Nobel Prize for Physics was awarded for scientific research on ground-breaking experimental methods that enable the measurement and manipulation of individual quantum systems [148]. The quantum properties of matter, such as superposition and entanglement, provide the framework for the development of quantum computers. Figure 4 shows the possible values of the state ψ of a bit and a qubit (quantum bit). A standard computer is based on binary data: a bit has only two independent values 0 or 1. The quantum computer is working with qubits that can have multiple states [42,170]. These possible values (states) can be represented using the Bloch sphere graph where α and β are complex numbers and probability amplitudes. This property allows quantum processors to perform multiple operations in parallel.

Research on quantum computers is very active, with around 8000 publications since 1997. A quantum computer existing today has several hundred qubits. Quantum processors need external cooling down to a temperature of about 0.015° K (around -273° C, very close to absolute zero). The development of quantum informatics based on superconducting circuits also requires accurate readout devices to gather the qubit states

[132]. The quantum computer is well-adapted for combinatorial calculations and uses simulated annealing algorithms for global optimization problems. Some large companies [112] have begun using quantum computers principally in cryptography and optimization problems using simulations (Grover's search algorithm) [42].

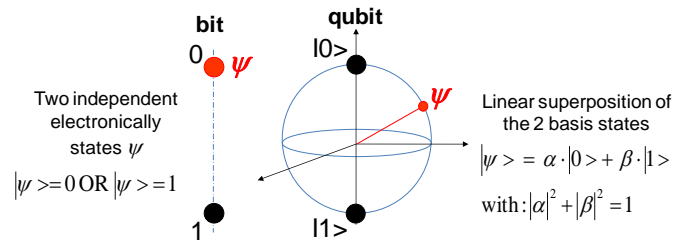


Figure 4: Values of states ψ for bit and qubit

A second set of technologies relies on the fact that silicon is transparent to infrared light, so that optical fibres can be used to interconnect computer elements or components inside the processor core. This opens a new way to build optical computers [146]. For that purpose, an optical nanowire switch was first designed [147] and, by combining two optical switches, a NAND logic gate was then developed. Similarly, in non-linear optics, the property of some materials to change their refractive index under an electric field (Kerr effect) was studied. This permitted creating logic gates (AND, OR, NOR...)[121]. Such components may be employed in the design of future optical processors and computers.

In addition, photons do not produce magnetic interferences with the environment and the heat generated by an optical system remains very low. Optical transistors can work at frequencies much higher than those of conventional electronic devices. Optical computers could thus be more powerful than current conventional computers. The main disadvantage of this technique is the inability to store photons and light. Furthermore, as photons propagate in a straight line, building interconnections causes major difficulties in a reduced space.

Another scientific track to replace current processors is based on the outcome of molecular biology. The principle of DNA computers (computing technology based on molecular biology), enunciated by Leonard Adleman in 1994, is to encode an instance of the problem with DNA strands and manipulate it by molecular biology to simulate operations that will isolate the expected solution of the problem [2,3]. As for quantum technology, DNA computers will be specialized in computing complex problems like non-deterministic algorithms in polynomial time because DNA strands can produce billions of potential answers simultaneously. However, the process of synthesis and reaction is very slow.

2.1.1 Technical properties of internal computer devices

Current processors work on the scheme of classical Turing machine and are constrained to perform calculations in sequence. The consequence is that it is less promising to deal with, in a given time, a large number of instances of computational problems of high numerical complexity. As shown in Figure 2, the performance of a computer is related to the number of processors and cores, the number of bits of the internal registers and the data bus, the clock and bus frequencies and the size of the cache memory.

A global metric was proposed to measure the theoretical computer performance in scientific calculations that use floating-point operations: floating-point operations per second (FLOPS). Equation 1 shows FLOPS formula.

$$FLOPS = N_{processor} N_{core/processor} f_{processor\ clock} \frac{FLOPs}{cycle} \quad (1)$$

With:

$N_{processor}$: Number of processors in microprocessor unit,

$N_{core/processor}$: Number of cores in processor,

$f_{processor\ clock}$: Frequency of processor clock,

$$\frac{FLOPs}{cycle} \approx 4 \text{ with actual processors.}$$

For a scientific computer with 2 processors containing 12 cores each and working at a clock frequency of 2.9 GHz, the theoretical number of floating-point operations per second is 278.4 GigaFLOPS. A laptop with a single-core 2.5 GHz processor has a theoretical performance of 10 GigaFLOPS. This metric is sometimes divided by the electrical power (FLOPS/watt) to analyse the energy efficiency of the computer. The number of FLOPS is used to compare the theoretical performance of computers but does not account for specific tasks of the computation and for the real load rate of each processor and core. New benchmarks are therefore proposed by the Standard Performance Evaluation Corporation, launched in 2000, to compare computer performances. These benchmarks are based on specific procedures applied to test the computer behaviour when running next-generation of industrial software (Dassault Systèmes: CATIA and Solidworks, Pro Eng: CREO, Siemens: NX...), to stress a system's processor, etc... [83]. These tests highlight the processor ability to process a set of operations in a limited time or to give global information on the computer's behaviour. But these tests do not give an indication of the traceability or the quality of the numerical results provided by the computer.

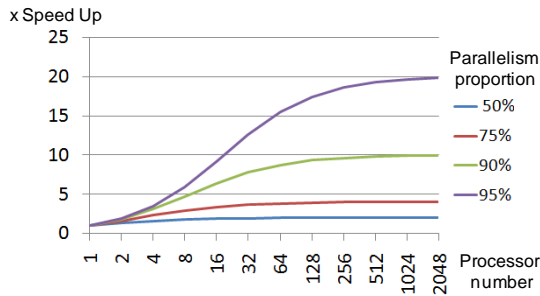


Figure 5: Amdahl's law

Equation 1 shows that the number of processors and cores influences the performance of calculation. Multiprocessor computers allow the program to complete several arithmetic operations simultaneously, thus increasing the processing capacity. This technique is called parallelism. The technique of parallelism can be used inside a processor to address the cores, between processors or for a pipeline technique. In the third case, the processor can start executing a new instruction without waiting for the previous one to be completed. To reduce the input/output bottleneck of instructions, a vector processor was developed with specific instructions optimized for fast handling of tables and quick matrix calculations.

Theoretically, it is expected to halve the processing time by sharing calculations between two processors with single core, to quarter the processing time by using 4 processors, etc. Unfortunately, not all scientific operations can be parallelized effectively.

$$Speedup = 1 / [(1 - \alpha) + \alpha / P] \quad (2)$$

With:

α : proportion of parallelism

$$P = N_{processor} N_{core/processor}$$

The empirical Amdahl's Law (Eq.2) can be used to define an upper limit to the parallelization contribution of software or hardware architectures [4,88,145]. It assumes a constant amount of data to be processed. Figure 5 shows the speedup of calculations versus the processors number used and the proportion of parallelized computer code. In Figure 6, another empirical law (Eq.3) is shown, known as the Gustafson-Barsis' relationship [78,96,113]. It is more optimistic than the previous one and reflects the fact that more data can be processed at the same time by increasing the number of processors.

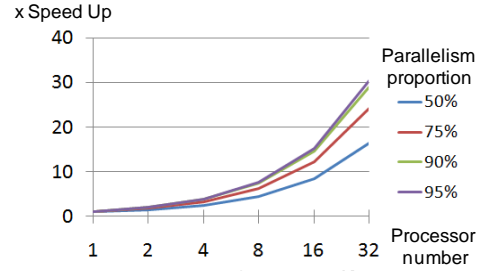


Figure 6: Gustafson-Barsis' law

$$Speedup = P - (1 - \alpha)(P - 1) \quad (3)$$

With:

α : proportion of parallelism

$$P = N_{processor} N_{core/processor}$$

2.1.2 Supercomputers and LINPACK/LAPACK benchmarks

A supercomputer is a computer designed to achieve the highest performances at the date of its manufacture. Its use is targeted to High-Performance Computing (HPC). These supercomputers have thousands of processors and hardware architectures allowing them to use the benefits of parallelism. To classify the most efficient machines in scientific computing, a TOP500 classification project was created. The LINPACK benchmark is used to test and to rank supercomputers for the TOP500 list [41,45,92]. It measures the time taken to solve a dense linear system of n equations in n unknowns, the solution being obtained by a partial Gaussian elimination, by $2/3 n^2 + n^3$ floating point operations [73]. The performance is then calculated by dividing the number of operations by the calculation time. To complement the FLOPS metric, two other indicators were created for this benchmark:

- R_{max} : maximum performance in LINPACK Giga FLOPS for the biggest computable mathematical problem on the computer,
- N_{max} : size of the mathematical problem giving R_{max} computable on the machine.
- R_{peak} represents the theoretical performance in GigaFLOPS of the computer.

To extend the use of LINPACK packages on computers using shared-memory vectors and parallel calculations, a new LINPACK package was introduced in this benchmark [7,29,44]. This software pack is being constantly improved, particularly in terms of accuracy and performance [46]. Table 2 presents the ten first supercomputers of the world. After three consecutive years as the world's No. 1 system of the Top500 ranking, Tianhe-2 of National Super Computer Center has been exceeded in performance by Sunway Super Computer. This computer has 10,649,600 cores and needs 15,371 kW of electrical power to obtain its best calculation performance.

Table 2: TOP500 ranking in June 2017

Top 500	Rmax	Year	Name	Total Cores	Country	Nmax	Power (kw)
1	93015	2016	Sunway	1.1E+07	China	12288000	15371
2	33863	2013	Tianhe-2	3120000	China	9960000	17808
3	19590	2017	Piz Daint	361760	Switzerland	3569664	2272
4	17590	2012	Titan	560640	United States	0	8209
5	17173	2011	Sequoia	1572864	United States	0	7890
6	14015	2016	Cori	622336	United States	6984960	3939
7	13555	2016	Oakforest	556104	Japan	9938880	2719
8	10510	2011	Riken	705024	Japan	11870208	12660
9	8587	2012	Mira	786432	United States	0	3945
10	8101	2015	Trinity	301056	United States	8847936	4233

In 2005 and in line with the new challenges of this world (global warming, resource reduction, sustainable development), a new ranking of supercomputers has been set up: Green500 [91]. It incorporates the calculation concepts used in the TOP500 ranking, but it is based on a new metric for supercomputer ranking: the power-performance defined by the number of FLOPS per Watt (FLOPS/W). Green500 proposes a ranking of the most energy-efficient supercomputers in the world. Table 3 presents the ten supercomputers ranked using this new metric. At present, the Tsubame 3.0 heterogeneous supercomputer (Tokyo Institute of Technology) obtains the top spot in the Green500 list and currently claims the title of the most energy-efficient (or greenest) supercomputer in the world. The Tsubame 3.0 heterogeneous supercomputer surpassed the fourteen gigaflops/watt milestone [91]. To maximize the power-performance metric, computer manufacturers use specialized cards named High Performance Computing (HPC) including many-core accelerators, on which parts of the computations are subcontracted. These new many-core accelerators are coupled to the CPU with an energy-efficient software design. For example, computer with HPC cards can treat large data up to 10 times faster than a single CPU.

Table 3: Green500 ranking in June 2017

Green 500	Top5 00	MFLOP S/W	Year	Name	Total Cores	Country	Rmax
1	61	14110	2017	TSUBAME3.0	36288	Japan	1998
2	466	14046	2017	kukai	10080	Japan	461
3	148	12681	2017	AIST AI Cloud	23400	Japan	961
4	306	10603	2017	RAIDEN GPU	11712	Japan	635
5	100	10428	2017	Wilkes-2	21240	UK	1193
6	3	10398	2017	Piz Daint	361760	Switzerland	19590
7	69	10226	2017	Gyokou	3E+06	Japan	1677
8	220	9797	2017	Res. Comp. Facility	16320	Japan	770
9	31	9462	2017	Facebook	60512	US	3307
10	32	9462	2016	DGX Saturn V	60512	US	3307

Without fundamental change in the design of supercomputing systems, the computer performance advances will not continue at their current pace [55,151].

2.1.3 Sub conclusion

The improvement of computer performance is now impacted by the constraints of sustainable development. The tests conducted under Green500 benchmark show that the manufacturers direct the development of their supercomputer to heterogeneous machines using High Performance Computing accelerators. These new hybrid systems, although energetically optimized, are still based on electronics. This technology is now well under control, but the resistivity of the circuits causes significant energy loss as heat. For example, the data can require up to 80% of the power consumed by a microprocessor. To solve this problem, new technologies have been proposed based on quantum physics, optics or molecular biology. The systems derived from these technologies are however specialized. They have functionalities

similar to the accelerators used in actual heterogeneous computers. Such new devices will therefore surely be coupled to the machines designed with current technology, thus allowing continuous improvement of the computer performance. Figure 7 summarizes the hardware items necessary to enhance and to optimize the computing performance of computers. However, the performance of computers not only depends on the hardware, but also on the manner in which data processing is implemented, i.e. the software. For most scientific calculations (outside grand challenges) computational time is usually not a limiting factor, and solutions to a sufficient accuracy can be determined in an acceptable amount of time. The precision of calculated results mainly depends on the quality of the data processing, in particular on the way the software is implemented.

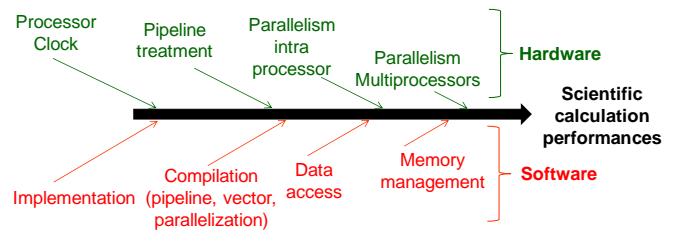


Figure 7: Parameters influencing scientific calculation performances

2.2 Technical advances of computer software

Transistor gates of current computer processors can only handle binary information, in two distinct states 0 and 1, called bit. At the hardware level, n bits are then gathered to build words that are transmitted to the internal registers of the processors and cores. They are treated as instructions or operands for arithmetic and logic operations. The set of instructions of the machine language directly achieves basic mathematical operations on integer numbers (addition, subtraction, multiplication, division, modulus ...). A floating-point unit is also generally embedded in the processor (or connected to it) realizing arithmetic calculations on real numbers and computing classical mathematical functions (sine, cosine, exponential, power, square root, ...). These operations, performed at the hardware level, are very efficient, but the range of the manipulated data and the accuracy of some calculation results are limited by the size of the internal processor registers. If a greater range or precision is required, the number of bits used to encode a number has to be increased. A software layer will then be added to perform the operations. The first electronic computer, the ENIAC, used decimal arithmetic. In most current computers, the use of binary encoding has however been generalized, principally for its calculation simplicity and its coherence with the hardware [37]. If another encoding base (decimal, hexadecimal ...) is chosen to represent a number, it will be necessary to add a software layer to achieve encryption and calculations. IEEE 754 standard provides the rules of encoding real numbers for binary floating-point arithmetic. However, only its last revision in 2008 defines the encoding of numbers in binary coded decimals. For a better understanding, the effect of type and base of encoding and the number of bits on the accuracy of calculations in precision engineering and metrology has to be explained.

2.2.1 Implementation properties

Computer hardware is designed with registers of fixed size, thus leading to a limited accuracy when performing arithmetic operations with the floating-point processor units. However, using programming methods, the number of bits used to represent numbers can be increased. Nevertheless, at the same time, the size required to store this information grows and may exceed the maximum available capacity of memory.

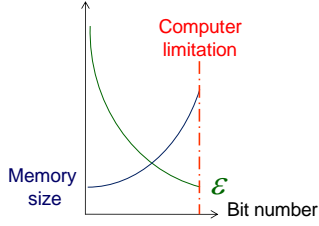


Figure 8: Limitation of computer encoding

Figure 8 shows that with a finite memory capacity, the software cannot decrease the gap ϵ between two successive real numbers (the precision of the floating-point arithmetic) to zero. Therefore, all real numbers cannot be represented by a computer. This is the first limit of computer calculations. Two phenomena are induced by this limit: rounding and cancellation [72]. In this context, floating point calculations can give entirely inaccurate results when no particular precaution is taken. To illustrate this, the computation of the simple function f of the variable M might be considered, presented in Equation 4 [140,19].

$$f(M) = M + M^2 + 1 - M - M^2 \text{ gives } 0 \text{ for } M > 10^8 \quad (4)$$

For large M values (in this case when M exceeds about 10^8) the value returned by software implementing IEEE arithmetic [100] is 0, while the true result is of course 1. The usual computer addition process is sequential as shown in Figure 9.

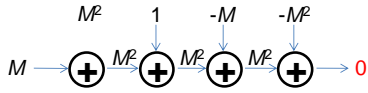


Figure 9: Usual computer addition process [19]

To compensate this bias, more efficient calculation methods have been developed but they are not automatically implemented in all usual programs. For example, computer calculations using a Two Sum algorithm (TS) [116] give true results (Figure 10).

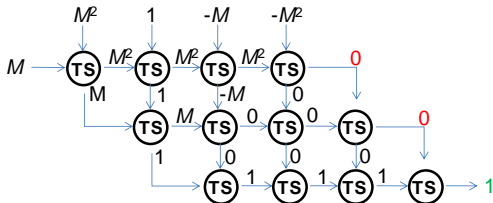


Figure 10: Two Sum algorithm [19]

If the order of operations in Equation 4 is changed (Figure 11), the right calculation result is also obtained. Therefore, the way the equations are implemented in the computer affects the quality of the result, which is called the implementation effect.

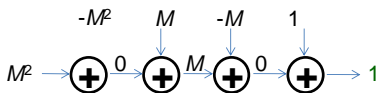


Figure 11: Implementation effect

2.2.2 IEEE 754-1985-2008 standard

For most computers, the representation of numbers in binary and decimal floating-point calculations is based on IEEE 754-1985-2008 standard [100,133,140], where binary floating point is commonly used in the computer field. This standard also defines special values.

2.2.2.1 Calculations with floating numbers

Calculation based on floating point encoding is used the most in computing, but problems may arise when only a few bits represent the real numbers because of the limited accuracy.

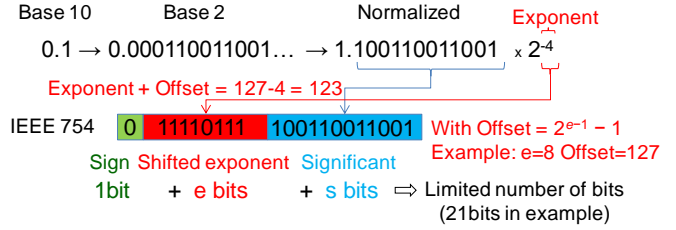


Figure 12: IEEE 754-2008 standard (floating point number)

The encryption of numbers and implicit conversions between the decimal and binary system during data input/output are the first source of rounding. This process is shown in Figure 12. In consequence, not all the decimal numbers (i.e., rational numbers, the ratios of integers) can be expressed exactly as binary floating-point numbers. As shown in Equation 5, a real number written using IEEE standard is defined by three integers: sign, shifted exponent and significant.

$$\text{Encoded number} = \text{sign} \times \text{significant} \times 2^{(\text{shifted exponent} - \text{offset})} \quad (5)$$

with $\text{offset} = 2^{e-1} - 1$, e being the bit number of the exponent

The data is represented in a scientific form. The most significant bit gives the sign, set to 0 when the number is positive and 1 if it is negative. The next e -bits define the exponent, shifted by a fixed offset to avoid negative values. Finally, the last binary digits define the normalized significant, truncated to the available numbers of remaining digits. Because the most significant bit is always 1 for a normalized number, this bit is not stored in the mantissa and is called the "hidden bit". The expression of the exponent offset and the whole formula permitting the definition of the real number are presented in Equation 5. In Figure 12 this representation was used to encode the real number 0.1 using a register of 21 bits. If this floating number is reconverted to base 10, the decimal value 0.099990845 is obtained. It clearly shows the rounding effects due to the limited accuracy of binary floating numbers. In double precision (encoding with 64 bits), the smallest positive and greatest negative normalized number different from zero are: $\pm 2^{-1022} \approx \pm 4.941 \times 10^{-308}$. As the mathematical infinite and the mathematical zero cannot be encoded, three specific exceptions are considered in IEEE-754-1985-2008 standard:

- the exponent offset and mantissa are both zero: the real number is ± 0 (according to the sign bit),
- the exponent offset is equal to $2^e - 1$, and the mantissa is zero: the number is \pm infinity (according to the sign bit),
- the exponent offset is $2^e - 1$, but the mantissa is not zero: the number is NaN (not a number).

To limit the rounding effects of numbers translated into base 2, the following rounding procedures have been included in IEEE-754-1985-2008 standard [133,140,38]. The four rounding modes [100] are:

- Round toward $-\infty$: $RD(x)$ is the largest floating-point number less than or equal to x ,
- Round toward $+\infty$: $RU(x)$ is the smallest floating-point number greater than or equal to x ,
- Round to nearest: $RN(x)$ is the floating-point number that is the closest to x ,

- Round toward zero: $RZ(x)$ is the closest floating-point number to x that is no greater in magnitude than x . It is equal to $RD(x)$ if $x \geq 0$, and to $RU(x)$ if $x \leq 0$.

Many studies have been conducted on the effect of rounding in the calculation of basic functions ($-$, $+$, x , $/$, \exp , \log ...) [40,75,110, 118,119,135,176]. In calculations with floating point numbers, the implementation of arithmetic operations [141]: addition [140], subtraction, multiplication, division, square root [111], fused-multiplication-addition [128] are the basis of scientific calculations. A lot of research work has been carried out to make these functions more precise, stable and robust despite the inaccuracy induced by the encoding of real numbers.

A Graphics Processing Unit (GPU) is an electronic circuit on a graphics card or CPU. It performs the computations required by the display functions. The GPU has a parallel structure. It is designed for the great number of calculations required by the realization of 3D rendering, for example. The manufacturers of these components had to develop effective and precise algorithms to perform such computations. Classical basic arithmetic functions, vector operations (dot product, cross product ...) and matrix calculation algorithms have thus been developed for graphical display and have been incorporated in the GPU hardware. These efficient procedures are accessible to software developers and are sometimes used to improve the performance of programs [174].

The guidelines of the IEEE-754-1985-2008 standard must be included in the programming of business software used in metrology and precision engineering. However, most traditional software has not yet implemented these guidelines.

2.2.2.2 Calculation with Decimal Floating-Point using Binary Coded Decimal

The last revision of the IEEE-754 standard introduced Decimal Floating-Point (DFP) formats to encode real numbers in scientific calculations [36,142]. The first domains of application were financial analysis, tax calculation and others, where accurate calculation is sought to avoid mistakes causing financial losses. The numerical value of a number is given by $sign \times significant \times 10^{Exponent}$. The mantissa and the exponent can be described in several ways: both values in Binary Coded Decimals (BCD), significant in BCD and exponent in binary or more complex compressed formats. Two methods were proposed by Intel [34] and IBM[35]: The first named binary integer decimal (BID), encodes the significant of the DFP number as an unsigned binary integer. The second, named densely packed decimal (DPD), encodes the significant as a compressed binary coded decimal integer.

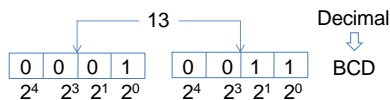


Figure 13: Binary Coded Decimal (BCD)

Finally, different ways exist to encode the same number in decimal floating-point. One way to code a number based on the intrinsic properties of a binary computer is to use the BCD method. The numbers are then represented by decimal digits, each of them encoded on four or eight bits (Figure 13). The basic arithmetic operations and mathematical functions have then been developed to make this DFP encoding method more accessible [8,30,51,53,74]. DFP is not yet used on all hardware/software products. Therefore, only binary coded floating-point numbers will be considered in the remainder of this section.

2.2.3 Numerical effects of ϵ -approximation and implementation

A floating-point number often results from a set of operations performed on a real machine that tries to compute a given function $F(X)$ with the best possible approximation. The notion of calculability was first introduced by Turing [167] as follows: a real number X is "computable", if there is an algorithm that takes $\epsilon > 0$ as input and produces an approximation x of X with $|x-X| < \epsilon$. To realize that, an exact real number X is converted and rounded by an algorithm to obtain a floating-point number approximation x . The error ϵ between these two numerical values mainly depends on the coding base, the number of bits used in coding (truncation error) and the rounding error.

In the following, the computing of a given function result $Y=F(X)$ is investigated. In such a procedure, the exact argument X first needs to be stored in the computer memory and therefore must be converted to its floating-point approximation x , thus introducing an input error $\Delta X=x-X$. The floating-point number x is next entered in a sequence of software instructions that are implemented to generate the desired function F . However, due to successive truncation and rounding errors and perhaps inaccurate algorithms, the function really realized by the calculator, denoted as f , may significantly differ from F . Therefore, the floating-point result $\bar{y}=f(x)$ calculated by the computer may itself deviate from $F(x)$. To be displayed or printed, this number is finally rounded and formatted to the decimal approximation y delivered to the user (Figure 14). The absolute computing error (output error) is then characterized by the difference ΔY between the value y given by the machine and the exact mathematical result $Y=F(X)$, i.e. $\Delta Y=y-Y$. It is divided by the true exact value Y to define the relative error δY i.e. $\delta Y=(y-Y)/Y$. Figure 14 shows the effect on the absolute error ΔY of the implementation of the function F in f . With poor implementation, the absolute error ΔY increases. This phenomenon is called data sensitivity.

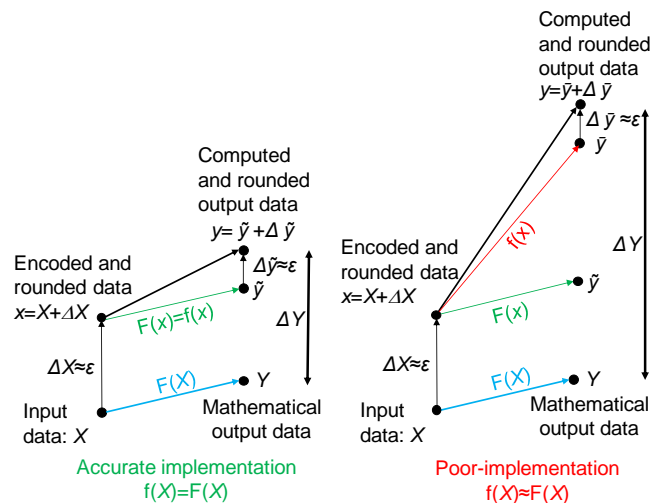


Figure 14: Computing error

In case of poor implementation, the absolute error ΔY can be divided in three main error contributors (Figure 14):

- The computational error ($Y-\bar{y}$) where $\bar{y}=F(x)$: It is only sensitive to the input error ΔX and the computer algorithm stability. ΔX is caused by the truncation or rounding of the exact real number X during encoding as floating-point.
- The Implementation error ($\bar{y}-y$) where $\bar{y}=f(x)$: The input error ΔX is amplified by errors of modelling and implementation. It must be pointed out that this error is not related to measurement uncertainties, even if poor and unstable programming may also amplify the dispersions of the measurement data entered in a

metrology or precision engineering software. Propagation of uncertainties in software is a topic that has already been considered in many research works [14,16,127,130,131]. This topic has also been treated in a fundamental keynote paper entitled "Measurement as inference" [52]. Measurement uncertainties are therefore not considered significantly in this paper. Error propagation is a research field intensively treated in laboratories of mathematics and computer science. Generally, the understanding of the accuracy of floating-point programs is based on the estimation of the condition number. It is an indicator that permits to know the algorithm behaviour when it is calculated at each program line. Condition number will be detailed in subsection 2.2.3.3.2 (Mastering of forward error).

- The error $(\hat{y}-y)$ generated by the conversion of the computed result to the decimal number displayed or printed to the user.

These numerical effects will be discussed in detail in the next subsections.

2.2.3.1 Loss of specific properties of arithmetic operations

Due to cancellation, rounding, overflow and underflow effects, careful implementation is required in the coding of sets of arithmetic operations. In fact, the commutative, associative and distributive properties of the operators may be lost in calculations performed with floating numbers and coded with a limited number of bits, i.e. $(A+B)+C \neq A+(B+C)$, $(AB)C \neq A(BC)$, $(A+B)C \neq AC+BC$, ... [33,93,94].

2.2.3.2 Numerical effects of cancellation or overflow

The subtraction of two nearly equal floating-point numbers may lead to a significant loss of relative accuracy. This phenomenon is named catastrophic cancellation.

$$f = 21bb - 2aa + 55bbb - 10aabb + a / 2b \quad (6)$$

with : $a = 77617$ and $b = 33096$

To demonstrate the effect on the cancellation of the number of bits used to encode floating-point numbers, the computing of Equation 6 represents a good example [152].

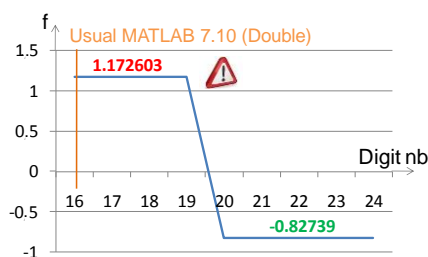


Figure 15: Effect of the number of bits on calculation

This function was computed with floating-point numbers of increasing bit length. The result is shown in Figure 15. As soon as the number of bits exceeds a size that enables the display of 19 decimal digits, an accurate value is returned by the software. Below this, the calculation result is false. In conclusion, to reduce the numerical effect of the cancellation, the number of bits must be increased. Arithmetic operations can also undergo overflow and underflow.

2.2.3.3 Backward and forward errors analyses

Forward and backward error analyses are two paradigms used to study error analysis and data sensitivity [28,32,87,175,176,181]. The backward error is defined as the

estimated input error $\Delta\hat{X}$ that would lead to a given fixed absolute error ΔY after computation. Its calculation process is shown in Figure 16. The forward error is the absolute error ΔY calculated for a fixed input error ΔX .

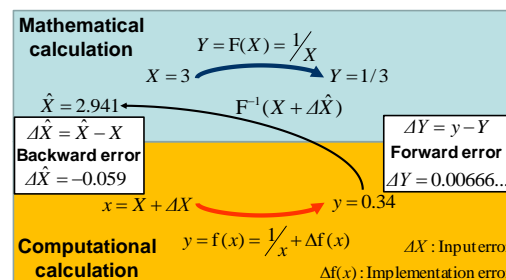


Figure 16: Calculation processes of forward and backward errors

The analysis of the change in backward error in relation to the forward error allows studying the quality of algorithms. This connection can differ from one geometrical problem to another. It can be summarized by the classification of Figure 17.

		Forward error	
		Small	Large
Backward error	Small	Accurate implementation	Ill-conditioned problem
	Large	Accuracy insensitive to implementation Stable process	Reducing backward error Unstable process

Figure 17: Forward error versus backward error

A large forward error of a computation can have the following sources:

- the amplification of a known input inaccuracy onto the output error. It can be characterized by the condition number. The condition number is a mathematical property of an algebraic function F and is therefore independent of the algorithm or the computer used to evaluate an expression.
- the amplification of some truncation and rounding errors generated by the algorithm used to compute the desired function. It depends on the number of bits used to encode the floating-points and can be reduced by implementing multi-precision calculations. This effect is called: stability or instability of the calculation process.

To minimise the numerical effect of ε -approximation and implementation, the backward error, the condition number and the process stability must be managed with caution.

2.2.3.3.1 Reduction of input error

With floating-point numbers, the input error mainly depends on the quality of the encoding and rounding. Increasing the number of bits is therefore an obvious solution to reduce backward errors in floating-point calculations. Some modern commercial floating-point computing software provide functions (such as the function `eps()` of Matlab that computes the floating-point relative accuracy) that allows the estimation the ε error. Figure 18 shows the result of the computation of constant π (pi) using Matlab software. The number was calculated with standard double precision floating-points and was displayed with 15 digits after the decimal point. Its relative and absolute errors were also defined. The same constant π was also computed with Excel spreadsheet. The result: 3.14159265358979 was obtained.

>> pi

```

ans = 3.141592653589793
>> eps(pi)
ans = 4.440892098500626e-016
>> eps(pi)*pi
ans = 1.395147399203453e-015

```

Figure 18: Numerical value of constant π , relative and absolute errors in case of calculations with double precision floating-point numbers

One digit was thus lost in comparison with Matlab. Multiple precision toolboxes are also available in some software systems to perform calculations with high fixed arbitrary precision. Quadruple precision floating-point numbers, compliant with IEEE 754-2008 standard, are also now progressively introduced in programming tools and hardware. It will permit computations with a precision of about 34 decimal digits. Figure 19 finally shows the result obtained with 300 decimal digit precision [90]. Such multiple precision computations can be realised with Maple or Mathematica software without any specific library. They require however enough memory space to encode the real numbers (Figure 8) and may lead to large computing times.

```

>> mp.Digits(300)
>> mp("pi")
ans=3.1415926535897932384626433832795028841971693993751058
209749445923078164062862089986280348253421170679821480865
132823066470938446095505822317253594081284811174502841027
019385211055596446229489549303819644288109756659334461284
756482337867831652712019091456485669234603486104543266482
1339360726024914128

```

Figure 19: Numerical value of constant π with 300 decimal digits

Figure 20 shows the increase of the calculation time needed to compute the constant π as a function of the number of encoding decimal digits. Various C libraries dedicated to multiple-precision floating-point computations with accurate rounding have already been developed (GNU-GMP, GNU-MPFR, FLINT, MPIR). Most of them are open-source and permit calculations with an arbitrary multiple precision. Links to such libraries are available in [168].

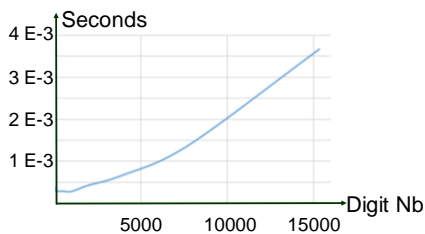


Figure 20: Calculation time of π in multiple precision

The calculation times must also be evaluated for computations requiring complex numerical operations (matrix inversion, least squares optimizations...). The U.S. National Institute of Standards and Technology (NIST) publishes a set of statistical reference datasets using multiple precision calculations with an accuracy of 500 decimals. Another solution to reduce the input error is to use BCD encoded numbers. However, industrial software currently does not offer toolkits to develop in BCD. Nevertheless, the potentialities of the actual object-oriented programming languages, such as C++, permits the overload of all the mathematical operators and most arithmetic functions. Thus, generic algorithms developed with classical floating-point numbers can be reused. This will permit evolving towards BCD processing [21].

2.2.3.3.2 Mastering of forward error

Equation 7 presents the demonstration of the condition number C in the case of a single-variable nonlinear function F . The relative

forward error is the result of the multiplication of the condition number C by the relative input error δX .

$$\delta Y = \frac{\Delta Y}{Y} = \frac{y - Y}{Y} = \frac{f(x) - F(X)}{F(X)} = \frac{f(X + \Delta X) - F(X)}{F(X)} \quad (7)$$

with : $f(X + \Delta X) \approx F(X + \Delta X) \approx F(X) + \Delta X F'(X)$

$$\delta Y \approx \frac{\Delta X}{X} \frac{X F'(X)}{F(X)} \approx \frac{X F'(X)}{F(X)} \delta X \approx C \delta X$$

In these expressions, X is the exact input argument of the mathematical function F to be calculated; F' is the derivative of F ; $f(x)$ represents the result provided by the computer; ΔX is the absolute input error; δX is the relative input error; δY is the relative forward error and C is the condition number. It must be pointed out that the approximation $f(X + \Delta X) \approx F(X + \Delta X)$ is only valid in case of accurate and stable software implementations. The Taylor expansion used in Equation 7 requires also a limited input error ΔX .

The condition number C represents an intrinsic property of the mathematical problem and has nothing to do with the computer. A computing condition number c can therefore also be defined. It takes the floating-point approximation x of X as input argument and is based on the result $f(x)$ really provided by the program implemented in the calculator. Equation 8 shows such definition and demonstrates that the mathematical meaning of the condition number may lead to inaccurate estimations of computing errors because it does not account for deviations caused by inaccurate or unstable implementation.

$$C = \frac{X F'(X)}{F(X)} \Rightarrow c = \frac{x f'(x)}{f(x)} \quad (8)$$

Equation 9 shows the symbolic definition of the condition number C . It is the ratio between the relative output error (change in output) and the relative input error (change in input).

$$C = \frac{\delta Y}{\delta X} \quad (9)$$

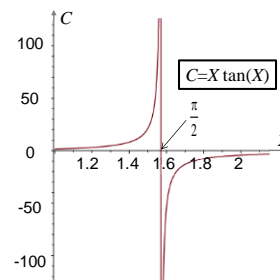


Figure 21: Behaviour of the condition number of the Cosine function

Estimating the condition number C (or c) is very important in understanding the accuracy of floating point software. When C (or c) is large, the relative input error is amplified and the accuracy of the computing results becomes poor. A problem characterized by a low condition number is called "well-conditioned", otherwise it is named "ill-conditioned". The condition number C may greatly depend on the input argument X . Thus, it can take large values up to infinity and this even for simple mathematical functions (cosine, sine, tangent...), this amplifies errors in scientific computation. As an example, Figure 21 shows the behaviour of the condition number calculated for the function cosine.

In case of a nonlinear differentiable function F of multiple variables X , the condition number is defined by equation 10,

where $J(X)$ represents the Jacobian matrix of F and $\|\dots\|$ is a given chosen matrix norm (usually the Euclidian norm).

$$C = \frac{\|X\| \|J(X)\|}{\|F(X)\|} \quad (10)$$

For non-singular systems of linear equations, given in the form $\mathbf{A}\mathbf{X}=\mathbf{Y}$, the condition number $\kappa(\mathbf{A})$ is computed as follows:

$$\kappa(\mathbf{A}) = \|\mathbf{A}^{-1}\| \|\mathbf{A}\| \quad (11)$$

Its value thus depends on the choice of norm. With the Euclidian norm, the condition number C is the ratio of the largest to the smallest singular value in the singular value decomposition (SVD) of a matrix.

The effects of the input error propagation can be mitigated by reducing the number of floating-point operations (FLOPs) performed in a computation. Matrix computation is intensively used for the implementation of approximation methods in metrology and precision engineering software. In the case of linear equation systems, the Gauss pivoting method is often used to obtain the solution. In a problem with n equations and n unknowns, $\lfloor n(n+1) \rfloor / 2$ divisions and $\lfloor (n(n-1)(2n+5)) \rfloor / 6$ additions and multiplications are needed to obtain the result. The Gauss pivoting method should therefore only be used for linear systems of limited size (less than a thousand of unknowns). For large systems (e.g. for Finite Element calculations) specific algorithms are to be used.

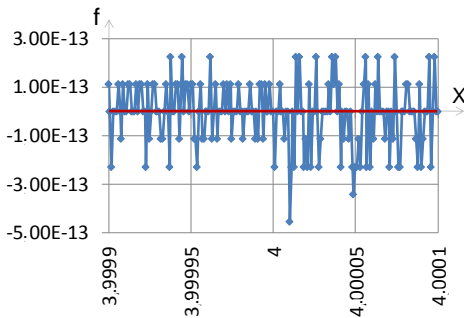


Figure 22: Stability or instability of computation

2.2.3.3.3 Calculation process stability

To illustrate the stability or instability of the calculation process [19], Figure 22 presents the behaviour of two implementations: f_1 and f_2 (Equation 12) of the same mathematical function F , when the input argument X tends to 4. The results plotted in blue are obtained from relation f_1 , while the red curve results from implementation f_2 . These computations were performed using MATLAB 7.10 software and standard double floating-point precision.

$$\begin{aligned} f_1 &= X^4 - 16X^3 + 96X^2 - 256X + 256 \\ f_2 &= (X - 4)(X - 4)(X - 4)(X - 4) \end{aligned} \quad (12)$$

The first implementation f_1 is affected by the rounding errors of the floating-point calculations. This impacts the stability of the calculation process. A computing process without subtractive cancellation is usually stable, especially when a small number of numerical operations is used. A few guidelines can be found in [87]. Backward error calculations can be used to test the stability of the calculation process (method) applied to solve linear equation systems [28,87]. If the backward error is small, it means that the result y found by the computer is close to the true

solution Y of the mathematical problem. To improve the stability of a numerical process, a scaling of the data can lower the condition number. In linear equation systems, a nearly optimal strategy is to equilibrate the rows or columns of the associated matrix.

2.2.3.3.4 Interval and ball arithmetic

The principle of interval arithmetic (IA) is to encode a real value by an interval provided by the computer. This interval evolved to shapes in N dimensions. Since the 1960s, this topic was intensively studied [48, 86, 114, 120, 137]. The input interval x can be represented by its lower \underline{x} and upper \bar{x} endpoints (interval arithmetic) or as a centre x_c and a radius r_x (ball arithmetic). The 1788-2015-IEEE Standard for IA defines basic IA operations of the commonly used mathematical interval models (Equation 13). IA estimates the upper and lower limits of an output, calculated from a set of inputs bounded by intervals.

$$\begin{aligned} x + y &= [\underline{x}, \bar{x}] + [\underline{y}, \bar{y}] = [\underline{x} + \underline{y}, \bar{x} + \bar{y}] \\ x - y &= [\underline{x}, \bar{x}] - [\underline{y}, \bar{y}] = [\underline{x} - \bar{y}, \bar{x} - \underline{y}] \\ x \cdot y &= [\underline{x}, \bar{x}] [\underline{y}, \bar{y}] \\ x / y &= [\underline{x}, \bar{x}] \left[\frac{1}{\underline{y}}, \frac{1}{\bar{y}} \right] \text{ if } \underline{y} \neq 0 \text{ and } \bar{y} \neq 0 \\ x \cdot y &= [\min(\underline{x} \underline{y}, \bar{x} \underline{y}, \underline{x} \bar{y}, \bar{x} \bar{y}), \max(\underline{x} \underline{y}, \bar{x} \underline{y}, \underline{x} \bar{y}, \bar{x} \bar{y})] \end{aligned} \quad (13)$$

The “dependency problem” that may lead to large over-estimations of computation errors, is a major difficulty in the application of IA. Very early, the wrapping effect of interval arithmetic was brought to the forefront. This effect is well introduced in the presentation of the one-dimensional problem detailed in [177].

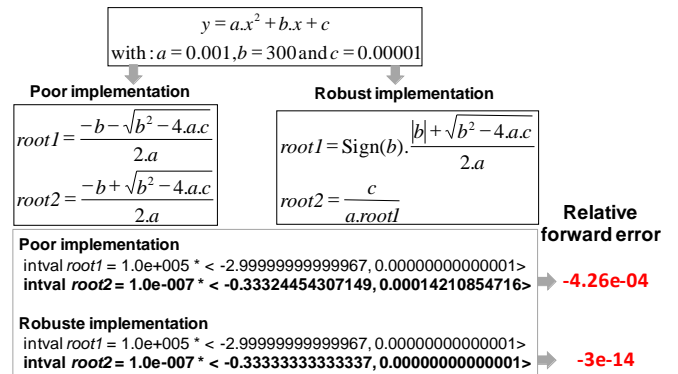


Figure 23: Self-validating implementation with ball arithmetic, applied to roots of a single-variable quadratic equation

To reduce its impact, coordinate transformations can be used. In N -dimension domains, new shapes for interval boundaries were also chosen as polytopes or ellipsoids [169]. For dynamic problems, Chebyshev or Interval Newton methods can be applied to solve nonlinear functions with intervals. Ball arithmetic seems to partially solve the over-estimation of computation errors. Many libraries for interval arithmetic (GNU Octave) or ball arithmetic (Mathemagix [169]) have already been developed. Matlab library INTLAB [153] also proposes tools to perform IA calculations. In addition, arithmetic intervals are also handled by Computer algebra systems, such as Mathematica or Maple. To illustrate the use of interval arithmetic for the self-validation of an algorithm, Figure 23 presents the results of the calculation of the roots of a single-variable quadratic equation.

2.2.4 Sub conclusion

In scientific computation, the limits of the use of floating-point numbers were intensively studied during the last fifty years (with activity intensified during the run up to the year 2000 due to anticipated problems with “millennium bugs”). Computational errors and algorithm instabilities are linked to truncation and rounding errors, generated by the encoding of the handled numbers into binary data of limited size, inaccuracies of the implemented mathematical basic functions, cancellation effects and overflows or underflows.

The first way to avoid these phenomena is to increase the number of bits used in the conversion of exact real numbers into floating-points. In fact, computing precision is closely linked to the number of bits assigned to store the floating-point significant. In addition, the range of numbers that can be encoded and handled is related to the number of bits assigned to the exponent. Quadruple precision calculations conforming to IEEE 754-2008 will soon be available for software developers and will provide outputs with 34 decimals. Many multi-precision libraries are now also available to engineers or researchers in precision engineering or metrology. However, the increase of the number of digits improves the computing accuracy at the expense of computation time. Therefore, only the routines that perform intensive scientific calculations are generally programmed with multi-precision libraries. But a careful handling of inputs and outputs is required to avoid rounding and truncation errors generated by data conversions between program modules of different types.

The second way to avoid computational errors and algorithm instabilities is to perform scientific calculations with decimal numbers. This is realized in pocket calculators and some supercomputers that have dedicated hardware, but not in laptops or desktops. Software solutions are available based on the BCD coding. However, these solutions currently remain reserved for IT developers who implement their own codes. The quality of scientific calculations is linked to the quality of the software implementation [5]. The adjective “well” or “ill” conditioned refers to the algebraic expression of a given function F . On the other hand, the adjective “stable or unstable” refers to the algorithm and the numerical results associated with a machine. When the algebraic expression is well conditioned, in principle one can always find a stable process to evaluate it. When the algebraic expression is very poorly conditioned, it is difficult to find a stable process to evaluate it. Combining an improperly conditioned algebraic expression with an unstable process is generally a recipe to obtain poor result. In exact arithmetic calculations performed with computer algebra systems (Mathematica, Maple, ...), only rational numbers are implemented, thus limiting the instabilities of algorithms. In floating-point calculations, a numerical certification of results can be realized by using interval or ball arithmetic. In this section, the intrinsic performances of computer hardware and software were only highlighted. The numerical result provided by a metrology or precision engineering software also depends on the quality of the model describing the physical problem and on its implementation. This is the subject of Section 3.

3. Modelling and implementation

The physical problems faced by researchers or engineers in precision engineering or metrology mainly deal with the quantification of measures (scalar quantities) (e.g. parameters of geometrical models) used to describe geometrical features of the measured object, to calibrate machine-tools, to compensate measuring devices, etc. In this section, the properties of the mathematical models used to describe the physical problem are discussed. The choices made in modelling have a significant

impact on the quality of the result. Numerical implementation of the mathematical approach simultaneously requires a suitable definition of the nominal geometric model, the deviations from the nominal features, and the solving method. All this modelling should be realized at the same level of quality. The global performance of the process will in fact be imposed by the software component of lowest quality. On the other hand, appropriate choices may improve the quality of numerical results even if calculations are based on a limited number of digits.

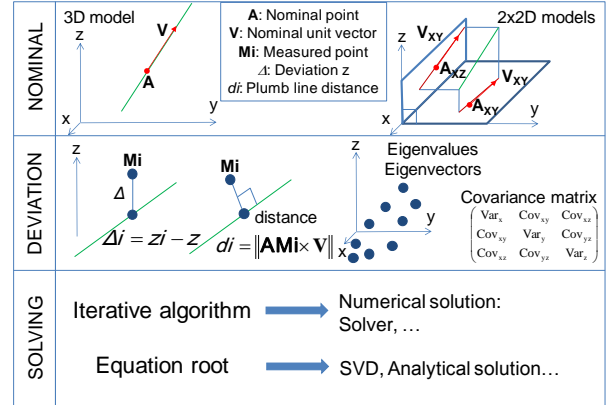


Figure 24: Nominal model, deviations, solving methods implemented to bring to the fore their impacts on the calculation accuracy

To show the effect of modelling, different least squares optimization algorithms were implemented in a spreadsheet application (Microsoft Excel) to approximate a straight line based on a set of acquired coordinates. Reference data provided by the National Metrology Institute of Germany (PTB) was used for that purpose. This reference data set includes 8 points. It is well known, in least squares optimization, that the barycentre of the measured coordinates lies on the approximating line. The problem thus comes down to the determination of the three components of the unit vector defining the line direction. These vector components V_{est} were evaluated by several solving methods and compared to the results V_{PTB} certified by PTB. The error of each calculation process was thus defined by the norm of the difference of the two vectors (Equation 14). All these calculations were performed with 64-bit floating point numbers.

$$Error = \|V_{est} - V_{PTB}\| \text{ with } V_{PTB} \begin{cases} -0.3642697527611 \\ -0.9312929845081 \\ 0.0009613685309 \end{cases} \quad (14)$$

Figure 24 details the different nominal models, the definitions of deviations (i.e. the distances between nominal and actual points) and the solving methods that were implemented. The resulting errors are summarized in Table 4.

The choice of the nominal model, the definition of deviations (distances) or deviation functions and solving methods have a great influence on the quality of the obtained result. In the case of a 3D line, calculating the eigenvectors of the coordinate covariance matrix or its Singular Value Decomposition (SVD) leads to the best precision. These algorithms are also the optimized solving solutions to be used in the case of a plane. Choice 8 (3D nominal model, 3D deviation function, description of the line unit vector by 2 independent angles) also gives results very close to the certified values. But in this case, the solver integrated in the spreadsheet application is applied, working as a black box. This does not allow a fine tuning of the optimization process. The three items: Nominal model, deviation function, and solving method will be further detailed in this section.

Table 4: Comparison of 9 computation processes

Choice	Nominal	Deviation	Resolution	Error
1	2x2D	Pt/line in Y and Z	Solver	2,54E-05
2	2x2D	Pt/line in Y and Z	Analytic	2,15E-06
3	3D	Distance Pt/line	Solver (6 dependant parameters)	1,66E-07
4	2x2D	Pt/line in Y and Z	Solver and reducing para.	1,52E-07
5	2x2D	Distance Pt/lines	Analytic	1,05E-07
6	2x2D	Eigenvalue/vector	Analytic	1,05E-07
7	3D	Distance Pt/Line	Solver (3 dependant parameters)	9,34E-09
8	3D	Distance Pt/Line	Solver (2 independant parameters)	7,46E-11
9	3D	Eigenvalue/vector	Analytic	7,22E-11

3.1 Guidelines to a smart implementation of a nominal model

Different general principles exist to guide researchers or engineers in modelling physical problems. One such basic rule is Occam's Razor (OR), attributed to an English Franciscan friar, William of Ockham (1287–1347). It is also called the Law of Parsimony (LP) and may be formulated in Latin as follows: Pluralitas non est ponenda sine necessitate (entities should not be multiplied unnecessarily) [164]. In science, Occam's razor is used as a heuristic to guide scientists in developing theoretical models [150, 159]. In precision engineering or metrology, this powerful rule leads researchers or engineers to use a limited number of parameters to explain a physical phenomenon. An additional consequence of the application of this principle is that it permits defining the minimum number of parameters required to characterize a model. This allows using variables that are statistically independent and thus simplifies uncertainty evaluation and propagation.

$$\begin{aligned}
 \mathbf{V} &= \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \text{ with } \|\mathbf{V}\| = 1 \Rightarrow 3 \text{ parameters and 1 constraint equation} \\
 \mathbf{VAR}(p_i) &= \begin{pmatrix} \text{var } v_x & \text{cov}(v_x, v_y) & \text{cov}(v_x, v_z) \\ \text{cov}(v_x, v_y) & \text{var } v_y & \text{cov}(v_y, v_z) \\ \text{cov}(v_x, v_z) & \text{cov}(v_y, v_z) & \text{var } v_z \end{pmatrix} \\
 \mathbf{V} &= \begin{pmatrix} \sin \theta \cos \varphi \\ \sin \theta \sin \varphi \\ \cos \theta \end{pmatrix} \Rightarrow 2 \text{ independent parameters} \Rightarrow \mathbf{VAR}(p_i) = \begin{pmatrix} \text{var } \varphi & 0 \\ 0 & \text{var } \theta \end{pmatrix}
 \end{aligned}
 \tag{15}$$

Equation 15 details the parametrisation of the unit direction vector for a straight line, corresponding to choices 7 and 8 of Table 4. Table 4 already highlighted that the choice of two independent parameters (two angles) gives a better estimation of the 3D line direction vector than a modelling by three dependent components. Another aspect to point out is the orthogonality of the coordinate basis that enables the characterization of a deviation. Figure 25 shows the calculation of a distance d , defined by the two components p_1 and p_2 of a vector in a 2D plane. It illustrates the effect of non-orthogonality of the coordinate basis on the description of the same deviation.

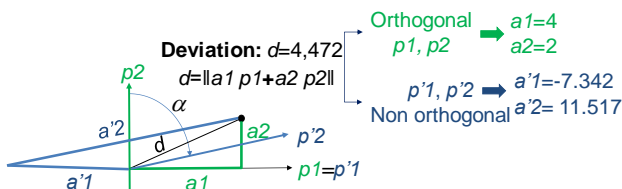


Figure 25: Orthogonal or non orthogonal parameters describing a deviation d

In Figure 25, the data plotted in green are described by an orthogonal coordinate system. It is non-orthogonal for the blue items. When the angle α approaches 90° , the coordinates a_1 and a_2 tend to infinite. In classical model of an aspheric shape [59],

potential numerical instabilities induced by such effect led the authors to propose a new mathematical definition using an orthogonal basis of the parameters.

The geometric characterization of surfaces, parts or products is, generally, based on the measurement of two-point distances, or angles between items (Figure 26). This data is scalar. In the 1970's, the development of CMM's enabled capturing the coordinates of a point in a reference frame. These coordinates are distances acquired in three orthogonal directions. A Cartesian approach of geometry is then used in modelling. Since the structures of real devices are not perfect, geometric models of CMMs or CNC machine structures were substantially improved, giving rise to the currently applied calibration and error compensation methods for three-dimensional measuring or manufacturing systems. The nominal mathematical models of calibration and measurement processes are characterized by a set of parameters (distances, angles and intrinsic parameters). The first type of nominal models aims thus to identify the geometric errors of machine structures and to compensate for these defects afterwards. The goal of the second kind of nominal models is to characterize the geometry of a surface area or an entire measuring object during an inspection process. Some parameters of a nominal model define the position and orientation of a geometric entity with respect to other geometric elements or a reference coordinate system derived from different features.

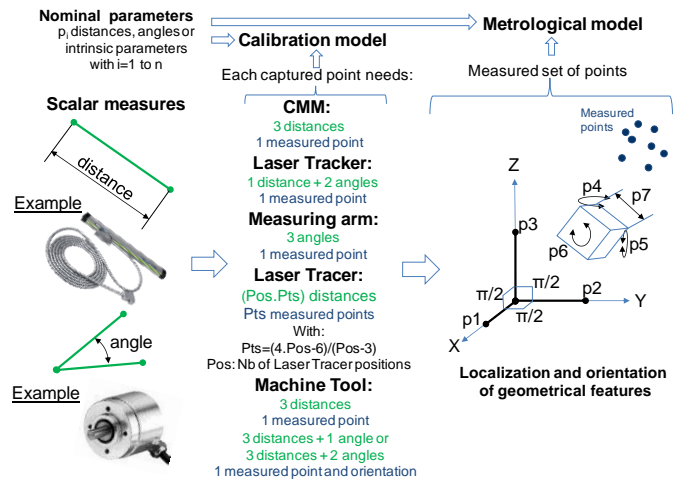


Figure 26: Links between measures and models

Other intrinsic parameters (dimensions, angles, curvatures, etc) define the shape of the geometrical elements. A minimum of 6 parameters is required to locate and orient a geometrical item in a 3D space (3 translations and 3 rotations). The mathematical models that describe the rotation of a geometrical entity are generally based on the Euler matrix transformations, Roll-Pitch-Yaw matrices (or the simplified linear representations: Small Screw Displacement) or Rodrigues' rotations. Euler's angles can describe transformations with large angles, but they degenerate for small rotations. Roll-Pitch-Yaw representations are well adapted to small rotations, but they cause problems for angles close to $\pi/2$. However, these two transformations use the minimum number of parameters (i.e. three) required to define any 3D rotation. Rodrigues' rotation has no angular limitation, but it requires an additional parameter. This representation is not minimal. The nominal geometric models for the calibration and error compensation of CMMs, CNC machine tools or other measuring devices were described in many papers [6,20,27,43, 50,99,155,171] and different CIRP Keynotes [65,156]. These

models will not be discussed here, but the general rules mentioned above can be applied to them, too.

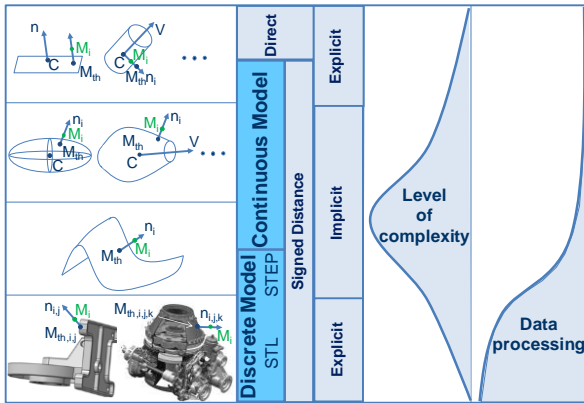


Figure 27: Model typology

The nominal mathematical models, used in metrology to describe surface areas or geometrical features, can be subdivided into two complementary forms: continuous models and discrete models (Figure 27). Continuous models are used in the description of basic surfaces (spheres, cylinders, aspheres, B-spline surfaces, gear flanks...).

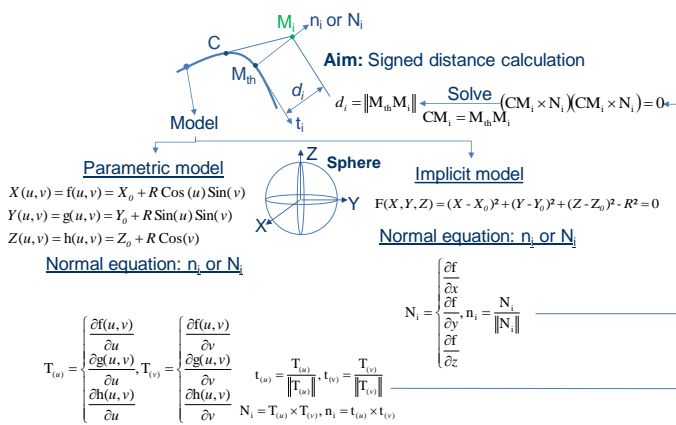


Figure 28: Plumb line distance calculation based on a parametric and implicit model

The mathematical definition for the surface of such a continuous model can be expressed in two ways:

- By an implicit equation: the 3D coordinates (X, Y, Z) of all points on the nominal surface are given by an equation in the form $F(X, Y, Z) = 0$. As an example, the implicit equation of a sphere is presented in Figure 28. This equation defines all those points, whose coordinates X, Y and Z fulfil the equation. They are located on a sphere around the center (X_0, Y_0, Z_0) with the radius R .

- By a parametric equation: the coordinates of all points on the surface are explicitly written as functions of two surface parameters u and v , i.e. $X(u, v), Y(u, v), Z(u, v)$. The parametric equation of a sphere is also shown in Figure 28. The choice between these two models is generally made when choosing the deviation. Generally, it is the nominal model giving the simplest metric equation that will be selected to limit potential numerical discrepancies. The next paragraph, dedicated to deviations, will provide further detail regarding this aspect. When the topology becomes more complex (e.g. a car body door in Figure 29), free form surfaces, free form shaped parts or full 3D masters [129, 130, 154] are split into a set of elementary surfaces that can still be described by implicit or parametric equations (set of planes in

STL files, set of basic surfaces and B-splines in STEP files). Such models are named discrete models. The accuracy of a full 3D master used in metrology is determined by the quality of the process used to translate the CAD model into a data exchange file (STL: CAD models in stereo-lithography or solid freeform fabrication technologies, IGES: Initial Graphics Exchange, ASME Y14.26M [102], VDAFS: Verband der Automobilindustrie-Flächenschnittstelle or "automotive industry association - surface data interface"[149], STEP: Standard for the Exchange of Product model data, ISO 10303 [13]). The native model implemented in a CAD system is the representation used the most in discrete models, since it does not require any translation and thus leads to the best accuracy. The quality of discrete models greatly depends on the conditions of continuity of the elementary surfaces: C0 (point continuity), C1 (slope continuity) and C2 (curvature continuity). The STL format transforms the CAD model into a set of planes, delimited by three triangle vertices and its normal. It does therefore not satisfy the continuum in slope (C1) and curvature (C2). The IGES format describes a volumetric geometric element by a set of parametric tiles or basic surfaces. The STEP neutral file presents the latest technological advances in the volume description of complex or simple features. Complex surfaces are described by a set of B-Splines. The degree of these parametric surfaces can guarantee the geometrical continuity in C0, C1 and C2. However, geometric discontinuities can still be observed with such a surface exchange format, depending on the quality of the translation module.

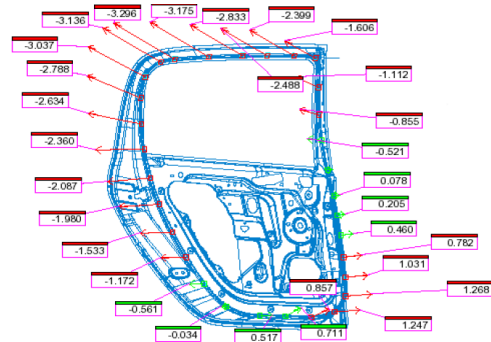


Figure 29: Full 3D master of a car body door, given in blue, and measured point deviations

3.2 Deviation calculation

ISO 17450-part 1 [15,104] defines the basic operations available to verify a dimensional or geometric specification: partition, extraction, filtration, collection, association and construction. The surface model is defined in the partition operation. Section 2 summarized some precautions to take in order to obtain a smart modelling of the studied metrological problem. In the association operation, a deviation quantity is required to approximate the measured coordinates to the nominal model.

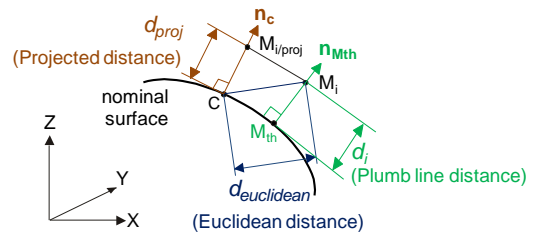


Figure 30: Distance definition

In order to handle all deviations in the same way, i.e. to give all measured points the same weight in approximation routines, a

general function valid for all captured points is required called the deviation function. Depending on the inspection task and geometric restrictions, three types of deviation functions are commonly applied: the Euclidean distance (\mathbf{CM}_i norm), the projected distance ($\mathbf{CM}_{i/proj}$ norm), and the plumb line distance ($\mathbf{M}_i\mathbf{M}_{th}$ norm). The two last distances are Signed Distances (SD), calculated between a measured point \mathbf{M}_i and the nominal geometric element (Figures 28 and 30). This nominal geometry can either be represented by a nominal point \mathbf{M}_{th} or \mathbf{C} corresponding to \mathbf{M}_i and the unit normal vector \mathbf{n}_i on the nominal surface in the environment of \mathbf{M}_i [70], or by the implicit or parametric nominal surface (or a real sub-patch of it). The latter gives the plumb line distance, defined by the smallest-possible distance magnitude between \mathbf{M}_i and the nominal surface. This distance vector crosses the nominal surface perpendicularly, i.e. its direction is given by $\text{grad } F(X,Y,Z)$ in implicit model definition [67] or partial derivatives in parametric model (Figure 28). Its distance value is defined positive when the point \mathbf{M}_i is located outside the material or when the dot product between the vector $\mathbf{M}_i\mathbf{M}_{th}$ and the normal vector \mathbf{n}_i on the modelled surface or curve is positive, equal to zero when the measured point lies on the surface or curve (dot product equal to zero) and negative otherwise (negative dot product). By convention, the normal vector on a surface is oriented to the outside of the material.

Using one of the 3 deviation function types defined before, other deviation functions (or measures of distance) can be defined, but they must satisfy some conditions. For example, in an inspection process for assessing the form or orientation of a geometric feature, compliant with the ISO 1101 standard, the deviation used to realize the association are no longer directly given by signed distances, but either by the difference between its maximum and minimum deviation or two times the maximum distance (minimum zone criterion) [139].

The law of conservation is a suitable guide for the choice of the deviation quantity. Several types of conservation principle are known in engineering science: conservation of mass, and conservation of energy, for example. In physics, a conservation law states that a measurable property of a system remains constant while the system's state might change. This definition can be easily applied to the field of precision engineering and metrology. For the three types of deviation functions explained before (see also Figures 27,28 and 30), it is obvious that signed distances and eigenvalues/vectors are independent of any change in the reference frame, whereas the unidirectional distance Δ depends on the selected direction of computation. Signed distances and eigenvalues respect the law of conservation and define dimensional quantities that allow the location of the 3D line in the space, independent of the reference system.

If the deviation function is based on a distance (Figures 27 and 30), its computing leads to three basic configurations: point-to-point distances or Euclidean distances (calibration of Machine-tools or Coordinate Measuring Machines, Iterative Closed Point (ICP) algorithms, etc), point-to-curve distances (approximation of circles or lines in metrology, toolpath optimization in manufacturing, etc) and point-to-surface distances (approximation of basic or complex surfaces in metrology, control of geometrical specifications with full 3D masters, etc). The latter two configurations could be based on projected distances or plumb line distances (Figure 30).

In many cases, the topology of the geometric element explicitly provides a sense for the normal vector. This is the case for the standard geometric elements circle, line, plane, cylinder, sphere, cone and torus. Discrete models using STL format or meshed surfaces can be added to this class. The signed distance is obtained using point-point, point-line and point-plane distance formulae. These cases are in the "Explicit" boxes of Figure 27. For

this first class of standard geometric elements the calculation of deviations does not present any difficulty. In the case of other continuous models (paraboloid, ellipsoid, sphere...) and discrete models using parametric surfaces (B-spline, Bezier, Coons...), the calculation of the deviation becomes much more difficult, because then it is necessary to define the minimum distance between each measured point \mathbf{M}_i and the approximating surface. This is generally achieved by the determination of plumb line distance. If the normal vector in the environment of \mathbf{M}_i is known, the projected distance is a reasonable estimation for the plumb line distance. Analytically, the plumb line distance is given by the orthogonal projection \mathbf{M}_{th} of point \mathbf{M}_i onto the surface. Figure 28 summarizes the computational process that enables the determination of this projection and therefore the computing of the signed distance. These two subclasses are merged in the "Implicit" box of Figure 27.

$$(\mathbf{CM}_i \times \mathbf{N}_i) = \mathbf{0} \Leftrightarrow (\mathbf{CM}_i \times \mathbf{N}_i)(\mathbf{CM}_i \times \mathbf{N}_i) = 0 \quad (16)$$

Equation 16 shows an implicit expression that can be used to determine the orthogonal projection \mathbf{M}_{th} in Figure 28. The degree of this equation depends on the model used. This degree can quickly increase which requires numerical iterative methods such as Gauss-Newton or Levenberg-Marquardt algorithms for its solution [138]. However, it must be pointed out that Equation 16 may lead to more than one solution if the normal vector line \mathbf{N}_i or \mathbf{n}_i intersects multiple surface points. To obtain the plumb line distance, it is then necessary to select the result with the smallest distance.


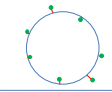
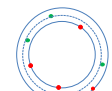
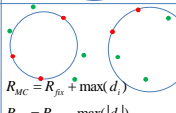
Best-fit criterion	Norm	PDF	Equation	Example
Least square	L2-Norm or Gaussian Norm	Prob($\hat{x}=d$) 	$\sum_{i=1}^k d_i^2$ k : SD number	
Minimum zone	T-Norm		$\max(d_i) - \min(d_i)$	
Minimum circumscribed or Maximum inscribed	T-Norm		$\max(d_i)$ $R_{MC} = R_{ex} + \max(d_i)$ $R_{MI} = R_{in} - \max(d_i)$	

Figure 31: Objective functions or approximation criterions

In the case of least squares approximation there is no need for some geometric elements such as planes or lines to compute distances between the measured coordinates and the theoretical nominal item. In fact, it can be shown that the barycentre of the measured points lies on the approximating element. Moreover, the vector which completes the characterization of the approximated feature (normal to the plane, direction vector of the line) can be deduced from the covariance matrix of the measured coordinates and corresponds to the eigenvector of lowest eigenvalue. It is this method, using an SVD factorisation, that obtained the best result in the test carried out in Figure 24. This algorithm thus avoids the iterations of the classical methods and therefore prevents successive rounding and cancellation errors of the computer. This case is labelled "Direct" in Figure 27. Figure 27 highlights two major difficulties encountered in computing the deviations: the level of complexity in the calculation of the distances that greatly increases when implicit equations are to be solved, and the volume of data to be processed. For explicit models, no real numerical difficulty exists. However, for 3D full masters, the number of deviations to be

processed increases, which leads to the management of large data files (big data).

3.3 Solution methods

In previous sub-sections, the nominal model and its parameters were chosen. The types of deviation functions and the distances between the measured points and the model were also defined. The signs and magnitudes of all distances depend on their relative position and orientation with respect to the position and orientation of the approximating geometric element (which is to be determined) and on the geometric features characterizing the element like cylinder radius or cone angle (which are also to be determined). This means that starting from a given cloud of captured measuring points, there is an infinite number of possible approaches for the parameters of the geometric element that represent, to a greater or lesser extent, a “good approximation”. To find the best-possible approximation, a criterion is required to differentiate a “good” from a “better” solution of the approximation problem. These criteria are called objective functions. Several types of objective functions are used in production metrology (see Section 3.3.1), primarily determined by the inspection task and the definition of tolerances. Mathematically, an objective function is a functional, i.e. a mapping from a vector space (more specifically: a space of functions) into the space of real numbers. This objective function assigns each possible approximation solution for the geometric element to one corresponding scalar value. In other words: the objective function creates a “ranking list” among the possible solutions, where the best-possible approximation can be determined unambiguously by the minimum scalar value of the functional [67,68,69].

3.3.1 Types of objective functions

The following will explain the types of objective functions, predominantly applied in production metrology. The selection of the approximation criterion is related to the solving method (numerical or analytical solving, iterative computation or root calculation) to be used to find the optimal parameters of the model. Since metrology and precision engineering software are mainly implemented with floating point numbers, this subsection will focus on the numerical behaviour of the criterion or the optimization method. The approximation methods used the most in metrology and precision engineering are least squares optimization, minimum zone evaluation, and calculation of minimum circumscribed or maximum inscribed feature. They are all approximated according to an objective function, which is a norm of the deviations d_i between the measured points and the geometrical element to be determined. This norm is called Lp-norm, written as

$$L_p - \text{Norm} = \left[\sum_{i=1}^k d_i^p \right]^{1/p} \quad \text{with } p = 1 \text{ to } \infty \quad (17)$$

where p is the degree of the norm, ranging between 1 and infinity. And k is the number of measured points.

Two special cases of this norm are mainly used as approximation criterion (Figure 31): the L2-Norm with $p = 2$ [60,157] that leads to least squares optimization (LSQ), and the infinite norm (L_∞ -Norm), where p tends to infinite, also called Tschebyscheff-Norm (T-Norm) [9,69, 71,158,178]. As shown in [71], calculations of minimum circumscribed or maximum inscribed features can also be realized by approximations according to the T-norm. The Probability Density Function (PDF) associated with each approximation criterion is shown in Figure 31 [10]. LSQ corresponds to the maximum likelihood estimation for Gaussian noise. The L1-Norm with $p = 1$ may also be used in specific cases.

An infinite norm would require the calculation of the functional in Equation 17 for a degree p tending to infinity, but that cannot be achieved numerically. The case of $p = \infty$ corresponds to a minimax problem, minimising the maximum residual distance. It can be implemented in a comparably simple way by selecting for p a value between 50 and 100 which provides a good estimation of the T-norm. Higher degree values (e.g. $p=300$ to 500) can improve this estimation, but require more decimal digits and thus more calculation time [71]. [67] and [70] suggest an upper and lower bound for the T-norm, both based on the Lp-norm with a finite p .

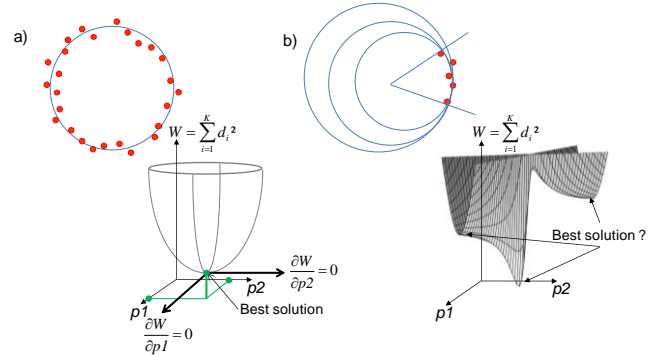


Figure 32: Numerical effect of an incomplete set of points: a) points distributed over a large angular range b) points distributed over a comparable small angular range [31]

From the point of view of computational convenience, a desired property of an approximation criterion is to provide an objective function with only one single minimum. Situations of non-uniqueness of the minimum are however reported by [158] in case of approximations according to the T-Norm (form inspection or maximum-inscribed criterion). In fact, it is possible to construct examples of approximating a plane according to the T-norm that have a number of local solutions that is almost the same as the number of data points. As shown Figure 32a, the L2-norm (least squares optimization) should present, theoretically, only one single minimum. This is the case, in practice, when the acquired coordinates are distributed over a large angular range of closed geometric features (circles, cylinders, cones, spheres, etc) or on a wide lateral extent of the measured surface. When the extent of the measured points is reduced, numerical problems are amplified due to the presence of local minima. These local minima are added by successive rounding and cancellation errors of floating point operations (see Figure 32b). It can lead to a poor parameter estimation of the approximating element. In the case of a circle, the origin of these numerical problems is the cancellation of high degree terms of the polynomial approximation used by the computer to calculate square roots [31]. This phenomenon of digital degeneracy can be observed for any type of surface. This shows the importance of the choice of the initial parameters (also called starting solution) required for an iterative numerical process. In the case of existing local minima, the algorithm will converge to the nearest local minimum and therefore not necessarily “find” the global optimum parameters. The use of floating point numbers in computer codes leads to this phenomenon. Using multi-precision libraries or computer algebra systems (Mathematica, Maple, ...) will limit the cancellation effects. Factors such as the choice of the optimization criterion (objective function), the distribution of the points measured on the geometric element, the mathematical model, etc, will have an impact on the success of a computation method applied to find the approximating parameters. In following subsection, the behaviour of the computation methods will be studied.

3.3.2 Calculation methods

In metrology or precision engineering software, two types of computation methods are used:

- Numerical and/or iterative computation methods,
- Symbolic computation methods.

The first one is strongly influenced by the use of floating point numbers and its limitations, whereas the second one, in theory, is not influenced by them.

3.3.2.1 Numerical and/or Iterative computation

The mathematical problems met in the field of precision engineering or metrology generally correspond to the optimization of objective functions with specific characteristics. Figure 33 details these characteristics, i.e.: what is the number of estimated parameters; is the objective function linear, quadratic or nonlinear, with or without constraints? Are specialized mathematical methods or algorithms used? The objective function may be deterministic or stochastic and may or may not require the calculation of derivatives.

As written in Section 2.2.4, the precision to which a numerically stable algorithm can solve an ill-conditioned problem is limited by the accuracy of the data. However, a numerically unstable algorithm can produce bad solutions even for well-conditioned problems. This means that an unstable algorithm can yield solutions that are less precise than theoretically achievable from the given data [11,66,162].

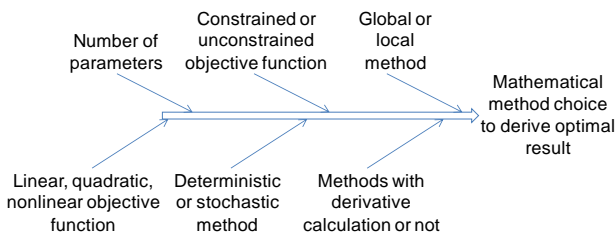


Figure 33: Characteristics of mathematical problem

To avoid these problems, numerical computation must respect three basic rules:

- the inverse problem used in parameter model approximation must be well conditioned,
- the applied algorithms must be numerically stable in order to achieve results with a given finite arithmetic precision,
- the software requires a careful implementation of the algorithms.

The stability of the optimization method with respect to rounding-off errors is a fundamental characteristic to obtain accurate numerical results.

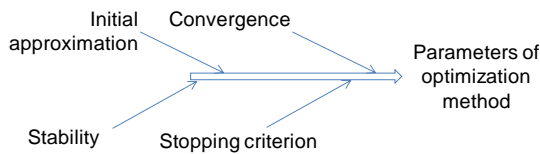


Figure 34: Parameters of optimization method

Figure 34 summarizes the parameters which influence the precision of the obtained result: stability, initial approximation, convergence, and stopping criterion. Newton's method is the basis for many optimization routines or root search programs (Figure 35). Optimization algorithms, generally, require computing derivatives of the first (Gradient or Jacobian) and often second order of the function (Hessian). The properties of the different mathematical methods (advantages and inconveniences) are summarized in Figure 35. The solving of the equations used in least-squares methods, can be performed by

specific calculations (Cholesky, QR factorisation [73]). Using the QR factorisation of the Jacobian matrix, for example, is more numerically stable than finding the Cholesky factorisation of the product of the transposed Jacobian matrix with itself. The condition number of the product is the square of the condition number of the Jacobian matrix; there will also be a loss of precision simply by forming the product. The use of a singular value decomposition of the Jacobian matrix is also numerically stable [73].

	Steepest descent methods (SDM)	Newton's method (NM)	Quasi Newton's method (QNM)
	Optimal, fixed or variable steps Conjugate gradients methods (linear prob.)	Newton-Raphson method	BFGS method L-BFGS method
	[12,57,58,85]	[17-18]	[49,126,136,144]
Adv	Conjugate gradient method is simpler to code and requires less storage space	If the cost function is quadratic, the global minimum is reached in 1 iteration Newton method's convergence is quadratic	Approximation of the Hessian which is locally re-estimated at each iteration L-BFGS method is able to handle large memory problems
Inv	Local method Successive determination of search directions and step lengths Preconditioning required	Hessian inverse is computed at each iteration With many parameters, calculation are long and expensive in storage Risk of divergence	Less information about cost function form BFGS is the best Quasi-Newton methods

	Least squares method (LSM)	Intelligence oriented algo.
	Linear: Moore-Penrose Non linear: Gauss-Newton Levenberg-Marquardt (LM)	Genetic algorithms Swarm colony optimization Bees algorithm Particle swarm
	[49,124,134,136,138,161]	[97,173,179,180]
Adv	Gauss-Newton (GN). Normal equations are used Levenberg-Marquardt Far from the solution, it reacts like a SDM, close as GN. More stable than GN	Simple algorithm Good flexibility Search for minimum or maximum overall facility for functions with mini or max local
Inv	For some very regular functions LM can converge slightly slower than GN	Method does not guarantee the true extreme discovery

Figure 35: Properties of mathematical methods

Quasi-Newton methods attempt to build an approximation of the Hessian matrix (or its inverse) that incorporates second order information by incorporating first order information as the optimisation proceeds. The Broyden-Fletcher-Goldfarb-Shanno Algorithm (BFGS) [49,126,136,144] is one of the most famous quasi-Newton algorithms for unconstrained optimization. Moving away from deterministic algorithms, intelligence-oriented algorithms (Genetic algorithms, Swarm algorithms) [97,173,179,180] with their simplicity are another way to search the solution of extreme problems with many local minima. An optimization toolbox has been implemented in Matlab software for solving complex optimization problems. It automatically selects the most efficient algorithm for the computed mathematical problem. Matlab uses several algorithms depending on the type of problem to be solved: interior reflective Newton

method, trust-region-dogleg, trust-region-reflective, Levenberg-Marquardt, simplex, BFGS, MiniMax, and so on.

3.3.3.2 Symbolic computation

In Section 3.3.3.1, numerical methods were investigated performing calculations related to problems of precision engineering and metrology. As an alternative, symbolic calculation is offered today to researchers and engineers [39].

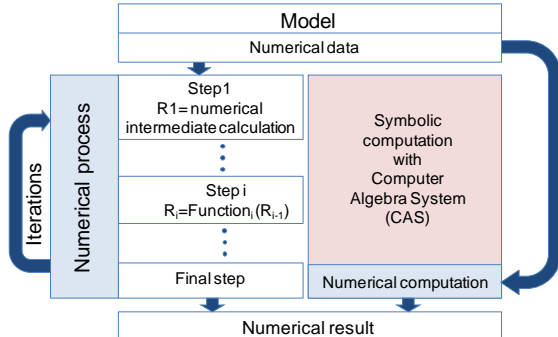


Figure 36: Numerical or symbolic computation

Figure 36 illustrates the difference between these two manners of handling a computational problem. In numerical methods, all the data is handled in a numerical form, usually as floating-point numbers, extending from the beginning of the computational process to its end. Rounding and cancellation errors may thus be generated at any intermediate calculation. Careful implementation of each step of the computational process is therefore required to obtain a correct result. Symbolic calculation, on the contrary, gives a formal solution of a mathematical problem. The numerical application is therefore performed at the end of the calculation process, which reduces the rounding and cancellation effects that arise with floating-point calculations. To avoid numerical degeneracy, floating-point numbers are not permitted in symbolic calculus. Decimal numbers are thus treated as rationals (ratio of two integers).

Polynome degree	Methods	Computation
1rd	Classical method	Symbolic
2th	Classical method	Symbolic
3th	Girolamo Cardano method	Symbolic
4th	Lodovico de Ferrari method	Symbolic
Abel Theorem & Galois Theorem		
5th	Sturm's theorems	Numerical
6th	Sturm's theorems	Numerical
...	Sturm's theorems	Numerical

Figure 37: Root research of univariate polynomial

Symbolic calculation is based on exact calculations and equations including parameters or numbers in arbitrary precision. Unfortunately, all mathematical problems cannot be processed in symbolic computation. The differentiation or integration of functions, the manipulation of polynomials, vectors or matrices (linear equations) are treated in symbolic calculation. The resolution of polynomial systems and systems of nonlinear multivariate equations [22,23,24,54] are also available in symbolic computation. Formal calculation solutions are offered by commercial computer algebra systems (Maple, Mathematica, ...) and open source software (GAP, Maxima, SAGE, ...).

The calculation of the deviations (distances) between the nominal model and the measured points often requires finding the roots of a polynomial. For example, in the case of a paraboloid, a 5th degree polynomial equation has to be solved. For an ellipsoid, the equation is of degree 6. Similarly, for the approximation of a plane using the SVD method, it is necessary to

determine the eigenvalues. They result from the roots of a 3rd order characteristic polynomial. The computation of roots of univariate polynomials is thus one core problem to be solved in metrology and precision engineering. Any non-constant real polynomial can be factored as a product of irreducible real polynomials of degrees 1 or 2.

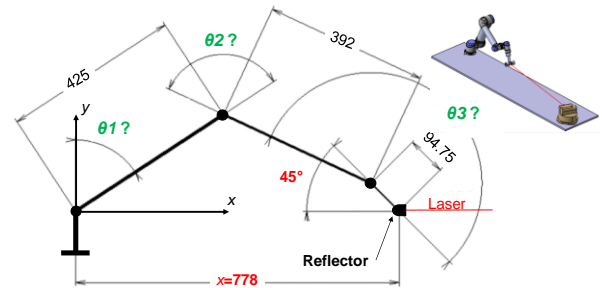


Figure 38: Robot calibration

This theorem does not provide any explicit decomposition algorithm. It only predicts what the final form of the result should be. Consequently, this raises the problem to (i) ascertain the existence of real roots and (ii), if they exist, to evaluate them with a certified precision. The direct method to prove the existence of roots is to formally exhibit them (when possible). One can find an explicit formula - using radicals - for each root of a polynomial of 1st, 2nd, 3rd and 4th degree. But from the 5th degree on, there is an insurmountable difficulty. The work of N.H. Abel and E. Galois [63] has highlighted, in fact, that the roots of polynomials of a degree greater than 4 cannot, in the general case, be expressed with radicals (Figure 37). It is therefore impossible to obtain an explicit formula determining the roots and, consequently, it is necessary to implement numerical methods with all of their well-known weaknesses. Sturm's sequence or Sturm's theorems can however be applied to this problem to define the number of roots existing in a given real range [160]. Dichotomic search algorithms can thus be used to find the roots with the desired precision.

Notations:

$\sin(\theta_i)=s_i, \cos(\theta_i)=c_i$ with $i=1$ to 3

Maple symbolic calculations:

with(Groebner);

P[1]:=-392*c2*s1-(9475/100)*c2*c3*s1-392*c1*s2-(9475/100)*c1*c3*s2-(9475/100)*c1*c2*s3+(9475/100)*s2*s3*s1-425*s1-778;

P[2]:=-(9475/100)*c1*s2*s3-(9475/100)*c3*s1*s2-(9475/100)*c2*s1*s3+392*c2*c1+(9475/100)*c2*c3*c1+425*c1-392*s1*s2;

P[3]:=c1*c2*c3-c1*c2*s3-c1*c3*s2-c1*s2*s3-c2*c3*s1-c2*s1*s3-c3*s1*s2+s1*s2*s3;

P[4]:=-c2*c3*s1-c1*c3*s2+c3*s1*s2-c1*c2*s3+c2*s1*s3+c1*s2*s3-c1*c2*c3+s1*s2*s3-sqrt(2);

P[5]:=c1^2+s1^2-1;

P[6]:=c2^2+s2^2-1;

P[7]:=c3^2+s3^2-1;

Eqs:= [P[1], P[2], P[3], P[4], P[5], P[6], P[7]];

Base_Eq:= Basis(Eqs, plex(c1, c2, c3, s1, s2, s3));

Groebner polynomial basis:

EQ1:-54258895713987415992*sqrt(2)+388332180977704960809+922584152329361887232*s3^2+(874147898398981466624*sqrt(2)-9930712645324111872)*s3

EQ2:-6060151255*sqrt(2)+1575385983048+(1662712247616*sqrt(2)+6097273168)*s3+894619342400*s2

EQ3:-493215819*sqrt(2)+4305672952+(-64351168*sqrt(2)-528392704)*s3+4564384400*s1

EQ4:-2024917716*sqrt(2)+15578795+(-264196352*sqrt(2)-2169337856)*s3+2104986688*c3

EQ5:-4479561+1179448*sqrt(2)+5331200*c2

EQ6:-493215819*sqrt(2)-4049835432+(-4274324544*sqrt(2)-528392704)*s3+4564384400*c1

Figure 39: Symbolic calculation for the example shown in Figure 38

The solving of linear equation systems is another classical problem treated by computer algebra software. Symbolic calculus can also treat nonlinear physical problems that can be modelled as a system of multivariate polynomial equations. This is achieved by using a Groebner basis. Groebner bases can be seen as the generalization of Gaussian elimination algorithm to nonlinear and multivariate polynomial systems.

The Groebner bases were introduced in 1965 by B. Buchberger [22,23,24], who proposed a calculation algorithm in his dissertation. He gave it the name of his thesis supervisor: W. Groebner. To illustrate the power of Groebner bases, the approach was applied to inverse kinematic calculations of a three-axis robot. Figure 38 shows an example of such an application with numerical values. The aim of this study is to calibrate a robot using a tracking laser. In the experiment, the optical reflector of the tracking laser must always keep a fixed orientation (45°), while the robot moves in a straight trajectory along the x axis. For each measured x-position (x=778mm in Figure 38), the joint angles ($\theta_1, \theta_2, \theta_3$) of the robot are to be calculated. The closures of the geometric and angular loops of the robot lead to multivariate polynomial equations of the sines and cosines of the three joint angles. These equations are completed by the quadratic relation combining the sine and the cosine of the same angle. A multivariate polynomial system (P[1], P[2], P[3], P[4], P[5], P[6], P[7]) is thus obtained [25]. The set of equations are summarised in Figure 39. Groebner based symbolic calculus was applied to this multivariate polynomial system using Maple software. In the best case, the results are provided in the form of a triangular system of univariate polynomials. For x=778, the computation gives 6 equations EQ1, EQ2, EQ3, EQ4, EQ5, EQ6. These equations are shown at the bottom of Figure40. The first equation EQ1 is a 2nd degree univariate polynomial equation that allows the computation of $\sin\theta_3$. Its solving gives two solutions corresponding to two possible configurations of the robot. The calculation of $\sin\theta_3$, step by step, leads to the solutions of the other variables. Table 5 shows one of the two geometric configurations of the robot that simultaneously reaches the position x=778 and respects the orientation angle of the reflector (45°).

Table 5: Set of computed robot angles

	(rd)	(°)
θ_1	0.9916	56.8183
θ_2	2.1263	121.8286
θ_3	3.4908	200.0103

A. Clement summarized the Groebner bases properties in five practical pieces of information:

- Roots of $P[i]$ with $i = 1$ to n of variable v_i are identical to the roots of basis (P[i]), i.e. $P[i]$ and basis (P[i]) show the same affine variety.

- If 1 is an element of basis (P[i]), the equation $1 = 0$ results in a contradiction and, leads to the conclusion that the polynomial system has no roots.

- $P[i]$ is called (and therefore also basis (P[i])) zero-dimensional if a finite number of roots exists. This property can be read directly, if each variable is a pure power of a dominant coefficient of one of the base polynomials (P[i]).

- Let a new constraint be represented by the polynomial $P[n+1]$. This new polynomial is redundant with respect to the system P, if and only if 1 belongs to the base for the new variable. This means that the satisfaction of $P[n+1]$ is inevitable if the system $P[i]$ for $i=1$ to n is satisfied, so that $P[n+1]$ can be eliminated.

- Basis (P[i]) based on the lexicographer order $v_1 > v_2 > v_3 > \dots > v_n$ is a triangular system in the sense that some polynomials contain only the variable x_1 , others only x_2 , ... so that the numerical

resolution is analogous to the triangular elimination of a linear system. Theoretically, Groebner based calculation method is the most efficient solver of polynomial equation systems, since it provides not only absolutely all algebraic roots of a multivariate polynomial system, but also the invariants of the associated manifold. The least "efficient" solver is then Newton's algorithm, since it, eventually, provides a unique solution. The usual numerical method for proving the existence of a root by calculating it numerically is insufficient for three reasons: (i) the solver used can provide a neighbouring solution, but different from the one sought (unstable algorithm), (ii) the algorithm can provide a non-existent solution and (iii) the precision of the result, with few exceptions, -cannot be certified.

The advantages and disadvantages of the Groebner bases can be summarized as follows:

- Groebner bases provide absolutely all roots in the form of a set of polynomials in "row echelon form". Each polynomial only depends on one independent variable and is of minimum degree. This leads to fast and "certifiable" numerical solving.
- However, there is a serious technical disadvantage: multivariate polynomial equation systems that can be treated on actual desktops or laptops are limited to 12 variables. Large computing times are also often necessary to obtain the results. Since the method provides absolutely all the roots, the problem is then to eliminate uninteresting solutions as soon as possible during the treatment to speed up the procedure.
- Groebner basis calculations work with complex numbers and therefore do not distinguish between real or complex roots. Sturm's theorem must therefore be applied to each polynomial basis to verify the existence of real roots.

4. Software validation and traceability

A significant, and sometimes dominant, contribution to measurement uncertainty arises from these numerical calculations. Unfortunately, the influence of evaluation software is often underestimated or simply ignored. One of the reasons for this is that neither developers nor end-users have access to validated test data (traceable soft gauges) for validating metrological software [79].

4.1 Computer aims/Standard

A helpful and detailed vocabulary of information technology including terms relating to software testing is given in ISO/IEC 2382:2015 [109]. There are many standards that give common and generic recommendations on the development, design, maintenance and validation of software. For example, general information is provided in ISO 9001 [108], ISO 20000-1 [105], and ISO 27001:05 [107]. However, concrete instructions for testing metrological algorithms are still very rare. Specific examples are mentioned in the following. ISO 10360-6 [103] gives instructions that allow the evaluation of least squares approximation algorithms in the field of coordinate metrology. ISO 5436-2 [106] describe sets of test data used in roughness metrology.

4.2 Testing methodologies

There are two basic approaches to test the correctness of software, (i) structural or white box testing, in which the source code is reviewed against its design (Figure 40), and (ii) functional or black box testing (Figure 41), in which the test data is supplied to the software and the results output by the software is compared against the expected outputs.

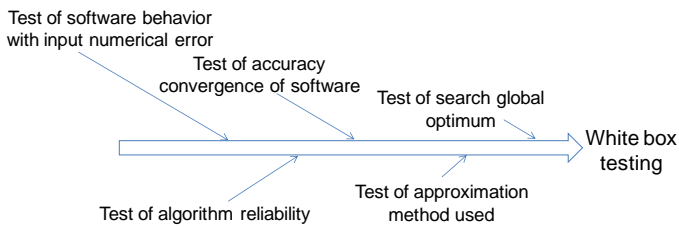


Figure 40: White box tests

In the case of black box testing, there is no need to have access to the source code, the software is regarded as a “black box”, that produces outputs for given inputs. While white box testing is required to find errors in coding (“bugs”), there are many reasons why correctly coded numerical software will not give perfectly accurate answers. First, the computations are performed in finite precision and some rounding errors will accrue, as discussed in Section 2.2. One of the early (and still current) challenges of numerical analysis is to understand how these errors build up. Next is to manage how to design algorithms for which the build-up is controlled such that error bounds can be established for the computed solutions [87,176]. Second, nonlinear computational tasks require iterative algorithms, for which convergence tolerances need to be set (Section 3.3). Determining these tolerances in such a way that they cope with different scalings (units) associated with the data and parameters can be difficult. It is possible that for some tasks the software terminates prematurely before an accurate solution has been found. Third, for nonlinear optimization tasks, the algorithm may converge accurately to a local minimum, but fails to find the global minimum (Section 3.3.1). Approximation of geometric elements according to the Chebyshev criteria can be prone to this type of behaviour. Fourth, the software might implement an approximation algorithm that determines for some sets of data a sufficiently accurate solution, but an inaccurate one for other data sets. Fifth, the user might apply the software incorrectly, for example, by assigning the input parameters incorrectly. For these and other reasons, well-engineered software could provide outputs that are not sufficiently accurate for the user’s requirements. These issues can be addressed by black box testing.

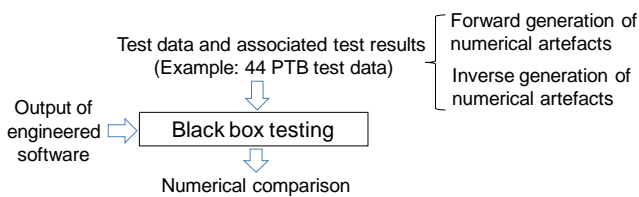


Figure 41: Black box principle

4.3 Generation of test data

Black box testing (Figure 41) requires the provision of test data and associated test results [117]. The value of a black box test using such test data depends on the extent to which the test input and output data are aligned with the intended computational aim of the software. It is assumed that the computational aim of the software can be expressed as a mathematical function $\mathbf{a}=f(\mathbf{x})$ relating the input data \mathbf{x} to the output data \mathbf{a} through a known, deterministic function $f:\mathfrak{R}^m \rightarrow \mathfrak{R}^n$ (It is not assumed that f can be described in a closed form, only that for any \mathbf{x} there is a unique \mathbf{a}). A numerical artefact for the computational aim f is a finite precision pair $\langle \mathbf{x}, \mathbf{a} \rangle$ such that nominally $\mathbf{a}=f(\mathbf{x})$.

4.3.1 Forward generation of numerical artefacts

One method of generating numerical artefacts is to assign input data \mathbf{x} and then apply a reference software S to evaluate \mathbf{a} .

Reference software is characterized by a high degree of confidence: it addresses the true computational intent, uses numerically stable algorithms, is implemented in software developed using recognized quality assurance methodologies, and is subjected to a high level of testing. Further confidence in the computed results can be gained by comparing independent implementations of the algorithms (analogous to inter-laboratory comparisons called round robin tests, common practice in metrology). The test results can be validated by comparing different independent implementations [80]. Additional assurance can be gained by implementing such software in extended precision with the expectation that any of the unavoidable numerical rounding errors are small enough to be negligible, compared to the outputs calculated with standard precision rounding.

4.3.2 Inverse generation of numerical artefacts

A second method for generating test data is to start with the “solution” \mathbf{a} and then generate data \mathbf{x} such that $\mathbf{a}=f(\mathbf{x})$ [26,61]. In practice, finding such \mathbf{x} can be a much easier problem than determining the solution \mathbf{a} for a given \mathbf{x} . For example, for nonlinear least square problems, the generation of input data \mathbf{x} for a given solution involves determining the null space of the associated Jacobian matrix, a standard problem in numerical linear algebra, which can be solved using the QR factorization [73]. Thus, the forward generation of data requires the solution of a nonlinear problem, while the inverse generation involves only the solution of a linear problem. Similarly, it is possible to generate data for Chebyshev approximation problems using of solutions in terms of points, at which the maximum error is attained [61, 98]. Again, the inverse data generation can be performed in extended precision to provide very high confidence in the accuracy of the numerical artefacts.

4.3.3 Numerical accuracy of numerical artefacts and numerical standards

A numerical artefact can be assumed as the digital equivalent of a physical artefact, used to check the performance of an instrument: the numerical artefact is used to check the performance of software addressing a computational aim – a “digital instrument”. For a numerical artefact $\langle \mathbf{x}, \mathbf{a} \rangle$, the very fact that $\langle \mathbf{x}, \mathbf{a} \rangle$ is represented in finite precision means that only under very special circumstances the relationship $\mathbf{a}=f(\mathbf{x})$ will hold exactly. A numerical standard is a numerical artefact, for which a quantitative measure (accuracy bounds, uncertainty) is known stating how far $\langle \mathbf{x}, \mathbf{a} \rangle$ is apart from a pair $\langle \mathbf{x}^*, \mathbf{a}^* \rangle$, for which $\mathbf{a}^*=f(\mathbf{x}^*)$ is given mathematically. Once the uncertainty associated with a numerical standard has been assessed, the numerical standard can be used to assess the accuracy of a digital instrument since it is possible to distinguish (in principle) the uncertainty contribution of the digital instrument from that associated with the standard.

$[\mathbf{x}]$ also denotes those elements of \mathfrak{R}^m that are rounded to \mathbf{x} in the implemented finite precision arithmetic, and $[\mathbf{a}]$ etc. is defined, similarly. The best that can be expected is that there exist a \mathbf{x}^* in $[\mathbf{x}]$ and an \mathbf{a}^* in $[\mathbf{a}]$ such that $\mathbf{a}^*=f(\mathbf{x}^*)$. This situation arises in generating numerical artefacts using extended precision. Assuming that the extended precision artefacts are sufficiently accurate, the standard precision artefacts will represent the finite precision representation of a mathematically exact $\langle \mathbf{x}^*, \mathbf{a}^* \rangle$. More generally, there will be a $V \subset \mathfrak{R}^m$ and $W \subset \mathfrak{R}^m$ such that $\mathbf{x}, \mathbf{x}^* \in V$ and $\mathbf{a}, \mathbf{a}^* \in W$ and $\mathbf{a}^*=f(\mathbf{x}^*)$ exactly. The diameters V and W specify the numerical accuracy bounds associated with the numerical artefact. The diameters of V and W (or similar measures) provide a statement of the accuracy of the numerical artefact. It is possible to evaluate V and W with $V=\{\mathbf{x}\}$. In this case, the diameter

of W is a measure of how far \mathbf{a} is apart from $f(\mathbf{x})$, the mathematically exact solution for the input \mathbf{x} , which represents an assessment of the *forward accuracy* of the numerical artefact. Alternatively, for V and $V=\{\mathbf{x}\}$, the diameter of V is a measure of how much we the input \mathbf{x} has to be modified in order to find an \mathbf{x}^* such that $\mathbf{a}=f(\mathbf{x}^*)$, that is, a set of input data for which the stated \mathbf{a} is the mathematically exact solution. Thus, the diameter of V in this case is a measure of the inverse or *backward accuracy* of the numerical artefact. The concepts of forward and backward accuracy here are directly related to those discussed in Section 2.2.3.3. The term backward accuracy reflects the practice in numerical analysis of assessing the accuracy of the computed solution in terms of perturbations of the input data – backward error analysis [26,61]. For metrology applications, the input \mathbf{x} usually represents measurement data, and the inverse accuracy of a numerical artefact can be compared directly with the probable measurement uncertainty associated with the data.

4.3.4 Required accuracy of numerical standards in dimensional metrology

If it is assumed that the best practice in dimensional metrology operates at a numerical accuracy of one part in 10^n , then a minimal requirement is that software should be accurate in one part in 10^{n+1} . Consequently, in order to assess such software, the numerical standards should be accurate in one part in 10^{n+2} or better. For IEEE double precision arithmetic, numerical standards will not be more accurate than one part in 10^6 but since n is in the region of 7 for dimensional metrology, providing numerical standard accurate to 1 part in 10^9 or better is easily achievable.

4.4 Performance metrics

Numerical standards can be used to assess the performance of software, the digital instrument. In a metrology context, the uncertainty contribution associated with the software needs to be assessed, along with all other uncertainty contributions. Performance metrics are a way of assessing the uncertainty contribution of software relative to a set of criteria. In general, there are two types of performance metrics: (i) those metrics assessing the accuracy of the computed results relative to the best possible accuracy given the conditioning of the numerical problem (Section 2.2.3.3), and (ii) those metrics assessing the accuracy relative to pre-assigned tolerances derived from user requirements.

4.4.1 Performance metrics relating to numerical accuracy

Suppose a numerical standard $\langle \mathbf{x}, \mathbf{a} \rangle$ has accuracy bounds V and W so that there is an $\mathbf{x}^* \in V$ and $\mathbf{a}^* \in W$ such that $\mathbf{a}^* = f(\mathbf{x}^*)$. Thus, \mathbf{a}^* must lie in the image $f(V) = \{f(\mathbf{y}) : \mathbf{y} \in V\}$. The diameter of $f(V)$ depends on the sensitivity of the solution to perturbations in the input data. It is also known that the distance of \mathbf{a}^* from \mathbf{a} is specified by the diameter of W , such that the distance of \mathbf{a}^* from \mathbf{a} is bounded by a combination of the diameter of $f(V)$ and the diameter of W , reflecting both the accuracy of the numerical standard and the sensitivity or conditioning of the computational problem. The combined bound represents the best possible accuracy that can be expected of a computed solution.

One approach, based on the GUM [76,77] to derive a performance metric base for this type of analysis is as follows. For numerical bounds V and W , let V_V and V_W be the variance matrices associated with the uniform distributions defined on V and W , respectively. Here, \mathbf{x} can be regarded as a best estimate of \mathbf{x}^* with associated variance matrix V_V and \mathbf{a} as best estimate of \mathbf{a}^* with associated variance matrix V_W . If f is a sufficiently smooth function of \mathbf{x} , the sensitivity matrix C of \mathbf{a} can be calculated with respect to \mathbf{x} .

Then

$$\begin{aligned} f(\mathbf{x}) &= f(\mathbf{x}^* + (\mathbf{x} - \mathbf{x}^*)) \approx f(\mathbf{x}^*) + C(\mathbf{x} - \mathbf{x}^*) \\ &= \mathbf{a}^* + C(\mathbf{x} - \mathbf{x}^*) \\ &= \mathbf{a} + (\mathbf{a} - \mathbf{a}^*) + C(\mathbf{x} - \mathbf{x}^*). \end{aligned}$$

The exact solution for \mathbf{x} is perturbed from \mathbf{a} by $(\mathbf{a} - \mathbf{a}^*) + C(\mathbf{x} - \mathbf{x}^*)$, where the first term has an associated variance matrix V_W and the second term has a variance matrix $CV_V C^T$. The best estimate of $f(\mathbf{x})$ is \mathbf{a} , and the variance matrix associated with this estimate is $V_{\mathbf{a}} = V_W + CV_V C^T$. A computed solution can be assessed based on its distance from \mathbf{a} , relative to the probabilistic distance derived from $V_{\mathbf{a}}$. A similar approach can be used to derive performance metrics relating to inverse numerical accuracy.

4.4.2 Performance metrics relating to user requirements

While it is preferable to have software that provides maximum achievable accuracy, in practice such software may be difficult to implement and/or computationally expensive to run. Under these circumstances, software calculating an approximate solution is developed instead, and it is necessary to assess whether the computed solution is sufficiently accurate for the user's requirements. A pragmatic approach is to say that a software is suitable for a given purpose if the uncertainty contribution associated with the software is small compared to other uncertainty contributions. Thus, in coordinate metrology involving measurement uncertainties of the order of 1 micrometer, an uncertainty contribution from software in the order of 10 nanometres will have no practical impact.

The user requirements may be specified in terms of the accuracy (or maximum permissible error) of the computed solution, for example, the accuracy in the computed diameter of a cylindrical shaft. Such a specification does not take into account the numerical sensitivity associated with the computational aim. If the data on the cylinder lies on a small arc of the cylinder surface, the diameter of the approximated cylinder is poorly determined and the computed solution will be unavoidably less accurate than for data distributed more uniformly on the cylinder surface.

Alternatively, the user requirements may be specified in terms of an equivalent measurement uncertainty derived using an inverse measure of accuracy: the computed solution must be the exact solution corresponding to a perturbation of the input data, where the perturbation is smaller than some pre-assigned tolerance. For the example of a cylinder approximation, the user may require that the computed diameter is exact as long as to the data differ from the input data by no more than 10 nanometres. These inverse types of user requirements automatically take into account the sensitivity of the computational task.

4.5 National Metrology Laboratories works

One of the first areas of metrology to become aware of the potentially large influence of numerical software was that of coordinate metrology. The diversity of evaluation algorithms and their different implementations led to inconsistent results when evaluating prismatic 2D and 3D objects consisting of lines, planes, cylinders, circles and cones. As a consequence, the National Physical Laboratory (NPL, UK) and the Physikalisch-Technische-Bundesanstalt (PTB, Germany) cooperated in some European projects to provide test data [47,77] and reference algorithms [62,9].

As a result of these and other initiatives, PTB provides a commercial offline software test for prismatic objects since 1995. To date it has been used by more than 200 companies for validating their evaluation software systems. In 2012, the design of this offline test became basis of the online test TraCIM (see below).

The National Institute of Standards and Technology (NIST, USA) provides algorithms testing the least-squares approximation of elementary geometries used in coordinate metrology. Specifically, the geometries for which testing is available include lines (2d and 3d data), planes, circles (2d and 3d data), spheres, cylinders, and cones. Testing is modelled after the ASME B89.4.10-2000 standard, Methods for Performance Evaluation of Coordinate Measuring System Software.

The NIST Algorithm Testing System computes with a precision much greater than the double precision normally applied in scientific computing. In fact, 60 digits of working precision are used to compute the reference fits. Because of this, the uncertainty of the reference least-squares approximations is limited only by the accuracy of the input data. For ASME B89.4.10 default test data sets, which are theoretically exact (i.e., data values are assumed to have infinite trailing zeros) the expanded uncertainty U ($k = 2$) is much less than 10^{-14} m for distances and 10^{-15} radians for angles.

NPL, NIST and PTB also provide test data and reference software in the field of roughness measurement according to ISO 5436-2 (2001) [101]. Since 2014, an established realization of the online validation is named Traceability for Computationally-Intensive Metrology (TraCIM) [81]. It allows service users to validate their software at the point of application. The service is operated by European national metrology institutes (NMIs). It is a fundamental principle that the TraCIM service is provided and hosted only by a NMI or an authorized organization (Figure 43). These institutions assume *delcredere* liability and finally guarantee for the correctness of the results.

TraCIM is registered as a word mark. It is operated as a legal non-profit association under German law and allows NMIs and designated institutes (DIs) to become members. The TraCIM association has been established with its main mission to provide quality rules for the TraCIM service [165]. The business concept and the income of the TraCIM service are strictly uncoupled from the TraCIM association. In association with and under the supervision of European metrology institutes, TraCIM aims to validate analysis algorithms in the field of metrology. In the following, they will be referred to as "algorithm tests" or simply "tests". Similar to the well-known calibration chain, which is related to physical standards, the NMIs transfer the numerical accuracy of evaluation algorithms from the highest metrological authority to the individual application. Computations are addressed, which are used to analyze measurands of the International System of Units (SI) and their derived units. The medium of choice for communication between the service provider and the user is the Internet. The principle is shown in Figure 42. On the left, the service provider is represented as the network of metrology institutes. This is of paramount importance, since the algorithm tests are to be carried out – or at least monitored – by the supreme metrological authority of a country. The metrology institutes are linked with each other under the umbrella of the TraCIM association. TraCIM's main task consists of describing quality guidelines and defining the technical infrastructure, under which the algorithm tests are to be performed. Each service provider is, however, solely responsible – and therefore held liable – for the extent of the algorithm tests provided, for the business workflow, for the maintenance of the datasets, for consultation upon installation as well as for running the tests. For this reason, each metrology institute runs its own server, i.e. each server has to be addressed individually, which leads to a different extent of services depending on each metrology institute. The metrology institutes, however, have the possibility of mutually providing algorithm tests as subcontractors, which allows a service provider to enhance the extent of services provided. The service users are essentially manufacturers of analysis software or measuring instruments.

This algorithm test service allows them to have their analysis algorithms validated by an independent metrology institute. This mainly serves to increase confidence in the products they offer on the market. In principle, they can have this service unlocked for their customers in order to have, for example, updates validated directly. Software engineers can already test their algorithms during the development phase to be on the safe side and, thus, make development faster. Each registered user, which supports the specifications of the client-server interface, can access individual tests via the internet. The service is available 24 hours a day on every day of the year at each location on the globe to which internet is provided.

Furthermore, the response times are considerably shorter than with the existing validation capabilities. A full test dataset includes test data as input quantities of an algorithm test, reference results and their assigned numerical uncertainties. Hereby, the test data are defined as being error-free. In contrast, the indication of the reference results, by analogy with the indication of measurement results, is defective in the case of geometrical measurements [115]. Hereby, it is up to the metrology institutes to develop procedures as long as these meet the requirements. The following is a description of the approach followed by PTB. To this end, all test datasets are computed, tested and archived in a database when setting up an algorithm.

This database – also called "golden dataset" – thus contains all sensitive test data and must therefore be protected from unauthorized access. Contrary to a reference software, for which test data can be specified externally, PTB's approach, consisting in the one-time computation of test datasets, is far more secure, since the correctness of a software component can never be guaranteed.

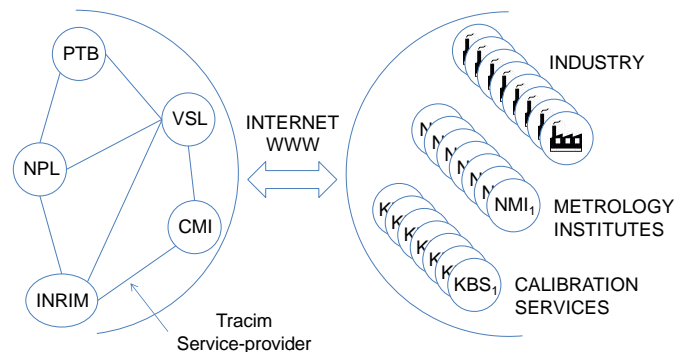


Figure 42: TraCIM service providing to industry and scientific institutes

In addition, specified datasets are practically not subject to ageing. Reference algorithms, in contrast, depend on the state of the art of the programming language, of the operating system, of the processor properties and, thus, need maintenance and are short-lived. Yet, the test data do not represent an inflexible system. They can be adapted to an individual application without losing their accuracy. This can mean, for example, that an SI unit or a derived unit is indicated. In the case of geometrical measurands, the test data can be represented with additional SI prefixes such as "nano-", "micro-", "milli-" etc. This does not affect the numerical presentation – and, thus, the accuracy. The same applies to the scaling of measuring ranges, which may only be realized in the form of decimal powers (i.e. $\times 10$; $\times 100$, etc.). The error bars of the reference results are yielded by means of comparative computations. Thus, the reference results of at least three independent software implementations are computed and compared with each other. The numerical accuracy is determined by varying the test data by means of a Monte Carlo simulation. For this purpose, the last decimal digits of the test data are randomly varied, and the dispersion of the corresponding

reference results is determined. After ignoring another decimal for safety's sake, this value is deemed the assigned numerical uncertainty. The presentation of the test datasets is adapted to the technical applications in question – and not to what is mathematically feasible. The test datasets are supposed to simulate frequent technical situations. Exceptions, which require a high degree of development and consultation effort should, as a rule, be avoided. TraCIM's IT architecture consists of four central modules. These are represented in Figure 43. The server is the core module. As a management module, it is operated by a competent metrology institute. It manages all of the operating data and controls the data flow to the other modules. The expert modules are developed by experts responsible for a particular individual test. Each expert module operates basically autonomously and deals with all logical processes in connection with a test. It makes the test datasets available on request, compares the test results computed by the users with its own reference results and, finally, issues the test report. Since the individual tests may vary significantly from one application to another, only few input parameters have been defined by TraCIM for the data traffic. This applies, for instance, to the support of a software interface in JAVA, which allows the expert system to be logged into the server system. Indispensable operating data such as the order number must also be transmitted via this interface. Since the formats of the test data can be freely selected, the expert is, to a large extent, free to design the test according to his needs. Furthermore, existing tests and test data structures can easily be integrated into the TraCIM system.

other tests, comparison of measurement data [82] and Chebyshev approximation [98], are offered as well. During the last years, many users from all over the world gained certificates for their metrological algorithms.

5. Conclusion

Traceability of intensive computation is needed due to the evolution of industrial systems towards a cybernetic industry. In metrology and precision engineering fields, the geometry of parts or mechanical assemblies to be checked becomes more complex in terms of topology and quantity of data to be processed.

In the near future, current processors will be replaced by new technologies such as quantum, DNA or optical processors. These advances on the hardware should make it possible to question the binary coding massively used in the computation of the current computers. However, the implementation of these new technologies in office computers will not be available immediately. Consequently, the use of floating point numbers, imposed by current hardware technology, will require precautions in the development of metrology and precision engineering software.

Software tools and methods are available to validate and trace the numerical results of metrology or precision engineering software. With actual computer technology, a multiple precision module or software and a calculation using decimal floating-point allow compensating the numerical degeneracy, which limits the numerical calculation accuracy. Standards have included, in their texts, the computation with quadruple precision and a great number of recommendations to reduce the numerical disturbances present within the current technology of computers. However, the software solutions offered today are not up-to-date in including these new recommendations.

When an algebraic expression of metrology or precision problems is well conditioned, one can always find a stable process to evaluate it. When the algebraic expression is very poorly conditioned, it is difficult to find a stable process to evaluate it. Combining an improperly conditioned algebraic expression with an unstable process, the obtained result will be poor. Over the past decade, the National Metrology Institutes have developed numerous methods to validate test data, like traceable soft gauges for validating metrological software. Symbolic computation is another way to realize high-precision calculation and to obtain certified solutions. To estimate the behaviour of the algorithms, research work on arithmetic or ball intervals provides computer solutions that determine the interval of numerical error. Probabilistic approaches are developed to certify the numerical calculation. In the future, new sets of reference data for software development and new hardware paradigms, e.g. survey implementation in line, are the best ways to check the traceability of computer calculations in metrology and precision engineering software. To help the software developers to validate their software at the point of use, online solution was proposed by National Metrology Institutes.

By managing in the implementation phase, the stability and conditioning of the computation, metrology or precision engineering software should be suitable to perform high-precision calculations. This is the new challenge for a new cyber industry.

Acknowledgments

The authors gratefully acknowledge contributions or assistance by: Alexandro Balsamo (INRIM), Edward Morse (UNCC), Klaus Wendt (PTB), Albert Weckenmann (QFM), Xiangqian Jane Jiang (Univ. Huddersfield), Harald Bosse (PTB), Jean-François Rameau (Dassault Systèmes), Jesse Groover (UNCC). We also would like to give thanks to Tyler Estler (NIST) for his valuable comments.

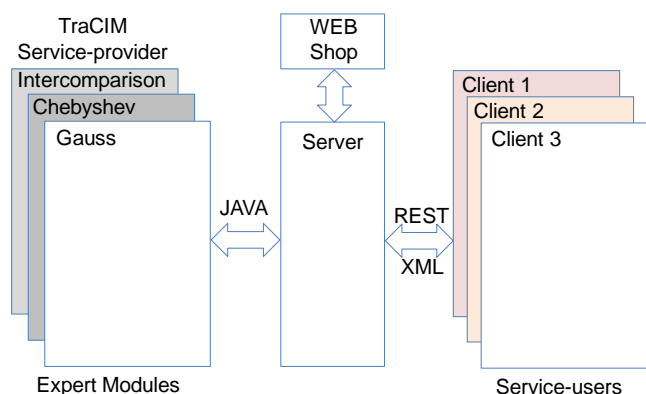


Figure 43: TraCIM's IT architecture

The formal specifications of the TraCIM server with regard to the user are, in contrast, more restrictive. The server-client communication runs via a REST interface. Hereby, the data are embedded into an XML structure. Then again, within this structure, free formats of test data (such as binary formats or established test data structures) can be defined, depending on the application. The expert is solely responsible for the test data format, the test data and the test results. The interface is available externally via a web shop, which is also connected to the server module. At the current implementation stage of the system, this interface is, however, not yet available. Similar to online shopping, interested users will be able to register via the internet and to order individual tests [166]. The precondition for running the test is, however, that the REST interface is supported. Until the web shop has been set up, interested users can contact an individual service provider or the TraCIM secretariat in order to register. In the current state of implementation, PTB offers the so-called "Gaussian test". This test is used in the field of length measurement. It is used to check the correct determination of the parameters for the adjustment of geometrical elements like 2D straight line, 3D straight line, 3D circle, 3D plane, cylinder, cone and sphere, according to the least-square approximation. Two

References

- [1] Arden W, Brillouët M, Coge P, Graef M, Huizing B, Mahnkopf R (2010) More-than-Moore, White paper, International Technology Roadmap for Semiconductors.
- [2] Adleman, L M (1994) Molecular computation of solutions to combinatorial problems. *Science*, 266(5187), 1021-1024.
- [3] Adleman, L M (1998) Computing with DNA. *Scientific American*, 279(2):54 - 61.
- [4] Amdahl G (1967) Validity of the single processor approach to achieving large scale computing capabilities. In: *Proceedings of the April 18-20, 1967 spring joint computer conference*. ACM, New York, 483-485.
- [5] Boldo S, Fillâtre J C, Melquiond G (2009) Combining Coq and Gappa for certifying floating-point programs. In *International Conference on Intelligent Computer Mathematics*. Springer Berlin Heidelberg.
- [6] Aguado S, Samper D, Santolaria J, Aguilar J J (2012) Identification strategy of error parameter in volumetric error compensation of machine tool based on laser tracker measurements *International Journal of Machine Tools & Manufacture* 53:00:00 160-169.
- [7] Anderson E, Bai Z, Bischof C, Blackford S, Demmel, J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Sorensen D (1999) *LAPACK Users' Guide* (Third ed.). Philadelphia, PA: Society for Industrial and Applied Mathematics.
- [8] Anderson M J, Tsen C, Wang L K, Compton K, Schulte M J (2009) Performance analysis of decimal floating-point libraries and its impact on decimal hardware and software solutions. *Computer Design, ICCD 2009 IEEE International Conference on* 465 - 471.
- [9] Anthony G T, Anthony H M, Bittner B, Butler B P, Cox M G, Drieschner R, Elligsen R, Forbes A B, Gross H, Hannaby S A, Harris P M, Kok J (1996) Reference software for finding Chebyshev best-fit geometric elements, *Precision Engineering* 19(1):28-36.
- [10] Aranda S, Linares J M, Sprauel J M (2010) Best-fit criterion within the context of likelihood maximization estimation. *Measurement*, 43(4), 538-548.
- [11] Arioli M, Demmel J W, Du J S (1989) Solving Sparse Linear Systems with Sparse Backward Error. *Journal on Matrix Analysis and Applications*, 10(2):165-190.
- [12] Arioli M, Gratton S (2012) Linear regression models, least-squares problems, normal equations, and stopping criteria for the conjugate gradient method, *Computer Physics Communications* 183:2322-2336.
- [13] ASME Y14.26M Digital Representation for Communication of Product Definition Data.
- [14] Bachmann J, Linares JM, Sprauel JM, Bourdet P, (2004) Aide in decision-making: contribution to uncertainties in three-dimensional measurement. *Precision Engineering* 28(1):78-8.
- [15] Ballu A, Mathieu L, Dantan J (2015) Formal Language for GeoSpelling. *Journal of Computing and Information Science in Engineering*, 15(2):021002-021002-6.
- [16] Beaman J, Morse E, (2010) Experimental evaluation of software estimates of task specific measurement uncertainty for CMMs. *Precision Engineering* 34(1):28-33.
- [17] Bertsekas D P (1999). *Nonlinear programming*. Belmont: Athena scientific.
- [18] Björck Å (1996) *Numerical methods for least squares problems*. Society for Industrial and Applied Mathematics.
- [19] Boldo S, Muller J M (2014) Les ordinateurs capables de calculer plus juste. *La Recherche* 492:46-52.
- [20] Bringmann B, Knapp W (2006) Model-based 'Chase-the-ball' calibration of a 5-axes machining center, *CIRP Annals-Manufacturing Technology*, 55(1):531-534.
- [21] Brisebarre N, Louvet N, Martin-Dorel E, Muller J M, Panhaleux A, Ercegovac M D (2010) Implementing decimal floating-point arithmetic through binary: some suggestions. In *ASAP 2010-21st IEEE International Conference on Application-specific Systems, Architectures and Processors*, 317-320.
- [22] Buchberger B (1965). *An Algorithm for Finding the Basis Elements of the Residue Class Ring of a Zero Dimensional Polynomial Ideal*. Ph.D. dissertation, University of Innsbruck. English translation by Michael Abramson in *Journal of Symbolic Computation* 41 (2006): 471-511.
- [23] Buchberger B (1986) Gröbner bases. *An Algorithmic Method in the Theory of Polynomial Ideals*. Computer algebra. Symbolic and algebraic computations. Mir, Moscow.
- [24] Buchberger B, Winkler F (1998) *Gröbner bases and applications* (Vol. 251). Cambridge University Press.
- [25] Buchberger B, Kauers M (2010) *Scholarpedia*, 5(10):7763. (doi:10.4249/scholarpedia.7763).
- [26] Butler B P, Cox M G, Forbes A B, Hannaby S A, Harris P M (1997), A methodology for testing the numerical correctness of approximation and optimisation software, in *The Quality of Numerical Software: Assessment and Enhancement*, Boisvert, R, ed., Chapman and Hall.
- [27] Caja J, Gómez E, Maresca P (2015) Optical measuring equipments. Part I: Calibration model and uncertainty estimation, *Precision Engineering* 40:00:00 298-304.
- [28] Chaitin-Chatelin F, Fraysse V. (1996) *Lectures on finite precision computations*. Siam.
- [29] Chen Z, Dongarra J, Luszczek P, Roche K (2003) Self-adapting software for numerical linear algebra and LAPACK for clusters, *Parallel Computing* 29:1723-1743.
- [30] Chen D, Han L, Choi Y, Ko S B (2012) Improved Decimal Floating-Point Logarithmic Converter Based on Selection by Rounding. *IEEE Transactions on Computers* 61(5):607-621.
- [31] Chernov N, Lesort C, (2005) Least Squares Fitting of Circles. *Journal of Mathematical Imaging and Vision*, 23(3):239-252.
- [32] Cline A K, Moler C B, Stewart G W, Wilkinson J H (1979) An Estimate for the Condition Number of a Matrix *SIAM Journal on Numerical Analysis*, 16(2):368-375.
- [33] Colonna J-F (1996) Kepler, von Neumann and God (More rounding-off error visualizations). *The Visual Computer* 12(7):346-349.
- [34] Cornea M (2009) IEEE 754-2008 Decimal Floating-Point for Intel, ARITH, Computer Arithmetic, IEEE Symposium on, Computer Arithmetic 225-228.
- [35] Cornea M, Anderson C, Harrison J, Tak Peter Tang P, Schneider E, Gvozdev E (2009) A software implementation of the IEEE 754R decimal floating-point arithmetic using the binary encoding format, *IEEE Transactions on Computers* 58(2):148-162.
- [36] Cowlshaw M F (2003) Decimal floating-point: algorithm for computers. In *Computer Arithmetic, Proceedings 16th IEEE Symposium on IEEE 104-111*.
- [37] Cowlshaw, M. F. (2003) Decimal floating-point: algorithm for computers. *Computer Arithmetic, 16th IEEE Symposium on IEEE 104-111*.
- [38] Daramy-Loirat C, Defour D, X de Dinechin F, Gallet M, Gast N, Muller J M (2005) CR-LIBM A library of correctly rounded elementary functions in double-precision on Application-specific Systems Architectures and Processors (ASAP).
- [39] Davenport J H, Siret Y, Tournier É (1988) *Computer algebra* (Vol. 5). London: Academic Press.
- [40] De Dinechin F, Lauter C, Muller JM (2007) Fast and correctly rounded logarithms in double-precision. *Rairo-Theoretical Informatics and Applications* 41(1):85-102
- [41] Deng Y, Zhang P, Marques C, Powell R, Zhang L (2013) Analysis of Linpack and power efficiencies of the world's TOP500 supercomputers, *Parallel Computing* 39(6-7):271-279.
- [42] DiCarlo L, Chow J M, Gambetta J M, Bishop L S, Johnson B R, Schuster D I, Majer J, Blais A, Frunzio L, Girvin S M, Schoelkopf R (2009) Demonstration of two-qubit algorithms with a superconducting quantum processor. *Nature* 460 240-244.
- [43] Dong L, Jindong T (2013) An accurate calibration method for a camera with telecentric lenses, *Optics and Lasers in Engineering* 51:00:00 538-541.
- [44] Dongarra J, Luszczek P, Petit A (2003) The LINPACK Benchmark: past, present and future. *Concurrency and Computation: Practice and Experience* 15(9):803-820.
- [45] Dongarra J J, Meuer H W, Strohmaier E (1997) TOP500 supercomputer sites, *Supercomputer* 13(1): 89-120.
- [46] Dongarra J, Faverge M., Ltaief H, Luszczek P (2014) Achieving numerical accuracy and high performance using recursive tile LU factorization with partial pivoting. *Concurrency and Computation: Practice and Experience* 26:1408-1431.
- [47] Drieschner R, Bittner B, Elligsen R (1991) *Testing Coordinate Measuring Machine Algorithms, Phase 2*. EC.
- [48] Edmonson W W, Van Emden M H (2008) Interval Semantics for Standard Floating-Point Arithmetic. *arXiv preprint arXiv:0810.4196*.
- [49] El-Hayek N, Noura H, Anwer N, Gibaru O, Damak M (2014) A new method for aspherical surface fitting with large-volume datasets. *Precision Engineering* 38:935-947.
- [50] Erkan T, Mayer J R R (2010) A cluster analysis applied to volumetric errors of five-axis machine tools obtained by probing an uncalibrated artefact, *CIRP Annals - Manufacturing Technology*, 59(1): 539-542.
- [51] Erle M A, Schulte M J, Hickmann B J (2007) Decimal floating-point multiplication via carry-save addition. *Proceedings of the 18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pages 46-55. IEEE Computer Society Conference Publishing Services.
- [52] Estler W T (1999) Measurement as Inference: Fundamental Ideas, *CIRP Annals - Manufacturing Technology* 48(2):611-631.
- [53] Farmahini-Farahani A, Tsen C, Compton K (2009) FPGA implementation of a 64-Bit BID-based decimal floating-point adder/subtractor. *International Conference on Field-Programmable Technology* 518 - 521.
- [54] Faugere J C (1999) A new efficient algorithm for computing Gröbner bases (F 4). *Journal of pure and applied algebra*, 139(1): 61-88.
- [55] Feng W C, Cameron K W (2007) The Green500 List: Encouraging sustainable supercomputing. *Computer* 40(12): 50-+.
- [56] Feynman R P (1982) *Simulating Physics with Computers*. *International Journal of Theoretical Physics*, 21(6/7): 467-488.
- [57] Fletcher R, Powell M J D (1963) A rapidly convergent descent method for minimization, *Computer Journal*, 6 :163-168.
- [58] Fletcher R, Reeves C M (1964) Function minimization by conjugate gradients. *Computer Journal*, 7 :149-154.
- [59] Forbes G W, Brophy C P (2008) Asphere, O Asphere, how shall we describe thee? In *Optical Systems Design* (pp. 710002-710002). International Society for Optics and Photonics.
- [60] Forbes A B (1989) Least-squares best-fit geometric elements. *NPL, DITC* 140/89.
- [61] Forbes A B and Minh H D (2012), Generation of numerical artefacts for geometric form and tolerance assessment, *Int. J. Metrol. Qual. Eng.*, 145-150.
- [62] Forbes A B (1990), Least Squares Best Fit Geometric Elements, *Algorithms for Approximation II*, Mason J C and Cox M G eds., London, Chapman & Hall, 311-319.
- [63] Galois E (1846). Sur les conditions de résolubilité des équations par radicaux. *Journal de mathématiques pures et appliquées*, 11 : 417-444.
- [64] GAO/IMTEC-92-26 Patriot Missile Software Problem, 1992.
- [65] Gebhardt M, Mayr J, Furrer N, Widmer T, Weikert S, Knapp W (2014) High precision grey-box model for compensation of thermal errors on five-axis machines, *CIRP Annals - Manufacturing Technology*, Volume 63(1):509-512.
- [66] Gilli M (2006). *Méthodes numériques*. Département d'économétrie, Université de Genève.
- [67] Goch G (1982) *Theorie der Prüfung gekrümmter Werkstück-Oberflächen in der Koordinatenmeßtechnik*, Helmut-Schmidt-University Hamburg-Germany (formerly University of German Forces), Dissertation
- [68] Goch G, Haupt, M (1990) Modifizierte Tschebyscheff-Approximation von Kreisen, *technische Rundschau* 41/90:50-53.

- [69] Goch G, Renker H J (1991) Efficient Multi-Purpose Algorithm for Approximation and Alignment Problems in Coordinate Measurement Techniques, CIRP Annals - Manufacturing Technology, 39(1):553-556.
- [70] Goch G, Tschudi, U, Pettavel, J (1992) A Universal Algorithm for the Alignment of Sculptured Surfaces, CIRP Annals - Manufacturing Technology, 41(1):597-600.
- [71] Goch G, Lübke K (2008) Tschebyscheff approximation for the calculation of maximum inscribed/minimum circumscribed geometry elements and form deviations, CIRP Annals - Manufacturing Technology 57(1):517-520.
- [72] Goldberg D (1991) What every computer scientist should know about floating-point arithmetic. Computing Surveys 23(1): 5-48.
- [73] Golub G H, Van Loan C F (1996), Matrix Computations, Johns Hopkins University Press, Baltimore.
- [74] Gonzalez-Navarro S, Tsen C, Schulte M J (2013) Binary Integer Decimal-Based Floating-Point Multiplication. IEEE Transactions on Computers 62(7):1460-1466.
- [75] Graillat S, Lefevre V, Muller JM (2015) On the maximum relative error when computing integer powers by iterated multiplications in floating-point arithmetic. Numerical Algorithms 70(3): 653-667.
- [76] GUM BIPM, IEC, ISO, IUPAC, IUPAP, OIML; "Guide to the expression of the uncertainty in measurement, First Edition". 1993, ISBN 92-6710188-9.
- [77] GUMS1: BIPM, IFCC, & IUPAP (2008). Evaluation of measurement data—Supplement 1 to the GUM: propagation of distributions using a Monte Carlo method. Technical report, International Organization for Standardization (ISO), Geneva.
- [78] Gustafson J L (1988) Reevaluating Amdahl's Law, Communications of the ACM, 1(5):532-533.
- [79] Härtig F Certificate for Involute Gear Evaluation Software American Gear Manufacturers Association, Fall Technical Meeting, 2006
- [80] Härtig F, Müller B, Wendt K, Franke M, Forbes A, Smith I (2015) Online validation of metrological software using the TraCIM system, XX IMEKO World Congress "Measurement in Research and Industry".
- [81] Härtig F, Müller B, Gahrens M, Franke F, Delpy H, Forbes A, Smith I (2014) Online validation of numerical algorithms in the field of metrology, 11th Laser Metrology for Precision Measurement and Inspection in Industry 2014
- [82] Härtig F, Tang J, Hutzschenreuter D, Wendt K, Kniel K, Shi Z Online (2015) Validation of Comparison Algorithms using the TraCIM-System, Journal of Mechanical Engineering and Automation, 2(7): 312-327
- [83] Henning J L (2000) SPEC CPU2000: measuring CPU performance in the New Millennium. Computer 33(7):28 - 35.
- [84] Hermann M, Pentek T, Otto B (2016) Design Principles for Industrie 4.0 Scenarios, HICSS, 49th Hawaii International Conference on System Sciences (HICSS), 3928-3937, doi:10.1109/HICSS.2016.488.
- [85] Hestenes M R, Stiefel E (1952) Methods of Conjugate Gradients for Solving Linear Systems", Journal of Research of the National Bureau of Standards. 49 (6):409-436.
- [86] Hickey T, Ju Q, Van Emden M H (2001) Interval arithmetic: From principles to implementation. Journal of the ACM (JACM), 48(5): 1038-1068.
- [87] Higham N J (2002) Accuracy and numerical stability of algorithms, SIAM, Philadelphia.
- [88] Hill M D, Marty, M R (2008). Amdahl's Law in the Multicore Era. Computer 41(7):33-38.
- [89] <http://wccftch.com/intel-10nm-cannonlake-ice-lake-tiger-lake-cpu/>
- [90] <http://www.advanpix.com/>
- [91] <http://www.green500.org/>
- [92] <http://www.top500.org>
- [93] <http://www.lactamme.polytechnique.fr/Mosaic/descripteurs/FloatingPointNumbers.01.Ang.html>
- [94] <http://www.lactamme.polytechnique.fr/Mosaic/descripteurs/OrdinateursEtCalculs.01.Ang.html>
- [95] <https://www.youtube.com/watch?v=kH8gehlirRE>
- [96] Huang T, Zhu Y, Qiu M, Yin, Wang X (2013) Extending Amdahl's law and Gustafson's law by evaluating interconnections on multi-core processors. Journal of Supercomputing 66(1):305-319.
- [97] Huang P-H, Leeb J-C (2010) Minimum zone evaluation of conicity error using minimum potential energy algorithms. Precision Engineering 34:709-717
- [98] Hutzschenreuter D, Härtig F, Wendt K, Lunze U, Löwe H (2015) Online Validation of Chebyshev Geometric Elements Algorithms using the TraCIM-System Journal of Mechanical Engineering and Automation, 5(3): 94-111
- [99] Ibaraki S, Kudo T, Yano T, Takatsuji T, Osawa S, Satoh O (2015) Estimation of three-dimensional volumetric errors of machining centers by a tracking interferometer, Precision Engineering 39:179-186.
- [100] IEEE Standard for Floating-Point Arithmetic IEEE Std 754-2008.
- [101] ISO 5436-2 Geometrical Product Specifications (GPS) - Surface texture: Profile method; Measurement standards - Part 02:00 Software measurement standards. International Organization for Standardization, 2001.
- [102] ISO 10303-1:1994 Industrial automation systems and integration - Product data representation and exchange - Part 01:00 Overview and fundamental principles
- [103] ISO (2001) Geometrical Product Specifications (GPS) -- Acceptance and reverification tests for coordinate measuring machines (CMM) -- Part 06:00 Estimation of errors in computing Gaussian associated features (10360 Part 6)
- [104] ISO 17450-1:2011, "Geometric Product Specification (GPS) - General concepts- Part 01:00 Model for geometrical specification and verification".
- [105] ISO (2011) Information technology -- Service management -- Part 01:00 Service management system requirements (20000 Part 1)
- [106] ISO (2012) Geometrical product specifications (GPS) -- Surface texture: Profile method; Measurement standards -- Part 02:00 Software measurement standards (5436 Part 2)
- [107] ISO (2013) Information technology -- Security techniques -- Information security management systems -- Requirements (27001)
- [108] ISO (2015) Quality management systems -- Requirements (9001)
- [109] ISO/IEC (2015) Information technology -- Vocabulary (2382)
- [110] Jeannerod C P, Louvet N, Muller, JM (2013) On the componentwise accuracy of complex floating-point division with an FMA, 21ST IEEE Symposium On Computer Arithmetic (ARITH), Proceedings Symposium on Computer Arithmetic 83-90
- [111] Jeannerod C P, Knöchel H, Monat C, Revy G (2010) Computing floating-point square roots via bivariate polynomial evaluation. LIP research report RR2008-38. 2010.
- [112] Johnston, H (2013) D-Wave sells second quantum computer-this time to NASA. Physics World, 26(7): 9.
- [113] Karp, Alan H. & Flatt, Horace P. (1990) Measuring Parallel Processor Performance". Communication of the ACM 33(5): 539-543.
- [114] Kearfott R B (1996) Interval computations: Introduction, uses, and resources. Euromath Bulletin, 2(1): 95-112.
- [115] Keller F, Wendt K, Härtig F (2015) Estimation of Test Uncertainty for TraCIM Reference Pairs, Series on advances in mathematics for applied sciences: 86 187 - 194 .
- [116] Knuth D E (1993) Artistic programming - a citation-classic commentary on the art of computer-programming, vol 2 seminumerical algorithms. Current contents/engineering technology & applied sciences.34 8 - 8.
- [117] Kok G J P, Harris, P M, Smith, I M and Forbes, A B, (2016) Reference data sets for testing metrology software, Metrologia,53(4).
- [118] Kornerup P, Lefevre V, Louvet N, Muller JM (2012) On the Computation of Correctly Rounded Sums. IEEE Transactions On Computers 61(3):289-298
- [119] Kornerup P, Lauter C, Lefevre V, Louvet N, Muller JM (2010) Computing Correctly Rounded Integer Powers in Floating-Point Arithmetic. ACM Transactions On Mathematical Software 37(1).
- [120] Kulisch U W (2009) Complete interval arithmetic and its implementation on the computer. In Numerical Validation in Current Hardware Architectures (pp. 7-26). Springer Berlin Heidelberg.
- [121] Kuzzyk M G, Pérez-Moreno J, (2013) Shafei J, Rules and scaling in nonlinear optics. Physics Reports 529:297-398.
- [122] Lee J, Bagheri B, Kao K-A (2014) Recent Advances and Trends of Cyber-Physical Systems and Big Data Analytics in Industrial. Informatics International Conference on Industrial Informatics, Brazil, doi:10.13140/2.1.1464.1920.
- [123] Leibniz G W, (1697) Brief an den Herzog Rudolf von Braunschweig-Wolfenbüttel vom 2. Januar 1697; Leibniz Museum Hannover
- [124] Levenberg K (1944) A method for the solution of certain non-linear problems in least squares. Quartely Journal of Applied Mathematics II, 2(2) :164-168.
- [125] Lions J-L, (1996) Ariane 5, Flight 501, Report of the Inquiry Board, European Space Agency.
- [126] Liu D C, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. Mathematical programming, 45(1):503-528.
- [127] Linares J M, Sprauel J M, Bourdet P (2009). Uncertainty of reference frames characterized by real time optical measurements: Application to Computer Assisted Orthopaedic Surgery. CIRP Annals-Manufacturing Technology, 58(1):447-450.
- [128] Louvet N, Muller J M, Panhaleux A (2010) Newton-Raphson Algorithms for Floating-Point Division using an FMA. Proceeding of the 21st IEEE International Conference Jul 2010 Rennes, France, 200-207.
- [129] Mailhe J, Linares J M, Sprauel J M (2009) The statistical gauge in geometrical verification Part I. Field of probability of the presence of matter, Precision Engineering 33(4): 333-341.
- [130] Mailhe J, Linares J M, Sprauel J M (2009) The statistical gauge in geometrical verification. Part II. The virtual gauge and verification process, Precision Engineering 33(4): 342-352.
- [131] Mailhe J, Linares JM, Sprauel JM, Bourdet P (2008) Geometrical checking by virtual gauge, including measurement uncertainties CIRP Annals - Manufacturing Technology 57(1): 513-516.
- [132] Mallet F, Ong F R, Palacios-Laloy A, Nguyen F, Bertet P, Vion D, Esteve D (2009) Single-shot qubit readout in circuit Quantum Electrodynamics. Nature Physics 5:791-795.
- [133] Markstein P (2008) The new IEEE-754 standard for floating point arithmetic. In Dagstuhl Seminar Proceedings. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.
- [134] Marquardt D W (1963) An algorithm for least squares estimation of non-linear parameters. Journal of the Society of Industrial and Applied Mathematics, 11(2) :431-441.
- [135] Martin-Dorel E, Melquiond G, Muller, JM (2013) Some issues related to double rounding. Bit Numerical Mathematics 53(4): 897-924
- [136] Mascarenhas W F (2014) The divergence of the BFGS and Gauss Newton methods, Mathematical Programming 147(1):253-276.
- [137] Moore R E, Bierbaum F (1979) Methods and applications of interval analysis Vol. 2. Philadelphia: Siam.
- [138] Moré J J (1978) The Levenberg-Marquardt algorithm: implementation and theory. Numerical analysis. Springer Berlin Heidelberg.
- [139] Moroni G, Petr S (2008) Geometric tolerance evaluation: A discussion on minimum zone fitting algorithms, Precision Engineering 32:232-237.
- [140] Muller J M, Brisebarre N, De Dinechin F, Jeannerod C P, Lefevre V, Melquiond G, Revol N, Stehle D, Torres S (2009) Handbook of floating-point arithmetic. Springer Science & Business Media.
- [141] Muller J M. Elementary Functions, Algorithms and Implementation. Birkhauser, Boston, 1997.
- [142] Muller J M (2005) Elementary Functions, Algorithms and Implementation. Boston, USA: Birkhäuser Verlag.

- [143] National Academy of Science and Engineering (2013) Recommendations for implementing the strategic initiative INDUSTRIE 4.0. Berlin, 15 Mai 2013.
- [144] Nocedal J (1980) Updating quasi-Newton matrices with limited storage. *Mathematics of computation*, 35(151):773-782.
- [145] Paul J M, Meyer B H (2007) Amdahl's law revisited for single chip systems. *International Journal of Parallel Programming* 35(2):101-123.
- [146] Piccione B, Cho C-H, van Vugt L K, Agarwal R (2012) All-optical active switching in individual semiconductor nanowires. *Nature Nanotechnology*, 07:00 640–645.
- [147] Piggott A Y, Lu J, Lagoudakis K G, Petykiewicz J, Babinec T M, Vuckovic J (2015) Inverse design and demonstration of a compact and broadband on-chip wavelength demultiplexer. *Nature photonics* 9(6):374 - 377.
- [148] Press release – Particle control in a quantum world. Royal Swedish Academy of Sciences. Retrieved 9 October 2012.
- [149] Rehwald P (1985), VDACS—An interface to transfer surface description data between CAD systems. *Computers & Graphics* 9(1):69-70.
- [150] Rodriguez-Fernández, J (1999) Ockham's razor. *Endeavour*, 23(3): 121-125.
- [151] Rohr D, Bach M, Neskovic G, Lindenstruth V, Pinke C, Philipsen O (2015) Lattice-CSC: Optimizing and Building an Efficient Supercomputer for Lattice-QCD and to Achieve First Place in Green500. *High Performance Computing, Isc High Performance 2015 Book Series: Lecture Notes in Computer Science* 9137:179-196.
- [152] Rump S M (2010) Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica* 19:287-449.
- [153] Rump S M (1999) INTLAB - INTERVAL LABORATORY. In Tibor Csendes, editor, *Developments in Reliable Computing*. Kluwer Academic Publishers, Dordrecht.
- [154] Savio E, De Chiffre L, Schmitt R (2007) Metrology of freeform shaped parts. *CIRP Annals-Manufacturing Technology* 56(2):810-830.
- [155] Schwenke H, Frank M, Hannaford J (2005) Error mapping of CMMs and machine tools by a single tracking interferometer. *CIRP Annals-Manufacturing Technology*, 54(1):475-478.
- [156] Schwenke H, Knapp W, Haitjema H, Weckenmann A, Schmitt R, Delbressine F (2008) Geometric error measurement and compensation of machines—An update. *CIRP Annals - Manufacturing Technology* 57(2): 660-675.
- [157] Shakarji C (1998) Least-squares fitting algorithms of the NIST algorithm testing system, *Journal of research of the National Institute of Standards and Technology*, 103:633–641.
- [158] Shakarji C, Clement A (2004) Reference Algorithms for Chebyshev and One-Sided Data Fitting for Coordinate Metrology, *CIRP Annals - Manufacturing Technology* 53(1): 439-442.
- [159] Soklakov A N (2002) Occam's razor as a formal basis for a physical theory. *Foundations of Physics Letters*, 15(2): 107-135.
- [160] Sturm M C (2009) Analyse d'un mémoire sur la résolution des équations numériques. In *Collected Works of Charles François Sturm* (pp.323-326). Birkhäuser Basel.
- [161] Sun W, McBride J W, Hill M (2010) A new approach to characterising aspheric surfaces, *Precision Engineering* 34:171–179.
- [162] Tarantola A (2005). *Inverse problem theory and methods for model parameter estimation*. Society for Industrial and Applied Mathematics.
- [163] Thompson S E, Parthasarathy S, (2006). Moore's law: the future of Si microelectronics. *Materials today*, 9(6), 20-25.
- [164] Thorburn, W M (1915) Occam's Razor. *Mind* XXIV (2): 287-288.
- [165] TraCIM e. V. (2014) Handelsregisterauszug VR 201236 OH-Nummer C1626559
- [166] TraCIM PTB (2016) Homepage of the TraCIM Service at PTB <https://tracim.ptb.de/tracim/index.jsf>, access déc-16
- [167] Turing A (1937) On Computable Numbers, with an Application to the Entscheidungsproblem, *Proc. London Math. Soc.*, 42:00:00 230-265
- [168] Van Der Hoeven J, Lecerf G, Quintin G. (2014) Modular SIMD arithmetic in Mathemagix. arXiv preprint arXiv:1407.3383.
- [169] Van Der Hoeven, J (2009). Ball arithmetic (<https://hal.archives-ouvertes.fr/hal-00432152v1>).
- [170] Veldhorst M, Yang C H, Hwang J C C, Huang W, Dehollain J P, Muhonen J T, Simmons S, Laucht A, Hudson F E, Itoh K M, Morello A, Dzurak A S (2015) A two-qubit logic gate in silicon. *Nature* 526:410-414.
- [171] Velenosi A, Campatelli G, Scippa A (2015) Axis geometrical errors analysis through a performance test to evaluate kinematic error in a five axis tilting-rotary table machine tool, *Precision Engineering* 39:00:00 224–233.
- [172] Vogel-Heuser B, Lee J, Leitão P (2015) Agents enabling cyber-physical production systems. *At - Automatisierungstechnik* 63(10): 777–789.
- [173] Wen X-L, Huang J-C, Sheng D-H, Wang F-L (2010) Conicity and cylindricity error evaluation using particle swarm optimization. *Precision Engineering* 34:338–344.
- [174] Whitehead N, Fit-Florea A (2011) Precision and Performance: Floating Point and IEEE 754 Compliance for NVIDIA GPUs. nVidia technical white paper, 2011
- [175] Wilkinson J H. (1961) Error analysis of direct methods of matrix inversion. *Journal of the ACM*, 8(3): 281-330.
- [176] Wilkinson, J H (1963), *Rounding Errors in Algebraic Processes*, HMSO.
- [177] Wu J, Luo Z, Zhang Y, Zhang N, Chen L (2013) Interval uncertain method for multibody mechanical systems using Chebyshev inclusion functions. *International Journal for Numerical Methods in Engineering*, 95(7): 608-630.
- [178] Zhang X, Zhang H, He X, Xu M, Jiang X (2013) Chebyshev fitting of complex surfaces for precision metrology, *Measurement* 46:3720–3724.
- [179] Zhang X, Jiang X, Scott P J (2011) Minimum zone evaluation of the form errors of quadric surfaces. *Precision Engineering* 35:383–389.
- [180] Zhang X, Jiang X, Scott J P (2011) A reliable method of minimum zone evaluation of cylindricity and conicity from coordinate measurement data, *Precision Engineering* 35:484–489.
- [181] Zhoulai F, Zhaojun B, Zhendong S. (2015) Automated backward error analysis for numerical code. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2015)*.