paceval / paceval_sources_documentation-external /

paceval Update readme.md ✓ dc6f3f1 · 3 months ago

Name	Name	Last commit date
..		
html	Update documentation	4 months ago
latex	Update documentation	4 months ago
media	Updated documentation	4 months ago
readme.md	Update readme.md	3 months ago

readme.md

paceval. the system independent mathematical engine 4.25

External documentation -

https://paceval.com/paceval_sources_documentation-external/html/index.html

What is paceval.?

paceval. is a mathematical engine that can calculate almost any complex mathematical expressions. The software reads a textual description of the expressions as a mathematical function, which may contain the basic arithmetic operations, the usual transcendental functions (trigonometry, exponential function, etc.) and other common operations.

Expressions can contain **any number of placeholders (variables)**. Calculations are performed in selectable precision (single, double and extended) and **distributed across all available processors** for maximum speed and effectiveness.

Additionally, **pace***val.* can also output an interval indicating the error limits due to the limited precision of floating point number formats.

The source code of **pace***val.* is in accordance with the C++ ISO standard (ISO/IEC 14882:1998) and therefore runs on every compiler since the 1990s. The library itself is particularly small and, thanks to its narrow programming interface, can be run immediately on the most important computer systems with little programming effort. The simple interface enables it to be used in all common programming languages via the foreign function interface (FFI).

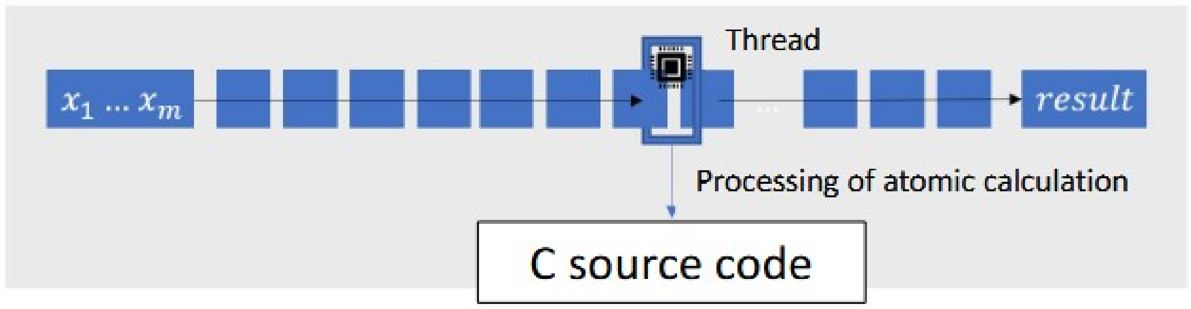
How does paceval work internally?

pace*val.* internally **creates and processes linked lists of atomic calculations** that represent the user's mathematical expressions. The usual approach is to create an expression tree from a text expression, see https://en.wikipedia.org/wiki/Binary_expression_tree. This has the well-known disadvantages such as memory consumption and the overhead and speed loss when creating and traversing the expression tree. Creating and processing an expression as a linked list offers many advantages over the usual approach, especially speed. The processing of the linked list when actually performing a calculation with values for the variables is done in a single C function that is called by each thread of the underlying system.

pace*val.* uses multithreading intensively to achieve maximum performance and energy efficiency of the respective hardware system. The main handling of multithreading in **pace***val.* in software is in the file paceval_fThreadHandling.cpp. **pace***val.* can also be used as a hardware driver for a new type of energy-efficient coprocessor. To do this, only the handling in the file paceval_fFPGAHandling.cpp has to be adapted to address the customer-specific hardware such as an FPGA.

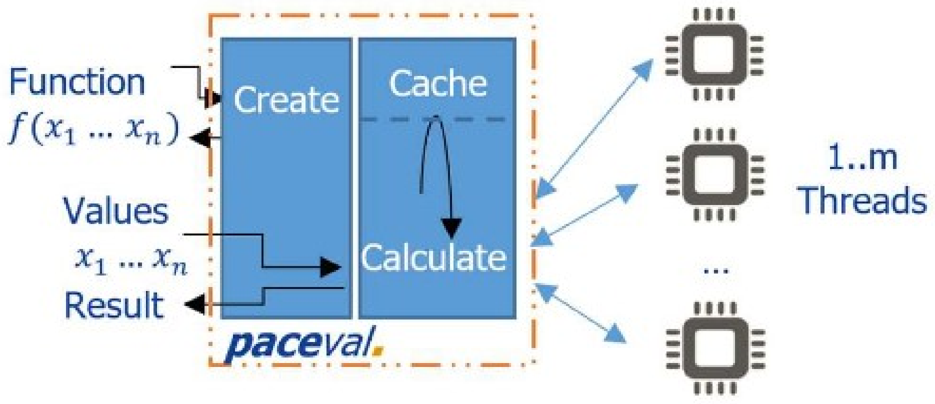
The source code performs this processing (this corresponds to the standard cycle used by all types of processors):

1. **FETCH** - Get the operator and operands (e.g. "addition of 2 and 3")
2. **DECODE** and **DECIDE** - Use the cached result from the cache or call the next step EXECUTE
3. **EXECUTE** - Execute the operator with the operands (e.g. call the C function to add 2 and 3 and get 5 as a result)
4. **WRITE BACK** - Cache the result of the atomic calculation (this includes lower and upper interval limits or errors)



“Create” and “Calculate”

Essentially **only two steps are necessary to perform calculations**. A computation object is first created ("Create" step) by the user passing a function and the set of variable identifiers. Concrete calculations can then be carried out again and again ("Calculate" step) with the object that has been created by the user passing the values for the variables.



In the **"Create" step**, a list of the individual atomic calculation rules is created with the computation object and aligned for maximum parallelizability. The user receives back a unique ID or "token" for the created object. Any number of computation objects can be created. In addition, there is no limit to the length of the function and the number of variables.

In the **"Calculate" step**, the calculation is carried out using the values for the variables. The list of individual atomic calculation rules is **distributed as partial sequences across all available processors or threads in the system** in order to achieve maximum speed. In addition, complex calculations of partial sequences that have been carried out once are **temporarily stored as a cache** so that they do not have to be calculated again if necessary.

Mathematical functions

All mathematical functions can be calculated and combined with logical operators. This allows **all financial, stochastic, engineering and scientific functions and also all models for machine learning** to be represented. In addition, the usual standard mathematical notation can easily be used.

The following operators, partial functions and symbols are currently supported:

- **Basic arithmetic operations** +, -, *, /
- **Logical operators** NOT, AND, OR, XOR, NAND, XNOR
- **Comparison operators** <, >, =, >=, <=, <>
- **Factorial** !, fac()
- **Constants** pi, e
- **Brackets** ()
- **Power/root functions** ^, sqr(), sqrt(), exp()
- **Logarithm functions and sigmoid function** lg(), ln(), sig()
- **Trigonometric functions and associated inverses** sin(), cos(), tan(), cot(), asin(), acos(), atan(), acot()
- **Hyperbolic functions and associated inverses** sinh(), cosh(), tanh(), coth(), arsinh(), arcosh(), artanh(), arcoth()
- **Numerical manipulations** sgn(), abs(), round(), ceil(), floor(), rand()
- **paceval specific numerical manipulations** ispos(), isposq(), isneg(), isnegq(), isnull()
- **Minimum and maximum** min, max
- **Modulo symmetric and mathematical variant** %, mod

The list of the main API

- paceval_CreateComputation() : Creates a computation object for the mathematical function f and the variables x1,x2,...,xn, i.e. in mathematical notation

  $$f(x_1, x_2, \ \dots, x_n)$$

  The user can also set whether calculations should be carried out with or without interval arithmetic. The reference returned when calling CreateComputation is an ID or "token" that uniquely represents the created calculation object, which is managed by **pace***val*. This token can then be used to carry out calculations and queries on the calculation object without having to retransmit the possibly very long function again.

- paceval_dGetComputationResult() [paceval_ldGetComputationResult(), paceval_fGetComputationResult()] : Performs a calculation on a computation object with the variables declared by paceval_CreateComputation() and with a set of values a1,a2,...,an for these variables, i.e. in mathematical notation

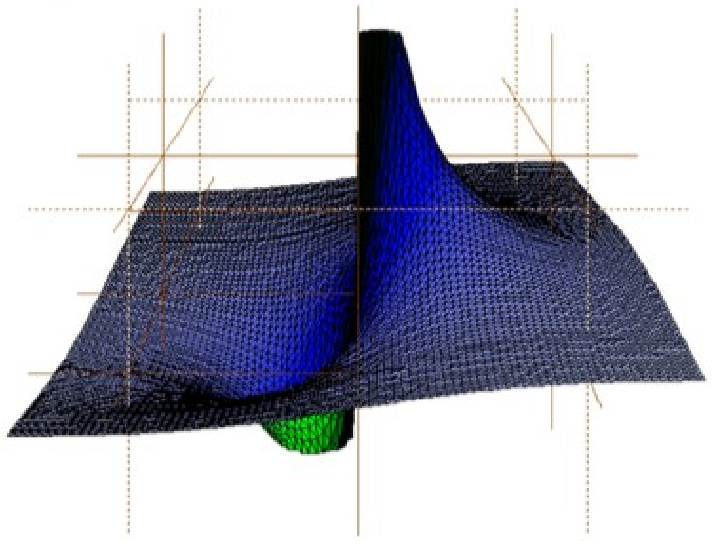
  $$y = f(a_1, a_2, \ \dots, a_n)$$

-

paceval_dGetComputationResultExt() [paceval_ldGetComputationResultExt(), paceval_fGetComputationResultExt()] : Performs multiple calculations in parallel on a computation object with the variables declared by paceval_CreateComputation() and with multiple sets of values a11,a12,...,a1n, a21,a22,...,a2n, ..., am1,am2,...,amn for these variables, i.e. in mathematical notation

$$\begin{gathered} y_1 = f(a_{11}, a_{12}, \ \dots, a_{1n}) \\ y_2 = f(a_{21}, a_{22}, \ \dots, a_{2n}) \\ \dots \\ y_m = f(a_{m1}, a_{m2}, \ \dots, a_{mn}) \end{gathered}$$

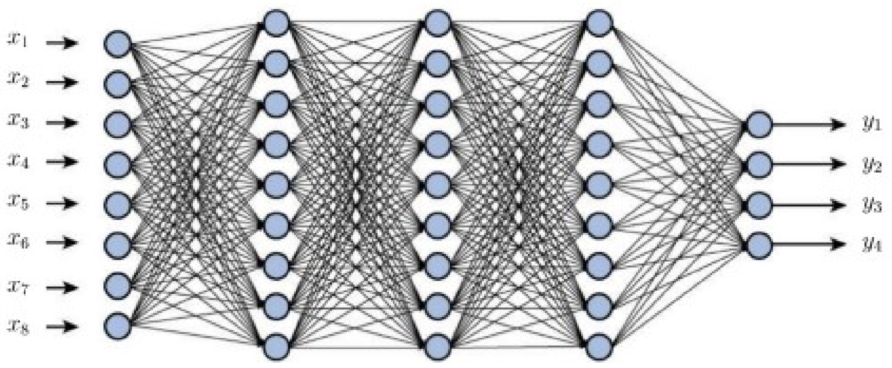
(Annotation: paceval_dGetComputationResultExt() should be used to get many results at once with a single call. This is useful, for example, when many different values need to be analyzed very quickly or when you plot 2D- or 3D-functions and graphics very fast.)



- paceval_dGetMultipleComputationsResults() [paceval_ldGetMultipleComputationsResults(), paceval_fGetMultipleComputationsResults()] : Performs multiple calculations in parallel on multiple computation objects with the variables declared by paceval_CreateComputation() and with the same set of values a1,a2,...,an for these variables, i.e. in mathematical notation

$$\begin{gathered} y_1 = f_1(a_1, a_2, \ \dots, a_n) \\ y_2 = f_2(a_1, a_2, \ \dots, a_n) \\ \dots \\ y_m = f_m(a_1, a_2, \ \dots, a_n) \end{gathered}$$

(Annotation: paceval_dGetMultipleComputationsResults() should be used to obtain results for several computation objects at the same time, e.g. for artificial neural network functions or decision trees, in which several functions with the same variable values are calculated in one step in order to decide on an object.)



- paceval_dGetMultipleComputationsResultsExt() [paceval_ldGetMultipleComputationsResultsExt(), paceval_fGetMultipleComputationsResultsExt()] : Performs multiple calculations in parallel on multiple computation objects using the variables declared by paceval_CreateComputation() and with multiple sets of values a11,a12,...,a1n, a21,a22,...,a2n, ..., am1,am2,...,amn for these variables, i.e. in mathematical notation

$$y_1 = f_1(a_{11}, a_{12}, \ \dots, a_{1n})$$
$$y_2 = f_2(a_{21}, a_{22}, \ \dots, a_{2n})$$
$$\dots$$
$$y_m = f_m(a_{m1}, a_{m2}, \ \dots, a_{mn})$$

  (Annotation: paceval_dGetMultipleComputationsResultsExt() should be used to obtain results for several computation objects at the same time and to carry out as many different evaluations as possible at once, e.g. for artificial neural network functions or decision trees, in which several functions with different variable values are calculated in one step in order to decide on several objects at the same time.

- paceval_DeleteComputation() : Deletes a computation object with the given token.