




How it works?

1. Enter text and formulas into the "Code" box on the left.
2. Press **F5** or click  to **calculate**. The results will appear in the "Output" box on the right as a professionally formatted **Html report**.
3. Click  to **print** or  to **copy** the output.

You can also **export** it to **Html** , **PDF**  or **MS Word**  document.

The language

The Calcpad language includes the following elements (click an item to insert):

- Real numbers: digits **0 - 9** and decimal point ".";
- Complex numbers: **re ± imi** (e.g. **3 - 2i**);
- Real vectors: [v_1 ; v_2 ; v_3 ; ...; v_n];
- Real matrices: [M_{11} ; M_{12} ; ... ; M_{1n} | M_{21} ; M_{22} ; ... ; M_{2n} ... | M_{m1} ; M_{m2} ; ... ; M_{mn}];
- Variables:
 - Latin letters: **a - z, A - Z**;
 - Greek letters: **α - ω, A - Ω**;
 - digits: **0 – 9**;
 - comma: " , " ;
 - prime symbols: ' , " , ' , ' ;
 - superscripts: **° , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , n , + , -** ;
 - special symbols: **¯ , ∅ , ∅ , ° , 4** ;
 - " _ " for subscript;

A variable name must start with a letter. Names are case sensitive.

- Constants:
 $\pi, e, \varphi, \gamma, g, G, M_E, M_S, c, h, \mu_0, \epsilon_0, k_e, e, m_e, m_p, m_n, N_A, \sigma, k_B, R, F, \gamma_c, \gamma_s, \gamma_a, \gamma_g, \gamma_w$
- Operators:
 - "!" – factorial;
 - "^" – exponent;
 - "/" – division;
 - "÷" – force division bar;
 - "\" – division;
 - "⊗" – modulo (remainder);
 - "*" – multiplication;
 - "-" – minus;
 - "+" – plus;
 - "≡" – equal to;
 - "≠" – not equal to;
 - "<" – less than;
 - ">" – greater than;
 - "≤" – less or equal;
 - "≥" – greater or equal;

“ \wedge ” – logical “AND”;

“ \vee ” – logical “OR”;

“ \oplus ” – logical “XOR”;

“ $=$ ” – assignment;

- Custom functions type $f(x; y; z; \dots)$;
- Built-in functions:
 - Trigonometric:
 - $\sin(x)$ - sine;
 - $\cos(x)$ - cosine;
 - $\tan(x)$ - tangent;
 - $\csc(x)$ - cosecant;
 - $\sec(x)$ - secant;
 - $\cot(x)$ - cotangent;
 - Hyperbolic:
 - $\sinh(x)$ - hyperbolic sine;
 - $\cosh(x)$ - hyperbolic cosine;
 - $\tanh(x)$ - hyperbolic tangent;
 - $\operatorname{csch}(x)$ - hyperbolic cosecant;
 - $\operatorname{sech}(x)$ - hyperbolic secant;
 - $\operatorname{coth}(x)$ - hyperbolic cotangent;
 - Inverse trigonometric:
 - $\operatorname{asin}(x)$ - inverse sine;
 - $\operatorname{acos}(x)$ - inverse cosine;
 - $\operatorname{atan}(x)$ - inverse tangent;
 - $\operatorname{atan2}(x; y)$ - the angle whose tangent is the quotient of y and x ;
 - $\operatorname{acsc}(x)$ - inverse cosecant;
 - $\operatorname{asec}(x)$ - inverse secant;
 - $\operatorname{acot}(x)$ - inverse cotangent;
 - Inverse hyperbolic:
 - $\operatorname{asinh}(x)$ - inverse hyperbolic sine;
 - $\operatorname{acosh}(x)$ - inverse hyperbolic cosine;
 - $\operatorname{atanh}(x)$ - inverse hyperbolic tangent;
 - $\operatorname{acsch}(x)$ - inverse hyperbolic cosecant;
 - $\operatorname{asech}(x)$ - inverse hyperbolic secant;
 - $\operatorname{acoth}(x)$ - inverse hyperbolic cotangent;
 - Logarithmic, exponential and roots:
 - $\log(x)$ - decimal logarithm;
 - $\ln(x)$ - natural logarithm;
 - $\log_2(x)$ - binary logarithm;
 - $\exp(x)$ - exponential function;
 - $\operatorname{sqr}(x)$ or $\operatorname{sqrt}(x)$ - square root;

- cbrt**(x) - cubic root;
- root**(x ; n) - n -th root;
- Rounding:
 - round**(x) - round to the nearest integer;
 - floor**(x) - round to the smaller integer (towards $-\infty$);
 - ceiling**(x) - round to the greater integer (towards $+\infty$);
 - trunc**(x) - round to the smaller integer (towards zero);
- Integer:
 - mod**(x ; y) - the remainder of an integer division;
 - gcd**(x ; y ; $z...$) - the greatest common divisor of several integers;
 - lcm**(x ; y ; $z...$) - the least common multiple of several integers;
- Complex:
 - abs**(x) - absolute value/magnitude;
 - re**(x) - the real part of a complex number;
 - im**(x) - the imaginary part of a complex number;
 - phase**(x) - the phase of a complex number;
- Aggregate and interpolation:
 - min**(A ; \vec{b} ; $c...$) - minimum of multiple values;
 - max**(A ; \vec{b} ; $c...$) - maximum of multiple values;
 - sum**(A ; \vec{b} ; $c...$) - sum of multiple values;
 - sumsq**(A ; \vec{b} ; $c...$) - sum of squares
 - srss**(A ; \vec{b} ; $c...$) - square root of sum of squares;
 - average**(A ; \vec{b} ; $c...$) - average of multiple value;
 - product**(A ; \vec{b} ; $c...$) - product of multiple values;
 - mean**(A ; \vec{b} ; $c...$) - geometric mean;
 - take**(n ; A ; \vec{b} ; $c...$) - returns the n -th element from the list;
 - line**(x ; A ; \vec{b} ; $c...$) - linear interpolation;
 - spline**(x ; A ; \vec{b} ; $c...$) - Hermite spline interpolation;
- Conditional and logical:
 - if**($cond$; $value\text{-if-true}$; $value\text{-if-false}$) - conditional evaluation;
 - switch**($cond1$; $value1$; $cond2$; $value2$; ...; $default$) - selective evaluation;
 - not**(x) - logical "NOT";
 - and**(A ; \vec{b} ; $c...$) - logical "AND";
 - or**(A ; \vec{b} ; $c...$) - logical "OR";
 - xor**(A ; \vec{b} ; $c...$) - logical "XOR";
- Other:
 - sign**(x) - sign of a number;
 - random**(x) - random number between 0 and x ;

◦ Vector:

Creational:

vector(n) - creates an empty vector with length n ;

fill($\vec{v}; x$) - fills the vector \vec{v} with value x ;

range($x_1; x_n; s$) - creates a vector with values spanning from x_1 to x_n with step s ;

Structural:

len(\vec{v}) - returns the length of the vector \vec{v} ;

size(\vec{v}) - the actual size of the vector \vec{v} (the index of the last non-zero element);

resize($\vec{v}; n$) - sets a new length n of the vector \vec{v} ;

join($A; \vec{b}; c...$) - creates a vector by joining the arguments in the list – matrices, vectors and scalars;

slice($\vec{v}; i_1; i_2$) - returns the part of the vector \vec{v} bounded by indexes i_1 and i_2 inclusive;

first($\vec{v}; n$) - the first n elements of the vector \vec{v} ;

last($\vec{v}; n$) - the last n elements of the vector \vec{v} ;

extract($\vec{v}; \vec{i}$) - extracts those elements from \vec{v} which indexes are contained in \vec{i} ;

Data:

sort(\vec{v}) - sorts the vector \vec{v} in ascending order;

rsort(\vec{v}) - sorts the vector \vec{v} in descending order;

order(\vec{v}) - the indexes of \vec{v} , in ascending order by the elements of \vec{v} ;

revorder(\vec{v}) - the indexes of \vec{v} , in descending order by the elements of \vec{v} ;

reverse(\vec{v}) - vector containing the elements of \vec{v} in reverse order;

count($\vec{v}; x; i$) - the number of elements of \vec{v} equal to x with index $\geq i$;

search($\vec{v}; x; i$) - the index of the first element in \vec{v} with index $\geq i$ that is equal to x ;

find($\vec{v}; x; i$) or

find_eq($\vec{v}; x; i$) - the indexes of all elements in \vec{v} , after the i -th, that are $= x$;

find_ne($\vec{v}; x; i$) - the indexes of all elements in \vec{v} , after the i -th, that are $\neq x$;

find_lt($\vec{v}; x; i$) - the indexes of all elements in \vec{v} , after the i -th, that are $< x$;

find_le($\vec{v}; x; i$) - the indexes of all elements in \vec{v} , after the i -th, that are $\leq x$;

find_gt($\vec{v}; x; i$) - the indexes of all elements in \vec{v} , after the i -th, that are $> x$;

find_ge($\vec{v}; x; i$) - the indexes of all elements in \vec{v} , after the i -th, that are $\geq x$;

lookup($\vec{a}; \vec{b}; x$) or

lookup_eq($\vec{a}; \vec{b}; x$) - all elements of \vec{a} for which the corresponding elements of \vec{b} are $= x$;

lookup_ne($\vec{a}; \vec{b}; x$) - all elements of \vec{a} for which the corresponding elements of \vec{b} are $\neq x$;

lookup_lt($\vec{a}; \vec{b}; x$) - all elements of \vec{a} for which the corresponding elements of \vec{b} are $< x$;

lookup_le($\vec{a}; \vec{b}; x$) - all elements of \vec{a} for which the corresponding elements of \vec{b} are $\leq x$;

lookup_gt($\vec{a}; \vec{b}; x$) - all elements of \vec{a} for which the corresponding elements of \vec{b} are $> x$;

lookup_ge($\vec{a}; \vec{b}; x$) - all elements of \vec{a} for which the corresponding elements of \vec{b} are $\geq x$;

Math:

norm_1(\vec{v}) - L1 (Manhattan) norm of the vector \vec{v} ;
norm(\vec{v}) or
norm_2(\vec{v}) or
norm_e(\vec{v}) - L2 (Euclidean) norm of the vector \vec{v} ;
norm_p($\vec{v}; p$) - Lp norm of the vector \vec{v} ;
norm_i(\vec{v}) - L ∞ (infinity) norm of the vector \vec{v} ;
unit(\vec{v}) - normalized form of the vector \vec{v} (with L2 norm = 1);
dot($\vec{a}; \vec{b}$) - scalar product of two vectors \vec{a} and \vec{b} ;
cross($\vec{a}; \vec{b}$) - cross product of two vectors \vec{a} and \vec{b} (with length 2 or 3);

○ Matrix:

Creational:

matrix($m; n$) - creates an empty matrix with dimensions $m \times n$;
identity(n) - creates an identity matrix with dimensions $n \times n$;
diagonal($n; d$) - creates a diagonal matrix with dimensions $n \times n$ and fills the diagonal with value d ;
column($m; c$) - creates a column matrix with dimensions $m \times 1$, filled with value c ;
utriang(n) - creates an upper triangular matrix with dimensions $n \times n$;
ltriang(n) - creates a lower triangular matrix with dimensions $n \times n$;
symmetric(n) - creates a symmetric matrix with dimensions $n \times n$;
vec2diag(\vec{v}) - creates a diagonal matrix from the elements of vector \vec{v} ;
vec2col(\vec{v}) - creates a column matrix from the elements of vector \vec{v} ;
join_cols($\vec{c}_1; \vec{c}_2; \vec{c}_3 \dots$) - creates a matrix by joining column vectors;
join_rows($\vec{r}_1; \vec{r}_2; \vec{r}_3 \dots$) - creates a matrix by joining row vectors;
augment($A; B; C \dots$) - creates a matrix by appending matrices $A; B; C$ side by side;
stack($A; B; C \dots$) - creates a matrix by stacking matrices $A; B; C$ one below the other;

Structural:

n_rows(M) - number of rows in matrix M ;
n_cols(M) - number of columns in matrix M ;
resize($M; m; n$) - sets new dimensions m and n for matrix M ;
fill($M; x$) - fills the matrix M with value x ;
fill_row($M; x; i$) - fills the i -th row of matrix M with value x ;
fill_col($M; x; j$) - fills the j -th column of matrix M with value x ;
copy($A; B; i; j$) - copies all elements from A to B , starting from indexes i and j of B ;
add($A; B; i; j$) - adds all elements from A to those of B , starting from indexes i and j of B ;
row($M; i$) - extracts the i -th row of matrix M as a vector;
col($M; j$) - extracts the j -th column of matrix M as a vector;
extract_rows($M; \vec{i}$) - extracts the rows from matrix M whose indexes are contained in vector \vec{i} ;

extract_cols($M; \vec{j}$) - extracts the columns from matrix M whose indexes are contained in vector \vec{j} ;

diag2vec(\vec{v}) - extracts the diagonal elements of matrix M to a vector;

submatrix($M; i_1; i_2; j_1; j_2$) - extracts a submatrix of M , bounded between rows i_1 and i_2 and columns j_1 and j_2 , incl.;

Data:

sort_cols($M; i$) - sorts the columns of M based on the values in row i in ascending order;

rsort_cols($M; i$) - sorts the columns of M based on the values in row i in descending order;

sort_rows($M; j$) - sorts the rows of M based on the values in column j in ascending order;

rsort_rows($M; j$) - sorts the rows of M based on the values in column j in descending order;

order_cols($M; i$) - the indexes of the columns of M in ascending order by the values in row i ;

revorder_cols($M; i$) - the indexes of the columns of M in descending order by the values in row i ;

order_rows($M; j$) - the indexes of the rows of M in ascending order by the values in column j ;

revorder_rows($M; j$) - the indexes of the rows of M in descending order by the values in column j ;

mcount($M; x$) - number of occurrences of value x in matrix M ;

msearch($M; x; i; j$) - vector with the two indexes of the first occurrence of x in matrix M , starting from indexes i and j ;

mfind($M; x$) - the indexes of all elements in matrix M equal to x ;

mfind_eq($M; x$) - the indexes of all elements in matrix M equal to x ;

mfind_ne($M; x$) - the indexes of all elements in matrix M not equal to x ;

mfind_lt($M; x$) - the indexes of all elements in matrix M less than x ;

mfind_le($M; x$) - the indexes of all elements in matrix M less than or equal to x ;

mfind_gt($M; x$) - the indexes of all elements in matrix M greater than x ;

mfind_ge($M; x$) - the indexes of all elements in matrix M greater than or equal to x ;

hlookup($M; x; i_1; i_2$) - the values from row i_2 of M , for which the elements from row i_1 are equal to x ;

hlookup_eq($M; x; i_1; i_2$) - the values from row i_2 of M , for which the elements from row i_1 are equal to x ;

hlookup_ne($M; x; i_1; i_2$) - the values from row i_2 of M , for which the elements from row i_1 are not equal to x ;

hlookup_lt($M; x; i_1; i_2$) - the values from row i_2 of M , for which the elements from row i_1 are less than x ;

hlookup_le($M; x; i_1; i_2$) - the values from row i_2 of M , for which the elements from row i_1 are less than or equal to x ;
hlookup_gt($M; x; i_1; i_2$) - the values from row i_2 of M , for which the elements from row i_1 are greater than x ;
hlookup_ge($M; x; i_1; i_2$) - the values from row i_2 of M , for which the elements from row i_1 are greater than or equal to x ;
vlookup($M; x; j_1; j_2$) - the values from column j_2 of M , for which the elements from column j_1 are equal to x ;
vlookup_eq($M; x; j_1; j_2$) - the values from column j_2 of M , for which the elements from column j_1 are equal to x ;
vlookup_ne($M; x; j_1; j_2$) - the values from column j_2 of M , for which the elements from column j_1 are not equal to x ;
vlookup_lt($M; x; j_1; j_2$) - the values from column j_2 of M , for which the elements from column j_1 are less than x ;
vlookup_le($M; x; j_1; j_2$) - the values from column j_2 of M , for which the elements from column j_1 are less than or equal to x ;
vlookup_gt($M; x; j_1; j_2$) - the values from column j_2 of M , for which the elements from column j_1 are greater than x ;
vlookup_ge($M; x; j_1; j_2$) - the values from column j_2 of M , for which the elements from column j_1 are greater than or equal to x ;

Math:

hprod($A; B$) - Hadamard product of matrices A and B ;
fprod($A; B$) - Frobenius product of matrices A and B ;
kprod($A; B$) - Kronecker product of matrices A and B ;
mnorm(M) or
mnorm_2(M) - L2 norm of matrix M ;
mnorm_1(M) - L1 norm of matrix M ;
mnorm_2(M) - Frobenius norm of matrix M ;
mnorm_i(M) - L_∞ norm of matrix M ;
cond(M) or
cond_e(M) - condition number of M based on the Euclidean norm of the matrix;
cond_1(M) - condition number of M based on the L1 norm;
cond_2(M) - condition number of M based on the L2 norm;
cond_i(M) - condition number of M based on the L_∞ norm;
det(M) - determinant of matrix M ;
rank(M) - rank of matrix M ;
trace(M) - trace of matrix M ;
transp(M) - transpose of matrix M ;
adj(M) - adjugate of matrix M ;
cofactor(M) - cofactor matrix of M ;
eigenvals(M) - eigenvalues of matrix M ;

eigenvecs(M) - eigenvectors of matrix M ;

eigen(M) - eigenvalues and eigenvectors of matrix M ;

cholesky(M) - Cholesky decomposition of a symmetric, positive-definite matrix M ;

lu(M) - LU decomposition of matrix M ;

qr(M) - QR decomposition of matrix M ;

svd(M) - singular value decomposition of M ;

inverse(M) - inverse of matrix M ;

lsolve(A ; \vec{b}) - solves the system of linear equations $A\vec{x} = \vec{b}$ using LDL^T decomposition for symmetric matrices, and LU for non-symmetric;

clsolve(A ; \vec{b}) - solves the linear matrix equation $A\vec{x} = \vec{b}$ with a symmetric, positive-definite matrix A using Cholesky decomposition;

Double interpolation:

take(x ; y ; M) - returns the element of matrix M at indexes x and y ;

line(x ; y ; M) - double linear interpolation from the elements of matrix M based on the values of x and y ;

spline(x ; y ; M) - double Hermite spline interpolation from the elements of matrix M based on the values of x and y .

- Comments: "Title" or 'text' in double or single quotes, respectively.

HTML, **CSS**, **JS** and **SVG** are allowed.

- Graphing and plotting:

$\$Plot\{f(x) @ x = a : b\}$ - simple plot;

$\$Plot\{x(t) | y(t) @ t = a : b\}$ - parametric;

$\$Plot\{f_1(x) \& f_2(x) \& \dots @ x = a : b\}$ - multiple;

$\$Plot\{x_1(t) | y_1(t) \& x_2(t) | y_2(t) \& \dots @ x = a : b\}$ - multiple parametric;

$\$Map\{f(x; y) @ x = a : b \& y = c : d\}$ - 2D color map of a 3D surface;

PlotHeight - height of plot area in pixels;

PlotWidth - width of plot area in pixels;

PlotStep - grid size for map plotting.

- Iterative and numerical methods:

$\$Root\{f(x) = const @ x = a : b\}$ - root finding for $f(x) = const$;

$\$Root\{f(x) @ x = a : b\}$ - root finding for $f(x) = 0$;

$\$Find\{f(x) @ x = a : b\}$ - similar to above, but x is not required to be a precise solution;

$\$Sup\{f(x) @ x = a : b\}$ - local maximum of a function;

$\$Inf\{f(x) @ x = a : b\}$ - local minimum of a function;

$\$Area\{f(x) @ x = a : b\}$ - adaptive Gauss-Lobatto numerical integration;

$\$Integral\{f(x) @ x = a : b\}$ - Tanh-Sinh numerical integration;

$\$Slope\{f(x) @ x = a\}$ - numerical differentiation;

$\$Sum\{f(x) @ k = a : b\}$ - iterative sum;

$\$Product\{f(k) @ k = a : b\}$ - iterative product;

$\$Repeat\{f(k) @ k = a : b\}$ - general inline iterative procedure;

Precision - relative precision for numerical methods [10^{-2} ; 10^{-16}] (default is 10^{-12})

- Program flow control:

Simple:

```
#if condition
    your code goes here
#end if
```

Alternative:

```
#if condition
    your code goes here
#else
    some other code
#end if
```

Complete:

```
#if condition1
    your code goes here
#else if condition2
    your code goes here
#else
    some other code
#end if
```

You can add or omit as many "#else ifs" as needed. Only one "#else" is allowed.

You can omit this too.

- Iteration blocks:

Simple:

```
#repeat number of repetitions
    your code goes here
#loop
```

With conditional break/continue:

```
#repeat number of repetitions
    your code goes here
    #if condition
        #break or #continue
    #end if
    some more code
#loop
```

With counter:

```
#for counter = start : end
    your code goes here
#loop
```

With condition:

```
#while condition  
    your code goes here  
#loop
```

- Modules and macros/string variables:

Modules:

```
#include filename - include external file (module);  
#local - start local section (not to be included);  
#global - start global section (to be included);
```

Inline string variable:

```
#def variable_name$ = content
```

Multiline string variable:

```
#def variable_name$  
    content line 1  
    content line 2  
    ...  
#end def
```

Inline macro:

```
#def macro_name$(param1$; param2$; ...) = content
```

Multiline macro:

```
#def macro_name$(param1$; param2$; ...)  
    content line 1  
    content line 2  
    ...  
#end def
```

- Output control:

```
#hide - hide the report contents;  
#show - always show the contents (default);  
#pre - show the next contents only before calculations;  
#post - show the next contents only after calculations;  
#val - show only the result, without the equation;  
#equ - show complete equations and results (default);  
#noc - show only equations without results (no calculations);  
#nosub - do not substitute variables (no substitution);  
#novar - show equations only with substituted values (no variables);  
#varsub - show equations with variables and substituted values (default);  
#round n - rounds the output to n digits after the decimal point.
```

- Breakpoints for step-by-step execution:

```
#pause - calculates down to the current line and waits for the user to resume manually;  
#input - renders an input form to the current line and waits for user input.
```

Each of the above commands is effective after the current line until the end of the report or another command that overwrites it.

- Units for trigonometric functions: **#deg** - degrees, **#rad** - radians, **#gra** - gradians;
- Separator for target units: **|**;
- Return angles with units: **ReturnAngleUnits** = 1;

- Dimensionless: **%**, **‰**;

- Angle: **°**, **'**, **"**, **deg**, **rad**, **grad**, **rev**;

- Metric units (SI and compatible):

Mass: **g**, **hg**, **kg**, **t**, **kt**, **Mt**, **Gt**, **dg**, **cg**, **mg**, **µg**, **Da** (or **u**);

Length: **m**, **km**, **dm**, **cm**, **mm**, **µm**, **nm**, **pm**, **AU**, **ly**;

Time: **s**, **ms**, **µs**, **ns**, **ps**, **min**, **h**, **d**, **w**, **y**;

Frequency: **Hz**, **kHz**, **MHz**, **GHz**, **THz**, **mHz**, **µHz**, **nHz**, **pHz**, **rpm**;

Speed: **kmh**;

Electric current: **A**, **kA**, **MA**, **GA**, **TA**, **mA**, **µA**, **nA**, **pA**;

Temperature: **°C**, **Δ°C**, **K**;

Amount of substance: **mol**;

Luminous intensity: **cd**;

Area: **a**, **daa**, **ha**;

Volume: **L**, **daL**, **hL**, **dL**, **cL**, **mL**, **µL**, **nL**, **pL**;

Force: **N**, **daN**, **hN**, **kN**, **MN**, **GN**, **TN**, **gf**, **kgf**, **tf**, **dyn**;

Moment: **Nm**, **kNm**;

Pressure: **Pa**, **daPa**, **hPa**, **kPa**, **MPa**, **GPa**, **TPa**, **dPa**, **cPa**, **mPa**, **µPa**, **nPa**, **pPa**,
bar, **mbar**, **µbar**, **atm**, **at**, **Torr**, **mmHg**;

Viscosity: **P**, **cP**, **St**, **cSt**;

Energy work: **J**, **kJ**, **MJ**, **GJ**, **TJ**, **mJ**, **µJ**, **nJ**, **pJ**,

Wh, **kWh**, **MWh**, **GWh**, **TWh**, **mWh**, **µWh**, **nWh**, **pWh**,

eV, **keV**, **MeV**, **GeV**, **TeV**, **PeV**, **EeV**, **cal**, **kcal**, **erg**;

Power: **W**, **kW**, **MW**, **GW**, **TW**, **mW**, **µW**, **nW**, **pW**, **hpM**, **ks**,

VA, **kVA**, **MVA**, **GVA**, **TVA**, **mVA**, **µVA**, **nVA**, **pVA**,

VAR, **kVAR**, **MVAR**, **GVAR**, **TVAR**, **mVAR**, **µVAR**, **nVAR**, **pVAR**;

Electric charge: **C**, **kC**, **MC**, **GC**, **TC**, **mC**, **µC**, **nC**, **pC**, **Ah**, **mAh**;

Potential: **V**, **kV**, **MV**, **GV**, **TV**, **mV**, **µV**, **nV**, **pV**;

Capacitance: **F**, **kF**, **MF**, **GF**, **TF**, **mF**, **µF**, **nF**, **pF**;

Resistance: **Ω**, **kΩ**, **MΩ**, **GΩ**, **TΩ**, **mΩ**, **µΩ**, **nΩ**, **pΩ**;

Conductance: **S**, **kS**, **MS**, **GS**, **TS**, **mS**, **µS**, **nS**, **pS**, **Ω**, **kΩ**, **MΩ**, **GΩ**, **TΩ**, **mΩ**, **µΩ**, **nΩ**, **pΩ**;

Magnetic flux: **Wb**, **kWb**, **MWb**, **GWb**, **TWb**, **mWb**, **µWb**, **nWb**, **pWb**;

Magnetic flux density: **T**, **kT**, **MT**, **GT**, **TT**, **mT**, **µT**, **nT**, **pT**;

Inductance: **H**, **kH**, **MH**, **GH**, **TH**, **mH**, **µH**, **nH**, **pH**;

Luminous flux: **lm**;

Illuminance: **lx**;

Radioactivity: **Bq**, **kBq**, **MBq**, **GBq**, **TBq**, **mBq**, **µBq**, **nBq**, **pBq**, **Ci**, **Rd**;

Absorbed dose: Gy, kGy, MGy, GGy, TGy, mGy, µGy, nGy, pGy;

Equivalent dose: Sv, kSv, MSv, GSv, TSv, mSv, µSv, nSv, pSv;

Catalytic activity: kat;

- Non-metric units (Imperial/US):

Mass: gr, dr, oz, lb (or lbm, lb_m), kipm (or kip_m), st, qr,

cwt (or cwt_uk, cwt_us), ton (or ton_uk, ton_us), slug;

Length: th, in, ft, yd, ch, fur, mi, ftm (or ftm_uk, ftm_us),

cable (or cable_uk, cable_us), nmi, li, rod, pole, perch, lea;

Speed: mph, knot;

Temperature: °F, Δ°F, °R;

Area: rood, ac;

Volume, fluid: fl_oz, gi, pt, qt, gal, bbl, or:

fl_oz_uk, gi_uk, pt_uk, qt_uk, gal_uk, bbl_uk,

fl_oz_us, gi_us, pt_us, qt_us, gal_us, bbl_us;

Volume, dry: (US) pt_dry, (US) qt_dry, (US) gal_dry, (US) bbl_dry,

pk (or pk_uk, pk_us), bu (or bu_uk, bu_us);

Force: ozf (or oz_f), lbf (or lb_f), kip (or kipf, kip_f), tonf (or ton_f), pdl;

Pressure: osi, osf psi, psf, ksi, ksf, tsi, tsf, inHg;

Energy/work: BTU, therm (or therm_uk, therm_us), quad;

Power: hp, hpE, hpS;

- Custom units - .Name = expression.

Names can include currency symbols: €, £, ₣, ¥, ¢, ₧, ₹, ₨, ₮.