Implementing an expression parser into the silting up model

- A new method parameter was introduced after time-step parameter in silting-up model.
- Two new methods were implemented for silting up:

  method 2:
  allows to specify the parameters used in the silting up equations. They are recommended to be set as follows to result in algorithms as before with method 1:

```
65.0   # parameter A in I_0 = A (initial infiltration capacity
12.2   # parameter B in I_end = B*(dg^C)*(fd^D)
0.52   # parameter C in I_end = B*(dg^C)*(fd^D)
-0.64  # parameter D in I_end = B*(dg^C)*(fd^D)
0.013  # parameter E in Cv = E*(fd^F)*(dg^G)*(t_cult^H)
-1.03  # parameter F in Cv = E*(fd^F)*(dg^G)*(t_cult^H)
0.7    # parameter G in Cv = E*(fd^F)*(dg^G)*(t_cult^H)
-0.19  # parameter H in Cv = E*(fd^F)*(dg^G)*(t_cult^H)
```

  The eight parameters must be specified after the last (readgrids) parameters of the [SiltingUpModel]-section

  method 3:
  Here it is possible to define custom expressions to calculate a number of equations. The control file must be extended by a tag called SiltingUpExpressions. Example:

```
SiltingUpExpressions {
W = ((P>0.05)&(P<76.2))*(11.89+8.73*log10(Abs(P+0.001))) + (P>=76.2)*28.33;
X = A;                           # start infiltration rate;
Y = B*K^C*(L*100)^D;             # end infiltration rate
C1 = (100*L)^F;     # F = SU_PAR06 C1 will be stored in a new internal variable
C2 = K^G;           # G = SU_PAR07 C2 will be stored in a new internal variable
C3 = (O+0.001)^H;   # H = SU_PAR08 C3 will be stored in a new internal variable
Z = (O<=0) + (O>0)*(E*C1*C2*C3); # E = SU_PAR05, O = time since last tillage
V = ((X-Y)*exp(-Z*Q)+Y)*R/60;    # potential infiltration
}
```

  Short description of the expression parser and the expression list syntax for method 3:

  - Expressions can be defined following algebraic rules.
  - Each line contains a single expression which must be closed with a semi colon.
  - Each assignment (e.g. A = 15) results in creating or updating a value in the internal variable list.
  - A number of values is already defined by WaSiM (as interface from the calling module), and WaSiM expects some other values to be defined after all expressions were called.

  The expression parser is based on the source code of the expression parser used in SpeQ Mathematics (http://www.speqmath.com/tutorials/expression_parser_cpp/index.html), written by Jos de Jong, 2007. It was adopted to the usage in WaSiM by simplifying the error handling (exceptions are to be handled by WaSiM), extracting the variable list as an external class (to be handled by WaSiM) and some other minor technical changes.

<u>Operators</u> (ascending precedence per line, no precedence within a line):

& | *<< >>*	*(AND, OR, BITSHIFTLEFT, BITSHIFTRIGHT)*
= <> < > <= >=	*(EQUAL, UNEQUAL, SMALLER, LARGER, SMALLEREQ, LARGEREQ)*
+ -	*(PLUS, MINUS)*
* / % ||	*(MULTIPLY, DIVIDE, MODULUS, XOR)*
^	*(POW)*
!	*(FACTORIAL)*

<u>Functions</u> (must be used with brackets):

Abs(arg), Exp(arg), Sign(arg), Sqrt(arg), Log(arg), Log10(arg), Sin(arg), Cos(arg), Tan(arg), ASin(arg), ACos(arg), ATan(arg), Factorial(arg)

<u>Variables</u>:

Pi, Euler (not only e, e is a predefined variable used by WaSiM to deliver a value to the expression parser interface). You can define your own variables, even with more than one significant character length, e.g. Info or Help etc. There is no distinction between upper and lower case in function names and variables.

<u>Other</u>:

Scientific notation supported

====> what values WaSiM defines for input (can be used in any expression)

A to J: values as used in soiltable with names *SU_PAR01* to *SU_PAR10*

K: grain size distribution Dg, internally calculated after

double FClay = log004+log2;

*double FSilt = 0.3326 * (log2+log6_3) + 0.3348 *(log6_3+log20) + 0.1704 * (log20+log36) + 0.1622 * (log36+log63) ;*

*double FSand = 0.1336 * (log63+log100) + 0.2005 *(log100+log200) + 0.3318 * (log200+log630) + 0.3341 *(log630+log2000);*

double FStones1 = (log2000+log6300);

double FStones2 = (log6300+log20000);

double FStones3 = (log20000+log63000);

double FStones4 = (log63000+log200000);

*double dg = (FClay*dFractionClay + FSilt*dFractionSilt + FSand*dFractionSand + FStones1*dFractionStones1 + FStones2*dFractionStones2 + FStones3*dFractionStones3 + Fstones4*dFractionStones4) / 2.0;*

with fractions of each grain size class taken from the soil table

L: fraction of sand

M: fraction of clay

N:	fraction of silt
O:	t_cult, time since last soil cultivation (in days)
P:	rain intensity in mm/h, taken from precipitation input
Q:	e_kin: accumulated cinetic energy: for all expressions resulting in W, X, Y or Z: result value of the last time time step; for V: value of the actual time step
R:	internal time step in minutes

====> what WaSiM expects for output: (ranging from Z downwards, will be used by WaSiM when going ahead)

Z:	silting up disposition SDISP
Y:	end infiltration rate i_inf
X:	start infiltration rate i0
W:	actual cinetic energy
V:	potential infiltration rate inf_pot, depending on energy, siting up disposition, inf_start and inf_infinite

Order of expressions evaluated by WaSiM:

- expressions returning W, X, Y and Z are independently of each other.
- expression V must be called as last call in any case, since WaSiM will update EKIN internally using the energy-result (in W) and V depends on all the other results W to Z
- other expressions for storing intermediate results may be defined at any position in the expression list before the results will be used in another expression
- In order to use custom parameters for variables A to J, each soil table entry may optionally contain values for parameters SU_PAR01 (=A) to SU_PAR10 (=J). If global values should be used, A to J should initialized manually by the expression list.

  Attention: using the expression parser is extremely time consuming. The overall model performance will going down to below 50%. It is recommended for site modelling only (one cell or a small number of cells) – or you have a really good computer to run a real watershed size model.