

77A

SMP Library

September 1982

XAbs

Abs simplification rules

absolute value - modulus - numerical value

S.Wolfram

Jul 1981

```

Abs[$x $$x] :: Abs[$x] Abs[$$x]
Abs[$x^( $n_>Natp[$n])] : Abs[$x]^$n
D[Abs[$x], {$x, 1, $y}] : Sign[$y]
Abs[Sign[$x_ ($x~0)]] : 1

```

~~Abs[\$x] ^ (\$n_ = Evenp[\$n]) ;~~

• Theta; Sign; Ramp; Delta

```

#I[1]:: <XAbs
#I[2]:: Abs[a b^2 c]
#O[2]: Abs[a] Abs[b]^2 Abs[c]

```

Warning: Redefines Abs.

XAck

XAck

Ackermann function

generalized power - general recursion - recursive function theory

C.Cole and S.Wolfram

Jul 1981

Ack [x, y]

An Ackermann function.

```

Ack[0, $y] : Mod[$y+1, 3]
Ack[$x, 0] : $x+1
Ack[$x, 1] : $x+2
Ack[$x, 2] : 2*$x
Ack[$x, 3] : 2^$x
Ack[$x, 4] :: Arrow[2, $x+1]
Ack[$x, $y] :: Ack[$x, $y] : Ack[Ack[$x-1, $y], $y-1]

```

Arrow [n, m]

Knuth arrow function $n^{(n^{(n^{\dots}))})}$ with m powers.

```

Arrow[$n, 0] : 1
Arrow[$n, 1] : $n
Arrow[$n, $m] :: $n^Arrow[$n, $m-1]

```

```

#I[1]:: <XAck
#I[2]:: Ar[10, Arrow[$i, 2]]
#O[2]:: {1, 4, 27, 256, 3125, 46656, 823543, 16777220, 387420500, 1.*~10}
#I[3]:: Ack[2, 5]
#O[3]:: 16
#I[4]:: Ack[3, 5]

```

XAllbut

XAllbut

List element deletion

list element removal - complement - exclusion

S.Wolfram

Aug 1982

Allbut [*list*, *e1*, *e2*, ...]

yields *list* with all occurrences of *e1*, *e2*, ... deleted.

```
Allbut[$list,$se] := {Lcl[t]; t:$list; Do[i,Len[List[$se]], \
t:Del[List[$se][i],t]; t}
```

Misfeatures: untested

XAny

XAny

List element condition test

scan - any - test for any elements - existence test

S.Wolfram
Jul 1981

Any [*temp*, *list*]

tests whether any of the elements of *list* yield "true" on application of the template *temp*.

```
Any_Tier
_Any[Smp] : {0, Inf}
Any[$temp, $list] :: In[$1 -> Ap[$temp, {$1}], $list]
```

• Pos; XScan

```
#I[1]:: <XAny
#I[2]:: Any[Evenp, Ar[5, Prime]]
#O[2]: 1
```


XArperm

Permutation generation

reorderings - symmetries

S.Wolfram

Jul 1981

Arperm[*n*, (*spec*: (*all*))]yields a list of the permutations of *n* elements which exhibit the symmetries *spec*.**Arperm**[*n*] :: Flat[Ar[Ar[*n*,*n*],List,Uneq],*n*]

```

<XList8
Arperm_Tier
Arperm[1]::{{1}}
Arperm[n,Natp[n]]::Arperm[n]:\
    Flat[Map[Ar[n,Ins[n,s1,s2]],Arperm[n-1]],1]
Arperm[n,Cyclic]::Ar[n,Cyc[Ar[n],s1]]
Arperm[n,Even]::Cat[Ar[n],Arperm[n],Evenp]
Arperm[n,Odd]::Cat[Ar[n!],Arperm[n],Oddp]

```

#I[1]:: <XArperm

#I[2]:: Arperm[3]

#O[2]:: {{3,2,1},{2,3,1},{2,1,3},{3,1,2},{1,3,2},{1,2,3}}

#I[3]:: Arperm[3,Cyclic]

#O[3]:: {{2,3,1},{3,1,2},{1,2,3}}

#I[4]:: Arperm[3,Even]

#O[4]:: {{2,3,1},{3,1,2},{1,2,3}}

#I[5]:: Arperm[3,Odd]

#O[5]:: {{3,2,1},{2,1,3},{1,3,2}}

Prerequisites: XList8

XBase

XBase

Number base conversion

radix arithmetic - positional notation - binary - ternary
octal - hexadecimal - scales of notation - integer conversion

S.Wolfram

Jul 1981

To10[cccc, n]

converts the number cccc from base n to base 10. cccc represents an integer whose digits are characters in the symbol name cccc. The "digit" 10 is represented by a, 11 by b and so on.

```
To10_Tier
To10[ $S_s$  -> Symbp[ $S_s$ ],  $S_b$  -> Natp[ $S_b$ ]] := (LcI[Xi]; Xi:ExpI[ $S_s$ ]; \
Sum[ $S_b^{(Len[Xi]-Xi)} * Xi[Xi]$ , {Xi, 1, Len[Xi]}])
```

From10[x, n]

converts the decimal integer x to a Base projection in base n.

```
From10_Tier
From10[ $S_n$  -> Natp[ $S_n$ ],  $S_b$  -> Natp[ $S_b-1$ ]] := (LcI[Xtot, Xres, Xi]; \
For[Xi:1; Xres: $S_n$ , Xres~0, Inc[Xi], Xtot[Xi]:Mod[Xres,  $S_b$ ]; \
Xres:Floor[Xres/ $S_b$ ]; ImpI[Rev[Xtot]])
```

• XTern

- #I[1]:: <XBase
- #I[2]:: From10[1452, 2]
- #O[2]:: "10110101100"
- #I[3]:: To10[X, 2]
- #O[3]:: 1452
- #I[4]:: From10[X, 16]
- #O[4]:: "5ac"
- #I[5]:: To10[X, 16]
- #O[5]:: 1452

XBell

XBell

Bell numbers

Stirling numbers – number of equivalence relations
combinatorial functions

S.Wolfram
Sep 1982

Bell[n]

n th Bell number.

`Bell[$n_?Natp[$n]] :: Sum[Sti2[$n,xi], {xi,1,$n}]`

[Sloane: Handbook of Integer Sequences, sect. 3.12]

XBer2V

XBer2V

Bernoulli polynomials

S.Wolfram
Feb 1982

Definition of values for integer index.

```
<XBerV  
Ber[ $n$ ,  $x$ ] := Sum[Ber[k] Comb[ $n$ ,  $k$ ]  $x^{(n-k)}$ , {k, 0,  $n$ }]
```

[MOS sect. 1.5.1]

XBerV

XBerV

Bernoulli numbers

S.Wolfram
Feb 1982

Definitions for special values of argument.

`Ber[0] : 1`

`Ber[1] : -1/2`

`Ber[2] : 1/6`

`Ber[4] : -1/30`

`Ber[m , Oddp[m]] : 0`

`Ber[m , Natp[$m/2$]] := (Lc[{ x , x }; x :1; x :0; Do[x , 0, $m-1$, x : $x+x$ Ber[x]; \`
 `x : x ($m+1-x$)/($x+1$); Ber[m]:- x /($m+1$))`

[AS 23.1.7]

XBit

Bitwise operations

binary - bitwise and - bitwise or - collate

S.Wolfram

Jul 1981

Bits[n]yields a list of binary bits corresponding to the integer n .

```
Bits_Tier
Bits[ $n$  Natp[ $n$ ]] :: (Lcl[Xtot, Xres, Xi]; Xi:1; Xres: $n$ ; \
  Loop[Xres~0, Xtot[Xi]:Mod[Xres,2]; Xres:Floor[Xres/2]; Inc[Xi]]; \
  Rev[Xtot])
```

Intbit[list]

finds the integer corresponding to a list of bits.

```
Intbit[ $list$  Contp[ $list$ ]] :: \
  Sum[2~(Len[ $list$ ]-Xi)  $list$ [Xi], {Xi,1,Len[ $list$ ]}]
```

Bitand[n, m]yields the bitwise conjunction of n and m .

```
Bitand_Comm
Bitand[ $n$  Natp[ $n$ ],  $m$  Natp[ $m$ ] &  $m$ > $n$ ] :: \
  (Lcl[Xn, Xm]; Xn:Bits[ $n$ ]; Xm:Bits[ $m$ ]; \
  Intbit[Rev[Ldist[Rev[Xn] & Ar[Len[Xn], Rev[Xm]]]])
Bitand[ $n$  Natp[ $n$ ],  $n$ ] ::  $n$ 
```

Bitor[n, m]yields the bitwise disjunction of n and m .

```
Bitor_Comm
Bitor[ $n$  Natp[ $n$ ],  $m$  Natp[ $m$ ]] :: \
  Intbit[Rev[Ldist[Rev[Bits[ $n$ ]] | Rev[Bits[ $m$ ]]]]]
```

```
#I[1]:: <XBit
#I[2]:: Bits[123]
#O[2]:: {1,1,1,1,0,1,1}
#I[3]:: Intbit[X]
#O[3]:: 123
#I[4]:: Bitand[123,514]
#O[4]:: 4
#I[5]:: Bitor[123,514]
#O[5]:: 635
```

XBox

Additional graphical objects

geometrical figures - square - box - circle - plotting
regions

S. Wolfram

Jul 1981

Box[{x,y}]

represents a unit box centred at the point x,y .

```
Box[{x,y}] := Line[{Pt[{x-0.5,y-0.5}],Pt[{x+0.5,y-0.5}], \
Pt[{x+0.5,y+0.5}],Pt[{x-0.5,y+0.5}], \
Pt[{x-0.5,y-0.5}]}]
```

Circle[{x,y}]

represents a unit circle centred at the point x,y .

```
Circle[{x,y}] := {Curve[Arc[20,Pt[{x,y}]+{Sin[2Pi $1/20], \
Cos[2Pi $1/20]}]]}]
```


XCatalan

XCatalan

Catalan numbers

convex polygon dissection – rooted planar trees

S.Wolfram

Sep 1982

Catnum [n]

n th Catalan number.

`Catnum[n] :: Comb[2n, n] / (n+1)`

[Sloane: Handbook of Integer Sequences, sect. 3.5]

XCf

Continued Fraction Numbers

continued fraction representation

S.Wolfram

Aug 1982

CfD[{a1, a2, ...}]converts the continued fraction number with coefficients *a1*, *a2*, ... to decimal form.

```
CfD[$list_>Contp[$list]] := (Lc[t]; t:Last[$list]; \
Do[1, Len[$list]-1, 1, -1, t:N[1/t+$list[[1]]]; t)
```

DCf[n, ord]converts the decimal number *n* to a continued fraction form to order *ord*.

```
DCf[{$n, $ord_>Natp[$ord]] := \
(Lc[n, ni, list]; n:$n; list: {}; Do[1, $ord, ni:Floor[n]; \
list:Cat[list, {ni}]; n:N[1/(n-ni)]]; list)
```

#I[1]:: <XCf

#I[2]:: N[Phi]

#D[2]: 1.61803

#I[3]:: DCf[X, 10]

#D[3]: {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}

#I[4]:: CfD[X]

#D[4]: 1.61818

#I[5]:: DCf[N[Pi], 10]

#D[5]: {3, 7, 15, 1, 292, 1, 1, 1, 2, 1}

Future enhancements: Treat arbitrary precision numbers; may be included in Cf[] internal code.

XChar

Character manipulation

text manipulation - character strings - ASCII - letters
words

S.Wolfram

Jul 1981

Upperp [str]

yields 1 if the first character of the string *str* is an upper case letter, and 0 otherwise.

```
Upperp[$str] :: (!Exp1[A][1]) <= Exp1[$str][1] <= (!Exp1[Z][1])
```

Lowerp [str]

yields 1 if the first character of the string *str* is a lower case letter, and 0 otherwise.

```
Lowerp[$str] :: (!Exp1[a][1]) <= Exp1[$str][1] <= (!Exp1[z][1])
```

Tolower [str]

converts all upper case letters in *str* to lower case.

```
Tolower[$str] :: Impl[Map[$1-(!Exp1[A][1]-Exp1[a][1]),Exp1[$str],1, \
(!Exp1[A][1]) <= $2 <= (!Exp1[Z][1])]]
```

```
#I[1]:: <XChar
```

```
#I[2]:: t1:"a message"
```

```
#O[2]:: "a message"
```

```
#I[3]:: t2:"A MESSAGE"
```

```
#O[3]:: "A MESSAGE"
```

```
#I[4]:: Lowerp[t1]
```

```
#O[4]:: 1
```

```
#I[5]:: Lowerp[t2]
```

```
#O[5]:: 0
```

```
#I[6]:: Tolower[t2]
```

```
#O[6]:: "a message"
```

XChi2

XChi2

Chi squared distribution

probability functions - errors - statistics - normal distribution
Gaussian distribution

S. Wolfram

Aug 1982

QChi2 [*chi2*, *nu*]

value of chi squared distribution at *chi2* with *nu* degrees of freedom.

$$\text{QChi2}[\{\text{\$chi2}, \text{\$nu}\}] :: \text{N}[\text{Gamma}[\text{\$nu}/2, \text{\$chi2}/2] / \text{Gamma}[\text{\$nu}/2]]$$

[AS 26.4.1]

XClass

Classified data statistics

binned data - histograms

C.Feynman
Aug 1981

Class [*datum*, *cent*, *spacing*]
returns the number closest to *datum* which is equal to *cent* modulo *spacing*. This is the class mark (the location of the center of the class) of the class in which *datum* belongs, given an arrangement of classes of constant width (equal to *spacing*), one of which is centered on *cent*.

```
Class($x,$cent,$space) : $cent + -1*(($space*Floor[1/2 + ($cent + -$x)\
/$space])
```

Classify [*d*, *cent*, *space*]
returns a list, the indices of which are the class marks of a set of classes whose spacing is *space*, and one of which is centered on *cent*. The elements of the list are the frequencies with which the numbers in the list *d* fall into the respective classes. The returned list is the smallest which will include all of *d*.

```
Classify($data,$c,$s) :: Proc[Lcl[rsit] ; rsit : Ar[{{Class[Ap[Min\
,$data],$c,$s],Class[Ap[Max,$data],$c,$s]},0} ; Map[rsit[Class\
[$y,$c,$s]] : rsit[Class[$y,$c,$s]] + 1,$data]\
; rsit]
```

CMode [*d*]
returns a list consisting of the modes of the data in *d*, i. e. the class mark or marks of the most crowded class or classes.

```
CMode[$data] :: Ap[Cat,Pos[Ap[Max,$data],$data]]
```

CMean [*d*]
returns the arithmetic mean of the data in *d*.

```
CMean[$data] :: N[Sum[Elem[$data,{n}]*Ind[$data,n],{n,1,\
Len[$data]}]/Ap[Plus,$data]]
```

CSD [*d*]
returns the standard deviation of the data in *d*.

```
CSD[$d] :: N[Sqrt[CVar[$d]*Ap[Plus,$d]/(Ap[Plus,$d] + -1)]]
```

CVar [*d*]
returns the variance of the data in *d*.

```
CVar[$d] :: N[(Sum[Ind[$d,n]^2*Elem[$d,{n}],{n,1,\
Len[$d]}] + -1*(Ap[Plus,$d]*CMean[$d]^2))/Ap[Plus,$d]]
```

CApprox [*d*]
returns an Err projection whose mean and standard deviation are the same as those of *d*.


```
CApprox[$d] :: Err[CMean[$d],CSD[$d]]
```

CMD[d]

returns the mean absolute deviation of the data in *d*.

```
CMD[$d] :: (Lcl[mean] ; mean : CMean[$d] ; Sum[Abs[Ind[$d,n] + -mean] \
  eElem[$d, {n}], {n,1,Len[$d]}) / Ap[Plus,$d]
```

Unclassify[d]

returns a list of unclassified data, which, if classified, would produce *d*. This is the "inverse" of **Classify**. It is of course not perfect, since much information is lost in the classification process.

```
Unclassify[$d] :: Purify[Ap[List,Map[Repl[Ind[$d,$x],Elem[$d,{$x}\
  ]],List[1..Len[$d]]]]]
Purify[$list] :: (Lcl[r]; r: {} ; Do[i,Len[$list],If[Numbp[Elem[$list,{i}]], \
  r:Cat[r,List[Elem[$list,{i}]]]]];r)
```

CFrac1[d,f]

is a projection which is called only by **CFract**. You should never need it.

```
CFrac1[$d,$frac] :: If[$frac > 0,Lcl[n,total,wanted] ; \
  For[n : 0 ; total : 0 ; wanted : $fracAp[Plus,$d],wanted > total, \
  n : n + 1 ; total : total + Elem[$d,{n}],n + 1,Lcl[n] ; n \
  : 0 ; Loop[Null,n : n + 1,Elem[$d,{n}] = 0]]
```

CFract[d,f]

returns the *f* fractile of the data in *d*.

```
CFract[$d f (Len[$d] > 1), $frac f ($frac >= 0 & 1 >= $frac)] :: \
  NI((Lcl[where,c,bl,fb,fb1,nn] ; where : CFract[$d,$frac] ; c : Ind[$d, \
  2] + -Ind[$d,1] ; bl : Ind[$d,where] + -1ec/2 ; fb \
  : Elem[$d,{where}] ; fb1 : Ap[Plus,Map[Elem[$d,{$n}], \
  {1..where + -1}]] ; nn : Ap[Plus,$d] ; \
  bl + ce(nne$frac + -fb1)/fb))
```

CMed[d]

returns the median of the data in *d*.

```
CMed[$d] :: CFract[$d,0.5]
```

CQ1[d]

returns the first quartile of the data in *d*.

```
CQ1[$d] :: CFract[$d,0.25]
```

CQ3[d]

returns the third quartile of the data in *d*.

```
CQ3[$d] :: CFract[$d,0.75]
```

CMin[d]

returns the minimum of the data in *d*.

```
CMin[$d] :: (Lcl[n] ; n : 0 ; Ind[$d,Loop[Null,n : n + 1,Elem[$d, \
  {n}] = 0]])
```

CMax[d]

returns the maximum of the data in *d*.


```
CMax[$d] := (Lc[n] ; n : Len[$d] + 1 ; Ind[$d, Loop[Null, n : n + -1, \
Elem[$d, {n}] = 0]])
```

```
#I[1]:: <xclass
#I[2]:: Class[2.43, 8, 1]
#O[2]: 2
#I[3]:: Class[2.43, .5, 1]
#O[3]: 2.5
#I[4]:: Classify[{22, 15, 35, 45, 36, 25, 16, 35, 29, 38, 28, 45, 36, 33, 20, 25, 15, 9, 48, 42, 4},
#O[4]: {[7.5]: 1, [12.5]: 2, [17.5]: 2, [22.5]: 3, [27.5]: 2, [32.5]: 3,
[37.5]: 3, [42.5]: 4, [47.5]: 1}
```

At this point, due to a garbage collector bug, SMP went into an infinite loop. We thus have to start over:

```
#I[1]:: <exampl
This file contains an example of classified data. It is the value of the variable a.
#I[2]:: <xclass
#I[3]:: a
#O[3]: {[ -3]: 1, [ -1]: 2, [ 1]: 0, [ 3]: 2, [ 5]: 4, [ 7]: 6, [ 9]: 5, [11]: 6,
[13]: 4, [15]: 2, [17]: 1, [19]: 1}
#I[4]:: CMode[a]
#O[4]: {7, 11}
#I[5]:: CMean[a]
#O[5]: 8.588235
#I[6]:: CSD[a]
#O[6]: 4.991615
#I[7]:: Unclassify[a]
#O[7]: {-3, -1, -1, 3, 3, 5, 5, 5, 5, 7, 7, 7, 7, 7, 7, 9, 9, 9, 9, 9, 11, 11, 11, 11, 11, 11, 13, 13,
13, 13, 15, 15, 17, 19}
```

XCode

Mixed radix operations

APL encode/decode

S.Wolfram

Jan 1982

Encode [radix, list]encodes *list* according to the specified mixed *radix*.

```

Encode[Radix, Listp[Radix], $list, (Listp[$list] \
&Len[$list]=Len[Radix]+1)] :: \
(Lcl[Xtot, Xi]; Xtot:$list[1]; Do[Xi, 2, Len[$list], \
Xtot:Xtot Radix[Xi-1]+$list[Xi]; Xtot)

```

Decode [radix, n]yields the number *n* in the specified mixed *radix*.

```

_Decode[Init] :: <XTri
Decode[Radix, Listp[Radix], $n, Natp[$n]] :: \
(Lcl[Xtot, Xdig, Xrad, Xi]; Xrad:Rev[Map[Ap[Mult, $1], Tri[Radix]]]; \
Xtot:$n; Do[Xi, Len[Radix], Xtot:Xtot-(Xdig[Xi]: \
Gint[Xtot/Xrad[Xi]])Xrad[Xi]; Cat[Rev[Xdig], {Xtot}])

```

Misfeatures: Decode definition is incorrect

Decode definition not yet correct.

Example:

```

time: {365, 24, 60}
Encode[time, {4, 2, 1, 5}]
Decode[time, %]

```


XCon

Tensor contraction

explicit tensors - inner products - generalized traces

S.Wolfram

Jul 1981

Con[list, ni, nj]

forms the contraction of the tensor list over its *nith* and *njth* indices.

```

Con[Slist, $ni_>Natp[$ni], $nj_>Natp[$nj]] := \
  (Lc1[Xt]; Xt:Trans[Trans[Slist, {1, $ni}], {2, $nj}]; \
  Rp[Plus, Ar[Len[Xt], Xt[$1, $1]])

```

```

#I[1]:: <XCon
#I[2]:: w3:ArI{2,2,2}, f]
#O[2]:  {{{f[1,1,1], f[1,1,2]}, {f[1,2,1], f[1,2,2]}},
        {{f[2,1,1], f[2,1,2]}, {f[2,2,1], f[2,2,2]}}}
#I[3]:: Con[X,1,2]
#O[3]:  {f[1,1,1] + f[2,2,1], f[1,1,2] + f[2,2,2]}
#I[4]:: Con[w3,2,3]
#O[4]:  {f[1,1,1] + f[1,2,2], f[2,1,1] + f[2,2,2]}
#I[5]:: w4:ArI{2,2,2,2}, f];
#I[6]:: Con[w4,2,4]
#O[6]:  {{f[1,1,1,1] + f[1,2,1,2], f[2,1,1,1] + f[2,2,1,2]},
        {f[1,1,2,1] + f[1,2,2,2], f[2,1,2,1] + f[2,2,2,2]}}

```

• Inner; Outer

XConfus

XConfus

Confusing function

S.Wolfram
Jul 1981

Conf [x]

is a confusing function.

```
Conf[$x, Nump[$x]] :: Conf[$x]:Rand[]
```

XConsSol

Numerical solution of equations

Newton's method - numerical inversion

T.Shaw

Feb 1982

ConsSol [*f*, *e1=e2*, *v*, {*\$\$args*}]

creates a top level smp function $f[*$$args*, *guess1*, *guess2*, *acc*]$ which returns the value of *v* (a variable in the *e1* or *e2*) which solves the equation. The *\$\$args* are parameters to the equation. They must be generic symbols. When given to the consed function, all parameters must be numeric. This function may then be consed, for increased efficiency.

```
_ConsSol[Smp] : 0
ConsSol - Tier
```

```
ConsSol[$f_Symbp[$f], $e1=$e2, $v_Symbp[$v]] :: \
  ConsSol[$f, $e1=$e2, $v, {}]
ConsSol[$f_Symbp[$f], $e1=$e2, $v_Symbp[$v], $args_Listp[$args]] :: \
  (Ap[Set, {Proj[$f, Cat[$args, {$g1, $g2, $acc}]}], \
    Ap[' ( Lcl[x1, x2, y1, y2] ; x1 : $g1 ; y1 : $g2 ; \
      y1 : $0 ; y2 : $1 ; \
      Loop[ fabs[y2 - y1] > fabs[$acc y2], Lcl[tmp] ; \
        tmp : x2 - y2 * ((x2-x1)/(y2-y1)) ; \
        x1:x2 ; y1:y2 ; x2:tmp ; y2:$2 ] ; \
      x2 ), \
    {S[$e1-$e2, $v->$g1], S[$e1-$e2, $v->$g2], S[$e1-$e2, $v->tmp] } ] ; $f)
```

```
#I[1]:: <XConsSol
#I[2]:: ConsSol[f, Cos[x] = $y x, x, {$y}]
#O[2]: ' f
#I[3]:: Cons[f]
#O[3]: {' f}
-#I[4]:: f[1, .5, 1.5, .0001]
#O[4]: 0.739085
#I[5]:: N[Cos[X]]
#O[5]: 0.739085
#I[6]:: f[5, .1, .9, .0001]
#O[6]: 0.196164
#I[7]:: N[Cos[X]]
#O[7]: 0.988821
#I[8]:: 5e06
#O[8]: 0.988821
```


XContig

Contiguous list generation

pad - fill - make rectangular - make square - make cubical
fill holes

S.Wolfram
Jul 1981

Contig[list, {n1, n2, ...}, elem]

renders list contiguous with *ni* entries at level *i* by inserting *elem* where necessary.

```
Contig[$list_>Listp[$list], $n_>Listp[$n], $elem] := \
  Contig0[$list, $n, $elem, 1]
Contig0[$list, $n, $elem, $lev_>$lev > Len[$n]] : $list
Contig0[$list_>~Listp[$list], $n, $elem, $lev] : $list
Contig0[$list_>Listp[$list], $n, $elem, $lev] := \
  Ar[$n[$lev], If[P[Proj[$list[$1], {0}]]=Proj], \
    $elem, Contig0[$list[$1], $n, $elem, $lev+1]]

#I[1]:: <XContig
#I[2]:: t: { [3]: a, [2]: b }
#O[2]: { [3]: a, [2]: b }
#I[3]:: Contig[t, {4}, 0]
#O[3]: { 0, b, a, 0 }
#I[4]:: Contig[t, {8}, x]
#O[4]: { x, b, a, x, x, x, x, x }
#I[5]:: Ar[{{2, 3}, {3, 4}}, f]
#O[5]: { [2]: { [3]: f[2, 3], [4]: f[2, 4] }, [3]: { [3]: f[3, 3], [4]: f[3, 4] } }
#I[6]:: Contig[X, {4, 4}, 0]
#O[6]: { 0, { 0, 0, f[2, 3], f[2, 4] }, { 0, 0, f[3, 3], f[3, 4] }, 0 }
```


Derivatives

S.Wolfram

Jul 1981

```

D[Ber[Sn, Sz], {Sz, Sm, Sx}]      :: Sn!/(Sn-Sm)!eBer[Sn-Sm, Sx]
D[Beta[Sx, Sy, 1], {Sx, 1, Sz}]   :: Beta[Sz, Sy, 1] (Psi[Sz] - Psi[Sz+Sy])
D[Beta[Sx, Sy, 1], {Sy, 1, Sz}]   :: Beta[Sx, Sz, 1] (Psi[Sz] - Psi[Sx+Sz])
D[CheT[Sn, Sz], {Sz, 1, Sx}]      :: Sn CheU[Sn-1, Sx]
D[CheT[Sn, Sz], {Sz, Sm, Sx}]     :: 2^(Sm-1) Gamma[Sm] Sn Geg[Sn-Sm, Sm, Sx]
D[CheU[Sn, Sz], {Sz, Sm, Sx}]     :: 2^Sm Sm!eGeg[Sn-Sm, Sm+1, Sx]
D[Chg[Sa, Sc, Sz], {Sz, Sm, Sx}]  :: \
    Poch[Sa, Sm] Chg[Sa+Sm, Sc+Sm, Sx] / Poch[Sc, Sm]
D[Coshi[Sz], {Sz, 1, Sx}]         :: Cosh[Sx]/Sx
D[Cos[Sz], {Sz, 1, Sx}]           :: Cos[Sx]/Sx
D[Ei[Sz], {Sz, 1, Sx}]            :: Exp[Sx]/Sx
D[EiIK[Sk, St], {St, 1, Su}]      :: (1-Sk^2 Sin[Su]^2) ^ (-1/2)
D[EiIE[Sk, St], {St, 1, Su}]     :: Sqrt[1-Sk Sin[Su]^2]
D[Erf[Sz], {Sz, 1, Sx}]           :: 2 Exp[-Sx^2] / Sqrt[Pi]
D[Erf[Sz], {Sz, Sm, Sx}]          :: -2 (-1)^Sm Exp[-Sx^2] Her[Sm-1, Sx]
D[Eul[Sn, Sz], {Sz, Sm, Sx}]      :: Sn!/(Sn-Sm)!eEul[Sn-Sm, Sx]
D[ExpI[1, Sz], {Sz, Sm, Sx}]      :: (-1)^Sm Exp[-Sx] Chg[1, 1+Sm, Sx]
D[ExpI[Sn, Sz], {Sz, 1, Sx}]      :: -ExpI[Sn-1, Sx]
D[FreC[Sz], {Sz, 1, Sx}]          :: Cos[Pi Sx^2 / 2]
D[FreS[Sz], {Sz, 1, Sx}]          :: Sin[Pi Sx^2 / 2]
D[Gamma[Sz], {Sz, 1, Sx}]         :: Gamma[Sx] Psi[Sx, 1]
D[Gamma[Sz, Sa], {Sz, 1, Sx}]     :: -Sa^(Sx-1) Exp[-Sa]
D[Geg[Sn, S1, Sz], {Sz, Sm, Sx}]  :: 2^Sm Poch[S1, Sm] Geg[Sn-Sm, S1+Sm, Sx]
D[Her[Sn, Sz], {Sz, Sm, Sx}]      :: 2^Sm Sn!/(Sn-Sm)!eHer[Sn-Sm, Sx]
D[Hg[Sa, Sb, Sc, Sz], {Sz, Sm, Sx}] :: \
    Poch[Sa, Sm] Poch[Sb, Sm] Hg[Sa+Sm, Sb+Sm, Sc+Sm, Sx] / Poch[Sc, Sm]
D[JacP[Sn, Sa, Sb, Sx], {Sz, Sm, Sx}] :: \
    Poch[Sa+Sb+Sc+1, Sm] JacP[Sn-Sm, Sa+Sm, Sb+Sm, Sx] / 2^Sm
D[JacCd[Sz, Sm], {Sz, 1, Sx}]     :: (Sm-1) JacSd[Sx, Sm] JacNd[Sx, Sm]
D[JacCn[Sz, Sm], {Sz, 1, Sx}]     :: -JacSn[Sx, Sm] JacDn[Sx, Sm]
D[JacCs[Sz, Sm], {Sz, 1, Sx}]     :: -JacNs[Sx, Sm] JacDs[Sx, Sm]
D[JacDc[Sz, Sm], {Sz, 1, Sx}]     :: (1-Sm) JacSc[Sx, Sm] JacNc[Sx, Sm]
D[JacDn[Sz, Sm], {Sz, 1, Sx}]     :: -Sm JacSn[Sx, Sm] JacCn[Sx, Sm]
D[JacDs[Sz, Sm], {Sz, 1, Sx}]     :: -JacCs[Sx, Sm] JacNs[Sx, Sm]
D[JacDc[Sz, Sm], {Sz, 1, Sx}]     :: JacSc[Sx, Sm] JacDc[Sx, Sm]
D[JacDn[Sz, Sm], {Sz, 1, Sx}]     :: Sm JacSd[Sx, Sm] JacCd[Sx, Sm]
D[JacDs[Sz, Sm], {Sz, 1, Sx}]     :: -JacDs[Sx, Sm] JacCs[Sx, Sm]
D[JacSc[Sz, Sm], {Sz, 1, Sx}]     :: JacDc[Sx, Sm] JacNc[Sx, Sm]
D[JacSd[Sz, Sm], {Sz, 1, Sx}]     :: JacCd[Sx, Sm] JacNd[Sx, Sm]
D[JacSn[Sz, Sm], {Sz, 1, Sx}]     :: JacCn[Sx, Sm] JacDn[Sx, Sm]
D[JacZ[Sz, Sm], {Sz, 1, Sx}]     :: JacDn[Sx, Sm]^2

```

D[KumU[a, c, z], { z, m, x }]	:: $(-1)^{-m} \text{Poch}[a, m] \text{KumU}[a+m, c+m, x]$
D[Lag[n, a, z], { $z, 1, x$ }]	:: $-\text{Lag}[1-n, 1+a, x]$
D[LegP[n, z], { $z, 1, x$ }]	:: $n (\text{LegP}[n-1, x] - x \text{LegP}[n, x]) / (1-x^2)$
D[Li[n, z], { $z, 1, x$ }]	:: $\text{Li}[n-1, x] / x$
D[Lob[z], { $z, 1, x$ }]	:: $\text{Log}[\text{Sec}[x]]$
D[Logi[z], { $z, 1, x$ }]	:: $1 / \text{Log}[x]$
D[Psi[z], { $z, 1, x$ }]	:: $\text{Psi}[x, 2]$
D[Psi[z, n], { $z, 1, x$ }]	:: $\text{Psi}[x, n+1]$
D[Sinh[z], { $z, 1, x$ }]	:: $\text{Sinh}[x] / x$
D[Sini[z], { $z, 1, x$ }]	:: $\text{Sin}[x] / x$
D[Zeta[z, a], { $a, 1, b$ }]	:: $-z \text{Zeta}[z+1, b]$

XDSol

XDSol

Series solution of differential equations

power series - Frobenius method

J.Greif and S.Wolfram

Oct 1981

DSol[eqn, y, x, ord, {y[0], y'[0], ...}]

gives a series solution to the ordinary differential equation eqn with dependent variable y and independent variable x and specified boundary conditions, accurate to order x^ord.

* XSerSol

```

DSol[Seqn1=Seqn2,$f,$x,$ord,$init] := \
(Lc1[Xa,Xeqn,Xlist,Xf]; Xf:=Sum[Xa[[i]] $x^i, {i,0,$ord}]; \
Xeqn:=S[Ex[S[Seqn1-Seqn2,$f->Xf]], $x^i_>($i>$ord)->0]; \
Xlist:=Union[Cat[Ar[$ord-1,Coef[$x^$1,Xeqn]], {S[Xeqn,$x->0]}], \
Ar[Len[$init],D[Xf,{x,$1-1,0}]-$init[$1]]]; \
Xlist:=Map[$2=0,Xlist]; \
Xlist:=Sol[Xlist,Ar[{{0,$ord}},Xa[$1]]]; \
S[Xf,Xlist])

```

#I[1]:: <XDSol

#I[2]:: eq:Dt[y,x]+a y=0

#O[2]: Dt[y,x] + a y = 0

#I[3]:: DSol[X,y,x,2,{1}]

$$\#O[3]: 1 - a x + \frac{a^2 x^2}{2}$$

XDap

XDap

Directional application

mapping - rotate - APL - column operations

S.Wolfram

Jul 1981

Dap [*f*, *list*, *n*]

applies the template *f* to "columns" of elements at level *n* in *list*.

```
Dap[$f,$list→Listp[$list],$n→(Natp[$n] & $n>1)] :: \  
    Trans[Map[Ap[$f,$x1],Trans[$list,$n]],$n-1]  
Dap[$f,$list→Listp[$list],1] :: Ap[$f,$list]
```

XData

XData

Data input

columnated input - data

S.Wolfram
Jul 1981

Data[file, (nline)]

reads data from file for nline lines or until the end of file is encountered, taking columns to be separated by tabs, single or double spaces.

```

Data_Tier
Data[$file] := Data[$file, 100000]
Data[$file, $nline_?Natp[$nline]] := \
  (Lcl[Xi, Xn, Xr]; Sxset[" ", List, 3]; Sxset[" ", List, 3]; \
  Sxset[" ", List, 3]; For[Xi: {}, Xn: 1, Xr: Rd[, {$file, Xn}]; \
  (Numbp[Xr] | Listp[Xr]) \
  & Xn <= $nline, Inc[Xn], Xi: Cat[Xi, {Xr}]]; " " := ; \
  " := ; " := ; Ret[Xi]

```


XData1

Data input with conversion

columnated input - data conversion

J.Greif

Aug 1982

Cdata[*file*, (*cols*: *Inf*), (*temp*)]

reads data from file *file* using the SMP input filter *sm pin* to convert Fortran E-format numbers, if present, to SMP *^ notation, and collect the data into a list of lists, each containing *cols* elements taken sequentially from the input. The template *temp* is applied to each sublist.

```

Cdata_ Tier
_Cdata[Smp]: {Inf, Inf, 0}
_Cdata[Init]: (<UnFlat; <XStr0)
Cdata[$file]: Cdata[$file, 1000000]
Cdata[$file, , $temp]: Cdata[$file, 100000, $temp]
Cdata[$file, $cols]: UnFlat[Run[CJoin["sm pin -mklist >&4 <", $file]], \
    $cols]
Cdata[$file, $cols, $temp]: Map[$temp, Cdata[$file, $cols]]

```

- Cdat [\$file, (\$cols:\$Inf), (\$#temp)]

XDiff

Finite differences

Forward differences - difference equations - finite elements

S.Wolfram

Jul 1981

Diff[f, x, x0, n]yields the *n*th forward finite difference of *f* with respect to *x* at the point $x=x_0$.
$$\text{Diff}[f, x, x_0, n] :: \text{Sum}[S[f, x \rightarrow x_0 + n - r] \setminus (-1)^r \text{Comb}[n, r], \{r, 0, n\}]$$

#I[1]:: <XDiff

#I[2]:: Diff[f[x], x, 0, 3]

#O[2]:: -f[0] + 3f[1] - 3f[2] + f[3]

#I[3]:: Ar[5, Ex[Diff[(x+a)^3, x, 1, \$1]]]

#O[3]:: {7 + 9a + 3 a², 12 + 6a, 6, 0, 0}

XDig

Digit manipulation

text manipulation – positional notation – digit extraction
number construction

S.Wolfram

Jul 1981

Updated Aug 1982

Dig[n,i]

yields the coefficient of 10^i in the number n .

```
Dig[$n_>Nump[$n], $i_>Intp[$i]] :: Mod[Floor[$n/10^$i], 10]
```

LDig[n]

yields a list of the digits in the integer n .

```
LDig[$n_>Nump[$n]] :: Expl[$n]
_F[Extr, 'LDig'] :: FLDig
_FLDig[init] :: <XPad
FLDig['F[$$x, $i1, $i2]] :: Flat[Map[LPad[Expl[$i], 8, 4], List[$$x]]]
```

NMake[list]

converts a list of digits to an integer.

```
NMake[$list] :: Make[Impl[$list]]
```

```
#I[1]:: <XDig
#I[2]:: Dig[456123.321, 4]
#O[2]: 5
#I[3]:: Dig[114.00425, -3]
#O[3]: 4
#I[4]:: LDig[1467120]
#O[4]: {1, 4, 6, 7, 1, 2, 0}
#I[5]:: NMake[X]
#O[5]: 1467120
#I[6]:: N[Pi, 20]
#O[6]:* (3.1415926535897932385)
#I[7]:: LDig[X]
#O[7]: {0, 0, 0, 3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 0, 3, 2, 3, 8, 5, 0}
```

Prerequisites: XPad

XDim

Dimensional analysis

units - physical quantities - similarity

S.Wolfram

Jul 1981

Fundamental dimensions:

- length
- mass
- time
- current
- temperature
- (luminous) intensity
- amount (of substance)

Derived dimensions

- SDim[1] : area -> length²
- SDim[2] : volume -> length³
- SDim[3] : frequency -> time⁻¹
- SDim[4] : density -> mass/volume
- SDim[5] : concentration -> amount/volume
- SDim[6] : velocity -> length/time
- SDim[7] : acceleration -> length/time²
- SDim[8] : force -> mass length time⁻²
- SDim[9] : pressure -> force/area
- SDim[10] : stress -> force/area
- SDim[11] : viscosity -> pressure time dynamic viscosity
- SDim[12] : energy -> force length
- SDim[13] : work -> force length
- SDim[14] : heat -> force length
- SDim[15] : power -> energy/time
- SDim[16] : charge -> current time
- SDim[17] : voltage -> power/current
- SDim[18] : emf -> voltage
- SDim[19] : field -> voltage/length electric field strength
- SDim[20] : resistance -> voltage/current
- SDim[21] : conductance -> resistance⁻¹
- SDim[22] : capacitance -> charge/voltage

SDim[23] : flux -> voltage time magnetic flux
SDim[24] : inductance -> resistance time
SDim[25] : dose -> energy/mass
SDim[26] : activity -> time⁻¹
SDim[27] : illuminance -> intensity steradian/area
SDim[28] : irradiance -> power/area
SDim[29] : entropy -> energy/temperature
SDim[30] : conductivity -> power temperature/length
thermal conductivity

XDios

XDios

Solution of Diophantine equations

integer equations

S.Wolfram
Jul 1981

PDios[eqn, {n1, {n1max, (n1min:0)}, (n1step:1)}], ...]
tests all specified values for the *ni*, printing those sets found to satisfy the equation or condition eqn.

```
PDios[Seqn, SSR] :: (Lcl[Xv]; Xv:Ar[Len[Xv:List[SSr]], Xv[1,1]]; \
Flat[Ar[Ar[Len[Xv], List[SSr][1,2]], Pr[SS1]; List[SS1], \
P[M[S[Seqn, Ldist[Xv->List[SS2]]]]], Len[Xv]-1])
```

#I[1]:: <XDios

#I[2]:: PDios[x^2+y^2=z^2, {x, 10}, {y, 10}, {z, 10}]

3	4	5
4	3	5
6	8	10
8	6	10

#O[2]: {{3, 4, 5}, {4, 3, 5}, {6, 8, 10}, {8, 6, 10}}

XDisc

XDisc

Polynomial discriminants

polynomial roots

J.Greif

Jul 1981

Disc[a, x]

forms the discriminant of the polynomial a in x , which must be zero if a has multiple roots.

```
Disc[a, x, Symbp[x]] :: Rslt[Expt[x, Sa]Sa - Sx D[a, x], D[a, x], Sx]
```

Conversion to eqn format

text-formatting - pretty-printing

J.Greif
May 1982**Eqn**[*expr*, *definitions*, *eqnum*]

writes eqn *definitions* and *expr* in eqn format. If number is given, the output is between .EQ .EN delimiters, with equation number *eqnum*. (If *eqnum*<0, no equation number is produced.) Otherwise, the output is placed between delimiters as an in-line expression. It restores the old print properties of functions read in from XEqnPr when finished with output. The print forms are read in as EQN properties from XEqnPr, and are locally converted to Pr properties. The user adds his own eqn constructs by giving any function an EQN property as in XEqnPr, and adding the name of the function to the list Prlist. The user adds his own eqn macros by adding them to the list Def, or adding the list Def to his own list.

```
_Eqn[Init]:: <XEqnPr
Eqn _Tier
Eqn[$expr, $defs _Listp[$defs], $num]:: \
  (Lcl[Xp, Xd]; Xp:LPropsav[Prlist, Pr]; \
  LProprest[Prlist, Pr, Map[_$1[EQN], Prlist]); \
  Xd:Cat[$defs]; \
```

remove symbolic indices

```
If[Len[Xd]>0, \
Pr["EQ "]; Do[1, 1, Len[Xd], Pr[Xd[i]]]; Pr["EN "]]; \
If[$num<0, Pr["EQ "], Pr["EQ ", $num], Pr["EQ ", $num]]; \
Pr[$expr]; Pr["EN"]; LProprest[Prlist, Pr, Xp]; )
```

```
Eqn[$expr, $defs _Listp[$defs]]:: \
  (Lcl[Xp, Xd]; Xp:LPropsav[Prlist, Pr]; \
  LProprest[Prlist, Pr, Map[_$1[EQN], Prlist]); \
  Xd:Cat[$defs]; \
  If[Len[Xd]>0, \
  Pr["EQ "]; Do[1, 1, Len[Xd], Pr[Xd[i]]]; Pr["EN "]]; \
  Pr["@ ", $expr, "@"]; LProprest[Prlist, Pr, Xp]; )
```

initialize things

Eqn[]

utilities, perhaps should go in XLProp

LPropsav _LProprest _Tier

save existing property, or Null if not set

```
LPropsav[$list, $prop]:: (Lcl[X]; X:Map[_$1[$prop], $list]; \
  Map[If[P[Match[_$X[$prop], $1]=0], $1, Null], X])
```

restore prop property to all items in list from oldlist


```

LProprest[$list,$prop,$oldist,(Len[$list]=Len[$oldist])]:=\
Ap[Set,{Map['_$1[$prop],$list],$oldist]}

#I[1]: <XEqn
#I[2]: Rex[38]
#O[2]: 2 x2 z2 (16 + x) (216 + 324x + 108y - 648x z
        + 12y z (2 + x) (50 + y2))

#I[3]: Eqn[X,Def]
.EQ
delim @@
define cint #"\o'\s+6\(\is\s0\(\ci'"#
.EN
⊙ 2 { x } sup { 2 } { z } sup { 2 } (16 + x)
    * (216 + 324x + 108y - 648x z
        + 12y z (2 + x) (50 + { y } sup { 2 } )) ⊙

#I[4]: Eqn[⊙,Def,3]
.EQ
delim @@
define cint #"\o'\s+6\(\is\s0\(\ci'"#
.EN
.EQ 3
2 { x } sup { 2 } { z } sup { 2 } (16 + x)
    * (216 + 324x + 108y - 648x z
        + 12y z (2 + x) (50 + { y } sup { 2 } ))
.EN

#I[5]: <end>

```

Warning: This is a prototype only. It is incomplete and preliminary.

XEqnPr

Printing forms for eqn output

typesetting - two-dimensional output - UNIX - TROFF
text processing - word processing

J.Greif
Jun 1982

```

Lb: " { "
Rb: " } "
Ni: "~~~~~"
_Pow[EQN] [[Sa, Sb]]:Fmt[, Lb, Sa, Rb, " sup ", Lb, Sb, Rb]
_Pow[EQN] [[Sa, 1/2]]:Fmt[, " sqrt ", Lb, Sa, Rb]
_Div[EQN] [[Sa, Sb]]:Fmt[, Lb, Sa, Rb, " over ", Lb, Sb, Rb]
_Sum[EQN] [[Sex, {Sv, Sstr, Send}]]:\
  Fmt[, " sum from ", Lb, Sv=Sstr, Rb, " to ", Lb, Send, Rb, Sex]
_Prod[EQN] [[Sex, {Sv, Sstr, Send}]]:\
  Fmt[, " prod from ", Lb, Sv=Sstr, Rb, " to ", Lb, Send, Rb, Sex]
_Int[EQN] [[Sex, {Sv, Sstr, Send}]]:\
  Fmt[, " int from ", Lb, Sstr, Rb, " to ", Lb, Send, Rb, Sex, , "d", Sv]
_Uneq[EQN] [[$x]]:Sx[" != ", {$x}]
_Err[EQN] [[Sa, Sb]]:Sx[" +- ", {Sa, Sb}]
_Union[EQN] [[$x]]:Fmt[, " union ", {$x}]
_Inter[EQN] [[$x]]:Fmt[, " inter ", {$x}]
_D[EQN] [[Sy, {Sx, Sn, Sz}]]:Fmt[, " left ", Ni, Lb, " partial "~Sn, Sy, Rb, " over ", Lb, \
  "partial ", Sx~Sn, Rb, " right ", " | sub ", Lb, Sx=Sz, Rb]

```

examples of user functions

```

_Dsum[EQN] [[$x]]:Sx[" ciplus ", {$x}]          direct sum
_Cint[EQN] [Sa, Sx]:Fmt[, " cint ", Sa, , "d", Sx]  contour integral

```

example of user definitions

```

Def[cint]: "define cint #""\o\'s+6\ (is\s8\ (ci\'""#"
                                     a contour integral sign
Def[delim]: "delim @@"

```

list of functions whose Pr properties will be saved

```

Prlist: { 'Pow, 'Div, 'Sum, 'Prod, 'Uneq, 'Err, 'Union, 'Inter, 'D, 'Dsum, 'Cint }

```


XEuIV

Euler numbers and polynomials

S.Wolfram

Feb 1982

Explicit forms for integer values of index.

$$\text{Eul}[\text{_}Oddp[\$n]] : 0$$

$$\text{Eul}[\text{_}Oddp[\$n], \$x] : 0$$

$$\text{Eul}[\text{_}Evenp[\$n]] : 2^{\$n} \text{Eul}[\$n, 1/2]$$

$$\text{Eul}[\text{_}Natp[\$n/2], \$x] : 2^{(\$n+1)/(\$n+1)} (\text{Ber}[\$n+1, (\$x+1)/2] - \backslash \text{Ber}[\$n+1, \$x/2])$$

[MOS sect. 1.5.2]

XEulgam

XEulgam

Generalized Euler-Mascheroni constants

S.Wolfram

Feb 1982

Eulgam[n]

represents the n th generalized Euler-Mascheroni constant.

```

Eulgam[0] : Euler
N[Eulgam[$n_>0]] := \
  N[Sum[Log[i]^-n/i, {i, 1, 20}] - Log[20]^-($n+1)/($n+1)]

```

[AS sect. 23.2]

• Euler

Misfeatures: Numerical accuracy of sum should be investigated.

XExDot

Dot product expansion

scalar products - distribution - reduction - vectors
scalars

S.Wolfram

Jul 1981

x_Scal

declares x to be a scalar.

Scalp[expr]

tests whether *expr* contains only scalar objects.

```
Scalp[$expr] :: Ap[And,Map[Pi[_$1[Type]=Scal],Cont[$expr]]]
```

ExDot[expr]

factors all declared scalars out of dot products.

```
ExDot[$expr] :: S[$expr, $$x.($a_Scalp[$a]).$$y-->$a $$x.$$y, \
($a_Scalp[$a]).$$x-->$a $$x, $$x.($a_Scalp[$a])-->$a $$x, \
$$x.((($a_Scalp[$a]) $$b).$$y-->$a $$x.$$b.$$y, \
((($a_Scalp[$a]) $$b).$$x-->$a $$b.$$x, \
$$x.((($a_Scalp[$a]) $$b)-->$a $$x.$$b, Inf]
```

ExMDot[expr]

sets all dot products of a matrix with its inverse to the identity.

```
ExMDot[$expr] :: S[$expr, Minv[$m].$m->1,$m.Minv[$m]->1, Inf]
```

```
#I[1]:: <XExDot
#I[2]:: x_y_Scal
#O[2]: Scal
#I[3]:: Scalp[x+y^2+1]
#O[3]: 1
#I[4]:: Scalp[a+x]
#O[4]: 0
#I[5]:: ExDot[a.(x b).(1+y).c]
#O[5]: x a.b.c (1 + y)
#I[6]:: ExMDot[a.Minv[b].b.Minv[a].c]
#O[6]: c
```

XFPow

Functionals

Functional powers – iterated functions – nested functions

S.Wolfram

Jul 1981

FPow[f, n, x]

yields n nested applications of f to x .

```

FPow_Tier
_FPow[Smpl]: {0, Inf, Inf}
FPow[Sf, $n, Natp[$n], $x] :: (Lcl[Xe]; Xe:$x; Rpt[Xe:Rp[Sf, {Xe}], $n])
FPow[Sf, 0, $x] : $x

```

#I[1]:: <XFPow

#I[2]:: FPow[f, 10, x^2]

#O[2]: f[f[f[f[f[f[f[f[f[f[f[x²]]]]]]]]]]]

#I[3]:: Ex[FPow[a \$x(1-\$x), 3, x]]

```

#O[3]:
  3      3 2      4 2      4 3      4 4      5 2      5 3      5 4
a x - a x + a x - 2 a x + a x + a x - 2 a x + a x
+ 2 a x - 6 a x + 6 a x - 2 a x + a x - 4 a x
+ 6 a x - 4 a x + a x

```


XFib

XFib

Fibonacci numbers

Golden ratio

S.Wolfram

Jul 1981

Fib[n]

yields the *n*th Fibonacci number.

```
Fib[$n_?Natp[$n]] := Floor[N[(Phi^$n/Sqrt[5]+1/2)]]
Fib[$n] := N[(Phi^$n-(-Phi)^(-$n))/Sqrt[5]]
```

XFierz

XFierz

Fierz transformations

Dirac bilinear covariants - Dirac gamma matrices - fermion factors
Clifford algebra - completeness relations

S.Wolfram

Jul 1981

DIM

denotes number of dimensions (default 4).

DIM : 4

Fz[k,l]

yields elements of the Fierz rearrangement matrix.

```
Fz[k,$l] :: (-1)~$l (DIM-2$l)/$k Fz[$k-1,$l] - \
              (DIM-$k+2)/$k Fz[$k-2,$l]
Fz[0,$l] :: -1/2~Floor[DIM/2]
Fz[l,$l] :: (-1)~($l+1) (DIM-2$l)/2~Floor[DIM/2]
```

Fierz[dim]

yields the complete Fierz transformation matrix for any natural number of dimensions *dim*.

```
Fierz[$dim, Natp[$dim]] :: (Lc[DIM]; DIM:$dim; \
  If[Evenp[DIM], Ar[{{0,DIM},{0,DIM}}, Fz], \
  Ar[{{0,(DIM+1)/2},{0,(DIM+1)/2}}, Fz]])
```

#I[1]:: <XFierz

#I[2]:: Fierz[3]

```
#O[2]:  {l0}: {l0}: -1/2, [1]: -1/2, [2]: -1/2},
         [1]: {l0}: -3/2, [1]: 1/2, [2]: 1/2},
         [2]: {l0}: -3/2, [1]: 1/2, [2]: 1/2}}
```

[T.Curtright (Univ. Florida)]

XFit

XFit

Curve fitting

linear fit - power fit - exponential fit - correlation coefficient
least squares fit - parameter determination - data analysis
smoothing - functional form

S.Wolfram
Jul 1981

Fit[{{x1,y1},{x2,y2},...},{x,y}]

obtains a linear fit for the relation between x and y.

```
Fit[$list, {x,y}] :: Ap[$y=$1 + $2 $x, Ap[Fit0, Trans[$list]]]
Fit0[$x,$y] :: (Lc1[Xb0,Xb1]; Xb1:(Len[$x] Ap[Plus,$x $y] - \
Ap[Plus,$x] Ap[Plus,$y]) / (Len[$x] Ap[Plus,$x^2] - \
Ap[Plus,$x]^2); Xb0:(Ap[Plus,$y] - Xb1 Ap[Plus,$x])/Len[$x]; \
{Xb0,Xb1})
```

FitExp[{{x1,y1},{x2,y2},...},{x,y}]

obtains an exponential fit of the form $y = a b^x$ for the relation between x and y.

```
FitExp[$list, {x,y}] :: Ap[N[$y = Exp[$1] Exp[$2]^$x], \
Ap[Fit0[$X1,N[Log[$X2]]], Trans[$list]]]
```

FitPow[{{x1,y1},{x2,y2},...},{x,y}]

obtains a power fit of the form $y = a x^b$ for the relation between x and y.

```
FitPow[$list, {x,y}] :: Ap[N[$y = Exp[$1] $x^$2], \
Ap[Fit0,N[Log[Trans[$list]]]]]
```

Corr[{{x1,y1},{x2,y2},...}]

yields the population correlation coefficient between the xi and yi.

```
_Corr[init] :: <Stat
Corr[$list] :: Nilc1[Xx,Xy]; {Xx,Xy}:Trans[$list]; \
Ap[Plus,(Xx - Mean[Xx])(Xy - Mean[Xy]) / Sqrt[Var[Xx] Var[Xy]]]
```

```
#I[1]:: <XFit
#I[2]:: t:Ar[5,N[{Exp[$]}]]
#O[2]:: {{1,5.436564},{2,14.77811},{3,40.17107},{4,109.1963},{5,296.8263}}
#I[3]:: Fit[X,{x,y}]
#O[3]:: 109.8776 + y = 67.71977x
#I[4]:: S[X,x->3]
#O[4]:: y = 93.28167
#I[5]:: FitPow[t,{x,y}]
#O[5]:: y = 3.953884 x2.421572
#I[6]:: FitExp[t,{x,y}]
#O[6]:: y = 2 2.718282x
```

#1[7]:: Corr[t]

#0[7]: .3278786

Future enhancements: Should give error bars on fitted parameters.

Function generation

lambda expression – make generic – create function
create pattern

S.Wolfram

Jul 1981

Fun[*expr*, {*a*, *b*, *c*, ...}]

yields a list to be used as the value of a symbol whose projections give values for *expr* with filters corresponding to *a*, *b*, ... in that order.

```
Fun_Tier
Fun[$expr, $list_>Listp[$list]] :: (Lcl[Xl, Xa]; Xl:Map[Dummy, $list]; \
    Ap[Set, {Ap[Xa, Xl], S[$expr, Ldist[$list->Xl]]}); Ret[Xa])

Dummy[$s_>Symbp[$s]] :: Make["$ ", $s]
```

XG

Dirac algebra

gamma matrix algebra - Chisholm identities - trace identities
Feynman diagrams - electrodynamics - quantum field theory

S.Wolfram

Jul 1981

Ginds

is a list of all symbols assigned type Gind.

```
Ginds :: Rel[Cont[, _$1[Type]=Gind]]
```

VCon[expr]

contracts any repeated pairs of indices in dot products in *expr*.

```
Con[sexpr] :: S[Ex[sexpr, , , , In['Vdot, $1]], \
  Vdot[$Xmu, _$Xmu[Type]=Gind, $Xmu] -> Ndim, \
  Vdot[$Xmu, _$Xmu[Type]=Gind, $Xv1] Vdot[$Xmu, $Xv2] -> \
  Vdot[$Xv1, $Xv2], Inf]
```

Transformations of products of Dirac gamma matrices.

```
SG[1] : ('G[$$x, $p, $q, (Ord[$p, $q]<0), $$y]) -> 2('G[$$x, $$y]) $p[1].$q[1] - \
  'G[$$x, $q, $p, $$y]
```

```
SG[2] : ('G[$p, $q, (Ord[$q, $p]<0)]) --> 2 $p[1].$q[1] - G[$q, $p]
```

```
SG[3] : ('G[$$x, $p, $q, (Ord[$q, $p]<0)]) --> 2 $p[1].$q[1] - G[$$x, $q, $p]
```

```
SG[4] : ('G[$p, $q, (Ord[$q, $p]<0), $$x]) --> 2 $p[1].$q[1] - G[$q, $p, $$x]
```

Chisholm's identity in four dimensions.

```
SG[5] : ('G[$mu, _$mu[Type]=Gind, $$x, _Oddp[Len[$$x], $mu]) --> -2 Rev[$$x]
```


XGFit

General least squares fitting

Regression - statistics - curve fitting - data analysis
parameter fitting - function fitting

S.Wolfram and P.Leyland

Feb 1982

GFit[[{*x1,y1*}, {*x2,y2*}, ...], *form*, {*par1,par2*, ...}]
finds values of the parameters *par1, par2, ..* which yield the least square deviation of the template *form* from the curve specified by the points {*x1,y1*}, {*x2,y2*}, ...

```
GFit_Tier
GFit[list_, listp[$list], f, spars_, listp[$spars]] := (Lc[Xd]; \
  Xd:Ap[Plus, (Map[f, Trans[list][1]]-Trans[list][2])^2]; \
  Sol[Ldist[Ar[Len[spars], 0][Xd, spars[$1]]]=0], spars][1])
```

```
#I[1]:: <XGFit
```

```
#I[2]:: t:Cat[Ar[{{0,1,0.1}}, {$1,N[Exp[$1]]}]
```

```
#O[2]:: {{0,1}, {0.1,1.10517}, {0.2,1.2214}, {0.3,1.34986}, {0.4,1.49182},
  {0.5,1.64872}, {0.6,1.82212}, {0.7,2.01375}, {0.8,2.22554},
  {0.9,2.4596}}
```

```
#I[3]:: form:a0+a1 $1+a2 $1^2
```

```
#O[3]:: a0 + a1 $1 + a2 $12
```

```
#I[4]:: pars:{a0,a1,a2}
```

```
#O[4]:: {a0,a1,a2}
```

```
#I[5]:: GFit[t,form,pars]
```

```
#O[5]:: {a0 -> 1.0064, a1 -> 0.88906, a2 -> 0.79764}
```

```
#I[6]:: GFit[t,a0+a1 $1+a2 $1^3, {a0,a1,a2}]
```

```
#O[6]:: {a0 -> 0.990118, a1 -> 1.16909, a2 -> 0.580704}
```

Future enhancements: Errors etc.

XGQInt

XGQInt

Gaussian quadrature integration

numerical integration - approximate integration - numerical quadrature
symbolic-numeric interface

GQInt [*f*, {*x*, *lo*, *hi*}, *npt*]

forms the numerical integral of *f* with respect to *x* between *lo* and *hi* using *npt*-point Gaussian quadrature.

```

GQInt[f, {x, lo, hi}, npt, In[npt, {6, 12, 24}]] := \
  N[(hi-lo)/2 Sum[S[f, x-> \
    ((hi-lo)GQx[npt, 1]+(hi+lo))/2] GQw[npt, 1], \
    {1, -npt/2, npt/2}]]

GQx[6, 0]:GQw[6, 0]:0
GQx[6, 1]:0.2386191868
GQx[6, 2]:0.6612093864
GQx[6, 3]:0.9324695142
GQx[6, -1]:-GQx[6, 1]
GQx[6, -2]:-GQx[6, 2]
GQx[6, -3]:-GQx[6, 3]
GQw[6, 1]:GQw[6, -1]:0.4679139345
GQw[6, 2]:GQw[6, -2]:0.3607615738
GQw[6, 3]:GQw[6, -3]:0.1713244923

```

```

#I[1]:: <XGQInt;GQInt;> <XGQInt
#I[2]:: GQInt[f[x], {x, -1, 1}, 6]
#O[2]: .1713245f[-.9324695] + .3607616f[-.6612094] + .4679139f[-.2386192]
      + .4679139f[.2386192] + .3607616f[.6612094]
      + .1713245f[.9324695]
#I[3]:: GQInt[x^2, {x, 0, 1}, 6]
#O[3]: .3333333

```


XGammaS

Gamma function

Euler Gamma function - Euler integral of first kind

S.Wolfram
Feb 1982

Functional equations

Recurrence relations

$$SGamma[1,1] : Gamma[z] \rightarrow (z-1) Gamma[z-1]$$

$$SGamma[1,2] : Gamma[z] \rightarrow Gamma[z+1]/z$$
CanGam[*expr*, *reps*]

applies recurrence relations until the arguments of all Gamma functions in *expr* are canonical, and vanish when the replacements *reps* are applied.

$$\text{CanGam}[\$expr, \$\$reps] :: S[\$expr, \text{Gamma}[\$1_ \text{Natp}[\text{Ex}[\$1, \$\$reps]]]] \rightarrow \backslash$$

$$(\text{Lc}[\{z, Xn\}; Xn: \text{Ex}[\$1, \$\$reps]]; Xz: \text{Ex}[\$1-Xn]; \backslash$$

$$\text{Prod}[Xz+Xi, \{Xi, 0, Xn-1\}] \text{Gamma}[Xz])$$

#I[1]:: <XGammaS

#I[2]:: t:Gamma[2-e/2]+Gamma[4+e]Gamma[1-e]

#O[2]:: Gamma[2 - e/2] + Gamma[1 - e] Gamma[4 + e]

#I[3]:: CanGam[t, e->0]

#O[3]::
$$\frac{-e \text{Gamma}[-e/2] (1 - e/2)}{2}$$

$$- e^2 \text{Gamma}[-e] \text{Gamma}[e] (1 + e) (2 + e) (3 + e)$$

#I[4]:: CanGam[t, e->1]

#O[4]:: Gamma[2 - e/2] + e Gamma[-1 + e] Gamma[1 - e] (-1 + e) (1 + e) (2 + e)

$$e (3 + e)$$

• XGammaV

XGammaV

XGammaV

Gamma function

Euler Gamma function - Euler integral of first kind

S.Wolfram

Feb 1982

Definitions for special values

 $\Gamma[1] : 1$ $\Gamma[2] : 1$ $\Gamma[n_Natp[n]] : (n-1)!$ $\Gamma[1/2] : \text{Sqrt}[\text{Pi}]$ $\Gamma[n_Natp[n-1/2]] : \text{Pi}^{(1/2)} 2^{(1/2-n)} (2n-2)!!$ $\Gamma[-1/2] : -2 \text{Sqrt}[\text{Pi}]$ $\Gamma[n_Natp[1/2-n]] : (-2)^{(1/2-n)} \text{Sqrt}[\text{Pi}] / (-2n)!!$

Derivatives

 $D[\Gamma[x], \{x, 1, x\}] : \Gamma[x] \text{Psi}[x]$

[AS sect. 8.1; GR sect. 8.3; MOS sect. 1.1]

XGenocchi

XGenocchi

Genocchi numbers

S.Wolfram
Sep 1982

Genocchi [n]

n th Genocchi number.

`Genocchi[n] := 2^(2-2n) n Eul[2n-1]`

[Sloane: Handbook of Integer Sequences, sect. 3.13]

XGenp

XGenp

Generic symbol test

dummy symbol test – generic predicate

S.Wolfram

Jul 1981

Genp[*expr*]

yields 1 if *expr* contains generic symbols, and 0 otherwise.

```
Genp[$expr] :: P[Len[Cont[$expr, _$1[Gen]]]]
```


XGr

Basic graph theory

network theory – nodes – arcs – incidence matrix
 adjacency matrix – graph representation – graph equivalence
 graph isomorphism – Euler circuits – graph traversability
 Hamilton circuits

S. Wolfram

Jul 1981

A non-directed graph is represented by its incidence matrix, which specifies the number of arcs (edges) connecting each pair of nodes.

Nodes [*list*]

gives the number of nodes in the graph represented by *list*.

```
Nodes[$list] :: Len[$list]
```

Arcs [*list*]

gives the number of arcs in the graph represented by *list*.

```
Arcs[$list] :: Ap[Plus, Flat[$list]]/2
```

Regs [*list*]

gives the number of regions (faces) for planar graphs represented by *list*.

```
Regs[$list] :: Arcs[$list] - Nodes[$list] + 2
```

Degree [*list*, *n*]

yields the degree of node *n* in the graph represented by *list*.

```
Degree[$list, $n] :: Ap[Plus, $list[$n]]
```

ToArc [*list*]

converts the incidence matrix *list* to a list of Arc projections representing arcs.

```
ToArc[$list] :: Map[Arc, Flat[Ar[Dim[$list], \
  Repl[{$1, $2}, $list[$1, $2]], 1]]
```

ToNode [*list*]

converts a list of arcs to an incidence matrix.

```
ToNode[$list] :: Ar[Ar[2, `Ap[Max, Flat[S[$list, Arc->list]]], \
  Len[Pos[Arc[List[$$x], $list]]]
```

Eulerp [*list*]

tests whether the graph represented by the incidence matrix *list* is Euler traversable.

```
Eulerp[$list] :: Ap[Plus, Map[Oddp, Map[Ap[Plus, $1], $list]]] <= 2
```

Hamp [*list*]

tests whether the graph represented by *list* is Hamilton traversable.


```
Hamp[$list] := (Lcl[Xu]; Xu:Ar[Len[$list],0]; Xu[1]:1; Hamp0[$list,1])
Hamp0[$list,$n] := (Lcl[Xt,Xi]; Xt:Xi:0; Loop[Xi<=Len[$list], \
    If[Xu[$list[$n,Xi]],0,Xu[Xi]:1; If[~(Xt:Hamp0[$list,Xi]), \
    Xu[Xi]:0]]; Inc[Xi,~Xt]; Xt)
```

Relab[*list*, *perm*]

relabs the nodes in the graph represented by the incidence matrix *list* according to the permutation *perm*.

```
_Relab[Init] := <XPerm0
Relab[$list,$perm] := Apper[$perm,Map[Apper[$perm,$i],$list]]
```

Isop[*gr1*, *gr2*]

tests for isomorphism of the graphs represented by incidence matrices *gr1* and *gr2*.

```
Isop[$gr1,$gr1] : 1
Isop[$gr1,$gr2_>Nodes[$gr1]~Nodes[$gr2]] : 0
Isop[$gr1,$gr2_>Arcs[$gr1]~Arcs[$gr2]] : 0
Isop[$gr1,$gr2_>Isop1[$gr1]~Isop1[$gr2]] : 0
    Isop1[$list] := Sort[Map[Ap[Plus,$i],$list]]
Isop[$gr1,$gr2_>~P[Isop2[$gr1]=Isop2[$gr2]]] : 0
    Isop2[$list] := Ex[Det[$list-XIam Ar[Dim[$list]]]]
```

```
#I[1]:: <XGr
#I[2]:: g: {{0,2,0,0},{2,0,1,1},{0,1,0,1},{0,1,1,0}}
#O[2]:: {{0,2,0,0},{2,0,1,1},{0,1,0,1},{0,1,1,0}}
#I[3]:: Nodes[g]
#O[3]:: 4
#I[4]:: Arcs[g]
#O[4]:: 5
#I[5]:: Regs[g]
#O[5]:: 3
#I[6]:: Degree[g,2]
#O[6]:: 4
#I[7]:: ToArc[g]
#O[7]:: {Arc[1,2],Arc[1,2],Arc[2,1],Arc[2,1],Arc[2,3],Arc[2,4],
    Arc[3,2],Arc[3,4],Arc[4,2],Arc[4,3]}
#I[8]:: ToNode[X]
#O[8]:: {{0,2,0,0},{2,0,1,1},{0,1,0,1},{0,1,1,0}}
#I[9]:: Eulerp[g]
#O[9]:: 1
#I[10]:: Hamp[g]
#O[10]:: 0
#I[11]:: Relab[g,{3,1,2,4}]
#O[11]:: {{0,0,1,1},{0,0,2,0},{1,2,0,1},{1,0,1,0}}
```


#I[15]:: Isop[g,g]

#O[15]: 1

#I[16]:: Isop[g,Sort[g]]

#O[16]: 8

XHal t

XHal t

Solution to halting problem

insoluble problem – oracle – Church's thesis

S.Wolfram

Jul 1981

Haltp[*prog*]
yields 1 if *prog* halts after a finite time.

```
Haltp[$prog] :: ($prog;1)
```

[Turing: A model for computation with applications to the ...]

XHarm

XHarm

Harmonic sequence

S.Wolfram

Jul 1981

Harm[n]

represents the n th partial sum of the harmonic sequence.

`Harm[$n] :: Sum[1/xi, {xi, 1, $n}]`

XHist

Histogram generation

data presentation - binning - discretization - quantization

S.Wolfram

Jul 1981

Hist[*list*, *nbin*]forms a histogram of the contents of *list* with *nbin* bins.

```
Hist[$list_Listp[$list], $nbin_Natp[$nbin]] := \
(Lcl[Xmin, Xmax, Xstep, Xhist, Xind]; Xstep: N[((Xmax: Ap[Max, $list]) - \
(Xmin: Ap[Min, $list])) / $nbin]; Xhist: Ar[{{Xmin, Xmax, Xstep}}, 0]; \
Do[Xi, Len[$list], Xind: Xmin + Xstep Floor[$list[Xi] / Xstep]; \
  Xhist[Xind]: Xhist[Xind] + 1; Xhist)
```

#I1]: <XHist

#I2]: t: Ar[10, Prime]

#O2]: {2, 3, 5, 7, 11, 13, 17, 19, 23, 29}

#I3]: Hist[t, 3]

#O3]: {[2]: 4, [11]: 3, [20]: 2, [29]: 1}

XHof

XHof

Hofstadter's recursive function

double recursion - generalized Fibonacci sequence - recursion testing

S.Wolfram

Jul 1981

Hof[n]

gives a recursive function defined by Hofstadter, whose values have several properties of randomness.

```
Hof[1]:Hof[2]:1  
Hof[$n_>Natp[$n]] :: Hof[$n]:Hof[$n-Hof[$n-1]]+Hof[$n-Hof[$n-2]]
```

XHorn

Horner representation

numerical evaluation of polynomials - polynomial rearrangement

S.Wolfram

Jul 1981

Horn[*poly*, *x*]constructs a Horner representation of the polynomial *poly* with respect to *x*.

```

Horn[$poly,$x] := (Lcl[$p,$n]; If[(Xn:Expt[$x,$poly])<-1,Ret[$poly]; \
Xp:Coef[$x^Xn,$poly]; Do[{Xl,Xn-1,Xp:Xp $x + Coef[$x^(Xn-Xl),$poly]}; \
 $x Xp + S[$poly,$x->0])

```

#I[1]:: <XHorn

#I[2]:: t:x³+4a x²+2x+c#O[2]:: c + 2x + 4a x² + x³

#I[3]:: Horn[t,x]

#O[3]:: c + x (2 + x (4a + x))

XInd

List index manipulation

S.Wolfram
Jul 1981

LInd[*list*]

yields a list of the indices in *list*.

```
LInd[$list] :: Ar[Len[$list], Ind[$list, $1]]
```

ToL[*list*]

writes entries in *list* as {index,value}.

```
ToL[$list] :: Ar[Len[$list], {Ind[$list, $1], Elem[$list, { $1 }]}]
```

Maxind[*list*]

yields the maximal index in *list*.

```
Maxind[$list_>Contp[$list]] :: Len[$list]
Maxind[$list] :: Ap[Max, LInd[$list]]
```

ToInd[*list*]

takes sublists {index,value} in *list*, and forms an indexed list [index]:value.

```
ToInd[$list] :: (Lcl[X]; Map[X[{ $1[1] : $1[2] }, $list]; X])
```

```
#I[1]:: <XInd
```

```
#I[2]:: t: { [a]: x2, [b]: x3, [c]: x+y }
```

```
#O[2]:: { [a]: x2, [b]: x3, [c]: x + y }
```

```
#I[3]:: LInd[X]
```

```
#O[3]:: { a, b, c }
```

```
#I[4]:: ToL[t]
```

```
#O[4]:: { { a, x2 }, { b, x3 }, { c, x + y } }
```

```
#I[5]:: ToInd[X]
```

```
#O[5]:: { [c]: x + y, [b]: x3, [a]: x2 }
```

XIndep

XIndep

Independent variable declaration

total derivatives - partial derivatives - differentials
functional independence

S.Wolfram
Jul 1981

Indep[{y1,y2,...},{x1,x2,...}]
defines the y_i to be independent of the x_j so that $Dt\{y_i, x_j\} = 0$.

`Indep[Sy, Sx] :: Map[Map[Dt[{S1,S2}:0, Sx], Sy]`

```

#I[1]:: <XIndep
#I[2]:: Dt[{y1,y2,y3},x]
#O[2]: {Dt[y1,x],Dt[y2,x],Dt[y3,x]}
#I[3]:: Indep[{y1,y2},{x,xp]}
#O[3]: {{0,0},{0,0}}
#I[4]:: Dt[{y1,y2,y3},x]
#O[4]: {0,0,Dt[y3,x]}

```


XInfo

Basic information theory

Shannon entropy - frequency - language analysis

S.Wolfram

Jul 1981

Shan[*prob*]

represents the Shannon entropy for a language whose symbols have relative frequencies as given in the list *prob*.

```
Shan[$prob_>Listp[$prob]] :: -N[Ap[Plus,Map[$x Log[$x,2],$prob]]/ \
    Ap[Plus,$prob]]
```

Freq[*list*]

yields a *list* of the relative frequencies of symbols appearing as elements of *list*.

```
Freq[$list_>Listp[$list]] :: (Lc[Xf]; Map[Xf[$1[1]]:$1[2], \
    Rex[$list][2]]; Re[Xf])
```

#1[1]:: <XInfo

#1[2]:: Ar[10,\$x~2]

#0[2]:: {1,4,9,16,25,36,49,64,81,100}

#1[3]:: Shan[X]

#0[3]:: -5.817603

#1[4]:: {a,a,b,c,a,c,d,e,a,b,c,b,b,e}

#0[4]:: {a,a,b,c,a,c,d,e,a,b,c,b,b,e}

#1[5]:: Freq[X]

#0[5]:: {e: 2, [d]: 1, [c]: 3, [b]: 4, [a]: 4}

XIntp

XIntp

Integer testing simplification

S.Wolfram
Jul 1981

```

Intp[($x Intp[$x]) + ($y Intp[$y])] : 1
Intp[($x Intp[$x]) ($y Intp[$y])] : 1

Intp[($n Intp[$n]) ^ ($m Natp[$m])] : 1
Intp[($n Abs[$n] > 1) ^ ($m Natp[-$m])] : 0

#I[1]:: <XIntp
#I[2]:: Intp[x]:1
#0[2]: 1
#I[3]:: Intp[x^2-2]
#0[3]: 1

```


XIter

General iterated forms

replication - iteration - generalized sum - generalized product

S.Wolfram

Jul 1981

Iter [*f*, *expr*, *var*, *lo*, *hi*]applies *f* to the set of values of *expr* attained when *var* takes on values *lo*, *lo*+1, *lo*+2, ... , *hi*.

```

Iter ⌊ Tier
  Iter[Smp]: {0, 0, 0, Inf, Inf}
  Iter[f, Sexpr, Svar, $lo, $hi, Intp[$hi-$lo]] :: \
    Rp[f, Ar[{{ $lo, $hi }}, S[Sexpr, Svar->$1]]]

```

#I[1]:: <XIter

#I[2]:: Iter[f, x+i^2, i, 1, 6]

#O[2]:: f[1 + x, 4 + x, 9 + x, 16 + x, 25 + x, 36 + x]

XItp

Lagrange interpolation of list values

interpolation - smoothing - extrapolation

S.Wolfram

Jul 1981

Itp[list, x]

uses all the values given in *list* to yield an optimal estimate for the value corresponding to an index *x*.

```

Itp_Tier
Itp[list, $x] := (Len[list]; xn:Len[list]; \
  Sum[Xx:Ind[list, i]; Prod[$x-Ind[list, j], {j, 1, xn}]/ \
  (($x-Xx) Prod[Xx-Ind[list, j], {j, 1, xn, 1, j}]) \
  Elem[list, {i}], {i, 1, Len[list]})

#I[1]:: <XItp
#I[2]:: t:Ar[{{0,1,0.2}}, N[Sin[$x]]]
#O[2]:: {[0]: 0, [1.2]: .1986693, [1.4]: .3894183, [1.6]: .5646425,
        [1.8]: .7173561, [1]: .841471}

#I[3]:: Itp[X, x]
#O[3]:: 25.8684x (-1 + x) (-4/5 + x) (-3/5 + x) (-2/5 + x)
        - 101.411x (-1 + x) (-4/5 + x) (-3/5 + x) (-1/5 + x)
        + 147.0423x (-1 + x) (-4/5 + x) (-2/5 + x) (-1/5 + x)
        - 93.40574x (-1 + x) (-3/5 + x) (-2/5 + x) (-1/5 + x)
        + 21.91331x (-4/5 + x) (-3/5 + x) (-2/5 + x) (-1/5 + x)

#I[4]:: Ex[X]
#O[4]:: .999978x + .0002439154 x2 - .1676164 x3 + .001612961 x4
        + .007252478 x5

#I[5]:: N[S[X, x->0.8]]
#O[5]:: .7173561
#I[6]:: N[Sin[0.8]]
#O[6]:: .7173561

```

• XLItp

XK11110

Input/Output removal

reclaim memory - conserve memory - save memory - forget past
 destroy input - destroy output - kill input - kill output
 kill labels - kill lines

S.Wolfram

Jul 1981

Updated Aug 1982

K11110[(n1:0),(n2:Len[#I])]

removes values assigned to #I and #0 lines numbered n1 through n2.

```

K11110_ Tier
K11110[$n1_ Natp[$n1], $n2_ Natp[$n2]] :: \
  (Ar[{$n1, $n2}], #I[$X1]:#0[$X1]:, )
K11110[] :: (#I:#0:)

#I[1]:: <K11110
#I[2]:: t:x;
#I[3]:: Rpt[t:t(1+t),2]
#0[3]: x (1 + x) (1 + x (1 + x))
#I[4]:: Ex[X]
#0[4]: x + 2 x2 + 2 x3 + x4
#I[5]:: Fac[X]
#0[5]: x (1 + x) (1 + x + x2)
#I[6]:: K11110[1,3]
#I[7]:: #0
#0[7]: {[6]: {[1]: , [2]: , [3]: }, [5]: x (1 + x) (1 + x + x2),
  [4]: x + 2 x2 + 2 x3 + x4, [0]: {"/u1/swp/ALL/swp.init"}}
#I[8]:: K11110[]
#I[9]:: #0
#0[9]: {[8]: }
#I[10]:: #I
#0[10]: {[9]:: #0, [8]:: K11110[]}

```

XLArith

List arithmetic

S.Wolfram
Jul 1981

LSum[list]

yields the sum of all elements in the list or set of nested lists *list*.

LSum[\$list] :: Ap[Plus,Flat[\$list]]

LProd[list]

yields the product of elements in *list*.

LProd[\$list] :: Ap[Mult,Flat[\$list]]

#I[1]:: <XLArith

#I[2]:: {{a,b},{1,2,3},d}

#O[2]:: {{a,b},{1,2,3},d}

#I[3]:: LSum[X]

#O[3]:: 6 + a + b + d

#I[4]:: LProd[~~2~~]

#O[4]:: 6a b d

XLCM

XLCM

Lowest common multiple

S.Wolfram
Jul 1981

LCM[n1,n2,...]
yields the lowest common multiple of *n1,n2...*

```
LCM Flat
LCM[$n1,$n2] :: ($n1 $n2)/Gcd[$n1,$n2]
```

XLChi2

List chi squared evaluation

goodness of fit - function fitting - curve fitting
 model comparison - model fitting

S.Wolfram

Jan 1982

XLChi2 [*list*, *form*, *i*]

forms the chi squared between elements of *list* and *form* as a function of *i*.

```
LChi2[$list_Listp[$list], $form, $i] := \
  Sum[(N[$list[[i]] - S[$form, $i -> i])^2], {i, 1, Len[$list]}]
```

```
#I[1]:: <XLChi2
```

```
#I[2]:: Ar[5, N[Exp[$1]]]
```

```
#O[2]:: {2.71828, 7.38906, 20.0855, 54.5982, 148.413}
```

```
#I[3]:: LChi2[X, 1+x+x^2/2, x]
```

```
#O[3]:: 18747.8
```


XLDiff

Differences of list elements

forward differences - pairwise differences - pairwise subtraction

S.Wolfram

Jan 1982

Updated Aug 1982

LDiff[list, {n:1}]

yields a list of n th rank forward differences between successive entries of *list*.

```
LDiff[{$list, Listp[$list]] := Ar[Len[$list]-1, $list[$1+1]-$list[$1]]
LDiff[{$list, 1}] := LDiff[$list]
LDiff[{$list, $n, Natp[$n]] := LDiff[LDiff[$list, $n-1]]
```

```
#I[1]:: <XLDiff
#I[2]:: Ar[5, f]
#O[2]:: {f[1], f[2], f[3], f[4], f[5]}
#I[3]:: LDiff[X]
#O[3]:: {-f[1] + f[2], -f[2] + f[3], -f[3] + f[4], -f[4] + f[5]}
#I[4]:: LDiff[0, 2]
#O[4]:: {f[1] - 2f[2] + f[3], f[2] - 2f[3] + f[4], f[3] - 2f[4] + f[5]}
#I[5]:: Ar[10, $1^3-4$1^2+7]
#O[5]:: {4, -1, -2, 7, 32, 79, 154, 263, 412, 607}
#I[6]:: Ar[3, LDiff[X, $1]]
#O[6]:: {{-5, -1, 9, 25, 47, 75, 109, 149, 195}, {4, 10, 16, 22, 28, 34, 40, 46},
        {6, 6, 6, 6, 6, 6, 6}}
```

XLItp

XLItp

Interpolation of contiguous list values

Lagrangian interpolation - function evaluation

S.Wolfram

Jul 1981

LItp[list, x]

uses all the values given in *list* to find an interpolated value for an element with index *x*.

```
LItp[$list, $x] := \
  (Len[$list]; $n:Len[$list]; \
  Sum[$list[k+Floor[(Xn+1)/2]] (-1)^(Floor[Xn/2]+k) \
  /((k+Floor[(Xn-1)/2])!*(Floor[Xn/2]-k)!*(Xn-Floor[(Xn+1)/2]-k)) \
  Prod[$x-Oddp[Xn]-t, {t, Evenp[Xn], 2Floor[Xn/2]}, \
  {k, -Floor[(Xn-1)/2], Floor[Xn/2]})
```

LItp2[list, x]

uses two-point (linear) interpolation to yield an estimate for an element of *list* with index *x*.

```
LItp2[$list, $x_> (1 <= $x <= Len[$list])] := \
  (Len[$list]; $x:Floor[$x]; (1-$x+$x) $list[$x]+($x-$x) $list[$x+1])
```

LItp3[list, x]

uses three-point Lagrange interpolation to estimate the value of an element of *list* with index *x*.

```
LItp3[$list, $x_> (2 <= $x <= Len[$list]-1)] := \
  (Len[$list]; $x:Floor[$x]; \
  $p($x-1)/2 $list[$x-1] + (1-$x+$x) $list[$x] + \
  $x($x+1)/2 $list[$x+1])
```

#I[1]:: <XLItp

#I[2]:: t:Ar[6, \$x^4]

#O[2]:: {1, 16, 81, 256, 625, 1296}

#I[3]:: LItp[t, 3.5]

#O[3]:: 158.8625

#I[4]:: 3.5^4

#O[4]:: 158.8625

#I[5]:: LItp2[t, 3.5]

#O[5]:: 168.5

#I[6]:: LItp3[t, 3.5]

#O[6]:: 154.75

#I[7]:: LItp[Ar[3], x]

$$\#O[7]:: \frac{(-3+x)(-2+x)}{2} - 2(-3+x)(-1+x) + \frac{3(-2+x)(-1+x)}{2}$$

#1[8]:: Ex[7]

#0[8]: x

XLPart

Subpart position lists

power set

S.Wolfram

Jul 1981

LPart[*expr*]

yields a list of the positions of all subparts of *expr*.

LPart[*Sexpr*] :: Pos[*S1*, *Sexpr*]

APart[*expr*]

yields a list of all subparts of *expr*, together with the positions at which they appear.

APart[*Sexpr*] :: Pos[*S1*, *Sexpr*, List[*Sexpr*[*SSx*], List[*SSx*]]

#I[1]:: <XLPart

#I[2]:: t:Rex[]

#O[2]: $2x(2 + 2x) + (-1 - 2x^2)(3 + y + \frac{3}{x})$

#I[3]:: LPart[t]

#O[3]: {{1,1}, {1,2,1}, {1,2,2}, {1,2}, {1}, {2,1,1}, {2,1,2,1}, {2,1,2,2}, {2,1,2},
 {2,1}, {2,2,1}, {2,2,2}, {2,2,3,1}, {2,2,3,2}, {2,2,3}, {2,2}, {2},
 {0}}

#I[4]:: APart[t]

#O[4]: {{x, {1,1}}, {2, {1,2,1}}, {2x, {1,2,2}}, {2 + 2x, {1,2}}, {2x(2 + 2x), {1}},
 {-1, {2,1,1}}, {x, {2,1,2,1}}, {2, {2,1,2,2}}, {-2x², {2,1,2}},
 {-1 - 2x², {2,1}}, {3, {2,2,1}}, {y, {2,2,2}}, {1, {2,2,3,1}},
 {x, {2,2,3,2}}, {-, {2,2,3}}, {3 + y + $\frac{3}{x}$, {2,2}},
 {(-1 - 2x²)(3 + y + $\frac{3}{x}$), {2}}, {' Plus', {0}}}

XLProp

List property assignment

multiple assignment - property - distribution over lists

J.Greif

Jul 1982

LProp[*list*, *prop*, (*value*:1)]assigns *value* to the *prop* property of each element of *list*.

```

LProp_Tier
LProp[$list,Listp[$list],$prop]::Map[Prset[$1,$prop],$list]
LProp[$list,Listp[$list],$prop,$val_Nump[$val]]:: \
  Ap[Set,{Map[Prop[$1,$prop],$list],$val]}]
LProp[$list,Listp[$list],$prop,$val]::Ap[Set,{Map[Prop[$1,$prop],\
  $list],$val]}]

#I[1]:: <XLProp
#I[2]:: LProp[{s,t,u},Trace,Lpr]
#O[2]: ' Lpr
#I[3]:: s[a+b I]/t[u[a+Ps[Exp[-x^2],x,0,1]] - 7e^456]
s[Cx[a,b]]
u[1 + a]
u[1 + a]
u[1 + a]
u[1 + a]
t[u[Ps[1,x,0,{0,3}],[[0]: 1 + a,[1]: 0,[2]: -1,[3]: 0]]] + A[-7,456]]

#O[3]: s -----
          2
          t[u[(1 + a) - x ] + -7e^456]

#I[4]:: LProp[{s,t,u},Ldist]
#O[4]: {Ldist,Ldist,Ldist}
#I[5]:: s[{x,y,z}]
{s[x],s[y],s[z]}
s[x]
s[y]
s[z]
#O[5]: {s[x],s[y],s[z]}
#I[6]:: <end>

```


XLUP

XLUP

LUP matrix methods

linear algebra - linear equation - determinant - matrix matrix inverse

J.Greif Jul 1982

<XMat3; <XPerm0; <XPerm1

Ldet [m] finds determinant of matrix m by the LUP decomposition.

```
Ldet[$m_>Squatp[$m]] :: (Lcl[XI]; If[(XI:Lup[$m])=0,XI, \ Tr[XI[2],Mult]*Sig[XI[3]],Tr[XI[2],Mult]*Sig[XI[3]])
```

Linv [m] finds inverse of matrix m by the LUP decomposition.

```
Linv[$m_>Squatp[$m]] :: (Lcl[XI]; If[P[(XI:Lup[$m])=0],Linv[$m], \ Trunc[Pmat[Pinv[XI[3]]].Tinv[XI[2]].Tinv[XI[1]],Len[$m]])
```

Ldiv [mat, rhs] solves the matrix equation mat.x = rhs for x by the LUP decomposition.

```
Ldiv[$mat_>Squatp[$mat], $rhs_>(Contp[$rhs]&Len[$rhs]=Len[$mat])] :: \ (Lcl[XI,Xd,Xy]; If[P[(XI:Lup[$mat])=0],Ldiv[$mat,$rhs], \ Xd=Embed[Trans[$rhs]]; \ Xy:Bsub[XI[1],Xd]; Trunc[Bsub[XI[2].Pmat[XI[3]],Xy],Len[$mat]])
Ldiv[$mat_>Squatp[$mat], $rhs_>(Matp[$rhs]&Len[$rhs]=Len[$mat])] :: \ (Lcl[XI,Xd,Xy]; If[P[(XI:Lup[$mat])=0],Ldiv[$mat,$rhs], \ Xd=Embed[$rhs]; \ Xy:Bsub[XI[1],Xd]; Trunc[Bsub[XI[2].Pmat[XI[3]],Xy],Dim[$rhs]])
```

Pmat [perm] converts a list denoting a permutation into the appropriate matrix.

```
Pmat[$p_>Permp[$p]] :: Ar[{Len[$p],Len[$p]}, $j=$p[$i]]
```

Embed [mat] embeds an n x m matrix into an identity matrix of dimension l x Max[m,l] where l is the first power of two not less than n.

```
Embed[$m_>Matp[$m]] :: (Lcl[XI,Xd]; Xd:Dim[$m]; XI:1; \ Loop[XI<Xd[1],XI:2*XI,]; If[P[XI=Xd[1]], $m, \ Ar[{XI,Max[XI,Xd[2]]}, If[$1<=Xd[1]&$2<=Xd[2], $m[$1,$2], $1=$2]])
```

Trunc [mat, dim] extract from a matrix mat the entries within the bounds given by dim. If dim is a number, extract the corresponding square matrix, otherwise extract according to the dimension list.

```
trunc_>Tier
trunc[$m_>Matp[$m], $d_>Natp[$d]] :: Ar[{ $d,$d},m[$i,$j]]
trunc[$m_>Matp[$m], $d_>Listp[$d]] :: Ar[{ $d[1],$d[2]}, $m[$i,$j]]
```


Tinv[m]

finds inverse of upper or lower triangular matrix m . See lemmas 6.5, 6.6 of reference below.

```
Tinv[Sm_Sqmatp[Sm]] :: (Lcl[Xm,Xd,Xp,Xl]; \
  If[P[Len[Sm]=1],{1/Sm[1,1]}], \
  Xm:QtrEmbed[Sm]; Xd:Tinv[Xm[4]]; Xp:Tinv[Xm[1]]; Xl:Len[Xp]; \
  Ar[2*Xl,2*Xl], Sel[Si<=Xl & Sj<=Xl,Xp[Si,Sj],Si>Xl&Sj>Xl, \
  Xd[Si-Xl,Sj-Xl],Si>Xl,(-Xd.Xm[3].Xp)[Si-Xl,Sj], \
  Xj>Xl,(-Xp.Xm[2].Xd)[Si,Sj-Xl]]);
```

Lup[m]

finds LUP decomposition of matrix m , returned as {L,U,P} where P is in the form of a permutation (contiguous list).

```
Lup_Tier
Lup[Sm_Matp[Sm]] :: Lup[Sm,Len[Sm],Len[Sm[1]]]
Lup[Sm,Sn_Matp[Sn],Sp_Matp[Sp]] :: (Lcl[Xb,Xd,Xg,Xh,Xl, \
  Xn,Xp,Xp3,Xfe,Xlup1,Xlup2,Xu]; Xn:Sn/2; \
  Xb:Ar[ {Xn,Sp},Sm]; If[P[(Xlup1:lup[Xb,Xn,Sp])=0],0, \
  Xd:Ar[ {Xn,Sp},Sm[Xn+1,S2]].Pmat[Pinv[Xlup1[3]]]; \
  B and D in notation of reference. F.(E~1) follows \
  Xfe:Ar[ {Xn,Xn},Xd].Tinv[Ar[ {Xn,Xn},Xlup1[2,1,Sj]]]; \
  Xg:Xd-Xfe.Xlup1[2]; \
  decompose G' and build up answer \
  If[P[(Xlup2:Lup[Ar[ {Xn,Sp-Xn},Xg[Si,Xn+Sj]],Xn,Sp-Xn)=0], \
  Rat[0], \
  find P3, H, L, U, P as matrix and convert latter to perm \
  Xp3:Ar[Sp,If[Si<=Xn,S1,Xn+Xlup2[3,S1-Xn]]]; \
  Xh:Xlup1[2].Pmat[Pinv[Xp3]]; Xp:Pmat[Xp3].Pmat[Xlup1[3]]; \
  Xl:Cat[Ar[ {Xn,Sp},If[S2<=Xn,Xlup1[1,S1,S2],0]], \
  Ar[ {Xn,Sp},If[S2<=Xn,Xfe[S1,S2],Xlup2[1,S1,S2-Xn]]]; \
  Xu:Cat[Xh,Ar[ {Xn,Sp},If[S2<=Xn,0,Xlup2[2,S1,S2-Xn]]]; \
  {Xl,Xu,Xp.Ar[Len[Xp]]}]]
Lup[Sm,1,Sp_Matp[Sp]] :: (Lcl[Xj,Xp]; \
  find non-zero column and perm making it first \
  Xp:Ar[Sp]; For[Xj:1,Sm[1,Xj]=0 & Xj<=Sp,,Inc[Xj]]; \
  Sel[Xj>Sp,0,Xj=1,{{{1}},Sm,Xp},1,Xp[1]:Xj;Xp[Xj]:1; \
  {{{1}},Sm.Pmat[Xp],Xp]]
```

Beub[m1,m2]

backsubstitutes to find solution x of matrix equation $m1.x=m2$ where $m1$ is upper or lower triangular. TEMPORARY - do in ordinary way

```
Beub[Sm_Sqmatp[Sm],Sr_Matp[Sp]] :: Mdiv[Sp,Sm]
```

Qtr[m]

partition a 2^n by 2^n matrix m into quarters, returning {m11,m12,m21,m22} where each element is one of the quarters.

```
Qtr[Sm_Sqmatp[Sm]] :: (Lcl[Xn]; Xn:Len[Sm]/2; \
  {Ar[ {Xn,Xn},Sm],Ar[ {Xn,Xn},Sm[S1+Xn,S2]],Ar[ {Xn,Xn},Sm[S1,S2+Xn]], \
  Ar[ {Xn,Xn},Sm[S1+Xn,S2+Xn]]})
```

[Aho, Hopcraft and Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974]

XLap

Laplace transforms

operational methods - differential equations

S.Wolfram

Jul 1981

Lap[expr, t, s]

represents the Laplace transform of expr from the t domain to s domain.

- Gamma[*n*, *natp*[*n*]]** : (*n*-1)!
- Lap[*Tier*]**
- Lap[*n*, *numbp*[*n*], *t*, *s*]** : *n*/*s*
- Lap[*t*, *t*, *s*]** : 1/*s*²
- Lap[*t*^{*p*}, *t*, *s*]** : Gamma[*p*+1]/*s*^(*p*+1)
- Lap[*x* + *ssx*, *t*, *s*]** :: Lap[*x*, *t*, *s*] + Lap[*ssx*, *t*, *s*]
- Lap[*x* *ssx*, *t*, *s*]** :: *x* Lap[*ssx*, *t*, *s*]
- Lap[*x*/*y*, *t*, *s*]** :: Lap[*x*, *t*, *s*]/*y*
- Lap[*x*=*y*, *t*, *s*]** :: Lap[*x*, *t*, *s*]=Lap[*y*, *t*, *s*]
- Lap[*D*[*y*, {*x*, *n*, *t*}], *t*, *s*]** :: $\sum_{k=0}^{n-1} \text{Lap}[S[y, x \rightarrow t], t, s] - \sum_{k=0}^{n-1} \text{D}[y, \{x, i-1, 0\}] s^{-(n-1)}, \{1, 1, n\}$
- Lap[Exp[*ssa* *t*], *t*, *s*]** :: 1/(*s*-*ssa*)*ns*
- Lap[Exp[*t*], *t*, *s*]** : 1/(*s*-1)*ssa*
- Lap[Exp[*ssa* *t*] *ssx*, *t*, *s*]** :: Lap[*ssx*, *t*, *s*-*ssa*]
- Lap[*(t+ssa)*^{*p*}, *t*, *s*]** :: $\frac{\text{Gamma}[p+1, ssa s] \text{Exp}[ssa s]}{s^{(p+1)}}$
- Lap[Log[*t*], *t*, *s*]** :: -Log[Euler *s*]/*s*
- Lap[*t*^{*p*} Log[*t*], *t*, *s*]** :: Gamma[*p*+1] (Psi[*p*+1]-Log[*s*])/s^(*p*+1)
- Lap[Log[*t*]², *t*, *s*]** : (Zeta[2] + Log[Euler *s*]²)/*s*
- Lap[Sin[*ssa* *t*], *t*, *s*]** : *ssa*/(*s*² + *ssa*²) *e*

XLatsum

Lattice sums

Madelung sums - crystal energies - zeta functions

S.Wolfram

Jul 1981

Latsum [*f*, *spec*]

sums the results of applying the template *f* to the sets of lattice points specified by *spec*.

```

Latsum[Smp]: {0, Inf}
Latsum[f, $spec] := Rp[Plus, Flat[Arc[$spec, $f]]]

```

```
#I[1]: <XLatsum
```

```
#I[2]: Latsum[1/($1^2+$2^3), {5, 4}]
```

```
#O[2]: 1.428285
```

```
#I[3]: Sum[Sum[1/(i^2+j^3), {j, 1, 4}], {i, 1, 5}]
```

```
#O[3]: 1.428285
```

XLdEq

XLdEq

Equation list distribution

matrix equations - sets of equations - simultaneous equations

S.Wolfram

Jul 1981

LdEq[*expr*]

distributes Eq over lists in *expr*, thus converting equations involving lists into lists of equations.

`LdEq[$expr] :: Ldist[$expr, 'Eq]`

`#1[1]:: <XLdEq`

`#1[2]:: {{a,b},{c,d}}. {1,2} = {5,-3}`

`#0[2]:: {a + 2b, c + 2d} = {5,-3}`

`#1[3]:: LdEq[X]`

`#0[3]:: {a + 2b = 5, 3 + c + 2d = 0}`

XLenex

Expanded length estimation

expansion - estimated size - total length - memory requirement

S.Wolfram

Jul 1981

Lenex[*expr*]yields an upper limit on the length of the expression resulting from expansion of *expr*.

```

Lenex[$x+$x] :: Lenex[$x]+Lenex[$x]
Lenex[$x $x] :: Lenex[$x] Lenex[$x]
Lenex[$x^$n] :: Natp[$n] :: Comb[Lenex[$x]+$n-1,$n]
Lenex[( $x $x )/$y] :: Lenex[$x] Lenex[$x]
Lenex[Log[$x $x]] :: Lenex[$x]+Lenex[$x]
Lenex[$x] : 1

```

• Size; Len

#I[1]:: <XLenex

#I[2]:: t:(1+x+x^2)(a+b+1)^3

#O[2]:: (1+a+b)^3 (1+x+x^2)

#I[3]:: Lenex[X]

#O[3]:: 38

#I[4]:: Ex[t]

```

#O[4]:: 1 + 3a + 3b + x + 6a b + 3a x + 3a b^2 + 3a x^2 + 3b x + 3b x^2 + 3a^2 b
+ 3a^2 x + 3a^2 x^2 + a^3 x + a^3 x^2 + 3b^2 x + 3b^2 x^2 + b^3 x + b^3 x^2
+ 6a b x + 6a b x^2 + 3a b^2 x + 3a b^2 x^2 + 3a^2 b x + 3a^2 b x^2
+ 3a^2 + a^3 + 3b^2 + b^3 + x

```

#I[5]:: Len[X]

#O[5]:: 38

#I[6]:: Lenex[t^5]

#O[6]:: 2856

XLer

XLer

Lerch transcendent

S.Wolfram
Jul 1981

```

SLer[1]:  Ler[1,$s,1] -> Zeta[$s]
SLer[2]:  Ler[1,$s,$a] -> Zeta[$s,$a]
SLer[3]:  Ler[$z,$s,1] -> Li[$s,$z]/$z
SLer[4]:  Ler[$z,$s,$a] -> I $z~$a Gamma[1-$s] (2Pi)^(s-1) \
          (Exp[-I Pi $s/2]Ler[Exp[-2 Pi I $a],1-$s,Log[$z]/(2Pi I)] \
          -Exp[I Pi ($s/2+2$a)]Ler[Exp[2Pi I $a],1-$s,1-Log[$z]/(2Pi I)])

```

[MOS pp. 33-4]

XLev

Level isolation

part extraction - structural operation - domains - depth

S.Wolfram

Jul 1981

Lev[*expr*, *n*]yields a list of parts of *expr* on level *n*.

```

Lev_Tier
Lev[Sexpr, $n] :: (Lcl[X]; X!:{}; Map[X!::Cat[X!, {$!}], Sexpr, {$n}; X!)

```

Lenlev[*expr*, *n*]finds the number of parts of *expr* at level *n*.

```

Lenlev_Tier
Lenlev[Sexpr, $n] :: (Lcl[Xn]; Xn::0; Map[`Inc[Xn], Sexpr, {$n}; Xn)

```

#I[1]:: <XLev

#I[2]:: t:Rex[]

#O[2]: 56 x² (x + z)

#I[3]:: Lev[X, 1]

#O[3]: {x², x + z}#I[4]:: Lev[*@2*, 2]

#O[4]: {x, 2, x, z}

#I[5]:: Lenlev[*@2*, 2]

#O[5]: 4

XLevi

XLevi

Levi-Civita tensor generation

totally antisymmetric tensor - basis tensor - epsilon tensor

S.Wolfram

Jul 1981

Levi [n]

yields the Levi-Civita total antisymmetric epsilon tensor in n dimensions.

`Levi[$n,Matp[$n]] :: Ar[Ar[$n,$n],Sig]`

XList0

Basic list manipulations

head - LISP CAR - beginning - prepend - catentate
 append - tail - take - select - part - insert - remove
 replicate - repeat

S.Wolfram
 Jul 1981

First [*list*]

yields the first entry in *list*.

```
First Tier
First[Elem, Listp[Elem]] :: Elem[Elem, {1}]
```

Prep [*elem*, *list*]

prepends *elem* to the beginning of *list*.

```
Prep Tier
Prep[Elem, Listp[Elem]] :: Cat[{Elem}, Elem]
```

• Cat

App [*elem*, *list*]

appends *elem* to the end of *list*.

```
App Tier
App[Elem, Listp[Elem]] :: Cat[Elem, {Elem}]
```

Tk [*n*, *list*]

takes the first *n* or last *-n* entries in *list*.

```
Tk Tier
Tk[Elem Natp[Elem], Listp[Elem]] :: Ar[Elem, Elem]
Tk[Elem Natp[-Elem], Listp[Elem]] :: \
  Cat[Ar[{Len[Elem]+{Elem}, 0}], Elem]
```

Ine [*elem*, *list*, *i*]

inserts the entry *elem* into *list* at position *i*.

```
Ine Tier
Ine[Elem, Listp[Elem], Elem Natp[Elem]] :: \
  Cat[Ar[Elem-1, Elem], {Elem}, Ar[{Elem, Len[Elem]}], Elem]
```

Rm [*list*, *n*]

removes the *n* th entry from *list*.

```
Rm Tier
Rm[Elem Listp[Elem], Elem Natp[Elem]] :: Cat[Ar[Elem-1, Elem], \
  Ar[{Elem, Len[Elem]}], Elem]
Rm[Elem Listp[Elem], Elem Natp[Elem]] :: \
  Cat[Ar[Len[Elem], Elem, Elem~Elem]]
```

Lrpt[list, n]

yields a list consisting of n repetitions of list.

```

Lrpt_Tier
Lrpt[$list, Contp[$list], $n, Natp[$n]] :: \
  Cat[Repl[$list, $n]]

```

```

#I[1]:: <XList0
#I[2]:: t: {a,b,c,d,e,f,g}
#O[2]: {a,b,c,d,e,f,g}
#I[3]:: First[t]
#O[3]: a
#I[4]:: Last[t]
#O[4]: g
#I[5]:: Prep[1, t]
#O[5]: {1, a, b, c, d, e, f, g}
#I[6]:: App[1, t]
#O[6]: {a, b, c, d, e, f, g, 1}
#I[7]:: Tk[4, t]
#O[7]: {a, b, c, d}
#I[8]:: Tk[-4, t]
#O[8]: {d, e, f, g}
#I[9]:: Ins[1, t, 4]
#O[9]: {a, b, c, 1, d, e, f, g}
#I[10]:: Rm[t, 4]
#O[10]: {a, b, c, e, f, g}
#I[11]:: Lrpt[{a,b,c}, 4]
#O[11]: {a,b,c,a,b,c,a,b,c,a,b,c}

```


XList1

Sublist manipulation

lists - list substitution - sublist positions - sequence positions

S.Wolfram

Jul 1981

LPos[*sub*, *list*]find positions at which the sublist *sub* appears in *list*.

```
LPos[$sub_>Contp[$sub], $list_>Contp[$list]] := \
  (Lcl[X1, X2]; X1: {}; Do[X1, Len[$list]-Len[$sub]+1, \
    For[X2:0, P[$sub[X2+1]=$list[X1+X2]] & X2<Len[$sub], \
      Inc[X2], 1; If[X2=Len[$sub], X1:Cat[X1, {X1}]]]; X1)
LPos[$sub_>(Contp[$sub] & Len[$sub]=1), $list_>Contp[$list]] := \
  Flat[Pos[$sub[1], $list, 2]]
```

LSub[*list2*, *list1*, *list*]substitutes *list2* for all occurrences of the sublist *list1* in *list*.

```
LSub[$list2, $list1, $list] := (Lcl[Xf]; Xf_>Flat; \
  S[S[Ap[Xf, $list], Ap[Xf, $list1]]->Ap[Xf, $list2], 1, Inf], Xf->list)
```

LS[*list*, *rep1*, *rep2*, ...]applies successively the replacements *repi* specified for sublists in *list*. (The *repi* may contain patterns.)

```
LS[$list, $$reps] := (Lcl[X]; X:$list; \
  Map[X:LSub[$1[2], $1[1], X], List[$$reps], 1, \
    ($2[0]='Rep') | ($2[0]='Repd'); X]
```

LIn[*list1*, *list*]yields 1 if *list1* is a sublist of *list*, and 0 otherwise.

```
LIn[$list1, $list] := (Lcl[Xf]; Xf_>Flat; ~P[~Match[Ap[Xf, \
  Cat[{$$X1}, $list1, {$$X2}], Ap[Xf, $list]]])
```

```
#I[1]:: <XList1
#I[2]:: t: {a,b,c,a,b,d,e,a,c}
#0[2]: {a,b,c,a,b,d,e,a,c}
#I[3]:: LPos[{a,b}, t]
#0[3]: {1,4}
#I[4]:: LSub[{g,h,i}, {a,b}, t]
#0[4]: {g,h,i,c,g,h,i,d,e,a,c}
#I[5]:: LS[t, {a,b}->{g,h,i}, {e}->{r,s}]
#0[5]: {g,h,i,c,g,h,i,d,r,s,a,c}
#I[6]:: LIn[{a,b}, t]
#0[6]: 1
#I[7]:: LIn[{a,d}, t]
```

2

#0173: 0

XList1

2

892

XLogic

Elementary propositional calculus

C.Feynman

Aug 1981

Note $p=q$ represents the logical biconditional "if and only if p then q".

Absorption laws

```

Sp | (Sp & Ssq) : Sp
Sp & (Sp | Ssq) : Sp
Sp || (Sp | Ssq) : ~Sp & Or[Ssq]
Sp || (Sp & Ssq) : Sp & ~Ssq
Sp & (Sp || Sq) : Sp & ~Sq
Sp | (Sp || Sq) : Sp | Sq
Sp | (~Sp & Ssq) : Sp | And[Ssq]
Sp & (~Sp | Ssq) : Sp & Or[Ssq]
Sp || (~Sp | Ssq) : 1
Sp || (~Sp & Ssq) : 1
Sp & (~Sp || Sq) : Sp & Sq
Sp | (~Sp || Sq) : Sp | ~Sq

```

Commutative and associative laws built in.

Distributive laws

```

SLogic[1]: Sp | (Ssq & Sr) --> (Sp | And[Ssq]) & (Sp | Sr)
SLogic[2]: Sp & (Ssq | Sr) --> (Sp & Or[Ssq]) | (Sp & Sr)

```

Idempotence

```

Sp & Sp : Sp
Sp | Sp : Sp
Sp || Sp : 0

```

Identity laws built in.

Complement laws

```

~Sp : Sp
Sp | ~Sp : 1
Sp & ~Sp : 0
Sp || ~Sp : 1
Sp => ~Sp : ~Sp

```

DeMorgan's laws

```

~(Sp | Sq) : (~Sp) & (~Sq)
~(Sp & Sq) : (~Sp) | (~Sq)
~(Sp || Sq) : (Sp=Sq)

```

Reflexive law

```

Sp => Sp : 1

```

Antisymmetric law

$(\$p \Rightarrow \$q) \& (\$q \Rightarrow \$p) : \$p = \q

Transitive law

$(\$p \Rightarrow \$q) \& (\$q \Rightarrow \$r) : (\$p \Rightarrow \$r) \& (\$p \Rightarrow \$q) \& (\$q \Rightarrow \$r)$

Consensus

SLogic[3] : $(\$p | \$\$q) \& (\sim \$p | \$\$r) \rightarrow (\$p | \$\$q) \& (\sim \$p | \$\$r) \& (\$\$q | \$\$r)$
 SLogic[4] : $(\$p \& \$\$q) | (\sim \$p \& \$\$r) \rightarrow (\$p \& \$\$q) | (\sim \$p \& \$\$r) | (\$\$q \& \$\$r)$

Complete sum and product

Cdisj[\$r] : S[S[\$r, SLogic[1], Inf], SLogic[2], Inf]
 Cconj[\$r] : S[S[\$r, SLogic[2], Inf], SLogic[1], Inf]

Alternate expressions

SLogic[5] : $\$p \Rightarrow \$q \rightarrow \$q | \sim \p
 SLogic[6] : $\$p || \$q \rightarrow (\$p | \$q) \& (\sim \$p | \sim \$q)$
 SLogic[7] : $\$p || \$q \rightarrow \$p \& \sim \$q | \sim \$p \& \q

Misfeatures: Loops infinitely on input.

XLogic2

Elementary logic with quantifiers

C.Feynman

Aug 1981

Quant [*q1, q2, ..., stat*]represents the statement *stat* subject to the quantifiers *q1, q2, ...***All [*x, (l: (all))*]**represents the quantifier "for all *x* in the list *l*". The second argument is optional; if it is not supplied the quantifier "for all *x*" is represented. In this case, automatic simplification is not possible**Some [*x, (l: (all))*]**represents the quantifier "for some *x* in the list *l*". The second argument may be omitted; in this case the quantifier "there exists an *x* such that" is represented. In this case, automatic simplification is not possible.

```
Quant[$$q, Quant[$$r]] : Quant[$$q, $$r]
Quant[$s] : $s
```

DeMorgan's law

```
~Quant[All[$x], $$q, $s] : Quant[Some[$x], ~Quant[$$q, $s]]
~Quant[Some[$x], $$q, $s] : Quant[All[$x], ~Quant[$$q, $s]]
```

Simplification of restricted-range quantifiers

Misfeatures: THESE HAVE NOT BEEN FULLY TESTED, DUE TO A BUG IN THE SIMPLIFIER WHILE THEY WERE BEING WRITTEN. If you use them and something goes wrong, it might be my fault.

```
Quant[$$x, All[$y, $z _> (Listp[$z] & ~ P[$z = {}]), $prop] :: Quant[$$x, \
  Ap[And, Map[S[$prop, $y -> $q], $z]]]
Quant[$$x, Some[$y, $z _> (Listp[$z] & ~ P[$z = {}]), $prop] :: Quant[$$x, \
  Ap[Or, Map[S[$prop, $y -> $q], $z]]]
```

XLogicPr

Logical truth table generation

Boolean algebra – state table – first-order logic

S.Wolfram

Jul 1981

PrTF[*expr*]

prints a table giving the values of the logical expression *expr* for all possible truth values of symbols appearing in it.

```
PrTF[$expr] := (Lc[ $\{x\}$ ]; Ap[Pr, Cat[ $\{x\}$ :Cont[$expr], {$expr}]]; \
  Ar[Ar[Len[ $\{x\}$ ], {{0,1}}], \
  Ap[Pr, S[Cat[ $\{x\}$ , {$expr}], Ldist[ $\{x\}$ ->List[ $\{S\}$ ]]]])
```

```
#I[1]:: <XLogicPr
```

```
#I[2]:: PrTF[(p&r)=>(q|~p)]
```

p	q	r	p & r => q ~ p
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

```
#O[2]: {{0}: {{0}: {{0}: 1, [1]: 1}}, [1]: {{0}: 1, [1]: 1}},
  [1]: {{0}: {{0}: 1, [1]: 0}, [1]: {{0}: 1, [1]: 1}}}
```


XLor

XLor

Lorentz vectors

relativistic mechanics – four vectors – Minkowski space

S.Wolfram

Jul 1981

Ldot [*list1*, *list2*]

forms the contraction of two Lorentz vectors with a metric of signature ---+

`Ldot[$list1,$list2] :: {-1,-1,-1,1}.($list1 $list2)`

XMKs

XMKs

MKS/SI units

physical quantities - dimensional analysis - physical constants
 units conversion - CGS units

S.Wolfram

Jul 1981

◀DIM

Units for fundamental dimensions

SSI[0,1] : length -> metre
 SSI[0,2] : mass -> kilogram
 SSI[0,3] : time -> second
 SSI[0,4] : current -> ampere
 SSI[0,5] : temperature -> kelvin
 SSI[0,6] : intensity -> candela
 SSI[0,7] : amount -> mole

fundamental units

SSI[1,1] : metre -> length
 SSI[1,2] : kilogram -> mass
 SSI[1,3] : second -> time
 SSI[1,4] : ampere -> current
 SSI[1,5] : kelvin -> temperature
 SSI[1,6] : candela -> intensity
 SSI[1,7] : mole -> amount

derived units

SSI[2,1] : hertz -> frequency
 SSI[2,2] : newton -> force
 SSI[2,3] : pascal -> pressure
 SSI[2,4] : joule -> energy
 SSI[2,5] : watt -> power
 SSI[2,6] : coulomb -> charge
 SSI[2,7] : volt -> voltage
 SSI[2,8] : ohm -> resistance
 SSI[2,9] : siemens -> conductance

SSI[2,10] : mho -> conductance
SSI[2,11] : farad -> capacitance
SSI[2,12] : weber -> flux
SSI[2,13] : tesla -> weber/area
SSI[2,14] : henry -> inductance
SSI[2,15] : gray -> dose
SSI[2,16] : becquerel -> activity
SSI[2,17] : lumen -> intensity steradian
SSI[2,18] : lux -> illuminance

multipliers

SSI[2,1] : exa -> 10^{18}
SSI[2,2] : peta -> 10^{15}
SSI[2,3] : tera -> 10^{12}
SSI[2,4] : giga -> 10^9
SSI[2,5] : mega -> 10^6
SSI[2,6] : kilo -> 10^3
SSI[2,7] : hecto -> 10^2
SSI[2,8] : deca -> 10
SSI[2,9] : deci -> 0.1
SSI[2,10] : centi -> 10^{-2}
SSI[2,11] : milli -> 10^{-3}
SSI[2,12] : micro -> 10^{-6}
SSI[2,13] : nano -> 10^{-9}
SSI[2,14] : pico -> 10^{-12}
SSI[2,15] : femto -> 10^{-15}
SSI[2,16] : atto -> 10^{-18}

XMOrd

Multiplicative orders

coding theory - arithmetic over finite fields
S.Wolfram
Oct 1982

<LCM

Prerequisites: XLCM

MOrd[k, n]

yields the multiplicative order of k modulo n . (Smallest j such that $k^j = 1$ modulo n). *

```

MOrd ↪ Tier
MOrd[Sk, Sn ↪ (Len[Nfac[Sn]] = 1)] :: (Lc[i, t]; For[i:1; t:B[Sk], \
  N[Mod[t, Sn]] ~ 1, Inc[i], t:t B[Sk]]; Ret[i])
MOrd[Sk, Sn ↪ (Gcd[Sk, Sn] ~ 1)] : 0
MOrd[Sk, Sn] :: Ap[LCM, Map[MOrd[Sk, $i[1] ~ $i[2]], Nfac[Sn]]]
MOrd[Sk, 1] : 0

```

GMOrd[k, {n1, n2, ...}]

generalized multidimensional multiplicative order of k .

```

GMOrd ↪ Tier
GMOrd[Sk, Slist ↪ (Listp[Slist] & Ap[And, Ldist[Gcd[Sk, Slist] = 1]])] : A
(Lc[t, i]; For[i:1; t:B[Sk], Ap[And, N[Ldist[Mod[t, Slist] ~ 1]]], \
  Inc[i], t:t B[Sk]]; Ret[i])

```

Misfeatures: Test in GMOrd should Ldist before failing Eq test

XMSet

Automatic memo definition

dynamic programming - look-up

T.Shaw

Jul 1981

expr ::= ***val***

assigns the value *val* to the projection *expr* in such a way as to store explicitly each form of *expr* requested.

```

MSet ↪ Tier
_MSet[Smp]:0
_MSet[Pr][Sexpr,Sval] :: Sx[" :: ", {Sexpr,Sv},4]
Sxset[":::",MSet,4]
Sexpr :: Sval :: Sexpr :: Sexpr : Sval

```

```

#I[1]:: <XMSet
#I[2]:: f[0]:f[1]:1
#O[2]: 1
#I[3]:: f[$x]::: $x f[$x-2]
#O[3]: ' f[$x] : $x f[$x - 2]
#I[4]:: f
#O[4]: {[0]: 1, [1]: 1, [$x]:: f[$x] : $x f[$x - 2]}
#I[5]:: f[6]
#O[5]: 48
#I[6]:: f
#O[6]: {[6]: 48, [4]: 8, [2]: 2, [0]: 1, [1]: 1,
        [$x]:: f[$x] : $x f[$x-2]}
#I[7]:: f[11]
#O[7]: 10395
#I[8]:: f
#O[8]: {[11]: 10395, [9]: 945, [7]: 105, [5]: 15, [3]: 3, [6]: 48, [4]: 8,
        [2]: 2, [0]: 1, [1]: 1, [$x]:: f[$x] : $x f[$x - 2]}

```

XMask

XMask

List element masking

select - choose - extract - remove

S.Wolfram

Aug 1982

Mask [*mask*, *list*]

masks out those elements of *list* for which the corresponding elements of *mask* are not determined to be true.

```
Mask[{$mask_>Contp[$mask], $list_>Contp[$list]}] :: \
  Cat[Ar[Len[$list], $list, $mask[$i]]]
```

```
#I[1]:: <XMask
#I[2]:: Ar[10]
#O[2]: {1,2,3,4,5,6,7,8,9,10}
#I[3]:: Map[Evenp, X]
#O[3]: {0,1,0,1,0,1,0,1,0,1}
#I[4]:: Mask[00, 02]
#O[4]: {2,4,6,8,10}
```


XMat1

Matrix input and generation

enter matrix - interactive matrix input - diagonal matrices

S.Wolfram

Jul 1981

MRd $[n, m]$

reads in turn each entry of an $n * m$ matrix.

```
MRd[$n,$m] :: Ar[{ $n,$m }, Pr[{ $1,$2 }, ": "; Rd[]]
```

Diag $[list]$

yields a matrix with diagonal *list* and all other entries zero.

```
Diag[$list_>Contp[$list]] :: Ar[Ar[2, `Len[$list]], \
  If[$X1=$X2, $list[$X1], 0]]
```

```
#I[1]:: <XMat1
```

```
#I[2]:: MRd[2,2]
```

```
{1,1} : a1
{1,2} : a1
{2,1} : b1
{2,2} : b2
```

```
#O[2]: {{a1,a2},{b1,b2}}
```

```
#I[3]:: Diag[3,2,a]
```

```
#O[3]: {{3,0,0},{0,2,0},{0,0,a}}
```

XMat2

XMat2

Structural matrix operations

diagonals - minors

S.Wolfram

Jul 1981

<XMat3

LDiag[mat]

yields a list of the elements on the leading diagonal of mat.

```
LDiag[Smat_Matp[Smat]] :: Ar[Min[Dim[Smat][1],Dim[Smat][2]], \
Smat[$X1,$X1]]
```

TDiag[mat]

yields a list of the elements on the trailing diagonal of mat.

```
TDiag[Smat_Matp[Smat]] :: Ar[Min[Dim[Smat][1],Dim[Smat][2]], \
Smat[$X1,Dim[Smat][2]-$X1+1]]
```

Minor[expr,i,j]

forms the ij th minor of the matrix expr.

```
_Minor[Init] :: <List0
Minor[Smat_Matp[Smat],Si_Matp[$i],Sj_Matp[$j]] :: \
Map[Rm[$X1,$j],Rm[Smat,$i]]
```

#I[1]:: <XMat2

#I[2]:: m:Ar[{3,3},f]

#O[2]:: {{f[1,1],f[1,2],f[1,3]},{f[2,1],f[2,2],f[2,3]},{f[3,1],f[3,2],f[3,3]}}

#I[3]:: LDiag[m]

#O[3]:: {f[1,1],f[2,2],f[3,3]}

#I[4]:: TDiag[m]

#O[4]:: {f[1,3],f[2,2],f[3,1]}

#I[5]:: Minor[m,2,3]

#O[5]:: {{f[1,1],f[1,2]},{f[3,1],f[3,2]}}

XMat3

Matrix character tests

square matrix – symmetric matrix – antisymmetric matrix
 diagonal matrix – projection matrix – matrix classes
 matrix types

S.Wolfram
 Jul 1981

Matp[*expr*]

yields 1 if *expr* represents a matrix (rank 2 tensor).

`Matp[$expr] :: Fullp[$expr,2]`

Sqmatp[*expr*]

yields 1 if *expr* represents a square matrix.

`Sqmatp[$expr] :: Matp[$expr] & P[Dim[$expr][1] = Dim[$expr][2]]`

Symp[*expr*]

yields 1 if *expr* represents a symmetric matrix.

`Symp[$expr] :: Matp[$expr] & P[$expr = Trans[$expr]]`

Asymp[*expr*]

yields 1 if *expr* represents an antisymmetric matrix.

`Asymp[$expr] :: Matp[$expr] & P[$expr = -Trans[$expr]]`

Diagp[*expr*]

yields 1 if *expr* represents a diagonal matrix.

`Diagp[$expr] :: Matp[$expr] & \`
`P[$expr-Diag[LDiag[$expr]] = Ar[Dim[$expr],2],0]`

Projcp[*expr*]

yields 1 if *expr* represents a projection matrix.

`Projcp[$expr] :: Matp[$expr] & Ex[$expr.$expr=$expr]`

XMat4

Algebraic matrix operations

matrix power - repeated transformation - hermitean adjoint
 matrix adjoint - characteristic polynomial - eigenvalue equation
 generalized trace

S.Wolfram

Jul 1981

<XMat3

Mpow[*mat*, *n*]yields the *n* th power of the matrix *mat* (for integer *n*).

```
Mpow[$mat_>Squatp[$mat],0] :: Ar[Ar[2,Len[$mat]]]
Mpow[$mat_>Squatp[$mat],$n_>Natp[$n]] :: Dot[Repl[$mat,$n]]
Mpow[$mat_>Squatp[$mat],$n_>Natp[-$n]] :: Dot[Repl[Minv[$mat],-$n]]
```

Adj[*expr*]forms the Hermitean adjoint of *expr*.

```
Adj[$expr] :: Conj[Trans[$expr]]
```

Cof[*expr*, *i*, *j*]forms the *ij* th cofactor of the matrix *expr*.

```
_Cof[Init] :: <XMat2
Cof[$expr_>Matp[$expr],$i_>Natp[$i],$j_>Natp[$j]] :: \
  Det[Minor[$expr,$i,$j]]
```

Charpol[*expr*, *var*]forms the characteristic polynomial for the matrix *expr* with respect to *var*.

```
Charpol[$expr_>Matp[$expr],$var] :: Det[$expr - $var Ar[Dim[$expr]]]
```

Gentr[*expr*, *k*]forms the *k* th order trace of the matrix *expr*.

```
Gentr[$expr_>Matp[$expr],$k_>Natp[$k]] :: (Lcl[Xiam]; \
  Coef[Xiam^$k,Ex[Charpol[$expr,Xiam]])
```

#I[1]:: <XMat4

#I[2]:: m: {{a1,a2},{b1,b2}}

#O[2]:: {{a1,a2},{b1,b2}}

#I[3]:: Cof[m,1,1]

#O[3]:: b2

#I[4]:: Charpol[m,x]

#O[4]:: -a2 b1 + (a1 - x) (b2 - x)

#I[5]:: Ex[X]

#O[5]:: a1 b2 - a1 x - a2 b1 - b2 x + x²


```
#I[6]:: Centr[m,2]
#O[6]: 1
#I[7]:: Centr[m,1]
#O[7]: -a1 - b2
```

• XPer

XMaxind

XMaxind

Find maximal index

S.Wolfram

Jul 1981

Maxind [*list*]

yields the maximal index in *list*.

```
Maxind[$list_Contp[$list]] :: Len[$list]
Maxind[$list] :: Ap[Max,Ar[Len[$list],Ind[$list,$1]]]
```

```
#I[1]:: <XMaxind
```

```
#I[2]:: t:{[4]:a,[5]:b,[2]:c,[1]:d}
```

```
#O[2]:: {[4]: a, [5]: b, [2]: c, [1]: d}
```

```
#I[3]:: Maxind[t]
```

```
#O[3]:: 5
```

XMorse

XMorse

Morse code translator

S.Wolfram

Jul 1981

Deciphers Morse code (international version).

```

" " = ""
"-." = "a"; ".-.-" = "b"; "-.-." = "c"; "-.-" = "d"; ".-" = "e"; ".-.-.-" = "f";
"-.-" = "g"; ".-.-.-" = "h"; ".-" = "i"; ".-.-.-" = "j"; ".-.-" = "k"; ".-.-.-" = "l";
"-.-" = "m"; ".-" = "n"; "-.-.-" = "o"; ".-.-.-" = "p"; "-.-.-" = "q"; ".-.-.-" = "r";
".-.-" = "s"; ".-.-" = "t"; ".-.-.-" = "u"; ".-.-.-" = "v"; ".-.-.-" = "w"; ".-.-.-" = "x";
".-.-.-" = "y"; ".-.-.-" = "z";

```

```

#I[1]:: <XMorse
#I[2]:: ... -- .-.
#O[2]: ssp

```


XN

Number type conversion

arbitrary precision - fixed precision - arbitrary accuracy
 fixed accuracy - type casting - type coercion - type conversion

S.Wolfram
 Aug 1982

FN[*number*]

converts *number* from arbitrary to fixed precision form.

```
FN[F[$1,$2,$3,$4,$5,$6]] :: ($1 + ($2 + $3/10*^3)/10*^3)*^($5-4)
FN[$n_>Nump[$n]] : $n
FN[$x] : $x
```

NF[*number*]

converts *number* to 12-digit precision form.

```
NF[$n_>Nump[$n]] :: $n F[1]
NF[$n] : $n
```

```
#I[1]:: <XN
#I[2]:: N[P]
#O[2]: 3.14159
#I[3]:: NF[X]
#O[3]:* (3.14159265359)
#I[4]:: FN[X]
#O[4]: 3.14159
```

Misfeatures: These coercions should be done automatically when they are required, making this file redundant.

XNAT

Natural units

S.Wolfram
Jul 1981

Basic constants :

- Qc velocity of light in vacuo
- Qhbar Planck's constant
- Qk Boltzmann's constant
- Qalpha Dimensionless fine structure constant
- QG Newton's constant of gravitation
- QN Avogadro's number.

<XDim

1. Conversion from system with Qhbar : Qc : Qk : 1

- SNAT[1,1] : length -> Qhbar/(Qc mass)
- SNAT[1,2] : time -> Qhbar/(Qc^2 mass)
- SNAT[1,3] : temperature -> (Qk mass Qc^2)^-1
- SNAT[1,4] : current -> mass (Qhbar^-1 Qc^5 alpha)^(1/2)
- SNAT[1,5] : amount -> QN

#I[1]:: <XNAT

#I[2]:: current energy/length

#O[2]: $\frac{\text{current energy}}{\text{length}}$

#I[3]:: Si[X,SDim]

#O[3]: $\frac{\text{current length mass}^2}{\text{time}}$

#I[4]:: Si[X,SNAT[1]]

#O[4]: $\frac{Qc^{1/2} \alpha^{1/2} \text{mass}^3}{Qhbar^{3/2}}$

XNMap

Multi-element Map

S.Wolfram

Jul 1981

NMap [*f*, *list*, *n*]applies *f* to groups of *n* elements in *list*.

```

NMap[Smp]: {0, Inf, Inf}
_MMap[Init] := <UnFlat
NMap[f, $list, Contp[$list], $n, Natp[$n]] := \
  Map[Ap[f, $X1], UnFlat[$list, $n]]

```

```

#I[1]:: <XNMap
#I[2]:: t:Ar[10]
#O[2]: {1,2,3,4,5,6,7,8,9,10}
#I[3]:: NMap[f, t, 2]
#O[3]: {f[1,2], f[3,4], f[5,6], f[7,8], f[9,10]}
#I[4]:: NMap[f, t, 3]
#O[4]: {f[1,2,3], f[4,5,6], f[7,8,9], f[10]}

```

XNSol

XNSol

Numerical solution of equations by Newton's method

S.Wolfram
Jul 1981

NSol [eqn, x, x0, acc]

attempts to find a solution for x in the equation eqn using Newton's method starting at the point $x = x0$, with accuracy acc .

```

NSol_ Tier
NSol [$a=$b, $x_:=Symbp[$x], $x0_:=Numbp[N[$x0]], $acc_:=Numbp[$acc]] :: \
(LcI [Xf, Xdf, Xx, Xx0]; Xx:=$x0; Xf:=$a-$b; Xdf:=D[Xf, $x]; \
Loop[, Xx:=N[(Xx0:Xx)-S[Xf/Xdf, $x->Xx]], Abs[Xx-Xx0]>$acc; Xx)

```

```

#I[1]:: <XNSol
#I[2]:: NSol [Sin[x]=x, x, 1, 1/1000]
#O[2]: .001476886
#I[3]:: N[Sin[X]]
#O[3]: .001476886

```


XNorm

XNorm

Vector norm

S.Wolfram

Jul 1981

Norm[*list*]

yields the norm of the vector represented by *list*.

```
Norm[$list] :: Sqrt[Abs[Plus,$list^2]]
```

XOrbit

XOrbit

Planetary orbits

Solar system - celestial mechanics - astronomical data
orbital elements - Kepler's laws

S.Wolfram
Jul 1981

[Handbook of British Astronomical Association (1982); Astronomical Ephemeris]

Fundamental parameters:

- Epoch : time in Julian days (T)
- LEpoch : longitude of planet at epoch (L)
- LPeri : longitude of perihelion
- LNode : longitude of ascending node
- Incl : inclination to ecliptic (i)
- Ecc : eccentricity (e)
- MDist : mean distance (a)
- Per : sidereal period in days (P)

Heliocentric quantities:

MAnom [t]

mean anomaly at time *t* from epoch

$$MAnom[t] : 2\pi t/Per + LEpoch - LPeri$$

EccAnom [t]

eccentric anomaly at time *t* from epoch

_PXi[Pr] : FmtI[{{1,-1},{2,-1},{3,-1},{2,0},{1,1},{2,1},{3,1}},"-","--","-", \

PInt
prints as

/

_PInt[Pr] : FmtI[{{1,1},{1,0},{1,-1}},"/","|","/"]

PSqrt
prints as

√

_PSqrt[Pr] : FmtI[{{1,-1},{2,0},{3,1}},"√","/","--"]

#I[1]:: <XPR

#I[2]:: PPI~2

#O[2]:: $\frac{\quad}{\quad}$

#I[3]:: f[Pr]:PSig

#O[3]:: $\frac{\quad}{\quad}$

#I[4]:: f+1/f

#O[4]:: $\frac{\quad}{\quad} + \frac{1}{\quad}$

XPad

List padding

left justify - right justify - trailing zeroes - leading zeroes

S.Wolfram

Aug 1982

LPad [*list*, *elem*, *len*]

pads out the list *list* on the left with element *elem* to length *len* if necessary.

```
LPad[list, elem, len] :: If[Len[list] >= len, list, \
    Cat[Ar[len-Len[list], `elem], list]]
```

RPad [*list*, *elem*, *len*]

pads out the list *list* on the right with element *elem* to length *len* if necessary.

```
RPad[list, elem, len] :: If[Len[list] >= len, list, \
    Cat[list, Ar[len-Len[list], `elem]]]
```

XPaul i

XPaul i

Pauli sigma matrices

S.Wolfram

Jul 1981

Sigma[i]

gives a representation of the the i th Pauli sigma matrix.

```
Sigma[0] : {{1,0},{0,1}}
Sigma[1] : {{0,1},{1,0}}
Sigma[2] : {{0,-I},{I,0}}
Sigma[3] : {{1,0},{0,-1}}
```


XPause

XPause

Pause

S.Wolfram

Jul 1981

Pause [n]

pause for n seconds.

```
Pause[ $\$n$ , Nump[ $\$n$ ]] :: (Lc[c0]; c0:Clock[1][1]; \
Loop[Clock[1][1]-c0< $\$n$ , 1])
```

XPeel

XPeel

Sublist peeling

S.Wolfram

Jul 1981

Peel [{*expr1*, *expr2*, ...}]

represents the sequence *expr1*, *expr2*, ... in a list. ("Peels" away lists).

Peel[*Slist* → *Listp*[*Slist*]] :: *Ap*[*Np*, *Slist*]

#I[1]:: <XPeel

#I[2]:: {Peel[{a,b}], {c,d}}

#O[2]:: {a,b, {c,d}}

XPer

XPer

Matrix permanents

determinants

L.Yaffe

August 1982

Per [*mat*]

yields the permanent of the matrix *mat*

[E.R.Caianello: "Combinatorics and Renormalization in Quantum Field Theory". p.29]

```

<Mat2
Per[Smat]=Squmatp[Smat] :: Sum[Smat[1,i] Per[Minor[Smat,1,i]], \
{i,1,Dim[Smat][1]}]
Per[{{Sx}}] :: Sx

```

#I[1]:: <Per

#I[2]:: m:Ar[3,3], f]

#O[2]: {{f[1,1], f[1,2], f[1,3]}, {f[2,1], f[2,2], f[2,3]}, {f[3,1], f[3,2], f[3,3]}}

#I[3]:: Per[m]

```

#O[3]: f[1,1] (f[2,2] f[3,3] + f[2,3] f[3,2])
        + f[1,2] (f[2,1] f[3,3] + f[2,3] f[3,1])
        + f[1,3] (f[2,1] f[3,2] + f[2,2] f[3,1])

```

#I[4]:: Det[m]

```

#O[4]: f[1,1] (f[2,2] f[3,3] - f[2,3] f[3,2])
        - f[1,2] (f[2,1] f[3,3] - f[2,3] f[3,1])
        + f[1,3] (f[2,1] f[3,2] - f[2,2] f[3,1])

```

XPerm0

Permutations

S.Wolfram

Jul 1981

A permutation is represented by a contiguous list of sequential integers in any order.

Permp[*expr*]

yields 1 if *expr* represents a permutation.

```
Permp[$expr] :: P[Sort[$expr]=Ar[Len[$expr]]]
Permp[$expr_>~Contp[$expr]] : 0
```

Fiper[*list1*, *list2*]

finds if possible a permutation which places the elements of *list2* in the order of *list1*.

```
Fiper[$list1_>Contp[$list1], \
      $list2_>(Contp[$list2]&Len[$list1]=Len[$list2])] :: \
Flat[Ar[Len[$list1], Pos[$list2[$X1], $list1]]]
```

Apper[*perm*, *list*]

applies the permutation *perm* to *list*.

```
Apper[$perm_>Permp[$perm], $list_>Contp[$list]] :: \
Ar[Len[$list], $list[$perm[$X1]]]
```

```
#I[1]:: <XPerm0
#I[2]:: Permp[{1,3,2,4}]
#O[2]:: 1
#I[3]:: Permp[{1,3,1,4}]
#O[3]:: 0
#I[4]:: t1:{a,c,d,b}
#O[4]:: {a,c,d,b}
#I[5]:: Fiper[{a,b,c,d},t1]
#O[5]:: {1,3,4,2}
#I[6]:: Apper[X,{a,b,c,d}]
#O[6]:: {a,c,d,b}
```


XPerm1

Elementary operations on permutations

S.Wolfram

Jul 1981

<XPerm8

Pcomp[perm.1, perm.2]forms the composition (product) of the two permutations *perm.1* and *perm.2*.
$$\text{Pcomp}[Sp1 \rightarrow \text{Permp}[Sp1], Sp2 \rightarrow \text{Permp}[Sp2]] :: \text{Ar}[\text{Len}[Sp1], Sp2[Sp1[\$X1]]]$$
Ppow[perm, n]forms the *n*th power of the permutation *perm*.
$$\begin{aligned} \text{Ppow}[Sp \rightarrow \text{Permp}[Sp], 0] &:: \text{Ar}[\text{Len}[Sp]] \\ \text{Ppow}[Sp \rightarrow \text{Permp}[Sp], Sn \rightarrow \text{Natp}[Sn]] &:: \begin{cases} \text{S}[Sp, \$X1 \rightarrow \text{Ar}[\text{Len}[Sp], \$X1[Sp[\$X2]]], Sn-1 \\ \text{Ppow}[Sp \rightarrow \text{Permp}[Sp], Sn \rightarrow \text{Natp}[-Sn]] :: \text{Ppow}[\text{Pinv}[Sp], -Sn] \end{cases} \end{aligned}$$
Pinv[perm]yields the inverse of the permutation *perm*.
$$\text{Pinv}[Sp \rightarrow \text{Permp}[Sp]] :: \text{Ar}[\text{Len}[Sp], \text{Pos}[\$X1, Sp][1, 1]]$$

```

#I[1]:: <XPerm1
#I[2]:: p1: {1,5,4,2,3}
#O[2]:: {1,5,4,2,3}
#I[3]:: p2: {5,1,3,2,4}
#O[3]:: {5,1,3,2,4}
#I[4]:: Pcomp[p1,p2]
#O[4]:: {5,4,2,1,3}
#I[5]:: Pcomp[p2,p1]
#O[5]:: {3,1,4,5,2}
#I[6]:: Ppow[p2,6]
#O[6]:: {4,5,3,1,2}
#I[7]:: Pinv[p1]
#O[7]:: {1,4,5,3,2}
#I[8]:: Pcomp[X,p1]
#O[8]:: {1,2,3,4,5}

```

XPermC

Cycle decomposition of permutations

S.Wolfram

Jul 1981

Cycles are represented by projections of the form $C[i_1, i_2, i_3, \dots]$.

$\langle \text{Perm} \rangle$

ToC[perm]

yields a list of cycles whose composition is *perm* (cycle decomposition).

```
ToC[Sp_>Permp[Sp]] := (Lc[{Xa, Xi, Xn, Xi}]; \
  Xa: {}; Xt: Ar[Len[Sp], 1]; Do[Xi, Len[Sp], If[Xt[Xi], \
    For[Xn: Sp[Xi]; Xi: {}, \
      Xt[Xn], Xn: Sp[Xn], Xt[Xn]: 0; Xi: Cat[Xi, {Xn}]]; \
  Xa: Cat[Xa, {Ap[C, Xi]}]]]; Xa)
```

ToP[cycs]

yields the permutation represented by the list of cycles *cycs*.

```
ToP[Scycs_>Listp[Scycs]] := (Lc[Xi]; Map[Ar[Len[Sc], \
  Xi[Cyc[Sc, -1][Xi]]: Sc[Xi]], Scycs, 1]; Xi)
```

#I[1]: $\langle \text{PermC} \rangle$

#I[2]: ToC[{1, 5, 7, 2, 4, 3, 6}]

#O[2]: {C[1], C[5, 4, 2], C[7, 6, 3]}

#I[3]: ToP[X]

#O[3]: {1, 5, 7, 2, 4, 3, 6}

XPhys

Fundamental physical constants

S.Wolfram
Jul 1981

1. Principal constants

speed of light in vacuo

SPhys[1,1] : Qc -> 2.997924588*⁸ metre second⁻¹

Planck's constant

SPhys[1,2] : Qh -> 6.626179*⁻³⁴ joule second

Dirac's constant

SPhys[1,3] : Qhbar -> 1.0545887*⁻³⁴ joule second

charge on electron

SPhys[1,4] : Qe -> 1.6021892*⁻¹⁹ coulomb

mass of electron

SPhys[1,5] : Qme -> 9.109534*⁻³¹ kilogram

Avogadro's number

SPhys[1,6] : QN -> 6.022045*²³ mole⁻¹

constant of gravitation

SPhys[1,8] : QG -> 6.6728*⁻¹¹ newton metre² kilogram⁻²

2. Atomic constants

fine structure constant

SPhys[2,1] : Qalpha -> 7.2973506*⁻³

Rydberg constant (infinite nuclear mass)

SPhys[2,2] : QRinf -> 1.097373177*⁷ metre⁻¹

Bohr radius

SPhys[2,3] : Qa0 -> 5.2917706*⁻¹¹ metre

electron Compton wavelength

SPhys[2,4] : QlambdaC -> 2.4263089*⁻¹² metre

classical "electron radius"

SPhys[2,5] : Qre -> 2.817938*⁻¹⁵ metre

Thomson cross-section

SPhys[2,6] : QsigmaT -> 6.6522448*⁻²⁹ metre²

Bohr magneton

SPhys[2,7] : QmuB -> 9.274078e^-24 joule tesla^-1

nuclear magneton
 SPhys[2,8] : QmuN -> 5.5858824e^-27 joule tesla^-1

gyromagnetic ratio of free proton
 SPhys[2,9] : Qgamma -> 2.6751987e^8 second^-1 tesla^-1

electron volt
 SPhys[2,10] : QeV -> 1.6021892e^-19 joule

3. Thermal constants

molar gas constant
 SPhys[3,1] : QR -> 8.31441 joule kelvin^-1 mole^-1

Loschmidt number
 SPhys[3,2] : QL -> 2.686754e^25 metre^3

Boltzmann constant
 SPhys[3,3] : Qk -> 1.380662e^-23 joule kelvin^-1

Stefan-Boltzmann constant
 SPhys[3,4] : Qsigma -> 5.67032e^-8 watt metre^-2 kelvin^-4

XPIhist

XPIhist

Histogram plotting

S.Wolfram

Jul 1981

PIhist[list]

plot *list* as a histogram.

```
PIhist[Slist] := Plot[Ar[Len[Slist], \
Line[{Pt[{Ind[Slist,$1],0}],Pt[{Ind[Slist,$1],Elem[Slist,{S1}]}]}, \
Pt[{Ind[Slist,$1+1],Elem[Slist,{S1}]}]},Pt[{Ind[Slist,$1+1],0}]}], \
Axes[0,0]]
```

XPlot

XPlot

Operations on plots

S.Wolfram
Jul 1981

PCat [*p1*, *p2*, ...]

combines the plots *p1*, *p2*, ... into a single plot.

```
PCat[Plot[Sp1, SSp1], Plot[Sp2, SSp2]] :: Plot[Union[Sp1, Sp2]]
```

PAP [*trx*, *try*, *pl*]

applies the templates *trx* and *try* respectively to each point in the plot *pl*.

```
PAP[Strx, Stry, Sp1] :: S[Sp1, Pt[{Sx, Sy}, SSf] --> \  
Pt[{Ap[Strx, {Sx}], Ap[Stry, {Sy}]}, SSf]]
```

Ycut [*plot*, {*ymin*, *ymax*}]

replot *plot* in region $ymin < y < ymax$.

```
Ycut[Plot[S1, SS2], {Symin, Symax}] :: Plot[S1, {Symin, Symax}]
```

```
Circle[{Sx, Sy}, Sr] :: \  
Graph[{Sr Sin[X1] + Sx, Sr Cos[X1] + Sy}, X1, 0, 2Pi][1, 1]
```

Future enhancements: Add functions to include labels, arrows etc.

XPolar

XPolar

Polar graphs

S.Wolfram

Jul 1981

Polar [*expr*, *theta*, *npt*]

yields a polar plot of *expr* obtained by evaluation at *npt* points in the angle *theta*.

```
Polar[sexpr, theta, npt] :: \  
Graph[{sexpr Cos[theta], sexpr Sin[theta]}, theta, 0, 2Pi, ,, npt]
```

XPoly

XPoly

Information on polynomials

S.Wolfram

Jul 1981

LCoef [*poly*, *var*]yields a list of the coefficients of powers of *var* in *poly*.
$$\text{LCoef}[\text{\$poly}, \text{\$var}] :: \text{Ar}[\text{Expt}[\text{\$var}, \text{\$poly}], \text{Coef}[\text{\$var}^{\text{\$X1}}, \text{\$poly}]]$$
LExpt [*poly*, *var*]yields a list of the exponents with which *var* appears in *poly*.
$$\text{LExpt}[\text{\$poly}, \text{\$var}] :: \text{Union}[\text{Expt}[\text{\$var}, \text{\$poly}], \text{List}]$$

#I[1]:: <XPoly

#I[2]:: t:Ex[(a+x)³ (1-x)²]
$$\begin{aligned} \#0[2]: & 3a^2x^2 - 6a^3x^3 + 3a^4x^4 + 3a^2x^2 - 6a^2x^2 + 3a^2x^3 - 2a^3x^3 + a^3x^2 \\ & + a^3x^3 - 2a^4x^4 + a^5x^5 \end{aligned}$$

#I[3]:: LCoef[t,x]

$$\#0[3]: \{3a^2 - 2a^3, 3a^3 - 6a^2 + a^3, 1 - 6a + 3a^2, -2 + 3a, 1\}$$

#I[4]:: LExpt[t,x]

#0[4]: {1,2,3,4,5}

XPolynom

Polygonal numbers

triangular numbers - pentagonal numbers - number theory functions
figurate numbers

S.Wolfram and P.Leyland

Jan 1982

Polynom[k,n]

k th *n*-gonal number (*n*:3 yields triangular numbers; *n*:4 squares)

```
Polynom[k_n Natp[k],n_n Natp[n]] : \  
k ((n-2)k+4-n)/2
```

XPow

XPow

Simplification of powers

radicals - square root - canonical form - prime decomposition

S.Wolfram

Feb 1982

Reduce power of composite number to product of powers of primes

```
<Primep
($x_Intp[$x] & ~Primep[$x])~$y :: \
  Ap[Mult,Map[Ap[$1~($2 $y),$0],Nfac[$x]]]
```

Warning: Redefines Pow; Pow processed more slowly.

XPrime

XPrime

Potentially prime numbers

S.Wolfram and P.Leyland
Jan 1982

* Nfac; Prime; XPrimep

Fer [n]

n th Fermat number.

$$\text{Fer}[\$n_Natp[\$n]] : B[2] \sim 2^{\$n} + 1$$

Mer [n]

n th Mersenne number.

$$\begin{aligned} _Mer[Init] &:: \langle XPrimep \\ \text{Mer}[\$n_Primep[\$n]] &: B[2] \sim \$n - 1 \end{aligned}$$

Inum [n]

n th I number.

$$\text{Inum}[\$n_Natp[\$n]] : (B[10] \sim \$n - 1) / 9$$

Jnum [n]

n th J number.

$$\text{Jnum}[\$n_Natp[\$n]] : B[10] \sim \$n + 1$$

XPrimep

XPrimep

Primality testing

S.Wolfram

Jul 1981

Primep[*expr*]

yields 1 if *expr* is a prime number, and 0 otherwise.

```
Primep[$expr] := Natp[$expr] & P[Rp[Plus,Nfac[$expr]][2]=1]
```

Future enhancements: Should use a serious primality testing algorithm.

XProj

Projection manipulation

S.Wolfram

Jul 1981

PCat [*proj1*, *proj2*]

yields a projection whose filters are the concatenation of those in the projections *proj1*, *proj2* (which must have the same projector).

```
PCat[$x_>Projp[$x], $y_>(Projp[$y] & $y[0]=$x[0])] :: \
  Proj[$x[0], {S[$x, $x[0]->Np], S[$y, $y[0]->Np]}]
```

Pap [*temp*, *proj1*, *proj2*, ...]

treats the filters of the projections *proj1*, *proj2*, ... as lists, and applies the template *temp* to them, yielding a projection with the resulting list as its filters.

```
_Pap[Smp]: {&}
Pap[Stemp, $$proj_>(Ap[And, Map[Projp, {$$proj}]] & \
  Ap[Eq, Map[$X[0], {$$proj}]])] :: \
  Proj[$$proj[1][0], Ap[temp, Map[S[$X1[0]->Np, {$$proj}]]]]
```

XPrtable

Tabular output

printing - tabulation - tabular data - boxes - ruled
columnar - report generation - formatting - matrices

J.Greif

Nov 1981

vline [w]

make a vertical line of length w

```
vline[$w] :: vthing[$w, ""]
```

vblank [w]

make a vertical blank of length w

```
vblank[$w] :: vthing[$w, " "]
```

vthing [w, x]

make a column of vertical things x of height w

```
vthing[$w, $x] :: Fmt[{$xyzzzy}: {0, $xyzzzy}}, Repl[ $x, $w]]
```

hline [w]

make a horizontal line of length w

```
hline[$w] :: hthing[$w, "-"]
```

hthing [w, x]

make a row of horizontal things x of length w

```
hthing[$w, $x] :: Fmt[Null, Repl[ $x, $w]]
```

box [x]

make a box around expression x

```
box[$x] :: (Lcl[ $h, $v, $f]; hv[ $h, $v, $x]; \
Fmt[{{1, 0}, {-1, 0}, {Inf, 0}, {1, -Inf}, {1, Inf}, {0, Inf}, {0, -Inf}}, \
 $x, vline[ $v], vline[ $v], \
 hline[ $h], hline[ $h], hline[1], hline[1]])
```

vbar [x]

print a list x horizontally with bars separating the elements

```
vbar[$x_ Listp[$x]] :: bar[$x, vline]
bar[$x_ Listp[$x], $z] :: (Lcl[ $v, $h, $s, $v1]; hv[ $h, $v, $x]; \
 $s: S[Ar[Len[$x], If[ $i=Len[$x], [Null, $x[ $i], Null], \
 [Null, $x[ $i], Null, $z[ $v]]]], { $su }-> $su, {0}])
```

col [x]

print list x horizontally inside box with bars separating the elts

```
col[$x_ Listp[$x]] :: box[Fmt[{$s}: {$s, 0}}, vbar[$x]]
```

topbot [x]

print object x with a horizontal bar above and below


```
topbot[$x] := (Lcl[Xf,Xh]; Xf:Prdsp[$x]; Xh:Xf[2]+Xf[3]+2; \
Fmt[{{1,0},{1,-Inf},{1,Inf},{0,Inf},{0,-Inf}},$x,\
hline[Xh],hline[Xh],hline[1],hline[1]])
```

hv[h,v,x]

find height v and width h of expression x

```
hv[$h,$v,$x] := (Lcl[Xf]; Xf:Prdsp[$x]; $h:Xf[2]+Xf[3]+2; \
$ v:Xf[4]+Xf[5]+1; )
```

colmat[x]

print out columnated, and boxed matrix, i.e. a table

```
colmat[$x_Fullip[$x,2]] := (Lcl[Xh,Xl,Xv,Xt,Xq]; Xt:Trans[$x]; \
Xl:Ar[Len[Xt], Fmt[{$s: {0,-2$s}},S[Xt[$i]},{s$u}->s$u,{0}]]); \
col[Xl])
```

Prmat[x]

print out matrix in 2-D form

```
Prmat[$x_Fullip[$x,2]] := (Lcl[Xh,Xl,Xv,Xt,Xq]; Xt:Trans[$x]; \
Xl:Ar[Len[Xt], Fmt[{$s: {0,-2$s}},S[Xt[$i]},{s$u}->s$u,{0}]]); \
Fmt[{$s: {s,0}},bar[Xl,vblank]])
```

```
#I[1]:: <XPrtable
```

```
#I[2]:: Ar[{3,3},Pow]
```

```
#O[2]: {{1,1,1},{2,4,8},{3,9,27}}
```

```
#I[3]:: Prmat[X]
```

```
1 1 1
```

```
#O[3]: 2 4 8
```

```
3 9 27
```

```
#I[4]:: colmat[O2]
```

```
#O[4]:
```

1	1	1
2	4	8
3	9	27

XP_{smp}

XP_{smp}

Heuristic simplification of rational forms

S.Wolfram

Jul 1981

P_{smp}[*expr*]

usually yields a usefully simplified form of the rational expression *expr*.

```
Psmp[Se] :: Map[Fac, Col[Ex[Se]]]
Psmp ⊥ Ldist
```


XQuadres

XQuadres

Quadratic residues

Number theory – modular arithmetic – rings

P.Leyland and S.Wolfram

Feb 1982

Quadres [p]

yields a list of all quadratic residues modulo p .

```
Quadres[$p_] := Union[Cat[Ar[{{0, Floor[$p/2]}}, Mod[$1^2, $p]]
```

XRAr

Array generation from recurrence relations

Sequence generation – congruential random number generation
recurrence relations

S.Wolfram

Aug 1982

RAr [*n*, *temp*, *start*]

generates *n* terms in a sequence starting from *start* by applying the template *temp* to each preceding term.

RAr[[*n*, *Temp*[[*n*], *Start*]] :: \

(Lc[[*t*]; t:*Start*; Ar[[*n*, (Lc[[*to*]; to:t; t:Ar[[*Temp*, {t}]; to]])

#I[1]:: <XRAr

#I[2]:: RAr[5, f[\$1], 1]

#O[2]: {1, f[1], f[f[1]], f[f[f[1]]], f[f[f[f[1]]]] }

#I[3]:: RAr[10, \$1+2, 1]

#O[3]: {1, 3, 5, 7, 9, 11, 13, 15, 17, 19 }

#I[4]:: RAr[10, Mod[13 \$1+7, 11], 1]

#O[4]: {1, 9, 3, 2, 8, 7, 10, 5, 6, 8 }

#I[5]:: RAr[3, \$1/(1+\$1), a]

#O[5]: { a, $\frac{a}{1+a}$, $\frac{a}{(1+a)(1+\frac{a}{1+a})}$ }

XRamp

XRamp

Ramp function

S.Wolfram

Jul 1981

Ramp [x]

represents the unit ramp function.

$$\text{Ramp}[x] : (\text{Abs}[x] + x)/2$$

XRandC

Continuous random number generation

S.Wolfram

Jul 1981

NRand[(*x:0*), (*sd:1*)]

generates a random number from a normal (Gaussian) distribution with mean *x* and standard deviation *sd*.

```
NRand_:=Tier
NRand[]:=NRand[0,1]
NRand[$mean]::NRand[$mean,1]
NRand[$mean,$sd]::\
  N[$mean + $sd Sqrt[-2 Log[Rand[]]] Cos[(IN[2Pi]) Rand[]]]
```

BRand[*rho*]

generates a pair of random numbers from a bivariate normal distribution with zero mean, unit variance and correlation coefficient *rho*.

```
BRand[$rho]::(Lc[{X1}; \
  {X1:NRand[], N[$rho X1 + Sqrt[1-$rho^2] NRand[]}]
```

ERand[*theta*]

generates a random number from the exponential distribution $\text{Exp}[-x/\theta]$.

```
ERand[$theta]::N[-$theta Log[Rand[]]]
```

ARRand[*a, b, dist, maxdist*]

uses an acceptance-rejection method to generate a random number between *a* and *b* from the distribution defined by the template *dist* whose maximum value is not less than *maxdist*. Number of attempts necessary is proportional to $(b-a)$ *maxdist*.

```
ARRand[$a,$b,$dist,$maxdist]::(Lc[{Xy}; \
  Loop[{Xy:N[$a + ($b-$a) Rand[]], \
  Rand[] >= N[Ap[$dist, {Xy}]/$maxdist]}; Xy)
```


XRandD

Discrete random number generation

S.Wolfram

Jul 1981

IRand [*n*]generates a random integer from a uniform distribution between 0 and *n*-1.`IRand[$n] :: Floor[Rand[$n]]`**PNorm** [*list*]

yields a normalized list of probabilities from a list of relative frequencies.

`PNorm[$list] :: N[$list/Rp[Plus,$list]]`**PCum** [*list*]

yields a list of cumulative probabilities.

`PCum[$list] :: (Lc[Xt]; Xt:0; Map[Xt:Xt+$Xi; Xt, PNorm[$list]])`**DRand** [{*cp1*, *cp2*, ...}]yields a random position in the list with distribution determined by the cumulative probabilities *cp_i*.

Misfeatures: Not optimal algorithm

`DRand[$list] :: (Lc[Xx]; Xx:Rand[]; Pos[$1_>$1>Xx, $list, 2, 1][1, 2, 1])`

XRandL

XRandL

Random list element selection

S.Wolfram

Jul 1981

LRand [list]

yields one of the elements of list, randomly chosen with equal probabilities.

```

_LRand[Init] :: <XRandD
LRand[$list, Listp[$list]] :: Ent[$list, {1+IRand[Len[$list]]}]

```

LDRand [list, prob]

yields one of the elements of list, randomly chosen with relative frequencies given by prob.

```

_LDRand[Init] :: <XRandD
LDRand[$list, $prob, (Contp[$prob] & Len[$prob]=Len[$list])] :: \
  $list[DRand[PCum[$prob]]]

```

ORand [list]

yields a random reordering of list.

```

_ORand[Init] :: <XPerm
ORand[$list] :: (Lc[X]; X:Ar[Len[$list], 'Rand[]]; \
  Rpper[Fiper[X], Sort[X]], $list)

```


XRepar t

XRepar t

Graphical part replacement

Repar t [*expr*]

replaces a graphically-selected part of *expr*.

```
Repar t[Se] :: Ap[Set, {Ap[Se, {L[Se]}], Rd["new part: "]}]
_Repar t[Smp]:0
```

XRev

XRev

Symbolic identities for list operations

S.Wolfram
Jul 1981

`Rev[Rev[$x]] : $x`

`Cyc[Cyc[$list,$n1],$n2] :: Cyc[$list,$n1+$n2]`

XRnd

XRnd

Integer rounding

truncation - type casting - entier - floor - ceiling

S.Wolfram
Sep 1982

Rnd[n]

rounds n to the nearest integer.

Rnd[\$n, Nump[\$n]] :: Floor[\$n+0.5]

- #I[1]:: <XRnd
- #I[2]:: Rnd[2.2]
- #O[2]: 2
- #I[3]:: Rnd[2.7]
- #O[3]: 3
- #I[4]:: Rnd[-3.6]
- #O[4]: -4

XRot2

XRot2

Rotations in two dimensions

S.Wolfram
Jul 1981

Vec2p[*expr*]
tests whether *expr* represents a two-dimensional vector.
Vec2p[\$expr] :: Contp[\$expr] & Len[\$expr]=2

RotM2[*theta*]
yields a matrix representing a two-dimensional rotation through an angle of *theta* radians.
RotM2[\$theta] : {{Cos[\$theta], Sin[\$theta]}, {-Sin[\$theta], Cos[\$theta]}}

Rot2[*vec*, *theta*, (*pt*: {*θ*, *θ*})]
rotates the two-dimensional vector *vec* about the point *pt* through an angle of *theta* radians.
Rot2 Tier
Rot2[\$vec, Vec2p[\$vec], \$theta] : RotM2[\$theta].\$vec
Rot2[\$vec, Vec2p[\$vec], \$theta, \$pt, Vec2p[\$pt]] :: \
\$pt + RotM2[\$theta].(\$vec-\$pt)

XRot3

Rotations in three dimensions

S.Wolfram

Jul 1981

Vec3p[*expr*]tests whether *expr* represents a three-dimensional vector.

```
Vec3p[expr] :: Contp[expr] & Len[expr]=3
```

RotM3[*phi*, *theta*, *psi*]yields a matrix representing a three-dimensional rotation specified by the Euler angles *phi*, *theta*, *psi*.

```
RotM3[phi, theta, psi] : \
  {{Cos[psi] Cos[phi] - Cos[theta] Sin[phi] Sin[psi], \
   Cos[psi] Sin[phi] + Cos[theta] Cos[phi] Sin[psi], \
   Sin[psi] Sin[theta]}, \
  {-Sin[psi] Cos[phi] - Cos[theta] Sin[phi] Cos[psi], \
   -Sin[psi] Sin[phi] + Cos[theta] Cos[phi] Cos[psi], \
   Cos[psi] Sin[theta]}, \
  {Sin[theta] Sin[phi], -Sin[theta] Cos[phi], Cos[theta]}}
```

Rot3[*vec*, *phi*, *theta*, *psi*, (*pt*: {0,0,0})]rotates the three-dimensional vector *vec* about the point *pt* through the Euler angles *phi*, *theta*, *psi*.

```
Rot3_Tier
Rot3[vec_Vec3p[vec], theta] : RotM3[theta].vec
Rot3[vec_Vec3p[vec], theta, spt_Vec3p[spt]] :: \
  spt + RotM3[theta].(vec-spt)
```

XRpoly

Random polynomial generation

S.Wolfram

Jul 1981

Ranup[(n:10)]generates a random univariate polynomial of size n .

```

Ranup_Tier
Ranup[Sn_Natp[Sn]] :: \
  Ex[Rex[Sn, {x,1}, {2,1}, {-1,1}], {Plus,1,-7}, {Mult,1,-5}]]]
Ranup[] :: Ranup[10]

```

Ranmp[(n:10),(nx:3)]generates a random polynomial of size n in an average of nx variables.

```

Ranmp_Tier
Ranmp[Sn_Natp[Sn], Snx_Natp[Snx]] :: \
  Ex[Rex[Sn, Cat[Ar[{10,10+Snx-1}], ImpI[{SX1}]], {2,1}, {-1,1}], \
    {Plus,1,-7}, {Mult,1,-5}]]]
Ranmp[Sn_Natp[Sn]] :: Ranmp[Sn,3]
Ranmp[] :: Ranmp[10,3]

```

#I[1]:: <XRpoly

#I[2]:: Ranup[10]

#O[2]: $168x^3 + 240x^4 + 78x^5$

#I[3]:: Ranup[10]

#O[3]: $2 + 5x + 8x^2$

#I[4]:: Ranmp[10,3]

#O[4]: $12 + 3a + 2b + 4c + 8a^2c$

#I[5]:: Ranmp[10,3]

```

#O[5]: 12b + 10a b + 4b c + 96 a c + 32 a c + 80 a c + 16 a b c
      + 2 b

```


XScan

Scan function

scan - search - mu function - find - first

S.Wolfram

Jan 1982

Scan[*temp*, *n*]yields a list of the first *n* positive integers satisfying the criterion *temp*.

```

_Scan[Smp]: {Inf, 0}
Scan_Tier
Scan[Stemp, Sn, Natp[Sn]] := (Lc[Xt, Xi, Xf]; For[Xi:0; Xt:1, \
Xi<Sn, Inc[Xt], If[Re[Ap[Stemp, {Xt}]], Xf[Inc[Xi]:Xt]; Xf)

```

Future enhancements: Extend to k-tuples of integers

```

#I[1]:: <XScan
#I[2]:: Scan[Intp[N[Sqrt[$1]]], 5]
#O[2]: {1, 4, 9, 16, 25}
#I[3]:: Scan[$1^2-4$1>3, 2]
#O[3]: {5, 6}

```

XSerSol

Series solution of differential equations

power series - ordinary differential equations

J.Greif

Aug 1982

PsSol [*dexpr*, *serord*, *bc*, (*dep*:*y*), (*ind*:*x*)]

solves the differential equation $dexpr = 0$ for the dependent function *dep* as a function of the independent variable *ind* in power series form to order *serord*, given boundary conditions *bc* expressed as a list of expressions to be set = 0

```

PsSol - Tier
_PsSol[Smp]:{0,Inf,0,0,Inf}
PsSol[$dex,$so,$bc]:PsSol[$dex,$so,$bc,y,x]
PsSol[$dex,$so,$bc,$dep,$ind]:(Lcl[Xy,Xdep,Xex,Xl,Xbc,Xl,Xa,qdiff]; \
  Xex:S[$dex,$dep->Xdep]; Xbc:S[$bc,$dep->Xdep]; \
  Xdep-Tier; Xy-Tier; \
  Xy[$x,$n]:Ps[1,$x,0,$n,{[S]:Xa[S]}]; \
  Xdep:Xy[$ind,$so]; \
  only now do we know indep var so can define qdiff\
  qdiff[Xdep[$y]]:D[Xdep[$ind],{$ind,1,$y}]; \
  qdiff[D[Xdep[$y],{$y,$n,$z}]]:D[Xdep[$ind],{$ind,$n+1,$z}]; \
  qdiff[$x]:D[$x,{$ind,1,$ind}]; \
  Xex:Si[Xex,Hold[D[Xdep[$x],{$x,$n,$z}]]-> \
    D[Xy[$ind,$so],{$ind,$n,$z}]]; \
  next line gets around incomplete simplification bug\
  Xex:S[Xex,DPAT-->0]; \
  Xex:Si[Xex,Xdep[$ind]->Xy[$ind,$so]; \
  make a properly truncated Ps if still a sum of terms\
  If[P[Xex[0]=Hold[Plus]],Xex:Ps[Xex,$ind,0,$so]; \
  Xbc:Si[Xbc,Hold[D[Xdep[$x],{$x,$n,$z}]]-> \
    D[Xy[$ind,$so],{$ind,$n,$z}]]; \
  next line gets around incomplete simplification bug\
  Xbc:Si[Xbc,DPAT-->0], Hold[Ps[$$x]]-->Ax[Ps[$$x]]; \
  Xbc:Si[Xbc,Xdep[$x]->Lim[Xy[$ind,$so,$ind,$x]]; \
  Xex is a Ps and Xbc is an expr or list \
  Xl:If[Listp[Xbc],Ldist[Xbc=0],{Xbc=0}]; \
  Xl:Union[Cat[Xl,Ldist[Xex[5]=0]]; \
  Xs:Sol[Xl,Cat[Ar[{$,so},Xa]]; \
  Ps[1,$ind,0,$so,Ar[{$,so},S[Xa[S]],Xs]]

```

Input niceties -- the differential expression can be entered in the form

$$f[x] y' + g[x] y'' + h[x] y - i[x]$$

```

Sxset["",qdiff,2]
_qdiff[Pr,$x]:Sx["",{$x},2]

```

Defeat encasement extension

```

_Ps[Extr,qdiff]:qdiff
_Ps[Extr,Pr]:Pr
_Ps[Extr,Eq]:Eq
_Ps[Extr,Hold]:Hold
_Ps[Extr,Rep]:Rep
_Ps[Extr,Sl]:Sl

```

#I[1]: <XSerSol

#I[2]: de: Cos[x] y' + Exp[-3x] y'' + Sin[x] y - Sin[14x]

#O[2]: e y Sin[x] + (y') Cos[x] + ((y'')) Exp[-3x] - Sin[14x]

#I[3]:: bc: {y[0]-2, y[0]'-1}

#O[3]:: {-2 + y[0], -1 + (y[0]')}

#I[4]:: Psol[de,8,bc]

$$\#O[4]:: 2 + x - \frac{x^2}{2} + \frac{5x^3}{3} + \frac{29x^4}{12} - \frac{515x^5}{24} - \frac{10091x^6}{240} + \frac{59567x^7}{840} + \frac{1448983x^8}{6720}$$

Misfeatures: Has fixed point of expansion 0. Assumes Taylor series will work. Thus solution must exist and be finite at $x=0$.

XSets

XSets

Elementary finite set theory

S.Wolfram
Jul 1981

Sets are represented as ordered lists with no repeated elements. A list may be placed in this canonical form by Union[list]. Once in this form, the equivalence of two sets is determined by Eq.

Union[set1, set2, ...]
forms the union of the sets set1, set2, ...

Inter[set1, set2, ...]
forms the intersection of the sets set1, set2, ...

Cmpl[set, univset]
yields the complement of set with respect to the universal set univset.

```
Cmpl[$list_>Listp[$list], $ulist_>Listp[$ulist]] :: \
  Cat[Ar[Len[$ulist], $ulist, ~In[$X1, $list]]]
```

Sub[set, crit]
yields a subset of set all of whose elements satisfy the condition crit.

```
Sub[$set, $crit] :: Flat[Ar[Len[$set], , , $crit]]
```

Subp[sub, set]
yields 1 if sub is a subset of set and 0 if it is not.

```
Subp[$sub, $set] :: Len[Union[$sub, $set]]-Len[Union[$set]]
```

Disjp[set1, set2]
yields 1 if the sets set1 and set2 are disjoint (have no elements in common).

```
Disjp[$set1, $set2] :: P[Inter[$set1, $set2]= {}]
```

Multset[set1, set2, ...]
forms the product set of the seti (set of all possible ordered n-tuples of elements from n sets).

```
Multset[$$set] :: Flat[Outer[List, $$set], Len[List[$$set]]-1]
```

Powset[set]
forms the set of all possible subsets of set (power set).

```
Powset[$l_>Contp[$l]] :: Flat[LoI[Xf, Xg]; S[Dist[Ap[Xg, Ar[Len[$l], \
  Xf[{}], {$l[$X1]}]]], {Xg, Xf, List, Xg}], Xg->Cat], Len[$l]-1]
```

```
#I[1]:: <XSets
#I[2]:: s1:Union[{a,b,a,c,d}]
#O[2]: {a,b,c,d}
```



```
#I[3]:: s2: {c, e, f}
#O[3]: {c, e, f}
#I[4]:: s3: Ar[5]
#O[4]: {1, 2, 3, 4, 5}
#I[5]:: Union[s1, s2, s3]
#O[5]: {1, 2, 3, 4, 5, a, b, c, d, e, f}
#I[6]:: Inter[s1, s2]
#O[6]: {c}
#I[7]:: Cmpl[s2, s1]
#O[7]: {a, b, d}
#I[8]:: Sub[s3, Evenp]
#O[8]: {2, 4}
#I[9]:: Subpl{a, c}, s1]
#O[9]: 1
#I[10]:: Disjp[s1, s3]
#O[10]: 1
#I[11]:: Multset[s1, s2]
#O[11]: {{{a, c}, {a, e}, {a, f}}, {{b, c}, {b, e}, {b, f}}, {{c, c}, {c, e}, {c, f}},
        {{d, c}, {d, e}, {d, f}}}
#I[12]:: Powset[s2]
#O[12]: {{{}, {f}, {e}, {e, f}, {c}, {c, f}, {c, e}, {c, e, f}}
```

XSetsSX

Set theory notation

S.Wolfram
Jul 1981

$a \cup b \cup c \dots$ or $a \cup b \cup c \dots$
stands for Union[a,b,c,...].
Sxset["U",Union,3]

$a \cap b \cap c \dots$ or $a \cap b \cap c \dots$
stands for Inter[a,b,c,...].
Sxset["I",Inter,3]

$a \setminus b$ or $a \setminus b$
stands for Cmpl[b,a].
Sxset["\","Cmpl",5]

$a \subset b$ or $a \subset b$
stands for Subp[a,b].
Sxset["C",Subp,4]

$a * b * c \dots$ or $a * b * c \dots$
stands for Multset[a,b,c,...].
Sxset["~*",Multset,3]

\tilde{a} or \tilde{a}
stands for Powset[a].
Sxset["^^",Powset,1]

XSfctI

XSfctI

Subfactorial

derangements - rencontres numbers

S.Wolfram

Sep 1982

SfctI [n]

subfactorial of n (number of derangements of n objects).

SfctI[\$n_] := Natp[\$n] : \$n SfctI[\$n-1] + (-1)^\$n
SfctI[1] := 1

[Sloane: Handbook of Integer Sequences, sect. 3.13]

XShowtime

XShowtime

Display of simplification times

S.Wolfram

Jul 1982

Showtime

causes simplification times for each output line to be printed.

```
Showtime :: (Pre :: (Lcl[X1];Pr[Time[X1:$1]];If[Valp[X1,X1,1]))
```

```
Showtime1 :: (Post :: (Pr[N[Last[#T]];S1))
```


XSign

XSign

Sign simplification rules

S.Wolfram

Jul 1981

```
Sign[$x $$x] :: Sign[$x] Sign[$$x]
Sign[$x^($n_?Evenp[$n])] : 1
Sign[Abs[$x]] : 1
```

XSol

XSol

Inverses of elementary transcendental functions

S.Wolfram
Jul 1981

```

Sol[Exp[$x]=$y,$x] :: Sol[$x=Log[$y],$x]
Sol[Log[$x]=$y,$x] :: Sol[$x=Exp[$y],$x]
Sol[Sin[$x]=$y,$x] :: Sol[$x=Asin[$y],$x]
Sol[Cos[$x]=$y,$x] :: Sol[$x=Acos[$y],$x]
Sol[Tan[$x]=$y,$x] :: Sol[$x=Atan[$y],$x]
Sol[Asin[$x]=$y,$x] :: Sol[$x=Sin[$y],$x]
Sol[Acos[$x]=$y,$x] :: Sol[$x=Cos[$y],$x]
Sol[Atan[$x]=$y,$x] :: Sol[$x=Tan[$y],$x]
Sol[Sinh[$x]=$y,$x] :: Sol[$x=Asinh[$y],$x]
Sol[Cosh[$x]=$y,$x] :: Sol[$x=Acosh[$y],$x]
Sol[Tanh[$x]=$y,$x] :: Sol[$x=Atanh[$y],$x]
Sol[Asinh[$x]=$y,$x] :: Sol[$x=Sinh[$y],$x]
Sol[Acosh[$x]=$y,$x] :: Sol[$x=Cosh[$y],$x]
Sol[Atanh[$x]=$y,$x] :: Sol[$x=Tanh[$y],$x]
Sol[Gd[$x]=$y,$x] :: Sol[$x=Agd[$y],$x]
Sol[Agd[$x]=$y,$x] :: Sol[$x=Gd[$y],$x]

```


XSpare

XSpare

Removal of almost all values

S.Wolfram
Jul 1981

Spare[*v1, v2, ...*]
removes all values except those of *v1, v2, ...*

```
_Spare[Smp]:0  
Spare[SSx] :: (Lcl['Spare,SSx]; Set[])
```

```
#I[1]:: <XSpare  
#I[2]:: a:b:c:1  
#O[2]: 1  
#I[3]:: Spare[a]  
#O[3]: Spare[' a]  
#I[4]:: {a,b,c}  
#O[4]: {1,b,c}
```

XStat

Univariate statistics

S. Wolfram

Jul 1981

Mean[list]gives the mean of the values in *list*.
$$\text{Mean}[list] :: \text{Ap}[\text{Plus}, list] / \text{Len}[list]$$
GMean[list]gives the geometric mean of the values in *list*.
$$\text{GMean}[list] :: \text{N}[\text{Ap}[\text{Mult}, list^{(1/\text{Len}[list])}]]$$
HMean[list]gives the harmonic mean of the values in *list*.
$$\text{HMean}[list] :: \text{N}[\text{Len}[list] / \text{Ap}[\text{Plus}, \text{Map}[1/\$, list]]]$$
Med[list]gives the median of the values in *list*.
$$\begin{aligned} \text{Med}[list \rightarrow \text{Oddp}[\text{Len}[list]]] &:: \text{Sort}[list][(\text{Len}[list]+1)/2] \\ \text{Med}[list \rightarrow \text{Evenp}[\text{Len}[list]]] &:: \backslash \\ & \quad (\text{Lc}[X]; X: \text{Sort}[list]; (X[\text{Len}[list]/2] + \backslash \\ & \quad \quad X[\text{Len}[list]/2+1])/2) \end{aligned}$$
MD[list]gives the mean deviation of the values in *list*.
$$\text{MD}[list] :: \text{Mean}[\text{Abs}[list - \text{Mean}[list]]]$$
Var[list]gives the variance of the values in *list*.
$$\text{Var}[list] :: \text{Mean}[(list - \text{Mean}[list])^2]$$
SD[list]gives the standard deviation of the values in *list*.
$$\text{SD}[list] :: \text{N}[\text{Sqrt}[\text{Var}[list] \text{Len}[list] / (\text{Len}[list] - 1)]]$$
Range[list]gives the range of the values in *list*.
$$\text{Range}[list] :: \text{Ap}[\text{Max}, list] - \text{Ap}[\text{Min}, list]$$
RMS[list]gives the root mean square of the values in *list*.
$$\text{RMS}[list] :: \text{N}[\text{Sqrt}[\text{Mean}[list^2]]]$$
Fract[list, r]yields the *r* fractile of the values in *list*.


```
_Fract[list] :: <L>tp
Fract[$list,$r] :: Ltp3[Sort[$list],$r Len[$list]]
```

Q1 [*list*]

yields the first quartile of the values in *list*.

```
Q1[$list] :: Fract[$list,1/4]
```

Q3 [*list*]

yields the third quartile of the values in *list*.

```
Q3[$list] :: Fract[$list,3/4]
```

QD [*list*]

yields the quartile deviation of the values in *list*.

```
QD[$list] :: (Q3[$list] - Q1[$list])/2
```

Mom [*list*, *n*]

yields the *n*th moment of the values in *list*.

```
Mom[$list,$n] :: Mean[$list^$n]
```

Cmom [*list*, *n*]

yields the *n*th central moment of the values in *list*.

```
Cmom[$list,$n] :: Mean[($list-Mean[$list])^$n]
```

Cum [*list*, *n*]

yields the *n*th cumulant of the values in *list*.

```
Cum[$list,1] :: Mean[$list]
Cum[$list,2] :: Var[$list]
Cum[$list,3] :: Cmom[$list,3]
Cum[$list,4] :: Cmom[$list,4] - 3 Cmom[$list,2]^2
Cum[$list,5] :: Cmom[$list,5] - 10 Cmom[$list,3] Cmom[$list,2]
```

Skew [*list*]

gives the coefficient of skewness of the values in *list*.

```
Skew[$list] :: Cmom[$list,3]/SD[$list]^3
```

Kurt [*list*]

gives the coefficient of kurtosis of the values in *list*.

```
Kurt[$list] :: Cmom[$list,4]/SD[$list]^4
```

Excess [*list*]

gives the coefficient of excess of the values in *list*.

```
Excess[$list] :: Cmom[$list,4]/SD[$list]^4 - 3
```

Qskew [*list*]

gives the quartile coefficient of skewness of the values in *list*.

```
Qskew[$list] :: (Q3[$list] - 2 Med[$list] + Q1[$list]) / \
                (Q3[$list] - Q1[$list])
```

Expc [*op*, *list*]

yields the expectation value of the operator represented by the template *op* on the values in *list*.

956

```

_ExpC[Smp]: {0, Inf}
ExpC[Sop, $list] :: Rel[Ap[Plus, Map[$op, $list]]] / Len[$list]

```

Char [list, x]

yields the characteristic function of the values in *list*.

```
Char[$list, $x] :: ExpC[Exp[I $! x], $list]
```

```

#I[1]:: <XStat
#I[2]:: t:Ar[20, 'Rand[]]
#O[2]:  { .7512881, .9799641, .3878826, .1427415, .8638817, .1446294, .4414654,
          .2389524, .664948, .1285033, .1148291, .243175, .5178353, .4259514,
          .6894183, .9818886, .82981599, .4496743, .86934847, .4439974 }

#I[3]:: Post:N
#O[3]:  N
#I[4]:: Mean[t]
#O[4]:  .4238211
#I[5]:: GMean[t]
#O[5]:  .3851757
#I[6]:: HMean[t]
#O[6]:  .1837451
#I[7]:: Med[t]
#O[7]:  .4337884
#I[8]:: SD[t]
#O[8]:  .2941784
#I[9]:: Range[t]
#O[9]:  .9581481
#I[10]:: RMS[t]
#O[10]: .5118389
#I[11]:: Q1[t]
#O[11]: .1427415
#I[12]:: Q3[t]
#O[12]: .6894183
#I[13]:: Skew[t]
#O[13]: .4857557
#I[14]:: Kurt[t]
#O[14]: 1.824871
#I[15]:: ExpC[Exp, t]
#O[15]: 1.592872

```


XStatus

Status information

S.Wolfram
Jul 1981

Status

prints status information.

```
Status :: \  
  (Pr["Memory used so far:",State][3],"blocks"); \  
  Pr["Total CPU time so far:",N[Clock][2]],"seconds"); \  
  Pr["Real time so far:",Clock][1],"seconds"); \  
  Pr["Number of input lines:",Len[#1]]; \  
  Pr["Number of user symbols:",Len[Cont[]]]; \  
  Pr["Total number of symbols:",Len[Cont[,1]])
```

XStr0

Basic character string manipulation

S.Wolfram
Jul 1981

Char [str, i]

yields the *i* th character in the string *str*.

```
Char[$str, $i] := (0 < $i < Len[$str]) :: Impl[Expl[$str][$i]]
```

CLen [str]

gives the number of characters in the string *str*.

```
CLen[$str] := Len[Expl[$str]]
```

CRev [str]

reverses the string *str*.

```
CRev[$str] := Impl[Rev[Expl[$str]]]
```

CJoin [str1, str2, ...]

concatenates the strings *str1*, *str2*, ...

```
CJoin[$$s] := Impl[Ap[Cat, Map[Expl, List[$$s]]]]
```

- #I[1]:: <XStr0
- #I[2]:: t: "A character string"
- #O[2]:: "A character string"
- #I[3]:: Char[t, 4]
- #O[3]:: h
- #I[4]:: Char[t, 2]
- #O[4]:: " "
- #I[5]:: CLen[t]
- #O[5]:: 18
- #I[6]:: CRev[t]
- #O[6]:: "gnirts retcarahc A"
- #I[7]:: CJoin[X, t, t]
- #O[7]:: "gnirts retcarahc AA character stringA character string"

XStr1

XStr1

Further character string manipulation

S.Wolfram
Jul 1981

<XList1

CRep[*str1*, *str*, *i*]

replaces the character at position *i* in *str* by the string *str1*.

```
CRep[$str1,$str,$i] :: (Lcl[XI]; XI:Exp1[$str]; XI[$i]:Exp1[$str1]; \
Impl[Flat[XI]])
```

CIns[*str1*, *str*, *i*]

inserts the character string *str1* at position *i* in *str*.

```
_CIns[Init] :: <XList0
CIns[$str1,$str,$i] :: Impl[Ins[Exp1[$str1],Exp1[$str],$i]]
```

CPos[*form*, *str*]

yields a list of the positions of the substring *form* in the string *str*.

```
CPos[$form,$str] :: LPos[Exp1[$form],Exp1[$str]]
```

CS[*str*, *rep1*, *rep2*, ...]

applies successively the replacements *repi* for substrings of the string *str*. Each replacement is used until it is no longer applicable. The special "generic character" represents any single character.

```
CS[$str,$$reps] :: Impl[Ap[LS,Cat[{Exp1[$str]}, \
S[Map[Exp1[$i[1]]->Exp1[$i[2]],List[$$reps]], \
({Exp1["$"] [1])->$X1}]]]
```

- #I[1]:: <XStr1
- #I[2]:: s:"the cat in the hat"
- #O[2]:: "the cat in the hat"
- #I[3]:: CRep[RRRR,s,5]
- #O[3]:: "the RRRRat in the hat"
- #I[4]:: CIns[" not",s,8]
- #O[4]:: "the cat not in the hat"
- #I[5]:: CPos[the,s]
- #O[5]:: {1,12}
- #I[6]:: CS[s,the->a,\$at->XXXXX]
- #O[6]:: "a XXXXX in a XXXXX"

XSubl

XSubl

Sublists

form sublists - unflatten - group - sequences

S.Wolfram

Jan 1982

Subl [*list*, *n*]generates a list of successive groups of *n* entries in *list*.

```
Subl[list, n] := Flatten[Table[list[[i + 1, i + n]], {i, 1, Length[list] - n + 1}]]
```

• UnFlat; Outer; Tri

#I[1]:: <XSubl

#I[2]:: t:Ar[8]

#O[2]:: {1, 2, 3, 4, 5, 6, 7, 8}

#I[3]:: Subl[t, 2]

#O[3]:: {{1, 2}, {2, 3}, {3, 4}, {4, 5}, {5, 6}, {6, 7}, {7, 8}}

#I[4]:: Subl[t, 7]

#O[4]:: {{1, 2, 3, 4, 5, 6, 7}, {2, 3, 4, 5, 6, 7, 8}}

XSum

XSum

Series summation

S.Wolfram
Jul 1981

Canonical forms for sums

Canonical form for limits

$$SSum[1] : Sum[se, {s1, s1, s2}] --> Sum[S[se, si -> si - s1 + 1, -1, 1], {si, 1, s2 - s1 + 1}]$$

Sums of rational functions

$$SSum[2] : Sum[se, {si, s1, s2}] --> Sum[Pf[se, si], {si, s1, s2}]$$

Simplification of sums

_Sum[Smp]:Inf

$$Sum[xx+ssx, {si, s1, s2}] :: Sum[xx, {si, s1, s2}] + Sum[ssx, {si, s1, s2}]$$

$$Sum[xx, {si -> (~In[si, xx]), s1, s2}] :: (s2 - s1 + 1) xx$$

$$Sum[ssn xx, {si -> (~In[si, ssn], s1, s2}] :: ssn Sum[xx, {si, s1, s2}]$$

$$Sum[xx/(ssn sy), {si -> (~In[si, ssn], s1, s2}] :: Sum[xx/sy, {si, s1, s2}]/ssn$$

Sums of positive powers

$$Sum[si, {si, 1, sn}] : sn(sn+1)/2$$

$$Sum[si^(sm -> Natp[sm]), {si, 1, sn}] : (Ber[sm+1, sn+1] - Ber[sm+1])/(sm+1)$$

$$Sum[(-sa)^si si^(sm -> Natp[sm]), {si -> (~In[si, sa], 1, sn}] : \ ((-1)^sn Eul[sm, sn+1] - Eul[sm, 0])/2$$

Geometric progression

$$Sum[sa^si, {si -> (~In[si, sa], 1, sn}] : (sa^sn - 1)/(sa - 1)$$

Warning: Sum redefined to simplify its arguments

XSumPR

XSumPR

Special output form for Sum

S.Wolfram

Jul 1981

```

<XPR
_Sum[Pr][[Sexpr, {Svar, $start, $end}]] := \
Fmt[{{1, 0}, {3, 0}, {1, -1}, {1, 1}}, PSig, Sexpr, Svar=$start, $end]

```

```
#I[1]:: <XSumPR
```

```
#I[2]:: 1+Sum[1/i^a, {i, 1, Inf}]
```

```
#O[2]:: 1 + 
$$\sum_{i=1}^{\infty} \frac{1}{i^a}$$

```


XSympol

Symmetric polynomial generation

Sympol [n, x]generates a list of all symmetric polynomials in n variables $x[i]$.

```
Sympol[$n_?Natp[$n], $x] := (Lc1[$x]; \
  Ps[Ex[Prod[(1+$x[$i] $x), {x_i, 1, $n}]], $x, 0, $n][5])
```

XTEST

XTEST

General testing routines

S.Wolfram
Jul 1981

Basic test function

```
_Test[Smp]:0
Test[$f,$arg,$n] :: \
  Rpt[ti:Rel[$arg];Pr[];Pr[Rel[$f],tin];Pr[tout:Rel[Rp[$f,tin]]], $n]
```

Seed random numbers

```
Seed[$n] :: Rand[1,$n]
```

Random argument generators

numb[x]

positive or negative number of maximum modulus x.

```
numb[$x] :: -$x + Rand[2$x]
```

pnumb[x]

positive number of maximum size x.

```
pnumb[$x] :: Rand[$x]
```

int[x]

positive or negative integer of maximum modulus x.

```
int[$x] :: Gint[numb[$x]]
```

pint[x]

positive integer of maximum modulus x.

```
pint[$x] :: Gint[1+Rand[$x]]
```

nnint[x]

non-negative integer of maximum modulus x.

```
pint[$x] :: Gint[Rand[$x]]
```

expr[n]

expression of size n.

```
expr[$n] :: Rex[$n]
```

temp[n]

template of size n.


```
temp[$n] :: S[Rex[$n], x->$1, z->$2]
```

```
<XRpoly
```

poly[*n*]

univariate polynomial of size *n*.

```
poly[$n] :: Ranup[$n]
```

mpoly[*n*, *m*]

multivariate polynomial in about *m* variables of size *n*.

```
mpoly[$n, $m] :: Ranmp[$n, $m]
```

list[*form*, *n*]

length $\leq n$ list of *forms*.

```
list[$form, $n] :: Ar[Gint[Rand[$n]+0.3], `Rei[$form]]
list ⊃ Nosmp
```

rpt[*form*, *n*]

repetition of $\leq n$ occurrences of *form*.

```
rpt[$form, $n] :: Rp[Np, Rp[list, {$form, $n}]]
rpt ⊃ Nosmp
```

When Cons works for Rand, Cons all numerical routines above, and allow numbers to be found from exponential distribution, as obtained in a Cons routine (exponential distribution is more realistic than uniform in most cases).

XTeX

XTeX

Tensor expansion

indicial tensor calculus - tensor canonicalization
tensor symmetries

S.Wolfram
Jul 1981

TEx[*f*[*x*₁, *x*₂, ...], *reor*]

constructs a sum of tensor obtained by reordering the *x_i* and assigning the factors specified by the permutation symmetries *reor*.

```
<XPerm8
<XArperm
TEx_1Tier
TEx[$f[$$x], $reor] := (Lc1[X], Xo, Xt, Xc, Xf); Xf[Reor]:$reor; \
Xo:Ap[Xf, List[$$x]]; Xt:0; Ap[Plus, Map[Xi:Apper[$1, List[$$x]]; \
If[In[Xf, Xc:Ap[Xf, Xi]/Xo], 0, Inc[Xt, Xc]; Xc Ap[$f, Xi]], \
Arperm[Len[List[$$x]]]]/Xt)
```

#I[1]: <XTeX

#I[2]: TEx[f[a,b], Sym]

#O[2]:
$$\frac{f[a,b] + f[b,a]}{2}$$

#I[3]: TEx[f[a,b,c], Sym]

#O[3]:
$$\frac{f[a,b,c] + f[a,c,b] + f[b,a,c] + f[b,c,a] + f[c,a,b] + f[c,b,a]}{6}$$

#I[4]: TEx[f[a,b,c], Cyclic]

#O[4]:
$$\frac{f[a,b,c] + f[b,c,a] + f[c,a,b]}{3}$$

XTensor

XTensor

Tensor manipulations

S.Wolfram
Jul 1981

Tenp[f]
determines whether *f* is of type Tensor.
Tenp[\$f] :: P[_\$f[Type]=Tensor]

CO
denotes the basis coordinates.
CO : {r, theta, phi, t}

DIM
denotes the dimensionality of spacetime (default 4).
DIM:Len[CO]

KD[mu, nu]
denotes the Kronecker delta symbol.
_KD[Reor]:Sym
KD[\$mu,\$mu] : DIM
KD[\$mu,\$nu_(\$mu~\$nu)] : 0

Gm[mu, nu]
represents the metric tensor, assumed symmetric.
Gm_Tensor
_Gm[Reor]:Sym
Gm[\$mu_ Symbp[\$mu],`\$nu] : KD[\$mu,\$nu]

Gm[\$mu_ Symbp[\$mu], \$nu_ Symbp[\$nu]] \$f[\$\$i1,`\$mu,\$\$i2] : \
\$f[\$\$i1,\$nu,\$\$i2]
Gm[\$mu_ Symbp[\$mu], \$nu_ Symbp[\$nu]] \$f[`\$mu,\$\$i2] : \
\$f[\$nu,\$\$i2]
Gm[\$mu_ Symbp[\$mu], \$nu_ Symbp[\$nu]] \$f[\$\$i1,`\$mu] : \
\$f[\$\$i1,\$nu]
Gm[\$mu_ Symbp[\$mu], \$nu_ Symbp[\$nu]] \$f[`\$mu] : \$f[\$nu]
Gm[`\$mu,`\$nu] \$f[\$\$i1,\$mu,\$\$i2] : \$f[\$\$i1,`\$nu,\$\$i2]
KD[\$mu_ Symbp[\$mu], \$nu_ Symbp[\$nu]] \$f[\$\$i1,\$mu,\$\$i2] : \
\$f[\$\$i1,\$nu,\$\$i2]

Component Manipulations:
R4_Tensor; R2_Tensor; R_Tensor
_ChrL[Reor]:Sym[2,3]
_GU[Reor]:Sym; _GL[Reor]:Sym
_B2[Reor]:Sym
_R4[Reor]:{Rsym[1,2],Rsym[3,4]}

Find
evaluates the elements of the tensor *x*.

```
Find[$x,$y]::Ap[Ar,{Ap[Ar[$y,$i]},{DIM}],$x[0]}
```

GL

denotes the metric tensor with both indices covariant; (default Minkowskian).

```
GL : Ar[{4,4},0]; GL[4,4]:B[r]; GL[1,1]:-A[r]; GL[2,2]:-r^2
      GL[3,3]:-(r Sin[theta])^2
```

GU

denotes the metric tensor with both indices contravariant.

```
GU :: Minv[GL]
```

```
Pr["1"]
Proc[]
```

ChrL

denotes the Christoffel symbols with all indices covariant.

```
ChrL[$i_>Natp[$i],$j_>Natp[$j],$k_>Natp[$k]] :: ChrL[$i,$j,$k] := \
      1/2 (Dt[GL[$i,$j],CO[$k]] + Dt[GL[$i,$k],CO[$j]] \
      - Dt[GL[$j,$k],CO[$i]])
```

```
Pr["2"]
Proc[]
```

```
Find['ChrL,3]
```

```
Pr["3"]
Proc[]
```

R4

denotes the Reimann curvature tensor.

```
R4[$a_>Natp[$a],$b_>Natp[$b],$c_>Natp[$c],$d_>Natp[$d]]::\
R4[$a,$b,$c,$d]:(Lcl[Xz,Xmu,Xnu,Xeta];\
      Xz : GL[$a,Xmu] Dt[GU[Xmu,Xnu] ChrL[Xnu,$b,$c],CO[$d]] \
      - GL[$a,Xmu] Dt[GU[Xmu,Xnu] ChrL[Xnu,$b,$d],CO[$c]] \
      + ChrL[$a,$d,Xeta] ChrL[Xeta,$b,$c] \
      - ChrL[$a,$c,Xeta] ChrL[Xeta,$b,$d];\
      Xz:Comp[Xz])
```

```
Pr["4"]
Proc[]
```

```
Find['R4,4]
```

```
Pr["5"]
Proc[]
```

R2

denotes the Ricci tensor.

```
R2[$a_>Natp[$a],$b_>Natp[$b]]::R2[$a,$b]:(Lcl[Xz,Xmu,Xnu];\
      Xz : GU[Xmu,Xnu] R4[Xmu,$a,Xnu,$b]; Comp[Xz])
```

```
Find[R2[,]]
```

R

denotes the contraction of the Ricci tensor.

R::R:(Lcl[Xz,Xmu,Xnu]; Xz : GU[Xmu,Xnu] R2[Xmu,Xnu]; Comp[Xz])
R

Comp

performs component manipulations.

```

Comp[$x]::S[$x, {Xr1,Xr2,Xr3,Xr4,Xr5,Xr6,Xr7,Xr8},Inf]
Xr1: Gm[$a_Natp[$a], $b_Natp[$b]]-->GL[$a,$b]
Xr2: Gm['$a_Natp[$a], '$b_Natp[$b]]-->GU[$a,$b]
Xr3: Gm[$$i, '$smu]-->0
Xr4: Sf[$$i1,$smu,$$i2] Sg[$$j1,'$smu,$$j2] --> \
      (Lcl[X1,X2]; \
      If[~P[$smu[0]='Mark] & ~P['$f='GU],Dist[Sum[Sum[ \
        GU[X1,X2] Sf[$$i1,X1,$$i2] Sg[$$j1,X2,$$j2] \
        ,{X2,1,DIM}],{X1,1,DIM}],{Mult,Plus,Plus}]]])
Xr5: Sf[$$i1,'$smu_Natp[$smu],$$i2] --> \
      (Lcl[X1]; \
      If[~P['$f='GU],Dist[Sum[ \
        GU[$smu,X1] Sf[$$i1,X1,$$i2] \
        ,{X1,1,DIM}],{Mult,Plus,Plus}]]])
Xr6: Sf[$$i, '$smu_Natp[$smu] --> (Lcl[X1];If[~P['$f='GU],Sum[ \
      GU[$smu,X1] Sf[$$i, '$X1],{X1,1,DIM}])
Xr7: Sf[$i, '$smu_Natp[$smu] --> \
      (Lcl[X1,X2]; \
      If[~P['$f='GU],Dist[ \
        Dt[Sf[$i],CO[$smu] - Sum[Sum[ \
          GU[X1,X2] ChrL[X1,$i,$smu] Sf[X2] \
          ,{X2,1,DIM}],{X1,1,DIM}],{Mult,Plus,Plus}]]])
Xr8: Sf[$$i, '$smu_Natp[$smu] --> \
      (Lcl[X1,X2,X1,Xz]; \
      If[~P['$f='GU],Dist[ \
        Dt[Sf[$$i],CO[$smu] - (X1:{$i};Sum[Xz:X1; \
          Xz[0]:'$f;Sum[Xz[X1]:'X2; \
          As[Xz] ChrL[X2,X1,$smu] \
          ,{X2,1,DIM}],{X1,1,Len[X1]}],{Mult,Plus,Plus}]]])

```

XTerm

XTerm

Terminal as standard output

T.Shaw
Sep 1981

```
Open[,,{,19}]
```


XTern

Balanced ternary representation

S.Wolfram and P.Leyland

Jan 1982

Tern[n]generates a balanced ternary representation of the integer n .

```

Tern[$n_] := Intp[$n] := (Lci[Xtot, Xres, Xi]; \
  Xres:=$n + (3^Ceil[N[Log[Abs[$n], 3]+2]]-1)/2; \
  For[Xi:1, Xres~0, Inc[Xi], Xtot[Xi]:Mod[Xres, 3]-1; \
  Xres:Ceil[Xres/3]; Rev[Xtot])

```

[Knuth, vol 2 (2nd ed)]

#I[1]:: <XTern

#I[2]:: Tern[12345]

#O[2]: {1, -1, 0, -1, 0, -1, 1, 1, -1, 1.12777e~10}

Misfeatures: example shows numerical error

XToTop

XToTop

Tensor index rotation

S.Wolfram

```
Totop[St] :: {[Sn]:: Ar[Cyc[Ar[Sn,Dim[St]],-1],St[[S1,S1]]}
```

Rotate tensor index to top

XToTop.new

XToTop.new

Tensor index rotation

S.Wolfram

```
Totop[$t] := {[$n]:: Ar[Cyc[Ar[$n, Dim[$t]], -1], $t[$$1, $1]}
```

Rotate tensor index to top

XTri

XTri

Triangular list generation

group - sequences - cumulative lists

S.Wolfram

Jan 1982

Tri [*list*]

form a triangular *list* from sequences of successive entries of *list*.

```
Tri[$list,Listp[$list]] :: Ar[Len[$list],Ar[$1,$list]]
```

• UnFlat; XSubl

```
#1[1]:: <XTri
```

```
#1[2]:: Ar[5]
```

```
#0[2]: {1,2,3,4,5}
```

```
#1[3]:: Tri[X]
```

```
#0[3]: {{1},{1,2},{1,2,3},{1,2,3,4},{1,2,3,4,5}}
```


XTrig

Trigonometric functions

S.Wolfram
Feb 1982

Misfeatures: Sin[17Pi/12] does not simplify completely, perhaps through multiple get confusions

Periodicity relation

$$\text{Sin}[(\$n \rightarrow \text{Nump}[\$n] \& \$n > 2) \text{ Pi}] : \text{Sin}[\text{Mod}[\$n, 2] \text{ Pi}]$$

Special values

$$\text{Sin}[(\$n \rightarrow \text{Ratp}[\$n, 12] \& 1 < \$n < 2) \text{ Pi}] : -\text{Sin}[\$n \text{ Pi} - \text{Pi}]$$

$$\text{Sin}[(\$n \rightarrow \text{Ratp}[\$n, 12] \& 1/2 < \$n < 1) \text{ Pi}] : \text{Cos}[\$n \text{ Pi} - \text{Pi}/2]$$

$$\text{Sin}[\text{Pi}/12] : (\text{Sqrt}[3] - 1) / (2\text{Sqrt}[2])$$

$$\text{Sin}[\text{Pi}/6] : 1/2$$

$$\text{Sin}[\text{Pi}/4] : 1/\text{Sqrt}[2]$$

$$\text{Sin}[\text{Pi}/3] : \text{Sqrt}[3]/2$$

$$\text{Sin}[5\text{Pi}/12] : (\text{Sqrt}[3] + 1) / (2\text{Sqrt}[2])$$

[CRC p 227]

XTrigR

XTrigR

Further trigonometric functions

versine - haversine

S.Wolfram

Feb 1982

 $\text{Vers}[a] : 1 - \text{Cos}[a]$ $\text{Hav}[a] : \text{Vers}[a]/2$ $\text{Exsec}[a] : \text{Sec}[a] - 1$ $\text{Covers}[a] : 1 - \text{Sin}[a]$ $\text{Cis}[a] : \text{Cos}[a] + I \text{Sin}[a]$

[CRC p. 226]

XTup

XTup

n-tuples

S.Wolfram

Jul 1981

Tupo[*tot*, *n*]

yields a list of all possible ordered *n*-tuples of *tot* elements.

```
Tupo[$tot, Natp[$tot], $n, Natp[$n]] := Tupo[$tot, $n] : \
Cat[Tupo[$tot-1, $n], Map[Cat[$1, {$tot}], Tupo[$tot-1, $n-1]]]
Tupo[$tot, 0] := {{{}}
Tupo[$tot, $n, ($n > $tot)] := {}
```

Tup[*tot*, *n*]

yields a list of all possible unordered *n*-tuples of *tot* elements.

```
_Tup[Init] := <List 0
Tup[$tot, Natp[$tot], $n, Natp[$n]] := Tup[$tot, $n] : \
Cat[Tup[$tot-1, $n], Flat[Map[Ar[$n, Ina[$tot, $1, $2]], \
Tup[$tot-1, $n-1]], 1]]
Tup[$tot, 0] := {{{}}
Tup[$tot, $n, ($n > $tot)] := {}
```

Tupa[*tot*, *n*]

yields a list of unordered *n*-tuples of *tot* elements, allowing repetitions of an element.

```
Tupa[$tot, $n] := Flat[Ar[Ar[$n, $tot], List], $n-1]
```

Pairs[*tot*]

yields a list of all possible partitionings of *tot* elements into pairs.

```
_Pairs[Init] := <Sets
Pairs[$n, Natp[$n/2]] := \
Union[Map[Sort, Trans[{Tupo[$n, $n/2], \
Map[Capl[$1, Ar[$n]], Tupo[$n, $n/2]}, $n/2, 2]]
```

```
#I[1] := <Tup
#I[2] := Tupo[3, 2]
#O[2] := {{1, 2}, {1, 3}, {2, 3}}
#I[3] := Tup[3, 2]
#O[3] := {{2, 1}, {1, 2}, {3, 1}, {1, 3}, {3, 2}, {2, 3}}
#I[4] := Tupa[3, 2]
#O[4] := {{1, 1}, {1, 2}, {1, 3}, {2, 1}, {2, 2}, {2, 3}, {3, 1}, {3, 2}, {3, 3}}
#I[5] := Pairs[4]
#O[5] := {{{1, 2}, {3, 4}}, {{1, 3}, {2, 4}}, {{1, 4}, {2, 3}}}
```

XTuring

XTuring

Turing machine simulation

S.Wolfram

Jul 1981

Tape [i]

is the *i*th symbol on the data tape. (Tape [i]:1 defines a tape consisting solely of 1's.)

Spec [st1, symb1]

is of the form {symb2, st2} and specifies that when the machine is in state *st1* and reads the symbol *symb1* from the tape, it writes *symb2* in place of *symb1* on the tape, and makes a transition to state *st2*. The machine halts when no applicable Spec exists.

Left

is a special symbol which when read causes the tape to advance under the reading head one symbol to the left.

Right

is a special symbol causing the tape to advance to the right.

The symbol Null represents a blank position on the tape.

Start [pos, st0]

starts the Turing machine in state *st0* and at position *pos* on the tape; if the operation of the machine terminates, it yields the final position and state.

```
Start[Spos, Sst0] :: (Lc[Xpos, Xst]; Xpos:Spos; Xst:Sst0; \
  Rpt[Next[Spec[Xst, Tape[Xpos]]], Inf]; {Xpos, Xst})
Next[$x_~Valp[$x]] :: Ret[]
Next[{Left, Sst}] :: (Xst:Sst; Dec[Xpos])
Next[{Right, Sst}] :: (Xst:Sst; Inc[Xpos])
Next[{Symb, Sst}] :: (Tape[Xpos]:Symb; Xst:Sst)
```

```
#I[1]:: <XTuring
#I[2]:: Tape[$i_~$i<5]:1
#O[2]: 1
#I[3]:: Spec[1,0]:{Right,1}
#O[3]: {Right,1}
#I[4]:: Spec[1,1]:{0,2}
#O[4]: {0,2}
#I[5]:: Spec[2,0]:{Right,2}
#O[5]: {Right,2}
#I[6]:: Spec[2,1]:{Right,1}
#O[6]: {Right,1}
```


2

XTuring

2

#I[7]: Start[1,1]

#O[7]: {5,1}

#I[8]: Tape

#O[8]: {[3]: 0, [1]: 0, [5] → (5 > 5)}: 1}

XUnFlat

XUnFlat

List unflattening

S.Wolfram

Jul 1981

UnFlat[list, n]

collects successive sets of n entries in list into sublists.

```

UnFlat[list_Contp[list], $n_?Natp[$n]] := \
  Cat[Ar[{{1, Len[list], $n}}, Cat[Ar[$n, list[$1+$2-1], \
    $1+$2 <= Len[list]+1]]]]
UnFlat[list_Contp[list], $n_?Natp[Len[list]/$n]] := \
  Cat[Ar[{{1, Len[list], $n}}, Ar[$n, list[$1+$2-1]]]]

```

```

#I[1]:: <XUnFlat
#I[2]:: t:Ar[10]
#O[2]:: {1,2,3,4,5,6,7,8,9,10}
#I[3]:: UnFlat[t,2]
#O[3]:: {{1,2},{3,4},{5,6},{7,8},{9,10}}
#I[4]:: UnFlat[t,3]
#O[4]:: {{1,2,3},{4,5,6},{7,8,9},{10}}

```


XUnmark

XUnmark

Mark removal

S.Wolfram
Jul 1981

Unmark [*expr*]
removes all marks in *expr*.

```
Unmark[$expr] :: S[$expr, '$->$]
```

XVecan

XVecan

Three-dimensional vector analysis

transformation of coordinate systems - orthogonal coordinates
curvilinear coordinates - separation of variables - partial differential equations

S.Wolfram
Feb 1982

CO: list of orthonormal coordinates
SF: list of scale factors associated with each coordinate
(diagonal components of metric).

Vecp[v]
yields 1 if v is a contiguous list of three elements, representing a 3-dimensional vector, and 0 otherwise.

```
Vecp[$v] :: Contp[$v] & Len[$v]=3
```

VAbs[vec]
yields the norm (modulus) of the vector vec.

```
VAbs[$list_Contp[$list]] :: Sqrt[Ap[Plus,$list^2]]
```

Default Cartesian coordinate system:

```
CO : {x,y,z}
SF : {1,1,1}
```

Assignments for other coordinate systems:

Cartesian coordinates

```
CAR :: {CO:{x,y,z},SF:{1,1,1}}
```

Circular cylindrical coordinates

```
CYL :: {CO:{r,phi,z},SF:{1,r,1}}
```

Spherical coordinates

```
SPH :: {CO:{r,th,phi},SF:{1,r,r Sin[th]}}
```

Parabolic coordinates

```
PAR :: {CO:{xi,eta,phi},SF:{Sqrt[xi^2+eta^2],Sqrt[xi^2+eta^2],xi eta}}
```

Parabolic cylinder coordinates

```
PARCYL :: {CO:{xi,eta,z},SF:{Sqrt[xi^2+eta^2],Sqrt[xi^2+eta^2],1}}
```

Elliptic cylinder coordinates

ELLCYL :: {CO: {xi, eta, z}, SF[1]:SF[2]:c Sqrt[Sinh[xi]^2+Sin[eta]^2]; \ SF[3]:1; SF}

Prolate ellipsoidal coordinates

ELLPRO :: {CO: {xi, eta, phi}, SF: {c Sqrt[xi^2-eta^2]/Sqrt[xi^2-1], \ c Sqrt[xi^2-eta^2]/Sqrt[1-eta^2], c Sqrt[1-eta^2] Sqrt[xi^2-1]}}

Oblate ellipsoidal coordinates

ELLOBL :: {CO: {xi, eta, phi}, SF: {c Sqrt[xi^2+eta^2]/Sqrt[1+xi^2], \ c Sqrt[xi^2+eta^2]/Sqrt[1-eta^2], c Sqrt[1+xi^2] Sqrt[1-eta^2]}}

Toroidal coordinates

TOR :: {CO: {xi, eta, phi}, SF[1]:SF[2]:c/(Cosh[xi]-Cos[eta]); \ SF[3]:c Sinh[xi]/(Cosh[xi]-Cos[eta]); SF}

Elliptic coordinates

ELL :: {CO: {lam, mu, nu}, ellf[slam]:(a^2+slam)(b^2+slam)(c^2+slam); \ SF: {Sqrt[(lam-mu)(lam-nu)]/(2Sqrt[ellf[lam]]), \ Sqrt[(mu-lam)(mu-nu)]/(2Sqrt[ellf[mu]]), \ Sqrt[(nu-lam)(nu-mu)]/(2Sqrt[ellf[nu]])}}

Bipolar coordinates

BIP :: {CO: {xi, eta, z}, SF[1]:SF[2]:c/(Cosh[xi]-Cos[eta]); SF[3]:1; SF}

Arc

arc length ds^2 with respect to coordinates CO.

Arc :: Sum[SF[X1]^2 Dt[CO[X1]]^2, {X1, 1, 3}]

Vol

volume element dV in coordinates CO.

Vol :: Prod[SF[X1] Dt[CO[X1]], {X1, 1, 3}]

Grad[f]

gradient of scalar expression f with respect to coordinates CO.

Grad[f] :: Ar[3, D[f , CO[s]]/SF[s]]

Dvg[f]

divergence of vector (list) f with respect to CO.

Dvg[$f \rightarrow$ Vecp[f]] :: Sum[D[f [X1] (SF[1] SF[2] SF[3])/SF[X1]^2, \ CO[X1]], {X1, 1, 3}]/(SF[1] SF[2] SF[3])

CurI[f]

curl of vector expression f with respect to CO.

CurI[$f \rightarrow$ Vecp[f]] :: \ {D[SF[3] f [3], CO[2]] - D[SF[2] f [2], CO[3]]/(SF[2]SF[3]), \ D[SF[1] f [1], CO[3]] - D[SF[3] f [3], CO[1]]/(SF[3]SF[1]), \ D[SF[2] f [2], CO[1]] - D[SF[1] f [1], CO[2]]/(SF[1]SF[2])}

Lap[f]

Laplacian of scalar expression f with respect to CO.

Lap[f] := Sum[D[(SF[1] SF[2] SF[3])/SF[X1]^2 D[f ,CO[X1]], \ CO[X1]], {X1, 1, 3}]/(SF[1] SF[2] SF[3])

#I[1]:: <XVecan

#I[2]:: t: {x^2+a y^2, x y^3, x y z}

#O[2]:: {a y^2 + x^2, x y^3, x y z}

#I[3]:: Dvg[X]

#O[3]:: 2x + x y + 3x y^2

#I[4]:: Grad[X]

#O[4]:: {2 + y + 3 y^2, x + 6x y, 0}

#I[5]:: Vol

#O[5]:: Dt[x] Dt[y] Dt[z]

#I[6]:: SPH

#O[6]:: {{r, th, phi}, {1, r, r Sin[th]}}

#I[7]:: Vol

#O[7]:: r^2 Dt[phi] Dt[r] Dt[th] Sin[th]

#I[8]:: r^2 Sin[th]^3 Cos[phi]

#O[8]:: r^2 Cos[phi] Sin[th]^3

#I[9]:: Lap[X]

#O[9]::
$$\frac{-r^2 \cos[\phi] \sin[th]^2 + 3r^2 \cos[\phi] \sin[th]^4 + 9r^2 \cos[\phi] \cos[th]^2 \sin[th]^2}{r^2 \sin[th]}$$

#I[10]:: Ex[X]

#O[10]:: 3Cos[phi] Sin[th]^3 - Cos[phi] Sin[th] + 9Cos[phi] Cos[th]^2 Sin[th]

Scf[cart]

computes the scale factors SF for the coordinates CO from the list *cart* of values for the Cartesian coordinates x,y,z in terms of the curvilinear coordinates CO[1],CO[2],CO[3].

Scf[$cart$] := (SF:Ar[3, VAbs[D[$cart$,CO[i]]]])

[e.g. MOS chap. 12; Morse & Feshbach]

XWarn

XWarn

Warning messages

S.Wolfram

Jul 1981

```
$x/0_ Prh[$x/0,"generated"] :: $x/0
0^$x_ (If[$x<0,Prh[0^$x,"generated"]]) :: 0^$x
Log[0_ Prh["Log[0] generated"]] :: Log[0]
Gamma[$n_ If[Natp[-$n],Prh[Gamma[$n],"generated"]]] :: Gamma[$n]

<$f_ Prh[$f,"file unavailable"] :: <$f

($a :: ($b_ Prh[$a::$b,"impossible assignment"])) :: ($a :: $b)
($a : ($b_ Prh[$a:$b,"impossible assignment"])) :: ($a : $b)

Minv[$m_ If[Listp[$m],Prh[$m,"inversion impossible"]]] :: Minv[$m]
```

```
#I[1]:: <XWarn
```

```
#I[2]:: 2/0
```

```
2
- generated
0
```

```
#O[2]: 2
-
0
```

XWatch

XWatch

External file input tracing

S.Wolfram

Jul 1981

Watch[*file*]

inputs the specified external *file*, printing each assignment in the file before it is made.

```
Watch[$file] := (Set[_Setd[_Trace; Lcl[Xr]; Xr:<$file; Close[]; \
                _Set[Trace]:_Setd[Trace];; Open[]; Xr)
```


XWhi

XWhi

Whittaker function

S.Wolfram
Jul 1981

SWhi \rightarrow Ldist

SWhi[1]: $WhiM[s1, sm, sz] \rightarrow Exp[-sz/2] sz^{(1/2+sm)} WhiM[1/2+sm-s1, 1+2sm, sz]$

SWhi[2]: $WhiW[s1, sm, sz] \rightarrow Gamma[-2sm]/Gamma[1/2-sm-s1] WhiM[sz] \backslash$
 $+ Gamma[2sm]/Gamma[1/2+sm-s1] WhiM[s1, -sm, sz]$

SWhi[3]: $WhiM[sk, sm, sz] \rightarrow Exp[-sz/2] sz^{(sm+1/2)} Chg[1/2+sm-sk, 1+2sm, sz]$

SWhi[4]: $WhiW[s1, sm, sz] \rightarrow Exp[-sz/2] sz^{(1/2+sm)} KumU[1/2+sm-s1, 1+2sm, sz]$

for special cases, see XChg and XKumU.

XWron

XWron

Wronskian and Jacobian

S.Wolfram

Jul 1981

Wron[{y1,y2,...},x]

forms the Wronskian of {y1,y2,...} with respect to x, which must vanish if the yi are linearly dependent.

$$\text{Wron}[\text{\$list}, \text{\$x}] :: \text{Det}[\text{Ar}[\text{Len}[\text{\$list}], \text{D}[\text{\$list}, \{\text{\$x}, \text{\$i-1}\}]]]$$
Jac[{x1,x2,...}, {u1,u2,...}]

forms the Jacobian of the transformation from {x1,x2,...} to {u1,u2,...}.

$$\text{Jac}[\text{\$x}, \text{\$u}, \text{Len}[\text{\$u}] = \text{Len}[\text{\$x}]] :: \text{Det}[\text{Ar}[\text{Len}[\text{\$x}], \text{D}[\text{\$x}, \text{\$u}[\text{\$i}]]]]]$$

#I[1]:: <XWron

#I[2]:: Wron[{x^2, Exp[x]}, x]

#O[2]: -2x Exp[x] + x² Exp[x]

#I[3]:: Jac[{x^2+y^2, x(y-1)}, {x, y}]

#O[3]: -2y (-1 + y) + 2 x²

XYear

XYear

Day numbers

S.Wolfram

Jul 1981

Year [month, day]

gives the number of each day in a non-leap year

```

Month : {[Jan]:31, [Feb]:28, [Mar]:31, [Apr]:30, [May]:31, [Jun]:30, \
        [Jul]:31, [Aug]:31, [Sep]:30, [Oct]:31, [Nov]:30, [Dec]:31}
Year  : (Lc{[Xt];Xt:0;Ar[{{[Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec]}}, \
        Xt+Ar[Inc[Xt,Month[$X1]]])

```

#I[1]:: Year[Aug,29]

#O[1]: 241

XYoung

XYoung

Young graphs

S.Wolfram

Jul 1981

{i1,i2,...} represents a Young graph with i1 boxes in the top row, i2 in the second,

PDim[list]

yields the dimensionality of the representation of the symmetric group corresponding to the Young graph specified by *list*.

```
PDim[list] := Ap[Plus,list]!eProd[Prod[list[[i]-list[[j]]+j-1, \
{j,i+1,Len[list]}], {i,1,Len[list]}] / \
Prod[(list[[i]+Len[list]-1)!, {i,1,Len[list]}]
```

YGMult[yg1,yg2]

yields a list of Young graphs occurring in the product of the representations of the symmetric group corresponding to *yg1* and *yg2*.

```
YGMult[{i1,i2}] := (Lc[{X1,X2}; \
X11:{Ar[Len[i1],Ar[i1[1],0]}]; \
X12:Flat[Ar[Len[i2],Ar[i2[1],1]]]; \
'o[X1,Len[X2],X11:Flat[Map[YGadd[X1,X2[Xi],X11,1]; \
X11:Cat[Ar[Len[X1],X11[1]],YGt0[1]&YGt1[1,X2[Xi]]& \
YGt2[1,X2[Xi]]]; Map[Len,X11,{2}])
```

```
YGadd[list,$n] := \
Cat[Ar[Len[list],YGadd1[list,$1,$n]],{Cat[list,{$n}}]}
YGadd1[list,$pos,$n] := \
Cat[Ar[$pos-1,list],{Cat[list[$pos],{$n}]}, \
Ar[{ $pos+1,Len[list]}],list]
```

Test for legal graphs

```
YGt0[list] := Ap[Ge,Map[Len,list]]
YGt1[list,$n] := Ap[Uneq,Map[$1[2],Pos[$n,list]]]
YGt2[list,$n] := YGt2p[Del[0,Flat[Map[Rev,list]]],$n]
YGt2p[list,$n] := \
(Lc[Xt,Zr]; Zr:Ar[$n,0]; For[Xt:1,Xc:1,Xc<=Len[list]&Ap[Ge,Zr], \
Inc[Xc],X1[list[Xc]]:X1[list[Xc]]+1; If[Xc>Len[list],1,0])
YGt3[list,$sun] := $sun >= Len[list]
```

```
#I[1] := <XYoung
#I[2] := YGMult[{1,1},{1,1}]
#O[2] := {{2,2},{2,1,1},{2,1,1},{1,1,1,1}}
#I[3] := Map[PDim,X]
#O[3] := {3,2,3,3,1}
#I[4] := PDim[{5,3,2,2,1}]
#O[4] := 21450
```


XZeta2S

Generalized Riemann zeta function

reflection formula - Rademacher's formula

S.Wolfram

Feb 1982

```

SZeta[2,1] : Zeta[sz, sr_>Ratp[sr]] --> \
2 Gamma[1-sz] (2Pi Den[sr])^(-sz-1) (Sin[Pi sz/2] \
Sum[Cos[2 Pi sr xn] Zeta[1-sz, xn/Den[sr]], {xn, 1, Den[sr]}] + \
Cos[Pi sz/2] Sum[Sin[2 Pi sr xn] Zeta[1-sz, xn/Den[sr]], {xn, 1, Den[sr]}])

```

[MOS sect. 1.4]

XZeta2V

XZeta2V

Generalized Riemann zeta function

S.Wolfram
Feb 1982

Definition of special values

- Zeta[s_n,1] : Zeta[s_n]
- Zeta[s_n,1/2] : Zeta[s_n]/(2[~]s_n-1)
- Zeta[0,s_a] : 1/2-s_a
- Zeta[s_n,Natp[s_n+1],s_a] : -Ber[-s_n+1,s_a]/(-s_n+1)

Derivatives

- D[Zeta[s_a,s_a],{s_n,1,0}] : Log[Gamma[s_a]] - Log[2Pi]/2
- D[Zeta[s_a,s_a],{s_a,1,s_b}] : -s_a Zeta[s_a+1,s_b]

[MOS sect. 1.4]

XZetaR

Functions related to Riemann zeta function

sums of reciprocal powers

S.Wolfram

Feb 1982

• Catb; Li

$\text{Eta}[n] : (1 - 2^{-(1-n)}) \text{Zeta}[n]$

$\text{Eta}[2] : \pi^2/12$

$\text{Eta}[4] : 7\pi^4/720$

[AS sect. 23.2]

$\text{Lambda}[n] : (1 - 2^{-n}) \text{Zeta}[n]$

$\text{Lambda}[2] : \pi^2/8$

$\text{Lambda}[4] : \pi^4/96$

[GR sect. 9.56]

$\text{Xi}[s] : s(s-1) \Gamma[s/2] / (2 \pi^{(s/2)}) \text{Zeta}[s]$

INDEX

- absolute value XAbs
- aCb XSetsSX
- Ack XAck
- adjacency matrix XGr
- Adj XMat4
- aIbIc... XSetsSX
- Allbut XAllbut
- All XLogic2
- antisymmetric matrix XMat3
- any XAny
- Any XAny
- APart XLPart
- APL encode/decode XCode
- APL XDap
- append XList0
- Apper XPerm0
- approximate integration XGQInt
- App XList0
- arbitrary accuracy XM
- arbitrary precision XM
- arcs XGr
- Arcs XGr
- Arc XVecan
- Arperm XArperm
- ARRand XRandC
- Arrow XAck
- ASCII XChar
- astronomical data XOrbit
- Asymp XMat3
- aUbUc... XSetsSX
- a\b XSetsSX
- a~b~c... XSetsSX
- basis tensor XLevi
- beginning XList0
- Bell XBell
- binary XBase XBit
- binned data XClass
- binning XHist
- Bitand XBit
- Bitor XBit
- Bits XBit
- bitwise and XBit
- bitwise or XBit
- Boolean algebra XLogicPr
- box XBox
- boxes XPrtable
- Box XBox
- box XPrtable
- BRand XRandC
- Bsub XLUP
- CanGam XGammaS
- canonical form XPow
- CRapprox XClass
- catentate XList0
- Catnum XCatalan
- Cdata XData1
- ceiling XRnd
- celestial mechanics XOrbit
- CfD XCF
- CFrac1 XClass
- CFract XClass
- CGS units XNKS
- character strings XChar
- characteristic polynomial XMat4
- Cherpol XMat4
- Char XStat XStr0
- Chisholm identities XG
- choose XMask
- ChrL XTensor
- Church's thesis XHalt
- CIns XStr1
- circle XBox
- Circle XBox
- CJoin XStr0
- Classify XClass
- Class XClass
- CLen XStr0
- Clifford algebra XFierz
- CMax XClass
- CMD XClass
- CMean XClass
- CMed XClass
- CMin XClass
- CMode XClass
- Cmom XStat
- Cmpl XSets
- Cof XMat4
- collate XBit
- colmat XPrtable
- column operations XDap
- columnar XPrtable
- columnated input XData1 XData
- col XPrtable
- combinatorial functions XBell
- complement XAllbut
- completeness relations XFierz
- Comp XTensor
- Conf XConfus
- congruential random number generation XRRr
- conserve memory XKIIIO
- ConsSol XConsSol
- Contig XContig
- continued fraction representation XCF
- convex polygon dissection XCatalan
- Con XCon
- correlation coefficient XFit
- Corr XFit
- CO XTensor
- CPos XStr1
- CQ1 XClass
- CQ3 XClass
- create function XFun
- create pattern XFun
- CRep XStr1

SMP LIBRARY / INDEX

- CRev XStr8
- crystal energies XLatsum
- CSD XClass
- CS XStr1
- cumulative lists XTr1
- Cum XStat
- Cur1 XVecan
- curve fitting XGFit XLCh12
- curvilinear coordinates XVecan
- CVar XClass
- Dap XDap
- data analysis XFit XGFit
- data conversion XData1
- data presentation XHist
- data XData
- Data XData
- DCf XCf
- Decode XCode
- Default XVecan
- Degree XGr
- depth XLev
- derangements XSfct1
- destroy input XKillIO
- destroy output XKillIO
- determinant XLUP
- determinants XPer
- diagonal matrices XMat1
- diagonal matrix XMat3
- diagonals XMat2
- Diagp XMat3
- Diag XMat1
- difference equations XDiff
- differential equations XLap
- differentials XIndep
- Diff XDiff
- digit extraction XDig
- Dig XDig
- dimensional analysis XMKS
- DIM XFierz XTensor
- Dirac bilinear covariants XFierz
- Dirac gamma matrices XFierz
- discretization XHist
- Disc XDisc
- Disjp XSets
- distribution over lists XLProp
- distribution XExDot
- domains XLev
- double recursion XHof
- DRand XRandD
- DSol XDSol
- dummy symbol test XGenp
- Dvg XVecan
- dynamic programming XMSet
- EccRnom XOrbit
- eigenvalue equation XMat4
- electrodynamics XG
- Embed XLUP
- Encode XCode
- enter matrix XMat1
- entier XRnd
- epsilon tensor XLevi
- Eqn XEqn
- ERand XRandC
- errors XChi2
- estimated size XLenex
- Euler circuits XGr
- Euler Gamma function XGammaS XGammaV
- Euler integral of first kind XGammaS XGammaV
- Eulerp XGr
- Eulgam XEulgam
- Excess XStat
- exclusion XAIBut
- ExDot XExDot
- existence test XAny
- ExMDot XExDot
- expansion XLenex
- Expc XStat
- explicit tensors XCon
- exponential fit XFit
- expr XTEST
- extract XMask
- extrapolation XItp
- fermion factors XFierz
- For XPrime
- Feynman diagrams XG
- Fib XFib
- Fierz XFierz
- figurate numbers XPolynum
- fill holes XContig
- fill XContig
- find XScan
- Find XTensor
- finite elements XDiff
- Fiper XPerm8
- first XScan
- first-order logic XLogicPr
- First XList8
- FitExp XFit
- FitPow XFit
- Fit XFit
- fixed accuracy XM
- fixed precision XN
- floor XRnd
- FN XN
- forget past XKillIO
- form sublists XSub1
- formatting XPrtable
- Forward differences XDiff
- forward differences XLDiff
- four vectors XLer
- FPow XFPow
- Fract XStat
- frequency XInfo
- Freq XInfo
- Frobenius method XDSol
- From18 XBase
- function evaluation XLitp
- function fitting XGFit XLCh12

SMP LIBRARY / INDEX

functional form XFit
functional independence XIndep
Functional powers XFPow
Fun XFun
Fz XFierz
gamma matrix algebra XG
Gaussian distribution XChi2
general recursion XRec
generalized Fibonacci sequence XHof
generalized power XRec
generalized product XIter
generalized sum XIter
generalized trace XMat4
generalized traces XCon
generic predicate XGenp
Genocchi XGenocchi
Genp XGenp
Gentr XMat4
geometrical figures XBox
GFit XGFit
Ginds XG
GL XTensor
GMean XStat
Gm XTensor
Golden ratio XFib
goodness of fit XLChi2
GQInt XGQInt
Grad XVecan
graph equivalence XGr
graph isomorphism XGr
graph representation XGr
graph traversibility XGr
group XSubl XTrl
GU XTensor
Haltp XHalt
Hamilton circuits XGr
Hamp XGr
Harm XHarm
haversine XTrigr
head XList0
hermitean adjoint XMat4
hexadecimal XBase
histograms XClass
Hist XHist
hline XPrtable
HMean XStat
Hof XHof
Horn XHorn
hthing XPrtable
hv XPrtable
incidence matrix XGr
Indep XIndep
indicial tensor calculus XTex
inner products XCon
insert XList0
insoluble problem XHalt
Ins XList0
Intbit XBit
integer conversion XBase

integer equations XDies
interactive matrix input XMat1
interpolation XItp
Inter XSets
int XTEST
Inum XPrime
IRand XRandD
Isop XGr
iterated functions XFPow
iteration XIter
Iter XIter
Itp XItp
Jac XWron.
Jnum XPrime
KD XTensor
Kepler's laws XOrbit
kill input XKIIIO
kill labels XKIIIO
kill lines XKIIIO
kill output XKIIIO
KIIIO XKIIIO
Kurt XStat
Lagrangian interpolation XLItp
lambda expression XFun
language analysis XInfo
Lap XLap XVecan
Latsum XLatsum
LCM XLCM
LCoeff XPoly
LdEq XLdEq
Ldet XLUP
Ldiag XMat2
LDiff XLDiff
LDig XDig
Ldot XLor
LDRand XRandL
leading zeroes XPad
least squares fit XFit
left justify XPad
Left XTuring
Lenex XLenex
Leniev XLev
letters XChar
Levi XLevi
Lev XLev
LExpt XPoly
LInd XInd
linear algebra XLUP
linear equation XLUP
linear fit XFit
Lin XList1
LISP CAR XList0
list element removal XRIIbut
list substitution XList1
lists XList1
list XTEST
LItp2 XLItp
LItp3 XLItp
LItp XLItp

SMP LIBRARY / INDEX

Ldiv XLUP
 Linv XLUP
 look-up XMSet
 Lowerp XChar
 LPad XPad
 LPart XLPart
 LPos XList1
 LProd XLArith
 LProp XLProp
 LRand XRandL
 Lrpt XList0
 LSub XList1
 LSum XLArith
 LS XList1
 Lup XLUP
 Madelung sums XLatsum
 make cubical XContig
 make generic XFun
 make rectangular XContig
 make square XContig
 MAnom XOrbit
 mapping XDap
 Mask XMask
 Matp XMat3
 matrices XPrtable
 matrix adjoint XMat4
 matrix classes XMat3
 matrix equations XLdEq
 matrix inverse XLUP
 matrix power XMat4
 matrix types XMat3
 matrix XLUP
 Maxind XInd XMaxind
 MD XStat
 Mean XStat
 Med XStat
 memory requirement XLenx
 Mer XPrime
 Minkowski space XLor
 minors XMat2
 Minor XMat2
 model comparison XLCh12
 model fitting XLCh12
 modular arithmetic XQuadres
 modulus XAbs
 Mom XStat
 mpoly XTEST
 Mpow XMat4
 MRd XMat1
 mu function XScan
 multiple assignment XLProp
 Multset XSets
 nested functions XFPow
 network theory XGr
 Newton's method XConsSol
 NF XN
 NMake XDig
 NMap XNMap
 nnint XTEST
 nodes XGr
 Nodes XGr
 normal distribution XCh12
 Norm XNorm
 NRand XRandC
 NSol XNSol
 number construction XDig
 number of equivalence relations XBell
 number theory functions XPolynum
 Number theory XQuadres
 numb XTEST
 numerical evaluation of polynomials XHorn
 numerical integration XGQInt
 numerical inversion XConsSol
 numerical quadrature XGQInt
 numerical value XAbs
 octal XBase
 operational methods XLap
 oracle XHalt
 ORand XRandL
 orbital elements XOrbit
 ordinary differential equations XSerSol
 orthogonal coordinates XVecan
 pad XContig
 Pairs XTop
 pairwise differences XLDiff
 pairwise subtraction XLDiff
 PAp XPlot
 Pap XProj
 parameter determination XFit
 parameter fitting XGFit
 part extraction XLev
 part XList0
 partial derivatives XIndep
 partial differential equations XVecan
 Pause XPause
 PCat XPlot XProj
 Pcomp XPerm1
 PCum XRandD
 PDelta XPR
 PDim XYoung
 PDios XDios
 Peel XPeel
 pentagonal numbers XPolynum
 PermP XPerm0
 Per XPer
 PGamma XPR
 physical constants XNKS
 physical quantities XDim XNKS
 PInt XPR
 pint XTEST
 Pinv XPerm1
 PLam XPR
 Piam XPR
 Pihist XPIhist
 plotting XBox
 Pmat XLUP
 PNorm XRandD
 pnumb XTEST

SMP LIBRARY / INDEX

- Polar XPolar
- polynomial rearrangement XHorn
- polynomial roots XDisc
- Polynum XPolynum
- poly XTEST
- positional notation XBase XDig
- power fit XFit
- power series XDSol XSerSol
- power set XLPart
- Powset XSets
- PPI XPR
- Ppow XPerml
- prepend XList0
- Prep XList0
- pretty-printing XEqn
- prime decomposition XPow
- Primep XPrimep
- printing XPrtable
- Prmat XPrtable
- probability functions XChi2
- Project XMat3
- projection matrix XMat3
- property XLPprop
- PrTF XLogicPr
- PSig XPR
- Pamp XPamp
- PSqrt XPR
- PsSol XSerSol
- PXI XPR
- Q1 XStat
- Q3 XStat
- QChi2 XChi2
- QD XStat
- Qskew XStat
- Qtr XLUP
- Quadres XQuadres
- quantization XHist
- quantum field theory XG
- Quant XLogic2
- R2 XTensor
- R4 XTensor
- Rademacher's formula XZeta25
- radicals XPow
- radix arithmetic XBase
- Ramp XRamp
- Range XStat
- Ranp XRpoly
- Ranup XRpoly
- RAR XRAR
- reclaim memory XKIIIO
- recurrence relations XRAR
- recursion testing XHof
- recursive function theory XAck
- reduction XExDot
- reflection formula XZeta25
- regions XBox
- Regression XGFit
- Regs XGr
- Relab XGr
- relativistic mechanics XLor
- remove XList0 XMask
- rencontres numbers XSfct1
- reorderings XARpera
- Report XReport
- repeat XList0
- repeated transformation XMat4
- replicate XList0
- replication XIter
- report generation XPrtable
- right justify XPad
- Right XTuring
- rings XQuadres
- RMS XStat
- Rm XList0
- Rnd XRnd
- rooted planar trees XCatalan
- Rot2 XRot2
- Rot3 XRot3
- rotate XDep
- RotM2 XRot2
- RotM3 XRot3
- RPad XPad
- rpt XTEST
- ruled XPrtable
- R XTensor
- save memory XKIIIO
- scalar products XExDot
- scalars XExDot
- scales of notation XBase
- Scalp XExDot
- scan XAny XScan
- Scan XScan
- Scf XVecan
- SD XStat
- search XScan
- select XList0 XMask
- separation of variables XVecan
- Sequence generation XRAR
- sequence positions XList1
- sequences XSub1 XTri
- sets of equations XLdEq
- Sfct1 XSfct1
- Shannon entropy XInfo
- Shan XInfo
- Showtime XShowtime
- Sigma XPauli
- similarity XDim
- simultaneous equations XLdEq
- Skew XStat
- smoothing XFit XItp
- Solar system XOrbit
- Some XLogic2
- Spare XSpare
- Spec XTuring
- Squatp XMat3
- square matrix XMat3
- square root XPow
- square XBox

SMP LIBRARY / INDEX

Start XTuring
 state table XLogicPr
 statistics XCh12 XGFit
 Status XStatus
 Stirling numbers XBell
 structural operation XLev
 sublist positions XList1
 Sub1 XSub1
 Subp XSets
 Sub XSets
 sums of reciprocal powers XZetaR
 symbolic-numeric interface XGQInt
 symmetric matrix XMat3
 symmetries XArpera
 Sympol XSympol
 Symp XMat3
 tabular data XPrtable
 tabulation XPrtable
 tail XList8
 take XList8
 Tape XTuring
 TDiag XMat2
 temp XTEST
 Tenp XTensor
 tensor canonicalization XTEx
 tensor symmetries XTEx
 ternary XBase
 Tern XTern
 test for any elements XAny
 text manipulation XChar XDig
 text processing XEqnPr
 text-formatting XEqn
 TEx XTEx
 Tinv XLUP
 Tk XList8
 To18 XBase
 ToArc XGr
 ToC XPermC
 ToInd XInd
 Tolower XChar
 ToL XInd
 ToMode XGr
 topbot XPrtable
 ToP XPermC
 total derivatives XIndep
 total length XLenex
 totally antisymmetric tensor XLevi
 trace identities XG
 trailing zeroes XPad
 transformation of coordinate systems XVecan
 triangular numbers XPolynum
 Tri XTri
 TROFF XEqnPr
 truncation XRnd
 Trunc XLUP
 Tupa XTup
 Tupo XTup
 Tup XTup
 two-dimensional output XEqnPr
 type casting XN XRnd
 type coercion XN
 type conversion XN
 typesetting XEqnPr
 Unclassify XClass
 unflatten XSub1
 UnFlat XUnFlat
 Union XSets
 units conversion XMKs
 units XDim
 UNIX XEqnPr
 Unmark XUnmark
 Upperp XChar
 VArbs XVecan
 Var XStat
 vbar XPrtable
 vblank XPrtable
 VCon XG
 Vec2p XRot2
 Vec3p XRot3
 Vecp XVecan
 vectors XExDot
 versine XTrigR
 vline XPrtable
 Vol XVecan
 vthing XPrtable
 Watch XWatch
 word processing XEqnPr
 words XChar
 Wron XWron
 XLCh12 XLCh12
 x_Scal XExDot
 Ycut XPlot
 Year XYear
 YGMult XYoung
 zeta functions XLatsum
 XMSet
 ~a XSetsSX

INDEX

- $\#1$ 1.3
 $\#0$ 1.3
 $\#T$ 1.3
 $\#X$ 6.3
 $\#I$ 6.3
 $\#O$ 6.3
 $\#T$ 6.3
 $\#$ 1.3
 3-j symbol $\#ig$ 8.6
 6-j symbol, Racah $\#ac$ 8.6
 $\beta(n)$ $\#atb$ 8.7
 $\Gamma(x)$ $\#amma$ 8.7
 $\Gamma(x, \alpha)$ $\#amma$ 8.7
 γ Euler 8.4
 δ function $\#elta$ 8.3
 $\zeta(z)$ Zeta 8.7
 $\zeta(z, \alpha)$ Zeta 8.7
 ϑ function $\#eta$ 8.3
 $\vartheta_i(u)$ $\#acth$ 8.10
 $\mu_k(n)$ $\#ob$ 8.11
 $\nu(n)$ $\#io$ 8.11
 $\Pi(k|t)$ $\#IIPi$ 8.10
 π $\#i$ 8.4
 $\rho_n(v, z)$ $\#cp$ 8.8
 $\sigma(u)$ $\#eiz$ 8.10
 $\Phi(z, s, \alpha)$ $\#er$ 8.7
 ϕ $\#hi$ 8.4
 $\phi(n)$ $\#otent$ 8.11
 $\psi(z)$ $\#si$ 8.7
 $\psi^{(n)}(z)$ $\#si$ 8.7
 $\sigma_k(n)$ $\#ivsig$ 8.11
 $Ai(z)$ $\#irAi$ 8.8
 $B(x, y)$ $\#eta$ 8.7
 $B(x, y, \alpha)$ $\#eta$ 8.7
 $B_n(x)$ $\#er$ 8.7
 B_n $\#er$ 8.7
 $Bi(z)$ $\#irBi$ 8.8
 $ber_n(z) + i bei_n(z)$ $\#elbe$ 8.8
 $C_n^{(1)}(x)$ $\#ag$ 8.9
 $C(z)$ $\#reC$ 8.8
 $Chi(z)$ $\#oshi$ 8.7
 $Ci(z)$ $\#osi$ 8.7
 $D_p(z)$ $\#ar$ 8.8
 $E_n(z)$ $\#ebE$ 8.8
 $E_n(z)$ $\#expi$ 8.7
 $E_n(x)$ $\#ei$ 8.7
 E_n $\#ei$ 8.7
 $E(k|t)$ $\#IIE$ 8.10
 $Ei(z)$ $\#ei$ 8.7
 $erf(z)$ $\#erf$ 8.8
 $erfc(z)$ $\#erfc$ 8.8
 $F_L(\eta, r)$ $\#ouF$ 8.8
 ${}_1F_1(a; c; z)$ $\#hg$ 8.8
 ${}_2F_1(a, b, c; z)$ $\#hg$ 8.9
 $G_L(\eta, r)$ $\#ouG$ 8.8
 $H_n(z)$ $\#trH$ 8.8
 $H_n(z)$ $\#er$ 8.8
 $H_n^{(1)}(z)$ $\#esH1$ 8.8
 $H_n^{(2)}(z)$ $\#esH2$ 8.8
 $I_n(z)$ $\#esI$ 8.8
 $J_n(z)$ $\#ngJ$ 8.8
 $J_k(n)$ $\#or$ 8.11
 $J_n(z)$ $\#esJ$ 8.8
 $j_n(z)$ $\#esj$ 8.8
 $K_n(z)$ $\#esK$ 8.8
 $k_\nu(z)$ $\#atk$ 8.8
 $K(k|t)$ $\#ellk$
 $ker_n(z) + i kei_n(z)$ $\#elke$ 8.8
 $L_n(z)$ $\#strL$ 8.8
 $L_n^{(a)}(z)$ $\#lag$ 8.8
 $Li_n(z)$ $\#li$ 8.7
 $li(z)$ $\#logi$ 8.7
 $M_{i,m}(z)$ $\#whiM$ 8.8
 $P_n^{(a,b)}(z)$ $\#jacP$ 8.9
 $P(u)$ $\#weiP$ 8.10
 $Q_n^{(m)}(z)$ $\#legP$ 8.9
 $S_n^{(m)}$ $\#sti2$ 8.6
 $S_n^{(m)}$ $\#sti1$ 8.6
 $S(z)$ $\#res$ 8.8
 $s_{m,n}(z)$ $\#lom$ 8.8
 $Shi(z)$ $\#inhi$ 8.7
 $Si(z)$ $\#ini$ 8.7
 $Sn(x|m)$ etc. $\#jacRn$ 8.10
 $T_n(x)$ $\#cheT$ 8.9
 $T(m, n, z)$ $\#tor$ 8.8
 $U_n(x)$ $\#cheU$ 8.9
 $UU(a, b, z)$ $\#kumU$ 8.8
 $W_{i,m}(z)$ $\#whiW$ 8.8
 $(x)_n$ $\#oc$ 8.7
 $Y_n(z)$ $\#esY$ 8.8
 $y_n(z)$ $\#esy$ 8.8
 \sim_s $\#setsSX$
 abort 1.5
 absolute value $\#abs$ 8.3
 absolute value $\#Xabs$
 $\#abs$ 8.3
 $\#cb$ $\#setsSX$
 access checking A.7
 accuracy 2.1
 $\#ack$ $\#Xack$
 $\#acosh$ 8.5
 $\#acos$ 8.5
 $\#acoth$ 8.5
 $\#acot$ 8.5
 $\#acsch$ 8.5
 $\#acsc$ 8.5
 adding parts 3.2
 addition $\#plus$ 8.2
 adjacency matrix $\#XGr$
 $\#adj$ $\#XMat4$
 $\#rex$ 7.10
 $\#agd$ 8.5
 $\#albc...$ $\#setsSX$
 aid 1.2

SMP HANDBOOK / INDEX

AirAi 8.8
 AirBi 8.8
 Airy function AirAi 8.8 AirBi 8.8
 Allbut XAllbut
 All XLogic2
 ambiguity, input 1.1
 ambiguous output 2.12
 analogue device 10.4
 analyse expression Aex 7.10
 And 5.
 Anger function AngJ 8.8
 AngJ 8.8
 angle brackets 0.
 antisymmetric matrix XMat3
 antisymmetric ordering A sym 7.7
 antisymmetric tensor Sig 9.6
 any XAny
 Any XAny
 APart XLPart
 APL encode/decode XCode
 APL XDap
 append Cat 7.7
 append XList0
 Apper XPerm0
 application of expressions 2.7
 apply rules 3.1
 apply Ap 7.2
 apply, multiple Map 7.2
 approximate integration XGQInt
 approximation Ax 9.5
 approximation, series 9.5
 App XList0
 Ap 7.2
 arbitrary accuracy XM
 arbitrary expressions 2.6
 arbitrary length integer B 2.1
 arbitrary magnitude number A 2.1
 arbitrary precision number F 2.1
 arbitrary precision XM
 arcs XGr
 Arcs XGr
 Arc XVecan
 Arep 3.3
 arguments, variable number of Tier 7.7
 arithmetic functions 8.2
 Arpera XArpera
 ARRand XRandC
 arrange Sort 7.7
 array generation Ar 7.1
 arrays 2.4
 arrays, definition of 3.2
 Arrow XAck
 Ar 7.1
 ASCII codes A.5
 ASCII XChar
 Asech 8.5
 Asec 8.5
 Asinh 8.5
 Asin 8.5

assemble As 7.3
 assertion testing Is 5.
 assertions 3.2
 assertions, relational 5.
 assignment 3.2
 assignment, property Prset 4.
 assignment, type Tyset 4.
 assistance 1.2
 associative functions 2.6
 associative Flat 7.7
 associativity 2.10 2.11
 assume P 5.
 assumption, character 7.6
 assumptions 3.2
 assumptions, relational 5.
 astronomical data XOrbit
 Asymp XMat3
 A sym 7.7
 asynchronous operations 10.9
 As 7.3
 Atanh 8.5
 Atan 8.5
 attributes 4.
 At 7.2
 aUbUc... XSetsSX
 automatic variables LcI 6.3
 Axes 10.2
 Ax 9.5
 a\b XSetsSX
 A 2.1
 a^b^c... XSetsSX
 bases 9.1
 basis tensor XLevi
 Bateman function Batk 8.8
 Batk 8.8
 beginning XList0
 Bell XBell
 Bernoulli numbers Ber 8.7
 Bernoulli polynomials Ber 8.7
 Ber 8.7
 BesH1 8.8
 BesH2 8.8
 BesI 8.8
 BesJ 8.8
 Besj 8.8
 BesK 8.8
 Bessel function, irregular spherical Besy 8.8
 Bessel function, irregular BesY 8.8
 Bessel function, modified BesI 8.8 BesK 8.8
 Bessel function, regular spherical Besj 8.8
 Bessel function, regular BesJ 8.8
 BesY 8.8
 Besy 8.8
 beta function Beta 8.7
 Beta 8.7
 biconditional, logical Eq 5.
 big floating point number F 2.1
 big integer B 2.1
 big number A 2.1

- binary file 10.7
- binary operator 2.11
- binary XBase XBit
- binned data XClass
- binning XHist
- binomial coefficient Comb 8.6
- Bitand XBit
- Bitor XBit
- bitpad 10.4
- Bits XBit
- bitwise and XBit
- bitwise or XBit
- blank Null 2.2
- block A.6
- blocks 6.3 Size 10.8
- Boolean algebra XLogicPr
- boolean operations 5.
- box XBox
- boxes XPrtable
- Box XBox
- box XPrtable
- braces 0. 2.4
- bracket levels 1.7
- BRand XRandC
- break 1.5
- browse Dsp 10.6
- Bsub XLUP
- bug report Send 10.6
- bytes A.6
- B 2.1
- C language 10.7
- call, function 2.3
- call, procedure 2.3
- CanGam XGammaS
- canonical form Reor 7.7
- canonical form XPow
- canonical ordering Ord 5. Reor 7.7
- CApprox XClass
- Cartesian "product", generalized Outer 9.6
- Cartesian product Omult 8.2
- Catalan beta function Catb 8.7
- Catalan's constant Catalan 8.4
- Catalan 8.4
- Catb 8.7
- catenate Cat 7.7
- catentate XList0
- Catnum XCatalan
- Cat 7.7
- Cb 7.9
- Cdata XData1
- ceiling XRnd
- Ceil 8.3
- celestial mechanics XOrbit
- CfD Xcf
- CFrac1 XClass
- CFract XClass
- Cf 9.5
- CGS units XMKS
- chameleonic expression 2.8
- chameleonic symbols 2.2
- Cham 4.
- change directory Dir 10.6
- channels, input/output A.3
- character codes A.5
- character determination 7.6
- character manipulation 10.5
- character replacement 2.11
- character strings XChar
- characteristic polynomial XMat4
- characteristics 4.
- Charpoi XMat4
- Char XStat XStr0
- Chebychef function of first kind CheT 8.9
- Chebychef function of second kind CheU 8.9
- CheT 8.9
- CheU 8.9
- Chg 8.8
- Chisholm identities XC
- choose statement Sel 6.1
- choose XMask
- ChrL XTensor
- Church's thesis XHalt
- CIns XStr1
- circle XBox
- Circle XBox
- CJoin XStr0
- Classify XClass
- Class XClass
- Clebsh-Gordan coefficient Wig 8.6
- CLen XStr0
- clicks A.6
- Clifford algebra XFierz
- Clock 10.9
- Close 10.3
- CMax XClass
- CMD XClass
- CMean XClass
- CMed XClass
- CMin XClass
- CMode XClass
- Cmom XStat
- Cmpl XSets
- code 10.7
- code file 10.7
- code files A.4
- Code 10.7
- coefficient Coef 7.9
- coefficient, numerical Nc 7.9
- Coef 7.9
- Cof XMat4
- collate XBit
- collect terms Cb 7.9 Col 7.9
- collect Fac 9.1
- collections 2.4
- collective function Ldist 7.7
- colmat XPrtable
- column operations XDap
- columnar XPrtable

SMP HANDBOOK / INDEX

- columnated input **XData1 XData**
Col 7.9
col XPrtable
 combinatorial coefficient **Comb** 8.6
 combinatorial functions 8.6
 combinatorial functions **XBell**
 combine denominator **Rat** 7.9
 combine expressions **Share** 10.8
 combine lists **Cat** 7.7
 combine memory **Share** 10.8
 combine **Cb** 7.9
Comb 8.6
 commands, monitor A.3
 commentary 1.2
 comments 2.9
 comments, external file A.2
 common denominator **Rat** 7.9
 common elements **Inter** 7.7
 common subexpressions **Share** 10.8
 communication **Send** 10.6
 commutative functions 2.6
 commutative **Comm** 4.
Comm 4.
 compilation 10.7
 compiled **Cons** 10.7
 complement **XArribut**
 Complementary error function **Erfc** 8.8
 completeness relations **XFierz**
 complex conjugate **Conj** 8.3
 complex number **Cx** 2.1
 compliment **Send** 10.6
 compulsory filters 0.
 computed goto statement **Set** 6.1
Comp XTensor
 concatenate **Cat** 7.7
 concurrent evaluation **Ser** 4.
 conditional, logical **Imp** 5.
 conditionals 6.1
 conditions on generic symbols **Gen** 4.
 confluent hypergeometric function **Chg** 8.8
Conf XConfus
 congruential random number generation **XRRar**
 conjugate **Conj** 8.3
 conjunction, logical **And** 5.
Conj 8.3
 conserve memory **XKIIID**
ConsSol XConsSol
 constant **Const** 4.
 constants, mathematical 8.4
 construction of programs 10.7
Const 4.
Cons 10.7
Cons 4.
 contains **In** 7.5
 content determination 7.5
 contents of expression **Rex** 7.10
 contents, list of **Cont** 7.5
 contiguous list, test for **Contp** 7.6
 contiguous lists 2.4
 contiguous, make list **Cat** 7.7
Contig XContig
 continuation lines 1.1
 continue **Ret** 6.3
 continued fraction approximation **Cf** 9.5
 continued fraction representation **XCf**
 contour plot **Graph** 10.2
Contp 7.6
 contraction **Inner** 9.6
 control of operations 2.5
 control of simplification 3.1
 control structures 6.
 control transfer **Jmp** 6.3
 controlled evaluation 3.3
Cont 7.5
 conventions 0.
 conventions, external files A.2
 conversion to polynomial **Rx** 9.5
 conversion, number A.3
 conversion, program 10.7
 convert list to projection **Ra** 7.3
 convert projection to list **Dis** 7.3
 convex polygon dissection **XCatalan**
Con XCon
 copy file **Save** 10.6
 copy **Open** 10.3
 core management 10.8
 coroutines 6.3
 correction 1.7
 correlation coefficient **XFit**
Corr XFit
Coshi 8.7
Cosh 8.5
 cosine integral function **Cosi** 8.7
CosI 8.7
Cos 8.5
Coth 8.5
Cot 8.5
CouF 8.8
CouG 8.8
 Coulomb wave function, irregular **CouG** 8.8
 Coulomb wave function, regular **CouF** 8.8
CO XTensor
CPos XStr1
CQ1 XClass
CQ3 XClass
 create function **XFun**
 create pattern **XFun**
CRep XStr1
CRev XStr8
 criterion 2.7
 criterion for pattern matching **Gen** 4.
 crystal energies **XLatsum**
Csch 8.5
Csc 8.5
CSD XClass
CS XStr1
 cumulative lists **XTri**
Cum XStat

SMP HANDBOOK / INDEX

- Cur1 XVecan
- currying Tier 7.7
- cursor operations 10.4
- curve fitting XGFit XLCh12
- Curve 10.2
- curvilinear coordinates XVecan
- CVar XClass
- Cx 2.1
- cycle Cyc 7.7
- Cyclic 7.7
- Cyc 7.7
- Dap XDap
- data analysis XFit XGFit
- data conversion XData1
- data point Err 2.1
- data presentation XHist
- data types Extr 4.
- data XData
- database 1.2
- Data XData
- DCf XCf
- deassignment 3.2
- debug output Trace 4.
- debugging aids 10.10
- declaration 3.2
- declaration, character 7.6
- declaration, type Tyset 4.
- decode Expl 10.5
- Decode XCode
- decrement Dec 3.2
- Dec 3.2
- default directories A.7
- default Null 2.2
- defaults 0.
- Default XVecan
- deferred simplification 3.5
- define rules 3.2
- defining values 3.2
- definite integration Int 9.4
- degrees Deg 8.4
- Degree XGr
- Deg 8.4
- delayed assignment 3.2 3.2
- delete parts Del 7.3
- deleting parts 3.2
- Delta 8.3
- Del 7.3
- denominator Den 7.9
- denominator, common Col 7.9 Rat 7.9
- Den 7.9
- depth 2.5 Dep 7.4
- depth XLev
- Dep 7.4
- derangements XSfct1
- derivative, partial D 9.4
- derivative, total Dt 9.4
- destroy input XKill10
- destroy output XKill10
- determinant Det 9.6
- determinant XLUP
- determinants XPer
- Det 9.6
- device-independent graphics A.5
- Dfct1 8.6
- diagonal matrices XMat1
- diagonal matrix XMat3
- diagonalization Simtran 9.6
- diagonals XMat2
- Diagp XMat3
- Diag XMat1
- difference equations XDiff
- differential equations XLap
- differential, total Dt 9.4
- differentials XIndep
- differentiation constant Const 4.
- differentiation D 9.4
- Diff XDiff
- digamma function Psi 8.7
- digit extraction XDig
- Dig XDig
- dilogarithm Li 8.7
- dimensional analysis XMKs
- dimensions Dim 7.4
- Dim 7.4
- DIM XFierz XTensor
- Dirac bilinear covariants XFierz
- Dirac function Delta 8.3
- Dirac gamma matrices XFierz
- directories, default A.7
- directory Dir 10.6
- Dir 10.6
- disassemble Dis 7.3
- discretization XHist
- Disc XDisc
- Disjp XSets
- disjunction, logical Or 5.
- disk file 10.3
- display 10.2 Pr 10.1
- display file Dsp 10.6
- display operations 10.4 At 7.2
- distribution 7.8 Dist 7.8
- distribution over lists XLProp
- distribution XExDot
- distribution, power Powdist 7.8
- distributive, list Ldist 7.7
- distributivity 7.8
- Dist 4.
- Dist 7.8
- Dis 7.3
- divide, matrix Mdiv 9.6
- division Div 8.2
- division, polynomial Pdiv 9.1
- divisor function Divsig 8.11
- divisors Divis 8.11
- Divis 8.11
- Divsig 8.11
- Div 8.2
- do loop Do 8.2

- documentation 1.2 2.9
- documentation, external file A.2
- domains 2.5
- domains **XLev**
- dot product **Dot** 8.2
- Dot** 8.2
- double factorial **Dfactl** 8.6
- double recursion **XHof**
- Do** 6.2
- DRand** **XRandD**
- draw 10.2
- DSol** **XDSol**
- Dsp** 10.6
- Dt** 9.4
- dummy expressions 2.6
- dummy index 2.8
- dummy symbol test **XGenp**
- dummy symbols 2.2
- Dvg** **XVcan**
- dyads 2.4
- dynamic programming **XMSet**
- D** 9.4
- E function, MacRobert **MacE** 8.9
- EccAnom** **XOrbit**
- echoing 1.1
- Edh** 10.5
- edit held form **Edh** 10.5
- edit mode 1.7
- edit **Ed** 10.5
- Ed** 10.5
- eigenvalue equation **XMat4**
- eigenvectors **Eig** 9.6
- Eig** 9.6
- Ei** 8.7
- elaboration 1.2
- electrodynamics **XG**
- element of list **Elem** 7.3
- element of **In** 7.5
- elementary functions 8.5
- elements, list 2.4
- Elem** 7.3
- elimination of equations **Sol** 9.3
- EllE** 8.10
- ellipses **Seq** 7.1
- Elliptic functions, Jacobian **JacAm** 8.10
- Elliptic integral of first kind **Ellk**
- Elliptic integral of second kind **EllE** 8.10
- Elliptic integral of third kind **EllPi** 8.10
- EllK** 8.10
- EllPi** 8.10
- else **If** 6.1
- Embed** **XLUP**
- encasement type extension **Exte** 4.
- encode **Impi** 10.5
- Encode** **XCode**
- end job 1.5
- end **Exit** 10.6
- enter matrix **XMat1**
- entier function **Floor** 8.3
- entier **XRnd**
- entries, list 2.4
- epsilon tensor **Sig** 9.6
- epsilon tensor **XLevi**
- Eqn** **XEqn**
- equality **Eq** 5.
- equality, numerical **Neq** 3.4
- equals 3.2
- equation **Eq** 5.
- equations, solution of **Sol** 9.3
- equivalence, expression 2.6
- equivalence, numerical **Neq** 3.4
- Eq** 5.
- ERand** **XRandC**
- erase 3.2
- Erfc** 8.8
- Erf** 8.8
- error correction 1.7
- Error function **Erf** 8.8
- Error function, complementary **Erfc** 8.8
- error, input 1.1
- errors **XChi2**
- errors, number with **Err** 2.1
- Err** 2.1
- escapes, monitor 1.6
- estimated size **XLenex**
- Euler circuits **XGr**
- Euler gamma function **Gamma** 8.7
- Euler Gamma function **XGammaS** **XGammaV**
- Euler integral of first kind **XGammaS** **XGammaV**
- Euler numbers **Eul** 8.7
- Euler polynomials **Eul** 8.7
- Euler's constant **Euler** 8.4
- Euler's totient function **Totient** 8.11
- Euler-Mascheroni constant **Euler** 8.4
- Eulerp** **XGr**
- Euler** 8.4
- Eulgam** **XEulgam**
- Eul** 8.7
- evaluation 3.1
- evaluation, order of 3.1 **Smp** 4.
- even number, test for **Evenp** 7.6
- even ordering **Sym** 7.7
- Evenp** 7.6
- Ev** 3.7
- exact integer **B** 2.1
- Excess** **XStat**
- exclusion **XAllbut**
- exclusive or **Xor** 5.
- ExDot** **XExDot**
- executable file 10.7
- execute **Run** 10.6
- existence test **XAny**
- exit 1.5 **Ret** 6.3
- exit codes A.7
- Exit** 10.6
- ExMDot** **XExDot**
- expansion 7.8 **Dist** 7.8
- expansion **XLenex**

SMP HANDBOOK / INDEX

expansion, power **PowDist** 7.8
ExpC XStat
ExpI 8.7
 explicit tensors **XCen**
 explode **ExpI** 10.5
ExpI 10.5
 exponent **ExpT** 7.9 **Pow** 8.2
 exponential fit **XFit**
 exponential function **Exp** 8.5
 exponential integral **Ei** 8.7 **ExpI** 8.7
 exponential notation 2.1
 expression size **Size** 10.8
 expressions 2.5
expr XTEST
ExpT 7.9
Exp 8.5
 extension, type **Extr** 4.
 external commands 1.6
 external file conventions A.2
 external file directory **Dir** 10.6
 external file information 1.2
 external file, copy **Save** 10.6
 external file, display **Dsp** 10.6
 external file, save **Save** 10.6
 external files 1.4 A.2
 external operations 10.6 A.3
 external program **Run** 10.6
 external programs 1.6
Exte 4.
 extract **XMask**
 extraction, part 7.3
 extrapolation **XItP**
Extr 4.
Ex 7.8
E 8.4
 factor **Cb** 7.9 **Fac** 9.1
 factor, numerical **Nc** 7.9
 factorial **FctI** 8.6
 factorial, double **DfctI** 8.6
 factorial, generalized **Gamma** 8.7
 factors of number **Nfac** 8.11
Fac 9.1
 false 5.
FctI 8.6
 fermion factors **XFierz**
For XPrime
 Feynman diagrams **XG**
Fib XFib
Fierz XFierz
 figurate numbers **XPolynum**
 file 10.3
 file characteristics **Open** 10.3
 file directories, default A.7
 file directory **Dir** 10.6
 file information 1.2
 file, code 10.7
 file, copy **Save** 10.6
 file, program 10.7
 files 1.4

fill holes **XContig**
 fill **XContig**
 filter collection **Tier** 7.7
 filters 2.3 A.3
 find part **Pos** 7.3
 find result 2.3
 find **XScan**
Find XTensor
 finite elements **XDiff**
Fiper XPerm0
 first **XScan**
 first-order logic **XLogicPr**
First XList0
FitExp XFit
FitPow XFit
Fit XFit
 fixed accuracy **XN**
 fixed precision **XN**
 flat functions 2.6
 flatten **Fiat** 7.7
Fiat 4.
Fiat 7.7
 floating point numbers 2.1
 floor **XRnd**
Floor 8.3
 flow control 6.
Fmt 10.1
FN XN
 folded size **Size** 10.8
 fonts, external file A.2
 for loop **For** 6.2
 forget past **XKillIO**
Fork 10.9
 form sublists **XSubI**
 form, test 7.6
 format **Fmt** 10.1
 format, syntactic **Sx** 10.1
 formatting **XPrtable**
 formatting, external file A.2
 FORTRAN language 10.7
 Forward differences **XDiff**
 forward differences **XLDiff**
For 6.2
 four vectors **XLor**
FPow XFPow
 fractional part **Floor** 8.3
 fractions 2.1
Fract XStat
FrC 8.8
 free core **Mem** 10.8
 free memory **Gc** 10.8
 free of **In** 7.5
 frequency **XInfo**
Freq XInfo
 Fresnel function **FrC** 8.8 **FrS** 8.8
FrS 8.8
 Frobenius method **XDSol**
From10 XBase
 full list, test for **FullP** 7.6

SMP HANDBOOK / INDEX

- Fullp 7.6
- function evaluation 3.1
- function evaluation XLItP
- function fitting XGFit XLCh12
- function, test for Projp 7.6
- functional form XFit
- functional independence XIndep
- Functional powers XFPow
- functions 2.3
- functions, mathematical 8.
- functions, transcendental 8.5
- Fun XFun
- Fz XFierz
- F 2.1
- G function, Meijer Mei 8.9
- g.c.d., polynomial Pgcd 9.1
- gamma function, Euler Gamma 8.7
- gamma function, incomplete Gamma 8.7
- gamma matrix algebra XG
- Gamma 8.7
- garbage collect Gc 10.8
- Gauss hypergeometric function Hg 8.9
- Gaussian distribution XCh12
- Gcd 8.11
- Gc 10.8
- Gd 8.5
- Gegenbauer functions Geg 8.9
- Geg 8.9
- general recursion XAck
- general symmetry Greor
- Generalized Fibonacci sequence XHof
- Generalized hypergeometric function Ghg 8.9
- generalized power XAck
- generalized product XIter
- generalized sum XIter
- generalized trace XMat4
- generalized traces XCon
- generalized zeta function Zeta 8.7
- generate array Ar 7.1
- generate symbol Make 10.5
- generic expressions 2.6
- generic predicate XGenp
- generic symbols 2.2
- Genocchi XGenocchi
- Genp XGenp
- Gentr XMat4
- genus of expressions Gen 4.
- Gen 2.6
- Gen 4.
- geometrical figures XBox
- Get 10.3
- Ge 5.
- GFit XGFit
- Ghg 8.9
- GIGI A.5
- Ginds XG
- global objects 1.3
- global switch Pre 1.3
- global switches Post 1.3
- GL XTensor
- GMean XStat
- Gm XTensor
- golden ratio Phi 8.4
- Golden ratio XFib
- goodness of fit XLCh12
- goto Jmp 6.3
- GQInt XGQInt
- gradient 0 9.4
- Grad XVecan
- graph 10.2
- graph equivalence XGr
- graph isomorphism XGr
- graph representation XGr
- graph traversibility XGr
- graphical input 10.4
- graphical output Open 10.3
- graphics output A.5
- Graph 10.2
- greater than Gt 5.
- greatest common divisor Gcd 8.11
- greatest common divisor, polynomial Pgcd 9.1
- greatest integer function Floor 8.3
- group XSub1 XTri
- grouping 2.10
- groups 2.4
- Gt 5.
- Gudermannian function Gd 8.5
- guest accounts A.7
- GU XTensor
- h.c.f., polynomial Pgcd 9.1
- halt 1.5
- Haltp XHalt
- Hamilton circuits XGr
- Hamp XGr
- Hankel function BesH1 8.8 BesH2 8.8
- hard code 10.7 Cons 10.7
- hard copy Hard 10.6
- Hard 10.6
- Harm XHarm
- hash code Hash 7.4
- Hash 7.4
- haversine XTrigR
- head XList0
- Heavyside function Theta 8.3
- height Dep 7.4
- held expression, test for Heldp 7.6
- held form 3.5
- Heldp 7.6
- help 1.2
- Hermite function Her 8.8
- hermitean adjoint XMat4
- Her 8.8
- hexadecimal XBase
- Hg 8.9
- hidden surface Surf 10.2
- histograms XClass
- Hist XHist
- hline XPrtable

- HMean** XStat
Hof XHof
 hold expression **Hold** 3.5
Hold 3.5
Horn XHorn
hthing XPrtable
hv XPrtable
 hyperbolic cosine integral function **Coshi** 8.7
 hyperbolic sine integral function **Sinhi** 8.7
 hypergeometric function, confluent **Chg** 8.8
 Hypergeometric function, Gauss **Hg** 8.9
 Hypergeometric function, generalized **Ghg** 8.9
 identifier **Lbl** 6.3
 identity **Eq** 5.
If 6.1
 image 2.3
 imaginary number **Cx** 2.1
 imaginary number, test for **Imagp** 7.6
 imaginary part **Im** 8.3
 imaginary unit **I** 2.2
Imagp 7.6
 immediate assignment 3.2 3.2
 immediate simplification 3.6
 implication, logical **Imp** 5.
 implode **ImpI** 10.5
ImpI 10.5
 impulse function **Delta** 8.3
Imp 5.
Im 8.3
 incidence matrix **XGr**
 includes **In** 7.5
 inclusive or **Or** 5.
 incomplete beta function **Beta** 8.7
 increment **Inc** 3.2
Inc 3.2
 indefinite integration **Int** 9.4
 indefinite summation **Sum** 9.2
Indep XIndep
 index in list **Ind** 7.3
 indices 2.4
 indicial tensor calculus **XTE**
Ind 7.3
 inequality **Uneq** 5.
 infile **Get** 10.3
 infinite recursion 3.1
 infinity **Inf** 2.2
 infix form 2.11 **Sx** 10.1
 information 1.2
 information, external file 1.2
Inf 2.2
 initialization A.7 **Init** 10.6
Init 10.6
Init 4.
 inner "product", generalized **Inner** 9.6
 inner product **Dot** 8.2
 inner products **XCon**
Inner 9.6
 input editing 1.7
 input error 1.1
 input expression **#I** 1.3
 input forms 2.10 2.11
 input lines 1.1
 input medium 10.3
 input operations 10.1
 input processing 2.11
 input simplification 3.6
 input syntax 2.
 input **Get** 10.3 **Rd** 10.1
 input, graphical 10.4
 insert **XList@**
 insoluble problem **XHalt**
Ins XList@
Intbit XBit
 integer conversion **XBase**
 integer divide **Mod** 8.3
 integer equations **XDios**
 integer part **Ceil** 8.3 **Floor** 8.3
 integer, arbitrary length **B** 2.1
 integer, test for **Intp** 7.6
 integers 2.1
 integration **Int** 9.4
 interactive matrix input **XMatI**
 interactive procedures 6.3
 internal code 10.7
 internal object **Sys** 4.
 internal representation **Struct** 10.10
 internal variables **Lcl** 6.3
 interpolation **XItp**
 interrupts 1.5
 intersection **Inter** 7.7
Inter 7.7
Inter XSets
Intp 7.6
Int 9.4
int XTEST
Inum XPrime
 inverse functions **SoI** 9.3
 inverse, matrix **Minv** 9.6
 inversion of equations **SoI** 9.3
 inversion **Not** 5.
 invert replacement **Irep** 3.3
 invert **Rev** 7.7
In 7.5
IRand XRand0
Irep 3.3
 Irregular Bessel function **BesY** 8.8
 Irregular Coulomb wave function **CouG** 8.8
 Irregular spherical Bessel function **BesY** 8.8
Isop XGr
Is 5.
 iterated functions **XFPow**
 iteration 6.2
 iteration **XIter**
Iter XIter
Itp XItp
I 2.2
JacAm 8.10
JacCd 8.10

SMP HANDBOOK / INDEX

- JacCn 8.10
- JacCs 8.10
- JacDc 8.10
- JacDn 8.10
- JacDs 8.10
- JacNc 8.10
- JacNd 8.10
- JacNs 8.10
- Jacobi functions JacP 8.9
- Jacobi symbol Jacsym 8.11
- Jacobi ϑ functions Jac θ 8.10
- Jacobian elliptic functions JacAm 8.10
- JacP 8.9
- JacSc 8.10
- JacSd 8.10
- JacSn 8.10
- Jacsym 8.11
- Jac θ 8.10
- Jac XWron
- Jmp 6.3
- Jnum XPrime
- job recording 1.4
- job termination 1.5
- joinWait 10.9
- Jonquiere function Li 8.7
- Jordan form Simtran 9.6
- Jordan's function Jor 8.11
- Jor 8.11
- joystick 10.4
- jump Jmp 6.3
- KD XTensor
- Kelbe 8.8
- Kelke 8.8
- Kelvin function, complex Kelbe 8.8 Kelke 8.8
- Kepler's laws XOrbit
- keywords 1.2
- kill input XKi11110
- kill labels XKi11110
- kill lines XKi11110
- kill output XKi11110
- killing values 3.2
- Ki11110 XKi11110
- Kronecker product Omult 8.2
- Kummer function Chg 8.8
- Kummer's U function KumU 8.8
- KumU 8.8
- Kurt XStat
- label Lbl 6.3
- Lagrangian interpolation XLitp
- Laguerre function Lag 8.8
- Lag 8.8
- lambda expression 2.7
- lambda expression XFun
- language analysis XInfo
- language structure 2.
- language, intermediate 10.7
- Lap XLap XVecan
- larger than Gt 5.
- Last 7.3
- Latsum XLatsum
- Laurent series Ps 9.5
- Lbl 6.3
- Lcl 6.3
- LCM XLCM
- LCoef XPoly
- LdEq XLdEq
- Ldet XLUP
- LDiag XMat2
- LDiff XLDiff
- LDig XDig
- Ldist 4.
- Ldist 7.7
- Ldot XLor
- LDRand XRandL
- leading zeroes XPad
- least integer function Ceil 8.3
- least squares fit XFit
- left justify XPad
- Left XTuring
- Legendre functions of second kind LegP 8.9
- LegP 8.9
- LegQ 8.9
- Lenex XLenex
- length Len 7.4
- Lenlev XLev
- Len 7.4
- Lerch transcendent Ler 8.7
- Ler 8.7
- less than Gt 5.
- let 3.2
- letters XChar
- levels 2.5
- Levi-Civita symbol Sig 9.6
- Levi XLevi
- Lev XLev
- lexical ordering Ord 5.
- LExpt XPoly
- library A.2
- library information 1.2
- light pen 10.4
- limit Lim 9.5
- Lim 9.5
- LInd XInd
- line printer Hard 10.6
- linear algebra XLUP
- linear equation XLUP
- linear fit XFit
- Line 10.2
- LIn XList1
- Liouville's function Lio 8.11
- Lio 8.11
- LISP CAR XList0
- list creation List 7.1
- list distribution Ldist 7.7
- list distributive Ldist 7.7
- list element removal XAllbut
- list function Ldist 7.7
- list generation 7.1Ar 7.1

- list manipulation 7.7
- list simplification 3.1
- list substitution `XList1`
- list template `List` 7.1
- list, test for `List` 7.1
- `Listp` 7.6
- lists 2.4
- lists `XList1`
- `List` 7.1
- `list XTEST`
- `LItP2 XLItP`
- `LItP3 XLItP`
- `LItP XLItP`
- `Li` 8.7
- `Lmdiv XLUP`
- `Lminv XLUP`
- `loadGet` 10.3
- `loadedCons` 10.7
- `Load` 10.7
- local objects `XX` 6.3
- local variables `Lci` 6.3
- `locateL` 10.4
- `locationPos` 7.3
- logarithm function `Log` 8.5
- logarithm integral `Logi` 8.7
- logical operations 5.
- `Logi` 8.7
- `Log` 8.5
- Lommel function `Lom` 8.8
- `Lom` 8.8
- look-up `XMSet`
- `Loop` 6.2
- `Lowerp XChar`
- `LPad XPad`
- `LPart XLPart`
- `LPos XList1`
- `LProd XLArith`
- `LProp XLProp`
- `Lpr` 10.1
- `LRand XRandL`
- `Lrpt XList0`
- `LSub XList1`
- `LSum XLArith`
- `LS XList1`
- `Lup XLUP`
- `L` 10.4
- `MacE` 8.9
- Maclaurin series `Ps` 9.5
- macro redefinition 2.11
- MacRobert E function `MacE` 8.9
- Madelung sums `XLatsum`
- `mail Send` 10.6
- make cubical `XContig`
- make generic `XFun`
- make rectangular `XContig`
- make square `XContig`
- make symbol name `Make` 10.5
- `Make` 10.5
- Mangoldt Λ function `ManL` 8.11
- `ManL` 8.11
- `MANom XOrbit`
- manual 1.2
- mapping `XDap`
- `Map` 7.2
- margins, display A.5
- `markL` 10.4
- Markov expression `Rex` 7.10
- `Mark` 2.3
- `Mask XMask`
- matching, pattern 2.6
- `Match` 2.6
- mathematical functions 8.
- `Matp XMat3`
- matrices 2.4
- matrices `XPrtable`
- matrix adjoint `XMat4`
- matrix classes `XMat3`
- matrix divide `Mdiv` 9.6
- matrix equations `XLdEq`
- matrix generation `Ar` 7.1
- matrix inverse `Minv` 9.6
- matrix inverse `XLUP`
- matrix manipulation 9.6
- matrix power `XMat4`
- matrix types `XMat3`
- matrix `XLUP`
- maximum memory A.7
- maximum `Max` 8.3
- `Maxind XInd XMaxind`
- `Max` 8.3
- `Mdiv` 9.6
- `MD XStat`
- `Mean XStat`
- `Med XStat`
- Meijer G function `Mei` 8.9
- `Mei` 8.9
- member `In` 7.5
- memory management 10.8
- memory requirement `XLenex`
- memory unit A.6
- memory usage `Mem` 10.8
- memory, maximum A.7
- memory, share `Share` 10.8
- `Mem` 10.8
- menus 1.2
- `Mer XPrime`
- messages 10.9
- `Mgen` 4.
- minimum `Min` 8.3
- Minkowski space `XLor`
- minors `XMat2`
- `Minor XMat2`
- `Minv` 9.6
- `Min` 8.3
- Mobius μ function `Mob` 8.11
- `Mob` 8.11
- model comparison `XLCh12`
- model fitting `XLCh12`

SMP HANDBOOK / INDEX

- Modified Bessel function **BesI** 8.8 **BesK** 8.8
- modify input **Ed** 10.5
- modify part **At** 7.2
- modular arithmetic **XQuadres**
- modulus **Abs** 8.3 **Mod** 8.3
- modulus **XAbs**
- modulus, polynomial **Pmod** 9.1
- Mod** 8.3
- Mon** **XStat**
- monitor commands **A.3**
- monitor escapes 1.6
- monitor program **Run** 10.6
- Monte Carlo **Rand** 8.3
- mouse 10.4
- move file **Save** 10.6
- mpoly** **XTEST**
- Mpow** **XMat4**
- MRd** **XMat1**
- mu** function **XScan**
- multi-generic symbols 2.2
- multigeneric symbols 2.6
- multinary operator 2.11
- multinary **Fiat** 7.7
- multinomial coefficient **Comb** 8.6
- multiple apply **Map** 7.2
- multiple assignment **XLProp**
- multiple integration **Int** 9.4
- multiple precision number **F** 2.1
- multiplication **Mult** 8.2
- multiplication, input of 2.10
- multiply **Rep1** 7.1
- multiplying out expressions 7.8
- multivalued functions 8.1
- Multset** **XSets**
- Mult** 8.2
- n-ary **Fiat** 7.7
- name, make **Make** 10.5
- names 2.2
- naming, external file **A.2**
- Natp** 7.6
- natural language 1.2
- natural number, test for **Natp** 7.6
- Nc** 7.9
- negation **Not** 5.
- Neq** 3.4
- nested functions **XFPow**
- nesting levels 1.7
- nesting **Dep** 7.4
- network theory **XGr**
- Newton's method **XConsSol**
- Nfac** 8.11
- NF** **XN**
- NMake** **XDig**
- NMap** **XNMap**
- nnint** **XTEST**
- nodes **XGr**
- Nodes** **XGr**
- norm **Abs** 8.3
- normal distribution **XChi2**
- Norm** **XNorm**
- notation 0.
- Not** 5.
- Np** 2.3
- NRand** **XRandC**
- NSol** **XNSol**
- null list 2.4
- null projection **Np** 2.3
- Null** 2.2
- number construction **XDig**
- number conversion **A.3**
- number of equivalence relations **XBell**
- number theory functions **XPolynum**
- Number theory **XQuadres**
- number, test for **Numbp** 7.6
- numbers 2.1
- Numbp** 7.6
- numb** **XTEST**
- numerator **Num** 7.9
- numerical coefficient **Nc** 7.9
- numerical coefficients 2.5
- numerical constant **Const** 4.
- numerical differentiation **D** 9.4
- numerical equality testing **Neq** 3.4
- numerical errors 2.1
- numerical evaluation 3.4
- numerical evaluation of polynomials **XHorn**
- numerical factor **Nc** 7.9
- numerical functions 8.3
- numerical integration **Int** 9.4
- numerical integration **XGQInt**
- numerical inversion **XConsSol**
- numerical overflow **A** 2.1
- numerical products **Prod** 9.2
- numerical programs 10.7
- numerical quadrature **XGQInt**
- numerical summation **Sum** 9.2
- numerical truncation 3.4
- numerical value **XAbs**
- Num** 7.9
- N** 3.4
- octal **XBase**
- odd number, test for **Oddp** 7.6
- odd ordering **Asym** 7.7
- Oddp** 7.6
- Omult** 8.2
- Open** 10.3
- operating system commands 1.6
- operational methods **XLap**
- operator 2.3
- operator form 2.11
- operators 2.10
- optimization 10.7
- options 0.
- oracle **XHalt**
- ORand** **XRandL**
- orbital elements **XOrbit**
- order of evaluation 3.1 **Smp** 4.
- order of operators 2.10

- orderSort 7.7
- orderingOrd 5.
- ordering, filterReor 7.7
- ordinary differential equationsXSerSol
Ord 5.
- orthogonal coordinatesXVecan
Or 5.
- outer "product", generalizedOuter 9.6
- outer productOmult 8.2
- Outer 9.6
- outfilePut 10.3
- outlineTree 7.4
- output expression #0 1.3
- output formFmt 10.1Pr 10.1
- output formatSx 10.1
- output forms 2.12
- output medium 10.3
- output operations 10.1
- output syntaxPr 10.1Sx 10.1
- outputLpr 10.1Put 10.3
- overall factorNc 7.9
- padXContig
- Pade approximantRa 9.5
- PairsXTup
- pairwise differencesXLDiff
- pairwise subtractionXLDiff
- paper copyHard 10.6
- PApXPlot
- PapXProj
- Parabolic cylinder functionsPar 8.8
- parallel evaluationSer 4.
- parallel processing 10.9
- parameter determinationXFit
- parameter fittingXGFit
- parameters 2.2
- parametric plotGraph 10.2
- Para 10.9
- parentheses 2.10
- parentheses, output of 2.12
- parenthesis levels 1.7
- parsing 2.10 2.11
- part extraction 7.3At 7.2
- part extractionXLev
- part modificationAt 7.2
- part removal 7.3
- part selectionAt 7.2
- partXList8
- partial derivativesXIndep
- partial differential equationsXVecan
- partial differentiationD 9.4
- partial fractionPf 9.1
- partial simplification 3.7
- partition functionPart 8.6
- parts of expressions 2.5
- parts, addition of 3.2
- parts, deletion of 3.2
- Part 8.6
- Par 8.8
- pass outputRun 10.6
- patterns 2.6
- PauseXPause
- PCatXPlotXProj
- PcompXPerm1
- Pcp 8.8
- PCumXRandD
- PDeltaXPR
- PDimXYoung
- PDiosXDios
- Pdiv 9.1
- PeelXPeel
- pentagonal numbersXPolynum
- permanent recordSave 10.6
- PermpXPerm8
- permutation symmetryReor 7.7
- PerXPer
- Pf 9.1
- PGammaXPR
- Pgcd 9.1
- Phi 8.4
- physical constantsXMKs
- physical quantitiesXDimXMKs
- picture 10.2Fmt 10.1
- PIntXPR
- pintXTEST
- PinvXPerm1
- Pi 8.4
- PLamXPR
- PlanXPR
- planeSurf 10.2
- PIhistXPIhist
- plot 10.2
- plottingXBox
- Plot 10.2
- Plus 8.2
- PmatXLUP
- Pmod 9.1
- PNormXRandD
- pnumbXTEST
- Pochhammer symbolPoc 8.7
- Poc 8.7
- pointPt 10.2
- pointer 10.4
- Poisson-Charlier polynomialsPcp 8.8
- polar plotGraph 10.2
- PolarXPolar
- polygamma functionPsi 8.7
- polylogarithmLi 8.7
- polynomial g.c.d.Pgcd 9.1
- polynomial manipulation 9.1
- polynomial modulusPmod 9.1
- polynomial quotientPdiv 9.1
- polynomial rearrangementXHorn
- polynomial rootsXDisc
- polynomial, test forPolyp 7.6
- PolynumXPolynum
- Polyp 7.6
- polyXTEST
- positionL 10.4Pos 7.3

SMP HANDBOOK / INDEX

- positional notation `XBase XDig`
- postfix form 2.11 `Sx 10.1`
- postprocessing `Post 1.3`
- `Post 1.3`
- `Pos 7.3`
- `Powdist 4.`
- `Powdist 7.8`
- power distribution `Powdist 7.8`
- power expansion 7.8 `Powdist 7.8`
- power fit `XFit`
- power series `Ps 9.5`
- power series `XDSol XSerSol`
- power set `XLPart`
- power `Pow 8.2`
- powers of `Expt 7.9`
- `Powset XSets`
- `Pow 8.2`
- `PPI XPR`
- `Ppow XPerm1`
- pre-simplification 3.6
- precedence 2.10
- precedence definition 2.11
- precision 2.1
- precision, arbitrary `F 2.1`
- precision, multiple `F 2.1`
- predicate testing 5.
- predicate `P 5.`
- predicates 7.6
- prefix form 2.11 `Sx 10.1`
- preparation `Init 10.6`
- prepend `XList0`
- preprocessing `Pre 1.3`
- `Prep XList0`
- prerequisites `Init 10.6`
- pretty-printing `XEqn`
- `Pre 1.3`
- `Prh 10.1`
- prime decomposition `XPow`
- prime factors `Nfac 8.11`
- prime number `Prime 8.11`
- `Primep XPrimep`
- `Prime 8.11`
- print file `Dsp 10.6`
- print form `Fmt 10.1`
- print held form `Prh 10.1`
- print `Pr 10.1`
- print, linear `Lpr 10.1`
- print, one-dimensional `Lpr 10.1`
- print, two-dimensional `Pr 10.1`
- printing properties `Pr 10.1`
- printing `XPrtable`
- printout `Hard 10.6`
- `Prmat XPrtable`
- probability functions `XChi2`
- problem report `Send 10.6`
- procedures 6.3
- process control 10.9
- processing 3.1
- `Proc 6.3`
- product `Mult 8.2 Prod 9.2`
- `Prod 9.2`
- profiling `Time 10.8`
- program construction 10.7
- program control 6.
- program files A.4
- program, run `Run 10.6`
- programming aids 10.10
- programs 6.3
- programs, external A.3
- `Prog 10.7`
- `Projcp XMat3`
- projection evaluation 3.1
- projection generation 7.1
- projection manipulation 7.7
- projection matrix `XMat3`
- projection simplification 3.1
- projection, test for `Projp 7.6`
- projections 2.3
- projector 2.3
- `Projp 7.6`
- `Proj 7.3`
- properties 4.
- property assignment `Prset 4.`
- property indirection `Type 4.`
- property list 4.
- property transfer `Type 4.`
- property `XLProp`
- `Prop 4.`
- protocol 1.4
- `Prset 4.`
- `PrTF XLogicPr`
- `Pr 10.1`
- `Pr 4.`
- pseudotensor unit `Sig 9.6`
- `PSig XPR`
- `Psi 8.7`
- `Pamp XPamp`
- `PSqrt XPR`
- `PsSol XSerSol`
- `Ps 9.5`
- `Pt 10.2`
- pure function 2.7
- push variable `Lcl 6.3`
- `Put 10.3`
- `PXi XPR`
- `P 5.`
- `Q1 XStat`
- `Q3 XStat`
- `QChi2 XChi2`
- `QD XStat`
- `Qskew XStat`
- `Qtr XLUP`
- `Quadres XQuadres`
- quantization `XHist`
- quantum field theory `XG`
- `Quant XLogic2`
- queries 1.2
- quit 1.5

SMP HANDBOOK / INDEX

quotient Div 8.2
 quotient, polynomial Pdiv 9.1
 quoting 3.5
 R2 XTensor
 R4 XTensor
 Racah 6-j symbol Rac 8.6
 Rac 8.6
 Rademacher's formula XZeta2S
 radians Deg 8.4
 radicals XPow
 radix arithmetic XBase
 Ramp XRamp
 random expression Rex 7.10
 random number Rand 8.3
 Rand 8.3
 Range XStat
 Ranmp XRpoly
 Ranup XRpoly
 RAr XRAr
 rational approximation Ra 9.5
 rational expression manipulation 7.9
 rational number, arbitrary length B 2.1
 rational number, test for Ratp 7.6
 rational numbers 2.1
 rationalize Rat 7.9
 Ratp 7.6
 Rat 7.9
 ravel Flat 7.7
 Ra 9.5
 Rdh 10.1
 Rd 10.1
 read file Get 10.3
 read held form Rdh 10.1
 read Rd 10.1
 real number, test for Realp 7.6
 real part Re 8.3
 real time operations Clock 10.9
 real-time interrupts 1.5
 Realp 7.6
 reclaim memory Gc 10.8
 reclaim memory XKillIO
 record 10.3 Open 10.3
 recording, job 1.4
 records 2.4
 rectangular array, test for Fullp 7.6
 recurrence relations XRAr
 recursion 3.1 Smp 4.
 recursion control Rec 4.
 recursion testing XHot
 recursive function theory XAck
 Rec 4.
 reduce memory Share 10.8
 reduced residue system Rrs 8.11
 reduction XExDot
 references 0.
 reflection formula XZeta2S
 regions 2.5
 regions XBox
 Regression XGFit

Regs XGr
 Regular Bessel function BesJ 8.8
 Regular Coulomb wave function CouF 8.8
 Regular spherical Bessel function Besj 8.8
 Relab XGr
 relation Eq 5.
 relational operations 5.
 relativistic mechanics X Lor
 release expression Rel 3.5
 Rel 3.5
 remainder Mod 8.3
 remainder, polynomial Pmod 9.1
 removal, part 7.3
 remove list brackets Flat 7.7
 remove parts Del 7.3
 remove XList0 XMask
 removing values 3.2
 rencontres numbers XSfctI
 reorder Reor 7.7
 reordering, filter Reor 7.7
 reorderings XArperm
 Reor 4.
 Reor 7.7
 Report XReport
 Rept 3.3
 repeat counts 2.5
 repeat Repl 7.1 Rpt 6.2
 repeat XList0
 repeated transformation XMat4
 repetition Rpt 6.2
 replace text 1.7
 replacement 3.3
 replacement type extension Extr 4.
 replicate Repl 7.1
 replicate XList0
 replication XIter
 Repl 7.1
 report generation XPrtable
 report Send 10.6
 representation, internal Struct 10.10
 representation, special Extr 4.
 Rep 3.3
 reshape Trans 9.6
 residue system, reduced Rrs 8.11
 restart 3.2
 restrict matching Gen 4.
 return Ret 6.3
 Ret 6.3
 reverse Rev 7.7
 revert Rev 7.7
 revise Edh 10.5
 Rev 7.7
 Rex 7.10
 Re 8.3
 Riemann sheets 8.1
 Riemann zeta function Zeta 8.7
 right justify XPad
 Right XTuring
 rings XQuadres

- RMS XStat
 Rm XList@
 Rnd XRnd
 rooted planar trees XCatalan
 Rot2 XRot2
 Rot3 XRot3
 rotate Cyc 7.7
 rotate XDap
 RotM2 XRot2
 RotM3 XRot3
 rounding Floor 8.3
 RPad XPad
 Rpt 6.2
 rpt XTEST
 Rrs 8.11
 Rslt 9.1
 ruled XPrtable
 rules, application of 3.1
 rules, definition of 3.2
 rules, simplification 3.3
 run command 1.6
 run program Run 10.6
 Run 10.6
 RXTensor
 save definitions Put 10.3
 save memory XKillIO
 save Open 10.3
 Save 10.6
 saving expressions 1.4
 scalar product Dot 8.2
 scalar products XExDot
 scalars XExDot
 scales of notation XBase
 Scalp XExDot
 scan XAny XScan
 Scan XScan
 Scf XVecan
 screen-oriented input 10.4
 script 1.4
 SD XStat
 search XScan
 Sech 8.5
 Sec 8.5
 seed random number Rand 8.3
 segments 6.3
 select statement Sel 6.1
 select At 7.2L 10.4
 select XList@ XMask
 Sel 6.1
 semantics 2.
 semaphores 10.9
 Send 10.6
 separation of variables XVecan
 sequence generation Seq 7.1
 Sequence generation XRRr
 sequence of expressions Np 2.3
 sequence positions XList1
 sequences 2.4
 sequences XSubi XTri
 sequential evaluation Ser 4.
 Seq 7.1
 serial evaluation Ser 4.
 series approximations 9.5
 series truncation Rx 9.5
 series, power Ps 9.5
 Ser 4.
 set Union 7.7
 Setd 3.2
 sets 2.4
 sets of equations XLdEq
 setting values 3.2
 Set 3.2
 Sfct1 XSfct1
 Shannon entropy XInfo
 Shan XInfo
 share memory Share 10.8
 Share 10.8
 shell escapes 1.6
 shell scripts A.3
 Showtime XShowtime
 shut Close 10.3
 side effects 3.2
 sigma function, Weierstrass Weiz 8.10
 Sigma XPauli
 signature Sig 9.6
 Sign 8.3
 Sig 9.6
 silent processing 1.1
 silent Close 10.3
 similarity transformation Simtran 9.6
 similarity XDim
 simplification 3.1
 simplification control Smp 4.
 simplification on input 3.6
 simplification, partial 3.7
 simplification, rational expressions 7.9
 Simtran 9.6
 simultaneous equations XLdEq
 sine integral function Sini 8.7
 Sinh 8.7
 Sinh 8.5
 Sini 8.7
 Sin 8.5
 size Len 7.4
 Size 10.8
 Si 3.3
 skeleton output Fmt 10.1
 skeleton Tree 7.4
 sketch Tree 7.4
 Skew XStat
 smoothing XFit XItp
 smp.end A.7
 smp.ini A.7
 smp.out 1.4
 smp.par A.7
 smpplot A.5
 Smp 3.1
 Smp 4.

SMP HANDBOOK / INDEX

- Solar system **XOrbit**
- solution of equations **Sol** 9.3
- solve **Sol** 9.3
- Sol** 9.3
- Some XLogic2**
- sorting **Ord** 5.
- Sort** 7.7
- spaceSize** 10.8
- Spare XSpare**
- special expression **Mark** 2.3
- special output form **Fmt** 10.1
- special output forms 2.12
- special-purpose programs A.2
- Spec XTuring**
- Spence function **Li** 8.7
- spherical Bessel function, irregular **Besy** 8.8
- spherical Bessel function, regular **Besj** 8.8
- spline **Curve** 10.2
- spur **Tr** 9.6
- Sqmatp XMat3**
- Sqrt** 8.2
- square matrix **XMat3**
- square root **Sqrt** 8.2
- square root **XPow**
- square **XBox**
- stack variables **Lcl** 6.3
- standard form 2.1
- starred output 2.12
- startup A.7
- Start XTuring**
- state table **XLogicPr**
- statement blocks 6.3
- statistical expression analysis 7.10
- statistical expression generation 7.10
- statistics **XChi2 XGFit**
- status interrupt 1.5
- status **Mem** 10.8
- Status XStatus**
- step function **Theta** 8.3
- Step** 10.10
- Sti1** 8.6
- Sti2** 8.6
- Stirling numbers **XBell**
- Stirling numbers, first kind **Sti1** 8.6
- Stirling numbers, second kind **Sti2** 8.6
- stop 1.5 **Exit** 10.6
- stop record **Close** 10.3
- storage 10.3
- storage management 10.8
- StrH** 8.8
- string manipulation 10.5
- string out **Lpr** 10.1
- string, make **Imp1** 10.5
- strings 2.2
- StrL** 8.8
- structural operation **XLev**
- structural operations 7.
- structure determination 7.4
- structure **Struct** 10.10
- Struct** 10.10
- Struve function **StrH** 8.8
- Struve function, modified **StrL** 8.8
- subfunctions 6.3
- sublist positions **XList1**
- Subl XSubl**
- subparts of expressions 2.5
- Subp XSets**
- subroutines 6.3
- subscript **Fmt** 10.1
- subscripted variables 2.3
- subscripts 2.3 2.4
- subsidiary input **XI** 6.3
- subsidiary output **XO** 6.3
- substitution 3.3
- Sub XSets**
- such that **Gen** 4.
- sumPlus** 8.2
- sums of reciprocal powers **XZetaR**
- Sum** 9.2
- superscript **Fmt** 10.1
- surface **Surf** 10.2
- Surf** 10.2
- suspend processing 1.5
- switch statement **Sel** 6.1
- switch, global **Post** 1.3 **Pre** 1.3
- Sxset** 2.11
- Sx** 10.1
- symbol evaluation 3.1
- symbol, test for **Symbp** 7.6
- symbolic-numeric interface **XGQInt**
- symbols 2.2
- symbols, list of **Cont** 7.5
- Symbp** 7.6
- symmetric matrix **XMat3**
- symmetric ordering **Sym** 7.7
- symmetric **Comm** 4.
- symmetries **XArperm**
- symmetry **Reor** 7.7
- symmetry, general **Greor**
- Sympol XSympol**
- Symp XMat3**
- Sym** 7.7
- synchronize processes **Wait** 10.9
- syntax 2. **Sx** 10.1
- syntax error 1.1
- syntax extension 2.11
- syntax modification 2.11
- syntax, output **Pr** 10.1 **Sx** 10.1
- system characteristics A.6
- system-defined object **Sys** 4.
- system-defined objects 2.2
- Sys** 4.
- S** 3.3
- table generation **Ar** 7.1
- tables 2.4
- tabs A.5
- tabular data **XPrtable**
- tabulation **XPrtable**

- tagLb1 6.3
- tailXList0
- takeXList0
- Tanh 8.5
- Tan 8.5
- TapeXTuring
- tautology testingIs 5.
- Taylor seriesPs 9.5
- TdiagXMat2
- Tektronix 4010 A.5
- Tektronix 4025 A.5
- template application 7.2
- templates 2.7
- temporary variablesLc1 6.3
- tempXTEST
- TenpXTensor
- tensor canonicalizationXTE_x
- tensor generationAr 7.1
- tensor manipulation 9.6
- tensor symmetriesXTE_x
- tensors 2.4
- terminal 10.3
- terminal characteristics A.5open 10.3
- terminate jobExit 10.6
- termination A.7
- termination, job 1.5
- termination, line 1.1
- terms, number ofLen 7.4
- ternaryXBase
- TernXTern
- test for any elementsXR_{ny}
- test form 7.6
- test, character 7.6
- test, numerical equalityNeq 3.4
- tests 6.1
- text 1.2 2.2 2.9
- text editing 1.7
- text manipulation 10.5
- text manipulationXCharXDig
- text preprocessing 2.11
- text processingXEqnPr
- text-formattingXEqn
- textual forms 0.
- textual replacement 2.11
- TExXTE_x
- thenIf 6.1
- theorem provingIs 5.
- theta functions, JacobiJacth 8.10
- Theta 8.3
- three-dimensional plotGraph 10.2
- tiered list 2.4
- Tier 4.
- Tier 7.7
- timeClock 10.9
- Time 10.8
- timing A.6#T 1.3Time 10.8
- TinvXLUP
- TkXList0
- To10XBase
- ToArcXGr
- ToCXPermC
- ToIndXInd
- ToLowerXChar
- ToLXInd
- ToNodeXGr
- topbotXPrtable
- ToPXPermC
- Toronto functionTor 8.8
- Tor 8.8
- total derivativesXIndep
- total differentiationDt 9.4
- total lengthXLen_x
- totally antisymmetric tensorXLevi
- totient function, Euler'sTotient 8.11
- Totient 8.11
- trace 1.5Step 10.10Tr 9.6
- trace identitiesXG
- traceback 1.5
- Trace 4.
- trailing zeroesXPad
- transcendental functions 8.5
- transfer controlJmp 6.3
- transformation of coordinate systemsXVecan
- translation 10.7
- transposeTrans 9.6
- Trans 9.6
- tree structure 2.5
- Tree 7.4
- triangular numbersXPolygon
- triangularize matrixTriang 9.6
- Triang 9.6
- trigamma functionPsi 8.7
- trigonometric functions 8.5
- TriXTri
- TROFFXEqnPr
- true 5.
- truncationR_x 9.5
- truncationXR_{nd}
- truncation, integerFloor 8.3
- TruncXLUP
- Tr 9.6
- TupaXTup
- TupoXTup
- TupXTup
- tutorials 1.2
- two-dimensional outputFmt 10.1
- two-dimensional outputXEqnPr
- type assignmentTyset 4.
- type assumptions 10.7
- type castingXNXR_{nd}
- type coercionXN
- type conversionXN
- type declarationTyset 4.
- type declarations 10.7
- type definitionType 4.
- type extensionExtr 4.
- typePr 10.1
- typesettingXEqnPr

typesetting, external file A.2
 Type 4.
 Tysset 4.
 Ultraspherical polynomials Gag 8.9
 Unclassify XClass
 underlining 0.
 unequal Uneq 5.
 Uneq 5.
 unexpected input 1.1
 unflatten XSubl
 UnFlat XUnFlat
 unfolded size Size 10.8
 Union 7.7
 Union XSets
 unique elements Union 7.7
 unitary transformation Simtran 9.6
 units conversion XNKS
 units XDim
 UNIX XEqnPr
 unknowns 2.2
 Unmark XUnmark
 unravel Flat 7.7
 unsimplified expressions 3.5
 unsimplified forms Smp 4.
 until loop Loop 6.2
 Upperp XChar
 user aid 1.2
 user communication Send 10.6
 user programs A.2
 VAbs XVecan
 Valp 7.6
 value assignment 3.2
 value, test for Valp 7.6
 values 3.1
 variable argument number Tier 7.7
 variable evaluation 3.1
 variable, test for Symbp 7.6
 variables 2.2
 variables, list of Cont 7.5
 Var XStat
 vbar XPrtable
 vblank XPrtable
 VCon XG
 Vec2p XRot2
 Vec3p XRot3
 Vecp XVecan
 vector coupling coefficient Wig 8.6
 vector, test for Contp 7.6
 vectors 2.4
 vectors XExDot
 verbose Trace 4.
 versine XTrigR
 vline XPrtable
 Voi XVecan
 vthing XPrtable
 Wait 10.9
 watch Step 10.10
 Watch XWatch
 wave function, Coulomb irregular CouG 8.8
 wave function, Coulomb regular CouF 8.8
 Weber function BesY 8.8 WebE 8.8
 WebE 8.8
 Weierstrass function WeIP 8.10
 Weierstrass σ function Weiz 8.10
 WeIP 8.10
 Weis 8.10
 Weiz 8.10
 while loop Loop 6.2
 Whim 8.8
 Whittaker M function Whim 8.8
 Whittaker W function WhiW 8.8
 WhiW 8.8
 Wigner 3-j symbol Wig 8.6
 Wig 8.6
 word processing XEqnPr
 words XChar
 writePr 10.1 Send 10.6
 Wron XWron
 XLChi2 XLChi2
 Xor 5.
 x_Scal XExDot
 Ycut XPlot
 Year XYear
 YGMult XYoung
 zeta function Zeta 8.7
 zeta function, generalized Zeta 8.7
 zeta functions XLatsum
 Zeta 8.7