Appendix: The GL(n)pack Manual

Kevin A. Broughan

January 19, 2006

§A.1 Introduction

This appendix is the manual for a set of functions written to assist the reader to understand and apply the theorems on $\mathbf{GL}(\mathbf{n}, \mathbf{R})$ set out in the main part of the book. The software for the package is provided over the world wide web at

http://www.math.waikato.ac.nz/∼kab

and is in the form of a standard Mathematica add-on package. To use the functions in the package you will need to have a version of Mathematica at level 4.0 or higher.

§A.1.1 Installation

First connect to the web site given in the paragraph above and click on the link for GL(n)pack listed under "Research" to get to the GL (n)pack home page. Instructions on downloading the files for the package will be given on the home page. If you have an earlier version of GL(n)pack first delete the file **gln.m**, the documentation **gln.pdf** and the validation program **glnval.nb**. The name of the file containing the package is **gln.m**. To install, if you have access to the file system for programs on your computer, place a copy of the file in the standard repository for Mathematica packages - this directory is called "Applications" on some systems. You can then type ≪**gln.m** and then press the Shift and Enter keys to load the package. You may need administrator or super-user status to complete this installation. Alternatively place the package file **gln.m** anywhere in your own file system where it is safe and accessible.

Instructions for Windows systems: The package can be loaded by typing

```
SetDirectory["c:\your\directory\path"];<<gln.m;ResetDirectory[]<Shift/Enter>
```

or

```
Get["gln.m",Path->{"c:\your\directory\path"}]<Shift/Enter>
```

where the path-name in quotes should be replaced by the actual path-name of directories and subdirectories which specify where the package has been placed on a given computer.

Instructions for Unix/Linux systems: These are the same as for Windows, but the path-name syntax style should be like **/usr/home/your/subdirectory**.

Instructions for Macintosh systems: These are the same as for Windows, but the path-name syntax style should be like **HD:Users:ham:Documents:**.

All systems: The package should load printing a message. The functions of GL(n)pack are then available to any Mathematica notebook you subsequently open.

§A.1.2 About this manual

This appendix contains a list of all of the functions available in the package GL(n)pack followed by a manual entry for each function in alphabetical order. Many functions contain the transcript of an example and reference to the part of the text to which the function relates, as well as lists of related functions. Each function has both a Mathematica style long name and a 3 letter/digit abbreviated name. Either can be used, but the error messages and usage information are all in terms of the long names. To contain information about bug fixes and updates to GL(n)pack consult the CUP web site for the text:**http://www.cup.co.uk/goldfeld**.

§A.1.3 Assistance for users new to computers or Mathematica

On the GL(n)pack web site (see above) there are links giving tutorial and other information for those people who want access to the package but are new or relatively new to computers. The manual entries assume familiarity with Mathematica, so some may require extra help. Alternatively sit down with someone familiar with Mathematica to see it at work.
There are many issues to do with computer algebra and mathematical software that will arise in any serious evaluation or use of Mathematica and GL(n)pack . Suffice to comment on one aspect: GL(n)pack function arguments are first evaluated and then checked for correct data type. If the user calls a function with an incorrect number of arguments or an argument of incorrect type, rather than issue a warning and proceeding to compute (default Mathematica style), GL(n)pack prints an error message, aborts the evaluation and returns the user to the top-level, no matter how deeply nested the function which makes the erroneous call happens to be placed. This is a tool for assisting users to debug programs which include calls to this package.

§A.1.4 Mathematica functions

Each GL(n)pack function is a standard Mathematica function and so will work harmoniously with built-in Mathematica functions and user functions. Useful standard functions include those for defining

functions (Module and Block for example) list and matrix manipulation and operations(“.” represents matrix multiplication), special functions (such as the KBessel), plotting functions and the linear algebra add-on package. Note however that a formatted matrix, returned by MatrixForm, is not recognized by Mathematica as a matrix. A matrix in Mathematica is just a list of lists of equal length.

§A.1.5 The data type *CRE*: Canonical Rational Expressions

Many GL(n)pack functions take symbolic arguments which are either explicit integers or real or complex numbers (exact or floating point) or mathematical expressions which could evaluate to numbers. These expressions are expected to be in the class sometimes called “Canonical Rational expressions” or *CRE*s. This class of expression is defined as follows: members are rational functions with numerical coefficients and with symbolic variables, any number of which may be replaced by function calls, or functions which are not evaluated (“noun forms”) or functions of any finite number of arguments each of which can be, recursively, a *CRE*. Some package functions will accept lists of *CRE*s or matrices with *CRE* elements. This should cover most user needs, but notice it excludes simple types like matrices with elements which are matrices. (The single exception is the GL(n)pack function MakeBlockMatrix, which takes as argument a matrix with matrix elements.) If a user is unsure regarding the data type of a mathematical expression, the GL(n) pack function **CreQ** can be used. See the manual entry.

§A.1.6 The algorithms in this package

The reader who uses this package may notice that many functions appear to run almost instantaneously, for example MakeBlockMatrix or HeckeMultiplicativeSplit. Others however take considerable time to complete, minutes or hours rather than seconds. This is often because the underlying algorithm employed is exponential, or in at least one instance, more than exponential. Improvements in this completion time are of course possible: The GL(n)pack code is interpreted, so there may be speed-ups attainable using the Mathematica function **Compile**, even though it has a restricted domain of application. Existing algorithms could be replaced by faster algorithms. The existing algorithms could be re-implemented in a compile-load-and-go language such as C++ or Fortran, or an interactive language allowing for compilation such as Common Lisp. This latter would be the best choice, because execution speed for compiled code is quite comparable to that of the two former choices, but its range of data types is vast, certainly sufficient for all of the package. Functions which will slow significantly as the dimension increases include GetCasimirOperator, ApplyCasimirOperator, KloostermanSum, MPSymmetricPowerLFun, and SpecialWeylGroup.

§A.1.7 Acknowledgements

I had assistance from Columbia University and the University of Waikato and a number of very helpful individuals while writing the package. These included Ross Barnett, Mike Eastwood, Sol Friedberg, Jeff Mozzochi and Eric Stade in addition to Dorian Goldfeld, who's detailed text and explicit approach made it possible. Their contribution is gratefully acknowledged.

Kevin Broughan
Hamilton, New Zealand
July 2005

§A.2 Functions for GL(n)pack

ApplyCasimirOperator[m,expr,iwa] (aco): The operator acts on an expression.

BruhatCVector[a] (bcv): The minors $(c_1, .., c_{n-1})$.

BruhatForm[a] (bru): The four Bruhat factors of a non-singular symbolic matrix.

BlockMatrix[a,rows,cols] (blm): Extract a general sub-block of a matrix.

CartanForm[a] (car): The two Cartan factors of a numeric non-singular matrix.

ConstantMatrix[c,m,n] (com): Construct a constant matrix of given size.

CreQ[e] (crq): Check a Canonical Rational Expression.

DiagonalToMatrix[d] (d2m): Convert a list to a diagonal matrix.

EisensteinFourierCoefficient[z,s,n] (efc): The GL(2) Fourier series n^{th} term.

EisensteinSeriesTerm[z,s,ab] (est): The n^{th}term of the Eisenstein series for GL(2).

ElementaryMatrix[n,i,j,c] (elm): Construct a specified elementary matrix.

FunctionalEquation[vs, i] (feq): Generate the affine parameter maps.

GetCasimirOperator[m,n,"x","y","f"] (gco): The operators for dimension n.

GetMatrixElement[a, i, j] (gme): Return a specified element.

GlnVersion[] (glv): Print the date of the current version.

HeckeEigenvalues[m,n,"a"] (hev): The values of (λ_m) for $GL(n)$.

HeckeMultiplicativeSplit[m](hms): Prepare to evaluate a Hecke Fourier coefficient.

HeckeCoefficientSum[m, ms, "x"](hcs): The right hand side of the Hecke sum.

HeckeOperator[n, z,"f"] (hop): of n^{th}order for square integrable forms on $\mathfrak{h}^n$.

HeckePowerSum[e, es,"B"](hps): Exponents for the Hecke sum at any prime.

HermiteFormLower[a] (hfl): The lower left Hermite form.

HermiteFormUpper[a] (hfu): The upper left Hermite form.

IFun[ν,z] (ifn): The power function $I_\nu(z)$.

InsertMatrixElement[e,i,j,a] (ime): Insert an expression in a given matrix.

IwasawaForm([a] (iwf): The product of the Iwasawa factors of a matrix.

IwasawaXMatrix[w] (ixm): Get the x matrix from the Iwasawa form.

IwasawaXVariables[w] (ixv): Get the x variables from the Iwasawa form.

IwasawaYMatrix[z] (iym): Get the y matrix from the Iwaswa form.

IwasawaYVariables[z] (iyv): Get the y variables from the Iwasawa form.

IwasawaQ[z] (iwq): Test to see if a matrix is in Iwasawa form.

KloostermanBruhatCell[a,x,c,w,y] (kbc): Solve $a = x.c.w.y$ for x and y.

KloostermanCompatibility[t1,t2,c,w] (klc): Relations for a valid sum.

KloostermanSum[t1,t2,c,w] (kls): Compute an explicit Kloosterman sum.

LanglandsForm[p,d] (llf): The three matrices of the Langlands decomposition.

LanglandsIFun(g,d,s) [lif]: Summand for the Langlands Eisenstein series.

LeadingMatrixBlock[a,i,j] (lmb): Extract a leading sub-block of a matrix.

LongElement[n] (lel): Construct the matrix called the long element.

LowerTriangleToMatrix[l] (ltm): Construct a lower triangular matrix.

MakeBlockMatrix[mlist] (mbm): Construct a matrix from submatrices.

MakeMatrix[“x”,m,n] (mkm): Make a matrix with indexed elements.

MakeXMatrix[n,“x”] (mxm): Construct a symbolic unimodular matrix.

MakeXVariables[n,"x"] (mxv): Construct a list of Iwasawa x variables.

MakeYMatrix[n,"y"] (mym): Construct a symbolic diagonal matrix.

MakeYVariables[n,"y"] (myv): Construct Iwasawa y variables.

MakeZMatrix[n,"x","y"] (mzm): Construct a symbolic Iwasawa z matrix.

MakeZVariables[n,"x","y"] (mzv): A list of the x and y variables.

MatrixColumn[m,j] (mcl): Extract a column of a given matrix.

MatrixDiagonal[a] (mdl): Extract the diagonal of a matrix.

MatrixJoinHorizontal[a,b] (mjh): Form a matrix by joining two matrices horizontally.

MatrixJoinVertical[a,b] (mjv): Form a matrix by joining two matrices vertically.

MatrixUpperTriangle[a] (mut): Extract the upper triangular elements.

MatrixLowerTriangle[a] (mlt): Extract the lower triangular elements.

MatrixRow[m,i] (mro): Extract a row of a matrix.

ModularGenerators[n] (mog): Construct the generators for $SL(n,\mathbb{Z})$.

MPEisensteinLambdas[v] (eil): The $\lambda_i(v)$ shifts.

MPEisensteinSeries[s,v] (eis): Minimal parabolic Eisenstein series.

MPEisensteinGamma[s,v] (eig): Gamma factors for minimal parabolic series.

MPExteriorPowerGamma[s,v,k] (epg): Exterior power gamma factors.

MPExteriorPowerLFun[s,v,k] (epl): Minimal parabolic exterior power L-function.

MPSymmetricPowerLFun[s,v,k] (spf): Minimal parabolic symmetric power L-function.

MPSymmetricPowerGamma[s,v,k] (spg): Symmetric powers gamma factors.

NRows[a] (nro): The row dimension of a matrix.

NColumns[a] (ncl): The column dimension of a matrix.

ParabolicQ[p,d] (paq): Test a matrix for membership in a given subgroup.

PluckerCoordinates[a] (plc): Compute the bottom row based minors.

PluckerInverse[Ms] (pli): Compute an element in $SL(n, \mathbb{Z})$ with given minors.

PluckerRelations[n,v] (plr): Compute quadratic relations between minors.

RamanujanSum[n,c] (rsm): Evaluate the Ramanujan sum $s(n, c)$.

RemoveMatrixRow[a,i] (rmr): Remove a row of a matrix non-destructively.

RemoveMatrixColumn[a,j] (rmc): Remove a matrix column non-destructively.

SchurPolynomial[k,x] (spl): The Schur multinomial $S_k(x_1, \cdots, x_n)$.

SmithForm[a] (smf): Compute the Smith form of an integer matrix.

SmithElementaryDivisors[a] (sed): Smith form elementary divisors.

SmithInvariantFactors[a] (sif): Smith form invariant factors.

SpecialWeylGroup[n] (swg): Weyl integer rotation group with det 1 elements.

SubscriptedForm[e] (suf) : Print arrays with integer arguments as subscripts.

SwapMatrixRows[a,i,j] (smr): Return a new matrix with rows swapped.

SwapMatrixColumns[a,i,j] (smc): Return a new matrix with columns swapped.

TailingMatrixBlock[a,i,j] (tmb): Extract a tailing matrix block.

UpperTriangleToMatrix[u] (utm): Form an upper triangular matrix.

VolumeFormDiagonal[“a”,n] (vfd): The volume form for diagonal matrices.

VolumeFormGln[“g”,n] (vfg): The volume form for $GL(n)$.

VolumeFormHn[“x”,“y”,n] (vfh): Volume form for the upper half plane.

VolumeFormUnimodular[“x”,n) (vfu): The volume form for the unimodular group.

VolumeBall[r,n] (vbl): The volume of a ball in n-dimensions.

VolumeHn[n] (vhn): The volume of the generalized upper half plane.

VolumeSphere[r,n] (vsp): The volume of a sphere in n-dimensions.

Wedge[$\mathbf{f_1}, \cdots, \mathbf{f_n}$] (weg): The wedge product for differential forms and the d operator.

WeylGenerator[n,i,j] (wge): Each matrix generator for the Weyl permutation group.

WeylGroup[n] (wgr): Compute the Weyl group of permutation matrices.

Whittaker[z,v,psi] (wit): Compute the generalized Whittaker function $W_{Jacquet}$ symbolically.

WhittakerGamma[v] (wig): Gamma factors for the Whittaker function $W_{Jacquet}$.

WMatrix[n] (wmx): A variation on the long element matrix with determinant 1.

ZeroMatrix[m,n] (zmx): The zero matrix of given dimensions.

---

§A.3 Function Descriptions and Examples

■ ApplyCasimirOperator (aco)

This function computes the Casimir operator acting on an arbitrary expression, or undefined function, of a matrix argument in Iwasawa form, and optionally other parameters, with respect to the Iwasawa variables, which must be specified by giving the matrix in Iwasawa form.
To simply compute the operator it is easier to use the function **GetCasimirOperator**. Because this function uses symbolic differentiation, only functions or expressions which can be differentiated correctly and without error by Mathematica may be used as valid arguments. Note also that no check is made that the user has entered valid Mathematica variables. Since all arguments are evaluated it is good practice to use the function **Clear** to ensure arguments which should evaluate to themselves do so. See Proposition 2.3.3, Example 3.4 and Proposition 2.3.5.
.

ApplyCasimirOperator[m, expr, iwa] ⟶ value

where **m** is positive integer with value **2** or more being the order of the operator, **expr** is a Mathematica expression, normally in the Iwasawa matrix or variables and other parameters, which can be symbolically differentiated, **iwa** is a numeric or symbolic matrix in Iwasawa form, and **value** is an expression or number being the result of applying the Casimir operator in the Iwasawa variables to the expression.

Example

```
In[1]:= z = MakeZMatrix[3, "x", "y"]

Out[1]= {{y[1] y[2], x[1, 2] y[1], x[1, 3]}, {0, y[1], x[2, 3]}, {0, 0, 1}}

In[2]:= ApplyCasimirOperator[2, Det[z.z], z]

Out[2]= 12 y[1]^4 y[2]^2

In[3]:= L = LongElement[3]

Out[3]= {{0, 0, 1}, {0, 1, 0}, {1, 0, 0}}

In[4]:= Simplify[ApplyCasimirOperator[3, a = IFun[{v1, v2}, L.z], z] / a]

Out[4]= 3 (2 v1^3 + 3 v1^2 v2 + v1 (-2 + 3 v2 - 3 v2^2) - 2 v2 (2 - 3 v2 + v2^2))
```

See also: GetCasimirOperator, MakeZMatrix, IwasawaForm, IwasawaQ.

■ BruhatCVector (bcv)

In the explicit Bruhat decomposition of a nonsingular matrix **a**, the diagonal matrix c has a special form, each element being the ratio of absolute values of minor determinants (c_i) of the original matrix

a with the element in the $(i,i)^{th}$ position being c_{n-i+1}/c_{n-i} for $2 \leq i \leq n-1$ with the $(n,n)^{\text{th}}$ element being $\mathbf{c_1}$ and the $(1,1)^{\text{th}}$, $det(w)det(a)/c_{n-1}$. This function returns those c_i.
See Section 10.3 and Proposition 10.3.6.

BruhatCVector[a] $\longrightarrow$ **c**

a is a non-singular $\mathbf{n} \times \mathbf{n}$ square matrix of CREs,

c is a list of $\mathbf{n-1}$ CREs $\{\mathbf{c_1}, \cdots, \mathbf{c_{n-1}}\}$.

Example:

```
In[17]:= LE3 = LongElement[3];
         BruhatCVector[{{1, 2, 3}, {4, 5, 7}, {7, 8, 9}}, LE3]

Out[18]= {7, 3}

In[19]:= LE4 = LongElement[4]; MatrixForm[LE4]

Out[19]//MatrixForm=
         ⎛0  0  0  1⎞
         ⎜0  0  1  0⎟
         ⎜0  1  0  0⎟
         ⎝1  0  0  0⎠

In[20]:= BruhatCVector[
          {{1, a, b, 3}, {0, 1, b, 1}, {a, 2, 0, b}, {1, 2, 3, 4}}, LE4]

Out[20]= {1, Abs[2 - 2 a], Abs[3 a + 2 b - 2 a b]}
```

See also: BruhatForm, LanglandsForm.

BruhatForm (bru)

This function finds the factors of a non-singular matrix, which may have entries which are polynomial, rational or algebraic expressions, so that the matrix can be expressed as the product of an upper triangular matrix with 1's on the leading diagonal (unipotent), a diagonal matrix, a permutation matrix (with a single 1 in each row and column) and a second unipotent matrix. When an additional constraint, namely, that the transpose of the permutation matrix times the second upper triangular matrix is lower triangular, then the factors are unique. This is the so-called Bruhat decomposition. See Section 10.3 definition 10.3.6. The decomposition is used in Section 10.6 to derive the fourier expansion of a minimal parabolic Eisenstein series.

$$\textbf{BruhatForm[a]} \longrightarrow \textbf{\{u}_1\textbf{,c,w,u}_2\textbf{\}}$$

a is a non-singular square CRE matrix,

$\mathbf{u_1}$ is an upper triangular unipotent matrix,

c is a diagonal matrix,

w is a permutation matrix,

$\mathbf{u_2}$ is an upper triangular unipotent matrix.

Example:

```
In[1]:= B = {{x, y, z}, {u, v, 0}, {0, 2, 1}}

Out[1]= {{x, y, z}, {u, v, 0}, {0, 2, 1}}

In[2]:= b = BruhatForm[B];

In[3]:= Map[MatrixForm, b]
```

$$\text{Out[3]=}\ \left\{\begin{pmatrix} 1 & \frac{x}{u} & \frac{1}{2}\left(-\frac{vx}{u}+y\right) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} \frac{1}{2}\left(\frac{vx}{u}-y\right)+z & 0 & 0 \\ 0 & u & 0 \\ 0 & 0 & 2 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & \frac{v}{u} & 0 \\ 0 & 1 & \frac{1}{2} \\ 0 & 0 & 1 \end{pmatrix}\right\}$$

See also:
CartanForm, HermiteForm, IwasawaForm, SmithForm, LanglandsForm.

BlockMatrix (blm)

This function returns a specified sub-block of a matrix. The entries of the sub-block must be contiguous.
Block matrices are used in a number of places, but most especially in Chapter X on Langlands Eisenstein series.

BlockMatrix[a, rows, columns] ⟶ b

a is a matrix of CREs,

rows is a list of two valid row indices for **a**, being the first and last sub-block rows,

columns is a list of two valid column indices for **a**, being the first and last sub-block columns,

b is the sub-block of **a** with the specified first and last row and column sub-block indices.

Example:

```
In[168]:= A = {{4, 8, u, 1, 2}, {8, 7, 2, 0, 1}, {4, 5, 1, 2, 0}};
          B = {{1, 1, 1}, {1, 4, v}, {2, 2, 2}};

In[170]:= M = Transpose[A].B; MatrixForm[M]

Out[170]//MatrixForm=
        ⎛  20    44    12 + 8 v ⎞
        ⎜  25    46    18 + 7 v ⎟
        ⎜4 + u  10 + u  2 + u + 2 v⎟
        ⎜  5     5        5     ⎟
        ⎝  3     6      2 + v   ⎠

In[171]:= MatrixForm[BlockMatrix[M, {1, 2}, {1, 3}]]

Out[171]//MatrixForm=
        ⎛ 20  44  12 + 8 v ⎞
        ⎝ 25  46  18 + 7 v ⎠
```

See also: LeadingMatrixBlock, TailingMatrixBlock, MakeBlockMatrix, MakeMatrix.

CartanForm (car)

This function gives a form of the Cartan decomposition of a numeric real non-singular square matrix, namely the factorization $a = k.exp(x)$ where k is orthogonal and x symmetric. It follows from this that the transpose of an invertible matrix satisfies an equation

$$^t a = k.a.k$$

for some orthogonal matrix k, where $^t a$ is the transpose of a. The function is restricted to numeric matrices is because eigenvectors and eigenvalues are used.

CartanForm[a] ⟶ {k, s}

a is a non-singular real numeric square matrix,

k is an orthogonal matrix,

s is the matrix exponential of a symmetric matrix.

Example:

```
In[84]:= PrintCartan[g_] := Module[{ans, k, S},
           ans = N[Cartan[g]];
           Print[MatrixForm[First[ans]]];
           Print[MatrixForm[Part[ans, 2]]];
           Return[True]]

In[88]:= g = {{1, 2, 0}, {0, 2, 3}, {0, 0, 1}}

Out[88]= {{1, 2, 0}, {0, 2, 3}, {0, 0, 1}}

In[89]:= PrintCartan[g]
```

$$\begin{pmatrix} -0.494923 & -0.00367894 & -0.868929 \\ 0.112604 & 0.991288 & -0.0683336 \\ 0.86161 & -0.131665 & -0.490196 \end{pmatrix}$$

$$\begin{pmatrix} -0.494923 & -0.764638 & 1.19942 \\ -0.00367894 & 1.97522 & 2.8422 \\ -0.868929 & -1.87453 & -0.695197 \end{pmatrix}$$

See also:
BruhatForm, IwasawaForm, LanglandsForm, HermiteFormLower, HermiteFormUpper, SmithForm.

ConstantMatrix (com)

This function constructs a constant matrix with specified element value.
This function can be used together with other functions to construct matrices.

ConstantMatrix[c, m, n] $\longrightarrow$ **a**

c is a CRE,

m is an integer with $\mathbf{m} \geq \mathbf{1}$,

n is an integer with $\mathbf{n} \geq \mathbf{1}$,

a is an **m** by **n** matrix having each element equal to **c**.

See also: ZeroMatrix, ElementaryMatrix, MatrixJoinHorizontal, MatrixJoinVertical.

CreQ (crq)

This function checks to see if its argument evaluates to a so-called Canonical Rational Expression (CRE), i.e. a number (real or complex, exact or floating point) or rational function in one or many variables with numerical coefficients, where any number of the variables can be replaced by function calls, including calls to undefined (so called noun) functions of one or many arguments with arguments being canonical rational expressions. This is the data type expected by GL(n)pack functions. See the introduction to the Appendix.

CreQ[e] $\longrightarrow$ **P**

e is a Mathematica expression,

P is **True** if **e** is a CRE and **False** otherwise.

Example:

```
In[93]:= CreQ[{x, y}]

Out[93]= False

In[94]:= CreQ["x"]

Out[94]= False

In[95]:= CreQ[2 x + y / (x + 1 + Sin[x + y])]

Out[95]= True
```

See also: ParabolicQ, KloostermanSumQ

DiagonalToMatrix (d2m)

This function takes a list and constructs a matrix with the list elements as the diagonal entries.
Diagonal matrices appear in many places, including in the Iwasawa and Bruhat decompositions.

DiagonalToMatrix[di] $\longrightarrow$ **a**

di is a non-empty list of CREs,

a is a square matrix of size the length of **di**, with zeros in off-diagonal positions, and with the diagonal entries being the elements of **di** and in the same order.

See also: MatrixDiagonal.

EisensteinFourierCoefficient (efc)

This function returns the n^{th} term of the Fourier expansion of an Eisenstein series for GL(2), with an explicit integer specified for n. See Section 3.1, especially Theorem 3.1.8.

EisensteinFourierCoefficient[z, s, n] $\longrightarrow$ **v**

z is a CRE,

s is a CRE,

n is an integer being the index of the $\mathbf{n}^{\text{th}}$ coefficient,

v is a complex number or symbolic expression representing the $\mathbf{n}^{\text{th}}$ Fourier term of the Eisenstein Fourier expansion for **GL(2)** with parameters **z** and **s**.

Example:

```
In[1]:= EisensteinFourierCoefficient[z, s, 4]
```

$$\text{Out[1]}= \frac{2^{2s}\,(-1+2^{3-6s})\,e^{8i\pi \mathrm{Re}[z]}\,\pi^{s}\,\mathrm{BesselK}[-\frac{1}{2}+s,\,8\pi\,\mathrm{Im}[z]]\,\sqrt{\mathrm{Im}[z]}}{(-1+2^{1-2s})\,\mathrm{Gamma}[s]\,\mathrm{Zeta}[2s]}$$

```
In[2]:= EisensteinFourierCoefficient[z, s, 0]
```

$$\text{Out[2]}= \mathrm{Im}[z]^{s}+\frac{\sqrt{\pi}\,\mathrm{Gamma}[-\frac{1}{2}+s]\,\mathrm{Im}[z]^{1-s}\,\mathrm{Zeta}[-1+2s]}{\mathrm{Gamma}[s]\,\mathrm{Zeta}[2s]}$$

See also: EisensteinSeriesTerm, IFun, LanglandsIFun

EisensteinSeriesTerm (est)

This function returns the term of the Eisenstein series $E(z, s)$ for GL(2), namely the summand of:

$$E(z,s) = \frac{1}{2} \sum_{a,b \in \mathbb{Z}, (a,b)=1} \frac{y^s}{|az+b|^{2s}}$$

with explicit values for the integers a, b.
See Definition 3.1.2.

EisensteinSeriesTerm[z, s, ab] ⟶ v

z is a CRE,

s is a CRE,

ab is a list of two integers $\{\mathbf{a}, \mathbf{b}\}$, at least one of which must be non-zero,

v is a complex number or symbolic expression representing the term of the Eisenstein series for **GL(2)** with parameters $\mathbf{z}, \mathbf{s}, \mathbf{a}, \mathbf{b}$.

Example:

```
In[1]:= EisensteinSeriesTerm[z, s, {3, 4}]
```

$$\text{Out[1]=} \quad \frac{1}{2} \text{Abs}[4 + 3\,z]^{-2\,s}\, \text{Im}[z]^{s}$$

```
In[2]:= EisensteinSeriesTerm[z, s, {12, 16}]

Out[2]= 0
```

See also: EisensteinFourierCoefficient, LanglandsIFun.

ElementaryMatrix (elm)

This function returns a square matrix having 1's along the leading diagonal and with a given element in a specified off-diagonal position.

ElementaryMatrix[n, i, j, c] $\longrightarrow$ **e**

n is a strictly positive integer being the size of the matrix,

i is a strictly positive integer being the row index of the off-diagonal entries,

j is a strictly positive integer with $\mathbf{i} \neq \mathbf{j}$, representing the column index of the off-diagonal entries,

c is a CRE to be placed at the $(\mathbf{i}, \mathbf{j})^{\mathbf{th}}$ position,

e is an **n** by **n** matrix with 1's on the leading diagonal and all other elements zero, except in the $(\mathbf{i}, \mathbf{j})^{\text{th}}$ position where it is **c**.

Example:

```
In[182]:= MatrixForm[ElementaryMatrix[4, 3, 1, x]]

Out[182]//MatrixForm=
        ⎛1  0  0  0⎞
        ⎜0  1  0  0⎟
        ⎜x  0  1  0⎟
        ⎝0  0  0  1⎠

In[183]:= {a1, a2, a3, a4}.%

Out[183]= {a1 + a3 x, a2, a3, a4}

In[184]:= %%.{{b1}, {b2}, {b3}, {b4}}

Out[184]= {{b1}, {b2}, {b3 + b1 x}, {b4}}
```

See also: SwapMatrixRows, SwapMatrixColumns.

FunctionalEquation (feq)

This function, for each index **i**, returns a list of affine combinations of its variables representing the i^{th} functional equation for the Jacquet Whittaker function of order $\mathbf{n} \geq \mathbf{2}$.
See Section 5.9, especially equations 5.9.5, 5.9.6 and Example 5.9.7.

FunctionalEquation[v, i] ⟶ vp

v is a list of CREs of length $\mathbf{n-1}$,

i is a strictly positive integer with $\mathbf{1 \leq i \leq n-1}$ being the index of the functional equation,

vp is a list of CREs representing the transformations required of the variables **vp** for the $\mathbf{i^{th}}$ functional equation.

Example:

```
In[96]:= Table[FunctionalEquation[{x1, x2, x3, x4}, i], {i, 1, 4}]
```

$$\text{Out[96]= } \left\{\left\{x1,\ x2,\ -\frac{1}{5}+x3+x4,\ \frac{2}{5}-x4\right\},\ \left\{x1,\ -\frac{1}{5}+x2+x3,\ \frac{2}{5}-x3,\ -\frac{1}{5}+x3+x4\right\},\right.$$
$$\left.\left\{-\frac{1}{5}+x1+x2,\ \frac{2}{5}-x2,\ -\frac{1}{5}+x2+x3,\ x4\right\},\ \left\{\frac{2}{5}-x1,\ -\frac{1}{5}+x1+x2,\ x3,\ x4\right\}\right\}$$

See also: Whittaker, WhittakerGamma, WhittakerStar.

GetCasimirOperator (gco)

This function computes the Casimir operator acting on an arbitrary noun function and with respect to the Iwasawa variables. Note that this function makes an explicit brute-force evaluation of the operator, so is not fast, especially for $n \geq 3$.
See Proposition 3.3.3 and Example 3.3.4.

GetCasimirOperator[m,n,"x","y","f"] $\longrightarrow$ Operator

m is positive integer with value 2 or more being the order of the operator,

n is a positive integer with value 2 or more being the dimension of the Iwasawa form,

"x" is a string, being the name of the symbol such that the variables in the upper triangle of the matrix given by the Iwasawa decomposition are $\mathbf{x}[\mathbf{i},\mathbf{j}]$,

"y" is a string, being the name of the symbol such that the terms in the first $\mathbf{n-1}$ positions of the leading diagonal of the Iwasawa decomposition are

$$\mathbf{y}[\mathbf{1}]\cdots\mathbf{y}[\mathbf{n-1}],\mathbf{y}[\mathbf{1}]\cdots\mathbf{y}[\mathbf{n-2}],\cdots,\mathbf{y}[\mathbf{1}].$$

"f" is a string being the name of a function of noun form (i.e. it should not be defined as an explicit Mathematica function or correspond to the name of an existing function) which will appear as partially differentiated by the computed Casimir operator,

Operator is an expression in the variables

$$(\mathbf{x}[\mathbf{i},\mathbf{j}],\mathbf{1}\leq\mathbf{i}<\mathbf{j}\leq\mathbf{n}),(\mathbf{y}[\mathbf{i}],\mathbf{1}\leq\mathbf{i}\leq\mathbf{n}-\mathbf{1})$$

and the partial derivatives of the function with name **"f"** with respect to argument slots of **f** arranged in the order $(\mathbf{x_{1,1}},\mathbf{x_{1,2}},\ldots,\mathbf{y_1},\ldots,\mathbf{y_{n-1}})$.

Example:

See also: IwasawaForm, ApplyCasimirOperator.

```
In[8]:= suf[GetCasimirOperator[3, 3, "x", "y", "f"]]
```

$$
\begin{aligned}
\text{Out[8]}= \; & 3\, y_1 \left(-y_2\, f^{(0,0,0,1,1)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] + \right. \\
& y_2^2\, f^{(0,0,0,1,2)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] + \\
& 2\, y_1\, f^{(0,0,0,2,0)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] - \\
& y_1\, y_2\, f^{(0,0,0,2,1)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] + \\
& 2\, y_1\, f^{(0,0,2,0,0)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] - \\
& y_1\, y_2\, f^{(0,0,2,0,1)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] + \\
& 4\, y_1\, x_{1,2}\, f^{(0,1,1,0,0)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] - \\
& 2\, y_1\, y_2\, x_{1,2}\, f^{(0,1,1,0,1)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] + \\
& 2\, y_1\, y_2^2\, f^{(0,2,0,0,0)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] + \\
& 2\, y_1\, x_{1,2}^2\, f^{(0,2,0,0,0)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] + \\
& y_1\, y_2^3\, f^{(0,2,0,0,1)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] - \\
& y_1\, y_2\, x_{1,2}^2\, f^{(0,2,0,0,1)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] - \\
& y_1^2\, y_2^2\, f^{(0,2,0,1,0)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] + \\
& 2\, y_1\, y_2^2\, f^{(1,1,1,0,0)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] + \\
& 2\, y_1\, y_2^2\, x_{1,2}\, f^{(1,2,0,0,0)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2] + \\
& \left. y_2^2\, f^{(2,0,0,1,0)}[x_{1,2}, x_{1,3}, x_{2,3}, y_1, y_2]\right)
\end{aligned}
$$

InsertMatrixElement (ime)

An element is inserted into a matrix returning a new matrix and leaving the original unchanged.

InsertMatrixElement[e, i, j, a] $\longrightarrow$ **b**

e is a CRE being the element to be inserted,

i is the row index of the position where the element is to be inserted,

j is the column index of the position where the element is to be inserted,

a is the original matrix of CREs,

b is a new matrix, being equal to **a** but with **e** in the $(\mathbf{i}, \mathbf{j})^{\mathbf{th}}$ position.

See also: DiagonalToMatrix, MatrixJoinHorizontal, MatrixJoinVertical.

GetMatrixElement (gme)

The specified element of a matrix is returned.

GetMatrixElement[a, i, j] $\longrightarrow$ **e**

a is a matrix of CREs,

i is the row index of the element,

j is the column index of the element,

e is the $(\mathbf{i}, \mathbf{j})^{\mathbf{th}}$ element of **a**.

See also: MatrixColumn, MatrixRow, MatrixBlock.

GlnVersion (glv)

This function prints out the date of the version of GL(n)pack which is being used, followed by the version of Mathematica. It has no argument, but the brackets must be given.

GlnVersion[] $\longrightarrow$ **True**

HeckeCoefficientSum (hcs)

This function takes a natural number m, a list of natural numbers $\{m_1, \cdots, m_{n-1}\}$ and a string for a function name and finds the terms in the sum right hand side

$$\lambda_m A(m_1, \cdots, m_{n-1}) = \sum A(c_0 m_1/c_1, c_1 m_2/c_2, \cdots, c_{n-2} m_{n-1}/c_{n-1})$$

where the summation is over all (c_i) such that $\prod_{i=0}^{n-1} c_i = m$ and $c_i | m_i, 1 \leq i \leq n-1$. See Theorem 9.3.11 and equation 9.3.17.

HeckeCoefficientSum[m, ms, "A"] ⟶ s

m is a natural number (i.e. a strictly positive integer) representing the index of the eigenvalue $\lambda_{\mathbf{m}}$,

ms is a list of natural numbers representing the multi-index of the Fourier coefficient **A**,

"A" is a string giving the name of a noun function for the Fourier coefficient,

s is a term or sum of terms being the right hand side of the expansion $\lambda_{\mathbf{m}}\mathbf{A}(\mathbf{m_1}, \cdots)$.

Example:

```
In[99]:= HeckeCoefficientSum[6, {12, 4, 5}, "A"]

Out[99]= A[2, 24, 5] + A[4, 6, 10] + A[8, 12, 5] + A[18, 8, 5] + A[36, 2, 10] + A[72, 4, 5]
```

See also: HeckeOperator, SchurPolynomial, HeckePowerSum, HeckeEigenvalue.

HeckeOperator (hop)

This function computes the n^{th} order Hecke operator which acts on square integrable forms on $\mathfrak{h}^n$.
See Section 9.3, especially formula 9.3.5.

$$\textbf{HeckeOperator[n, z, f]} \longrightarrow \mathbf{T_n(f(z))}$$

n is a natural number being the order of the operator,

z is a square matrix of CREs of size **n**,

f is a string being the name of a function of a square matrix of size **n**,

$\mathbf{T_n(f(z))}$ is an expression representing the action of the $\mathbf{n}^{\text{th}}$ Hecke operator on the matrix function $\mathbf{f(z)}$.

Example:

```
In[21]:= z = {{x, x^2, 2}, {1, x, x + 1}, {0, 2, x}}

Out[21]= {{x, x^2, 2}, {1, x, 1 + x}, {0, 2, x}}

In[22]:= HeckeOperator[2, z, "f"]

Out[22]= f[{{x, x^2, 2}, {1, x, 1 + x}, {0, 4, 2 x}}] +
          f[{{x, x^2, 2}, {1, 2 + x, 1 + 2 x}, {0, 4, 2 x}}] +
          f[{{x, x^2, 2}, {2, 2 x, 2 (1 + x)}, {0, 2, x}}] +
          f[{{x, 2 + x^2, 2 + x}, {1, x, 1 + x}, {0, 4, 2 x}}] +
          f[{{x, 2 + x^2, 2 + x}, {1, 2 + x, 1 + 2 x}, {0, 4, 2 x}}] +
          f[{{2 x, 2 x^2, 4}, {1, x, 1 + x}, {0, 2, x}}] +
          f[{{1 + x, x + x^2, 3 + x}, {2, 2 x, 2 + 2 x}, {0, 2, x}}]

In[26]:= f[z_] := Sum[z[[i, i]], {i, 1, Length[z]}]

In[28]:= HeckeOperator[12, z, "f"]

Out[28]= 2804 + 4688 x
```

See also: HeckeEigenvalue.

HeckePowerSum (hps)

This function takes a natural number m, a list of natural numbers $\{m_1, \cdots, m_{n-1}\}$ and a string for a function name and finds the powers of any fixed prime in the sum right hand side

$$\lambda_m A(m_1, \cdots, m_{n-1}) = \sum A(c_0 m_1/c_1, c_1 m_2/c_2, \cdots, c_{n-2} m_{n-1}/c_{n-1})$$

where the summation is over all (c_i) such that $\prod_{i=0}^{n-1} c_i = m$ and $c_i | m_i, 1 \leq i \leq n-1$ in case m and each of the m_i is a power of a fixed prime. The powers that appear in the expansion are the same for any prime. The purpose of this function is to simplify the study of the multiplicative properties of the Fourier coefficients.
See Theorem 9.3.11 and equation 9.3.17.

HeckePowerSum[a, as, "B"] $\longrightarrow$ list

a is a non negative integer, being the power **a** of any prime **p** such that $\mathbf{p^a}$ is the index of the eigenvalue $\lambda_{\mathbf{p^a}}$,

as is a list of non negative integers representing the powers of a fixed prime which appear in the multi-index of a Fourier coefficient,

list consists of a sum of terms $\mathbf{B}[\mathbf{b_{1,i}}, \cdots, \mathbf{b_{n-1,i}}]$ such that the corresponding term in the Hecke sum would have a value $\mathbf{A}(\mathbf{p}^{\mathbf{b_{1,i}}}, \cdots)$.

Example:

```
In[100]:=
        HeckePowerSum[2, {3, 4, 5}, "B"]

Out[100]=
        B[1, 6, 5] + B[2, 4, 6] + B[2, 5, 4] + B[3, 2, 7] +
         B[3, 3, 5] + B[3, 4, 3] + B[3, 5, 5] + B[4, 3, 6] + B[4, 4, 4] + B[5, 4, 5]
```

See also: HeckeOperator, SchurPolynomial, HeckeEigenvalue, HeckeCoefficientSplit,

HeckeEigenvalue (hev)

This function returns the value of the m^{th} eigenvalue of the ring of HeckeOperators acting on square integrable automorphic forms $f(z)$ for $\mathfrak{h}^n$. Note that when the Euler product of a Maass form is known, the Fourier coefficients which appear in the expressions for the eigenvalues (the A in $\lambda_m = A(m, 1, \cdots, 1)$) can be expressed in terms of Schur polynomials in the parameters which appear in the Euler product.
See the text Section 9.3, especially Theorem 9.3.11.

HeckeEigenvalue[m, n, a] $\longrightarrow \lambda_{\mathbf{m}}$

m is a natural number representing the index of the eigenvalue $\lambda_{\mathbf{m}}$,

n is a positive integer of size two or more being the dimension of **GL(n)**,

a is a string representing the name of a function of $\mathbf{n-1}$ integers being the Fourier coefficients of a given Maass form which is an eigenfunction of all of the Hecke operators.

$\lambda_{\mathbf{m}}$ is an expression representing the $\mathbf{m^{th}}$ Hecke eigenvalue.

Example:

```
In[39]:= HeckeEigenvalue[2*3^4*5^2, 6, "a"]

Out[39]= a[2, 1, 1, 1, 1]
          (-a[1, 1, 1, 3, 1] + a[1, 3, 1, 1, 1]^2 + 2 a[1, 1, 3, 1, 1] a[3, 1, 1, 1, 1] -
            3 a[1, 3, 1, 1, 1] a[3, 1, 1, 1, 1]^2 + a[3, 1, 1, 1, 1]^4)
          (-a[1, 5, 1, 1, 1] + a[5, 1, 1, 1, 1]^2)
```

See also: HeckeOperator, HeckeMultiplicativeSplit, SchurPolynomial, HeckePowerSum.

HeckeMultiplicativeSplit (hms)

This function takes a list of natural numbers $\{m_1, \cdots, m_{n-1}\}$, finds the primes and their powers that divide any of the m_i, and returns of list of lists of those primes and their powers. The purpose of this function is the evaluation of the Hecke Fourier coeffients of a Maass form in terms of Schur polynomials when the Euler product coeffients of the form are known. If $p_1, \cdots, p_r$ are the primes and $k_{i,j}$ is the maximum power of p_i dividing m_j, then the Fourier coefficient

$$A(m_1, \cdots, m_{n-1}) = \prod_{i=1}^{r} A(p_i^{k_{i,1}}, \cdots, p_i^{k_{i,n-1}}).$$

See Theorem 9.3.11 and equation 7.4.14.

HeckeMultiplicativeSplit[m] $\longrightarrow$ list

m is a list of natural numbers representing the multi-index of the Fourier coefficient,

list consists of sublists, each being a prime $\mathbf{p_i}$ and a list of $\mathbf{n-1}$ powers of that prime $\mathbf{k_{i,j}}$.

Example:

```
In[98]:= HeckeMultiplicativeSplit[{2*3^2*5^4, 2^3*3*5, 5}]

Out[98]= {{2, {1, 3, 0}}, {3, {2, 1, 0}}, {5, {4, 1, 1}}}
```

See also:
HeckeOperator, SchurPolynomial, HeckeEigenvalue, HeckeCoefficientSum.

HermiteFormLower (hfl)

This function computes the lower left Hermite form h of a non-singular integer matrix a, and a unimodular matrix l such that $a = l.h$. This Hermite form is a lower triangular integer matrix with strictly positive elements on the diagonal of increasing size, and such that each element in the column below a diagonal entry is non-negative and less than the diagonal entry.
See Theorem 3.11.1.

HermiteFormLower[a] ⟶ {l, h}

a is a non-singular integer matrix,

l is a unimodular matrix,

h is a lower triangular integer matrix, being the Hermite form of **a**.

Example:

```
In[4]:= m4 = {{5, 2, -4, 7}, {1, 6, 0, -3}, {1, 2, -2, 4}, {7, 1, 5, 6}};

In[8]:= {l, h} = HermiteFormLower[m4]

Out[8]= {{{-5, -2, -4, 7}, {1, 3, 0, -3}, {-3, -1, -2, 4}, {-4, -4, 5, 6}},
         {{241, 0, 0, 0}, {158, 4, 0, 0}, {35, 1, 1, 0}, {238, 2, 0, 1}}}

In[9]:= Map[MatrixForm, %]

Out[9]=
```

$$\left\{\begin{pmatrix} -5 & -2 & -4 & 7 \\ 1 & 3 & 0 & -3 \\ -3 & -1 & -2 & 4 \\ -4 & -4 & 5 & 6 \end{pmatrix}, \begin{pmatrix} 241 & 0 & 0 & 0 \\ 158 & 4 & 0 & 0 \\ 35 & 1 & 1 & 0 \\ 238 & 2 & 0 & 1 \end{pmatrix}\right\}$$

```
In[10]:= l.h - m4

Out[10]= {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}}
```

See also: HermiteFormUpper, SmithForm

HermiteFormUpper (hfu)

This function computes the upper Hermite form h of a non-singular integer matrix a, and a unimodular matrix l such that $a = l.h$. This Hermite form is an upper triangular integer matrix with strictly positive elements on the diagonal of increasing size, and such that each element in the column above a diagonal entry is non-negative and less than the diagonal entry.
See Theorem 3.11.1.

HermiteFormUpper[a] ⟶ {l, h}

a is a non-singular integer matrix,

l is a unimodular matrix,

h is an upper triangular integer matrix, being the Hermite form of **a**.

Example:

```
In[107]:=
m4 = {{5, 2, -4, 7}, {1, 6, 0, -3}, {1, 2, -2, 4}, {7, 1, 5, 6}};

In[108]:=
MatrixForm[m4]

Out[108]//MatrixForm=
⎛5  2  -4  7 ⎞
⎜1  6  0   -3⎟
⎜1  2  -2  4 ⎟
⎝7  1  5   6 ⎠

In[109]:=
{l, h} = HermiteFormUpper[m4];

In[110]:=
Map[MatrixForm, %]

Out[110]=
⎧⎛5  2  -3  -2⎞  ⎛1  0  0  144⎞⎫
⎨⎜1  6  -3  -3⎟, ⎜0  1  1  262⎟⎬
⎩⎜1  2  -2  -1⎟  ⎜0  0  2   91⎟⎭
 ⎝7  1   2  -3⎠  ⎝0  0  0  482⎠

In[111]:=
l.h - m4

Out[111]=
{{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}}
```

See also: HermiteFormLower, SmithForm

IFun (ifn)

This function returns the value

$$I_\nu(z) = \prod_{i=1}^{n-1} \prod_{j=1}^{n-1} y_i^{b_{i,j}\nu_j}$$

where $b_{i,j} = ij$ if $i + j \leq n$ and $(n - i)(n - j)$ if $i + j > n$. The $n \times n$ matrix z is real and non-singular, or has CRE elements which could evaluate to a real non-singular matrix. The variables y_i are those in the Iwasawa decomposition of z.
See Section 2.4, 5.1.1.

$$\mathbf{IFun}[\nu,\mathbf{z}] \longrightarrow \mathbf{v}$$

ν is a list of $\mathbf{n-1}$ CREs,

$\mathbf{z}$ is an $\mathbf{n} \times \mathbf{n}$ non-singular matrix of CREs,

$\mathbf{v}$ is the product $\mathbf{I}_\nu(\mathbf{z})$.

Example:

```
In[23]:= m = {{1, x[1, 2], x[1, 3]}, {0, 1, x[2, 3]}, {0, 0, 1}}.
            {{y[1] y[2], 0, 0}, {0, y[1], 0}, {0, 0, 1}};

In[22]:= MatrixForm[m]

Out[22]//MatrixForm=
        ⎛ y[1] y[2]  x[1, 2] y[1]  x[1, 3] ⎞
        ⎜     0          y[1]      x[2, 3] ⎟
        ⎝     0           0           1    ⎠

In[16]:= IFun[{v1, v2}, m]

Out[16]= y[1]^(v1+2 v2) y[2]^(2 v1+v2)
```

See also: IwasawaForm, IwasawaYVariables.

IwasawaForm (iwf)

This function computes the Iwasawa form of a non-singular real matrix a. This consists of the product of an upper triangular unipotent matrix x and a diagonal matrix y with strictly positive diagonal entries such that, for some non-singular integer matrix u, real orthogonal matrix o and constant diagonal matrix δ, $a = u.x.y.o.\delta$. This function returns a single matrix $z = x.y$.
See Section 1.2.

IwasawaForm[a] ⟶ z

a is a non-singular square matrix of CREs,

z is an upper-triangular matrix with positive diagonal entries, being the Iwasawa form of a.

Example:

The Iwasawa decomposition of the matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

is found.

```
In[8]:= g = {{a, b}, {c, d}};

In[9]:= IwasawaForm[g]
```

$$\text{Out[9]= } \left\{\left\{\frac{b\,c - a\,d}{c^2 + d^2},\ \frac{a\,c + b\,d}{c^2 + d^2}\right\}, \{0, 1\}\right\}$$

```
In[12]:= g = {{y, x, x^2}, {2 y, 0, x}, {0, x, 1}};

In[13]:= MatrixForm[IwasawaForm[g]]
```

Out[13]//MatrixForm=

$$\begin{pmatrix} \frac{x\,(2+x-2\,x^2)\,y}{\sqrt{1+x^2}\,\sqrt{x^4+4\,y^2+4\,x^2\,y^2}} & \frac{\sqrt{\frac{x^4}{1+x^2}+4\,y^2}\ (-x^3+x^5+2\,y^2+2\,x^2\,y^2)}{\sqrt{1+x^2}\ (x^4+4\,y^2+4\,x^2\,y^2)} & \frac{2\,x^2}{1+x^2} \\ 0 & \frac{\sqrt{\frac{x^4}{1+x^2}+4\,y^2}}{\sqrt{1+x^2}} & \frac{x}{1+x^2} \\ 0 & 0 & 1 \end{pmatrix}$$

See also:
IwasawaQ, MakeZMatrix, IwasawaXMatrix, IwasawaYMatrix, IwasawaXVariables, IwasawaYVariables.

IwasawaXMatrix (ixm)

This function returns the unipotent matrix x corresponding to the decomposition $z = x.y$ of a matrix z in Iwasawa form.
See Proposition 1.2.6.

IwasawaXMatrix[w] $\longrightarrow$ **x**

w is a square non-singular matrix of CREs which must be in Iwasawa form,

x an upper-triangular matrix with 1's on the diagonal and values $\mathbf{x_{i,j}}$ in each $\mathbf{i,j}$ position above the diagonal.

Example:

In this example the x-matrix, x-variables, y-matrix and y-variables are extracted from a generic matrix in Iwasawa form.

```
In[23]:= m = {{1, x[1, 2], x[1, 3]}, {0, 1, x[2, 3]}, {0, 0, 1}}.
            {{y[1] y[2], 0, 0}, {0, y[1], 0}, {0, 0, 1}};

In[22]:= MatrixForm[m]

Out[22]//MatrixForm=
        ⎛ y[1] y[2]  x[1, 2] y[1]  x[1, 3] ⎞
        ⎜    0          y[1]      x[2, 3] ⎟
        ⎝    0           0           1    ⎠

In[17]:= MatrixForm[IwasawaXMatrix[m]]

Out[17]//MatrixForm=
        ⎛ 1  x[1, 2]  x[1, 3] ⎞
        ⎜ 0     1     x[2, 3] ⎟
        ⎝ 0     0        1    ⎠

In[18]:= IwasawaXVariables[m]

Out[18]= {x[1, 2], x[1, 3], x[2, 3]}

In[19]:= IwasawaYMatrix[m] // MatrixForm

Out[19]//MatrixForm=
        ⎛ y[1] y[2]   0    0 ⎞
        ⎜    0      y[1]  0 ⎟
        ⎝    0       0    1 ⎠

In[20]:= IwasawaYVariables[m]

Out[20]= {y[1], y[2]}
```

See also: IwasawaForm, IwasawaXMatrix, IwasawaYVariables, IwasawaYMatrix.

IwasawaXVariables (ixv)

This function returns the x-variables from a matrix $z = x.y$ in Iwasawa form. These are the elements in the strict upper triangle of the matrix x in row order.
See Proposition 1.2.6.

$$\textbf{IwasawaXVariables[w]} \longrightarrow \textbf{l}$$

w is a square non-singular matrix of CREs which must be in Iwasawa form,

l is a list of the form $\{\mathbf{x_{1,2}}, \cdots, \mathbf{x_{1,n}}, \mathbf{x_{2,3}}, \cdots, \mathbf{x_{n-1,n}}\}$.

See also: IwasawaForm, IwasawaXMatrix, IwasawaYVariables, IwasawaYMatrix.

IwasawaYMatrix (iym)

This function returns the y-matrix from the decomposition $z = x.y$ of a matrix z in Iwasawa form.
See Proposition 1.2.6.

$$\textbf{IwasawaYMatrix[z]} \longrightarrow \textbf{y}$$

z is a square non-singular matrix of CREs which must be in Iwasawa form,

y a diagonal matrix where the $\mathbf{i^{th}}$ diagonal slot has the value $\mathbf{y_1 \cdots y_{n-i}}$ for $\mathbf{1 \leq i \leq n-1}$ where the $(\mathbf{n},\mathbf{n})^{\mathbf{th}}$ position has the value 1.

See also:
IwasawaForm, IwasawaXMatrix, IwasawaYVariables, IwasawaXVariables.

IwasawaYVariables (iyv)

This function returns a list of the y-variables from the Iwasawa decomposition of a matrix $z = x.y$.
See Proposition 1.2.6.

$$\textbf{IwasawaYvariables[z]} \longrightarrow \textbf{L}$$

z is a square non-singular matrix of CREs which must be in Iwasawa form,

L a list $\{\mathbf{y_1}, \cdots, \mathbf{y_{n-1}}\}$ of the y-variables of the Iwasawa form.

See also:
IwasawaForm,IwasawaYMatrix, IwasawaXVariables,IwasawaXMatrix.

IwasawaQ (iwq)

This function tests a Mathematica form or expression to see whether it is a non-singular square matrix in Iwasawa form.
See Section 1.2.

$$\textbf{IwasawaQ[z]} \longrightarrow \textbf{value}$$

z is a Mathematica form,

value is **True** if **z** is a matrix of CREs in Iwasawa form, **False** otherwise.

See also: IwasawaForm, MakeZMatrix.

KloostermanBruhatCell (kbc)

This function takes an explicit permutation matrix w with all other arguments being symbolic. It returns rules which solve for x and y in the square matrix Bruhat decomposition equation $a = x.c.w.y$ assuming c is in "Friedberg form", x and y are unipotent and y satisfies ${}^t w.{}^t y.w$ is upper triangular. These rules are not unique. See Chapter XI, especially Section 11.2. Also 10.6.3.

KloostermanBruhatCell[a,x,c,w,y] $\longrightarrow$ rules

a is a symbol which will be used as the name of an $\mathbf{n} \times \mathbf{n}$ matrix.

x is a symbol which will be used as the name of a unipotent matrix.

c is a symbol which will be used as the name of an array **c**[**i**] representng a list of $\mathbf{n-1}$ non-zero integers specifying the diagonal of a matrix. (Note that the 1st element of the diagonal represents the term $\mathbf{det(w)/c[n-1]}$, the second $\mathbf{c[n-1]/c[n-2]}$ and so on down to the last **c**[**1**] as in the notation of 11.2.1.)

w is an $\mathbf{n} \times \mathbf{n}$ matrix which is zero except for a single 1 in each row and column, being an explicit element of the Weyl Group $\mathbf{W_n}$.

y is a symbol which will be used as the name of a unipotent matrix which satisfies ${}^{\mathbf{t}}\mathbf{w}.{}^{\mathbf{t}}\mathbf{y}.\mathbf{w}$ is upper triangular making the decomposition unique, given **a**.

rules is a list of rules of the form $\mathbf{x[i,j]} \rightarrow \mathbf{eij}$ or $\mathbf{y[i,j]} \rightarrow \mathbf{eij}$ where the **eij** are expressions in the **a**[**i**, **j**] and **c**[**i**].

Example:

```
In[29]:= w = WeylGroup[4][[19]]

Out[29]= {{0, 0, 0, 1}, {1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}}

In[30]:= suf[KloostermanBruhatCell[a, x, c, w, y]]
```

$$\text{Out[30]=}\ \left\{x_{1,2} \to \frac{c_2\, a_{1,1}}{c_3},\ x_{1,3} \to \frac{c_1\, a_{1,2}}{c_2},\ x_{1,4} \to \frac{a_{1,3}}{c_1},\right.$$
$$x_{2,3} \to \frac{c_1\, a_{2,2}}{c_2},\ x_{2,4} \to \frac{a_{2,3}}{c_1},\ x_{3,4} \to \frac{a_{3,3}}{c_1},\ y_{1,2} \to 0,\ y_{1,3} \to 0,$$
$$y_{1,4} \to \frac{c_2\,(c_1\, a_{2,4} - a_{2,3}\, a_{4,4}) + c_1\, a_{2,2}\,(-c_1\, a_{3,4} + a_{3,3}\, a_{4,4})}{c_1\, c_3},$$
$$\left. y_{2,3} \to 0,\ y_{2,4} \to \frac{c_1\, a_{3,4} - a_{3,3}\, a_{4,4}}{c_2},\ y_{3,4} \to \frac{a_{4,4}}{c_1}\right\}$$

See also: BruhatForm, BruhatCVector, KloostermanCompatibility, KloostermanSum.

KloostermanCompatibility (klc)

This function takes an explicit permutation matrix w, with remaining arguments symbolic, and returns a list of values, each element being a different type of constraint applicable to any valid Kloosterman sum based on w. The first element is a list of forms restricting the characters. The second is a set of divisibility relations restricting the values of the diagonal matrix c. And the third is the set of minor relations. A typical approach to forming Kloosterman sums would be to first run this function, determine a valid set or sets of parameters from the symbolic output, and then run **KloostermanSum** using explicit integer values of valid parameters. See Chapter XI, Proposition 11.2.10, Lemma 10.6.3.

KloostermanCompatibility[t1,t2, c, w, v] $\longrightarrow$ {characters, divisibilities, minors}

t1 is a symbol representing the character $\mathbf{e^{2\pi i \sum_{i=1}^{n-1} t1[i,i+1]}}$ of $\mathbf{U_n}(\mathbb{R})$.

t2 is a symbol representing another character of $\mathbf{U_n}(\mathbb{R})$.

c is a symbol representing the diagonal of a matrix in Friedberg notation. (Note that the 1st element of the diagonal is the term $\mathbf{det(w)/c_{n-1}}$, the second $\mathbf{c_{n-1}/c_{n-2}}$ and so on down to the last $\mathbf{c_1}$ as in the notation of 11.2.1.)

w is an $\mathbf{n \times n}$ matrix which is zero except for a single 1 in each row and column, representing an explicit element of the Weyl Group $\mathbf{W_n}$.

v is a symbol representing the generic name of any bottom row-based minor.

characters is a list of expressions relating the elements of **t1**, **t2** and the $\mathbf{c_i}$. Each expression must vanish if an explicit Kloosterman sum is to be valid.

divisibility is a list of lists each sublist being of the form $\{\mathbf{c_i}, \mathbf{1}\}$ or $\{\mathbf{c_i}, \mathbf{c_j}\}$. The former means valid sums must have the $\mathbf{c_i} = \pm\mathbf{1}$. The latter means they must have $\mathbf{c_i}|\mathbf{c_j}$.

minors is a list of rules of the form $\mathbf{v}[\{\mathbf{j_1}, \mathbf{j_2}, \cdots, \mathbf{j_i}\}] \rightarrow \mathbf{c_i}$ or $\mathbf{0}$, giving the constraints on minors.

Example:

See also: KloostermanBruhatCell, BruhatForm, BruhatCVector, KloostermanSum, PluckerRelations, PluckerCoordinates, PluckerInverse.

```
In[1861]:=
        MatrixForm[w = WeylGroup[4][[17]]]

Out[1861]//MatrixForm=
        ⎛0  0  1  0⎞
        ⎜0  0  0  1⎟
        ⎜1  0  0  0⎟
        ⎝0  1  0  0⎠

In[1882]:=
        SubscriptedForm[klc[t1, t2, c, w, v]]

Out[1882]=
        {{c2 t13/c12 - t21, c2 t11/c32 - t23}, {{c1, c2}, {c3, c2}},
          {v[{2}] → c1, v[{1}] → 0, v[{1, 2}] → c2, v[{1, 2, 4}] → c3, v[{1, 2, 3}] → 0}}
```

KloostermanSum (kls)

This function computes the generalized Kloosterman sum for $\mathbf{SL}(\mathbf{n},\mathbb{Z})$ for $\mathbf{n} \geq \mathbf{2}$ as given by Definition 11.2.2. When $\mathbf{n} = \mathbf{2}$ this coincides with the classical Kloosterman sum. More generally the sum is

$$S(\theta_1, \theta_2, c, w) := \sum_{\gamma = b_1 c w b_2} \theta_1(b_1)\theta_2(b_2)$$

where

$$\gamma \in U_n(\mathbb{Z})\backslash SL(n,\mathbb{Z}) \cap G_w/\Gamma_w$$

and $\Gamma_w = {}^t w.{}^t U_n(\mathbb{Z}).w \cap U_n(\mathbb{Z})$ and G_w is the Bruhat cell associated to the permutation matrix w. Since these sums are only well defined for some particular compatible values of the arguments the user is advised to first run **KloostermanCompatibility** with an explicit **w** to determine those values. Note that the complexity of the algorithm is $O(\prod_{1 \leq i \leq n-1} |c_i|^n) = O(c^{n^2})$ where $c = \max|c_i|$. See Chapter XI.

KloostermanSum[t1, t2, c, w] ⟶ value

t1 is a list of $\mathbf{n-1}$ integers representing a character of $\mathbf{U_n}(\mathbb{R})$.

t2 is a list of $\mathbf{n-1}$ integers representing another character of $\mathbf{U_n}(\mathbb{R})$.

c is a list of $\mathbf{n-1}$ non-zero integers specifying the diagonal of a matrix. (Note that the 1st element of the matrix is $\mathbf{det(w)/c_{n-1}}$, the second $\mathbf{c_{n-1}/c_{n-2}}$ and so on down to the last $\mathbf{c_1}$ as in the notation of 11.2.1.)

w is an $\mathbf{n} \times \mathbf{n}$ matrix which is zero except for a single 1 in each row and column, representing an explicit element of the Weyl subroup $\mathbf{W_n}$ of $\mathbf{GL}(\mathbf{n},\mathbb{R})$.

value is a sum of complex exponentials being a Kloosterman sum when it is well defined.

Example:

This $n = 4$ example shows how KloostermanCompatibility should be run after selecting a permutation matrix. Then KloostermanSum is called with compatible arguments.

```
In[37]:= w = WeylGroup[4][[17]]

Out[37]= {{0, 0, 1, 0}, {0, 0, 0, 1}, {1, 0, 0, 0}, {0, 1, 0, 0}}

In[31]:= suf[klc[t1, t2, c, w, v]]
```

Out[31]= $\left\{\left\{\frac{c_2\, t1_3}{c_1{}^2} - t2_1,\ \frac{c_2\, t1_1}{c_3{}^2} - t2_3\right\},\ \{\{c_1, c_2\}, \{c_3, c_2\}\},\ \{v[\{2\}] \to c_1,\ v[\{1\}] \to 0,\ v[\{1, 2\}] \to c_2,\ v[\{1, 2, 4\}] \to c_3,\ v[\{1, 2, 3\}] \to 0\}\right\}$

```
In[32]:= kls[{3, 7, 12}, {4, 13, 1}, {3, 3, 3}, w]
```

Out[32]= $9\, e^{-\frac{2 i \pi}{3}} + 8\, e^{\frac{2 i \pi}{3}}$

Example:

This first illustrates commutativity of the LongElement sums (c/f [Fridberg, 1987, Proposition 2.5]), then Proposition 11.2.4 and finally is given an example of a classical sum showing it is real.

```
In[21]:= kls[{4, 13}, {6, 7}, {3, 3}, LongElement[3]]

Out[21]= 4 + 3 e^(-2 i π/3) + 3 e^(2 i π/3)

In[22]:= kls[{6, 7}, {4, 13}, {3, 3}, LongElement[3]]

Out[22]= 4 + 3 e^(-2 i π/3) + 3 e^(2 i π/3)

In[23]:= kls[{9, 7}, {1, 13}, {3, 3}, LongElement[3]]

Out[23]= 4 + 3 e^(-2 i π/3) + 3 e^(2 i π/3)

In[40]:= a = kls[{4, 13}, {6, 7}, {12, 31}, LongElement[3]]

Out[40]= 2 e^(-11 i π/186) + e^(11 i π/186) + e^(-17 i π/186) + 2 e^(17 i π/186) + 2 e^(-25 i π/186) + 2 e^(25 i π/186) + 2 e^(-29 i π/186) + 2 e^(29 i π/186) +
  2 e^(-i π/6) + e^(i π/6) + e^(-37 i π/186) + 2 e^(37 i π/186) + e^(-41 i π/186) + 2 e^(41 i π/186) + 2 e^(-47 i π/186) + e^(47 i π/186) +
  e^(-53 i π/186) + 2 e^(53 i π/186) + 2 e^(-71 i π/186) + e^(71 i π/186) + 2 e^(-73 i π/186) + e^(73 i π/186) + e^(-77 i π/186) + 2 e^(77 i π/186) +
  2 e^(-79 i π/186) + e^(79 i π/186) + e^(-83 i π/186) + 2 e^(83 i π/186) + 2 e^(-91 i π/186) + e^(91 i π/186) + 2 e^(-95 i π/186) + e^(95 i π/186) +
  2 e^(-103 i π/186) + e^(103 i π/186) + 2 e^(-107 i π/186) + e^(107 i π/186) + e^(-109 i π/186) + 2 e^(109 i π/186) + e^(-113 i π/186) +
  2 e^(113 i π/186) + 2 e^(-115 i π/186) + e^(115 i π/186) + e^(-133 i π/186) + 2 e^(133 i π/186) + 2 e^(-139 i π/186) + e^(139 i π/186) +
  e^(-145 i π/186) + 2 e^(145 i π/186) + e^(-149 i π/186) + 2 e^(149 i π/186) + e^(-5 i π/6) + 2 e^(5 i π/6) + 2 e^(-157 i π/186) +
  2 e^(157 i π/186) + 2 e^(-161 i π/186) + 2 e^(161 i π/186) + 2 e^(-169 i π/186) + e^(169 i π/186) + 2 e^(-175 i π/186) + e^(175 i π/186)

In[25]:= b = kls[{35, 25}, {18, 38}, {12, 31}, LongElement[3]];

In[26]:= a - b

Out[26]= 0

In[35]:= s = KloostermanSum[{24}, {13}, {43}, LongElement[2]]

Out[35]= e^(-2 i π/43) + e^(2 i π/43) + 2 e^(-8 i π/43) + 2 e^(8 i π/43) + 2 e^(-10 i π/43) + 2 e^(10 i π/43) + 2 e^(-12 i π/43) + 2 e^(12 i π/43) +
  2 e^(-20 i π/43) + 2 e^(20 i π/43) + 2 e^(-24 i π/43) + 2 e^(24 i π/43) + 2 e^(-28 i π/43) + 2 e^(28 i π/43) + 2 e^(-30 i π/43) +
  2 e^(30 i π/43) + 2 e^(-32 i π/43) + 2 e^(32 i π/43) + 2 e^(-38 i π/43) + 2 e^(38 i π/43) + 2 e^(-42 i π/43) + 2 e^(42 i π/43)

In[36]:= Im[ExpToTrig[s]]

Out[36]= 0
```

See also: BruhatForm, BruhatCVector, KloostermanCompatibility, KloostermanBruhatCell, PluckerCoordinates, PluckerInverse, PluckerRelations.

LanglandsForm (llf)

This function returns a list of the three matrices of the Langlands decomposition of a square matrix in a parabolic subgroup specified by a partition of the matrix dimension.
See Section 10.2.

LanglandsForm[p, d] ⟶ {u, c, m}

p is a square matrix of CREs,

d is a list of **r** positive integers of length at most **n** with sum **n**,

u is a unipotent block upper triangular matrix,

c is a diagonal matrix with **r** diagonal blocks each being a positive constant times the identity,

m is a block diagonal matrix with each diagonal block having determinant $\pm\mathbf{1}$.

Example:

```
In[1]:= d = {2, 2}; a = {{1, 2, 3, 4}, {5, 6, 7, 8}, {0, 0, 1, 2}, {0, 0, 3, 4}}

        MatrixForm[a]
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 3 & 4 \end{pmatrix}$$

```
In[19]:= {u, c, m} = LanglandsForm[a, d];

In[20]:= Map[MatrixForm, {u, c, m}]
```

$$\left\{\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & -2 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 \\ 0 & 0 & 0 & \sqrt{2} \end{pmatrix}, \begin{pmatrix} \frac{1}{2} & 1 & 0 & 0 \\ \frac{5}{2} & 3 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \sqrt{2} \\ 0 & 0 & \frac{3}{\sqrt{2}} & 2\sqrt{2} \end{pmatrix}\right\}$$

```
In[21]:= MatrixForm[u.c.m]

Out[21]//MatrixForm=
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 3 & 4 \end{pmatrix}$$

See also: ParabolicQ, LanglandsIFun.

LanglandsIFun (lif)

This function computes the summand for Langlands' Eisenstein series with respect to a specified parabolic subgroup.
See Chapter X, Definition 10.4.5.

$$\textbf{LanglandsIFun[g, p, s]} \longrightarrow \mathbf{I_s(g.z)}$$

g is a non-singular matrix of CREs in the parabolic subgroup specified by the second argument,

p is a list of **r** positive integers representing a partition of the matrix dimension,

s is list of **r** CREs such that $\sum_{\mathbf{i=1}}^{\mathbf{r}} \mathbf{d_i s_i} = \mathbf{0}$,

$\mathbf{I_s(g.z)}$ is the summand for the Langlands Eisenstein series.

Example:

```
In[20]:= g = {{1, x, 3}, {4, 5, x^2}, {0, 0, x + 1}};

In[21]:= MatrixForm[g]

Out[21]//MatrixForm=
```

$$\begin{pmatrix} 1 & x & 3 \\ 4 & 5 & x^2 \\ 0 & 0 & 1+x \end{pmatrix}$$

```
In[11]:= d = {2, 1};

In[23]:= LanglandsIFun[g, d, {1 + I, -2 - 2 I}]

Out[23]= Abs[5 - 4 x]^(1+I) Abs[1 + x]^(-2-2 I)
```

See also: LanglandsForm.

LeadingMatrixBlock (lmb)

This function extracts a leading matrix block of specified dimensions.

LeadingMatrixBlock[a, i, j] ⟶ b

a is a matrix of CREs,

i is a valid row index for **a**,

j is a valid column index for **a**,

b is the leading block of **a** with **i** rows and **j** columns.

See also: BlockMatrix, TailingMatrixBlock, GetMatrixElement.

LongElement (lel)

This function constructs the so-called long element of the group GL(n,ℤ), a matrix with 1's along the reverse leading diagonal and 0's elsewhere. See Chapter V.

LongElement[n] ⟶ w

n is a strictly positive integer,

w is an **n** by **n** matrix with 1's down the reversed leading diagonal and 0's elsewhere.

Example:

```
In[267]:= MatrixForm[LongElement[4]]

Out[267]//MatrixForm=
        ⎛0  0  0  1⎞
        ⎜0  0  1  0⎟
        ⎜0  1  0  0⎟
        ⎝1  0  0  0⎠
```

See also: WMatrix, ModularGenerators.

LowerTriangleToMatrix (ltm)

This function takes a list of lists of increasing length and forms a matrix with zeros in the upper triangle and the given lists constituting the rows of the lower triangle.

LowerTriangleToMatrix[**l**] ⟶ **a**

l is a list of lists of CREs of strictly increasing length representing the elements of a lower triangular sub-matrix including the diagonal. The first has length 1 and each successive sub-list has length 1 more than that preceding sub-list.

a is a full matrix with 0 in each upper triangular position.

Example:

```
In[185]:= MatrixForm[
            LowerTriangleToMatrix[
               {{a}, {b, c}, {d, e, f}}]]
```

```
Out[185]//MatrixForm=
        ⎛a  0  0⎞
        ⎜b  c  0⎟
        ⎝d  e  f⎠
```

See also: UpperTriangleToMatrix.

MakeBlockMatrix (mbm)

This function takes a list of lists of matrices and creates a single matrix wherein the j^{th} matrix element of the i^{th} sublist constitutes the i, j^{th} sub-block of this matrix. In order that this construction succeed, the original matrices must have compatible numbers of rows and columns, i.e. the matrices in each sublist must have the same number of rows for that sublist and for each j the j^{th} matrix in each sublist must have the same number of columns. In spite of this restriction, the function is a tool for building matrices rapidly when they have a natural block structure.

MakeBlockMatrix[A] $\longrightarrow$ **B**

A is a list of lists of equal length of matrix elements, each matrix having CRE elements,

B is a single matrix with sub-blocks being being the individual matrices in **A**.

Examples:

```
In[101]:=
        a11 = IdentityMatrix[2]; a12 = ZeroMatrix[2, 4];
        a21 = ConstantMatrix[x, 3, 2]; a22 = ConstantMatrix[1, 3, 4];
        MakeBlockMatrix[{{a22, a11}, {a21, a21}}]

        MakeBlockMatrix::arg3 : The submatrices must have compatible row and column numbers.

Out[103]=
        $Aborted

In[104]:=
        MakeBlockMatrix[{{a11, a12}, {a21, a22}}] // MatrixForm

Out[104]//MatrixForm=
        ⎛ 1  0  0  0  0  0 ⎞
        ⎜ 0  1  0  0  0  0 ⎟
        ⎜ x  x  1  1  1  1 ⎟
        ⎜ x  x  1  1  1  1 ⎟
        ⎝ x  x  1  1  1  1 ⎠
```

See also: ConstantMatrix, ZeroMatrix,LongElement, WeylGroup, SpecialWeylGroup.

MakeMatrix (mkm)

This function returns a symbolic

$$\textbf{MakeMatrix[“a”, m, n]} \longrightarrow \textbf{A}$$

“a” is a string being the name of the generic symbolic matrix element variable $\mathbf{a}[\mathbf{i},\mathbf{j}]$,

m is a strictly positive integer representing the number of rows of A,

n is a strictly positive integer representing the number of columns of A,

A is a symbolic matrix with $\mathbf{i},\mathbf{j}^{\mathbf{th}}$ entry $\mathbf{a}[\mathbf{i},\mathbf{j}]$.

Examples:

```
In[105]:=
        MakeMatrix["a", 5, 6] // MatrixForm

Out[105]//MatrixForm=
        ⎛ a[1, 1]  a[1, 2]  a[1, 3]  a[1, 4]  a[1, 5]  a[1, 6] ⎞
        ⎜ a[2, 1]  a[2, 2]  a[2, 3]  a[2, 4]  a[2, 5]  a[2, 6] ⎟
        ⎜ a[3, 1]  a[3, 2]  a[3, 3]  a[3, 4]  a[3, 5]  a[3, 6] ⎟
        ⎜ a[4, 1]  a[4, 2]  a[4, 3]  a[4, 4]  a[4, 5]  a[4, 6] ⎟
        ⎝ a[5, 1]  a[5, 2]  a[5, 3]  a[5, 4]  a[5, 5]  a[5, 6] ⎠
```

See also:
MakeYMatrix, MakeZMatrix, MakeBlockMatrix, ZeroMatrix, ConstantMatrix, InsertMatrixElement, WeylGroup, ModularGenerators, LongElement, WMatrix, SpecialWeylGroup.

MakeXMatrix (mxm)

This function returns a symbolic upper triangular square matrix of given dimension with 1's on the leading diagonal, i.e. a unipotent matrix.
See Definition 1.2.3.

MakeXMatrix[n, "x"] ⟶ u

n is a strictly positive integer representing the size of the matrix,

"x" is a string being the name of the generic symbolic matrix element variable $\mathbf{x}[\mathbf{i}, \mathbf{j}]$,

u is an upper-triangular symbolic matrix with 1's on the leading diagonal.

Examples:

```
In[8]:= MatrixForm[MakeXMatrix[4, "x"]]

Out[8]//MatrixForm=
⎛1  x[1, 2]  x[1, 3]  x[1, 4]⎞
⎜0     1     x[2, 3]  x[2, 4]⎟
⎜0     0        1     x[3, 4]⎟
⎝0     0        0        1   ⎠

In[9]:= MakeXVariables[4, "x"]

Out[9]= {x[1, 2], x[1, 3], x[1, 4], x[2, 3], x[2, 4], x[3, 4]}

In[11]:= MatrixForm[MakeYMatrix[4, "y"]]

Out[11]//MatrixForm=
⎛y[1] y[2] y[3]      0       0   0⎞
⎜      0        y[1] y[2]    0   0⎟
⎜      0           0      y[1]  0⎟
⎝      0           0        0   1⎠

In[12]:= MakeYVariables[4, "y"]

Out[12]= {y[1], y[2], y[3]}

In[24]:= MatrixForm[MakeZMatrix[4, "x", "y"]]

Out[24]//MatrixForm=
⎛y[1] y[2] y[3]  x[1, 2] y[1] y[2]  x[1, 3] y[1]  x[1, 4]⎞
⎜      0            y[1] y[2]      x[2, 3] y[1]  x[2, 4]⎟
⎜      0               0              y[1]       x[3, 4]⎟
⎝      0               0               0           1    ⎠

       MakeZVariables[3, "x", "y"]

Out[25]= {x[1, 2], x[1, 3], x[1, 4], x[2, 3], x[2, 4], x[3, 4], y[1], y[2], y[3]}
```

See also: MakeXVariables, MakeYMatrix, MakeYVariables, MakeZMatrix, MakeZVariables.

MakeXVariables (mxv)

This function returns a list of the x-variables which appear in the symbolic generic Iwasawa form of a square matrix of given dimension.
See Definition 1.2.3.

MakeXVariables[n, "x"] ⟶ l

n is a strictly positive integer representing the size of the matrix,

"x" is a string being the name of the generic list element variable $\mathbf{x}[\mathbf{i},\mathbf{j}]$,

l is a list of the **x**-variables in order of increasing row index.

See also: MakeXMatrix, MakeYMatrix, MakeYVariables, MakeZMatrix.

MakeYMatrix (mym)

This function returns a symbolic diagonal matrix of given dimension with values on the leading diagonal being the product of the y-variables of a matrix expressed in Iwasawa form.
See Definition 1.2.3 and the manual entry for MakeXMatrix.

MakeYMatrix[n, "y"] ⟶ d

n is a strictly positive integer representing the size of the matrix,

"y" is a string being the name of the generic symbolic matrix element variable $\mathbf{y}[\mathbf{i}]$ such that the $\mathbf{j^{th}}$ diagonal element is the product $\mathbf{y}[\mathbf{1}]\mathbf{y}[\mathbf{2}]\cdots\mathbf{y}[\mathbf{n}-\mathbf{j}]$.

d is a diagonal matrix.

See also: MakeXMatrix, MakeXVariables, MakeYVariables, MakeZMatrix.

MakeYVariables (myv)

This function returns a symbolic list of the $n-1$ y-variables which would occur in the Iwasawa form of a matrix of size $n \times n$.
See Definition 1.2.3 and the manual entry for MakeXMatrix.

MakeYVariables[n, "y"] ⟶ l

n is a strictly positive integer representing the size of the matrix,

"y" is a string being the name of the generic variable $\mathbf{y}[\mathbf{i}]$,

l is a list of the form $\{\mathbf{y}[\mathbf{1}],\cdots,\mathbf{y}[\mathbf{n}-\mathbf{1}]\}$.

See also: MakeXMatrix, MakeXVariables, MakeYMatrix, MakeZMatrix, MakeZVariables.

MakeZMatrix (mzm)

This function returns a symbolic upper triangular square matrix of given dimension being in generic Iwasawa form.
See Example 1.2.4 and the manual entry for MakeXMatrix.

MakeZMatrix[n, "x","y"] $\longrightarrow$ u

n is a strictly positive integer representing the size of the matrix,

"x" is a string being the name of the generic symbolic Iwasawa x-variable $\mathbf{x}[\mathbf{i},\mathbf{j}]$,

"y" is a string being the name of the generic symbolic Iwasawa y-variable $\mathbf{y}[\mathbf{i}]$,

u is an upper-triangular symbolic matrix with $\mathbf{i},\mathbf{j}^{\mathbf{th}}$ element having the form $\mathbf{x}[\mathbf{i},\mathbf{j}]\mathbf{y}[\mathbf{1}]\cdots\mathbf{y}[\mathbf{n}-\mathbf{j}]$.

See also: MakeXMatrix, MakeXVariables, MakeYMatrix, MakeYVariables, MakeZVariables.

MakeZVariables (mzv)

This function returns a list of the variables which occur in the Iwasawa form for a matrix with generic symbolic entries and of given size.
See the manual entry for MakeXMatrix.

MakeZVariables[n, "x", "y"] $\longrightarrow$ l

n is a strictly positive integer representing the size of the matrix,

"x" is a string being the name of the generic symbolic matrix element $\mathbf{x}[\mathbf{i},\mathbf{j}]$ with $\mathbf{i}>\mathbf{j}$,

"y" is a string being the name of the generic symbolic matrix element $\mathbf{y}[\mathbf{i}]$,

l is a list of the Iwasawa variables with the **x**-variables first in order of increasing row index followed by the **y**-variables:

$$\{\mathbf{x}[\mathbf{1},\mathbf{2}],\cdots,\mathbf{x}[\mathbf{1},\mathbf{n}],\mathbf{x}[\mathbf{2},\mathbf{3}],\cdots,\mathbf{x}[\mathbf{n}-\mathbf{1},\mathbf{n}],\mathbf{y}[\mathbf{1}],\cdots,\mathbf{y}[\mathbf{n}-\mathbf{1}]\}.$$

See also: MakeXMatrix, MakeXVariables, MakeYMatrix, MakeYVariables, MakeZMatrix.

MatrixColumn (mcl)

This function returns a given column of a matrix.

MatrixColumn[m, j] $\longrightarrow$ c

m is a matrix of CREs.

j is a valid column index for **m**.

c is the $\mathbf{j}^{\mathbf{th}}$ column of **m** returned as a list.

See also: MatrixRow.

MatrixDiagonal (mdl)

This function extracts the diagonal of a matrix.

MatrixDiagonal[a] ⟶ d

a is a square matrix of CREs,

d is a list, being the diagonal entries of **a** in the same order.

See also: DiagonalToMatrix.

MatrixJoinHorizontal (mjh)

This function assembles a new matrix by placing one matrix to the right of another compatible matrix.

MatrixJoinHorizontal[a, b] ⟶ c

a is a matrix of CREs,

b is a matrix with the same number of rows as **a**,

c is a matrix with block decomposition **c** = [**a**|**b**].

Example:

```
In[285]:= A = {{a, b, c}, {d, e, f}}; B = {{1, 1}, {2, 2}};

In[292]:= MatrixForm[MatrixJoinHorizontal[A, B]]

Out[292]//MatrixForm=
        ⎛ a  b  c  1  1 ⎞
        ⎝ d  e  f  2  2 ⎠
```

See also: MatrixJoinVertical.

MatrixJoinVertical (mjv)

This function assembles a new matrix by placing one matrix above another compatible matrix.

MatrixJoinVertical[a, b] $\longrightarrow$ **c**

a is a matrix of CREs,

b is a matrix with the same number of columns as **a**,

c is a matrix with block decomposition having **a** above **b**.

See also: MatrixJoinHorizontal.

MatrixUpperTriangle (mut)

This function extracts the elements in the upper triangle, including the diagonal, of a square matrix and returns a list of lists of the elements from each row.

MatrixUpperTriangle[**a**] $\longrightarrow$ **t**

a is a square matrix of CREs,

t is a list of lists of elements with the $\mathbf{i}^{\mathbf{th}}$ element of the $\mathbf{j}^{\mathbf{th}}$ list being the $(\mathbf{j}, \mathbf{i}+\mathbf{j}-\mathbf{1})^{\mathbf{th}}$ element of **a**.

```
In[303]:= A = {{a, b, c}, {d, e, f}, {g, h, i}};

In[316]:= MatrixForm[A]

Out[316]//MatrixForm=
        a  b  c
        d  e  f
        g  h  i

In[320]:= MatrixUpperTriangle[A]

Out[320]= {{a, b, c}, {e, f}, {i}}
```

See also: MatrixLowerTriangle, UpperTriangleToMatrix.

MatrixLowerTriangle (mlt)

This function extracts the elements in the lower triangle of a square matrix, including the diagonal, and returns them as a list of lists.

MatrixLowerTriangle[**a**] $\longrightarrow$ **t**

a is a square matrix of CREs,

t is a list of lists where the $\mathbf{i}^{\mathbf{th}}$ element of the $\mathbf{j}^{\mathbf{th}}$ list represents the $(\mathbf{j}, \mathbf{i})^{\mathbf{th}}$ element of **a**.

See also: MatrixUpperTriangle, LowerTriangleToMatrix.

MatrixRow (mro)

This function returns a given row of a matrix.

MatrixRow[m, i] ⟶ r

m is a matrix of CREs,

i is a row index of **m**,

r is a list representing the $\mathbf{i^{th}}$ row of **m**.

See also: MatrixColumn.

ModularGenerators (mog)

This function returns a list of two matrix generators for the subgroup of the group of integer matrixes with determinant 1, i.e. generators of SL(n,ℤ).
See Chapter 5, especially Section 5.9.

ModularGenerators[n] ⟶ g

n is a positive integer with **n ≥ 2**,

g is a list of two **n** by **n** matrices which will generate **SL(n,ℤ)**.

Example:

```
In[174]:= Map[MatrixForm, ModularGenerator[6]]
```

$$\text{Out[174]=}\ \left\{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}\right\}$$

See also: WeylGenerator, WeylGroup, SpecialWeylGroup, WMatrix, LongElement.

MPEisensteinLambdas (eil)

This function computes the functions $\lambda_i(v) : \mathbb{C}^{n-1} \to \mathbb{C}$ such that the L-function associated with the minimal parabolic Eisenstein series $L_{E_v}(s)$ is a product of shifted zeta values

$$L_{E_v}(s) = \prod_{i=1}^{n} \zeta(s - \lambda_i(v)).$$

See Chapter X, 10.4.1 and Theorem 10.8.6.

MPEisensteinLambdas[v] $\longrightarrow$ **L**

v is a list of $\mathbf{n-1}$ CREs with $\mathbf{n \geq 2}$ representing complex parameters,

L is a list of affine expressions in the elements of **v** representing the functions $\lambda_{\mathbf{i}}(\mathbf{v})$.

Example:

```
In[10]:= suf[MPEisensteinLambdas[{v[1], v[2], v[3]}]]
```

$$\text{Out[10]=}\ \left\{-\frac{3}{2} + 3\,v_1 + 2\,v_2 + v_3,\ -\frac{1}{2} - v_1 + 2\,v_2 + v_3,\ \frac{1}{2} - v_1 - 2\,v_2 + v_3,\ \frac{3}{2} - v_1 - 2\,v_2 - 3\,v_3\right\}$$

See also: MPEisensteinSeries, MPEisensteinGamma, MPExteriorPowerGamma, MPExteriorPowerLFun, MPSymmetricPowerLFun, MPSymmetricPowerGamma.

MPEisensteinSeries (eis)

This function computes the L-function associated with the minimal parabolic Eisenstein series $E_v(z)$ as a product of shifted zeta values

$$L_{E_v}(s) = \prod_{i=1}^{n} \zeta(s - \lambda_i(v)).$$

See Chapter X, 10.4.1 and Theorem 10.8.6.

MPEisensteinSeries[s,v] $\longrightarrow$ **Z**

s is a CRE representing a complex number,

v is a list of $\mathbf{n-1}$ CREs with $\mathbf{n \geq 2}$ representing complex parameters,

Z is a product of **n** values of the Riemann zeta function at shifted arguments.

Example:

See also: MPEisensteinLambdas, MPEisensteinGamma, MPExteriorPowerGamma, MPExteriorPowerLFun, MPSymmetricPowerLFun, MPSymmetricPowerGamma.

```
In[4]:= suf[MPEisensteinSeries[s, {v[1], v[2]}]]

Out[4]= Zeta[1 + s - 2 v1 - v2] Zeta[s + v1 - v2] Zeta[-1 + s + v1 + 2 v2]
```

MPEisensteinGamma (eig)

This function computes the gamma factors for the minimal parabolic Eisenstein series

$$G_{E_v}(s) = \pi^{-ns/2} \prod_{i=1}^{n} \Gamma\Big(\frac{s - \lambda_i(v)}{2}\Big).$$

See Chapter X, Theorem 10.8.6.

MPEisensteinGamma[s,v] $\longrightarrow$ **G**

s is a CRE representing a complex number,

v is a list of $\mathbf{n-1}$ CREs with $\mathbf{n \geq 2}$ representing complex parameters,

G is the gamma factor for the minimal parabolic Eisenstein series functional equation.

Example:

```
In[5]:= suf[MPEisensteinGamma[s, {v[1], v[2]}]]

Out[5]= π^(-3 s/2) Gamma[1/2 (1 + s - 2 v1 - v2)]
        Gamma[1/2 (s + v1 - v2)] Gamma[1/2 (-1 + s + v1 + 2 v2)]
```

See also: MPEisensteinLambdas, MPEisensteinSeries, MPExteriorPowerGamma, MPExteriorPowerLFun, MPSymmetricPowerLFun, MPSymmetricPowerGamma.

MPExteriorPowerGamma (epg)

This function returns the gamma factors for the k^{th} symmetric L-function associated with a minimal parabolic Eisenstein series. See the introduction to Chapter XIII.

MPExteriorPowerGamma[s,v,k] $\longrightarrow$ **G**

s is a CRE representing a complex number,

v is a list of $\mathbf{n-1}$ CREs with $\mathbf{n \geq 2}$ representing complex parameters,

k is a natural number $\mathbf{k \geq 1}$ representing the order of the exterior power,

G is the gamma factor for the functional equation of the exterior power.

```
In[9]:= MPExteriorPowerGamma[s, {v1, v2}, 2]
```

$$\text{Out[9]=}\quad \pi^{-3\,s/2}\,\text{Gamma}\left[\frac{1}{2}\,(1+s-v1-2\,v2)\right]\text{Gamma}\left[\frac{1}{2}\,(s-v1+v2)\right]\text{Gamma}\left[\frac{1}{2}\,(-1+s+2\,v1+v2)\right]$$

Example:

See also: MPEisensteinLambdas, MPEisensteinSeries, MPEisensteinGamma, MPExteriorPowerLFun, MPSymmetricPowerLFun, MPSymmetricPowerGamma.

MPExteriorPowerLFun (epl)

This function returns the k^{th} exterior power of the L-function of a minimal parabolic Eisenstein series as a product of zeta values. See the introduction to Chapter XIII. This function can be used to show that exterior power L-functions satisfy a functional equation.

MPExteriorPowerLFun[s,v,k] $\longrightarrow$ **Z**

s is a CRE representing a complex number,

v is a list of $\mathbf{n-1}$ CREs with $\mathbf{n \geq 2}$ representing complex parameters,

k is a natural number $\mathbf{k \geq 1}$ representing the order of the exterior power,

Z is a product of Riemann zeta function values.

Example:

```
In[7]:= suf[MPExteriorPowerLFun[s, {v[1], v[2]}, 2]]

Out[7]= Zeta[1 + s - v1 - 2 v2] Zeta[s - v1 + v2] Zeta[-1 + s + 2 v1 + v2]
```

See also: MPEisensteinLambdas, MPEisensteinSeries, MPEisensteinGamma, MPExteriorPowerGamma, MPSymmetricPowerLFun, MPSymmetricPowerGamma.

MPSymmetricPowerLFun (spf)

This function returns the k^{th} symmetric power of the L-function of a minimal parabolic Eisenstein series as a product of zeta values. See the introduction to Chapter XIII. This can be used to show that symmetric power L-functions satisfy a functional equation.

MPSymmetricPowerLFun[s,v,k] $\longrightarrow$ **Z**

s is a CRE representing a complex number,

v is a list of $\mathbf{n-1}$ CREs with $\mathbf{n \geq 2}$ representing complex parameters,,

k is a natural number $\mathbf{k \geq 1}$ representing the order of the exterior power,

Z is a product of Riemann zeta function values.

Example:

```
In[6]:= suf[MPSymmetricPowerLFun[s, {v[1], v[2]}, 3]]

Out[6]= Zeta[s] Zeta[1 + s - 3 v1] Zeta[-1 + s + 3 v1] Zeta[1 + s - 3 v2]
         Zeta[3 + s - 6 v1 - 3 v2] Zeta[2 + s - 3 v1 - 3 v2] Zeta[s + 3 v1 - 3 v2]
         Zeta[-1 + s + 3 v2] Zeta[-2 + s + 3 v1 + 3 v2] Zeta[-3 + s + 3 v1 + 6 v2]
```

See also: MPEisensteinLambdas, MPEisensteinSeries, MPEisensteinGamma, MPExteriorPowerGamma, MPExteriorPowerLFun, MPSymmetricPowerGamma.

MPSymmetricPowerGamma (spg)

This function returns the gamma factors for the k^{th} symmetric L-function associated with a mimimal parabolic Eisenstein series. See the introduction to Chapter XIII.

MPSymmetricPowerGamma[s,v,k] $\longrightarrow$ G

s is a CRE representing a complex number,

v is a list of $\mathbf{n-1}$ CREs with $\mathbf{n \geq 2}$ representing complex parameters,

k is a natural number $\mathbf{k \geq 1}$ representing the order of the exterior power,

G is the gamma factors for the $\mathbf{k}^{\text{th}}$ symmetric power L-function.

Example:

```
In[8]:= suf[MPSymmetricPowerGamma[s, {v[1], v[2]}, 3]]
```

$$\text{Out[8]=}\ \pi^{-5\,s}\,\Gamma\left[\frac{s}{2}\right]\Gamma\left[\frac{1}{2}\,(1+s-3\,v_1)\right]\Gamma\left[\frac{1}{2}\,(-1+s+3\,v_1)\right]\Gamma\left[\frac{1}{2}\,(1+s-3\,v_2)\right]\Gamma\left[\frac{1}{2}\,(3+s-6\,v_1-3\,v_2)\right]\Gamma\left[\frac{1}{2}\,(2+s-3\,v_1-3\,v_2)\right]\Gamma\left[\frac{1}{2}\,(s+3\,v_1-3\,v_2)\right]\Gamma\left[\frac{1}{2}\,(-1+s+3\,v_2)\right]\Gamma\left[\frac{1}{2}\,(-2+s+3\,v_1+3\,v_2)\right]\Gamma\left[\frac{1}{2}\,(-3+s+3\,v_1+6\,v_2)\right]$$

See also: MPEisensteinLambdas, MPEisensteinSeries, MPEisensteinGamma, MPExteriorPowerGamma, MPExteriorPowerLFun, MPSymmetricPowerLFun.

NRows (nro)

This function gives the number of rows of a matrix.

NRows[a] $\longrightarrow$ m

a is a matrix of CREs,

m is the number of rows of **a**.

See also: NColumns.

NColumns (ncl)

This function gives the number of columns of a matrix.

NColumns[a] $\longrightarrow$ n

a is a matrix of CREs,

n is the number of columns of **a**.

See also: NRows.

ParabolicQ (paq)

This function tests a square matrix to see whether or not it is in a given parabolic subgroup as specified by a non-trivial partition of the matrix dimension.
See Chapter X, especially Section 10.1.

ParabolicQ[a, d] ⟶ ans

a is an **n** × **n** square matrix with entries which are CREs,

d is list of at most **n** positive integers with sum **n**,

ans is **True if a** is in the specified subgroup and **False** otherwise.

Example:

```
In[1]:= d = {2, 2}; a = {{1, 2, 3, 4}, {5, 6, 7, 8}, {0, 0, 1, 2}, {0, 0, 3, 4}}

Out[1]= {{1, 2, 3, 4}, {5, 6, 7, 8}, {0, 0, 1, 2}, {0, 0, 3, 4}}

In[2]:= MatrixForm[a]

Out[2]//MatrixForm=
        ⎛1  2  3  4⎞
        ⎜5  6  7  8⎟
        ⎜0  0  1  2⎟
        ⎝0  0  3  4⎠

In[3]:= ParabolicQ[a, d]

Out[3]= True

In[57]:= a2 = {{1, 2, 3, 4}, {5, 6, 7, 8}, {0, 0, 1, 2}, {x, 0, 3, 4}}

Out[57]= {{1, 2, 3, 4}, {5, 6, 7, 8}, {0, 0, 1, 2}, {x, 0, 3, 4}}

In[58]:= ParabolicQ[a1, d]

Out[58]= False
```

See also: LanglandsForm, LanglandsIFun.

PluckerCoordinates (plc)

This function takes an $\mathbf{n} \times \mathbf{n}$ square matrix and returns a list of lists of the so-called Plücker coordinates, namely the values of all of the bottom $\mathbf{j} \times \mathbf{j}$ minors with $\mathbf{1} \leq \mathbf{j} \leq \mathbf{n} - \mathbf{1}$.
See Chapter XI, Section 11.3, Theorem 11.3.1.

PluckerCoordinates[a] ⟶ value

a is an $\mathbf{n} \times \mathbf{n}$ matrix of CREs.

value is a list of lists being the values of all of the $\mathbf{j} \times \mathbf{j}$ minor determinants with $\mathbf{1} \leq \mathbf{j} \leq \mathbf{n} - \mathbf{1}$ based on the bottom row and taking elements from the bottom **j** rows. The $\mathbf{j^{th}}$ sublist has the $\mathbf{j} \times \mathbf{j}$ minors in lexical order of the column indices.

Example:

```
In[31]:= PluckerCoordinates[{{1, 2}, {4 x, 5 y}}]

Out[31]= {{4 x, 5 y}}

In[32]:= suf[PluckerCoordinates[MakeMatrix["x", 3, 3]]]

Out[32]= {{x3,1, x3,2, x3,3},
           {-x2,2 x3,1 + x2,1 x3,2, -x2,3 x3,1 + x2,1 x3,3, -x2,3 x3,2 + x2,2 x3,3}}

In[33]:= m = {{12, 3, 4, -1, 7}, {3, 0, 2, 1, 0}, {4, 5, 6, 7, 0},
            {0, 2, 19, 3, 1}, {1, 2, 3, 4, 5}}; MatrixForm[m]

Out[33]//MatrixForm=
        ⎛12  3   4   -1  7⎞
        ⎜ 3  0   2    1  0⎟
        ⎜ 4  5   6    7  0⎟
        ⎜ 0  2  19    3  1⎟
        ⎝ 1  2   3    4  5⎠

In[34]:= plc[m]

Out[34]= {{1, 2, 3, 4, 5}, {-2, -19, -3, -1, -32, 2, 8, 67, 92, 11},
           {-45, 9, 37, 153, 374, 51, 99, 412, -1, -578},
           {360, 1310, 34, -1462, 414}}
```

See also: PluckerRelations, PluckerInverse, KloostermanSum.

PluckerInverse (pli)

This function takes a list of lists of integers, which could be the Plücker coordinates arising from a square matrix, and returns such a matrix having determinant 1. The matrix is not unique but **PluckerInverse** followed by **PluckerCoordinates** gives the identity, provided the list of lists of integers is compatible, i.e. arises from some matrix. See Section 11.3.

$$\textbf{PluckerInverse}[\textbf{Ms}] \longrightarrow \textbf{a}$$

Ms is a list of $\mathbf{n-1}$ sublists of integers $\mathbf{Ms} = \{\{\mathbf{M_1}, \cdots, \mathbf{M_n}\}, \{\mathbf{M_{12}}, \cdots\}, \cdots\}$, representing the Plücker coordinates of a matrix in lexical order.

a is an integer matrix having those Plücker coordinates or **False** in case they are incompatible.

Example:

```
In[2162]:=
        g = ModularGenerators[4]; a = g[[1]]; b = g[[2]];
        m = a.a.a.b.b.b.b.a.b.b.b.a.b.b.a.a.a.b.a.a.a.b.b.b.a.a.b.a.a.a.a.a.a.b;
        MatrixForm[m]
```

```
Out[2163]//MatrixForm=
        ⎛0  -1  -4  -10⎞
        ⎜0   0  -1   -3⎟
        ⎜0   0   0   -1⎟
        ⎝1   8  16   40⎠
```

```
In[2164]:=
        Ms = plc[m]
```

```
Out[2164]=
        {{1, 8, 16, 40}, {0, 0, 1, 0, 8, 16}, {0, 0, 1, 8}}
```

```
In[2165]:=
        a = pli[Ms]; MatrixForm[a]
```

```
Out[2165]//MatrixForm=
        ⎛0  -1   0   0⎞
        ⎜0   0  -1   0⎟
        ⎜0   0   0  -1⎟
        ⎝1   8  16  40⎠
```

```
In[2166]:=
        plc[a]
```

```
Out[2166]=
        {{1, 8, 16, 40}, {0, 0, 1, 0, 8, 16}, {0, 0, 1, 8}}
```

See also: PluckerCoordinates, PluckerRelations, KloostermanSum, KloostermanCompatibility.

PluckerRelations (plr)

This function computes all the known quadratic relationships between the minors of a generic square $n \times n$ matrix known as the Plücker coordinates. See Chapter XI. In case $n = 2$ there are none and for $n = 3$ one. For $\mathbf{n > 3}$ the number grows dramatically. No claim is made that this function returns, for any given n, a complete set of independent relationships. By "complete" is meant sufficient to guarantee the coordinates arise from a member of $SL(n, \mathbb{Z})$.

PluckerRelations[n, v] $\longrightarrow$ relations

n is a positive integer with $\mathbf{n \geq 2}$.

v is a symbol representing the generic name used for the Plucker coordinates so $\mathbf{v}[\{\mathbf{i_1}, \cdots, \mathbf{i_j}\}]$ is the matrix minor based on the last **j** rows and the columns indexed by $\mathbf{i_1}, \cdots, \mathbf{i_j}$ with these indices in strictly increasing order.

relations is a list of quadratic expressions with coefficients $\pm\mathbf{1}$ in the $\mathbf{v}[\{\cdots\}]$s, which vanish whenever the values of the $\mathbf{v}[\{\cdots\}]$s come from the minors of an $\mathbf{n} \times \mathbf{n}$ matrix.

Example:

```
In[1]:= PluckerRelations[3, v]

Out[1]= {v[{3}] v[{1, 2}] - v[{2}] v[{1, 3}] + v[{1}] v[{2, 3}]}

In[2]:= PluckerRelations[4, v]

Out[2]= {v[{3}] v[{1, 2}] - v[{2}] v[{1, 3}] + v[{1}] v[{2, 3}],
         v[{4}] v[{1, 2}] - v[{2}] v[{1, 4}] + v[{1}] v[{2, 4}],
         v[{4}] v[{1, 3}] - v[{3}] v[{1, 4}] + v[{1}] v[{3, 4}],
         v[{4}] v[{2, 3}] - v[{3}] v[{2, 4}] + v[{2}] v[{3, 4}],
         v[{1, 4}] v[{2, 3}] - v[{1, 3}] v[{2, 4}] + v[{1, 2}] v[{3, 4}],
         v[{1, 4}] v[{1, 2, 3}] - v[{1, 3}] v[{1, 2, 4}] +
          v[{1, 2}] v[{1, 3, 4}], v[{4}] v[{1, 2, 3}] - v[{3}] v[{1, 2, 4}
          v[{2}] v[{1, 3, 4}] - v[{1}] v[{2, 3, 4}], v[{2, 4}] v[{1, 2, 3}
          v[{2, 3}] v[{1, 2, 4}] + v[{1, 2}] v[{2, 3, 4}],
         v[{3, 4}] v[{1, 2, 3}] - v[{2, 3}] v[{1, 3, 4}] +
          v[{1, 3}] v[{2, 3, 4}], v[{3, 4}] v[{1, 2, 4}] -
          v[{2, 4}] v[{1, 3, 4}] + v[{1, 4}] v[{2, 3, 4}]}

In[3]:= Map[Function[n, Length[PluckerRelations[n, v]]],
          {2, 3, 4, 5, 6, 7}]

Out[3]= {0, 1, 10, 47, 160, 458}
```

See also: PluckerCoordinates, PluckerInverse, KloostermanSum

RamanujanSum (rsm)

This function computes the Ramanujan sum $s(n, c)$ for explicit natural number values of n, c, namely

$$s(n,c) = \sum_{r=1,(r,c)=1}^{c} e^{2\pi i \frac{r}{c}}.$$

See Propositions 3.1.4, 3.1.7.

RamanujanSum[n,c] $\longrightarrow$ **s**

n is a strictly positive integer,

c is a strictly positive integer,

s is an integer being the Ramanujan sum.

Example:

```
In[27]:= Table[RamanujanSum[n, 1] - MoebiusMu[n], {n, 1, 10}]

Out[27]= {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}

In[31]:= Table[RamanujanSum[n, 23 n] / EulerPhi[n], {n, 1, 20}]

Out[31]= {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
```

RemoveMatrixRow (rmr)

A given row is removed from a matrix, leaving the original unchanged.

RemoveMatrixRow[a, i] $\longrightarrow$ **b**

a is a matrix of CREs,

i is a valid row index of **a**.

b is a matrix with all rows identical to **a** except the $\mathbf{i^{th}}$ which is missing.

See also: RemoveMatrixColumn.

RemoveMatrixColumn (rmc)

A given row is removed from a matrix, creating a new matrix and leaving the original unchanged.

RemoveMatrixColumn[a, j] $\longrightarrow$ **b**

a is a matrix of CREs,

j is a valid column index of **a**,

b is a matrix with all columns identical to **a** except the $\mathbf{j^{th}}$ which is missing.

See also: RemoveMatrixRow.

SchurPolynomial (spl)

This function computes the Schur polynomial in n variables $x_1, \cdots, x_n$ with $n-1$ exponents $k_1, \cdots, k_{n-1}$, that is to say the ratio of the determinant of a matrix with i,j^{th} element 1 for $i = n$ and $x_j^{k_1+\cdots+k_{i-1}+n-i}$ for $1 \le i \le n-1$, to the determinant of the matrix which is 1 for $i = n$ and x_j^{n-i} for $1 \le i \le n-1$.
See Section 7.4.

$$\textbf{SchurPolynomial[x, k]} \longrightarrow \mathbf{S_k(x_1, \cdots, x_n)}$$

$\mathbf{x}$ is list of $\mathbf{n}$ CREs,

$\mathbf{k}$ is a list of $\mathbf{n-1}$ CREs,

$\mathbf{S_k(x_1, \cdots, x_n)}$ is the Schur polynomial.

Example:

```
In[4]:= SchurPolynomial[{x, y, z}, {a, b}]
```

$$\text{Out[4]=} \quad \frac{x^{2+a+b}\,(y^{1+a}-z^{1+a})+y^{1+a}\,z^{1+a}\,(y^{1+b}-z^{1+b})+x^{1+a}\,(-y^{2+a+b}+z^{2+a+b})}{(x-y)\,(x-z)\,(y-z)}$$

```
In[5]:= SchurPolynomial[{1, x, x^2, x^3}, {2, 2, 2}]
```

$$\text{Out[5]=} \quad x^8\,(1-x+x^2)^2\,(1+x+x^2)^5\,(1+x^3+x^6)$$

```
In[6]:= SchurPolynomial[{1, x^2, x, x^3}, {1, 2, 3}]
```

$$\text{Out[6]=} \quad x^5\,(1+x+x^2)\,(1+2x+4x^2+7x^3+10x^4+13x^5+17x^6+19x^7+21x^8+22x^9+21x^{10}+19x^{11}+17x^{12}+13x^{13}+10x^{14}+7x^{15}+4x^{16}+2x^{17}+x^{18})$$

See also: HeckeMultiplicativeSplit.

SmithForm (smf)

This function returns the Smith form diagonal matrix d of a square non-singular matrix a with integer entries. This matrix d satisfies $0 < d_{i,i}$ and $d_{i,i} \mid d_{i+1,i+1}$ for all $i \leq n-1$. It also returns unimodular matrixes l, r such that $a = l.d.r$.
See Theorem 3.11.2.

SmithForm[a] ⟶ **{l, d, r}**

a is non-singular integer matrix,

l is a unimodular matrix,

d is a diagonal matrix, being the Smith Form of **a**,

r is a unimodular matrix.

Example:

The Smith form of a 4 by 4 matrix is computed and the result checked.

```
In[4]:= m4 = {{5, 2, -4, 7}, {1, 6, 0, -3}, {1, 2, -2, 4}, {7, 1, 5, 6}};

        {l, s, r} = SmithForm[m4]

        {{{5, -4, 186, -23}, {1, 0, 0, 0}, {1, -2, 89, -11}, {7, 5, -178, 22}},
         {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 964}},
         {{1, 6, 0, -3}, {0, -49, 1, 41}, {0, 118, 0, 1}, {0, 1, 0, 0}}}

        Map[MatrixForm, %]

        ⎧⎛5  -4   186  -23⎞  ⎛1  0  0   0⎞  ⎛1   6  0  -3⎞⎫
        ⎪⎜1   0     0    0⎟, ⎜0  1  0   0⎟, ⎜0  -49  1  41⎟⎪
        ⎨⎜1  -2    89  -11⎟  ⎜0  0  1   0⎟  ⎜0  118  0   1⎟⎬
        ⎪⎜7   5  -178   22⎟  ⎝0  0  0  964⎠  ⎝0   1  0   0⎠⎪
        ⎩⎝                ⎠                               ⎭

        l.s.r - m4

        {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}}
```

See also: SmithElementaryDivisors, SmithInvariantFactors, HermiteFormLower, HermiteFormUpper.

SmithElementaryDivisors (sed)

This function computes the elementary divisors of a non-singular $n \times n$ integer matrix a, i.e. for each j with $1 \leq j \leq n$, the gcd $d_j(a)$ of all of the $j \times j$ minor determinants. If s_j is the j^{th} diagonal entry of the Smith form then $s_j = d_j(a)/d_{j-1}(a)$.

SmithElementaryDivisors[a] $\longrightarrow$ **l**

a is a non-singular **n** $\times$ **n** integer matrix,

l is a list of the **n** Smith form elementary divisors of **a** in the order $\{\mathbf{d_1}(\mathbf{a}), \cdots, \mathbf{d_n}(\mathbf{a})\}$.

See also: SmithForm, SmithInvariantFactors, HermiteFormUpper, HermiteFormLower.

SmithInvariantFactors (sif)

This function computes the invariant factors of the Smith form of a non-singular integer matrix a. These are all of the prime powers which appear in the diagonal entries of the Smith form of a.

SmithInvariantFactors[a] $\longrightarrow$ **l**

a is an $n \times n$ non-singular integer matrix,

l is a list of prime powers.

See also: SmithForm, SmithElementaryDivisors, HermiteFormUpper, HermiteFormLower.

SpecialWeylGroup (swg)

This function, for each natural number n, returns the group of $n \times n$ matrices with each entry being 0 or ± 1, and having determinant 1. There are $2^{n-1}n!$ such matrices.
See Sections 6.3 and 6.5.

SpecialWeylGroup[n] ⟶ g

n is a natural number representing the matrix dimension,

g is a list of **n** × **n** matrices representing the Weyl group.

Example:

```
In[3]:= w = SpecialWeylGroup[3];

In[4]:= Map[MatrixForm, w]
```

$$\text{Out[4]=}\ \left\{\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}, \begin{pmatrix}1&0&0\\0&0&1\\0&-1&0\end{pmatrix}, \begin{pmatrix}1&0&0\\0&-1&0\\0&0&-1\end{pmatrix}, \begin{pmatrix}1&0&0\\0&0&-1\\0&1&0\end{pmatrix},\right.$$
$$\begin{pmatrix}0&1&0\\1&0&0\\0&0&-1\end{pmatrix}, \begin{pmatrix}0&1&0\\0&0&1\\1&0&0\end{pmatrix}, \begin{pmatrix}0&1&0\\-1&0&0\\0&0&1\end{pmatrix}, \begin{pmatrix}0&1&0\\0&0&-1\\-1&0&0\end{pmatrix},$$
$$\begin{pmatrix}0&0&1\\1&0&0\\0&1&0\end{pmatrix}, \begin{pmatrix}0&0&1\\0&1&0\\-1&0&0\end{pmatrix}, \begin{pmatrix}0&0&1\\-1&0&0\\0&-1&0\end{pmatrix}, \begin{pmatrix}0&0&1\\0&-1&0\\1&0&0\end{pmatrix},$$
$$\begin{pmatrix}-1&0&0\\0&1&0\\0&0&-1\end{pmatrix}, \begin{pmatrix}-1&0&0\\0&0&1\\0&1&0\end{pmatrix}, \begin{pmatrix}-1&0&0\\0&-1&0\\0&0&1\end{pmatrix}, \begin{pmatrix}-1&0&0\\0&0&-1\\0&-1&0\end{pmatrix},$$
$$\begin{pmatrix}0&-1&0\\1&0&0\\0&0&1\end{pmatrix}, \begin{pmatrix}0&-1&0\\0&0&1\\-1&0&0\end{pmatrix}, \begin{pmatrix}0&-1&0\\-1&0&0\\0&0&-1\end{pmatrix}, \begin{pmatrix}0&-1&0\\0&0&-1\\1&0&0\end{pmatrix},$$
$$\left.\begin{pmatrix}0&0&-1\\1&0&0\\0&-1&0\end{pmatrix}, \begin{pmatrix}0&0&-1\\0&1&0\\1&0&0\end{pmatrix}, \begin{pmatrix}0&0&-1\\-1&0&0\\0&1&0\end{pmatrix}, \begin{pmatrix}0&0&-1\\0&-1&0\\-1&0&0\end{pmatrix}\right\}$$

```
In[5]:= Map[Det, w]

Out[5]= {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}

In[7]:= Length[SpecialWeylGroup[5]] - 2^4 5!

Out[7]= 0
```

See also: WeylGroup, WMatrix, WeylGenerator, ModularGenerators, LongElement.

SubscriptedForm (suf)

This function takes a Mathematica expression and prints it out in such a way that subexpressions of the form $\mathbf{x[n_1, n_2, \cdots, n_j]}$, where the $\mathbf{n_i}$ are explicit integers, are printed in the style

$$\mathbf{x_{n_1,n_2,\cdots,n_j}}.$$

The value of this function is for improving the look of expressions for inspection and should not be used otherwise. Compare the Mathematica function MatrixForm. Not all expressions can be subscripted using this function.

$$\mathbf{SubscriptedForm[e]} \longrightarrow \mathbf{f}$$

e is a Mathematica expression,

f is a subscripted rendition of the same expression.

Example:

In[2395]:=
```
a = MakeMatrix["x", 4, 4]
```

Out[2395]=
```
{{x[1, 1], x[1, 2], x[1, 3], x[1, 4]}, {x[2, 1], x[2, 2], x[2, 3], x[2, 4]},
 {x[3, 1], x[3, 2], x[3, 3], x[3, 4]}, {x[4, 1], x[4, 2], x[4, 3], x[4, 4]}}
```

In[2396]:=
```
SubscriptedForm[a]
```

Out[2396]=
```
{{x1,1, x1,2, x1,3, x1,4}, {x2,1, x2,2, x2,3, x2,4},
 {x3,1, x3,2, x3,3, x3,4}, {x4,1, x4,2, x4,3, x4,4}}
```

In[2397]:=
```
MatrixForm[%]
```

Out[2397]//MatrixForm=

$$\begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} \\ x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} \\ x_{4,1} & x_{4,2} & x_{4,3} & x_{4,4} \end{pmatrix}$$

In[2398]:=
```
b = x[0, 2, -4] + f[y[1] - 3 * y[2]] / (x[1, 2, 3] + y[3]);
```

In[2399]:=
```
SubscriptedForm[b]
```

Out[2399]=

$$x_{0,2,-4} + \frac{f[y_1 - 3\,y_2]}{y_3 + x_{1,2,3}}$$

SwapMatrixRows (smr)

Two rows of a matrix are exchanged creating a new matrix and leaving the original unchanged.

SwapMatrixRows[a, i, j] ⟶ b

a is a matrix of CREs,

i is a valid row index for **a**,

j is a valid row index for **a**,

b is a matrix equal to **a** except the $\mathbf{i^{th}}$ and $\mathbf{j^{th}}$ rows have been exchanged.

See also: SwapMatrixColumns, ElementaryMatrix.

SwapMatrixColumns (smc)

Two columns of a matrix are exchanged creating a new matrix and leaving the original unchanged.

SwapMatrixColumns[a, i, j] ⟶ b

a is a matrix of CREs,

i is a valid column index for **a**,

j is a valid column index for **a**,

b is a matrix equal to **a** except the $\mathbf{i^{th}}$ and $\mathbf{j^{th}}$ columns have been exchanged.

See also: SwapMatrixRows, ElementaryMatrix.

TailingMatrixBlock (tmb)

This function returns a tailing matrix block of specified dimensions leaving the original matrix unchanged.

TailingMatrixBlock[a, i, j] ⟶ b

a is a matrix of CREs,

i is a positive integer less than the number of rows of **a**,

j is a positive integer less than the number of columns of **a**,

b is the tailing block of **a** with **i** rows and **j** columns.

See also: LeadingMatrixBlock, BlockMatrix.

UpperTriangleToMatrix (utm)

This function takes a list of lists of strictly decreasing length and forms a matrix with zeros in the lower triangle and with the given lists constituting the rows of the upper triangle. The length of the matrix is the length of the first sub-list. The last sub-list has length 1 and each successive sublist has length one less than the preceding sub-list.

UpperTriangleToMatrix[u] ⟶ a

u is a list of lists of CREs of decreasing length representing the elements of an upper triangular sub-matrix including the diagonal.

a is a full matrix with 0 in each lower-triangular position.

See also: MatrixUpperTriangle, LowerTriangleToMatrix.

VolumeFormDiagonal (vfd)

This function computes the differential volume form for the set of diagonal matrices

$$\bigwedge_{i=1}^{n} da_i,$$

where the product is the wedge product.
See Sections 1.4 and 1.5.

VolumeFormDiagonal[“a”, n] ⟶ Form

“a” is a string which will be the name of a one-dimensional array symbol,

n is a positive integer representing the dimension of the form,

Form is the diagonal volume form based on the variables **a**[**i**].

See also: VolumeFormGln, VolumeFormHn, VolumeFormUnimodular.

VolumeFormGln (vfg)

This function computes the differential volume form for the matrix group $GL(n, \mathbb{R})$ using the wedge product.
See Sections 1.4 and 1.5 and Proposition 1.4.3.

VolumeFormGln[“g”, n] ⟶ Form

“g” is a string which will be the name of a two-dimensional array symbol,

n is a positive integer representing the dimension of the matrices,

Form is the diagonal volume form based on the variables **g**[**i**, **j**].

Example:

```
In[62]:= VolumeFormGln[g, 2]
```

$$\text{Out[62]=}\quad \frac{d[g[1,1]] \wedge d[g[1,2]] \wedge d[g[2,1]] \wedge d[g[2,2]]}{(-g[1,2]\,g[2,1] + g[1,1]\,g[2,2])^2}$$

See also: VolumeFormHn, VolumeFormDiagonal, VolumeFormUnimodular.

VolumeFormHn (vfh)

This function computes the differential volume form for the generalized upper half plane.
See Definition 1.2.3 and Proposition 1.5.3.

VolumeFormHn[“x”, “y”, n] ⟶ Form

“x” is a string which will be the name of a two-dimensional array symbol,

“y” is a string which will be used as the name of a one-dimensional array symbol,

n is a positive integer representing the dimension of the matrices which appear in the Iwasawa decomposition,

Form is the volume form based on the variables $\mathbf{x}[\mathbf{i},\mathbf{j}]$, $\mathbf{y}[\mathbf{j}]$.

Example:

```
In[60]:= VolumeFormHn["x", "y", 3]
```

$$\text{Out[60]=}\ \frac{\mathrm{d}[x[1,\,2]] \wedge \mathrm{d}[x[1,\,3]] \wedge \mathrm{d}[x[2,\,3]] \wedge \mathrm{d}[y[1]] \wedge \mathrm{d}[y[2]]}{y[1]^3\, y[2]^3}$$

See also: VolumeFormGln, VolumeFormDiagonal, VolumeFormUnimodular.

VolumeFormUnimodular (vfu)

This function computes the differential volume form for the group of unimodular matrices, i.e. real upper-triangular with 1’s along the leading diagonal.
See Sections 1.4 and 1.5.

VolumeFormHn[“x”, n] ⟶ Form

“x” is a string being the name of an array symbol,

n is a positive integer representing the dimension of the matrices,

Form is the volume form based on the variables $\mathbf{x}[\mathbf{i},\mathbf{j}]$.

Example:

```
In[57]:= VolumeFormUnimodular[x, 3]
```

$$\text{Out[57]=}\ \mathrm{d}[x[1,\,2]] \wedge \mathrm{d}[x[1,\,3]] \wedge \mathrm{d}[x[2,\,3]]$$

See also: VolumeFormHn, VolumeFormDiagonal, VolumeFormGln.

VolumeBall (vbl)

This function computes the volume of an n-dimensional ball with given radius.

VolumeBall[r, n] ⟶ Vol

r is a CRE representing the radius of the ball,

n is a positive integer, being the dimension of the ball,

Vol is the **n**-dimensional volume of the ball.

Example:

```
In[342]:= {VolumeSphere[r, 3], VolumeSphere[r, 5]}

Out[342]= {4 π r^2, 8 π^2 r^4 / 3}

In[343]:= {VolumeBall[r, 3], VolumeBall[r, 5]}

Out[343]= {4 π r^3 / 3, 8 π^2 r^5 / 15}
```

See also: VolumeSphere, VolumeHn.

VolumeHn (vhn)

This function computes the volume of the generalized upper half-plane using the volume element **VolumeFormHn**.
See Example 1.5.2 and Proposition 1.5.3.

VolumeHn[n] ⟶ Vol

n is an integer with $\mathbf{n} \geq \mathbf{2}$ representing the order of the upper half-plane, being the size of the matrices appearing in the Iwasawa form,

Vol is a real number.

See also: VolumeBall, VolumeSphere.

VolumeSphere (vsp)

This function computes the n-dimensional volume of the sphere S^n in $\mathbb{R}^{n+1}$.

VolumeSphere[r, n] ⟶ Vol

r is a CRE being the radius of the sphere,

n is the dimension of the sphere,

Vol is the volume of the sphere computed using **n**-dimensional Lebesgue measure.

See also: VolumeBall, VolumeHn.

Wedge,d

This function computes the Wedge product of any finite number of functions or differential forms in an arbitrary number of dimensions. It works with the differential form operator **d**. Note that these functions have a different construction from others in GL (n)pack , and have limited error control. An alternative to the function **Wedge** is the infix operator which may be entered into Mathematica by typing a backslash, and open square bracket, the word "Wedge" and then a closing square bracket. It prints like circumflex, but is not the same. Note that wedge products of vectors are not currently supported
See Sections 1.4, 1.5, 5.6, 5.7 and 5.8.

$$\mathbf{Wedge}[\mathbf{f_1}, \mathbf{f_2}, ..., \mathbf{f_n}] \longrightarrow \mathbf{value}$$

$\mathbf{f_i}$ is an expression or a form,

value is the wedge product of the functions or forms $\mathbf{f_i}$.

Example:

In this example the function Wedge is used in conjunction with the differential form generator function **d**. Note that symbols, such as **a**, can be declared to be constant explicitly by setting, $\mathbf{d}[\mathbf{a}] = \mathbf{0}$.

```
In[64]:= Wedge[a d[x] + b d[y], u d[x] + v d[y]]

Out[64]= -b u d[x] ∧ d[y] + a v d[x] ∧ d[y]

In[65]:= d[a] = 0; d[b] = 0; d[c] = 0;

In[68]:= (a d[x] + b d[y]) ∧ (Exp[a x] d[y])

Out[68]= a ℮^(a x) d[x] ∧ d[y]

In[69]:= d[%]

Out[69]= 0

In[71]:= d[a y d[x] + b d[y] + c x d[z]]

Out[71]= -a d[x] ∧ d[y] + c d[x] ∧ d[z]

In[72]:= d[x^4]

Out[72]= 4 x^3 d[x]
```

See also: VolumeFormGln, VolumeFormHn.

WeylGenerator (wge)

This function returns a set of matrix generators for the Weyl subgroup of the group of integer matrixes with determiant ± 1, which consists of all matrices with exactly one ± 1 in each row and column. A single call returns a single generator. See Chapter VI.
Also see the manual entry for SpecialWeylGroup.

$$\mathbf{WeylGenerator[n,i,j]} \longrightarrow \mathbf{g}$$

n is a positive integer with $\mathbf{n} \geq \mathbf{2}$,

i is a positive integer with $\mathbf{i} \leq \mathbf{n}$,

j is a positive integer with $\mathbf{i} \neq \mathbf{j} \leq \mathbf{n}$,

g a matrix with 1's along the leading diagonal and zeros elsewhere, except in the $(\mathbf{i},\mathbf{j})^{\mathbf{th}}$ position where the value is -1 and $(\mathbf{j},\mathbf{i})^{\mathbf{th}}$ position where the value is 1 and where the corresponding $(\mathbf{i},\mathbf{i})^{\mathbf{th}}$ and $(\mathbf{j},\mathbf{j})^{\mathbf{th}}$ diagonal elements are 0.

Example:

```
In[168]:= MatrixForm[WeylGenerator[5, 2, 3]]

Out[168]//MatrixForm=
        ⎛1  0  0  0  0⎞
        ⎜0  0 -1  0  0⎟
        ⎜0  1  0  0  0⎟
        ⎜0  0  0  1  0⎟
        ⎝0  0  0  0  1⎠
```

See also: ModularGenerators, LongElement.

WeylGroup (wgr)

This function returns, for each whole number n, a list of all of the Weyl group of n by n permutation matrices.
See the proof of Proposition 1.5.3.

$$\mathbf{WeylGroup[n]} \longrightarrow \{\mathbf{m_1}, \mathbf{m_2}, \cdots, \mathbf{m_k}\}$$

n is a positive integer with $\mathbf{n} \geq \mathbf{1}$,

$\mathbf{m_j}$ is a matrix with a single 1 in each row and column.

Example:

```
In[5]:= Map[MatrixForm, WeylGroup[3]]

Out[5]= {⎛1  0  0⎞, ⎛1  0  0⎞, ⎛0  1  0⎞, ⎛0  1  0⎞, ⎛0  0  1⎞, ⎛0  0  1⎞}
        ⎜0  1  0⎟  ⎜0  0  1⎟  ⎜1  0  0⎟  ⎜0  0  1⎟  ⎜1  0  0⎟  ⎜0  1  0⎟
        ⎝0  0  1⎠  ⎝0  1  0⎠  ⎝0  0  1⎠  ⎝1  0  0⎠  ⎝0  1  0⎠  ⎝1  0  0⎠
```

See also: WeylGenerators, SpecialWeylGroup, LongElement.

Whittaker (wit)

This function computes a symbolic interated integral representatin of the generalized Jacquet Whittaker function $W_{Jacquet}$ (also written W_J) of order n, for $n \geq 2$, as defined by Equation 5.5.1. See Proposition 3.4.6, Section 3.4, Equation 5.5.1 and Equation 5.5.5. The algorithm uses the recursive representation of the Whittaker function defined by Stade [Stade, 1990, Theorem 2.1] related to that used in the book as follows. Let W_S and W_S^* be Stade's Whittaker and Whittaker starred functions respectively and let Γ represent the gamma factors for either form. Then

$$Q * W_S^* = \Gamma * W_J = W_J^* = W_S$$

where

$$\begin{aligned} Q &= I_\nu(y) \prod_{j=1}^{n-1} y_j^{-\mu_j}, \\ \mu_j &= \sum_{k=1}^{n-j} r_{j,k}, \\ r_{j,k} &= (\sum_{i=k}^{k+j-1} \frac{n\nu_i}{2}) - \frac{j}{2}. \end{aligned}$$

Whittaker[z, v, m, u] $\longrightarrow$ {coef, char, gam, value}

z is an $\mathbf{n} \times \mathbf{n}$ non-singular CRE matrix,

v is a list of $\mathbf{n-1}$ CREs,

m is a list of $\mathbf{n-1}$ CREs ($\mathbf{m_i}$) representing a character $\psi_{\mathbf{m}}(\mathbf{x}) = \mathbf{e}^{\mathbf{2\pi i}(\sum_{\mathbf{1 \leq i \leq n-1}} \mathbf{m_{n-i+1} x_{i,i+1}})}$,

u is a symbol which will be used to form the dummy variables in the iterated integral,

coef is the coefficient $\mathbf{c}_{\nu,\mathbf{m}}$ defined in Proposition 5.5.2,

char is the value of the character $\psi_{\mathbf{m}}(\mathbf{x})$ for $\mathbf{z} = \mathbf{x.y}$ the Iwasawa form,

gam is a product of terms as returned by **WhittakerGamma**,

value is a symbolic expression being the value of the Whittaker function at **My** where $\mathbf{z} = \mathbf{x.y}$ with parameters **v** and character $\psi_{\mathbf{1},\mathbf{1},\mathbf{1},..,\mathbf{1}}$. In this expression the K-Bessel function at a complex argument and parameter $\mathbf{K}_\nu(\mathbf{z})$, is represented by the noun function $\mathbf{K}[\nu, \mathbf{z}]$.

Example:

```
In[19]:= Whittaker[{{y, 0}, {0, 1}}, {v}, {m}, w]
```

$$\text{Out[19]= } \left\{\text{Abs}[m]^{-1+v},\ 1,\ \pi^{-v}\,\text{Gamma}[v],\ \frac{2\,\pi^{v}\,\sqrt{y}\,\sqrt{\text{Abs}[m]}\,K\left[-\frac{1}{2}+v,\ 2\,\pi\,y\,\text{Abs}[m]\right]}{\text{Gamma}[v]}\right\}$$

```
In[1]:= Whittaker[IdentityMatrix[6], {1, 1, 1, 1, 1}, {1, 1, 1, 1, 1}, u]
```

$$\text{Out[1]= } \left\{1,\ 1,\ \frac{4998263614010039534235859662310294189453125}{8192\,\pi^{92}},\right.$$

$$\left(268435456\,\pi^{92}\right.$$

$$\int_0^\infty\int_0^\infty\int_0^\infty\int_0^\infty\left(\int_0^\infty\int_0^\infty \frac{1}{u[1]\,u[2]}\left(K\left[\frac{19}{10},\ \frac{2\,\pi\,u[1]\,u[4]}{u[2]\,u[5]}\right]\right.\right.$$

$$K\left[\frac{57}{10},\ \frac{2\,\pi\sqrt{1+\frac{1}{u[1]^2}}\,u[3]}{u[4]}\right]K\left[\frac{57}{10},\right.$$

$$\left.\frac{2\,\pi\sqrt{\left(1+u[1]^2\right)\left(1+\frac{1}{u[2]^2}\right)}\,u[4]}{u[5]}\right]$$

$$\left.\left.K\left[\frac{57}{10},\ \frac{2\,\pi\sqrt{1+u[2]^2}\,u[5]}{u[6]}\right]\right)\mathbb{d}u[2]\,\mathbb{d}u[1]\right)K\left[\frac{25}{2},\right.$$

$$2\,\pi\sqrt{1+\frac{1}{u[3]^2}}\,\Big]\,K\left[\frac{25}{2},\ 2\,\pi\sqrt{\left(1+u[3]^2\right)\left(1+\frac{1}{u[4]^2}\right)}\,\right]$$

$$K\left[\frac{25}{2},\ 2\,\pi\sqrt{\left(1+u[4]^2\right)\left(1+\frac{1}{u[5]^2}\right)}\,\right]$$

$$K\left[\frac{25}{2},\ 2\,\pi\sqrt{\left(1+u[5]^2\right)\left(1+\frac{1}{u[6]^2}\right)}\,\right]$$

$$\left.K\left[\frac{25}{2},\ 2\,\pi\sqrt{1+u[6]^2}\,\right]\mathbb{d}u[6]\,\mathbb{d}u[5]\,\mathbb{d}u[4]\,\mathbb{d}u[3]\right)\Big/$$

$$\left.4998263614010039534235859662310294189453125\right\}$$

See also: WhittakerGamma.

WhittakerGamma (wig)

This function returns the gamma factors for the generalized Jacquet Whittaker function. See Definition 5.9.2. Note that although this definition differs from that in [Stade, 1990], the gamma factor that it represents is the same.

WhittakerGamma[v] ⟶ value

v is a list of $\mathbf{n-1}$ CREs which, if any are numerical, satisfy $\Re\mathbf{v_i} > \mathbf{1/n}$.

value is an expression, being the product of the gamma factors for the Whittaker function of order **n**.

See also: Whittaker.

WMatrix (wmx)

This function returns the so-called w-matrix, with $(-1)^{\lfloor n/2 \rfloor}$ in the $(1,n)^{th}$ position and 1 in every other reversed diagonal position, a member of SL(n,$\mathbb{Z}$).
See Section 5.5.

$$\mathbf{WMatrix[n]} \longrightarrow \mathbf{w}$$

n is a strictly positive integer with $\mathbf{n} \geq \mathbf{2}$.

w is an **n** by **n** matrix with each element 0, except the $(\mathbf{1},\mathbf{n})^{\mathbf{th}}$ which is $(-\mathbf{1})^{\lfloor \mathbf{n/2} \rfloor}$ and every $(\mathbf{i},\mathbf{n}-\mathbf{i}+\mathbf{1})^{\mathbf{th}}$ which is 1 for $\mathbf{2} \leq \mathbf{i} \leq \mathbf{n}$.

Example:

```
In[2]:= Map[MatrixForm, Map[WMatrix, {2, 3, 4, 5, 6}]]
```

$$\text{Out[2]=}\ \left\{\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}\right\}$$

See also: LongElement.

ZeroMatrix (zmx)

This function returns a zero matrix of given dimensions.

$$\mathbf{ZeroMatrix[m,n]} \longrightarrow \mathbf{Z}$$

m is a strictly positive integer representing the number of matrix rows,

n is a strictly positive integer representing the number of matrix columns,

Z is a zero matrix with **m** rows and **n** columns.

See also: ConstantMatrix.
