Mathematical Computing

IMT2b2β

Department of Mathematics
University of Ruhuna

A.W.L. Pubudu Thilan

Chapter 2

Handling Algebraic Expressions and doing Symbolic Computations

Symbolic vs numeric computation

- Results of symbolic computation are exact.
- Numeric computation computes with numeric approximations.
- That means results are not exact.

The sum and product calculations

Summation calculations

- The function **sum** is used for summation calculations.
- **sum**(**expr**, **i**, **i_0**, **i_1**) represents a summation of the values of **expr** as the index **i** varies from **i_0** to **i_1**.
- **sum** evaluates its summand **expr** and lower and upper limits **i_0** and **i_1**.
- **sum** does not evaluate the index **i**.
- The **simpsum** option simplifies the sum whenever possible.

Summation calculations

Examples

(i) $sum(i, i, 1, 100)$;

(ii) $sum(i^2, i, 1, 100)$;

(iii) $sum(a[i], i, 1, 10)$;

(iv) $sum(a(i), i, 1, 10)$;

(v) $sum(a(i), i, 1, n)$;

(vi) $sum(1/k, k, 1, n)$;

(vii) $sum(k, k, 1, n)$;

(viii) Simplyfy above.

(ix) $sum(1/k, k, 1, inf)$;

(x) Is it possible to simply above?

(xi) $sum(1/4^k, k, 1, inf)$;

(xii) Simplyfy above.

(xiii) $sum(3^k + k^2 + k, k, 0, n)$;

(xiv) Simplyfy above.

Product calculations

- The function **product** is used for product calculations.
- **product**(**expr**, **i**, **i_0**, **i_1**) represents a product of the values of **expr** as the index **i** varies from **i_0** to **i_1**.
- **product** evaluates **expr** and lower and upper limits **i_0** and **i_1**.
- **product** does not evaluate the index **i**.
- The **simpproduct** option simplifies the product whenever possible.

Product calculations

Examples

(i) $product(x + i, i, 1, 5);$

(ii) $product(x + k * (k + 1)/2, k, 1, 5);$

(iii) $product(k^3, k, 1, 10);$

(iv) $product(a[k], k, 1, 10);$

(v) $product(t(k), k, 0, 10);$

(vi) $product(t(k), k, 1, n);$

(vii) $product(t(k), k, 1, inf);$

(viii) $product(k, k, 1, n);$

The sets

Construct sets

- To construct the empty set, write **set();** or {};.
- We can use **set(...);** or { **...** }; to make non empty set.
- To construct a set with members **t_1**, ..., **t_n**, write **set**(**t_1**, ..., **t_n**); or {**t_1**, ..., **t_n**};.
- Sets are always displayed with curly braces.
- If a member is listed more than once, simplification eliminates the redundant member.

Construct sets

Examples

(i) $set();$

(ii) $set(a, b, c, d);$

(iii) $set(l, t, l);$

(iv) $set(a, b, c, set(d));$

(v) $set(a, b, c, set(b));$

(vi) $set(a, c, d, e, [b, c, e]);$

(vii) $\{\};$

(viii) $\{a, b, c, d\};$

(ix) $\{k, [l]\};$

(x) $\{l, t, l\};$

Functions for sets

adjoin (x, s);

- Returns the union of the set **s** with $\{x\}$.
- **adjoin** complains if **s** is not a literal set.
- **adjoin(x, s);** and **union(set(x), s);** are equivalent.
- However, **adjoin** may be somewhat faster than **union**.

Functions for sets

adjoin (x, s);⇒ Examples

(i) $adjoin(c, \{a, b, d, e\})$;

(ii) $adjoin(c, \{c, b, d, e\})$;

(iii) $union(set(c), \{a, b, d, e\})$;

(iv) $union(set(c), \{c, b, d, e\})$;

(v) $union(set(1), \{2, 3, 56, 1\})$;

Functions for sets

cardinality (s);

- Returns the number of distinct elements of the set **s**.
- **cardinality** ignores redundant elements.

Functions for sets

cardinality (s)⇒Examples

(i) $cardinality(\{\});$

(ii) $cardinality(\{a, b, c\});$

(iii) $cardinality(\{a, a, b, c\});$

(iv) $cardinality(set(a, b, c, d));$

Functions for sets

disjoin (x, s);

- Returns the set **s** without the member **x**.
- If **x** is not a member of **s**, return **s** unchanged.
- **disjoin** complains if **s** is not a literal set.
- **disjoin(x, s);**, **delete(x, s);**, and **setdifference(s, set(x));** are all equivalent.
- Of these, **disjoin();** is generally faster than the others.

Functions for sets

disjoin (x, s);⇒ Examples

(i) $disjoin(2, \{4, 5, 2, 9\});$

(ii) $disjoin(a, \{e, c, a, b\});$

(iii) $disjoin(e + f, \{4, e + f, \%pi, 45\});$

(iv) $disjoin(1, \{4, 5, 2, 9\});$

(v) $disjoin(e, \{4, e + f, \%pi, 45\});$

(vi) $delete(2, \{4, 5, 2, 9\});$

(vii) $setdifference(\{4, 5, 2, 9\}, set(2));$

Functions for sets

intersect(s_1, ..., s_n);

- Returns a set containing the elements that are common to the sets **s_1** through **s_n**.
- **intersection();** is the same as **intersect();**.
- **intersect();** complains if any argument is not a literal set.

Functions for sets

intersect(s_1, ..., s_n); $\Rightarrow$ Examples

Define following sets.
$S\_1 : \{a, b, c, d\}$;
$S\_2 : \{d, e, f, g\}$;
$S\_3 : \{c, d, e, f\}$;
$S\_4 : \{u, v, w\}$;

(i) Find common elements of S_1 and S_2.

(ii) Find common elements of S_2 and S_3.

(iii) Find common elements of S_1, S_2 and S_3.

(iv) Find common elements of S_1, S_2, S_3 and S_4.

Functions for sets

union(s_1, ..., s_n);

- Returns the union of the sets **s_1** through **s_n**.
- **union();** complains if any argument is not a literal set.

Functions for sets

union(s_1, ..., s_n); $\Rightarrow$ Examples

Define following sets.
$S\_1 : \{a, b, c, d\}$;
$S\_2 : \{d, e, f, g\}$;
$S\_3 : \{c, d, e, f\}$;
$S\_4 : \{u, v, w\}$;

(i) Find the union of S_1 and S_2.

(ii) Find the union of S_2 and S_3.

(iii) Find the union of S_1, S_2 and S_3.

(iv) Find the union of S_1, S_2, S_3 and S_4.

Functions for sets

Some other function on sets

disjointp (s, t);

- Returns **true** if and only if the sets **s** and **t** are disjoint.

elementp (a, s);

- Returns **true** if and only if **a** is a member of the set **s**.

emptyp (s);

- Return **true** if and only if **s** is the empty set or the empty list.

powerset(s);

- Returns the set of all subsets of the set **s** .

The Lists

What is a list?

- Lists are used to combine several items into a single object.
- They are displayed by enclosing its elements between square brackets.
- A list may even contain other lists as its entries.
- It may also consists of one or zero elements.

What is a list?

Examples

(i) $[\,]$;

(ii) $[a, b, c, d, e]$;

(iii) $[2, 3, 5, 6, 7, 23]$;

(iv) $[2, 3, 1, 8, [3, 5]]$;

(v) $l : [-3, 45, 57, 19]$;

(vi) $emtyList : [\,]$;

Accessing lists elements

- An element of a list can be accessed by its index enclosed in square brackets.
- The first element has index 1.
- The largest index must not exceed the length of the list.
- **length(L);** returns the length of list **L**.

Accessing lists elements

More commands on lists elements access

Command	**Description**
first(**L**);	First element of list **L**
second(**L**);	Second element of a list **L**
rest(**L**, **n**);	Returns **L** with its first **n** elements removed
last(**L**);	Last element of list **L**

Accessing lists elements

Examples

(i) Define a list with elements 2,5,7,9,3,6,23,11,14,[-22,45,19].

(ii) Assign the list to a variable L.

(iii) Find the lenght of L.

(iv) Access 3rd element of the list.

(v) Access first element of the list.

(vi) Access last element of the list.

(vii) Try an index out of the size of the list.

Lists elements manipulation

- To append elements of one list to another **append();** is used.
- To delete elements of a list **delete();** is used.

Lists elements manipulation

Examples

(i) Define a list with elements $-2, 3, 5, 1, 89, 34, [45, 67, 98], 31, 89$.

(ii) Assign the list to a variable **l**.

(iii) Append element 99 to the list. Try **append**(**l**, [**99**]);.

(iv) Append elements 67 and 78 to the list. Try **append**(**l**, [**67**, **78**]);.

(v) Remove firts element from the list. Try **delete**($-$**2**, **l**);.

Lists elements manipulation

Remark

- **append**(); and **delete**(); commands return a new list and do not modify the existing one.
- To modify original list one has to replace the original list.

Set vs lists

- Maxima treats lists and sets as distinct objects.
- This feature makes it possible to work with sets that have members that are either lists or sets.
- If you need to apply a set function to a list, use the **setify** function to convert it to a set.

Set vs lists

Examples

(i) $setify([2, 8, 4, f, w, 1])$;

(ii) $setify([a, b, c, d, e])$;

(iii) $setify([23, 45, 67, 89])$;

(iv) $setify([a, a, b, b, c, c])$;

(v) $setify([2, 2, 3, 3, 4, 4, 4, 4])$;

Expand, Factor, and Simplify

Evaluation of expressions in Maxima

- Simplification of expression is quite a difficult job for a computer algebra system even though there are established routines for standard simplification procedures.
- When Maxima evaluates expressions it usually does not perform all "obvious" expansions or simplifications.
- However, Maxima provides a couple of commands that try to do particular simplifications or transformations of an expression.

Expand expression

expand (expr);

- Expand expression **expr**.
- Products of sums and exponentiated sums are multiplied out.
- Numerators of rational expressions which are sums are split into their respective terms.
- Multiplication are distributed over addition at all levels of **expr**.

Expand expression

rateexpand (expr);

- For polynomials one should usually use **ratexpand();** which uses a more efficient algorithm.
- Command **ratexpand** also cancels out common divisors in rational expressions.

Expand expression

Examples

Expand following expansions.

(i) $(x-y)(x+y)$

(ii) $(x+2)^2$

(iii) $(x+1)^5$

(iv) $(x+3)^3(y+1)^5$

(v) $(x+y)^2/(x^2-y^2)$

(vi) $(x+y)^2/(x^2-y^2)$

Factor expression

factor(expr);

- Factors the expression **expr**, containing any number of variables or functions, into factors irreducible over the integers.

Factor expression

factor(expr);⇒Examples

Factor following expressions.

(i) $x^2 - y^2$

(ii) $x^2 + 4x + 4$

(iii) $(x+1)^5$

(iv) $x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1$

(v) $y^4 + 4xy^3 + 6x^2y^2 + 4x^3y + x^4$

(vi) $-8y - 4x + z^2(2y + x)$

(vii) $20!$

(viii) $2^{63} - 1$

Simplify expression

- Maxima provides commands to find equivalent but simpler expression for more complex one.
- Thus it applies several rules which embody conventional notions of simplicity.

Simplify expression

ratsimp();

- The command **ratsimp** tells maxima to simplify.
- However, Maxima by default is set to simplify rational functions and has been told to ignore algebraic (radical) simplifications in order to optimize the other types of simplifications it attempts.
- However, when we would like it to also perform algebraic simplifications we add the instruction **algebraic : true;**.

Simplify expression

ratsimp();⇒Examples

Simplify following expressions.

(i) $(x-2)/(x^2-4)$

(ii) $(x^2-y^2)/(x+y)^2$

(iii) $\sin(x/(x^2+x)) = \exp((\log(x)+1)^2 - \log(x)^2)$

(iv) $(x^2+4x+4)/(x+2)$

(v) $((x-1)^{(3/2)} - (x+1))\sqrt{(x-1)}\sqrt{(x-1)(x+1)}$

(vi) $1/(\sqrt{x}-2)$

(vii) Do simplification of $1/(\sqrt{x}-2)$ with **algebraic:true**.

Simplify expression

radcan(expr);

- The Maxima function **radcan(expr);** may be useful for simplifying expressions which contain logs, exponentials, and radicals.
- For some expressions **radcan** is quite time consuming.
- This is the cost of exploring certain relationships among the components of the expression for simpli cations based on factoring and partial fraction expansions of exponents.

Simplify expression

Functions to extract numerator and denominator

num(expr)

- Returns the numerator of **expr** if it is a ratio.
- If **expr** is not a ratio, **expr** is returned.

denom(expr)

- Returns the denominator of the rational expression **expr**.

Linear Algebra

Vectors

- Vectors are implemented in Maxima by means of lists.
- That is, their elements enclosed in square brackets **[...]**.
- Vector addition and multiplication with a scalar are performed by operators +, - and *.
- The scalar product is computed by the dot operator **.**.
- * performs a pointwise multiplication.
- The **k**-th element of a vector can be retrieved using **[k]**.

Vectors

Vectors ⇒ Examples

(i) [3,1,4]+[-1,5,9];

(ii) [4,-3,9]-[4,5,3];

(iii) 3*[4,2,7];

(iv) [2,0,5].[4,7,3];

(v) [3,4,3]*[6,7,1];

Matrices

- A **matrix** is a rectangular array of numbers, symbols, or expressions.
- The individual items in a matrix are called its **elements** or **entries**.
- Matrices are created by means of command **matrix** which takes the row vectors of the matrix as its arguments.
- These must have the same length.

Matrices

Operators used for matrices manipulation

Operators	Description
$\mathbf{A} + \mathbf{B}$	Sum of matrices **A** and **B**
$\mathbf{A} - \mathbf{B}$	Difference of matrices **A** and **B**
$\mathbf{s} * \mathbf{A}$	Multiply matrix **A** with scalar **s**
A.**B**	Product of matrices **A** and **B**
$\mathbf{A}\hat{}\hat{}\mathbf{n}$	**n**-th power of matrix **A**, i.e., **A**.**A**..**A**
$\mathbf{A}\hat{}\hat{}(-\mathbf{1})$	Inverse of matrix **A**

Matrices

Operations on matrices

- Matrices of the same size can be added or subtracted element by element.
- The rule for matrix multiplication is more complicated, and two matrices can be multiplied only when the number of columns in the first equals the number of rows in the second.
- $*$ and $\wedge$ perform a pointwise multiplication and exponentiation, respectively.

Matrices

Operations on matrices⇒Examples

(i) $A : matrix([2,1],[5,6]);$

(ii) $B : matrix([1,0],[7,3]);$

(iii) $A + B;$

(iv) $2 * A;$

(v) $A.B;$

(vi) $A\hat{}\hat{}3;$

(vii) $A\hat{}\hat{}(-1);$

(viii) $A * B;$

(ix) $A\hat{}2;$

(x) $B\hat{}5;$

Matrices

Retrieve or replace matrices elements

- The **i**-th rows of a matrix **M** can be retrieved by means of index [**i**].
- Element $\mathbf{a_{ij}}$ can be retrieved or replaced using index [**i**, **j**].

Matrices

Retrieve or replace matrices elements⇒Examples

(i) Define matrix **A** with rows [3,2],[1,5].

(ii) Retrieve the first row.

(iii) Retrieve the second row.

(iv) Retrieve the second element of the second row.

(v) Replace above element by 12.

(vi) Assign new matrix to **B**.

Matrices

More operations on matrices

Command	**Description**
diagmatrix(**n**, **x**)	Diagonal matrix of size **n** with diagonal element **x**
ident(**n**)	Identity matrix of order **n**
transpose(**A**)	Transpose of matrix **A**
invert(**A**)	Inverse of matrix **A** (same as **A**^^(−**1**))
determinant(**A**)	Determinant of matrix **A**
rank(**A**)	Rank of matrix **A**

Matrices

More operations on matrices⇒Examples

(i) Obtain Identity matrix of order 3.

(ii) Obtain diagonal matrix of size 3 with the diagonal element 6.

(iii) Define matrix **A** with rows [3,2],[1,4].

(iv) Obtain transpose of matrix **A**.

(v) Obtain inverse of matrix **A**.

(vi) Obtain determinant of matrix **A**.

(vii) Obtain rank of matrix **A**.

Matrices

Characteristic equation

- The characteristic equation of a matrix **A** is $|\mathbf{A} - \mathbf{I}.\mathbf{x}| = \mathbf{0}$, where **I** is the identity matrix.
- Command **charpoly(A,x)** returns the characteristic polynomial for matrix **A** with respect to variable **x**.

Matrices

Characteristic equation⇒Example

(i) Define matrix **A** with rows [1,2,1],[1,4,4],[7,3,5].

(ii) Obtain the corresponding characteristic equation by hand.

(iii) Use command **charpoly(A,x)** to obtain above characteristic equation.

Matrices

Eigenvalues

- The solutions to the characteristic equation are eigenvalues of the matrix.
- Eigenvalues can be computed by means of command **eigenvalues**.
- It returns a list of two elements.
- The first one is the list of eigenvalues while the second element is a list of their corresponding algebraic multiplicities.

Matrices

Eigenvalues⇒Example

(i) Define matrix **A** with rows [1,0],[6,6].

(ii) Obtain the corresponding characteristic equation.

(iii) Obtain eigenvalues using command **eigenvalues**.

Matrices

Eigenvectors

- If **A** is a square matrix, a non-zero vector **v** is an eigenvector of **A** if there is a scalar λ such that $\mathbf{Av}=\lambda\mathbf{v}$.
- The scalar λ is said to be the eigenvalue of **A** corresponding to **v**.
- An eigenspace of **A** is the set of all eigenvectors with the same eigenvalue together with the zero vector.
- However, the zero vector is not an eigenvector.

Matrices

Eigenvectors⇒Command to obtain eigenvectors

- Command **eigenvectors** computes both eigenvalues and the eigenvector.
- It returns a list of two elements: the result of eigenvalues and a list of the corresponding eigenspaces.
- Each eigenspace is represented by a list that contain its basis vectors ("eigenvectors").

Matrices

Eigenvectors⇒Command to obtain eigenvectors⇒Example

(i) Define matrix **A** with rows [1,0],[6,6].

(ii) Obtain eigenvectors using command **eigenvectors**.

Thank You