Multiplicities and tensor product coefficients for $A_\sigma$

Charles Cochet

To cite this version:

Charles Cochet. Multiplicities and tensor product coefficients for $A_\sigma$. 2003. hal-00000431

HAL Id: hal-00000431
https://hal.archives-ouvertes.fr/hal-00000431
Preprint submitted on 20 Jun 2003

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Multiplicities and tensor product coefficients for A_r

Charles Cochet*

20th June 2003

Abstract

We apply some recent developments of Baldoni-DeLoera-Vergne [1] on vector partition functions, to Kostant and Steinberg formulas, in the case of A_r. We therefore get a fast MAPLE program that computes for A_r: the multiplicity $c^\mu_\lambda$ of the weight $\mu$ in the representation $V(\lambda)$ of highest weight $\lambda$; the multiplicity $c^\nu_{\lambda,\mu}$ of the representation $V(\nu)$ in $V(\lambda) \otimes V(\mu)$. The computation also gives the locally polynomial functions $c^\mu_\lambda$ and $c^\nu_{\lambda,\mu}$.

1 Introduction

In this short note, we are interested in the two following problems in the case of $A_r$:

**Mult:** Computation of the multiplicity $c^\mu_\lambda$ of the weight $\mu$ in the representation $V(\lambda)$ of highest weight $\lambda$.

**Tens:** Computation of the multiplicity $c^\nu_{\lambda,\mu}$ of the representation $V(\nu)$ in the tensor product of representations of highest weights $\lambda$ and $\mu$.

The approach to these problems is through vector partition functions, namely number of integral points in lattice polytopes. More precisely, let $\Phi$ be a $n \times N$ integral matrix with column vectors $\Phi_1, \ldots, \Phi_N$. Fix a $n$-dimensional vector $a$. The rational convex polytope associated to $\Phi$ and $a$ is

$$P(\Phi, a) = \left\{ x \in \mathbb{R}^N : \sum_{i=1}^{N} x_i \Phi_i = a, x_i \geq 0 \right\}.$$  

We assume that $a$ is in the cone $C(\Phi)$ spanned by non-negative linear combinations of the vectors $\Phi_i$. We also assume that $\ker(\Phi) \cap \mathbb{R}_+^N = \{0\}$, so that the cone $C(\Phi)$ is acute. The vector partition function is then by definition

$$k(\Phi, a) = \left| P(\Phi, a) \cap \mathbb{N}^N \right|,$$  

that is the number of non-negative integral solutions $(x_1, \ldots, x_N)$ of the equation $\sum_{i=1}^{N} x_i \Phi_i = a$. If $\Phi$ is the matrix of positive roots of $A_r$, then $k(\Phi, a)$ is denoted by $k(A^+_r, a)$ and it is called Kostant partition function.

Let $\Sigma_{r+1}$ be the set of permutations of $(r+1)$ elements. This is the Weyl group of $A_r$. Kostant multiplicity formula asserts that

$$c^\mu_\lambda = \sum_w (-1)^{\epsilon(w)} k \left( A^+_r, w(\lambda + \rho) - (\mu + \rho) \right),$$  

where $\rho$ is half the sum of positive roots. Here, the sum is over the elements $w \in \Sigma_{r+1}$ such that $w(\lambda + \rho) - (\mu + \rho)$ is in the cone generated by non-negative combinations of positive roots. Moreover $\epsilon(w)$ is the signature of $w$.

Steinberg formula asserts that

$$c^\nu_{\lambda,\mu} = \sum_{w, w'} (-1)^{\epsilon(w)\epsilon(w')} k \left( A^+_r, w(\lambda + \rho) + w'(\mu + \rho) - (\nu + 2\rho) \right).$$  

Here, the sum is over couples $(w, w') \in \Sigma_{r+1} \times \Sigma_{r+1}$ such that $w(\lambda + \rho) + w'(\mu + \rho) - (\nu + 2\rho)$ is in the cone $C(A^+_r)$.

We use results of Baldoni-DeLoera-Vergne [1] and Baldoni-Vergne [2] on vector partition functions to obtain an efficient MAPLE program. Vector partition function is computed via inverse Laplace formula, involving iterated residues of rational functions.

Recall that LE program (see [3]) uses Freudenthal and Klymik formulas. The program LE is designed to work for any root system, while our program is designed specially for large parameters in $A_r$.

---

*cochet@math.jussieu.fr or http://www.math.jussieu.fr/~cochet
2 Baldoni-DeLoera-Vergne formula

Consider a \( r + 1 \) real dimensional vector space. Let \( A^+_r \) (the positive root system of \( A_r \)) be defined by

\[ A^+_r = \{(e_i - e_j) : 1 \leq i < j \leq (r + 1)\}. \]

Let \( E_r \) be the vector space spanned by the elements \((e_i - e_j)\). Then

\[ E_r = \{a \in \mathbb{R}^{r+1} : a = a_1e_1 + \cdots + a_r e_r + a_{r+1}e_{r+1} \text{ with } a_1 + a_2 + \cdots + a_r + a_{r+1} = 0\}. \]

The vector space \( E_r \) is of dimension \( r \) and the map

\[ f : \mathbb{R}^r \rightarrow E_r \]

defined by

\[ a = (a_1, a_2, \ldots, a_r) \mapsto a_1e_1 + \cdots + a_r e_r - (a_1 + \cdots + a_r)e_{r+1} \]

explicitly provides an isomorphism of \( E_r \) with the Euclidean space \( \mathbb{R}^r \). The hyperplane arrangement (setting \( z_{r+1} = 0 \)) generated by \( A^+_r \) is given by the following set of hyperplanes in \( \mathbb{C}^r \):

\[ \{z_i : 1 \leq i \leq r\} \cup \{z_i - z_j : 1 \leq i < j \leq r\}. \]

Let \( R_{A_r} \) be the set of rational functions \( f(z_1, z_2, \ldots, z_r) \) on \( \mathbb{C}^r \), with poles on the hyperplanes \( z_i = z_j \) or \( z_i = 0 \). For a permutation \( w \in \Sigma_r \) define the linear form on \( R_{A_r} \):

\[ \text{IRes}_{z \mapsto 0} f = \text{Res}_{z_{w(1)} = 0} \text{Res}_{z_{w(2)} = 0} \cdots \text{Res}_{z_{w(r)} = 0} f(z_1, z_2, \ldots, z_r) = \text{Res}_{z_{w(1)} = 0} \text{Res}_{z_{w(2)} = 0} \cdots \text{Res}_{z_{w(r)} = 0} f(z_{w^{-1}(1)}, z_{w^{-1}(2)}, \ldots, z_{w^{-1}(r)}). \]

In particular for \( w = \text{id} \) the linear form \( f \mapsto \text{IRes}_{z \mapsto 0} f \) defined by

\[ \text{IRes}_{z \mapsto 0} f = \text{Res}_{z_1 = 0} \text{Res}_{z_2 = 0} \cdots \text{Res}_{z_r = 0} f(z_1, z_2, \ldots, z_r) \]

is called the iterated residue.

Let \( C(A^+_r) \subset E_r \) be the cone generated by positive roots. A subset \( \sigma \) of \( A^+_r \) is called a basic subset if \( \{\sigma\} \) form a vector space basis of \( E_r \). The chamber complex is the polyhedral subdivision of the cone \( C(A^+_r) \) which is defined as the common refinement of the simplicial cones \( C(\sigma) \) running over all possible basic subsets of \( A^+_r \). The pieces of this subdivision are called chambers. See \( \text{IV} \) and \( \text{V} \) for the computation of chambers for \( A^+_r \).

A wall is a hyperplane in \( E_r \) spanned by \((r - 1)\) vectors of \( A^+_r \). A vector \( v \in C(A^+_r) \) is regular if it is not on a wall. This means that for every strict subset \( I \subset \{1, \ldots, r+1\} \) we have \( \sum_{i \in I} v_i \neq 0 \).

Let \( \text{Sp}(a) \) be the set of permutations \( w \in \Sigma_r \) such that:

\[
\begin{align*}
\text{if } a_{w(1)} &\geq 0 & \text{then } w(1) < w(2) & \text{else } w(1) > w(2) \\
\text{if } a_{w(1)} + a_{w(2)} &\geq 0 & \text{then } w(2) < w(3) & \text{else } w(2) > w(3) \\
& \quad \cdots & & \\
\text{if } a_{w(1)} + \cdots + a_{w(i)} &\geq 0 & \text{then } w(i) < w(i+1) & \text{else } w(i) > w(i+1) \\
& \quad \cdots & & \\
\text{if } a_{w(1)} + \cdots + a_{w(r-1)} &\geq 0 & \text{then } w(r-1) < w(r) & \text{else } w(r-1) > w(r)
\end{align*}
\]

An element of \( \text{Sp}(a) \) will be called a special permutation for \( a \). Remark that if \( a_i \geq 0 \) for all \( i \leq r \), then \( \text{Sp}(a) = \{\text{id}\} \). Also remark that \( \text{Sp}(a) \) is a subset of the subgroup \( \Sigma_r \) of the Weyl group \( \Sigma_{r+1} \) of \( A_r \).

Given \( a \in C(A^+_r) \cap \mathbb{Z}^{r+1} \), define \( \text{def}(a) = a + \varepsilon(\sum_{i=1}^r e_i - re_{r+1}) \) with \( \varepsilon = \frac{1}{2r} \).

**Lemma 2.1** The deformed vector \( \text{def}(a) \) verifies:

- \( \text{def}(a) \) is regular.
- \( a \in C(A^+_r) \) if and only if \( \text{def}(a) \in C(A^+_r) \).

**Proof:**

- Let \( I_a \) be a strict subset of \( \{1, \ldots, r+1\} \) such that \( \sum_{i \in I_a} \text{def}(a)_i = 0 \).

  First, assume that \( r + 1 \not\in I_a \). We can re-index \( a \) in order to get \( I_a = \{1, \ldots, k\} \) with \( k \leq r \). Thus \((a_1 + \varepsilon) + \cdots + (a_k + \varepsilon) = 0 \) means that the integer \( a_1 + \cdots + a_k \) equals \( -\frac{k}{2r} \). But \( 0 < k \leq r \) implies \( 0 < \frac{k}{2r} < 1 \), contradiction.

  Now, assume that \( r + 1 \in I_a \). We can also assume that \( I_a = \{1, \ldots, k, r+1\} \) with \( k \leq r \). By definition \((a_1 + \varepsilon) + \cdots + (a_k + \varepsilon) + (a_{r+1} - re) = 0 \), therefore the integer \( a_1 + \cdots + a_k + a_{r+1} \) is equal to \( \frac{k}{2r} \). But \( k \leq r \) leads to \(-\frac{1}{2} < \frac{k}{2r} < \frac{1}{2} \), hence \( k = r \). Consequently \( I_a = \{1, \ldots, r+1\} \), contradiction.

- Note that the coordinates \( a_i \) of \( a \) are integers. Now the integer \( a_1 + \cdots + a_i + \frac{1}{2r} \) is non-negative, because \( 0 < \frac{1}{2r} < 1 \). Hence \( a \in C(A^+_r) \) is equivalent to \( \text{def}(a) \in C(A^+_r) \).
Now we can state the formula that was implemented:

**Theorem 2.2 (Baldoni-DeLoera-Vergne)** For \( a \in C(A^+_z) \cap \mathbb{Z}^{r+1} \), the Kostant partition function is given by:

\[
k(A^+_z, a) = \sum_{w \in \text{Sp}(a')} (-1)^{\ell(w)} \text{Res}_z \left( \frac{(1+z_1)^{n_1+r-1}(1+z_2)^{n_2+r-2}\cdots(1+z_r)^{n_r}}{z_1\cdots z_r \prod_{1 \leq i < j \leq r}(z_i - z_j)} \right)
\]

where \( a' = \begin{cases} \text{def}(a) & \text{if } a \text{ is regular,} \\ \text{otherwise.} & \end{cases} \)

In particular, if \( a_i \geq 0 \) for \( 1 \leq i \leq r \), we have

\[
k(A^+_z, a) = \text{Res}_{z_1=0} \text{Res}_{z_2=0} \cdots \text{Res}_{z_r=0} \left( \frac{(1+z_1)^{n_1+r-1}(1+z_2)^{n_2+r-2}\cdots(1+z_r)^{n_r}}{z_1\cdots z_r \prod_{1 \leq i < j \leq r}(z_i - z_j)} \right).
\]

### 3 Deus ex machina

This section features a brief description of the algorithms that were implemented with the software MAPLE. This program is available at [http://www.math.jussieu.fr/~cochet](http://www.math.jussieu.fr/~cochet)

#### 3.1 How to use the program

The initial data are only vectors: two for computing the multiplicity \( c^\lambda_\mu \), three for computing the tensor product coefficient \( c^\lambda_\mu_\rho \).

Our program works with weights represented in the canonical basis of \( \mathbb{R}^{r+1} \), and not fundamental weights basis of \( A_r \) like \( \mathbb{P}E \). The translation between these two approaches is performed via the procedures fundamental and fundamentalInverse. For example fundamental([2,1,3]) returns [1,4].

Therefore computing the multiplicity \( c^\lambda_\mu \) is done by typing in multiplicity(lambda, mu) where \( \lambda \) and \( \mu \) are lists of \( r+1 \) rationals such that \( \sum_{i=1}^{r+1} \lambda_i = \sum_{i=1}^{r+1} \mu_i \) and \( \lambda_i - \lambda_{i+1} \in \mathbb{N}, \mu_i - \mu_{i+1} \in \mathbb{Z} \).

For computing the tensor product coefficient \( c^\lambda_\mu_\rho \), the syntax is tensor_product(lambda, mu, nu) where \( \lambda, \mu \) and \( \nu \) are lists of \( r+1 \) rationals such that \( \sum_{i=1}^{r+1} (\lambda_i + \mu_i) = \sum_{i=1}^{r+1} \nu_i \) and \( \lambda_i - \lambda_{i+1} \in \mathbb{N}, \mu_i - \mu_{i+1} \in \mathbb{N}, \nu_i - \nu_{i+1} \in \mathbb{N} \).

In the examples, we use the vector \( \theta = re_1 + (r-1)e_2 + \cdots + 1e_{r-1} - \frac{r(r+1)}{2}e_{r+1} \). Its decomposition in the fundamental weights basis is the \( r \)-dimensional vector \((1,\ldots,1,1+r(r+1)/2)\).

### 3.2 Implementation

The elements we need to compute are:

1. The vector \( a' = \text{def}(a) \) obtained by deforming the initial parameter \( a \).
2. The residues that appear in theorem 2.4.
3. The two sets of permutations that appear in Kostant and Steinberg formulas (see (1) and (2))
4. The set of special permutations \( \text{Sp}(\text{def}(a)) \).

Because of lemma 2.1, we may use \( \text{def}(a) \) instead of \( a \) and we do this to simplify the procedures. We compute the vector \( \text{def}(a) \) via the straightforward MAPLE procedure defvector. This takes care of the first part.

Computation of residues is done iteratively. The function \( F \) which reside we need to compute is a product of a certain number of functions. This allows to take the residues by introducing little by little the part of the function \( F \) containing the needed variable. See a detailed explanation of this procedure in [1].

Let \( u, v \in E' \). A valid permutation for \( u \) and \( v \) is a permutation \( w \in \Sigma_{r+1} \) such that \( w(u) - v \in C(A^+_z) \). We denote by \( V(u,v) \) the set of valid permutations for \( u, v \). Hence, Kostant formula for \( A_r \) rewrites as

\[
c^w_\lambda = \sum_{w \in V(\lambda, \mu) \cap \Sigma_{r+1}} (-1)^{\ell(w)} k(A^+_z, w(\lambda + \rho) - (\mu + \rho)).
\]

Given a set of chambers \( \{C_\rho\}_{\rho \in V(\lambda, \mu)} \) of \( C(A^+_z) \), it follows from [1] that \( c^w_\lambda \) is polynomial when \( w: \lambda - \mu \in \mathbb{P}E \), for \( w \in V(\lambda, \mu) \). In particular, the function \( N \mapsto c^w_{N^\lambda, N^\mu} \) is a polynomial in \( N \) of degree less or equal to \( \frac{r(r+1)}{2} \).

Let us explain our implementation with the symbolic language MAPLE of the procedure valid_permutations designed to find the set \( V(u,v) \). The method is quite simple: we build the permutations iteratively. This allows us not examining all permutations and saving much time. Recall that we have to find all permutations \( w \)'s such that \( u_{w(1)} \geq v_1, u_{w(1)} + u_{w(2)} \geq v_1 + v_2, \) etc. For any sequence \( x \) of indices, we denote by \( u_x \) the sum \( \sum_{i \in x} u_i \).
Step 1. Let \( X \) be the set of all indices \( i \) such that \( u_i \geq v_1 \).

Step 2. For each \( x \in X \), we find all indices \( i_x \) such that \( u_x + u_{i_x} \geq v_x + v_{i_x} \). Let \( X_{\text{new}} \) be the set of such \([x, i_x]\), for all \( x \) and \( i_x \). Then \( X \leftarrow X_{\text{new}} \).

We repeat \( r \) times step 2, and obtain the list \( X \) of \((r+1)\)-uples representing permutations of 1, \( \ldots \), \( r+1 \). The second step is treated in the procedure next\_index\_valid\_permutations. The procedure valid\_permutations contains first step and a for \( \ldots \) do loop executing \( r \) times step 2.

**Remark 3.1** We reduce computing time by using the following three tracks.

1. We compute once and for all the vector \( v' = [v_1, v_1 + v_2, \ldots, v_1 + v_r + v_{r+1}] \).

2. We build at the same time of \( X = \{[i_1, \ldots, i_p], \ldots, \{\text{other sets of indices}\}\} \) the set \( S_X \) of partial sums associated to each \([i_1, \ldots, i_p]\). More precisely \( S_X = [u_{i_1} + \ldots + u_{i_p}, \ldots, \{\text{other partial sums}\}] \).

3. We use tables instead of lists.

Now let us examine the couples of permutations involved in Steinberg formula. Let \( u_1, u_2, v \in E^r \). A valid couple of permutations for \( u_1, u_2 \) and \( v \) is a couple \((u_1, u_2) \in \Sigma_{r+1} \times \Sigma_{r+1} \) such that \( w_1(u_1) + w_2(u_2) - v \in C(A^r) \). We denote by \( V(u_1, u_2, v) \) the set of valid couples of permutations for \( u_1, u_2 \) and \( v \). Hence Steinberg formula rewrites as

\[
\begin{align*}
\sum_{(w, w') \in V} (1)^{c(w) - c(w')} k(A^t, w(\lambda + \rho) + w'(\mu + \rho) - (\nu + 2\rho)).
\end{align*}
\]

The procedure computing valid couples of permutations is similar to the former.

To compute the subset \( Sp(s) \) of \( \Sigma_r \), we use the procedure special\_permutations. This procedure is very similar to the previous one. We stress that the MAPLE function combin\_permute is impractical and does not go very far because of memory limitations.

## 4 Test of the program

Let \( \theta \) be the \( r \)-dimensional vector \((1, \ldots, 1, 1 + r(r+1)/2) \) (fundamental weights decomposition in \( A_r \)). It translates as \((r, r - 1, \ldots, 1, -r + r(r+1)/2) \) in the canonical basis of \( \mathbb{R}^{r+1} \). We used this vector to check the well-known fact that the multiplicity of the weight 0 in the representation of \( A_r \) of highest weight \( N\theta \) is given by the dimension of the representation of \( A_{r-1} \) of highest weight \( N\rho \), which is \((N + 1)^{(r-1)/2}\).

In this test, we compute for various \( A_r \) (\( r = 1, \ldots, 8 \)):

- \( c^\nu_\lambda \) with \( \lambda = N\theta \) and either \( \mu = 0 \) (worst case), or \( \mu = \lfloor 9N/10 \rfloor \theta \) (intermediate case).
- \( c^\nu_\lambda \) with \( \lambda = \mu = N\theta \) and either \( \nu = 0 \) (worst case), or \( \nu = \lfloor 9N/10 \rfloor \theta \) (intermediate case).

Tests were made with bi-processor PIII 1,13GHz. The notation "-" in an array means that we did not try the computation (and not that computation failed).

Recall (see for example \( \text{A} \) and \( \text{B} \)) that counting integral points in a lattice polytope is polynomial in the size of input if dimension is fixed, and NP-hard if dimension is not fixed. The figures \( \text{A} \) and \( \text{B} \) emphasize this result.

In figure \( \text{B} \) the letter I stands for intermediate case \((\mu = \lfloor 9N/10 \rfloor \theta \) for \( c^\nu_\lambda \) and \( \nu = 2\lfloor 9N/10 \rfloor \theta \) for \( c^\nu_{\lambda, \mu} \)), while \( W \) stands for worst case \((\mu = 0 \) for \( c^\nu_\lambda \) and \( \nu = 0 \) for \( c^\nu_{\lambda, \mu} \)).

<table>
<thead>
<tr>
<th></th>
<th>( N = 10^3 )</th>
<th>( N = 10^4 )</th>
<th>( N = 10^4 )</th>
<th>( N = 10^4 )</th>
<th>( N = 10^4 )</th>
<th>( N = 10^4 )</th>
<th>( N = 10^4 )</th>
<th>( N = 10^4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c^\nu_\lambda, I, A_7 )</td>
<td>12.5 s</td>
<td>16.0 s</td>
<td>17.5 s</td>
<td>18.7 s</td>
<td>17.63 s</td>
<td>19.4 s</td>
<td>20.3 s</td>
<td>21.3 s</td>
</tr>
<tr>
<td>( c^\nu_\lambda, W, A_7 )</td>
<td>204.9 s</td>
<td>221.0 s</td>
<td>235.6 s</td>
<td>251.5 s</td>
<td>259.1 s</td>
<td>261.5 s</td>
<td>283.8 s</td>
<td>297.0 s</td>
</tr>
<tr>
<td>( c^\nu_{\lambda, \mu}, I, A_6 )</td>
<td>40.5 s</td>
<td>47.0 s</td>
<td>50.3 s</td>
<td>52.6 s</td>
<td>53.8 s</td>
<td>57.0 s</td>
<td>58.4 s</td>
<td>59.9 s</td>
</tr>
<tr>
<td>( c^\nu_{\lambda, \mu}, W, A_4 )</td>
<td>13.5 s</td>
<td>13.7 s</td>
<td>13.8 s</td>
<td>14.0 s</td>
<td>14.1 s</td>
<td>14.4 s</td>
<td>15.1 s</td>
<td>15.2 s</td>
</tr>
</tbody>
</table>

Figure 1: Time of computation, when size of input grows.
The computation can also be done with parameters, giving \((N + 1)^{r(r-1)/2}\) as expected.

<table>
<thead>
<tr>
<th>Algebra</th>
<th>Time</th>
<th>Multiplicity (c_0^r)</th>
<th>Time</th>
<th>Polynomial (N \mapsto c_0^N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A_2)</td>
<td>&lt; 0.1 s</td>
<td>2 = 2^1</td>
<td>&lt; 0.1 s</td>
<td>((N + 1)^1)</td>
</tr>
<tr>
<td>(A_3)</td>
<td>&lt; 0.1 s</td>
<td>8 = 2^3</td>
<td>&lt; 0.1 s</td>
<td>((N + 1)^3)</td>
</tr>
<tr>
<td>(A_4)</td>
<td>&lt; 0.1 s</td>
<td>64 = 2^6</td>
<td>&lt; 0.1 s</td>
<td>((N + 1)^6)</td>
</tr>
<tr>
<td>(A_5)</td>
<td>0.4 s</td>
<td>1024 = 2^{10}</td>
<td>1.4 s</td>
<td>((N + 1)^{10})</td>
</tr>
<tr>
<td>(A_6)</td>
<td>7.6 s</td>
<td>32768 = 2^{15}</td>
<td>36.2 s</td>
<td>((N + 1)^{15})</td>
</tr>
<tr>
<td>(A_7)</td>
<td>169.3 s</td>
<td>2097152 = 2^{21}</td>
<td>2091 s</td>
<td>((N + 1)^{21})</td>
</tr>
<tr>
<td>(A_8)</td>
<td>9401 s</td>
<td>268435456 = 2^{28}</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Figure 2: Multiplicity of 0 in \(V(N\theta)\) when rank increases

Acknowledgements: Marc A. A. van Leeuwen explained me his software \(\text{LiE}\) and its internal mechanisms. Moreover, his competence in \(\text{MAPLE}\) has deeply influenced my method of programing.

References


