Orthogonality Properties of Demazure Characters

May 21, 2004

Chers JYT et BL, il serait bienvenu que LLT ait encore une publication à son actif. Je n'avais pas encore eclairci le produit scalaire approprie aux caracteres de Demazure, ce devrait etre le sujet de cette note, en ajoutant des modules de Verma si le coeur vous en dit. J'ai besoin des caracteres K et de leur 'top component' $\widehat{K}$' (calculee avec les $\widehat{\pi}_i$ au lieu de π_i; il faudrait trouver une terminologie).

1 Required

$\mathfrak{Pol}$= space of Laurent polynomials in $x_1, \ldots, x_n$

Reversing vectors: $u \to u^{\approx} := [u_n, \ldots, u_1]$

Isobaric divided differences π_i, $\widehat{\pi}_i := \pi_i - 1$

Maximal isobaric divided difference π_ω

Involution $\clubsuit$: $x_i \to 1/x_{n+1-i}$

Conjugation:

$$\clubsuit\, \pi_i \,\clubsuit = \pi_{n+1-i}$$

Corollary:

$$\clubsuit\, \pi_\sigma \,\clubsuit = \pi_{\omega\, \sigma\, \omega}$$

Kernel. Introduce $y_1, \ldots, y_n$ and define

$$\Omega := 1/ \prod_{i+j \le n+1} (1 - x_i y_j)$$

Demazure characters (key polynomials) recursively defined :

For every decreasing partition λ (dominant case):

$$K_\lambda(\mathbb{A}) = \widehat{K}_\lambda(\mathbb{A}) := \cdots a_3^{\lambda_3} a_2^{\lambda_2} a_1^{\lambda_1} \ , \tag{1}$$

and for every $v \in \mathbb{N}^n$, for every i such that $v_i > v_{i+1}$, then

$$K_{vs_i}(\mathbf{x}) := K_v(\mathbf{x})\,\pi_i \qquad \& \qquad \widehat{K}_{vs_i}(\mathbf{x}) := \widehat{K}_v(\mathbf{x})\,\widehat{\pi}_i\ . \tag{2}$$

2 Scalar product

Add an infinity of variables $x_{n+1}, x_{n+2}, \ldots$ and define $\pi_{\aleph}$ to be the operator on $\mathfrak{Pol}$ (with values in $\mathfrak{Sym}$ = symmetric functions in an infinity of variables) which is the limit of $\pi_{N\ldots 1}$, $N \to \infty$.

Definition. Scalar product : Given two polynomials f, g of n variables, then

$$(f,g) := \text{ Constant Term of } \ f(\mathbf{x})g(\mathbf{x}^{\clubsuit}) \prod_{1\le i<j\le n} (1 - x_i/x_j) \tag{3}$$

Notice that the scalar product is symmetrical in its arguments, though its expression is not.

$$(f,g) := f(\mathbf{x})\, g(\mathbf{x}^{\clubsuit})\, \pi_{\aleph}\,.$$

Proposition. Given two polynomials f, g of n variables, then

$$(f,g) := f(\mathbf{x})\, g(\mathbf{x}^{\clubsuit})\, \pi_{\aleph}\,.$$

Proof. Test on two monomials x^u, x^v of the same degree. Since the image of a monomial under $\pi_{N\ldots 1}$ is $\pm$ a Schur function, or 0, up to a power of $x_1..x_N$, one sees that the scalar product of x^u, x^v is 0 or $\pm S_{0..0}$. It is non-zero iff there exists a permutation σ (considered here as a vector) such that

$$[u_1 - v_n, \ldots, u_n - v_1] = [1, \ldots, n] - \sigma\ .$$

In that case $(x^u, x^v) = (-1)^{\ell(\sigma)}$

L'ecriture du produit scalaire en terme de $\pi_{\aleph}$ est utile pour le calcul, parceque $\pi_i \pi_{\aleph} = \pi_{\aleph}$.

Lemma. For any $f \in \mathfrak{Sym}(n)$, then

$$(\sigma_1(\mathbf{xy})\,,\ f(\mathbf{x})) = f(\mathbf{y}) \tag{4}$$

and therefore, the restriction of the scalar product to symmetric functions is the usual scalar product.

Its extends Weyl's expression of the scalar product on $\mathfrak{Sym}(n)$:

$$\text{Constant Term of } \ (-1)^{\binom{n}{2}}\, f(\mathbf{x})\Delta(\mathbf{x})\, g(x^{\clubsuit})\Delta(\mathbf{x}^{\clubsuit})\ . \tag{5}$$

```
# Extends Weyl's scalar product
# two homogeneous polynomials of the same degree  $dg$
#           vand := (1-x1/x2)(1-x1/x3)(1-x2/x3)
# Instead of computing the constant term of a Laurent pol, one multiplies by
# (x1..x.n)^k  and computes the coefficient of this monomial
ScalWeyl:=proc(pol1,pol2,dg,card)  local i,j,k,vand,res;
 k:=dg+card*(card-1)/2;
 vand:=convert([seq(seq(1-cat(x,i)/cat(x,j),j=i+1..card),i=1..card-1)],`*`);
 res:=expand(
  convert([seq(cat(x,i)^k,i=1..card)],`*`)*pol1*vand
      *subs({seq(cat(x,i)=1/cat(x,card+1-i),i=1..card)},pol2));
 for i from 1 to card do
   res:=coeff(res,cat(x,i),k)
 od;
 res
end:
TestMonom:=proc(u)  local mon,dg,n,lc,v,i;
   dg:=convert(u,`+`);
   n:=nops(u);
   mon:=convert([seq(cat(x,i)^u[i],i=1..n)],`*`);
   lc:=ListCompo(dg,'allowzeros','lg'=n);
     for v in lc do
      lprint(v,[seq(u[i]-v[n+1-i],i=1..n)],
      ScalWeyl(mon, convert([seq(cat(x,i)^v[i],i=1..n)],`*`),dg,n))
     od;
end:
ACE> TestMonom([1,2]);
[3, 0]   [1, -1]   0
[2, 1]   [0, 0]   1
[1, 2]   [-1, 1]   -1
[0, 3]   [-2, 2]   0
ListIncrCodes:=proc(dg,lg)  local u,i;
 [seq([0$(lg-nops(u)),seq(u[nops(u)+1-i],i=1..nops(u))],
             u= ListPart(dg,maxlg=lg))]
end:
TestSchurWeyl:=proc(dg,lg)  local lc,u,pol; # restriction to Schur f.
 lc:=ListIncrCodes(dg,lg);
 for u in lc do
   pol:=Tox(Y[op(u)]);
   for v in lc do
    if ScalWeyl(pol,Tox(Y[op(v)]),dg,lg)<>0 then
```

```
     lprint(u,v)   fi;
   od;
  od;
end:
ACE> TestSchurWeyl(4,3);
[0, 0, 4]   [0, 0, 4]
[0, 1, 3]   [0, 1, 3]
[0, 2, 2]   [0, 2, 2]
[1, 1, 2]   [1, 1, 2]
```

Define $\mathbb{B} := \mathbf{y}^{\clubsuit} = \{y_n^{-1}, \ldots, y_1^{-1}\}$, i.e. $b_i = 1/x_{n+1-i}$, and $\mathcal{N} := \prod_{1 \le i < j \le n}(x_j - b_i)$. For any $k \le n$, let $\mathbb{B}_k := \{b_1, \ldots, b_k\}$.

Lemma. For any $v \in \mathbb{N}^n$, then

$$\mathcal{N}\, x^v\, \pi_\omega = S_{v^{\approx} + \rho}(\mathbf{x} - \mathbb{B}_{n-1}, \ldots, \mathbf{x} - \mathbb{B}_0)$$

Preuve facile en ecrivant le produit d'un monome par $\mathcal{N}$ comme une fonction de Schur.

Since $\Omega = \pm y^{01\cdots}\, \mathcal{N}\, \sigma_1(\mathbf{xy})$, one deduces from the lemma the following theorem when $f(\mathbf{x})$ is a monomial, and therefore for any $f(\mathbf{x})$:

Theorem. Ω is a reproducing kernel with respect to the scalar product, i.e.

$$(\Omega, f(\mathbf{x})) = f(\mathbf{y})$$

Using the decomposition of Ω as the sum

$$\sum K_v(\mathbf{x})\, \widehat{K}_{v^{\approx}}(\mathbf{y})$$

given in "Double Crystal", one gets the following property of Demazure's characters.

Corollary

$$(K_v, \widehat{K}_{u^{\approx}}) = \delta_{u,v}\ .$$

Orthogonality Properties of Key Polynomials for the classical groups

Desole pour les misprints. Pour eviter les erreurs d'indices, je donne les programmes, et utilise $y_1, ..., y_n$ au lieu de $y_1, ..., y_{2n}$ ou $y_1, ..., y_{2n+1}$.

Je traite ci-dessous les types A,B,C.

1 Type A

Key polynomials = images of dominant monomials under all products of π_i (resp. $\widehat{\pi}_i$). Indexed by vectors in $\mathbb{N}^n$ and denoted K_v and $\widehat{K}_v$.

Scalar product:

$$(f, g) := \text{ Constant Term of } f(\mathbf{x})g(\mathbf{x}^{\clubsuit}) \prod_{1 \le i < j \le n} (1 - x_i/x_j) \ , \qquad (1)$$

using the involution $\clubsuit : x_i \to 1/x_{n+1-i}$

```
# terme constant de prod_{i<j}(1-x.i/x.j) pol1(x1,x2..) pol2(1/x.n ...1/x1)
#  4 eme argument optionnel= nom des variables
# produit scalaire pour lequel K[v] est orthogonal a \Khat[v^\omega]
CT:=proc(n,pol1,pol2)  local i,j,res,eul,b;
 if nargs>3 then b:=args[4]
 else b:=`x`
 fi;
 eul:=convert([seq(seq(1-cat(b,i)/cat(b,j),j=i+1..n),i=1..n-1)],`*`);
 res:=pol1*eul*convert([seq(1/cat(b,i),i=1..n)],`*`)*
      subs({seq(cat(b,i)=1/cat(b,n-i+1),i=1..n)},pol2);
 for i from 1 to n do
   res:=residue(res,cat(b,i)=0);   #  print(factor(res));
```

```
 od;
 res
end:
```

Reproducing kernel: Introduce $y_1, \ldots, y_n$

$$\Omega := 1/\prod_{i+j\leq n+1}(1 - x_i y_j)$$

Theorem. Ω is a reproducing kernel with respect to the scalar product, i.e.

$$(\Omega, f(\mathbf{x})) = f(\mathbf{y})$$

and

$$(K_v, \widehat{K}_{u^{\approx}}) = \delta_{u,v} \ .$$

with $u^{\approx} := [u_n, \ldots, u_1]$

```
NoyauA:=proc(n)  local i,j;
 1/convert([seq(seq(1-cat(x,i)*cat(y,j),i=1..n+1-j),j=1..n)],`*`)
end:
ACE> CT(3,NoyauA(3), x1*x3^4);
                                        4
                                     y3  y1

ACE> CT(3,K2x(a*K[2,0,3]+b*K[0,2,1]+c*K[1,0,1]),
                                  KB2x(e*KB[1,2,0]+f*KB[3,0,2]));
                                     f a + b e
```

2 Type C_n

Add one generator s_n to the symmetric group, acting on vectors $v \in \mathbb{Z}^n$ by :

$$v \to [v_1, \ldots, v_{n-1}, -v_n]$$

Considering vectors as exponents of monomials, s_n acts by $x_n \to 1/x_n$, leaving $x_1, \ldots, x_{n-1}$ invariant.

One adds an extra isobaric divided difference:

$$\pi_n : f \to (x_n f - f^{s_n}/x_n)/(x_n - 1/x_n)$$

which amounts to taking an extra variable x_{n+1} and specialize it to $1/x_n$.

As before, $\widehat{\pi}_n := \pi_n - 1$. It can also be written

$$f \to f\,\widehat{\pi}_n := (f - f^{s_n})/(x_n^2 - 1)$$

The family $\pi_1, \ldots, \pi_n$ satisfies the braid relations, with

$$\pi_{n-1}\,\pi_n\,\pi_{n-1}\,\pi_n = \pi_n\,\pi_{n-1}\,\pi_n\,\pi_{n-1}$$

$$\widehat{\pi}_{n-1}\,\widehat{\pi}_n\,\widehat{\pi}_{n-1}\,\widehat{\pi}_n = \widehat{\pi}_n\,\widehat{\pi}_{n-1}\,\widehat{\pi}_n\,\widehat{\pi}_{n-1}$$

Symplectic key polynomials are defined by starting again with all possible dominant monomials, and taking all images under products of π_i (resp. $\widehat{\pi}_i$)

They are indexed by vectors $v \in \mathbb{Z}^n$ and denoted $K_v^{\mathcal{S}}$ and $\widehat{K}_v^{\mathcal{S}}$. Putting an appropriate order on monomials, compatible with the Bruhat order, one has that x^v is the leading term of $K_v^{\mathcal{S}}$ and $\widehat{K}_v^{\mathcal{S}}$.

C-Scalar Product

$$(f,g)^{\mathcal{S}} := f\,g\prod_{i<j}(1-\frac{x_i}{x_j})\prod_{i\le j}(1-x_ix_j)\cap x^{0\ldots0} \tag{2}$$

$$= f\,g\prod_{i<j}(1-\frac{x_i}{x_j})\,\sigma_1\big(-S^2\big)\cap x^{0\ldots0} \tag{3}$$

($\cap x^{0\ldots0}$ means taking the constant term.)

```
# symplectic scalar product
CTS:=proc(n,pol1,pol2)  local i,j,res,eul;
 eul:=convert([seq(seq( 1-cat(x,i)/cat(x,j),i=1..j-1),j=2..n),
  seq(seq( 1-cat(x,i)*cat(x,j),i=1..j),j=1..n),
   seq(1/cat(x,i),i=1..n)],'*');# 1/x1..x.n pour utiliser residus
 res:=pol1*eul*pol2;
 for i from 1 to n do
   res:=residue(res,cat(x,i)=0);
 od;
 res
end:
```

Theorem. For any $u, v \in \mathbb{Z}^n$,

$$\big(\widehat{K}_u^{\mathcal{S}}\,,\,K_v^{\mathcal{S}}\big)^{\mathcal{S}} = 0 \text{ or } 1 \text{ when } u = -v$$

```
CTS(3,KS([-1,0,2]), KBS([1,0,2]));
                                    0

CTS(3,KS([-1,0,2]), KBS([1,0,-2]));
                                    1
```

Kernel

$$\Omega^{\mathcal{S}} := \sigma_1(XY - \Lambda^2 X) \prod_{i \le j} (1 - x_i/y_j)^{-1}$$

```
NoyauS:=proc(n)  local i,j;
 convert([seq(seq( 1-cat(x,i)*cat(x,j),i=1..j-1),j=2..n),
  seq(seq( 1/(1-cat(x,i)*cat(y,j)),i=1..n),j=1..n),
  seq(seq( 1/(1-cat(x,i)/cat(y,j)),i=1..j),j=1..n)],`*`)
end:
```

```
ACE> NoyauS(2);
                        1 - x1 x2
       ------------------------------------------------  *
       (1 - x1 y1) (1 - x2 y1) (1 - x1 y2) (1 - x2 y2)

                                 1
                      -------------------------------
                      /     x1 \ /     x1 \ /     x2 \
                      |1 - ----| |1 - ----| |1 - ----|
                      \     y1 / \     y2 / \     y2 /
```

Theorem.

$$\Omega^{\mathcal{S}} = \sum_{v \in \mathbb{N}^n} \widehat{K}_v(\mathbf{x}) K^{\mathcal{S}}_{-v}(\mathbf{y})$$

Symmetrizing in x, one obtains Littlewood's formula:

$$\sigma_1(X(Y + Y^\vee) - \Lambda^2 X) = \sum_\lambda S_\lambda(\mathbf{x}) Sp_\lambda(\mathbf{y} + \mathbf{y}^\vee)$$

Using the function which expands in the basis $\widehat{K}_v(x)$ (= taking the scalar product with $K_{v^\sim}$), one can check examples of the theorem :

```
ExtractCoeffKB(pol_xy,v,x):
 extracts the coefficient of KB[v](x) in pol_xy, and
 change in the result the variables y_i into x_i.
```

```
ExtractCoeffKB(NoyauS(3),[1,0,2],x);
             1       1        2              1     x1 x3
     3 x1 + ---- + ------ + x1  x2 + 3 x2 + ---- + -----
            x1         2                    x2     x2
                  x1 x3
```

```
                  x2 x3      x1 x2         2      2
          2 x3 + ----- + 2 ----- + x3 x1  + x3  x1
                  x1         x3

                                                2      2      2
         2     2         1        x1      x3     x1     x2
  + x3 x2  + x3  x2 + ------ + ----- + --- + --- + ---
                        2       x2 x3   x2     x3     x3
                     x3  x2

                    2     2     2
                   x1    x2    x3    x1    x2
           +       --- + --- + --- + --- + ---
                   x2    x1    x1      2     2
                                       x3    x3

                 x2      2       2
     +          ----- + ---- + x2  x1 + 2 x3 x2 x1
                x1 x3    x3
```

```
simplify(%- KS([-1,0,-2]));
                                    0
```

3 Type B_n

The extra isobaric divided differences are now :

$$f \to f\,\widehat{\pi}_n := (f - f^{s_n})/(x_n - 1)$$

and

$$\pi_n := \widehat{\pi}_n + 1\ .$$

Key polynomials $K_v^{\mathcal{O}}$ and $\widehat{K}_v^{\mathcal{O}}$ are defined as all images of dominant momials under products of isobaric divided differences.

The scalar product for the orthogonal case is :

B-Scalar Product

$$(f,g)^{\mathcal{O}} := f\,g\prod_{i<j}(1-\frac{x_i}{x_j})\prod_{i<j}(1-x_ix_j)\prod_i(1-x_i)\ \cap\ x^{0\ldots0} \qquad (4)$$

$$= f\,g\prod_{i<j}(1-\frac{x_i}{x_j})\,\sigma_1\big(-\Lambda^1-\Lambda^2\big)\ \cap\ x^{0\ldots0} \qquad (5)$$

i.e. one takes for the orthogonal case a factor $\sigma_1(-\Lambda^1-\Lambda^2)$ instead of $\sigma_1(-S^2)$
in the symplectic case (reverse choice for the generating functions !).

```
CTO:=proc(n,pol1,pol2)  local i,j,res,eul;
 eul:=convert([seq(seq( 1-cat(x,i)/cat(x,j),i=1..j-1),j=2..n),
  seq(seq( 1-cat(x,i)*cat(x,j),i=1..j-1),j=1..n), seq(1-cat(x,i),i=1..n),
           seq(1/cat(x,i),i=1..n)],`*`);# 1/x1..x.n pour utiliser residus
 res:=pol1*eul*pol2;
 for i from 1 to n do
   res:=residue(res,cat(x,i)=0);
 od;
 res
end:
```

Theorem. For any $u, v \in \mathbb{Z}^n$,

$$\left(\widehat{K}_u^{\mathcal{O}}\,,\, K_v^{\mathcal{O}}\right)^{\mathcal{O}} = 0 \text{ or } 1 \text{ when } u = -v$$

```
CTO(3,KO([-1,0,2]), KBO([1,0,2]));
                                        0

CTO(3,KO([-1,0,2]), KBO([1,0,-2]));
                                        1
```

Kernel

$$\Omega^{\mathcal{O}} := \sigma_1(XY - \Lambda^2 X) \prod_i (1+x_i) \prod_{i\le j} (1 - x_i/y_j)^{-1} = \Omega^{\mathcal{S}} \prod_i (1+x_i)\ .$$

Theorem.

$$\Omega^{\mathcal{O}} = \sum_{v\in\mathbb{N}^n} \widehat{K}_v(\mathbf{x})\, K^{\mathcal{O}}_{-v}(\mathbf{y})$$

Symmetrizing in x, one obtains Littlewood's formula:

$$\sigma_1(X(Y+1+Y^\vee) - S^2 X) = \sum_\lambda S_\lambda(\mathbf{x}) \mathcal{O}_\lambda(\mathbf{y}+1+\mathbf{y}^\vee)\ .$$

(the factor $\sigma_1(X - S^2X)$ is equal to $\prod_i (1+x_i)\sigma_1(-\Lambda^2 X)$.)

4 The π_i are self-adjoint.

The B and C scalar products are of the type

$$(f,g) = fg\,H\prod_{i<j}(1-x_i/x_j)\cap x^{0\dots0}\ ,$$

with H some fixed symmetric function.

Adding extra variables $x_{n+1}, x_{n+2}, \dots, x_\infty$, one can use the corresponding π_ω operator, that we shall denote $\pi_\aleph$.

One checks that the scalar product can be expressed in terms of $\pi_\aleph$, by checking what it gives on monomials with exponents in $\mathbb{Z}^n$.

Indeed,

$$x^v\prod_{i<j}(1-x_i/x_j)\cap x^{0\dots0} = x^v\,\pi_\aleph\big|_{x_i=0}$$

because non-nullity in either sides corresponds to the case $v=\rho-\rho^\sigma$ for some permutation σ.

As a corollary, since $\pi_i\,\pi_\aleph = \pi_\aleph$, we have that π_i, $i=1,\dots,n$–1 are self-adjoint with respect to the scalar product.

To prove this property for π_n, we need the precise value of the symmetric function H.

In the type C-case, we have the factor

$$\sigma_1(-S^2)\prod_{i<j}(1-x_i/x_j)\ .$$

It can be written as the following determinant :

$$x_2^{-1}\cdots x_n^{-n+1}\begin{vmatrix} x_1^{n-1}-x_1^{n+1} & \cdots & x_1^0-x_1^{2n}\\ \vdots & & \vdots\\ x_n^{n-1}-x_1^{n+1} & \cdots & x_n^0-x_1^{2n}\end{vmatrix}$$

Computing the scalar product (f,g) involves extracting the coefficient of x_n^0 in

$$fg(x_n^0-x_n^2),\ fg(x_n^{-1}-x_n^3),\dots,\ fg(x_n^{-n+1}-x_n^{n+1}$$

Therefore, to show that π_n is self-adjoint, one has to show that

$$(f\pi_n)g(x^{-i}-x^{i+2})\cap x^{0\dots0} = f\,(g\pi_n)(x^{-i}-x^{i+2})\cap x^{0\dots0}$$

i.e.

$$\Big((xf-f^{s_n}/x)gx-(xg-g^{s_n}/x)fx\Big)\,\frac{x^{-i-1}-x^{i+1}}{x-1/x}\cap x^{0\dots0}$$
$$= (f\,g^{s_n}-f^{s_n}g)\,\frac{x^{-i-1}-x^{i+1}}{x-1/x}\cap x^{0\dots0}=0$$

and this is indeed the case, because the last expression is antisymmetrical in $x_n, 1/x_n$.

Similarly, in the type B case,

$$\sigma_1(-\Lambda^1 - \Lambda^2) \prod_{i<j} (1 - x_i/x_j)$$

can be written as the determinant

$$x_2^{-1} \cdots x_n^{-n+1} \begin{vmatrix} x_1^{n-1} - x_1^n & \cdots & x_1^0 - x_1^{2n-1} \\ \vdots & & \vdots \\ x_n^{n-1} - x_1^n & \cdots & x_n^0 - x_1^{2n-1} \end{vmatrix}$$

Computing the scalar product (f, g) involves extracting the coefficient of x_n^0 in

$$fg(x_n^0 - x_n),\ fg(x_n^{-1} - x_n^2), \ldots,\ fg(x_n^{-n+1} - x_n^n)$$

To show that π_n is self-adjoint, one has to show that

$$(f\pi_n)g(x^{-i} - x^{i+1}) \cap x^{0\ldots0} = f\,(g\pi_n)(x^{-i} - x^{i+1}) \cap x^{0\ldots0}$$

i.e.

$$\Big((xf - f^{s_n})g - (xg - g^{s_n})f\Big) \frac{x^{-i} - x^{i+1}}{x-1} \cap x^{0\ldots0}$$
$$= (f\, g^{s_n} - f^{s_n} g) \frac{x^{-i} - x^{i+1}}{x-1} \cap x^{0\ldots0} = 0$$

and this is still the case, because the last expression is antisymmetrical in $x_n, 1/x_n$.

Orthogonality Properties of Key Polynomials for the classical groups

Voici l'état des réflexions après le week-end de travail avec Bernard

1 Type A_{n-1}

Notes deja envoyees, résumé:

Isobaric divided differences π_i, $\widehat{\pi}_i := \pi_i - 1$, $i = 1..n-1$.

Maximal isobaric divided difference π_ω

Involution $\clubsuit$: $x_i \to 1/x_{n+1-i}$

Conjugation:

$$\clubsuit\, \pi_i\, \clubsuit = \pi_{n+1-i}$$

Dominant Key polynomials = dominant monomials

General polynomials = images under all products of π_i (resp. $\widehat{\pi}_i$). Indexed by vectors in $\mathbb{N}^n$ and denoted K_v and $\widehat{K}_v$.

Scalar product:

$$(f, g) := \text{ Constant Term of } f(\mathbf{x})g(\mathbf{x}^{\clubsuit}) \prod_{1 \le i < j \le n} (1 - x_i/x_j) \qquad (1)$$

Reproducing kernel: Introduce $y_1, \ldots, y_n$

$$\Omega := 1/ \prod_{i+j \le n+1} (1 - x_i y_j)$$

Theorem. Ω is a reproducing kernel with respect to the scalar product, i.e.

$$(\Omega, f(\mathbf{x})) = f(\mathbf{y})$$

Using the theorem in "Double Crystal", one gets: (mais il faut donner preuve directe, je n'avais pas le produit scalaire a l'epoque)

Corollary

$$(K_v, \widehat{K}_{u^{\approx}}) = \delta_{u,v} \ .$$

with $u^{\approx} := [u_n, \ldots, u_1]$

2 Type C_n

Add one generator s_n to the symmetric group, acting on vectors $v \in \mathbb{Z}^n$ by :

$$v \to [v_1, \ldots, v_{n-1}, \ -v_n]$$

Considering vectors as exponents of monomials, s_n acts by $x_n \to 1/x_n$, leaving $x_1, \ldots, x_{n-1}$ invariant.

One adds an extra isobaric divided difference:

$$\pi_n : f \to (x_n f - f^{s_n}/x_n)/(x_n - 1/x_n)$$

which amounts to taking an extra variable x_{n+1} and specialize it to $1/x_n$.

As before, $\widehat{\pi}_n := \pi_n - 1$. The family $\pi_1, \ldots, \pi_n$ satisfies the braid relations, with

$$\pi_{n-1}\,\pi_n\,\pi_{n-1}\,\pi_n = \pi_n\,\pi_{n-1}\,\pi_n\,\pi_{n-1}$$

$$\widehat{\pi}_{n-1}\,\widehat{\pi}_n\,\widehat{\pi}_{n-1}\,\widehat{\pi}_n = \widehat{\pi}_n\,\widehat{\pi}_{n-1}\,\widehat{\pi}_n\,\widehat{\pi}_{n-1}$$

Symplectic key polynomials are defined by starting again with all possible dominant monomials, and taking all images under products of π_i (resp. $\widehat{\pi}_i$)

They are indexed by vectors $v \in \mathbb{Z}^n$ and denoted $K_v^{\mathcal{S}}$ and $\widehat{K}_v^{\mathcal{S}}$. Putting an appropriate order on monomials, compatible with the Bruhat order, one has that x^v is the leading term of $K_v^{\mathcal{S}}$ and $\widehat{K}_v^{\mathcal{S}}$.

Kernel

$$\Omega^{\mathcal{S}} := \sigma_1(XY - \Lambda^2 X) \prod_{i<j} (1 - x_i y_j)^{-1}$$

Theorem.

$$\Omega^{\mathcal{S}} = \sum_{v \in \mathbb{N}^n} \widehat{K}_v^{\mathcal{S}}(\mathbf{x}) K_{-v}^{\mathcal{S}}(\mathbf{y})$$

C'est donc une somme impliquant les pol clefs pour les types A et C (puisque $\widehat{K}_v^{\mathcal{S}} = \widehat{K}_v$ dans le cas ou $v \in \mathbb{N}^n$). Je n'ai pas reussi a obtenir une fonction generatrice pour une somme sur $\mathbb{Z}^n$, de meme que je ne connais pas le lien entre les pol clefs symplectiques et les pol clefs usuels pour les variables $y, 1/y$.

The image of $\prod_{i<j}(1-x_iy_j)^{-1}$ under π^x_ω (symmetrizer for the symmetric group) is $\sigma_1(XY^\vee)$, with $Y^\vee = \{1/y_1, \dots, 1/y_n\}$. On the other hand, each $\widehat{K}_v$ is sent to 0, when v is not dominant, because $\pi_i\,\widehat{\pi}_i = 0$.

Therefore, the theorem implies

$$\sigma_1(XY - \Lambda^2 X)\sigma_1(XY^\vee) = \sum_v K_v(\mathbf{x})K^{\mathcal{S}}_{-v}(\mathbf{y})$$

sum over all increasing vectors in $\mathbb{N}^n$. In that case, K_v is a Schur function, and $K^{\mathcal{S}}_{-v}$ is a symplectic Schur function, so that the last formula is in fact the identity of Littlewood

$$\sigma_1(X(Y+Y^\vee) - \Lambda^2 X) = \sum_\lambda S_\lambda(\mathbf{x})Sp_\lambda(\mathbf{y}+\mathbf{y}^\vee)$$

To prove the theorem, we have to use another scalar product:

$$(f,g)^{\mathcal{S}} := f\,g\prod_{i<j}(1-\frac{x_i}{x_j})\prod_{i\le j}(1-x_ix_j)^{-1} \cap x^{0\dots0}$$

(i.e. $\cap x^{0\dots0}$ means taking the constant term) (Notez $i \le j$ dans le deuxieme produit).

Theorem. For any $u, v \in \mathbb{Z}^n$,

$$\left(\widehat{K}^{\mathcal{S}}_u\,,\,K^{\mathcal{S}}_v\right)^{\mathcal{S}} = 0 \text{ or } 1 \text{ when } u = -v$$

```
ACE> aa:=ListBar(2,2); # All weights of degree 2
aa:=[[1,1],[-1,1],[1,-1],[-1,-1],[2,0],[0,2],[-2,0],[0,-2]];

# Polynomes KS
ACE>for v in aa do lprint(v,':',Laurent2Exp(expand(KBS(v)),2)) od:

[1, 1]   :   x[1,1]
[-1, 1]   :   x[0,0]+x[-1,1]
[1, -1]   :   x[1,-1]
[-1, -1]   :   x[-1,-1]
[2, 0]   :   x[2,0]
[0, 2]   :   x[1,1]+x[0,2]
[-2, 0]   :   x[0,0]+x[-1,-1]+x[-1,1]+x[-2,0]
[0, -2]   :   x[0,0]+x[1,-1]+x[0,-2]
# symplectic scalar product
CTS:=proc(n,pol1,pol2)  local i,j,res,eul;
```

```
 eul:=convert([seq(seq( 1-cat(x,i)/cat(x,j),i=1..j-1),j=2..n),
  seq(seq( 1-cat(x,i)*cat(x,j),i=1..j),j=1..n),
   seq(1/cat(x,i),i=1..n)],`*`);# 1/x1..x.n pour utiliser redidus
 res:=pol1*eul*pol2;
 for i from 1 to n do
   res:=residue(res,cat(x,i)=0);
 od;
 res
end:

lpol1:=[seq(KS(u),u=aa)]:
lpol2:=[seq(KBS(u),u=aa)]:

# Checks all scalar products and prints the non-zero ones
for i from 1 to nops(aa)  do
  for j from 1 to nops(aa) do res:=CTS(2,lpol1[i],lpol2[j]);
  if res<>0 then lprint(aa[i],aa[j],res) fi od od:

[1, 1]    [-1, -1]   1
[-1, 1]   [1, -1]   1
[1, -1]   [-1, 1]   1
[-1, -1]  [1, 1]   1
[2, 0]    [-2, 0]   1
[0, 2]    [0, -2]   1
[-2, 0]   [2, 0]   1
[0, -2]   [0, 2]   1
```

Preuve: en verifiant que les π_i sont auto-adjoints par rapport au produit scalaire (pas encore verifie)