Talmi-Moshinksy transformation for 2D harmonic oscillator functions

Yang Li[*]
Department of Physics and Astronomy, Iowa State University, Ames, IA 50011

Dated[†]: January 20, 2017

1 Talmi-Moshinsky transformation

Let $\phi_{nm}(\vec{p})$ be harmonic oscillator functions in 2 dimensions. The exact definition is,

$$\phi_{nm}(\vec{p}) = \sqrt{\frac{4\pi n!}{(n+|m|)!}} \exp(-p^2/2) p^{|m|} L_n^{|m|}(p^2) e^{im\theta}, \tag{1}$$

where $p = |\vec{p}|$, $\theta = \arg \vec{p}$, and we have taken the oscillator length $b = \sqrt{M\Omega} = 1$. $L_n^a(z)$ is the associate Laguerre polynomial,

$$L_n^a(z) = \sum_{k=0}^{n} (-1)^k \binom{n+a}{n-k} \frac{z^k}{k!}. \tag{2}$$

This definition admits orthonormality:

$$\int \frac{\mathrm{d}^2 p}{(2\pi)^2} \phi^*_{n'm'}(\vec{p}) \phi_{nm}(\vec{p}) = \delta_{nn'} \delta_{mm'}. \tag{3}$$

The Talmi-Moshinsky transformation (TMT) reads,

$$\phi_{n_1 m_1}(\vec{p}_1) \phi_{n_2 m_2}(\vec{p}_2) = \sum_{NMnm} \mathcal{M}^{NMnm}_{n_1 m_1 n_2 m_2}(\delta)\ \phi_{NM}(\vec{P}) \phi_{nm}(\vec{q}) \tag{4}$$

where

$$\begin{aligned} \vec{P} &= \cos\delta\, \vec{p}_1 + \sin\delta\, \vec{p}_2, \quad \vec{q} = \sin\delta\, \vec{p}_1 - \cos\delta\, \vec{p}_2 \\ \Rightarrow \quad \vec{p}_1 &= \cos\delta\, \vec{P} + \sin\delta\, \vec{q}, \quad \vec{p}_2 = \sin\delta\, \vec{P} - \cos\delta\, \vec{q}. \end{aligned} \tag{5}$$

Coefficients $\mathcal{M}^{NMnm}_{n_1 m_1 n_2 m_2}$ are real numbers, called the Talmi-Moshinsky coefficients (TMCs). They satisfy the following properties:

1. $\mathcal{M}^{NMnm}_{n_1 m_1 n_2 m_2} \propto \delta_{2N+|M|+2n+|m|, 2n_1+|m_1|+2n_2+|m_2|} \delta_{M+m, m_1+m_2}$.

   In other words, TMT conserves oscillator energy and angular momentum. This property also means, for given n_1, m_1, n_2, m_2, the number of non-vanishing TMCs is finite.

2. Since the inverse transformation of the momenta is the same as the transformation itself, the inverse TMT is the same:

$$\phi_{NM}(\vec{P}) \phi_{nm}(\vec{q}) = \sum_{n_1 m_1 n_2 m_2} \mathcal{M}^{n_1 m_1 n_2 m_2}_{NMnm}(\delta)\ \phi_{n_1 m_1}(\vec{p}_1) \phi_{n_2 m_2}(\vec{p}_2) \tag{6}$$

---

[*] `leeyoung@iastate.edu`

[†] Update:March 9, 2017

3. Combining Eqs. (4 & 6), we get the normalization of the TMCs:

$$\sum_{NMnm} \mathcal{M}^{NMnm}_{n_1m_1n_2m_2} \mathcal{M}^{n'_1m'_1n'_2m'_2}_{NMnm} = \delta_{n_1,n'_1}\delta_{n_2,n'_2}\delta_{m_1,m'_1}\delta_{m_2,m'_2}. \tag{7}$$

4. Integral representation:

$$\mathcal{M}^{NMnm}_{n_1m_1n_2m_2}(\delta) = \int \frac{\mathrm{d}^2 p_2}{(2\pi)^2} \int \frac{\mathrm{d}^2 p_2}{(2\pi)^2} \phi_{n_1m_1}(\vec{p}_1)\phi_{n_2m_2}(\vec{p}_2)\phi^*_{NM}(\vec{P})\phi^*_{nm}(\vec{q}). \tag{8}$$

5.

$$\mathcal{M}^{N,-M,n,-m}_{n_1,-m_1,n_2,-m_2}(\delta) = (-1)^m \mathcal{M}^{NMnm}_{n_1m_1n_2m_2}(\delta). \tag{9}$$

6. Swapping $\vec{p}_1$ and $\vec{p}_2$ we get:

$$\mathcal{M}^{NMnm}_{n_1m_1n_2m_2}(\delta) = (-1)^m \mathcal{M}^{NMnm}_{n_2m_2n_1m_1}(\pi/2 - \delta). \tag{10}$$

7. Taking $\vec{p}_1 = 0$ and $\vec{p}_2 = 0$,

$$\sum_{N,n} \mathcal{M}^{N,0,n,0}_{n_1,m_1,n_2,m_2}(\delta) = \delta_{m_1,0}\delta_{m_2,0}. \tag{11}$$

2 Factorization of the center-of-mass motion

Talmi-Moshinksy transformation is typically used to separate the center-of-mass motion within the harmonic oscillator (HO) basis. The HO basis, together with the discretized plane wave basis, are the only two known analytic bases that preserve the factorization of the center-of-mass motion within a finite basis truncation. For fixed quanta $\{n_1, m_1, n_2, m_2\}$, the fact that there are only finite number of non-vanishing TMCs as constrained by the conservation laws, reflects this important property.

Consider a two-body wave function expressed in the HO basis:

$$\psi(\vec{p}_1, \vec{p}_2) = \sum_{n_1,m_1,n_2,m_2} f_{n_1m_1n_2m_2}\phi_{n_1m_1}(\vec{p}_1)\phi_{n_2m_2}(\vec{p}_2). \tag{12}$$

This wave function is the eigenstate of some Hamiltonian H that acts only on the relative part of the system, namely, the Hamiltonian is boost invariant. Then, the wave function can be written as the product of the wave function of the center-of-mass motion $\Phi_{\text{cm}}(\vec{P})$ and that of the relative motion $\varphi_{\text{rel}}(\vec{q})$:

$$\psi(\vec{p}_1, \vec{p}_2) = \Phi_{\text{cm}}(\vec{P})\varphi_{\text{rel}}(\vec{q}). \tag{13}$$

where $\vec{P} = \vec{p}_1 + \vec{p}_2$, and $\vec{q} = (\vec{p}_1 - \vec{p}_2)/2$. It can be shown that the existence of factorization implies the factorization of coefficients,

$$\sum_{n_1,m_1,n_2,m_2} f_{n_1m_1n_2m_2}\mathcal{M}^{NMnm}_{n_1m_1n_2m_2}(\tfrac{\pi}{4}) = \chi_{NM} \cdot \xi_{nm}, \tag{14}$$

where

$$\Phi_{\text{cm}}(\vec{P}) = \sum_{N,M} \chi_{NM}\phi_{NM}(\vec{P}/\sqrt{2}), \quad \varphi_{\text{rel}}(\vec{q}) = \sum_{n,m} \xi_{nm}\phi_{nm}(\sqrt{2}\vec{q}). \tag{15}$$

In particular, it is convenient to extend the Hamiltonian H with a Lagrangian multiplier to manipulate the center-of-mass motion: $H \to H + \lambda_{\text{cm}}H_{\text{cm}}$. A typical choice of H_{cm} is the HO Hamiltonian for center-of-mass motions $H_{\text{cm}} = \vec{P}^2/(2M) + (M/2)\Omega^2\vec{R}^2$. The resulting center-of-mass wave function is the ground state harmonic oscillator function. Therefore, $\chi_{NM} = \delta_{N0}\delta_{M0}$. The relative part of the wave function can be extracted as,

$$\varphi_{\text{rel}}(\vec{q}) = \sum_{nm} \sum_{n_1,m_1,n_2,m_2} f_{n_1m_1n_2m_2}\mathcal{M}^{00nm}_{n_1m_1n_2m_2}(\tfrac{\pi}{4})\, \phi_{nm}(\sqrt{2}\vec{q}). \tag{16}$$

In light-front dynamics, the situation is very similar: the boost transformations are also kinematical and Galilean[1]. The factorization of the center-of-momentum motion in a finite truncated basis is retained if the HO basis within the holographic variable is employed:

$$\psi(x, \vec{p}_{1\perp}, \vec{p}_{2\perp}) = \sum_{n_1,m_1,n_2,m_2} f_{n_1m_1n_2m_2}(x)\phi_{n_1m_1}(\vec{p}_{1\perp}/\sqrt{x})\phi_{n_2m_2}(\vec{p}_{2\perp}/\sqrt{1-x}). \tag{17}$$

The factorization reads,

$$\psi(x, \vec{p}_{1\perp}, \vec{p}_{2\perp}) = \Phi_{\text{cm}}(\vec{P}_\perp)\varphi_{\text{rel}}(x, \vec{k}_\perp) \tag{18}$$

where $\vec{P}_\perp = \vec{p}_{1\perp} + \vec{p}_{2\perp}$, and $\vec{k}_\perp = p_{1\perp} - x\vec{P}_\perp = (1-x)\vec{p}_{1\perp} - x\vec{p}_{2\perp}$. Use the a Lagrangian multiplier term: $H \to H + \lambda_{\text{cm}}\big(\vec{P}_\perp^2 + (P^+\Omega)^2\vec{R}_\perp^2\big)$, $\vec{R}_\perp = x\vec{r}_{1\perp} + (1-x)\vec{r}_{2\perp}$. The relative part of the wave function can be extracted as,

$$\varphi_{\text{rel}}(x, \vec{k}_\perp) = \sum_{nm}\sum_{n_1,m_1,n_2,m_2} f_{n_1m_1n_2m_2}(x)\mathcal{M}^{00nm}_{n_1m_1n_2m_2}(\delta)\ \phi_{nm}(\vec{k}/\sqrt{x(1-x)}), \tag{19}$$

where $\tan\delta = \sqrt{(1-x)/x}$.

3 Generating function

The TMCs can be obtained using the generating function technique. The exponential generating function for harmonic oscillator functions is

$$\exp(-\vec{p}^2/2 + 2\vec{p}\cdot\vec{r} - \vec{r}^2) = \sum_{n=0}^{+\infty}\sum_{m=-\infty}^{+\infty}\frac{(-1)^n}{\sqrt{4\pi(n+|m|)!n!}}\phi_{nm}(\vec{p})e^{-im\vartheta}r^{2n+|m|}, \tag{20}$$

where $r = |\vec{r}|$ and $\vartheta = \arg\vec{r}$. Note the identity:

$$\big(\tfrac{1}{2}\vec{p}_1^2 - 2\vec{p}_1\cdot\vec{r}_1 + \vec{r}_1^2\big) + \big(\tfrac{1}{2}\vec{p}_2^2 - 2\vec{p}_2\cdot\vec{r}_2 + \vec{r}_2^2\big) = \big(\tfrac{1}{2}\vec{P}^2 - 2\vec{P}\cdot\vec{R} + \vec{R}^2\big) + \big(\tfrac{1}{2}\vec{q}^2 - 2\vec{q}\cdot\vec{r} + \vec{r}^2\big) \tag{21}$$

where

$$\vec{R} = \cos\delta\,\vec{r}_1 + \sin\delta\,\vec{r}_2, \quad \vec{r} = \sin\delta\,\vec{r}_1 - \cos\delta\,\vec{r}_2. \tag{22}$$

Apply the generating function Eq. (20),

$$\begin{aligned}&\sum_{n_1,m_1,n_2,m_2}\frac{(-1)^{n_1+n_2}}{4\pi\sqrt{(n_1+|m_1|)!n_1!(n_2+|m_2|)!n_2!}}\phi^{m_1}_{n_1}(\vec{p}_1)\phi^{m_2}_{n_2}(\vec{p}_2)e^{-im_1\vartheta_1-im_2\vartheta_2}r_1^{2n_1+|m_1|}r_2^{2n_2+|m_2|}\\ =&\sum_{N,M,n,m}\frac{(-1)^{N+n}}{4\pi\sqrt{(N+|M|)!N!(n+|m|)!n!}}\phi^M_N(\vec{P})\phi^m_n(\vec{q})e^{-iM\varTheta-im\vartheta}R^{2N+|M|}r^{2n+|m|}\end{aligned} \tag{23}$$

where $r_1 = |\vec{r}_1|$, $\vartheta_1 = \arg\vec{r}_1$, $r_2 = |\vec{r}_2|$, $\vartheta_2 = \arg\vec{r}_2$, $R = |\vec{R}|$, $\varTheta = \arg\vec{R}$, $r = |\vec{r}|$, $\vartheta = \arg\vec{r}$. Expand either side, and identify the same terms, the TMCs can be obtained. The analytic expression reads,

$$\begin{aligned}\mathcal{M}^{NMnm}_{n_1m_1n_2m_2}(\delta) =\ & \delta_{m_1+m_2,m+M}\delta_{2n_1+|m_1|+2n_2+|m_2|,2N+|M|+2n+|m|}(-1)^{N+n+n_1+n_2+M+m_1}\\ &\times(\sin\delta)^{2N+|M|}(\cos\delta)^{2n+|m|}(\tan\delta)^{2n_1+|m_1|}\sqrt{\frac{n_1!(n_1+|m_1|)!n_2!(n_2+|m_2|)!}{N!(N+|M|)!n!(n+|m|)!}}\\ &\times\sum_{a=\max\{0,\varepsilon_1^- - e^-\}}^{\min\{\varepsilon_1^-,E^-\}}\binom{E^-}{a}\binom{e^-}{\varepsilon_1^- - a}\frac{(-1)^a}{(\tan\delta)^{2a}}\sum_{b=\max\{0,\varepsilon_1^+ - e^+\}}^{\min\{\varepsilon_1^+,E^+\}}\binom{E^+}{b}\binom{e^+}{\varepsilon_1^+ - b}\frac{(-1)^b}{(\tan\delta)^{2b}}.\end{aligned} \tag{24}$$

where $\varepsilon_1^\pm = 2n_1 + (|m_1| \pm m_1)/2$, $E^\pm = 2N + (|M| \pm M)/2$, $e^\pm = 2n + (|m| \pm m)/2$. The complicated lower and upper limits came from restricting the range of the arguments in binomial coefficients such that for binomial coefficient $\binom{n}{k}$, $0 \le k \le n$ $(0! \equiv 1)$.

---

[1] The light-front transverse boosts involve the longitudinal momentum: $\vec{p}_{i\perp} \to \vec{p}_{i\perp} - p_i^+\vec{\beta}_\perp$.

4 Algorithm

Mathematica implementation

```
(*************************************************************************)
BeginPackage["TalmiMoshinsky‘"]

TalmiMoshinsky::usage =
"TalmiMoshinsky[N, M, n, m, n1, m1, n2, m2, tan(delta)] computes the Talmi-Moshinsky
coefficients for 2D harmonic oscillator functions."

TMC::usage="TMC[N, M, n, m, n1, m1, n2, m2, tan(delta)] or TMC[N, M, n, m, n1, m1, n2, m2,
x1, x2] calculates the 2D Talmi-Moshinsky coefficient with generalized binomial and
multinomial coefficients. tan(delta)=sqrt(x2/x1)."

Begin["TalmiMoshinsky‘Private‘"]

Print["* * * TalmiMoshinsky.m * * *"];
Print["Yang Li <leeyoung@iastate.edu>, Oct. 05, 2011, updated: Jan. 20, 2017"];
Print["This package provides the calculation of Talmi-Moshinsky transformation brackets.

Functions:
TalmiMoshinsky[N, M, n, m, n1, m1, n2, m2, Tan[delta]]
computes the Talmi-Moshinsky coefficients for 2D harmonic
oscillator functions."];

Print["TMC[N, M, n, m, n1, m1, n2, m2, Tan[delta]] and
TMC[N, M, n, m, n1, m1, n2, m2, x2, x1] are aliases for TalmiMoshinsky, with
Tan[delta] = Sqrt[x2/x1]."];
Print["* * * * * * * * * * * * * *"];

Clear[TMC, TalmiMoshinsky];

TMC[N1_Integer, M1_Integer, N2_Integer, M2_Integer, n1_Integer, m1_Integer,
n2_Integer, m2_Integer, t_] := TalmiMoshinsky[N1, M1, N2, M2,
n1, m1, n2, m2, t]

TMC[N1_Integer, M1_Integer, N2_Integer, M2_Integer, n1_Integer, m1_Integer,
n2_Integer, m2_Integer, x1_, x2_] := TalmiMoshinsky[N1, M1, N2,
M2, n1, m1, n2, m2, Sqrt[x2/x1]]

TalmiMoshinsky[N1_Integer, M1_Integer, N2_Integer, M2_Integer, n1_Integer,
m1_Integer, n2_Integer, m2_Integer, t_] :=
  Module[
  {s, c, r, E1p, E1m, E2p, E2m, e1p, e1m, pref, part1, part2},

      If[2*N1+Abs[M1]+2*N2+Abs[M2]==2*n1+Abs[m1]+2*n2+Abs[m2] &&
     M1+M2==m1+m2,

     s=t/Sqrt[1+t^2];
     c=1/Sqrt[1+t^2];
     r=-1/t^2;

     E1p = N1 + (Abs[M1]+M1)/2;
     E1m = N1 + (Abs[M1]-M1)/2;
     E2p = N2 + (Abs[M2]+M2)/2;
     E2m = N2 + (Abs[M2]-M2)/2;
     e1p = n1 + (Abs[m1]+m1)/2;
     e1m = n1 + (Abs[m1]-m1)/2;

     pref = (-1)^(N1+N2+n1+n2+M2+m1) * t^(2*n1+Abs[m1])
                  * s^(2*N1+Abs[M1]) * c^(2*N2+Abs[M2])
                  * Sqrt[n1!/N1!*n2!/N2!*(n1+Abs[m1])!/(N1+Abs[M1])!
                       *(n2+Abs[m2])!/(N2+Abs[M2])!];
     part1 = Sum[ r^i * Binomial[E1m, i] * Binomial[E2m, e1m-i],
                         {i, Max[0, e1m-E2m], Min[e1m,E1m]}];
     part2 = Sum[ r^i * Binomial[E1p, i] * Binomial[E2p, e1p-i],
                         {i, Max[0, e1p-E2p], Min[e1p,E1p]}];
     pref*part1*part2
  ,
  0
  ]
]

End[ ]

EndPackage[ ]
(***************************************************************************)
```

Fortran implementation

```
!***************************************************************************!
  double precision function TMC(nn, mm, n, m, n1, m1, n2, m2, tandelta)
  implicit none
  !
  ! TMC(nn, mm, n, m; n1, m1, n2, m2; b2/b1)
  ! TMC(nn, mm, n, m; n1, m1, n2, m2; sqrt(x2/x1))
  !
  ! we actuall only need Binomial coefficients for n>0,0<=m<=n here
  integer, intent(in) :: nn, mm, n, m, n1, m1, n2, m2
  double precision, intent(in) :: tandelta
  integer :: a, b, EEp, EEm, Ep, Em, E1p, E1m
  double precision :: sindelta, cosdelta, s1, s2, t
  double precision, external :: Binomial, lognm
!
  if(2*nn+abs(mm)+2*n+abs(m)==2*n1+abs(m1)+2*n2+abs(m2).and.mm+m==m1+m2) then
!
 sindelta = tandelta/sqrt(1d0+tandelta**2)
 cosdelta = 1d0/sqrt(1d0+tandelta**2)
 t = -1d0/(tandelta*tandelta)
!
 EEp = nn + (abs(mm)+mm)/2
 EEm = nn + (abs(mm)-mm)/2
 Ep  = n  + (abs(m)+m)/2
 Em  = n  + (abs(m)-m)/2
 E1p = n1 + (abs(m1)+m1)/2
 E1m = n1 + (abs(m1)-m1)/2
!
     s1 = 0d0
 do a = max(0,E1m-Em), min(E1m,EEm)
   s1 = s1 + t**a * binomial(EEm,a) * binomial(Em,E1m-a)
 enddo
 s2 = 0d0
 do b = max(0,E1p-Ep), min(E1p,EEp)
    s2 = s2 + t**b * binomial(EEp,b) * binomial(Ep,E1p-b)
 enddo
!
     TMC = (1-2*mod(abs(nn+n+n1+n2+m+m1),2))*s1*s2*tandelta**(2*n1+abs(m1)) &
     * sindelta**(2*nn+abs(mm)) * cosdelta**(2*n+abs(m))                &
 * exp(lognm(n1,m1)+lognm(n2,m2)-lognm(nn,mm)-lognm(n,m))

  else
 TMC = 0d0
  endif
!
  end function

!==============================================================================
!   Auxiliary functions: factorials, binomials, multinomials etc             !
!==============================================================================

    Function LogNM(n, m)
    implicit none
!   log(sqrt((n+abs(m))!n!))

    double precision :: logNM, logn
    integer :: n, m, i

    logn = 0D0;
    do i = 2, n
        logn = logn + log(dble(i));
    end do

    logNM = logn;
    do i = n+1, n+abs(m)
        logNM = logNM + log(dble(i));
    end do

    logNM = 0.5D0 * (logNM + logn);

    End Function logNM

    double precision Function Binomial(n,m)
    implicit none
    ! binomial coefficients, n >= 0, m >= 0, and n >= m
    ! Binomial(n, m) = n!/(m! * (n-m)!);
    ! Binomial(n, m) = Binomial(n, n-m);
    integer, intent(in) :: n, m
    integer :: k, i

    k = min(m,n-m)
    Binomial = 1D0
    do i =  1, k
        Binomial = Binomial*dble(n-k+i)/dble(i)
    end do

    End Function
!*************************************************************************!
```