Crossing matrix of $X \to 3\pi$

```
begin
    using SymPy
    import PyCall
    #
    PyCall.pyimport_conda("sympy.physics.quantum.spin", "sympy")
    PyCall.pyimport_conda("sympy.physics.wigner",       "sympy")

    import_from(sympy.physics.quantum.spin, (:WignerD,), typ=:Any)
    import_from(sympy.physics.wigner)
    import_from(sympy.physics.quantum.spin)
end
```

```
wignerd (generic function with 1 method)
```

```
wignerd(j,m1,m2,θ) = WignerD(j,m1,m2, 0, θ, 0).doit()
```

$(m_X, m_1, m_2, m_3, s, t)$

```
begin
    mX, m1, m2, m3 = @vars m_X m_1 m_2 m_3 positive=true
    s, t = @vars s t positive=true
    mX, m1, m2, m3, s, t
end
```

$(\lambda_{Xs}, \lambda_s, \lambda_{Xt}, \lambda_t)$

```
λXs, λs, λXt, λt = @vars lambda_Xs lambda_s lambda_Xt lambda_t positive=true
```

$$\left(\cos(\omega), \sin(\omega) \right)$$

$$\Rightarrow \frac{n}{\sqrt{\lambda_{Xs}}\sqrt{\lambda_{Xt}}} \quad \Rightarrow \frac{2m_X\sqrt{\phi}}{\sqrt{\lambda_{Xs}}\sqrt{\lambda_{Xt}}}$$

```
begin
    # crossing angle
    ω, = @vars omega
    n, ϕ = @vars n phi
    #
    cosω = n/sqrt(λXt*λXs)
    sinω = 2*sqrt(phi)*mX/sqrt(λXt*λXs)
    #
    cos(ω)=>cosω, sin(ω)=>sinω
end
```

Kt (generic function with 1 method)

```
begin
    K(λ, Yj, Kallen) = (2sqrt(ϕ))^λ * sqrt(Kallen)^Yj
    Ks(λ, Yj) = K(λ, Yj, λXs)
    Kt(λ, Yj) = K(λ, Yj, λXt)
end
```

paritywignerd (generic function with 1 method)

```
function paritywignerd(J,λ,λ′,fac)
    wd1 = wignerd(J, λ,λ′, ω)
    wd2 = wignerd(J, λ, -λ′, ω)
    return wd1 + fac * (-1)^λ′ * wd2
end
```

ℂij (generic function with 1 method)

```
ℂij(J,λ,λ′,YX,fac) = paritywignerd(J,λ,λ′,fac) / Ks(λ, -YX) * Kt(λ′, -YX)
```

Yj (generic function with 1 method)

```
Yj(j,YX) = abs(j - YX) - j
```

const ηπ = -1

```
const ηπ = -1
```

YX (generic function with 1 method)

```
YX(J,ηX) = J - (1 + ηX) / Sym(2)
```

ℂ (generic function with 1 method)

```
function ℂ(J;ηX,js,jt)
    γₓ = γX(J,ηX)
    γⱼs, γⱼt = γj(js,γₓ), γj(jt,γₓ)
    ηs = ηπ^3 * ηX
    λmin = ηX==1 ? 1 : 0
    #
    ((js<γₓ) || (jt < γₓ)) &&
        error("not implemented, need ℂij(J, λ, λ′, γjs, γjt, ηs)")
    #
    m = [ℂij(J, λ, λ′, γₓ, ηs) for λ in λmin:js, λ′ in λmin:jt]
    return m
end
```

replacesincos (generic function with 1 method)

```
replacesincos(e) = simplify(
    subs(
        sympy.expand_trig(e),
        cos(ω) => cosω, sin(ω) => sinω))
```

cosmetics (generic function with 1 method)

```
cosmetics(e) = simplify(subs(e, λXs*λXt=>n^2 + 4mX^2*ϕ))
```

Examples

```
md"""
## Examples
"""
```

$$J^{PC} = 1^{--}\ \rho\pi$$

ℂϕ =

$$[1]$$

```
ℂϕ = replacesincos.(ℂ(1; ηX=1, js=1, jt=1))
```

$$J^{PC} = 1^{++}\ f_2\pi$$

ℂa1 =

$$\begin{bmatrix} \frac{2n}{\lambda_{Xt}} & \frac{4\sqrt{2}m_X\phi}{\lambda_{Xt}} \\ -\frac{\sqrt{2}m_X}{\lambda_{Xt}} & \frac{n}{\lambda_{Xt}} \end{bmatrix}$$

```
ℂa1 = replacesincos.(ℂ(1; ηX=-1, js=1, jt=1))
```

$J^{PC} = 2^{-+}\ \rho_3\pi$

ℂπ2 =

$$\begin{bmatrix} -4m_X^2\phi + 2n^2 & 4\sqrt{6}m_X n\phi & 8\sqrt{6}m_X^2\phi^2 \\ -\sqrt{6}m_X n & -4m_X^2\phi + n^2 & 4m_X n\phi \\ \frac{\sqrt{6}m_X^2}{2} & -m_X n & 2m_X^2\phi + n^2 \end{bmatrix}$$

```
ℂπ2 = cosmetics.(replacesincos.(ℂ(2; ηX=-1, js=2, jt=2) .* λXt^2))
```

$J^{PC} = 3^{++}$

ℂa3 =

$$\begin{bmatrix} 2n\left(-6m_X^2\phi + n^2\right) & 8\sqrt{3}m_X\phi\left(-m_X^2\phi + n^2\right) & 8\sqrt{30}m_X^2 n\phi^2 & 32 \\ 2\sqrt{3}m_X\left(m_X^2\phi - n^2\right) & n\left(-11m_X^2\phi + n^2\right) & 2\sqrt{10}m_X\phi\left(-2m_X^2\phi + n^2\right) & 4\sqrt{\cdot} \\ \frac{\sqrt{30}m_X^2 n}{2} & \frac{\sqrt{10}m_X\left(2m_X^2\phi - n^2\right)}{2} & n\left(-2m_X^2\phi + n^2\right) & 2\sqrt{6}m_X\phi\left(2m\right. \\ -\frac{\sqrt{5}m_X^3}{2} & \frac{\sqrt{15}m_X^2 n}{4} & -\frac{\sqrt{6}m_X\left(2m_X^2\phi + n^2\right)}{2} & n\left(3m\right. \end{bmatrix}$$

```
ℂa3 = cosmetics.(replacesincos.(ℂ(3; ηX=-1, js=3, jt=3) .* λXt^3))
```