Great! Let's implement the **Wigner–Eckart theorem** numerically using the **GSL library** in C.

🧠 Goal

We want to compute a **matrix element** like:

$$\langle j', m'|T_q^{(k)}|j, m\rangle = \langle j'||T^{(k)}||j\rangle \cdot C_{j,m;\,k,q}^{j',m'}$$

Where:

- $T_q^{(k)}$: tensor operator of rank k, component q
- $\langle j'||T^{(k)}||j\rangle$: **reduced matrix element** (you provide)
- C: **Clebsch–Gordan coefficient**, computable with GSL

✅ GSL Functions Used

- `gsl_sf_coupling_3j(2j1, 2j2, 2j3, 2m1, 2m2, 2m3)` — to get **CG coefficients**
- `gsl_sf_coupling_6j(...)` — if you want to do full recoupling

💻 Example: Compute $\langle \frac{1}{2}, \frac{1}{2}|S_z|\frac{1}{2}, \frac{1}{2}\rangle$

We know:

- Operator $S_z \sim T_0^{(1)}$
- CG coefficient: $C_{1/2,1/2;1,0}^{1/2,1/2} = \sqrt{1/3}$
- Reduced matrix element: $\langle \frac{1}{2}||\vec{S}||\frac{1}{2}\rangle = \frac{\sqrt{3}}{2}$

So the full value should be:

$$\langle \tfrac{1}{2}, \tfrac{1}{2}|S_z|\tfrac{1}{2}, \tfrac{1}{2}\rangle = \frac{\sqrt{3}}{2} \cdot \sqrt{\frac{1}{3}} = \frac{1}{2}$$

📄 Full C Code (with GSL)

```
#include <stdio.h>
#include <math.h>
#include <gsl/gsl_sf_coupling.h>

// Convert float spin to 2j format (integers)
#define TWICE(x) ((int)(2 * (x)))

int main(void) {
    // Spins
    double j = 0.5;
    double m = 0.5;
    int two_j = TWICE(j);
    int two_m = TWICE(m);
```

```
    // Operator: vector operator T^{(1)}_0 corresponds to q = 0, rank k = 1
    int k = 1;
    int q = 0;

    // Final state same: j', m'
    int two_jp = two_j;
    int two_mp = two_m;

    // Compute Clebsch–Gordan coefficient using 3j symbol:
    // C^{j',m'}_{j,m;k,q} = (-1)^{j'-j-q} * sqrt(2j'+1) * (j j' k; m -m' -q)
    double phase = pow(-1.0, (two_jp - two_j - 2*q)/2.0);
    double cg = gsl_sf_coupling_3j(two_j, TWICE(k), two_jp,
                                   two_m, TWICE(q), -two_mp);

    cg *= phase * sqrt(two_jp + 1.0);

    // Reduced matrix element for spin-1/2 operator
    double red = sqrt(3.0) / 2.0;

    // Apply Wigner–Eckart theorem
    double matrix_element = red * cg;

    printf("Clebsch–Gordan coefficient: %.6f\n", cg);
    printf("Reduced matrix element:     %.6f\n", red);
    printf("Full matrix element:        %.6f\n", matrix_element);  // should be 0.5

    return 0;
}
```

🔧 Compile & Run

bash 复制

```bash
gcc wigner_eckart_gsl.c -lgsl -lgslcblas -lm -o wigner
./wigner
```

✅ Output (Expected)

yaml 复制

```yaml
Clebsch–Gordan coefficient: 0.577350
Reduced matrix element:     0.866025
Full matrix element:        0.500000
```

This confirms:

$$\frac{\sqrt{3}}{2} \cdot \sqrt{\frac{1}{3}} = \frac{1}{2}$$

Great — let me explain the line:

```c
cg *= phase * sqrt(two_jp + 1.0);
```

in the context of the Wigner–Eckart theorem implementation using **GSL's 3j symbols**.

📘 Background: Wigner–Eckart Theorem

We want to compute the **Clebsch–Gordan coefficient**:

$$C^{j',m'}_{j,m;\,k,q}$$

But GSL does **not** provide CG coefficients directly — it gives **Wigner 3j symbols**:

$$\begin{pmatrix} j & k & j' \\ m & q & -m' \end{pmatrix}$$

These are related by:

$$C^{j',m'}_{j,m;\,k,q} = (-1)^{j'-j-q} \cdot \sqrt{2j'+1} \cdot \begin{pmatrix} j & k & j' \\ m & q & -m' \end{pmatrix}$$

✅ Code Explanation

So in the C code:

```c
double cg = gsl_sf_coupling_3j(...);  // computes the Wigner 3j symbol
```

Then:

```c
double phase = pow(-1.0, (two_jp - two_j - 2*q)/2.0);
```

This computes $(-1)^{j'-j-q}$ — note that `two_jp`, `two_j`, and `2q` are used to keep integers (GSL needs integers), then divided by 2 to recover the float exponent.

Finally:

```c
cg *= phase * sqrt(two_jp + 1.0);
```

This multiplies by:

- The phase factor: $(-1)^{j'-j-q}$
- The normalization: $\sqrt{2j'+1}$

Together, this converts the Wigner 3j symbol into the actual **Clebsch–Gordan coefficient** $C^{j',m'}_{j,m;\,k,q'}$ as required by the **Wigner–Eckart theorem**.

✅ Summary

Term	Meaning
`phase`	$(-1)^{j'-j-q}$, phase convention
`sqrt(two_jp + 1.0)`	equals $\sqrt{2j'+1}$, normalization factor
`cg *= ...`	converts Wigner 3j → CG coefficient for use in Wigner–Eckart

Would you like me to write a helper function that wraps this as `gsl_clebsch_gordan(j, m, k, q, j', m')`?