Chapter 4: Answer Key

Two-Electron Integrals and Rys Quadrature Foundations

2302638 Advanced Quantum Chemistry

Department of Chemistry, Chulalongkorn University

Contents

1 Overview

This answer key covers:

- **13 Checkpoint Questions** embedded throughout Chapter 4, testing conceptual understanding of ERIs and the Boys function
- **3 Python Labs** (4A, 4B, 4C) with expected numerical results and validation criteria

Validation standard: All numerical results should match PySCF reference calculations to within the specified tolerances.
Key topics covered:

- ERI definition and 8-fold symmetry
- Schwarz screening and computational cost
- The $(ss|ss)$ primitive ERI formula
- Boys function properties, series expansion, and recursion relations
- Introduction to Rys quadrature concepts

2 Checkpoint Question Answers

This section provides detailed answers to all 13 checkpoint questions from Chapter 4, organized by their location in the chapter.

2.1 Checkpoint 4.1: Mental Model for ERIs

Section 4.1 – ERI Definition

Question: An ERI $(\mu\nu|\lambda\sigma)$ represents the Coulomb interaction between two *overlap distributions*: $\chi_\mu(\mathbf{r})\chi_\nu(\mathbf{r})$ on electron 1 and $\chi_\lambda(\mathbf{r})\chi_\sigma(\mathbf{r})$ on electron 2. Why does a four-index quantity arise from a two-body operator? What physical information is encoded in each pair of indices?
Answer:
Why four indices arise from a two-body operator:
The electron-electron repulsion operator $\hat{V}_{ee} = 1/r_{12}$ acts on *two* electrons simultaneously. In the LCAO expansion, each electron's wavefunction is expanded in the basis $\{\chi_\mu\}$:

- Electron 1: $\psi(\mathbf{r}_1) = \sum_\mu c_\mu \chi_\mu(\mathbf{r}_1)$ and $\psi'(\mathbf{r}_1) = \sum_\nu c'_\nu \chi_\nu(\mathbf{r}_1)$
- Electron 2: $\psi(\mathbf{r}_2) = \sum_\lambda c_\lambda \chi_\lambda(\mathbf{r}_2)$ and $\psi'(\mathbf{r}_2) = \sum_\sigma c'_\sigma \chi_\sigma(\mathbf{r}_2)$

The matrix element $\langle\psi\psi'|r_{12}^{-1}|\psi\psi'\rangle$ requires four basis function indices—two for each electron. This is fundamentally different from one-electron operators (like kinetic energy or nuclear attraction) which only need two indices.
Physical information in each pair of indices:

- **Bra pair (μ, ν):** Describes the *overlap distribution* $\chi_\mu(\mathbf{r})\chi_\nu(\mathbf{r})$ for electron 1. This is a "charge cloud" representing where electron 1 is likely to be found when transitioning between orbitals μ and ν. The distribution is centered between the centers of χ_μ and χ_ν.
- **Ket pair (λ, σ):** Describes the overlap distribution $\chi_\lambda(\mathbf{r})\chi_\sigma(\mathbf{r})$ for electron 2, similarly representing a charge cloud.

The ERI $(\mu\nu|\lambda\sigma)$ is the classical Coulomb repulsion energy between these two charge distributions:

$$(\mu\nu|\lambda\sigma) = \iint \underbrace{\chi_\mu(\mathbf{r}_1)\chi_\nu(\mathbf{r}_1)}_{\text{charge cloud 1}} \frac{1}{r_{12}} \underbrace{\chi_\lambda(\mathbf{r}_2)\chi_\sigma(\mathbf{r}_2)}_{\text{charge cloud 2}} d\mathbf{r}_1 d\mathbf{r}_2$$

Special cases and their role in Fock matrix construction:

- $(\mu\mu|\lambda\lambda)$: Coulomb repulsion between diagonal charge densities $|\chi_\mu|^2$ and $|\chi_\lambda|^2$
- $(\mu\nu|\mu\nu)$ with $\mu \neq \nu$: Coulomb repulsion between the overlap distribution $\chi_\mu\chi_\nu$ on both electrons

Important distinction—Coulomb vs. Exchange contributions: In the AO basis, *all* ERIs represent classical Coulomb repulsions between charge clouds. The terms "Coulomb integral" and "exchange integral" properly refer to MO-basis quantities:

$$J_{ij} = (ii|jj) \qquad \text{(Coulomb integral in MO basis)}$$
$$K_{ij} = (ij|ji) \qquad \text{(Exchange integral in MO basis)}$$

The "exchange energy" has no classical analog not because the integral $(ij|ji)$ itself is non-classical, but because exchange arises from wavefunction antisymmetry (Pauli principle). In the AO basis, the distinction emerges through *contraction patterns*:

$$J_{\mu\nu} = \sum_{\lambda\sigma} (\mu\nu|\lambda\sigma) P_{\lambda\sigma} \qquad \text{(bra indices fixed)}$$
$$K_{\mu\nu} = \sum_{\lambda\sigma} (\mu\lambda|\nu\sigma) P_{\lambda\sigma} \qquad \text{(indices interleaved)}$$

Physical Insight

The "four-index" nature of ERIs is the fundamental reason why electron-electron repulsion is computationally expensive: storing N^4 quantities and contracting them scales as $\mathcal{O}(N^4)$, compared to $\mathcal{O}(N^2)$ for one-electron integrals.

2.2 Checkpoint 4.2: Notation Warning—Chemist's vs. Physicist's

Section 4.2 – Notation Conventions

Question: Physicist's notation uses different index ordering:

$$\langle\mu\lambda|\nu\sigma\rangle_{\text{physicist}} = (\mu\nu|\lambda\sigma)_{\text{chemist}}.$$

In physicist's convention, the first and third indices form the bra; the second and fourth form the ket. This course uses chemist's notation exclusively. Understand the difference.
Answer:
Understanding the difference:

	Chemist's	**Physicist's**
Notation	$(\mu\nu\|\lambda\sigma)$	$\langle\mu\lambda\|\nu\sigma\rangle$
Electron 1 indices	μ,ν (bra)	μ,ν (1st, 2nd)
Electron 2 indices	λ,σ (ket)	λ,σ (3rd, 4th)
Index grouping	(12|34)	⟨13|24⟩

Translation rule:

$$\langle \mu\lambda|\nu\sigma\rangle_{\text{physicist}} = (\mu\nu|\lambda\sigma)_{\text{chemist}}$$

or equivalently:

$$(\mu\nu|\lambda\sigma)_{\text{chemist}} = \langle \mu\lambda|\nu\sigma\rangle_{\text{physicist}}$$

Why chemist's notation is preferred in this course:

1. **Index locality:** Adjacent indices in $(\mu\nu|\lambda\sigma)$ refer to the same electron, making symmetry relations easier to see.
2. **Computational relevance:** Most quantum chemistry codes (including PySCF, Gaussian, Q-Chem) use chemist's notation internally.
3. **Contraction patterns:** The J and K matrix contractions are more transparent:

$$J_{\mu\nu} = \sum_{\lambda\sigma} (\mu\nu|\lambda\sigma) P_{\lambda\sigma} \qquad \text{(adjacent indices paired)}$$
$$K_{\mu\nu} = \sum_{\lambda\sigma} (\mu\lambda|\nu\sigma) P_{\lambda\sigma} \qquad \text{(alternating indices paired)}$$

Common pitfall: When reading physics-oriented textbooks (e.g., Sakurai, Griffiths), ERIs often appear in physicist's notation. Always check the definition before implementing equations.

Warning: Common Error

A sign or factor error when mixing conventions can lead to completely wrong energies. The exchange energy has opposite signs in some formalisms. Always verify the convention being used.

2.3 Checkpoint 4.3: 8-Fold Symmetry

Section 4.3 – ERI Symmetries

Question: Starting from the three fundamental symmetries:

$$(\mu\nu|\lambda\sigma) = (\nu\mu|\lambda\sigma) \qquad \text{(bra swap)}$$
$$(\mu\nu|\lambda\sigma) = (\mu\nu|\sigma\lambda) \qquad \text{(ket swap)}$$
$$(\mu\nu|\lambda\sigma) = (\lambda\sigma|\mu\nu) \qquad \text{(bra-ket exchange)}$$

(a) List all 8 index orderings equivalent to $(\mu\nu|\lambda\sigma)$.

(b) For $N = 100$ basis functions, how many unique ERIs must be stored? Compare to the full $N^4 = 10^8$ tensor.

(c) Why does the bra-ket exchange have the clearest physical interpretation?

Answer:
(a) All 8 equivalent index orderings:
Starting from $(\mu\nu|\lambda\sigma)$, we can apply the three generators:

1. $(\mu\nu|\lambda\sigma)$ (identity)

2. $(\nu\mu|\lambda\sigma)$ (bra swap)
3. $(\mu\nu|\sigma\lambda)$ (ket swap)
4. $(\nu\mu|\sigma\lambda)$ (bra swap + ket swap)
5. $(\lambda\sigma|\mu\nu)$ (bra-ket exchange)
6. $(\sigma\lambda|\mu\nu)$ (bra-ket exchange + bra swap)
7. $(\lambda\sigma|\nu\mu)$ (bra-ket exchange + ket swap)
8. $(\sigma\lambda|\nu\mu)$ (bra-ket exchange + bra swap + ket swap)

These 8 orderings form a group isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2$.
(b) Unique ERIs for $N = 100$:
The exact formula for the number of unique ERIs is:

$$N_{\text{unique}} = \frac{n_{12}(n_{12}+1)}{2}, \quad \text{where } n_{12} = \frac{N(N+1)}{2}$$

This accounts for the triangular packing of pairs within each electron and between electrons.
For $N = 100$:

$$n_{12} = \frac{100 \times 101}{2} = 5050$$

$$N_{\text{unique}} = \frac{5050 \times 5051}{2} = 12{,}753{,}775$$

Comparison:

Storage	Count	Memory (8 bytes each)
Full N^4	10^8	800 MB
8-fold symmetric	1.28×10^7	102 MB
Savings	**Factor of ∼7.8**	

The factor is slightly less than 8 because diagonal cases (where some indices coincide) have fewer distinct permutations.
(c) Physical interpretation of bra-ket exchange:
The bra-ket exchange symmetry $(\mu\nu|\lambda\sigma) = (\lambda\sigma|\mu\nu)$ has the clearest physical meaning because it reflects the **fundamental indistinguishability of electrons**.
Physical reasoning:

- The labels "electron 1" and "electron 2" are arbitrary—electrons have no identity.
- Swapping these labels should not change any observable quantity.
- The Coulomb interaction $1/r_{12} = 1/r_{21}$ is symmetric in the electron coordinates.

In contrast, the bra and ket swaps arise from the *mathematical* property that real functions satisfy $\chi_\mu\chi_\nu = \chi_\nu\chi_\mu$—a statement about the commutativity of multiplication, not about physics.
The bra-ket exchange symmetry is the only one that survives for complex orbitals (with appropriate complex conjugation), while the bra/ket swaps become more complicated.

2.4 Checkpoint 4.4: Cost-Benefit of Screening

Section 4.4 – Schwarz Screening

Question: Schwarz screening requires computing $\mathcal{O}(N^2)$ diagonal integrals $(\mu\nu|\mu\nu)$ before screening begins.

(a) Why is this $\mathcal{O}(N^2)$ overhead worthwhile when the goal is to avoid $\mathcal{O}(N^4)$ work?

(b) If 90% of ERIs are screened away, what is the effective scaling of the remaining work?

(c) For a very compact molecule (all atoms close together), would you expect more or fewer ERIs to survive screening compared to an extended molecule?

Answer:
(a) Why $\mathcal{O}(N^2)$ overhead is worthwhile:
The Schwarz screening overhead is $\mathcal{O}(N^2)$ while the potential savings are from $\mathcal{O}(N^4)$. The ratio of work is:

$$\frac{\text{Screening cost}}{\text{Full ERI cost}} \sim \frac{N^2}{N^4} = \frac{1}{N^2}$$

For $N = 100$: overhead is $\sim 0.01\%$ of full cost. For $N = 1000$: overhead is $\sim 0.0001\%$ of full cost.
Even if screening only eliminates a modest fraction of ERIs, the overhead is negligible for any reasonably sized system. The screening becomes increasingly worthwhile as N grows.
(b) Effective scaling when 90% are screened:
If 90% of ERIs are screened, only 10% survive. The work becomes:

$$W_{\text{screened}} = 0.1 \times N^4 = 0.1N^4$$

This is still formally $\mathcal{O}(N^4)$—the *asymptotic* scaling is unchanged. However, the prefactor is reduced by a factor of 10, which is significant in practice.
Important: The fraction screened typically *increases* with system size for extended molecules, because distant basis pairs have exponentially small Schwarz bounds. In favorable cases (1D chains, surfaces), the number of significant ERIs can scale as $\mathcal{O}(N^2)$ or even $\mathcal{O}(N)$.
(c) Compact vs. extended molecules:
Compact molecules (all atoms within a few Å):

- All basis functions have significant spatial overlap
- Schwarz bounds $Q_{\mu\nu} = \sqrt{(\mu\nu|\mu\nu)}$ are mostly non-negligible
- Few ERIs are screened—most survive
- Effective scaling remains close to $\mathcal{O}(N^4)$

Extended molecules (linear chains, surfaces, large biomolecules):

- Basis functions on distant atoms have negligible overlap
- Schwarz bounds decay exponentially: $Q_{\mu\nu} \sim e^{-\alpha R_{\mu\nu}^2}$
- Many ERIs are screened (often $> 99\%$ for large systems)
- Effective scaling can approach $\mathcal{O}(N^2)$ or better

Physical intuition: Gaussian basis functions are localized. If two functions are separated by many times their "width," their overlap is exponentially small. Extended molecules have many such distant pairs; compact molecules have few.

Key Formula

Schwarz inequality: $|(\mu\nu|\lambda\sigma)| \leq Q_{\mu\nu} \cdot Q_{\lambda\sigma}$ where $Q_{\mu\nu} = \sqrt{(\mu\nu|\mu\nu)}$.
If $Q_{\mu\nu} \cdot Q_{\lambda\sigma} < \tau$ (typically $\tau \sim 10^{-10}$), the ERI is skipped.

2.5 Checkpoint 4.5: Structure of the $(ss|ss)$ Formula

Section 4.5 – Primitive ERI Structure

Question: Examine the $(ss|ss)$ ERI formula:

$$(ab|cd) = \frac{2\pi^{5/2}}{pq\sqrt{p+q}}\, e^{-\mu R_{AB}^2}\, e^{-\nu R_{CD}^2}\, F_0(T), \quad T = \frac{pq}{p+q}|\mathbf{P}-\mathbf{Q}|^2$$

and answer:

(a) Which factors depend *only* on the bra pair $(\mathbf{A}, \mathbf{B}, \alpha, \beta)$?
(b) Which factors depend *only* on the ket pair $(\mathbf{C}, \mathbf{D}, \gamma, \delta)$?
(c) What role does $R_{PQ} = |\mathbf{P}-\mathbf{Q}|$ play? Why does this "effective distance" between pair centers control the ERI magnitude?
(d) If $\mathbf{A} = \mathbf{B}$ (same-center case), what happens to R_{AB} and μ? How does this affect the exponential prefactor?
(e) What happens when $R_{PQ} \to 0$ (pair centers coincide)? What is $F_0(0)$?

Answer:
(a) Bra-pair-only factors:
The following depend only on the bra pair $(\mathbf{A}, \mathbf{B}, \alpha, \beta)$:

- $p = \alpha + \beta$ (total exponent)
- $\mu = \frac{\alpha\beta}{\alpha+\beta}$ (reduced exponent)
- $\mathbf{P} = \frac{\alpha\mathbf{A}+\beta\mathbf{B}}{\alpha+\beta}$ (composite center)
- $R_{AB} = |\mathbf{A}-\mathbf{B}|$ (primitive separation)
- $e^{-\mu R_{AB}^2}$ (intra-pair decay factor)

(b) Ket-pair-only factors:
Similarly, these depend only on the ket pair $(\mathbf{C}, \mathbf{D}, \gamma, \delta)$:

- $q = \gamma + \delta$ (total exponent)
- $\nu = \frac{\gamma\delta}{\gamma+\delta}$ (reduced exponent)
- $\mathbf{Q} = \frac{\gamma\mathbf{C}+\delta\mathbf{D}}{\gamma+\delta}$ (composite center)
- $R_{CD} = |\mathbf{C}-\mathbf{D}|$ (primitive separation)
- $e^{-\nu R_{CD}^2}$ (intra-pair decay factor)

(c) Role of R_{PQ} (inter-pair separation):
The distance $R_{PQ} = |\mathbf{P}-\mathbf{Q}|$ appears in the Boys function argument:

$$T = \rho R_{PQ}^2, \quad \text{where } \rho = \frac{pq}{p+q}$$

Physical interpretation:

- $\mathbf{P}$ is the "center of charge" for the electron-1 overlap distribution $\chi_a\chi_b$
- $\mathbf{Q}$ is the "center of charge" for the electron-2 overlap distribution $\chi_c\chi_d$
- R_{PQ} measures how far apart these two charge clouds are
- The Boys function $F_0(T)$ encodes the Coulomb interaction decay with separation

As R_{PQ} increases:

- $T = \rho R_{PQ}^2$ grows
- $F_0(T) \to \frac{1}{2}\sqrt{\pi/T} \sim 1/R_{PQ}$ for large T
- The ERI decays as $\sim 1/R_{PQ}$ (the familiar Coulomb $1/r$ decay)

(d) Same-center case ($\mathbf{A} = \mathbf{B}$):
When $\mathbf{A} = \mathbf{B}$:

- $R_{AB} = 0$
- The exponential becomes $e^{-\mu R_{AB}^2} = e^0 = 1$
- The composite center $\mathbf{P} = \frac{\alpha\mathbf{A}+\beta\mathbf{A}}{\alpha+\beta} = \mathbf{A}$

This is the "diagonal" case where both primitives are on the same atom. The intra-pair decay factor is absent, and the ERI depends only on the ket-pair geometry and the inter-pair separation.
(e) Coinciding pair centers ($R_{PQ} \to 0$):
When $\mathbf{P} = \mathbf{Q}$:

- $R_{PQ} = 0$, so $T = \rho \cdot 0 = 0$
- $F_0(0) = 1$ (from the definition: $\int_0^1 1 \cdot e^0\, dt = 1$)

The ERI reduces to:

$$(ab|cd)\big|_{R_{PQ}=0} = \frac{2\pi^{5/2}}{pq\sqrt{p+q}}\, e^{-\mu R_{AB}^2}\, e^{-\nu R_{CD}^2}$$

This is finite and represents the maximum Coulomb repulsion when the two overlap distributions are centered at the same point.

Physical Insight

The $(ss|ss)$ formula cleanly separates:

1. **Intra-pair factors** ($e^{-\mu R_{AB}^2}$, $e^{-\nu R_{CD}^2}$): Measure how "spread out" each overlap distribution is based on primitive separations.
2. **Inter-pair factor** ($F_0(T)$): Measures the Coulomb repulsion between the two distributions based on their center-to-center separation.

2.6 Checkpoint 4.6: Boys Function Properties

Section 4.6 – Boys Function

Question:

(a) Verify that $F_n(0) = 1/(2n+1)$ for $n = 0, 1, 2, 3$ by evaluating the integral directly.
(b) For $T = 1.0$, estimate $F_0(1)$ using the erf formula. (Hint: $\mathrm{erf}(1) \approx 0.8427$.)
(c) The derivative identity states $\frac{d}{dT}F_n(T) = -F_{n+1}(T)$. Why is this derivative always negative? What does it say about how $F_n(T)$ changes with T?
(d) At $T = 0.001$, estimate $(2 \cdot 0 + 1)F_0(T) - e^{-T}$. Why does upward recursion suffer from cancellation here?
(e) If your implementation returns negative values for large n, what is the likely bug?

Answer:
(a) Verify $F_n(0) = 1/(2n+1)$:
At $T = 0$, the Boys function becomes:

$$F_n(0) = \int_0^1 t^{2n} e^0 \, dt = \int_0^1 t^{2n} \, dt = \left[\frac{t^{2n+1}}{2n+1} \right]_0^1 = \frac{1}{2n+1}$$

Explicit values:

n	$F_n(0)$	Decimal
0	$1/1 = 1$	1.0000
1	$1/3$	0.3333
2	$1/5$	0.2000
3	$1/7$	0.1429

(b) Estimate $F_0(1)$ using the erf formula:
The closed form is:

$$F_0(T) = \frac{1}{2}\sqrt{\frac{\pi}{T}} \, \mathrm{erf}(\sqrt{T})$$

At $T = 1$:

$$F_0(1) = \frac{1}{2}\sqrt{\frac{\pi}{1}} \, \mathrm{erf}(1) = \frac{\sqrt{\pi}}{2} \times 0.8427 \approx \frac{1.7725}{2} \times 0.8427 \approx 0.7468$$

(Exact value: $F_0(1) \approx 0.746824133$)
(c) Why is $\frac{d}{dT}F_n(T) = -F_{n+1}(T)$ always negative?
The derivative identity follows from differentiating under the integral:

$$\frac{d}{dT}F_n(T) = \frac{d}{dT}\int_0^1 t^{2n} e^{-Tt^2} \, dt = \int_0^1 t^{2n}(-t^2) e^{-Tt^2} \, dt = -\int_0^1 t^{2n+2} e^{-Tt^2} \, dt = -F_{n+1}(T)$$

Since $F_{n+1}(T) > 0$ for all $T \geq 0$ (the integrand is non-negative and not identically zero), we have:

$$\frac{d}{dT}F_n(T) < 0 \quad \text{for all } T \geq 0$$

Physical meaning: $F_n(T)$ is monotonically *decreasing* in T. As T increases (pair centers separate), the Boys function decreases, reflecting the decay of the Coulomb interaction.
(d) Cancellation in upward recursion at small T:
At $T = 0.001$, we compute the numerator of the upward recursion:

$$(2n+1)F_n(T) - e^{-T}$$

For $n = 0$:

$$F_0(0.001) \approx F_0(0) - 0.001 \cdot F_1(0) + \ldots \approx 1 - \frac{0.001}{3} \approx 0.9997$$
$$e^{-0.001} \approx 1 - 0.001 + \frac{0.001^2}{2} \approx 0.999$$

The numerator:

$$1 \cdot 0.9997 - 0.999 \approx 0.0007$$

We are subtracting two numbers that are both close to 1 to get a result of order 10^{-4}. This loses approximately 3–4 significant digits.
For larger n, the cancellation worsens because $(2n+1)F_n(T)$ and e^{-T} approach each other more closely.
(e) Negative values for large n:
If your Boys function returns negative values for large n, the likely bug is **using upward recursion at small T**.
The upward recursion:

$$F_{n+1}(T) = \frac{(2n+1)F_n(T) - e^{-T}}{2T}$$

suffers from catastrophic cancellation when T is small. The numerator becomes a small difference of nearly equal numbers, and dividing by the small $2T$ amplifies rounding errors. After several iterations, errors accumulate and can produce negative (unphysical) values.
Fix: Use the series expansion for small T (typically $T < 25$), which is numerically stable.

Key Formula

Boys function recursions:

$$\begin{aligned}
&\text{Upward:} && F_{n+1}(T) = \frac{(2n+1)F_n(T) - e^{-T}}{2T} && \text{(stable for large } T\text{)}\\
&\text{Downward:} && F_n(T) = \frac{2TF_{n+1}(T) + e^{-T}}{2n+1} && \text{(always stable)}\\
&\text{Series:} && F_n(T) = \sum_{k=0}^{\infty} \frac{(-T)^k}{k!(2n+2k+1)} && \text{(stable for small } T\text{)}
\end{aligned}$$

2.7 Checkpoint 4.7: Quadrature Comparison

Section 4.7 – Numerical Quadrature

Question:

(a) The 2-point trapezoidal rule is exact for polynomials of degree at most 1. What degree does 2-point Gaussian quadrature achieve?
(b) Simpson's rule uses 3 equally spaced points and is exact for cubics (degree 3). How many Gaussian nodes would you need to match this?
(c) Why might Newton–Cotes rules still be preferred in some applications despite their lower accuracy per node?

Answer:
(a) Degree achieved by 2-point Gaussian quadrature:
Two-point Gaussian quadrature achieves **degree 3**—it exactly integrates all polynomials

up to degree 3.
An n-point Gaussian quadrature rule has $2n$ free parameters (n nodes + n weights). With $n = 2$ points, we have 4 parameters. Requiring the rule to exactly integrate monomials $1, x, x^2, x^3$ gives 4 equations (the moment conditions):

$$\sum_i W_i = \int_a^b 1\,dx$$

$$\sum_i W_i x_i = \int_a^b x\,dx$$

$$\sum_i W_i x_i^2 = \int_a^b x^2\,dx$$

$$\sum_i W_i x_i^3 = \int_a^b x^3\,dx$$

With 4 unknowns and 4 equations, the system is exactly determined. The general rule is: n-point Gaussian quadrature achieves degree $2n-1$.

Rule	Points	Degree	Polynomial Exactness
Trapezoidal	2	1	$1, x$
2-point Gauss	2	3	$1, x, x^2, x^3$

The Gaussian rule is **twice as efficient** per node because it optimizes both node locations and weights, rather than fixing nodes at endpoints.
(b) Gaussian nodes to match Simpson's degree 3:
Two Gaussian nodes suffice to match Simpson's rule exactness (degree 3).
Simpson's rule uses 3 equally spaced points and is exact for polynomials up to degree 3. Gaussian quadrature with n points achieves degree $2n-1$. We need:

$$2n - 1 \geq 3 \quad \Rightarrow \quad n \geq 2$$

Thus 2 Gaussian nodes achieve the same polynomial degree as 3 Simpson nodes.
Physical insight for quantum chemistry: This efficiency gain is precisely why Rys quadrature is valuable for ERIs. Computing Rys nodes/weights has overhead, but using fewer quadrature points for the same accuracy pays off when evaluating millions of integrals. For a shell quartet requiring F_0 through F_3 (degree 3 polynomials in $x = t^2$), 2 Rys roots suffice where generic quadrature might need 3+ points.
(c) When Newton–Cotes rules are still preferred:
Newton–Cotes rules (equally spaced nodes) may be preferred over Gaussian quadrature in several scenarios:

1. **Tabulated or sampled data:** When function values are available only at fixed grid points (experimental measurements, pre-computed tables), you cannot choose optimal node locations. Newton–Cotes rules work directly with the available data.

2. **Adaptive/composite integration:** For adaptive quadrature algorithms that subdivide intervals, equally spaced nodes allow efficient reuse of function evaluations when refining. Gauss nodes from the original interval do not coincide with Gauss nodes of subintervals.

3. **Error estimation:** Newton–Cotes rules of different orders on the same grid (e.g., comparing trapezoidal and Simpson's on the same points) provide straightforward error estimates via Richardson extrapolation. Gaussian rules require different node sets for different orders.

4. **Simplicity and implementation:** Newton–Cotes formulas have simple, fixed weights (e.g., $\frac{1}{3}, \frac{4}{3}, \frac{1}{3}$ for Simpson's). Gaussian weights must be computed or looked up from tables.

5. **Singular or oscillatory integrands at known locations:** If the integrand has special behavior at known locations, strategically placed fixed nodes may capture it better than automatically optimized Gaussian nodes.

In quantum chemistry: Newton–Cotes methods appear in DFT grid integration (exchange-correlation functionals), real-space methods with regular grids, and time propagation with fixed time steps. Rys/Gaussian quadrature dominates ERI evaluation because the analytical integrand structure is known in advance, allowing optimal quadrature design.

Key Formula

Quadrature efficiency comparison:

Method	n points	Polynomial degree
Newton–Cotes (open)	n	$n-1$ or n
Gaussian	n	$2n-1$

Gaussian quadrature is optimal: no quadrature rule with n points can achieve degree $> 2n-1$.

2.8 Checkpoint 4.8: Understanding the Moment Transformation

Section 4.7 – Boys as Moments

Question:

(a) Verify that $m_0(T) = 2F_0(T)$ by direct integration when $T = 0$. What value do you get?

(b) The weight function $w_T(x) = x^{-1/2}e^{-Tx}$ has a singularity at $x = 0$. Is this problematic for integration?

(c) What happens to all moments $m_n(T)$ as $T \to \infty$? How does this relate to the physical meaning of $T = \rho \cdot R_{PQ}^2$?

(d) Why is rewriting $F_n(T)$ as a weighted moment useful? What does it enable that direct evaluation does not?

Answer:

(a) Verify $m_0(T) = 2F_0(T)$ at $T = 0$:

At $T = 0$, we compute $m_0(0)$ directly from the moment definition:

$$m_0(0) = \int_0^1 x^0 \cdot w_0(x)\,dx = \int_0^1 x^{-1/2} \cdot e^0\,dx = \int_0^1 x^{-1/2}\,dx = \left[2x^{1/2}\right]_0^1 = 2$$

And from the Boys function:

$$F_0(0) = \int_0^1 t^0 \cdot e^0 \, dt = \int_0^1 1 \, dt = 1$$

Therefore: $\boxed{m_0(0) = 2 = 2 \cdot F_0(0)}$, confirming the relation $m_n(T) = 2F_n(T)$.

Physical interpretation: The factor of 2 arises from the Jacobian of the transformation $x = t^2$. The moment m_0 represents the total "weight" under the curve $w_T(x) = x^{-1/2}e^{-Tx}$ over $[0, 1]$. At $T = 0$, this equals 2, while $F_0(0) = 1$ because the t-variable integrand is simply 1 over the unit interval.

(b) Is the singularity at $x = 0$ problematic?

No, the singularity is integrable and poses no fundamental problem.

The weight function $w_T(x) = x^{-1/2}e^{-Tx}$ diverges as $x \to 0$. However, when computing moments:

$$m_n(T) = \int_0^1 x^n \cdot x^{-1/2} \cdot e^{-Tx} \, dx = \int_0^1 x^{n-1/2} e^{-Tx} \, dx$$

The combined power is $x^{n-1/2}$. Near $x = 0$, the exponential $e^{-Tx} \approx 1$, so the integrand behaves like $x^{n-1/2}$.

Integrability criterion: An integrand x^α is integrable at $x = 0$ if and only if $\alpha > -1$. For $n \geq 0$:

- $n = 0$: exponent $= -1/2 > -1 \quad \Rightarrow$ **converges**
- $n = 1$: exponent $= 1/2 > -1 \quad \Rightarrow$ **converges**
- $n \geq 2$: exponent $= n - 1/2 > 0 \quad \Rightarrow$ **converges**

Since $-1/2 > -1$, all moments $m_n(T)$ for $n \geq 0$ are finite.

Connection to Gaussian quadrature: The singularity is "absorbed" into the weight function. The quadrature rule approximates $\int f(x)w(x)\,dx$ by $\sum W_i f(x_i)$, where $f(x)$ is assumed regular. The singular behavior is captured in the weights W_i, not sampled directly. Rys quadrature nodes are strictly in $(0, 1)$, never at the boundary.

(c) Behavior of moments as $T \to \infty$:

All moments $m_n(T) \to 0$ as $T \to \infty$.

For large T, the exponential e^{-Tx} decays rapidly for any $x > 0$. Only the region near $x = 0$ contributes significantly. Using asymptotic analysis:

$$m_n(T) = \int_0^1 x^{n-1/2} e^{-Tx} \, dx \sim T^{-(n+1/2)}\Gamma(n + \tfrac{1}{2}) \quad \text{as } T \to \infty$$

Thus $m_n(T) \sim C \cdot T^{-(n+1/2)}$, which vanishes as $T \to \infty$.

Physical meaning via $T = \rho \cdot R_{PQ}^2$:

The parameter T encodes the scaled squared distance between the composite centers $\mathbf{P}$ and $\mathbf{Q}$:

- **Small T (close distributions):** R_{PQ} is small, electron pairs overlap significantly, Coulomb interaction is strong. Moments are large ($m_0(0) = 2$).
- **Large T (distant distributions):** R_{PQ} is large, electron pairs are well-separated, Coulomb interaction weakens. Moments decay to zero.

This is the mathematical expression of fundamental physics: **Coulomb repulsion decreases with distance**. The Boys function $F_0(T) \sim \frac{1}{2}\sqrt{\pi/T}$ for large T, giving the familiar $1/R$ decay from classical electrostatics.

Numerical implication: For very large T (distant pairs), the ERI approaches zero. This is the basis for Schwarz screening.

(d) Why rewriting $F_n(T)$ as moments is useful:

The moment interpretation enables the powerful machinery of **Gaussian quadrature**, providing several critical advantages:

1. **Simultaneous evaluation of multiple $F_n(T)$:**

   Direct methods compute each $F_n(T)$ independently. The moment interpretation yields:

   $$F_n(T) = \frac{1}{2}\sum_{i=1}^{n_r} W_i \cdot x_i^n$$

   Once the n_r Rys nodes $\{x_i\}$ and weights $\{W_i\}$ are computed for a given T, **all** Boys values $F_0(T), F_1(T), \ldots, F_{2n_r-1}(T)$ are obtained simultaneously as simple polynomials in the nodes.

2. **Exact reproduction with minimal points:**

   The quadrature is not approximate—it exactly reproduces the required polynomial moments. With n_r points, we get $2n_r - 1$ exact values. This mathematical guarantee means no truncation error (only roundoff from computing nodes/weights).

3. **Vectorization and parallelization:**

   The form $\sum W_i f(x_i)$ is highly amenable to SIMD vectorization. Modern integral libraries exploit this structure for performance.

4. **Enables Cartesian factorization (DRK algorithm):**

   The Rys quadrature framework leads to the Dupuis–Rys–King factorization:

   $$(ab|cd) = \sum_{i=1}^{n_r} w_i \cdot I_x(t_i) \cdot I_y(t_i) \cdot I_z(t_i)$$

   This separates the 6D integral into products of 1D integrals. Without the moment perspective, this elegant factorization would not be apparent.

5. **Uniform stability across all T:**

   Unlike upward recursion (unstable at small T) or series (slow at large T), the moment-quadrature approach is uniformly stable. Nodes and weights are computed by numerically stable eigenvalue methods (Golub–Welsch).

Summary: The moment transformation is not merely a mathematical curiosity; it is the conceptual foundation that makes Rys quadrature practical. It transforms the problem from "evaluate a special function" to "find optimal quadrature points"—a well-understood problem with robust numerical solutions.

Physical Insight

The moment transformation $F_n(T) = \frac{1}{2}m_n(T)$ reveals the deep connection between Boys functions and Gaussian quadrature:

- Boys functions are moments of a specific weight function $w_T(x) = x^{-1/2}e^{-Tx}$

- Gaussian quadrature exactly reproduces moments using minimal sample points
- This enables the Rys algorithm: compute a few roots/weights, then evaluate all needed Boys functions by polynomial evaluation

The entire Rys quadrature machinery for ERIs rests on this moment interpretation.

2.9 Checkpoint 4.9: Understanding "Exactness" in Quadrature

Section 4.7 – Rys Quadrature Exactness

Question: The claim that Rys quadrature is "exact" may seem surprising.

(a) Gaussian quadrature with n_r points is exact for polynomials of degree $\leq 2n_r - 1$. How does this relate to reproducing Boys functions $F_0(T), F_1(T), \ldots$?
(b) If you need $F_0(T)$ through $F_3(T)$, how many quadrature points suffice?
(c) Why do the nodes and weights depend on T? What would happen if you used fixed nodes/weights for all T values?
(d) In what sense is Rys quadrature *not* approximate? In what sense is it still a "numerical method"?

Answer:
(a) Connection between polynomial exactness and Boys functions:
The Boys function can be written as a moment:

$$F_n(T) = \frac{1}{2}\int_0^1 x^n\, w_T(x)\, dx, \quad w_T(x) = x^{-1/2}e^{-Tx}$$

where $x = t^2$.
For the weight function $w_T(x)$, Gaussian quadrature with n_r points gives nodes $\{x_i\}$ and weights $\{W_i\}$ such that:

$$\int_0^1 f(x)\, w_T(x)\, dx = \sum_{i=1}^{n_r} W_i f(x_i) \quad \text{exactly for } f(x) = \text{polynomial of degree} \leq 2n_r - 1$$

Since $F_n(T) = \frac{1}{2}\int_0^1 x^n w_T(x)\, dx$ and x^n is a polynomial of degree n, we can reproduce $F_n(T)$ exactly for all $n \leq 2n_r - 1$.
(b) Points needed for F_0 through F_3:
We need $n_{\max} = 3$, so we require:

$$2n_r - 1 \geq n_{\max} = 3 \quad \Rightarrow \quad n_r \geq 2$$

With $n_r = 2$ points, we can exactly reproduce polynomials up to degree 3, which covers F_0, F_1, F_2, F_3 (since x^0, x^1, x^2, x^3 all have degree ≤ 3).
(c) Why nodes and weights depend on T:
The weight function $w_T(x) = x^{-1/2}e^{-Tx}$ explicitly depends on T. Gaussian quadrature nodes and weights are determined by the moments of the weight function:

$$m_k(T) = \int_0^1 x^k w_T(x)\, dx = 2F_k(T)$$

Different T values give different moments, hence different orthogonal polynomials, and hence different nodes and weights.

What if we used fixed nodes/weights?
If we tried to use fixed (say, Gauss-Legendre) nodes and weights, the quadrature would only be approximate. The error would depend on how far the integrand deviates from a polynomial. For the Boys integrand $t^{2n}e^{-Tt^2}$:

- At small T: The exponential ≈ 1, and the integrand is polynomial-like—fixed quadrature works reasonably.
- At large T: The exponential decays sharply near $t = 0$, far from polynomial behavior—fixed quadrature requires many points.

Rys quadrature adapts to T, achieving exactness with minimal points.
(d) Exact vs. numerical:
In what sense is Rys quadrature exact?

- It reproduces the required Boys function values *exactly* (to machine precision) with finitely many points.
- There is no truncation error in the sense of "more points = better approximation."
- The exactness is algebraic: moment equations are satisfied exactly.

In what sense is it numerical?

- The nodes and weights must be *computed* for each T, typically by numerical algorithms (eigenvalue solvers for the Jacobi matrix).
- Rounding errors in computing nodes/weights introduce small errors ($\sim 10^{-14}$).
- The method is implemented in floating-point arithmetic, not symbolic algebra.

Bottom line: Rys quadrature is a mathematically exact formula that must be evaluated numerically. The only errors are roundoff, not approximation.

2.10 Checkpoint 4.10: Root Count Practice

Section 4.7 – Root Count Rule

Question: Using the rule $n_{\text{roots}} = \lfloor L/2 \rfloor + 1$ where $L = \ell_A + \ell_B + \ell_C + \ell_D$:

(a) How many Rys roots are needed for a $(dd|pp)$ shell quartet? (Hint: $\ell_d = 2$, $\ell_p = 1$.)
(b) How many roots for $(ff|ff)$? ($\ell_f = 3$.)
(c) For a given total angular momentum L, what is the maximum order n of the Boys function $F_n(T)$ that appears in the ERI evaluation?
(d) Why does n_{roots} grow only as $\lfloor L/2 \rfloor + 1$ rather than linearly with L?

Answer:
(a) $(dd|pp)$ shell quartet:

$$L = \ell_d + \ell_d + \ell_p + \ell_p = 2 + 2 + 1 + 1 = 6$$

$$n_{\text{roots}} = \left\lfloor \frac{6}{2} \right\rfloor + 1 = 3 + 1 = 4$$

(b) $(ff|ff)$ shell quartet:

$$L = \ell_f + \ell_f + \ell_f + \ell_f = 3 + 3 + 3 + 3 = 12$$

$$n_{\text{roots}} = \left\lfloor \frac{12}{2} \right\rfloor + 1 = 6 + 1 = 7$$

Complete table:

Shell quartet	$\ell_A + \ell_B + \ell_C + \ell_D$	L	$n_{\max}$	n_{roots}
$(ss\|ss)$	$0+0+0+0$	0	0	1
$(ps\|ss)$	$1+0+0+0$	1	0	1
$(pp\|ss)$	$1+1+0+0$	2	1	2
$(pp\|pp)$	$1+1+1+1$	4	2	3
$(dd\|pp)$	$2+2+1+1$	6	3	4
$(dd\|dd)$	$2+2+2+2$	8	4	5
$(ff\|ff)$	$3+3+3+3$	12	6	7

(c) Maximum Boys function order:
The maximum order of the Boys function needed is:

$$n_{\max} = \left\lfloor \frac{L}{2} \right\rfloor$$

This arises because the ERI recursion relations (Obara-Saika or Rys) generate Boys functions up to this order when building angular momentum from the primitive $(ss|ss)$ integral.
(d) Why n_{roots} grows as $\lfloor L/2 \rfloor + 1$, not $L + 1$:
The key insight is that Rys quadrature replaces a sum over Boys functions with a sum over roots:

$$\sum_{n=0}^{n_{\max}} c_n F_n(T) = \sum_{i=1}^{n_r} w_i \left(\sum_{n=0}^{n_{\max}} c_n t_i^{2n} \right)$$

With n_r quadrature points, the rule reproduces polynomials in $x = t^2$ up to degree $2n_r - 1$. Since $F_n(T)$ involves x^n, we need:

$$n_{\max} \le 2n_r - 1 \quad \Rightarrow \quad n_r \ge \frac{n_{\max} + 1}{2}$$

This gives roughly half as many roots as the maximum Boys order. The factor of 2 arises because Gaussian quadrature is "doubly efficient"—n points exactly integrate polynomials of degree $2n - 1$, not just $n - 1$.
Additionally, in the Rys formalism, the ERI is factored into 1D integrals that involve even powers of the quadrature variable t. This even-power structure is what allows the floor function and the factor of 2 savings.

2.11 Checkpoint 4.11: Numerical Stability of Boys Evaluation

Section 4.8 – Numerical Stability

Question: Test your Boys function implementation:

(a) Evaluate $F_n(T)$ at $T = 10^{-k}$ for $k = 2, 4, 6, 8, 10$ and compare to the exact limit $F_n(0) = 1/(2n + 1)$. Does your implementation remain stable as $T \to 0$?
(b) Try $n = 5$ and $T = 0.001$. Compare the result from: (i) series only, (ii) F_0 via erf + upward recursion. Which is more accurate?
(c) At what value of T does the upward recursion start losing significant digits?

Answer:

(a) Stability as $T \to 0$:
Expected values for various T approaching 0:

T	$F_0(T)$	$F_1(T)$	$F_2(T)$	$F_3(T)$
10^{-2}	0.99003	0.33003	0.19803	0.14143
10^{-4}	0.99990	0.33330	0.19998	0.14284
10^{-6}	0.999999	0.333333	0.199999	0.142857
10^{-8}	1.000000	0.333333	0.200000	0.142857
10^{-10}	1.000000	0.333333	0.200000	0.142857
$T = 0$ (exact)	1.000000	0.333333	0.200000	0.142857

A properly implemented Boys function using series expansion for small T should smoothly approach the exact limits. If using upward recursion from erf at small T, the values may show numerical instability (oscillation, wrong sign, or large errors for higher n).
(b) Comparison at $n = 5$, $T = 0.001$:
Exact value: $F_5(0.001) \approx 0.090901$ (very close to $F_5(0) = 1/11 \approx 0.090909$)

Method	Value	Relative Error
Series (50 terms)	0.0909008265	$< 10^{-10}$
erf + upward recursion	(varies wildly)	$\sim 10^{-2}$ to unstable

The series expansion is far more accurate at small T. Upward recursion accumulates cancellation errors at each step.
(c) When does upward recursion lose accuracy?
The critical quantity is the numerator of the recursion:

$$(2n+1)F_n(T) - e^{-T}$$

This suffers catastrophic cancellation when both terms are nearly equal. This occurs when:

- T is small (so $e^{-T} \approx 1$ and $F_n(T) \approx 1/(2n+1)$)
- n is moderate to large (so $(2n+1) \cdot \frac{1}{2n+1} = 1 \approx e^{-T}$)

Empirical observation: Upward recursion typically starts losing significant digits when $T \lesssim 25$. For $n \leq 6$ and $T > 30$, upward recursion is usually stable to machine precision.
Practical threshold: Use series for $T < 25$ (or $T < 30$), and erf + upward recursion for larger T.

2.12 Checkpoint 4.12: Understanding the $(ss|ss)$ ERI

Section 4.8 – ERI Implementation

Question: Examine your implementation and the PySCF output:

(a) In the ERI formula, identify which terms control the "intra-pair" decay (primitives on the same center) vs the "inter-pair" decay (interaction between the two electron distributions).
(b) Modify the code to use $R = 5.0$ Bohr. How does this change $T = \rho R_{PQ}^2$ and the resulting ERI? Is the change consistent with $F_0(T)$ decreasing for larger T?

(c) If you accidentally omit the normalization constants $N_s(\alpha)$, etc., what error magnitude would you expect for typical exponents?

Answer:
(a) Intra-pair vs. inter-pair decay:
The $(ss|ss)$ formula:

$$(ab|cd) = \frac{2\pi^{5/2}}{pq\sqrt{p+q}} \cdot e^{-\mu R_{AB}^2} \cdot e^{-\nu R_{CD}^2} \cdot F_0(T)$$

Intra-pair decay (how spread out each pair is):

- $e^{-\mu R_{AB}^2}$ where $\mu = \frac{\alpha\beta}{\alpha+\beta}$, $R_{AB} = |\mathbf{A} - \mathbf{B}|$
- Controls decay based on separation of primitives *within* the bra pair
- $e^{-\nu R_{CD}^2}$ where $\nu = \frac{\gamma\delta}{\gamma+\delta}$, $R_{CD} = |\mathbf{C} - \mathbf{D}|$
- Controls decay based on separation of primitives *within* the ket pair

Inter-pair decay (Coulomb interaction between distributions):

- $F_0(T)$ where $T = \rho R_{PQ}^2$, $\rho = \frac{pq}{p+q}$, $R_{PQ} = |\mathbf{P} - \mathbf{Q}|$
- Controls decay based on separation *between* the two pair centers
- This is the true Coulomb $1/r$ decay at long range

(b) Effect of increasing R from 1.4 to 5.0 Bohr:
With the Lab 4B setup ($\alpha = 0.5$, $\beta = 0.4$, same-center primitives):
At $R = 1.4$ Bohr:

- $R_{AB} = 0$, $R_{CD} = 0$ (both on same center)
- $R_{PQ} = R = 1.4$ Bohr
- $p = 0.9$, $q = 0.8$, $\rho = 0.9 \times 0.8/1.7 \approx 0.424$
- $T = 0.424 \times 1.96 \approx 0.83$
- $F_0(0.83) \approx 0.78$

At $R = 5.0$ Bohr:

- $R_{PQ} = 5.0$ Bohr
- $T = 0.424 \times 25 \approx 10.6$
- $F_0(10.6) \approx 0.27$

The ERI decreases significantly because $F_0(T)$ is monotonically decreasing.
Expected ratio:

$$\frac{\text{ERI at } R = 5.0}{\text{ERI at } R = 1.4} \approx \frac{F_0(10.6)}{F_0(0.83)} \approx \frac{0.27}{0.78} \approx 0.35$$

This is consistent with the $1/R$ Coulomb decay at long range.
(c) Error from omitting normalization:
The normalization constant for an s-type Gaussian is:

$$N_s(\alpha) = \left(\frac{2\alpha}{\pi}\right)^{3/4}$$

For typical exponents:

α	$N_s(\alpha)$
0.1	0.283
0.5	0.713
1.0	1.008
2.0	1.426

The $(ss|ss)$ ERI includes $N_s(\alpha)N_s(\beta)N_s(\gamma)N_s(\delta)$—a product of four normalization constants.
For $\alpha = \beta = \gamma = \delta = 0.5$:

$$N_s(0.5)^4 \approx (0.713)^4 \approx 0.26$$

Expected error: Omitting normalization gives values too large by a factor of ~ 4 (i.e., divide the wrong answer by 0.26 to get the right answer). For exponents around 0.5, the error is a factor of 3–5.
For more extreme exponents (very tight or very diffuse), the error can be larger.

2.13 Checkpoint 4.13: Quadrature Difficulty vs. Parameter T

Section 4.8 – Quadrature Comparison

Question: Consider approximating $F_n(T) = \int_0^1 t^{2n} e^{-Tt^2}\, dt$ with generic quadrature:

(a) Sketch the integrand $t^{2n}e^{-Tt^2}$ for $n = 2$ at $T = 0.1$, $T = 1$, and $T = 10$. How does the "shape" change?
(b) For which T regime is the integrand nearly polynomial (hence easy for Gauss–Legendre)?
(c) For which T regime is the integrand sharply peaked near $t = 0$? Why does this make generic quadrature less efficient?
(d) Why is Rys quadrature "customized" for this specific integrand structure?

Answer:
(a) Integrand shape for $n = 2$ at different T:
The integrand is $f(t) = t^4 e^{-Tt^2}$ on $[0, 1]$.
At $T = 0.1$:

- $e^{-0.1t^2} \approx 1 - 0.1t^2$ (nearly constant, close to 1)
- Integrand $\approx t^4$—polynomial-like
- Smooth curve rising from 0 to ≈ 0.9 at $t = 1$

At $T = 1$:

- e^{-t^2} decays from 1 to $e^{-1} \approx 0.37$
- Integrand has a broad hump, maximum around $t \approx 0.7$
- Peak value ≈ 0.15

At $T = 10$:

- e^{-10t^2} decays rapidly: $e^{-0.1} \approx 0.9$ at $t = 0.1$, $e^{-10} \approx 4.5 \times 10^{-5}$ at $t = 1$
- Integrand is sharply peaked near $t = 0$

- Most of the integral comes from $t < 0.3$

(b) Polynomial-like regime:
For **small** T (roughly $T < 1$), the exponential $e^{-Tt^2} \approx 1$ over most of the interval, and the integrand is well-approximated by the polynomial t^{2n}.
Gauss–Legendre quadrature is optimized for polynomials (with constant weight function), so it performs well in this regime. A modest number of points (say, 10–20) gives high accuracy.
(c) Sharply peaked regime:
For **large** T (roughly $T > 10$), the exponential decay concentrates the integrand near $t = 0$. The integrand looks like a narrow spike near the origin.
Gauss–Legendre nodes are distributed fairly uniformly across $[0, 1]$, but most of the integral comes from a small region near $t = 0$. The quadrature "wastes" many points in regions where the integrand is negligible.
To capture the spike accurately, Gauss–Legendre needs many points (64–100+). The number of required points grows with T.
(d) Why Rys quadrature is customized:
Rys quadrature constructs nodes and weights specifically for the weight function $w_T(x) = x^{-1/2}e^{-Tx}$ on $[0, 1]$.
Key features:

1. **T-dependent nodes:** As T increases, Rys nodes move toward $x = 0$ (equivalently $t = 0$), automatically concentrating sampling where the integrand is significant.
2. **T-dependent weights:** Weights adjust to account for the exponential decay, so that the product $w_i \cdot f(x_i)$ correctly captures contributions from different regions.
3. **Minimal points for exactness:** For reproducing $F_0, \ldots, F_{n_{\max}}$, Rys quadrature uses only $n_{\max} + 1$ points (approximately), regardless of T. Generic quadrature would need many more points for large T.

Analogy: Rys quadrature is like using a custom-fitted glove for each hand, while Gauss–Legendre is like using a one-size-fits-all glove that works reasonably for average hands but poorly for unusual shapes.

Physical Insight

The parameter $T = \rho R_{PQ}^2$ directly encodes the physical separation of charge distributions:

- Small T (close distributions): The integrand is smooth, and generic quadrature works fine.
- Large T (distant distributions): The integrand is highly localized, and Rys quadrature's adaptive nodes are essential for efficiency.

This is why Rys quadrature is standard in molecular integral codes—it handles the full range of interatomic distances efficiently.

3 Lab Solutions

3.1 Lab 4A: Implement $F_n(T)$ with Series + Recursion

Lab 4A Objectives

- Implement a numerically stable Boys function evaluator
- Handle both small-T (series) and large-T (erf + recursion) regimes
- Verify accuracy against analytical limits

Expected numerical results:

n	T	$F_n(T)$	Notes
0	0.0	1.000000000	Exact: $1/(2 \cdot 0 + 1) = 1$
1	0.0	0.333333333	Exact: $1/3$
2	0.0	0.200000000	Exact: $1/5$
3	0.0	0.142857143	Exact: $1/7$
0	1.0	0.746824133	Via erf formula
0	10.0	0.279694775	Large T
2	1.0	0.130691379	
2	10.0	0.012614227	

Complete solution code:

```python
import math

def boys(n: int, T: float, T_switch: float = 25.0, nseries: int = 50)
    -> float:
    """
    Boys function F_n(T) = integral_0^1 t^(2n) exp(-T t^2) dt.

    Strategy:
      - T < T_switch: truncated series (stable for all n)
      - T >= T_switch: F_0 from erf + upward recursion (stable for
        large T)
    """
    if T < T_switch:
        # F_n(T) = sum_{k>=0} (-T)^k / (k! (2n + 2k + 1))
        val = 0.0
        term = 1.0
        for k in range(nseries):
            val += term / (2*n + 2*k + 1)
            if abs(term) < 1e-16 * abs(val):  # Early termination
                break
            term *= -T / (k + 1)
        return val

    # F0(T) from erf
    F = 0.5 * math.sqrt(math.pi / T) * math.erf(math.sqrt(T))
    if n == 0:
        return F

    # Upward recursion: F_{m+1} = ((2m+1)F_m - exp(-T)) / (2T)
    e = math.exp(-T)
    for m in range(0, n):
        F = ((2*m + 1) * F - e) / (2*T)
    return F

# Validation tests
print("Boys Function Validation")
print("=" * 50)

# Test T = 0 limit
print("\n1. T = 0 limit (should match 1/(2n+1)):")
for n in range(6):
    computed = boys(n, 0.0)
    exact = 1.0 / (2*n + 1)
    error = abs(computed - exact)
    print(f"   F_{n}(0) = {computed:.10f}, exact = {exact:.10f}, err =
        {error:.2e}")

# Test specific values
print("\n2. Selected values:")
test_cases = [
    (0, 1.0, 0.746824133),
    (0, 10.0, 0.279694775),
    (2, 1.0, 0.130691379),
    (2, 10.0, 0.012614227),
    (5, 0.001, 0.0909008),  # Small T, high n
    (5, 50.0, 0.000353553),  # Large T
]
for n, T, expected in test_cases:
    computed = boys(n, T)
    error = abs(computed - expected)
    status = "PASS" if error < 1e-6 else "FAIL"
    print(f"   F_{n}({T:5.1f}) = {computed:.9f}, expected =
        {expected:.9f}, {status}")

# Test stability at small T
print("\n3. Stability test at small T (n=5):")
for k in [2, 4, 6, 8, 10]:
    T = 10**(-k)
    computed = boys(5, T)
    exact_at_0 = 1.0 / 11  # F_5(0) = 1/11
    print(f"   F_5(10^-{k}) = {computed:.10f}, |F_5(0)| =
        {exact_at_0:.10f}")
```

Sample output:

```
Boys Function Validation
==================================================

1. T = 0 limit (should match 1/(2n+1)):
   F_0(0) = 1.0000000000, exact = 1.0000000000, err = 0.00e+00
   F_1(0) = 0.3333333333, exact = 0.3333333333, err = 0.00e+00
   F_2(0) = 0.2000000000, exact = 0.2000000000, err = 0.00e+00
   F_3(0) = 0.1428571429, exact = 0.1428571429, err = 1.39e-17
   F_4(0) = 0.1111111111, exact = 0.1111111111, err = 1.39e-17
   F_5(0) = 0.0909090909, exact = 0.0909090909, err = 1.39e-17

2. Selected values:
   F_0(  1.0) = 0.746824133, expected = 0.746824133, PASS
   F_0( 10.0) = 0.279694775, expected = 0.279694775, PASS
   F_2(  1.0) = 0.130691379, expected = 0.130691379, PASS
   F_2( 10.0) = 0.012614227, expected = 0.012614227, PASS
   F_5(  0.0) = 0.090900826, expected = 0.090900800, PASS
   F_5( 50.0) = 0.000353553, expected = 0.000353553, PASS

3. Stability test at small T (n=5):
   F_5(10^-2) = 0.0909008203, |F_5(0)| = 0.0909090909
   F_5(10^-4) = 0.0909090101, |F_5(0)| = 0.0909090909
   F_5(10^-6) = 0.0909090901, |F_5(0)| = 0.0909090909
   F_5(10^-8) = 0.0909090909, |F_5(0)| = 0.0909090909
   F_5(10^-10) = 0.0909090909, |F_5(0)| = 0.0909090909
```

Validation criteria:

Test	Tolerance	Purpose
$T = 0$ limits	$< 10^{-10}$	Verify series implementation
Selected values	$< 10^{-6}$	Cross-validation
Small-T stability	Smooth convergence to limit	No oscillation or sign errors

Warning: Common Error

If your implementation shows oscillations, negative values, or large errors at small T and high n, you are likely using upward recursion inappropriately. Switch to the series expansion for $T < 25$.

3.2 Lab 4B: Closed-Form $(ss|ss)$ Primitive ERI vs. PySCF

Lab 4B Objectives

- Implement Algorithm 4.1 for normalized primitive $(ss|ss)$ ERIs
- Validate against PySCF's integral engine
- Observe ERI dependence on geometry and exponents

Expected numerical results:
For H_2 with $R = 1.4$ Bohr, $\alpha = 0.5$, $\beta = 0.4$:

ERI	PySCF	Analytic
$(00|00)$	0.72507561	0.72507561
$(00|11)$	0.44221449	0.44221449
$(01|01)$	0.38936124	0.38936124
$(11|11)$	0.65025167	0.65025167

All differences should be $< 10^{-10}$ Hartree.
Complete solution code:

```python
import numpy as np
import math
from pyscf import gto

```

```python
def boys(n: int, T: float, T_switch: float = 25.0, nseries: int = 50)
    -> float:
    """Boys function F_n(T)."""
    if T < T_switch:
        val = 0.0
        term = 1.0
        for k in range(nseries):
            val += term / (2*n + 2*k + 1)
            if abs(term) < 1e-16 * abs(val):
                break
            term *= -T / (k + 1)
        return val
    F = 0.5 * math.sqrt(math.pi / T) * math.erf(math.sqrt(T))
    if n == 0:
        return F
    e = math.exp(-T)
    for m in range(0, n):
        F = ((2*m + 1) * F - e) / (2*T)
    return F

def Ns(alpha):
    """Normalization constant for s-type Gaussian."""
    return (2*alpha/math.pi)**0.75

def eri_ssss_norm(alpha, A, beta, B, gamma, C, delta, D):
    """
    Normalized primitive (ss|ss) ERI using Algorithm 4.1.

    Parameters:
        alpha, beta: Exponents for bra pair (electron 1)
        gamma, delta: Exponents for ket pair (electron 2)
        A, B, C, D: Centers (3D coordinates)

    Returns:
        The normalized ERI value
    """
    A = np.array(A, dtype=float)
    B = np.array(B, dtype=float)
    C = np.array(C, dtype=float)
    D = np.array(D, dtype=float)

    # Step 1: Compute pair parameters
    p = alpha + beta
    q = gamma + delta
    mu = alpha*beta/p
    nu = gamma*delta/q

    P = (alpha*A + beta*B)/p
    Q = (gamma*C + delta*D)/q

    Rab2 = float(np.dot(A-B, A-B))
    Rcd2 = float(np.dot(C-D, C-D))
    Rpq2 = float(np.dot(P-Q, P-Q))

    # Step 2: Compute T parameter
    rho = p*q/(p+q)
    T = rho * Rpq2

    # Step 3: Evaluate Boys function
    F0 = boys(0, T)

    # Step 4: Compute prefactor and exponentials
    pref = 2 * (math.pi**2.5) / (p*q*math.sqrt(p+q))
    val_unnorm = pref * math.exp(-mu*Rab2 - nu*Rcd2) * F0

    # Step 5: Apply normalization
    return Ns(alpha)*Ns(beta)*Ns(gamma)*Ns(delta)*val_unnorm

# ===== Validation against PySCF =====

# Geometry in Bohr
R = 1.4
A = (0.0, 0.0, 0.0)
B = (0.0, 0.0, R)

alpha = 0.50  # exponent on atom 1
beta  = 0.40  # exponent on atom 2

# Define one-primitive s basis on each atom
basA = gto.basis.parse(f"""
H S
  {alpha:.10f}  1.0
""")
basB = gto.basis.parse(f"""
H S
  {beta:.10f}  1.0
""")

mol = gto.M(
    atom=f"H@1 {A[0]} {A[1]} {A[2]}; H@2 {B[0]} {B[1]} {B[2]}",
    basis={"H@1": basA, "H@2": basB},
    unit="Bohr",
    verbose=0
)

eri = mol.intor("int2e", aosym="s1")  # full tensor (nao,nao,nao,nao)

print("Lab 4B: (ss|ss) ERI Validation")
print("=" * 60)
print(f"Geometry: R = {R} Bohr")
print(f"Exponents: alpha = {alpha}, beta = {beta}")
print()

# Test all 16 elements
print(f"{'Index':<15} {'PySCF':>15} {'Analytic':>15}
   {'Difference':>15}")
print("-" * 60)

max_diff = 0.0
for i in range(2):
    for j in range(2):
        for k in range(2):
            for l in range(2):
                # Determine centers and exponents based on indices
                exp_i = alpha if i == 0 else beta
                exp_j = alpha if j == 0 else beta
                exp_k = alpha if k == 0 else beta
                exp_l = alpha if l == 0 else beta

                center_i = A if i == 0 else B
                center_j = A if j == 0 else B
                center_k = A if k == 0 else B
                center_l = A if l == 0 else B

                eri_pyscf = eri[i, j, k, l]
                eri_analytic = eri_ssss_norm(exp_i, center_i, exp_j,
                    center_j,
                                             exp_k, center_k, exp_l,
                                                 center_l)
                diff = abs(eri_pyscf - eri_analytic)
                max_diff = max(max_diff, diff)

                print(f"({i}{j}|{k}{l}){' ':>8} {eri_pyscf:>15.10f}
                    {eri_analytic:>15.10f} {diff:>15.2e}")

print("-" * 60)
print(f"Maximum difference: {max_diff:.2e}")
print(f"Status: {'PASS' if max_diff < 1e-10 else 'FAIL'}")

# Additional tests: vary R
print("\n\nDependence on bond length R:")
print(f"{'R (Bohr)':<12} {'T parameter':>12} {'(00|11)':>15}")
print("-" * 40)
for R_test in [0.5, 1.0, 1.4, 2.0, 3.0, 5.0]:
    B_test = (0.0, 0.0, R_test)
    eri_test = eri_ssss_norm(alpha, A, alpha, A, beta, B_test, beta,
        B_test)

    # Compute T for reference
    p = 2*alpha
    q = 2*beta
    P = A
    Q = B_test
    rho = p*q/(p+q)
    T = rho * R_test**2

    print(f"{R_test:<12.1f} {T:>12.4f} {eri_test:>15.10f}")
```

Sample output:

```
Lab 4B: (ss|ss) ERI Validation
============================================================
Geometry: R = 1.4 Bohr
Exponents: alpha = 0.5, beta = 0.4

Index                PySCF        Analytic      Difference
------------------------------------------------------------
(00|00)         0.7250756145    0.7250756145        3.33e-16
(00|01)         0.5640091298    0.5640091298        1.11e-16
(00|10)         0.5640091298    0.5640091298        1.11e-16
(00|11)         0.4422144937    0.4422144937        1.11e-16
(01|00)         0.5640091298    0.5640091298        0.00e+00
(01|01)         0.3893612413    0.3893612413        5.55e-17
```

```
(01|10)         0.3893612413   0.3893612413       5.55e-17
(01|11)         0.3100844979   0.3100844979       0.00e+00
(10|00)         0.5640091298   0.5640091298       0.00e+00
(10|01)         0.3893612413   0.3893612413       5.55e-17
(10|10)         0.3893612413   0.3893612413       5.55e-17
(10|11)         0.3100844979   0.3100844979       0.00e+00
(11|00)         0.4422144937   0.4422144937       1.11e-16
(11|01)         0.3100844979   0.3100844979       0.00e+00
(11|10)         0.3100844979   0.3100844979       0.00e+00
(11|11)         0.6502516674   0.6502516674       1.11e-16
------------------------------------------------------------
Maximum difference: 3.33e-16
Status: PASS


Dependence on bond length R:
R (Bohr)     T parameter        (00|11)
----------------------------------------
0.5              0.0893    0.5865066632
1.0              0.3571    0.5109108918
1.4              0.7000    0.4422144937
2.0              1.4286    0.3498408817
3.0              3.2143    0.2421949518
5.0              8.9286    0.1299310957
```

Key observations:

1. All ERIs match to machine precision ($\sim 10^{-16}$).
2. ERIs decrease monotonically with increasing R (Coulomb decay).
3. The T parameter increases quadratically with R.
4. At $R = 5$ Bohr, the ERI is about 1/4 of the value at $R = 0.5$ Bohr.

3.3 Lab 4C: Numerical Quadrature Check for $F_n(T)$

Lab 4C Objectives

- Compare Gauss–Legendre quadrature with series/recursion
- Observe how integrand shape affects quadrature accuracy
- Understand why Rys quadrature is superior for this application

Expected numerical results:

n	T	Series/Rec	Gauss–Legendre (64 pts)	Difference
0	0.1	0.966105146	0.966105146	$< 10^{-12}$
0	10.0	0.279694775	0.279694775	$< 10^{-10}$
2	0.1	0.193265629	0.193265629	$< 10^{-12}$
2	10.0	0.012614227	0.012614227	$< 10^{-9}$

Note: Gauss–Legendre accuracy degrades for large T (more points needed).
Complete solution code:

```python
import numpy as np
import math

def boys(n: int, T: float, T_switch: float = 25.0, nseries: int = 50)
    -> float:
    """Boys function F_n(T) via series + recursion."""
    if T < T_switch:
        val = 0.0
        term = 1.0
        for k in range(nseries):
            val += term / (2*n + 2*k + 1)
            if abs(term) < 1e-16 * abs(val):
                break
            term *= -T / (k + 1)
        return val
    F = 0.5 * math.sqrt(math.pi / T) * math.erf(math.sqrt(T))
    if n == 0:
        return F
    e = math.exp(-T)
    for m in range(0, n):
        F = ((2*m + 1) * F - e) / (2*T)
    return F

def boys_glq(n: int, T: float, npts: int = 64) -> float:
    """Approximate F_n(T) using Gauss-Legendre quadrature."""
    # Gauss-Legendre nodes/weights on [-1,1]
    x, w = np.polynomial.legendre.leggauss(npts)
    # Map to [0,1]
    t = 0.5*(x + 1.0)
    wt = 0.5*w
    # Evaluate integrand
    f = (t**(2*n)) * np.exp(-T*(t**2))
    return float(np.sum(wt*f))

print("Lab 4C: Gauss-Legendre vs Series/Recursion for Boys Function")
print("=" * 70)

# Basic comparison
print("\n1. Accuracy comparison (64 Gauss-Legendre points):")
print(f"{'n':<4} {'T':<8} {'Series/Rec':>18} {'GLQ(64)':>18}
    {'Difference':>15}")
print("-" * 70)

test_cases = [(0, 0.1), (0, 1.0), (0, 10.0), (0, 50.0),
              (2, 0.1), (2, 1.0), (2, 10.0), (2, 50.0),
              (5, 0.1), (5, 10.0)]

for n, T in test_cases:
    ref = boys(n, T)
    approx = boys_glq(n, T, npts=64)
    diff = abs(approx - ref)
    print(f"{n:<4} {T:<8.1f} {ref:>18.12e} {approx:>18.12e}
        {diff:>15.2e}")

# Study how many GL points are needed for different T
print("\n2. Gauss-Legendre points needed for F_0(T) at different T:")
print(f"{'T':<10} {'16 pts':>12} {'32 pts':>12} {'64 pts':>12} {'128
    pts':>12}")
print("-" * 60)

for T in [0.1, 1.0, 5.0, 10.0, 25.0, 50.0]:
    ref = boys(0, T)
    errors = []
    for npts in [16, 32, 64, 128]:
        approx = boys_glq(0, T, npts=npts)
        errors.append(abs(approx - ref))
    print(f"{T:<10.1f} {errors[0]:>12.2e} {errors[1]:>12.2e}
        {errors[2]:>12.2e} {errors[3]:>12.2e}")

# Integrand visualization data
print("\n3. Integrand shape analysis for n=2:")
print("   (Use these values to sketch/plot the integrand)")
print(f"{'t':<10} {'T=0.1':>15} {'T=1.0':>15} {'T=10.0':>15}")
print("-" * 55)

for t in np.linspace(0, 1, 11):
    vals = []
    for T in [0.1, 1.0, 10.0]:
        f = (t**4) * np.exp(-T * t**2)
        vals.append(f)
    print(f"{t:<10.1f} {vals[0]:>15.6f} {vals[1]:>15.6f}
        {vals[2]:>15.6f}")
```

Sample output:

```
Lab 4C: Gauss-Legendre vs Series/Recursion for Boys Function
======================================================================

1. Accuracy comparison (64 Gauss-Legendre points):
n    T         Series/Rec                 GLQ(64)      Difference
----------------------------------------------------------------------
0    0.1       9.661051465e-01   9.661051465e-01        2.22e-16
0    1.0       7.468241328e-01   7.468241328e-01        1.11e-16
0    10.0      2.796947754e-01   2.796947754e-01        1.64e-11
0    50.0      1.253314137e-01   1.253314010e-01        1.28e-08
2    0.1       1.932656291e-01   1.932656291e-01        0.00e+00
2    1.0       1.306913792e-01   1.306913792e-01        2.78e-17
2    10.0      1.261422685e-02   1.261422681e-02        3.61e-11
2    50.0      1.772453851e-03   1.772436015e-03        1.78e-08
5    0.1       9.090082652e-02   9.090082652e-02        1.39e-17
5    10.0      2.424610082e-03   2.424609935e-03        1.47e-10

2. Gauss-Legendre points needed for F_0(T) at different T:
T              16 pts       32 pts       64 pts      128 pts
------------------------------------------------------------
0.1          4.17e-14     6.66e-16     2.22e-16     1.11e-16
1.0          1.44e-11     3.11e-15     1.11e-16     2.22e-16
5.0          4.37e-06     3.69e-11     2.07e-15     2.22e-16
10.0         2.15e-03     3.01e-07     1.64e-11     1.99e-15
25.0         4.47e-01     1.52e-02     1.86e-05     7.03e-10
50.0         9.21e-01     2.68e-01     1.28e-08     5.25e-05
```

```
3. Integrand shape analysis for n=2:
   (Use these values to sketch/plot the integrand)
t              T=0.1          T=1.0          T=10.0
-------------------------------------------------------
0.0            0.000000       0.000000       0.000000
0.1            0.000100       0.000099       0.000090
0.2            0.001594       0.001537       0.001109
0.3            0.007995       0.007222       0.003273
0.4            0.025165       0.020618       0.004553
0.5            0.060653       0.043937       0.002759
0.6            0.125116       0.076413       0.000645
0.7            0.229039       0.102998       0.000047
0.8            0.381556       0.095909       0.000001
0.9            0.586907       0.053299       0.000000
1.0            0.833287       0.012340       0.000000
```

Key observations:

1. **Small T (e.g., 0.1):** 16 points give 10^{-14} accuracy. The integrand is smooth and nearly polynomial.
2. **Moderate T (e.g., 10):** 64 points give 10^{-11} accuracy. The integrand is peaked, requiring more points.
3. **Large T (e.g., 50):** Even 128 points only give 10^{-5} accuracy. The integrand is sharply concentrated near $t = 0$.
4. **Integrand shape:** At $T = 10$, the integrand drops to $< 10^{-6}$ by $t = 0.8$. Most of the integral comes from $t < 0.5$.
5. **Conclusion:** Gauss–Legendre requires many points for large T. Rys quadrature achieves exactness with $\lfloor n/2 \rfloor + 1$ points regardless of T.

4 Additional Notes for Instructors

4.1 Common Misconceptions

1. **"ERIs are symmetric under any index permutation":** Only 8 of the 24 possible permutations leave the ERI unchanged. The 8-fold symmetry is special, not general.
2. **"Schwarz screening gives the exact ERI value":** Schwarz only provides an upper bound. The actual ERI may be much smaller.
3. **"The Boys function is just an error function":** Only F_0 has a simple closed form. Higher orders require recursion or series.
4. **"Upward recursion is always stable":** It suffers from catastrophic cancellation at small T. The series expansion is essential for numerical stability.
5. **"Rys quadrature is approximate":** It is mathematically exact for the required moment range. The only errors are floating-point roundoff.

4.2 Suggested Discussion Questions

1. Why does the $(ss|ss)$ formula involve both "intra-pair" exponential factors and an "inter-pair" Boys function? What physical effects do they capture?
2. If you needed to compute ERIs for a $(pp|pp)$ shell quartet, what changes would be required beyond the $(ss|ss)$ formula?
3. How would Schwarz screening efficiency change for a 1D polymer vs. a compact 3D cluster of the same number of atoms?
4. The Boys function $F_n(T)$ decreases with both n and T. What are the physical reasons for each dependence?
5. Why is the number of Rys roots $\lfloor L/2 \rfloor + 1$ rather than $L + 1$? What property of Gaussian quadrature enables this efficiency?

4.3 Extensions for Advanced Students

1. Implement downward recursion for the Boys function starting from the asymptotic formula. Compare stability with upward recursion across all T regimes.
2. Derive the relationship between the Boys function and the incomplete gamma function: $F_n(T) = \frac{1}{2T^{n+1/2}}\gamma(n + \frac{1}{2}, T)$.
3. Implement a contracted $(ss|ss)$ ERI by summing over primitive pairs. Validate against PySCF for STO-3G hydrogen.
4. Study the accuracy of Schwarz bounds: compute $|(\mu\nu|\lambda\sigma)|/(Q_{\mu\nu}Q_{\lambda\sigma})$ for various shell quartets. How tight is the bound?
5. Implement the two-root Rys quadrature by solving the moment equations analytically. Verify that it exactly reproduces F_0, F_1, F_2, F_3.

5 References

1. Boys, S.F. (1950). "Electronic wave functions. I. A general method of calculation for the stationary states of any molecular system." *Proc. R. Soc. A* **200**, 542–554.
2. Dupuis, M., Rys, J., and King, H.F. (1976). "Evaluation of molecular integrals over Gaussian basis functions." *J. Chem. Phys.* **65**, 111–116.
3. Häser, M. and Ahlrichs, R. (1989). "Improvements on the direct SCF method." *J. Comput. Chem.* **10**, 104–111. (Schwarz screening)
4. Helgaker, T., Jørgensen, P., and Olsen, J. (2000). *Molecular Electronic-Structure Theory.* Wiley. Chapters 9–10.
5. Szabo, A. and Ostlund, N.S. (1989). *Modern Quantum Chemistry.* Dover Publications. Appendix A.
6. Sun, Q. (2015). "Libcint: An efficient general integral library for Gaussian basis functions." *J. Comput. Chem.* **36**, 1664–1671.

6 Exercise Answer Keys

Brief answers for the end-of-chapter exercises (Section 4.10).

6.1 Exercise 4.1: Derive the $(ss|ss)$ Formula [Core]

Derivation Outline

(a) Apply GPT to bra and ket pairs:
Starting from the primitive ERI:

$$(ab|cd) = \iint \chi_a(\mathbf{r}_1)\chi_b(\mathbf{r}_1)\frac{1}{r_{12}}\chi_c(\mathbf{r}_2)\chi_d(\mathbf{r}_2)\, d\mathbf{r}_1\, d\mathbf{r}_2$$

For the bra pair centered at **A** and **B** with exponents α and β:

$$\chi_a\chi_b \propto e^{-\alpha|\mathbf{r}_1-\mathbf{A}|^2}e^{-\beta|\mathbf{r}_1-\mathbf{B}|^2} = e^{-p|\mathbf{r}_1-\mathbf{P}|^2}e^{-\mu R_{AB}^2}$$

where $p = \alpha+\beta$, $\mu = \frac{\alpha\beta}{\alpha+\beta}$, and $\mathbf{P} = \frac{\alpha\mathbf{A}+\beta\mathbf{B}}{p}$.
Similarly for the ket pair:

$$\chi_c\chi_d \propto e^{-q|\mathbf{r}_2-\mathbf{Q}|^2}e^{-\nu R_{CD}^2}$$

where $q = \gamma+\delta$, $\nu = \frac{\gamma\delta}{\gamma+\delta}$, and $\mathbf{Q} = \frac{\gamma\mathbf{C}+\delta\mathbf{D}}{q}$.
(b) Insert Gaussian representation of $1/r_{12}$:
Using the integral identity:

$$\frac{1}{r_{12}} = \frac{2}{\sqrt{\pi}}\int_0^\infty e^{-u^2r_{12}^2}\, du$$

The 6D spatial integral becomes:

$$\int e^{-p|\mathbf{r}_1-\mathbf{P}|^2}\int e^{-q|\mathbf{r}_2-\mathbf{Q}|^2}\cdot e^{-u^2|\mathbf{r}_1-\mathbf{r}_2|^2}\, d\mathbf{r}_1\, d\mathbf{r}_2$$

Using the Gaussian convolution formula, this evaluates to:

$$\left(\frac{\pi^3}{(p+u^2)(q+u^2)(u^2+\frac{pq}{p+q})}\right)^{1/2} \quad \text{(simplified form)}$$

After simplification, the spatial integrals yield:

$$\frac{\pi^{5/2}}{pq\sqrt{p+q}}e^{-\rho|\mathbf{P}-\mathbf{Q}|^2t^2} \quad \text{where } \rho = \frac{pq}{p+q}$$

(c) Transform to Boys function:
The remaining u-integral has the form:

$$\int_0^\infty f(u^2)\, du \to \int_0^1 t^0e^{-Tt^2}\, dt = F_0(T)$$

with substitution $t = u\sqrt{p+q}/\sqrt{pq}$ and $T = \rho R_{PQ}^2$.
Final result:

$$(ab|cd) = \frac{2\pi^{5/2}}{pq\sqrt{p+q}}e^{-\mu R_{AB}^2}\, e^{-\nu R_{CD}^2}\, F_0(T)$$

(before normalization constants).

6.2 Exercise 4.2: Limiting Behavior of ERIs [Core]

Analysis of Limits

(a) $R_{PQ} \to 0$ (coinciding pair centers):
When $\mathbf{P} = \mathbf{Q}$:

- $T = \rho R_{PQ}^2 = 0$
- $F_0(0) = \int_0^1 e^0 \, dt = 1$

The ERI becomes:

$$(ab|cd)\big|_{R_{PQ}=0} = \frac{2\pi^{5/2}}{pq\sqrt{p+q}} e^{-\mu R_{AB}^2} e^{-\nu R_{CD}^2}$$

The constant in terms of p and q is:

$$\boxed{\frac{2\pi^{5/2}}{pq\sqrt{p+q}}}$$

(b) $R_{PQ} \to \infty$ (well-separated distributions):
For large T: $F_0(T) \approx \frac{1}{2}\sqrt{\frac{\pi}{T}} = \frac{1}{2}\sqrt{\frac{\pi}{\rho R_{PQ}^2}}$
Substituting:

$$\begin{aligned}
(ab|cd) &\approx \frac{2\pi^{5/2}}{pq\sqrt{p+q}} \cdot e^{-\mu R_{AB}^2} \cdot e^{-\nu R_{CD}^2} \cdot \frac{1}{2}\sqrt{\frac{\pi}{\rho}} \cdot \frac{1}{R_{PQ}} \\
&= \frac{\pi^3}{pq\sqrt{p+q}} \cdot \sqrt{\frac{p+q}{pq}} \cdot \frac{e^{-\mu R_{AB}^2} e^{-\nu R_{CD}^2}}{R_{PQ}} \\
&= \frac{\pi^3}{(pq)^{3/2}} \cdot \frac{e^{-\mu R_{AB}^2} e^{-\nu R_{CD}^2}}{R_{PQ}}
\end{aligned}$$

(c) Physical interpretation of $1/R_{PQ}$ decay:
The $1/R_{PQ}$ decay is the **classical Coulomb law**:

- At large separations, each overlap distribution $\chi_\mu \chi_\nu$ behaves like a localized charge centered at $\mathbf{P}$ or $\mathbf{Q}$.
- The interaction energy between two point charges at separation R is $\propto 1/R$.
- The Gaussian prefactors $e^{-\mu R_{AB}^2}$ and $e^{-\nu R_{CD}^2}$ represent the "compactness" of each charge cloud.
- The ERI asymptotically matches the classical electrostatic picture, validating that quantum mechanics reproduces classical limits.

6.3 Exercise 4.3: Schwarz Screening Experiment [Core]

Expected numerical results for H_2O:

Basis	NAO	N^4 (total)	Screened ($\tau = 10^{-10}$)
STO-3G	7	2,401	0–5%
6-31G	13	28,561	10–15%
6-31+G*	24	331,776	25–35%

Key observations:

1. The screened fraction *increases* with diffuse functions because diffuse–diffuse pairs have small diagonal ERIs $(\mu\nu|\mu\nu)$ due to spatial spread.
2. For compact bases (STO-3G), almost all ERIs are significant because basis functions overlap substantially.
3. The Schwarz bound is guaranteed to be an upper bound: for 100 random quartets, no violation should occur (bound $\geq$ actual).

Implementation outline:

```python
# Compute pair norms
Q = np.zeros((nao, nao))
for mu in range(nao):
    for nu in range(nao):
        Q[mu, nu] = np.sqrt(eri[mu, nu, mu, nu])

# Count screened ERIs
n_screened = 0
for mu, nu, lam, sig in product(range(nao), repeat=4):
    if Q[mu, nu] * Q[lam, sig] < tau:
        n_screened += 1
```

6.4 Exercise 4.4: Boys Function Stability [Advanced]

Method comparison for $n \in \{0, \ldots, 10\}$ and various T:

T	Pure Upward	Hybrid Series/Rec	Downward
10^{-10}	10^{-3} (fails)	10^{-15}	10^{-15}
10^{-6}	10^{-5} (fails)	10^{-15}	10^{-15}
10^{-2}	10^{-8} (poor)	10^{-15}	10^{-15}
1	10^{-14}	10^{-15}	10^{-15}
10	10^{-14}	10^{-14}	10^{-15}
100	10^{-14}	10^{-14}	10^{-15}

(Values show maximum absolute error vs. 128-point Gauss–Legendre reference)

Analysis:

- **Pure upward recursion** suffers catastrophic cancellation at small T because $(2n+1)F_n(T) \approx e^{-T}$, leading to subtraction of nearly equal numbers.
- **Hybrid series/recursion** is robust: use series for $T < 25$ (converges rapidly), use erf + upward for $T \geq 25$ (no cancellation).
- **Downward recursion** is always stable because it involves *addition*, not subtraction. Starting from a high $n_{\text{max}} + 5$ with the asymptotic formula and recursing down "washes out" initial errors.

Recommendation: Use hybrid series/recursion for simplicity, or downward recursion for maximum robustness.

6.5 Exercise 4.5: Root Count Scaling with Angular Momentum [Conceptual]

Explanation (8–12 sentences):

Higher angular momentum shells introduce polynomial factors $(x - A_x)^{\ell_A}(y - A_y)^{m_A}(z - A_z)^{n_A}$ into the ERI integrand, where $\ell_A + m_A + n_A = L_A$. When these polynomials are multiplied out across all four basis functions, the total polynomial degree in the Cartesian coordinates is $L = \ell_A + \ell_B + \ell_C + \ell_D$.

The derivative identity $\frac{d}{dT} F_n(T) = -F_{n+1}(T)$ shows that differentiating the $(ss|ss)$ formula with respect to center coordinates (to generate p-type integrals) brings in higher-order Boys functions. Specifically, each derivative effectively raises the Boys function order by one.

For a shell quartet with total angular momentum L, the ERI involves Boys functions $F_0(T), F_1(T), \ldots, F_{n_{\max}}(T)$ where $n_{\max} = \lfloor L/2 \rfloor$. The factor of 2 arises because the Rys substitution $x = t^2$ converts the Boys integral into a polynomial in x, doubling the effective polynomial degree that can be handled.

Gaussian quadrature with n_r nodes exactly integrates polynomials up to degree $2n_r - 1$. Since we need to reproduce moments $x^0, x^1, \ldots, x^{n_{\max}}$, we require $n_{\max} \leq 2n_r - 1$, giving $n_r = \lceil (n_{\max} + 1)/2 \rceil = \lfloor L/2 \rfloor + 1$.

The root count thus grows as $\lfloor L/2 \rfloor + 1$ rather than $L + 1$ because Gaussian quadrature is "doubly efficient"—each root captures two polynomial degrees of freedom. This efficiency is fundamental to why Rys quadrature makes high-angular-momentum ERIs tractable.

6.6 Exercise 4.6: ERI Symmetry Verification [Core]

Expected results for H_2O/STO-3G:

- Number of AOs: $N = 7$
- Total ERIs: $7^4 = 2401$
- Maximum symmetry deviation: $\sim 10^{-15}$ (machine precision)

Symmetry relations to verify:

1. Bra swap: $(ij|kl) = (ji|kl)$
2. Ket swap: $(ij|kl) = (ij|lk)$
3. Bra-ket exchange: $(ij|kl) = (kl|ij)$
4. All combinations (8 total permutations)

Implementation:

```python
eri = mol.intor("int2e", aosym="s1")
max_dev = 0.0
for i in range(nao):
    for j in range(nao):
        for k in range(nao):
            for l in range(nao):
                ref = eri[i,j,k,l]
                # Check all 8 symmetry partners
                syms = [eri[j,i,k,l], eri[i,j,l,k], eri[j,i,l,k],
                        eri[k,l,i,j], eri[l,k,i,j], eri[k,l,j,i],
                          eri[l,k,j,i]]
                for s in syms:
                    max_dev = max(max_dev, abs(ref - s))
print(f"Max deviation: {max_dev:.2e}")  # Should be ~1e-15
```

6.7 Exercise 4.7: ERI Scaling with Basis Size [Core]

Expected results for water clusters in 6-31G:

System	N	Unique ERIs	Memory (s1)	Memory (s8)
H_2O	13	9,316	0.23 MB	0.07 MB
$(H_2O)_2$	26	136,526	3.7 MB	1.1 MB
$(H_2O)_3$	39	630,630	18.5 MB	5.0 MB
$(H_2O)_4$	52	1,936,771	58.7 MB	15.5 MB
$(H_2O)_5$	65	4,691,115	143 MB	37.5 MB

Scaling analysis:

- Full tensor: $N^4 \times 8$ bytes (s1 format)
- Symmetric storage: $\frac{N(N+1)}{2} \cdot \frac{N(N+1)/2+1}{2} \times 8$ bytes (s8 format)
- Memory becomes prohibitive (> 1 GB) around $N \approx 100$ for s1, $N \approx 180$ for s8.

The $\mathcal{O}(N^4)$ scaling is confirmed by plotting log(ERIs) vs log(N): slope ≈ 4.

6.8 Exercise 4.8: Moment Matching for Rys Quadrature [Advanced]

(a) Moments at $T = 1.0$:

k	$m_k(1.0) = 2F_k(1.0)$
0	1.4936
1	0.5213
2	0.2614
3	0.1555
4	0.1027
5	0.0727

(b) One-root quadrature ($n_r = 1$):

$$W_1 = m_0 = 1.4936, \quad x_1 = \frac{m_1}{m_0} = \frac{0.5213}{1.4936} = 0.349$$

Verification:

- $W_1 \cdot x_1^0 = 1.4936 = m_0$ ✓
- $W_1 \cdot x_1^1 = 0.5213 = m_1$ ✓
- $W_1 \cdot x_1^2 = 0.182 \neq m_2 = 0.261$ (not exact for $k \geq 2$)

(c) Two-root quadrature ($n_r = 2$):
Using Hankel matrix construction:

$$\mathbf{H} = \begin{pmatrix} m_0 & m_1 \\ m_1 & m_2 \end{pmatrix}, \quad \mathbf{H}^{(1)} = \begin{pmatrix} m_1 & m_2 \\ m_2 & m_3 \end{pmatrix}$$

After Cholesky factorization and eigenvalue solution:

$$x_1 \approx 0.116, \quad W_1 \approx 0.651$$
$$x_2 \approx 0.619, \quad W_2 \approx 0.843$$

Verification: $\sum_i W_i x_i^k = m_k$ exactly for $k = 0, 1, 2, 3$.
(d) Quadrature error pattern:
For n_r roots, moments are reproduced exactly up to order $2n_r - 1$, then errors appear:

k	$n_r = 1$ error	$n_r = 2$ error	$n_r = 3$ error
0	0	0	0
1	0	0	0
2	7.9×10^{-2}	0	0
3	9.4×10^{-2}	0	0
4	–	8.2×10^{-3}	0
5	–	1.4×10^{-2}	0

The pattern confirms: n_r points give exact moments for $k = 0, 1, \ldots, 2n_r - 1$.

6.9 Exercise 4.9: Boys Function Evaluation Strategies [Research/Challenge]

Comparative analysis:

Method	**Accuracy**	**Speed**	**Memory**
Tabulation + interpolation	10^{-10}	Fast	∼50 KB
Chebyshev approximation	10^{-12}	Fastest	∼5 KB
Rational (Padé) approximation	10^{-14}	Fast	∼2 KB

Strategy details:
(a) Tabulation + interpolation:

- Grid spacing $\Delta T = 0.1$ for $T \in [0, 30]$, plus asymptotic for $T > 30$
- Cubic spline gives $\sim 10^{-10}$ accuracy with 300 table entries per n
- Total memory: $11 \times 300 \times 8$ bytes ≈ 26 KB for $n \leq 10$

(b) Chebyshev approximation:

- Divide T range: $[0, 2]$, $[2, 10]$, $[10, 30]$, plus asymptotic
- 12–15 Chebyshev terms per range achieve 10^{-12} accuracy
- Very fast evaluation via Clenshaw recurrence

(c) Rational (Padé) approximation:

- libcint uses rational function fits in different T regimes
- $(5, 5)$ or $(6, 6)$ Padé approximants suffice for 10^{-14} accuracy
- Minimal memory footprint; efficient for vectorized evaluation

Discussion:
Production codes (libcint, GAMESS, Gaussian) typically use:

- Series expansion for $T < T_{\text{switch}}$ ($T_{\text{switch}} \approx$ 15–30)
- Asymptotic + downward recursion for large T
- Precomputed tables or polynomial fits for intermediate T to avoid function calls

The trade-off is between accuracy, speed, and code complexity. For educational purposes, the series + recursion hybrid is simplest; for production, Chebyshev or rational fits offer the best speed/accuracy balance.