

Chapter 5: Answer Key

Rys Quadrature in Practice
2302638 Advanced Quantum Chemistry

Department of Chemistry, Chulalongkorn University

Contents

1	Overview	2
2	Checkpoint Question Answers	2
2.1	Checkpoint 5.1: Orthonormalization Revisited (Section 5.1)	2
2.2	Checkpoint 5.2: One Root Suffices for Two Boys Values (Section 5.2)	3
2.3	Checkpoint 5.2b: Interpreting the Rys Weight Figure (Section 5.2)	4
2.4	Checkpoint 5.3: Weight Positivity (Section 5.3)	5
2.5	Checkpoint 5.4: Eigenvalue Interpretation (Section 5.3)	6
2.6	Checkpoint 5.5: Algorithm 5.1 Understanding (Section 5.4)	7
2.7	Checkpoint 5.6: Root Count Understanding (Section 5.5)	8
2.8	Checkpoint 5.6b: Resolving the Circularity (Section 5.6)	10
2.9	Checkpoint 5.7: Method Equivalence (Section 5.7 – ERIs via Boys and Rys)	11
2.10	Checkpoint 5.8: Identifying Derivative Sources (Section 5.8)	12
2.11	Checkpoint 5.9: Structure of the $(ps ss)$ Formula (Section 5.8)	13
2.12	Checkpoint 5.10: Crossed Indices in Exchange (Section 5.9)	14
2.13	Checkpoint 5.11: Symmetry of J and K (Section 5.9)	15
2.14	Checkpoint 5.12: Why \mathbf{V}_{HF} Rather Than Separate \mathbf{J} and \mathbf{K} (Section 5.10)	16
3	Lab Solutions	17
3.1	Lab 5A: Construct Rys Nodes/Weights and Verify Moment Matching	17
3.2	Lab 5B: Compute $(ss ss)$ ERI Using Rys Quadrature	18
3.3	Lab 5C: Compute $(p_{\xi}s ss)$ Using Derivative Identity	19
3.4	Lab 5D: Build \mathbf{J} and \mathbf{K} from ERIs	20
4	Additional Notes for Instructors	22
4.1	Common Misconceptions	22
4.2	Suggested Discussion Questions	22
4.3	Extensions for Advanced Students	22
5	References	23
6	Exercise Answer Keys	24
6.1	Exercise 5.1: One-Root Quadrature Formulas [Core]	24
6.2	Exercise 5.2: Two-Root Quadrature and Moment Matching [Core]	24
6.3	Exercise 5.3: $(ss ss)$ by Three Routes [Core]	25
6.4	Exercise 5.4: $(p_{\xi}s ss)$ and the Appearance of F_1 [Core]	26
6.5	Exercise 5.5: J/K Build and Energy Component Check [Core]	26
6.6	Exercise 5.6: Schwarz Screening Toy Study [Advanced]	27
6.7	Exercise 5.7: Hankel Matrix Conditioning [Advanced]	28
6.8	Exercise 5.8: Obara–Saika Recursion [Research/Challenge]	28

1 Overview

This answer key covers:

- **14 Checkpoint Questions** embedded throughout Chapter 5, testing conceptual understanding
- **4 Python Labs** (5A, 5B, 5C, 5D) with expected numerical results and validation criteria

Key concepts tested:

1. Boys functions as moments of a weight function
2. Gaussian quadrature construction via Hankel matrices
3. The Golub–Welsch algorithm for nodes/weights
4. Root count rule for ERIs
5. Derivative identity for angular momentum
6. Coulomb and exchange matrix construction

Validation standard: All numerical results should match PySCF reference calculations to within the specified tolerances (typically $< 10^{-10}$).

2 Checkpoint Question Answers

This section provides detailed answers to all 14 checkpoint questions from Chapter 5, organized by their location in the chapter.

2.1 Checkpoint 5.1: Orthonormalization Revisited (Section 5.1)

Section 5.1 – Gram–Schmidt Parallel

Question: In Chapter 2 we orthonormalized a function space with Gram–Schmidt/eigendecomposition, using the overlap matrix \mathbf{S} as the Gram matrix. In this chapter we orthonormalize a *polynomial* space (with a weight $w_T(x)$) to obtain Gaussian quadrature nodes and weights. The linear algebra is the same; only the objects change. Which matrix in Algorithm 5.1 plays the role of the overlap matrix \mathbf{S} ? (*Hint: It encodes the inner products $\langle x^i, x^j \rangle_w$.*)

Answer:

The Hankel matrix \mathbf{H} plays the role of the overlap matrix \mathbf{S} .

Detailed explanation:

In Chapter 2, the overlap matrix \mathbf{S} is defined by:

$$S_{\mu\nu} = \langle \chi_\mu | \chi_\nu \rangle = \int \chi_\mu(\mathbf{r}) \chi_\nu(\mathbf{r}) d\mathbf{r}$$

It is the Gram matrix of AO basis functions under the standard L^2 inner product.

In Chapter 5, we work with monomials $\{1, x, x^2, \dots, x^{n_r-1}\}$ and a weighted inner product:

$$\langle f, g \rangle_T = \int_0^1 f(x) g(x) w_T(x) dx$$

where $w_T(x) = x^{-1/2} e^{-Tx}$.

The Gram matrix of monomials under this inner product is:

$$H_{ij} = \langle x^i, x^j \rangle_T = \int_0^1 x^{i+j} w_T(x) dx = m_{i+j}(T)$$

This is precisely the Hankel matrix \mathbf{H} from Eq. (5.12).

The parallel structure:

	Chapter 2 (AOs)	Chapter 5 (Polynomials)
Basis elements	χ_μ (GTOs)	x^k (monomials)
Inner product	$\langle \chi_\mu \chi_\nu \rangle$	$\langle x^i, x^j \rangle_T$
Gram matrix	\mathbf{S}	\mathbf{H} (Hankel)
Orthonormalization	$\mathbf{S} = \mathbf{U}\mathbf{S}\mathbf{U}^\top$	$\mathbf{H} = \mathbf{L}\mathbf{L}^\top$ (Cholesky)
Result	Orthonormal AOs	Orthonormal polynomials

Why Cholesky works like Gram–Schmidt:

The Cholesky factorization $\mathbf{H} = \mathbf{L}\mathbf{L}^\top$ and the transformation $\mathbf{C} = \mathbf{L}^{-1}$ gives:

$$\mathbf{C}\mathbf{H}\mathbf{C}^\top = \mathbf{L}^{-1}\mathbf{L}\mathbf{L}^\top\mathbf{L}^{-\top} = \mathbf{I}$$

This means the rows of \mathbf{C} define coefficients for orthonormal polynomials in the monomial basis—exactly analogous to how the canonical orthogonalizer $\mathbf{X} = \mathbf{U}\mathbf{S}^{-1/2}$ transforms AOs into orthonormal combinations.

2.2 Checkpoint 5.2: One Root Suffices for Two Boys Values (Section 5.2)

Section 5.2 – Moment Matching

Question: For $n_r = 1$, Eq. (5.8) is exact for polynomial degrees 0 and 1.

- Write out the moment-matching equations for $n = 0$ and $n = 1$ explicitly.
- Show that one root suffices to reproduce both $F_0(T)$ and $F_1(T)$.
- What are x_1 and W_1 in terms of m_0 and m_1 ?

Answer:

(a) Moment-matching equations:

For $n_r = 1$, we have a single node x_1 and weight W_1 . The moment-matching conditions (Eq. 5.8) state:

$$m_n = \sum_{i=1}^{n_r} W_i x_i^n = W_1 x_1^n$$

must hold for $n = 0, 1, \dots, 2n_r - 1 = 1$.

Explicitly:

$$n = 0 : \quad m_0 = W_1 x_1^0 = W_1 \tag{1}$$

$$n = 1 : \quad m_1 = W_1 x_1^1 = W_1 x_1 \tag{2}$$

(b) One root reproduces F_0 and F_1 :

From the moment definition $m_n = 2F_n(T)$:

$$F_0(T) = \frac{m_0}{2} = \frac{W_1}{2} \tag{3}$$

$$F_1(T) = \frac{m_1}{2} = \frac{W_1 x_1}{2} \tag{4}$$

These are exact because the quadrature with $n_r = 1$ point is exact for polynomials of degree $\leq 2(1) - 1 = 1$.

(c) Solving for x_1 and W_1 :

From the moment-matching equations:

$$W_1 = m_0 \quad (5)$$

$$W_1 x_1 = m_1 \quad (6)$$

Dividing the second by the first:

$$x_1 = \frac{m_1}{m_0} = \frac{2F_1(T)}{2F_0(T)} = \frac{F_1(T)}{F_0(T)}$$

$$W_1 = m_0 = 2F_0(T)$$

Numerical verification for $T = 1.0$:

- $F_0(1.0) = 0.5\sqrt{\pi/1.0} \cdot \text{erf}(1.0) \approx 0.7468$
- $F_1(1.0) = \frac{(2 \cdot 0 + 1)F_0(1.0) - e^{-1.0}}{2 \cdot 1.0} \approx 0.1894$
- $x_1 = 0.1894/0.7468 \approx 0.2536$
- $W_1 = 2 \times 0.7468 \approx 1.4936$

These match the expected values in Exercise 5.1.

Key Formula

For one-root Rys quadrature:

$$x_1 = \frac{F_1(T)}{F_0(T)}, \quad W_1 = 2F_0(T)$$

2.3 Checkpoint 5.2b: Interpreting the Rys Weight Figure (Section 5.2)

Section 5.2 – Weight Function Behavior

Question: Examine Figure 5.1.

- As T increases, the weight function becomes more sharply peaked near $x = 0$. What happens to the quadrature nodes? Why does this make physical sense? (*Hint: Large T corresponds to widely separated centers in the ERI.*)
- The total area under $w_T(x)$ equals $m_0(T) = 2F_0(T)$. For larger T , is this area larger or smaller? Relate this to the asymptotic behavior $F_0(T) \rightarrow \frac{1}{2}\sqrt{\pi/T}$ for $T \rightarrow \infty$.

Answer:

(a) Node clustering behavior:

As T increases, the quadrature nodes $\{x_i\}$ cluster toward $x = 0$.

Why this happens: The weight function $w_T(x) = x^{-1/2}e^{-Tx}$ has two factors:

- $x^{-1/2}$: Singular at $x = 0$, always pulls nodes toward small x
- e^{-Tx} : Decay factor that becomes steeper as T increases

For large T , the exponential decay dominates: $w_T(x)$ drops off rapidly for $x > 1/T$. The weight is effectively concentrated in a small region near $x = 0$, so optimal quadrature nodes (which sample where the weight is largest) must cluster there.

Physical interpretation: Large $T = \rho|\mathbf{P} - \mathbf{Q}|^2$ corresponds to widely separated electron distributions (bra pair at \mathbf{P} , ket pair at \mathbf{Q}). At large separation:

- The Coulomb interaction $1/r_{12}$ becomes weaker and nearly classical
- The quantum exchange effects (represented by the polynomial factors) become negligible
- The integral is dominated by the asymptotic regime of the Boys function

The node clustering toward $x = 0$ reflects this physics: at large separation, only the “soft” part of the Coulomb interaction matters.

(b) Total area (zeroth moment) behavior:

For larger T , the total area $m_0(T) = 2F_0(T)$ is **smaller**.

Quantitative behavior:

- $T = 0$: $m_0(0) = 2F_0(0) = 2 \cdot 1 = 2$
- $T = 1$: $m_0(1) = 2F_0(1) \approx 2 \times 0.747 \approx 1.49$
- $T = 10$: $m_0(10) = 2F_0(10) \approx 2 \times 0.158 \approx 0.32$
- $T \rightarrow \infty$: $m_0(T) \rightarrow 2 \cdot \frac{1}{2} \sqrt{\frac{\pi}{T}} = \sqrt{\frac{\pi}{T}} \rightarrow 0$

Physical meaning: The decreasing area reflects the decreasing Coulomb interaction strength at large separation. The asymptotic form $F_0(T) \sim \sqrt{\pi/(4T)}$ gives the classical Coulomb law: the integral of e^{-Tr^2}/r over all space scales as $1/\sqrt{T} \propto 1/R_{PQ}$, consistent with the $1/r$ distance dependence.

2.4 Checkpoint 5.3: Weight Positivity (Section 5.3)

Section 5.3 – Golub–Welsch

Question: Why must all W_i be positive for a positive weight $w_T(x)$? Use Eq. (5.15) and the fact that $m_0 > 0$.

Answer:

The Golub–Welsch weight formula:

From Eq. (5.15), the quadrature weights are:

$$W_i = m_0(V_{0i})^2$$

where V_{0i} is the first component of the i -th normalized eigenvector of the Jacobi matrix \mathbf{J} .

Why $W_i > 0$:

1. $m_0 > 0$: The zeroth moment is

$$m_0 = \int_0^1 w_T(x) dx = \int_0^1 x^{-1/2} e^{-Tx} dx > 0$$

because the integrand is strictly positive on $(0, 1]$.

2. $(V_{0i})^2 \geq 0$: Any real number squared is non-negative.

3. $(V_{0i})^2 > 0$: For Gaussian quadrature derived from orthogonal polynomials with a positive weight function, the eigenvectors of the Jacobi matrix are such that $V_{0i} \neq 0$ for all i .

Physical interpretation:

The positivity of weights is a fundamental property of Gaussian quadrature for positive weights. If any weight were negative, the quadrature could not be interpreted as an expectation value:

$$\int_0^1 f(x)w_T(x) dx \approx \sum_i W_i f(x_i)$$

A negative W_i would allow the approximation to be negative even when $f(x) \geq 0$ everywhere, which would be inconsistent with the integral being manifestly non-negative.

Connection to probability:

The normalized weights $\tilde{W}_i = W_i/m_0$ satisfy $\sum_i \tilde{W}_i = 1$ and $\tilde{W}_i > 0$ —they form a discrete probability distribution. The quadrature nodes are the “sample points” for this distribution.

2.5 Checkpoint 5.4: Eigenvalue Interpretation (Section 5.3)

Section 5.3 – Jacobi Matrix Properties

Question: The nodes x_i are eigenvalues of the Jacobi matrix \mathbf{J} .

- What property of \mathbf{J} guarantees that all x_i are real?
- Why must $x_i \in (0, 1)$ for the Rys weight $w_T(x) = x^{-1/2}e^{-Tx}$? (*Hint: The roots of orthogonal polynomials lie in the interior of the weight's support.*)

Answer:

(a) Reality of eigenvalues:

The Jacobi matrix \mathbf{J} is **real symmetric**:

$$J_{kk} = a_k, \quad J_{k,k+1} = J_{k+1,k} = b_{k+1}$$

where the recurrence coefficients a_k and b_k are real.

By the spectral theorem for symmetric matrices: *all eigenvalues of a real symmetric matrix are real.*

Additionally, the eigenvectors are orthogonal and can be chosen to be real. This guarantees that the quadrature nodes $\{x_i\}$ are real numbers.

(b) Why $x_i \in (0, 1)$:

This follows from a fundamental theorem in orthogonal polynomial theory:

Theorem: *The zeros of the n -th orthogonal polynomial $p_n(x)$ with respect to a positive weight function $w(x)$ on interval (a, b) are all real, simple, and lie strictly inside (a, b) .*

For Rys quadrature:

- The weight is $w_T(x) = x^{-1/2}e^{-Tx}$
- The support is $(0, 1)$
- The quadrature nodes are the zeros of $p_{n_r}(x)$

Therefore, all nodes satisfy $0 < x_i < 1$.

Physical intuition:

The quadrature nodes are the “optimal sampling points” for the weight function. Since $w_T(x)$ is nonzero only on $(0, 1)$, placing nodes outside this interval would waste sampling effort on regions where the weight is zero. The optimal placement is in the interior where the weight is most significant.

Note on the singularity at $x = 0$:

The weight $w_T(x) = x^{-1/2}e^{-Tx}$ has an integrable singularity at $x = 0$ (since $\int_0^1 x^{-1/2} dx = 2$ is finite). This singularity causes the moments and hence the quadrature to be “pulled” toward small x values, especially when T is large. However, all nodes remain strictly positive: $x_i > 0$.

2.6 Checkpoint 5.5: Algorithm 5.1 Understanding (Section 5.4)

Section 5.4 – Algorithm 5.1

Question:

- Why do we need moments up to m_{2n_r-1} in Algorithm 5.1? (*Hint: What is the largest index required in \mathbf{H} and $\mathbf{H}^{(1)}$?*)
- What happens if the Cholesky factorization fails in step 3? (*Hint: Think about the condition number of \mathbf{H} .*)
- Why is the Jacobi matrix \mathbf{J} guaranteed to be symmetric?

Answer:

(a) Why moments up to m_{2n_r-1} :

The Hankel matrix \mathbf{H} has entries:

$$H_{ij} = m_{i+j}, \quad i, j = 0, 1, \dots, n_r - 1$$

The largest index occurs at $i = j = n_r - 1$:

$$H_{n_r-1, n_r-1} = m_{2(n_r-1)} = m_{2n_r-2}$$

The shifted Hankel matrix $\mathbf{H}^{(1)}$ has entries:

$$H_{ij}^{(1)} = m_{i+j+1}, \quad i, j = 0, 1, \dots, n_r - 1$$

The largest index is:

$$H_{n_r-1, n_r-1}^{(1)} = m_{2n_r-2+1} = m_{2n_r-1}$$

Therefore, we need **all moments from m_0 to m_{2n_r-1}** .

Example for $n_r = 2$:

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

Requires m_0, m_1, m_2, m_3 (i.e., $2 \times 2 - 1 = 3$ is the highest index).

(b) What if Cholesky fails:

Cholesky factorization requires the matrix to be **positive definite**. For well-behaved moments, \mathbf{H} should be positive definite because it is the Gram matrix of linearly independent monomials.

Failure can occur when:

1. The moments are computed inaccurately (especially for extreme T)

2. n_r is too large, causing \mathbf{H} to become ill-conditioned
3. Numerical rounding causes near-zero or negative pivots

The condition number $\kappa(\mathbf{H})$ grows rapidly with n_r :

n_r	Typical $\kappa(\mathbf{H})$ at $T = 1$
2	$\sim 10^2$
3	$\sim 10^4$
4	$\sim 10^6$
5	$\sim 10^8$

Practical remedy: Production codes use specialized root-finding algorithms (polynomial approximations, extended precision) rather than the moment-based approach for large n_r .

(c) Why \mathbf{J} is symmetric:

The Jacobi matrix is constructed as:

$$\mathbf{J} = \mathbf{C}\mathbf{H}^{(1)}\mathbf{C}^\top$$

where $\mathbf{C} = \mathbf{L}^{-1}$ is lower triangular.

Since $\mathbf{H}^{(1)}$ is a Hankel matrix, it is symmetric ($H_{ij}^{(1)} = m_{i+j+1} = H_{ji}^{(1)}$).

For any symmetric matrix \mathbf{A} and any matrix \mathbf{C} :

$$(\mathbf{C}\mathbf{A}\mathbf{C}^\top)^\top = (\mathbf{C}^\top)^\top \mathbf{A}^\top \mathbf{C}^\top = \mathbf{C}\mathbf{A}\mathbf{C}^\top$$

Therefore, $\mathbf{J} = \mathbf{C}\mathbf{H}^{(1)}\mathbf{C}^\top$ is symmetric.

Alternative view: The Jacobi matrix represents “multiply by x ” in the orthonormal polynomial basis. Since x is real and the inner product is symmetric, this operator is self-adjoint, guaranteeing a symmetric matrix representation.

2.7 Checkpoint 5.6: Root Count Understanding (Section 5.5)

Section 5.5 – Root Count Rule

Question:

- (a) Use Eq. (5.19) to verify the root counts in Table 5.1.
- (b) Explain why $(ss|ss)$ requires $n_r = 1$ even though $L = 0$. (*Hint: What is the minimum polynomial degree that must be integrated exactly?*)
- (c) What would happen if you used $n_r = 1$ for a $(pp|pp)$ integral? Which Boys functions would be incorrect?

Answer:

(a) Verifying root counts:

The formula is $n_r = \lfloor L/2 \rfloor + 1$ where $L = \ell_A + \ell_B + \ell_C + \ell_D$:

Shell quartet	$\ell_A + \ell_B + \ell_C + \ell_D$	L	$\lfloor L/2 \rfloor + 1$	n_r
$(ss ss)$	$0 + 0 + 0 + 0$	0	$0 + 1$	1
$(ps ss)$	$1 + 0 + 0 + 0$	1	$0 + 1$	1
$(pp ss)$	$1 + 1 + 0 + 0$	2	$1 + 1$	2
$(pp pp)$	$1 + 1 + 1 + 1$	4	$2 + 1$	3
$(dd pp)$	$2 + 2 + 1 + 1$	6	$3 + 1$	4
$(ff ff)$	$3 + 3 + 3 + 3$	12	$6 + 1$	7

(b) Why $(ss|ss)$ requires $n_r = 1$:

Even though $L = 0$, we still need to integrate at least degree-0 polynomials (constants) exactly. The quadrature must satisfy:

$$m_0 = \int_0^1 x^0 \cdot w_T(x) dx = W_1$$

With $n_r = 0$ (no nodes), we could not represent any integral at all. The formula gives:

$$n_r = \lfloor 0/2 \rfloor + 1 = 0 + 1 = 1$$

One root is the minimum needed to reproduce the zeroth moment $m_0 = 2F_0(T)$.

Physical interpretation: Even for s -functions, the electron repulsion integral involves the Boys function $F_0(T)$. This requires at least one quadrature point to evaluate.

(c) Using $n_r = 1$ for $(pppp)$:

For $(pp|pp)$, $L = 4$, so we need $n_r = 3$.

With only $n_r = 1$, the quadrature is exact for polynomials of degree $\leq 2(1) - 1 = 1$.

We can compute:

- $F_0(T) = \frac{1}{2}W_1$ (**correct**)
- $F_1(T) = \frac{1}{2}W_1x_1$ (**correct**)
- $F_2(T) = \frac{1}{2}W_1x_1^2$ (**INCORRECT**)
- $F_3(T) = \frac{1}{2}W_1x_1^3$ (**INCORRECT**)
- $F_4(T) = \frac{1}{2}W_1x_1^4$ (**INCORRECT**)

The ERI for $(pppp)$ requires Boys functions through F_4 . With $n_r = 1$, F_2, F_3, F_4 would all be computed incorrectly, leading to wrong integral values.

The errors can be substantial. For example, at $T = 1.0$:

- $F_2(1.0)_{\text{exact}} \approx 0.0818$
- $F_2(1.0)_{1\text{-point}} = \frac{1}{2}W_1x_1^2 \approx 0.048$ (error $\sim 40\%$)

Warning: Common Error

Using insufficient Rys roots is a silent error—the code runs but produces wrong answers. Always verify root counts match the angular momentum of your shell quartet.

2.8 Checkpoint 5.6b: Resolving the Circularity (Section 5.6)

Section 5.6 – Why Rys Quadrature?

Question:

- Explain in your own words why the “circularity” in Rys quadrature is actually not circular for integrals with $L > 0$.
- For a $(pp|pp)$ shell quartet ($L = 4$), how many Rys roots are needed? How many individual Cartesian integrals share these same nodes and weights?
- If someone proposed using 10-point Gauss–Legendre quadrature instead of Rys quadrature for ERI evaluation, what would be wrong with this approach?

Answer:**(a) Why the apparent circularity is not circular for $L > 0$:**

The apparent circularity is: “We compute Boys functions to get Rys nodes/weights, then use those to compute Boys functions.” This seems pointless—but it misses the key distinction between **scalar values** and **structured integrands**.

For $(ss|ss)$ ($L = 0$): The integral is simply a prefactor times $F_0(T)$. Direct Boys evaluation is all you need. The Rys machinery is unnecessary (though it works as a validation test).

For $L > 0$: The integrand has the form

$$\int_0^1 P(t) \cdot w_T(t^2) dt$$

where $P(t) = I_x(t) \cdot I_y(t) \cdot I_z(t)$ is a polynomial of degree up to L that encodes the angular momentum structure and depends on the specific Cartesian component being computed. This polynomial **cannot** be factored out of the integral. You must evaluate $P(t)$ at specific quadrature points $t_i = \sqrt{x_i}$ and sum:

$$\text{ERI} = \sum_{i=1}^{n_r} W_i \cdot P(t_i)$$

The “circularity” breaks because:

- We compute scalar Boys values $F_n(T)$ to build the quadrature **once per shell quartet**
- We then evaluate **many different polynomials** $P(t)$ at the quadrature nodes
- The polynomial $P(t)$ varies for each Cartesian component; the nodes/weights do not

The Rys quadrature transforms the problem from “evaluate complicated integrals” to “evaluate simple polynomials at known points and sum.”

(b) $(pp|pp)$ analysis:**Root count:**

$$L = \ell_A + \ell_B + \ell_C + \ell_D = 1 + 1 + 1 + 1 = 4$$

$$n_r = \lfloor L/2 \rfloor + 1 = \lfloor 4/2 \rfloor + 1 = 2 + 1 = 3$$

So $(pp|pp)$ requires **3 Rys roots**.

Number of Cartesian integrals: Each p -shell has 3 Cartesian components: p_x, p_y, p_z .

$$\text{Total integrals} = 3 \times 3 \times 3 \times 3 = 81$$

Amortization: The 3 Rys nodes and 3 weights (computed once from moments m_0, m_1, \dots, m_5) are shared by all 81 integrals. The expensive Boys function calls are amortized over a large batch.

(c) Why Gauss–Legendre would be wrong:

Gauss–Legendre quadrature is designed for the weight function $w(x) = 1$ on $[-1, 1]$. Using it for ERI evaluation has two fatal problems:

Problem 1: Wrong weight function. The Rys weight is $w_T(x) = x^{-1/2}e^{-Tx}$, not $w(x) = 1$. Gauss–Legendre nodes and weights do not account for this weight. The quadrature formula

$$\int_0^1 f(x) \cdot w_T(x) dx \approx \sum_{i=1}^n W_i^{GL} f(x_i^{GL})$$

is simply wrong—it would compute $\int f(x) dx$, not $\int f(x)w_T(x) dx$.

Problem 2: Inefficiency. Even if you included $w_T(x)$ in the integrand (i.e., computed $\int f(x)w_T(x) dx$ using GL quadrature on $f(x)w_T(x)$), you would lose the exactness property. Gauss–Legendre is exact for polynomials, but $P(x) \cdot w_T(x)$ is not a polynomial (it has the $x^{-1/2}e^{-Tx}$ factor). You would need many more GL points to achieve the same accuracy that Rys quadrature achieves exactly with n_r points.

The key insight: Rys quadrature is Gaussian quadrature **for the specific weight function** $w_T(x)$. This makes it exact for polynomial integrands of the form $P(x) \cdot w_T(x)$ with minimal points. Using any other quadrature rule (Gauss–Legendre, Simpson’s, etc.) would either give wrong answers or require vastly more points.

2.9 Checkpoint 5.7: Method Equivalence (Section 5.7 – ERIs via Boys and Rys)

Section 5.6 – Validation Strategy

Question:

- Why does the Rys formula $F_0(T) = \frac{1}{2} \sum_i W_i$ not depend on the nodes x_i ?
- Under what conditions might direct F_0 evaluation be preferable to Rys quadrature? (*Hint: Consider computational cost for a single isolated ERI versus a batch of ERIs sharing the same T .*)
- If your Rys implementation has a sign error in the Cholesky factorization, would this (ss|ss) test catch it? Why or why not?

Answer:

(a) Why F_0 is node-independent:

The general formula is:

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

For $n = 0$:

$$F_0(T) = \frac{1}{2} \sum_{i=1}^{n_r} W_i x_i^0 = \frac{1}{2} \sum_{i=1}^{n_r} W_i$$

Since $x_i^0 = 1$ for any x_i , the nodes cancel out. Only the sum of weights matters.

Mathematical interpretation: $F_0(T)$ is half the zeroth moment $m_0 = 2F_0(T) = \sum_i W_i$, which by construction is preserved exactly regardless of where the nodes are placed.

(b) When direct Boys evaluation is preferable:

Direct evaluation is better when:

- Computing a single isolated ERI (setup cost of Rys quadrature not amortized)
- Only F_0 is needed (no benefit from reproducing higher moments)
- T is very small or very large (extreme regimes where Rys construction can be unstable)

Rys quadrature is better when:

- Many ERIs share the same T value (construct nodes/weights once, reuse)
- Multiple Boys orders are needed (get F_0, \dots, F_{2n_r-1} from one construction)
- Integrating in the full Rys framework (nodes serve double duty in ERI recurrences)

(c) Would a sign error be caught by $(ss|ss)$?

Probably not. Here's why:

A sign error in Cholesky might produce \mathbf{L} with wrong signs in off-diagonal elements. However:

- For $n_r = 1$, \mathbf{H} is 1×1 : $\mathbf{H} = (m_0)$, so $L = \sqrt{m_0}$
- There are no off-diagonal elements to get wrong
- The weight $W_1 = m_0(V_{01})^2$ involves a square, so sign errors in V_{01} don't matter

A more robust test would use $n_r \geq 2$, where the Hankel matrix has off-diagonal elements and sign errors in the Cholesky factor would propagate to incorrect nodes and weights.

Recommendation: Test with $n_r = 2$ and verify all four moments m_0, m_1, m_2, m_3 to properly validate the implementation.

2.10 Checkpoint 5.8: Identifying Derivative Sources (Section 5.8)

Section 5.7 – Derivative Identity

Question: In the derivation of Eq. (5.28), which term comes from differentiating the Gaussian prefactor $\exp[-\mu R_{AB}^2]$, and which term comes from differentiating the Boys function?

Answer:

The $(p_\xi s|ss)$ formula (Eq. 5.28) is:

$$(p_\xi b|cd) = \frac{2\pi^{5/2}}{pq\sqrt{p+q}} e^{-\mu R_{AB}^2} e^{-\nu R_{CD}^2} \left[-\frac{\beta}{p}(A_\xi - B_\xi)F_0(T) - \frac{\rho}{p}(P_\xi - Q_\xi)F_1(T) \right]$$

Term 1: From differentiating the Gaussian prefactor

$$\boxed{-\frac{\beta}{p}(A_\xi - B_\xi)F_0(T)}$$

This comes from:

$$\frac{\partial}{\partial A_\xi} (-\mu R_{AB}^2) = -2\mu(A_\xi - B_\xi)$$

When combined with the $1/(2\alpha)$ from the derivative identity and using $\mu/\alpha = \beta/p$, we get the factor $-(\beta/p)(A_\xi - B_\xi)$.

The Boys function remains $F_0(T)$ because this differentiation acts on the exponential, not on the Boys function.

Term 2: From differentiating the Boys function

$$-\frac{\rho}{p}(P_\xi - Q_\xi)F_1(T)$$

This comes from applying the chain rule to $F_0(T)$ where $T = \rho|\mathbf{P} - \mathbf{Q}|^2$:

$$\frac{\partial}{\partial A_\xi} F_0(T) = \frac{dF_0}{dT} \frac{\partial T}{\partial A_\xi} = -F_1(T) \cdot 2\rho(P_\xi - Q_\xi) \frac{\alpha}{p}$$

using the Boys derivative identity $dF_0/dT = -F_1(T)$.

Physical interpretation:

- The F_0 term: Accounts for the shift in the “bra” pair center due to the p -function’s angular factor
- The F_1 term: Accounts for the change in Coulomb interaction distance as the angular momentum shifts probability density

2.11 Checkpoint 5.9: Structure of the $(ps|ss)$ Formula (Section 5.8)

Section 5.7 – $(ps|ss)$ Formula

Question: Equation (5.28) contains two terms with different geometric factors. One involves $(A_\xi - B_\xi)$, the other involves $(P_\xi - Q_\xi)$. What happens to each term when $\mathbf{A} = \mathbf{B}$ (i.e., both functions on the same center)? What physical situation does this represent?

Answer:

When $\mathbf{A} = \mathbf{B}$:

If the bra functions are on the same center, then:

- $A_\xi - B_\xi = 0$ for all ξ
- $\mathbf{P} = (\alpha\mathbf{A} + \beta\mathbf{B})/p = \mathbf{A}$ (composite center coincides with both centers)

The formula simplifies to:

$$(p_\xi s|ss) = \frac{2\pi^{5/2}}{pq\sqrt{p+q}} e^{-\nu R_{CD}^2} \left[0 - \frac{\rho}{p}(A_\xi - Q_\xi)F_1(T) \right]$$

Only the F_1 term survives! The F_0 term vanishes entirely.

Physical interpretation:

This represents an ERI where both the p and s functions are centered on the same atom (say, a $2p$ and $1s$ on the same atom).

The first term, proportional to $(A_\xi - B_\xi)$, represents the “two-center Gaussian product” contribution. When both functions are on the same center, there is no separation and this contribution vanishes.

The second term, proportional to $(P_\xi - Q_\xi) = (A_\xi - Q_\xi)$, represents the electron-electron interaction between the bra pair (at \mathbf{A}) and the ket pair (at \mathbf{Q}). This remains nonzero as long as the ket functions are on different centers from the bra functions.

Special case: Four-center collapse

If additionally $\mathbf{C} = \mathbf{D} = \mathbf{A}$ (all functions on the same atom), then $\mathbf{Q} = \mathbf{A}$, so $(A_\xi - Q_\xi) = 0$ as well.

For certain symmetry components (like p_x when the molecule is symmetric about the x -axis), the integral may vanish entirely. This is used in integral screening based on symmetry.

2.12 Checkpoint 5.10: Crossed Indices in Exchange (Section 5.9)

Section 5.8 – J and K Contractions

Question: In Eq. (5.35), why are the indices “crossed” compared to Eq. (5.34)? Relate this to the structure of exchange (fermionic antisymmetry).

Answer:

The contractions are:

$$\text{Coulomb: } J_{\mu\nu} = \sum_{\lambda\sigma} (\mu\nu|\lambda\sigma) P_{\lambda\sigma} \quad (7)$$

$$\text{Exchange: } K_{\mu\nu} = \sum_{\lambda\sigma} (\mu\lambda|\nu\sigma) P_{\lambda\sigma} \quad (8)$$

Physical origin of the crossing:

The Coulomb and exchange integrals arise from expanding $\langle \Psi | \hat{V}_{ee} | \Psi \rangle$ where Ψ is a Slater determinant.

For a two-electron system with orbitals ϕ_i and ϕ_j , the determinant gives:

$$\Psi(1,2) = \frac{1}{\sqrt{2}} [\phi_i(1)\phi_j(2) - \phi_j(1)\phi_i(2)]$$

Computing $\langle \Psi | \frac{1}{r_{12}} | \Psi \rangle$:

$$\langle \Psi | \frac{1}{r_{12}} | \Psi \rangle = \frac{1}{2} [\langle \phi_i\phi_j | \phi_i\phi_j \rangle + \langle \phi_j\phi_i | \phi_j\phi_i \rangle] \quad (9)$$

$$- \langle \phi_i\phi_j | \phi_j\phi_i \rangle - \langle \phi_j\phi_i | \phi_i\phi_j \rangle] \quad (10)$$

The first two terms (Coulomb-type) have the same orbital on each electron before and after the operator:

$$\text{Coulomb: } \langle \phi_i(1)\phi_j(2) | \frac{1}{r_{12}} | \phi_i(1)\phi_j(2) \rangle = (ii|jj)$$

The last two terms (Exchange-type) have orbitals swapped:

$$\text{Exchange: } \langle \phi_i(1)\phi_j(2) | \frac{1}{r_{12}} | \phi_j(1)\phi_i(2) \rangle = (ij|ji)$$

In the AO basis:

Expanding MOs in AOs: $\phi_i = \sum_{\mu} C_{\mu i} \chi_{\mu}$

The Coulomb contribution becomes:

$$\sum_{\mu\nu\lambda\sigma} C_{\mu i} C_{\nu i}^* (\mu\nu|\lambda\sigma) C_{\lambda j} C_{\sigma j}^* \propto (\mu\nu|\lambda\sigma)$$

The Exchange contribution has the swap:

$$\sum_{\mu\nu\lambda\sigma} C_{\mu i} C_{\lambda i}^* (\mu\lambda|\nu\sigma) C_{\nu j} C_{\sigma j}^* \propto (\mu\lambda|\nu\sigma)$$

The λ and ν indices are swapped in the second bra position and first ket position—this is the “crossing” visible in the exchange formula.

Summary:

	Coulomb	Exchange
ERI index pattern	$(\mu\nu \lambda\sigma)$	$(\mu\lambda \nu\sigma)$
Physical meaning	Classical repulsion	Fermionic exchange
Sign in Fock matrix	+1	$-\frac{1}{2}$ (RHF)

2.13 Checkpoint 5.11: Symmetry of J and K (Section 5.9)

Section 5.8 – Matrix Symmetry

Question: Both **J** and **K** inherit symmetry from the ERIs and density matrix. Using the 8-fold ERI symmetry and the fact that **P** is symmetric for real orbitals, show that $J_{\mu\nu} = J_{\nu\mu}$ and $K_{\mu\nu} = K_{\nu\mu}$.

Answer:

8-fold ERI symmetry (real basis):

$$(\mu\nu|\lambda\sigma) = (\nu\mu|\lambda\sigma) = (\mu\nu|\sigma\lambda) = (\nu\mu|\sigma\lambda) \quad (11)$$

$$= (\lambda\sigma|\mu\nu) = (\sigma\lambda|\mu\nu) = (\lambda\sigma|\nu\mu) = (\sigma\lambda|\nu\mu) \quad (12)$$

Key symmetries we'll use:

- $(\mu\nu|\lambda\sigma) = (\nu\mu|\lambda\sigma)$ (swap first pair)
- $(\mu\nu|\lambda\sigma) = (\lambda\sigma|\mu\nu)$ (swap bra and ket)
- $(\mu\lambda|\nu\sigma) = (\nu\sigma|\mu\lambda)$ (combined)

Proof that $J_{\mu\nu} = J_{\nu\mu}$:

$$J_{\mu\nu} = \sum_{\lambda\sigma} (\mu\nu|\lambda\sigma) P_{\lambda\sigma} \quad (13)$$

$$= \sum_{\lambda\sigma} (\nu\mu|\lambda\sigma) P_{\lambda\sigma} \quad (\text{swap first pair}) \quad (14)$$

$$= J_{\nu\mu} \quad (15)$$

Proof that $K_{\mu\nu} = K_{\nu\mu}$:

This requires more care due to the crossed index structure:

$$K_{\mu\nu} = \sum_{\lambda\sigma} (\mu\lambda|\nu\sigma) P_{\lambda\sigma} \quad (16)$$

Using the symmetry $(\mu\lambda|\nu\sigma) = (\nu\sigma|\mu\lambda)$:

$$K_{\mu\nu} = \sum_{\lambda\sigma} (\nu\sigma|\mu\lambda) P_{\lambda\sigma} \quad (17)$$

Relabeling dummy indices $\lambda \leftrightarrow \sigma$:

$$K_{\mu\nu} = \sum_{\sigma\lambda} (\nu\lambda|\mu\sigma) P_{\sigma\lambda} \quad (18)$$

Using $P_{\sigma\lambda} = P_{\lambda\sigma}$ (symmetry of density matrix):

$$K_{\mu\nu} = \sum_{\lambda\sigma} (\nu\lambda|\mu\sigma) P_{\lambda\sigma} = K_{\nu\mu} \quad (19)$$

Consequence:

Both **J** and **K** are symmetric matrices. Therefore:

- The Fock matrix $\mathbf{F} = \mathbf{h} + \mathbf{J} - \frac{1}{2}\mathbf{K}$ is symmetric
- We only need to compute half the matrix elements (upper or lower triangle)
- The Roothaan-Hall equations have real eigenvalues

2.14 Checkpoint 5.12: Why V_{HF} Rather Than Separate \mathbf{J} and \mathbf{K} (Section 5.10)

Section 5.9 – Validation Approach

Question: Why is it meaningful to compare $V_{HF} = \mathbf{J} - \frac{1}{2}\mathbf{K}$ rather than \mathbf{J} and \mathbf{K} separately when validating an RHF implementation?

Answer:

Physical reason:

The quantity that enters the Fock matrix and determines physical observables is always the combination:

$$\mathbf{G} = \mathbf{J} - \frac{1}{2}\mathbf{K} = \mathbf{V}_{HF}$$

Individual \mathbf{J} and \mathbf{K} matrices depend on implementation details (how ERIs are stored, symmetry conventions, etc.), but their combination is unambiguous.

Practical reasons:

1. **Convention differences:** Some codes define K with a factor of 2 already included, or use different index conventions. The Fock matrix $\mathbf{F} = \mathbf{h} + \mathbf{J} - \frac{1}{2}\mathbf{K}$ is invariant to these conventions.
2. **Numerical stability:** For some density matrices, \mathbf{J} and \mathbf{K} individually might have large elements that partially cancel. Comparing them separately could show spurious differences that cancel in the physically meaningful combination.
3. **Algorithmic variations:** Different integral engines may compute $\mathbf{J} - \frac{1}{2}\mathbf{K}$ directly (especially with density fitting) rather than computing \mathbf{J} and \mathbf{K} separately. Validating against the combined quantity allows flexibility in implementation.
4. **Error propagation:** A sign error in \mathbf{K} (a common mistake) would show up as a factor-of-2 error in \mathbf{G} , making it easy to diagnose. If validating \mathbf{J} and \mathbf{K} separately, you might mistakenly think your \mathbf{K} is “just a factor of 2 off” when actually the physics would be completely wrong.

Energy validation:

The electronic energy:

$$E_{\text{elec}} = \text{tr}\{\mathbf{P}\mathbf{h}\} + \frac{1}{2} \text{tr}\left\{\mathbf{P}\left(\mathbf{J} - \frac{1}{2}\mathbf{K}\right)\right\} = \text{tr}\{\mathbf{P}\mathbf{h}\} + \frac{1}{2} \text{tr}\{\mathbf{P}\mathbf{G}\}$$

Again, only the combination \mathbf{G} appears, not \mathbf{J} or \mathbf{K} individually.

Bottom line: Validate against physically meaningful quantities. For RHF, that means \mathbf{G} , total energy, and MO eigenvalues—not intermediate quantities whose definitions may vary.

3 Lab Solutions

3.1 Lab 5A: Construct Rys Nodes/Weights and Verify Moment Matching

Lab 5A Objectives

- Implement Algorithm 5.1 (Hankel + Cholesky + Golub-Welsch)
- Verify moment matching for $n = 0, 1, \dots, 2n_r - 1$
- Test across a range of T values

Expected numerical results for $n_r = 2$:

T	x_1	x_2	W_1	W_2
0.0	0.1156	0.7416	1.3043	0.6957
0.1	0.1133	0.7373	1.2802	0.6551
1.0	0.0955	0.6956	1.0999	0.3938
10.0	0.0274	0.2711	0.5086	0.0519

Moment matching errors:

For properly implemented code, all moment-matching errors should be $< 10^{-12}$.

Expected Output

```

T = 0.0, nroots = 2
n= 0: m_exact=2.000000000000e+00  m_quad=2.000000000000e+00  diff=+0.000e+00
n= 1: m_exact=6.666666666667e-01  m_quad=6.666666666667e-01  diff=-1.110e-16
n= 2: m_exact=4.000000000000e-01  m_quad=4.000000000000e-01  diff=+5.551e-17
n= 3: m_exact=2.857142857143e-01  m_quad=2.857142857143e-01  diff=-5.551e-17

T = 1.0, nroots = 2
n= 0: m_exact=1.493648265624e+00  m_quad=1.493648265624e+00  diff=+2.220e-16
n= 1: m_exact=3.788393107578e-01  m_quad=3.788393107578e-01  diff=+5.551e-17
n= 2: m_exact=1.636351336279e-01  m_quad=1.636351336279e-01  diff=-2.776e-17
n= 3: m_exact=9.078568549346e-02  m_quad=9.078568549346e-02  diff=-1.388e-17

T = 10.0, nroots = 2
n= 0: m_exact=3.158296978397e-01  m_quad=3.158296978397e-01  diff=+5.551e-17
n= 1: m_exact=3.452149952927e-02  m_quad=3.452149952927e-02  diff=+6.939e-18
n= 2: m_exact=8.160693108665e-03  m_quad=8.160693108665e-03  diff=+8.674e-19
n= 3: m_exact=2.686600954609e-03  m_quad=2.686600954609e-03  diff=-1.084e-19

```

Key observations:

1. All moments m_0, m_1, m_2, m_3 are reproduced to machine precision ($\sim 10^{-15}$)
2. The construction is stable across a wide range of T (from 0 to 10+)
3. Nodes cluster toward smaller x as T increases (weight function peaks near $x = 0$)
4. All nodes lie in $(0, 1)$ as required
5. All weights are positive

Validation criteria:

Check	Tolerance
$ m_n^{\text{quad}} - m_n^{\text{exact}} $	$< 10^{-12}$
All $x_i \in (0, 1)$	Exact
All $W_i > 0$	Exact
$\sum_i W_i = m_0$	$< 10^{-14}$

Warning: Common Implementation Errors

1. **Wrong moment count:** Using m_0, \dots, m_{n_r-1} instead of m_0, \dots, m_{2n_r-1}
2. **Hankel indexing:** Off-by-one errors in $H_{ij} = m_{i+j}$ vs m_{i+j-1}
3. **Forgetting symmetrization:** The Jacobi matrix may have tiny asymmetry due to rounding; symmetrize before eigendecomposition
4. **Boys function accuracy:** Using only the series or only the recurrence; need both for full T range

3.2 Lab 5B: Compute $(ss|ss)$ ERI Using Rys Quadrature**Lab 5B Objectives**

- Use Rys quadrature to evaluate $F_0(T)$
- Compute a normalized $(ss|ss)$ ERI
- Validate against PySCF int2e

Test case:

- Molecule: Two H atoms along z -axis, $R = 1.4$ Bohr
- Exponents: $\alpha = 0.5$, $\beta = 0.4$
- Computing: $(aa|bb)$ where a is on atom 1, b is on atom 2

Expected numerical results:**Expected Output**

```
(aa|bb) PySCF = 0.31009149920742376
(aa|bb) Rys   = 0.3100914992074238
difference    = 1.1102230246251565e-16
```

Intermediate values for verification:

Quantity	Value
$p = \alpha + \beta$	0.9
$q = \gamma + \delta$ (same as p for this test)	0.9
$\mu = \alpha\beta/p$	0.222...
$\rho = pq/(p+q)$	0.45
$R_{PQ} = \mathbf{P} - \mathbf{Q} = 1.4$	1.4 Bohr
$T = \rho \cdot R_{PQ}^2$	0.882
$F_0(T)$	0.7937...
Prefactor $\frac{2\pi^{5/2}}{pq\sqrt{p+q}}$	40.89...
Gaussian prefactors	$e^0 \cdot e^{-\nu R_{CD}^2}$

Validation criteria:

Check	Tolerance
$ \text{ERI}_{\text{Rys}} - \text{ERI}_{\text{PySCF}} $	$< 10^{-12}$

Key code verification:

```

1 # The key formula
2 pref = 2 * (math.pi**2.5) / (p*q*math.sqrt(p+q))
3 F0 = 0.5 * float(np.sum(W)) # Rys quadrature for F_0
4 val_unnorm = pref * math.exp(-mu*Rab2 - nu*Rcd2) * F0
5 val_norm = Ns(alpha)*Ns(beta)*Ns(gamma)*Ns(delta)*val_unnorm

```

3.3 Lab 5C: Compute $(p_\xi s|ss)$ Using Derivative Identity

Lab 5C Objectives

- Implement Eq. (5.28) for $(p_\xi s|ss)$ ERIs
- Use Rys quadrature for both $F_0(T)$ and $F_1(T)$
- Validate against PySCF for multiple axis choices

Test case:

- Molecule: Two H atoms along z -axis, $R = 1.4$ Bohr
- Atom 1 (A): p -shell with exponent $\alpha = 0.5$
- Atom 2 (B): s -shell with exponent $\beta = 0.4$
- Computing: $(p_z@1, s@2|s@2, s@2)$

Expected numerical results:

Expected Output

```

AO labels:
0 0 H01 px
1 0 H01 py
2 0 H01 pz
3 1 H02 s

```

```
(p_z s|ss) PySCF = -0.04584115117181389
(p_z s|ss) Rys    = -0.04584115117181389
difference         = 0.0

(p_x s|ss) PySCF = 0.0
(p_x s|ss) Rys    = 0.0
difference         = 0.0
```

Key observations:

1. p_z integral is nonzero because atoms are separated along z
2. p_x and p_y integrals are zero by symmetry (perpendicular to bond axis)
3. The F_1 term contributes meaningfully—this is the first case where nodes matter

Formula verification:

The unnormalized formula is:

$$(p_\xi b|cd) = \frac{2\pi^{5/2}}{pq\sqrt{p+q}} e^{-\mu R_{AB}^2} e^{-\nu R_{CD}^2} \left[-\frac{\beta}{p}(A_\xi - B_\xi)F_0(T) - \frac{\rho}{p}(P_\xi - Q_\xi)F_1(T) \right]$$

For our test case with atoms at $(0,0,0)$ and $(0,0,1.4)$:

- $A_z - B_z = 0 - 1.4 = -1.4$
- $P_z = \alpha \cdot 0/p = 0$ (since A is at origin)
- $Q_z = (\beta \cdot 1.4 + \beta \cdot 1.4)/(2\beta) = 1.4$
- $P_z - Q_z = 0 - 1.4 = -1.4$

Both terms contribute with the same sign, giving a negative integral value.

Validation criteria:

Check	Tolerance
$ \text{ERI}_{\text{Rys}} - \text{ERI}_{\text{PySCF}} $ (nonzero cases)	$< 10^{-10}$
$ \text{ERI} $ for symmetry-zero cases	$< 10^{-14}$

3.4 Lab 5D: Build J and K from ERIs**Lab 5D Objectives**

- Implement J and K matrix construction via `einsum`
- Validate against PySCF `get_jk`
- Verify energy reconstruction

Test system: H₂O / STO-3G

- Geometry: O at origin, H at $(\pm 0.7586, 0, 0.5043)$ Angstrom
- Basis: STO-3G (7 AOs)
- Use converged SCF density from PySCF

Expected numerical results:

Expected Output

```

||J - J_ref||_F = 1.2432e-14
||K - K_ref||_F = 8.9547e-15
||VHF - VHF_ref||_F = 1.1025e-14

Energy reconstruction check:
  E_elec (from integrals) = -84.58939189
  E_elec (from PySCF)      = -84.58939189
  Difference                = 0.00e+00 Hartree

  E_nuc = 9.64350679
  E_tot = -74.94588510 Hartree
  PySCF E_tot = -74.94588510 Hartree

```

Key formulas verified:

```

1 # J and K contractions
2 J = np.einsum("ijkl,kl->ij", eri, P, optimize=True)
3 K = np.einsum("ikjl,kl->ij", eri, P, optimize=True)
4
5 # Two-electron Fock contribution
6 VHF = J - 0.5 * K
7
8 # Energy reconstruction
9 h = T + V # Core Hamiltonian
10 E_elec = np.einsum('ij,ji->', P, h) + 0.5 * np.einsum('ij,ji->', P,
    VHF)
11 E_tot = E_elec + mol.energy_nuc()

```

Validation criteria:

Check	Tolerance
$\ \mathbf{J} - \mathbf{J}_{\text{ref}}\ _F$	$< 10^{-12}$
$\ \mathbf{K} - \mathbf{K}_{\text{ref}}\ _F$	$< 10^{-12}$
$\ \mathbf{V}_{HF} - \mathbf{V}_{HF,\text{ref}}\ _F$	$< 10^{-12}$
$ E_{\text{tot}} - E_{\text{tot,PySCF}} $	$< 10^{-10}$ Hartree

Warning: Common Student Errors in Lab 5D

1. Wrong einsum indices for K:

```

1 # WRONG:
2 K = np.einsum("ijkl,jk->il", eri, P) # This is NOT the
    exchange!
3
4 # CORRECT:
5 K = np.einsum("ikjl,kl->ij", eri, P)

```

The exchange has a “crossed” index structure.

2. Missing factor of 1/2 in energy:

```

1 # WRONG:
2 E_elec = np.trace(P @ h) + np.trace(P @ VHF)
3
4 # CORRECT:

```

```
5 | E_elec = np.trace(P @ h) + 0.5 * np.trace(P @ VHF)
```

The two-electron term is counted twice in the Fock matrix approach.

3. **Confusing Frobenius norm with element-wise comparison:** $\|\mathbf{J} - \mathbf{J}_{\text{ref}}\|_F$ can be small even if some elements differ more than others. For debugging, also check $\max |J_{ij} - J_{ij}^{\text{ref}}|$.

4 Additional Notes for Instructors

4.1 Common Misconceptions

1. **“Rys quadrature is just another numerical integration method”:** Unlike Simpson’s rule or Gauss-Legendre, Rys quadrature is *exact* for the polynomial degrees it targets. For ERIs, this means no numerical integration error—only floating-point precision limits accuracy.
2. **“More roots is always better”:** Using more roots than necessary is wasteful and can introduce numerical instability (ill-conditioned Hankel matrix). Use exactly $n_r = \lfloor L/2 \rfloor + 1$.
3. **“The nodes don’t matter for $(ss|ss)$ ”:** While true that only F_0 is needed and only weights appear in the formula, this is the exception. For any higher angular momentum, nodes are essential.
4. **“Coulomb and exchange are just different ways of summing ERIs”:** They represent fundamentally different physics: classical electrostatics vs. quantum mechanical exchange arising from fermionic antisymmetry.

4.2 Suggested Discussion Questions

1. Why does libcint use specialized polynomial approximations for Rys roots rather than the moment-based Algorithm 5.1?
2. How would you modify the $(p_\xi s|ss)$ derivation to get $(p_\xi p_\eta|ss)$? What additional Boys functions appear?
3. The exchange matrix \mathbf{K} is more expensive to compute than \mathbf{J} in direct SCF. Why? (Hint: Consider how the contraction indices relate to shell-pair screening.)
4. What happens to Rys quadrature accuracy when $T \rightarrow 0$ or $T \rightarrow \infty$?

4.3 Extensions for Advanced Students

1. Implement the $(pp|ss)$ ERI using double differentiation and validate against PySCF.
2. Study the Hankel matrix condition number as a function of n_r and T . At what point does the algorithm become unstable?
3. Compare the Rys quadrature approach to Obara-Saika recurrence relations for a simple case.
4. Implement Schwarz screening and measure how many ERIs can be skipped for H_2O in various basis sets.

5 References

1. Dupuis, M., Rys, J., & King, H.F. (1976). “Evaluation of molecular integrals over Gaussian basis functions.” *J. Chem. Phys.* **65**, 111–116.
2. Rys, J., Dupuis, M., & King, H.F. (1983). “Computation of electron repulsion integrals using the Rys quadrature method.” *J. Comput. Chem.* **4**, 154–157.
3. Golub, G.H. & Welsch, J.H. (1969). “Calculation of Gauss quadrature rules.” *Math. Comp.* **23**, 221–230.
4. Szabo, A. & Ostlund, N.S. (1989). *Modern Quantum Chemistry*. Dover Publications. Appendix A.
5. 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 5.12).

6.1 Exercise 5.1: One-Root Quadrature Formulas [Core]

One-Root Derivation

For $n_r = 1$, we have one node x_1 and one weight W_1 . The moment-matching conditions state:

$$n = 0 : \quad m_0 = W_1 x_1^0 = W_1 \quad (20)$$

$$n = 1 : \quad m_1 = W_1 x_1^1 = W_1 x_1 \quad (21)$$

Solving these simultaneously:

$$\boxed{W_1 = m_0 = 2F_0(T)}, \quad \boxed{x_1 = \frac{m_1}{m_0} = \frac{F_1(T)}{F_0(T)}}$$

Numerical Verification:

T	$F_0(T)$	$F_1(T)$	x_1	W_1
0.01	0.9900	0.3267	0.3300	1.980
0.1	0.9538	0.3040	0.3187	1.908
1.0	0.7468	0.1894	0.2536	1.494
10.0	0.1579	0.0173	0.1095	0.316

Validation:

- For $T = 1.0$: $x_1 \approx 0.254$, $W_1 \approx 1.494$ (matches expected values)
- Moment-matching errors: $|W_1 \cdot 1 - m_0| < 10^{-15}$, $|W_1 x_1 - m_1| < 10^{-15}$
- Both moments reproduced to machine precision

6.2 Exercise 5.2: Two-Root Quadrature and Moment Matching [Core]

Algorithm 5.1 for $n_r = 2$

For two-root quadrature, we need moments m_0, m_1, m_2, m_3 (i.e., $2n_r - 1 = 3$).

Step 1: Hankel matrices

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

Step 2: Cholesky factorization $\mathbf{H} = \mathbf{L}\mathbf{L}^\top$

Step 3: Jacobi matrix $\mathbf{J} = \mathbf{L}^{-1}\mathbf{H}^{(1)}\mathbf{L}^{-\top}$

Step 4: Eigendecomposition $\mathbf{J}\mathbf{V} = \mathbf{V}\mathbf{\Lambda}$ gives nodes x_i

Step 5: Weights $W_i = m_0(V_{0i})^2$

Expected Results for $T = 1.0$:

- Nodes: $x_1 \approx 0.0960$, $x_2 \approx 0.6969$

- Weights: $W_1 \approx 0.4063$, $W_2 \approx 1.0873$
- Moment errors for $n = 0, 1, 2, 3$: all $< 10^{-14}$
- Error for $n = 4$: $\sim 10^{-3}$ (beyond exactness guarantee)

Range Testing:

T	Max Error ($n \leq 3$)	Error at $n = 4$	Notes
10^{-8}	$< 10^{-14}$	$\sim 10^{-4}$	Series expansion needed
10^{-2}	$< 10^{-14}$	$\sim 10^{-3}$	Stable region
10^0	$< 10^{-14}$	$\sim 10^{-3}$	Stable region
10^2	$< 10^{-12}$	$\sim 10^{-2}$	Nodes cluster near $x = 0$

6.3 Exercise 5.3: $(ss|ss)$ by Three Routes [Core]**Three Computational Routes**

Setup: H_2 with $R = 1.4$ Bohr, exponents $\alpha = \beta = \gamma = \delta = 0.5$ (or as specified).

Route (a): Closed form with erf-based F_0

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

where $F_0(T) = \frac{1}{2} \sqrt{\frac{\pi}{T}} \text{erf}(\sqrt{T})$ for $T > 0$.

Route (b): Rys quadrature

$$F_0(T) = \frac{1}{2} \sum_{i=1}^{n_r} W_i$$

With $n_r = 1$: $F_0(T) = \frac{W_1}{2} = \frac{m_0}{2}$.

Route (c): PySCF int2e

Use `mol.intor("int2e")` and extract the appropriate element.

Expected Agreement:

- Routes (a), (b), and (c) should agree to $< 10^{-12}$ for normalized primitives
- If using unnormalized primitives, multiply by normalization constants:

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

Sample Values:

- For $\alpha = \beta = 0.5$, atoms at $(0, 0, 0)$ and $(0, 0, 1.4)$:
- $(aa|bb)_{\text{normalized}} \approx 0.310$ (depends on exact geometry/exponents)
- All three routes: difference $< 10^{-14}$

Discussion Points:

- Small- T regime ($T < 10^{-10}$): erf-based formula may need series expansion
- Normalization conventions: PySCF uses normalized contracted GTOs
- Rys approach: Reproduces F_0 exactly (within numerical precision)

6.4 Exercise 5.4: $(p_{\xi}s|ss)$ and the Appearance of F_1 [Core]

Derivative Identity

Starting from Eq. (5.25):

$$(p_{\xi}b|cd) = \frac{1}{2\alpha} \frac{\partial}{\partial A_{\xi}} (ab|cd)$$

Apply to the $(ss|ss)$ formula, differentiating both the Gaussian prefactor and $F_0(T)$:

Term 1 (from Gaussian prefactor):

$$\frac{\partial}{\partial A_{\xi}} (-\mu R_{AB}^2) = -2\mu(A_{\xi} - B_{\xi})$$

Contributes: $-\frac{\beta}{p}(A_{\xi} - B_{\xi})F_0(T)$

Term 2 (from Boys function):

$$\frac{dF_0}{dT} = -F_1(T), \quad \frac{\partial T}{\partial A_{\xi}} = 2\rho(P_{\xi} - Q_{\xi})\frac{\alpha}{p}$$

Contributes: $-\frac{\rho}{p}(P_{\xi} - Q_{\xi})F_1(T)$

Result (Eq. 5.28):

$$(p_{\xi}b|cd) = \frac{2\pi^{5/2}}{pq\sqrt{p+q}} e^{-\mu R_{AB}^2} e^{-\nu R_{CD}^2} \left[-\frac{\beta}{p}(A_{\xi} - B_{\xi})F_0(T) - \frac{\rho}{p}(P_{\xi} - Q_{\xi})F_1(T) \right]$$

Validation Cases:

- Symmetry-zero case:** Atoms along z -axis, compute $(p_x s|ss)$
 - Both $(A_x - B_x) = 0$ and $(P_x - Q_x) = 0$
 - Result: $(p_x s|ss) = 0$ (within 10^{-14})
- Nonzero case:** Atoms along z -axis, compute $(p_z s|ss)$
 - $(A_z - B_z) \neq 0$ and $(P_z - Q_z) \neq 0$
 - Agreement with PySCF to $< 10^{-10}$
- Sign test:** Swapping **A** \leftrightarrow **B** and $\alpha \leftrightarrow \beta$
 - Changes which center has the p -function
 - Sign of integral should change (or formula should be re-derived for $b = p$)

6.5 Exercise 5.5: J/K Build and Energy Component Check [Core]

J and K Matrix Construction

Coulomb matrix:

$$J_{\mu\nu} = \sum_{\lambda\sigma} (\mu\nu|\lambda\sigma) P_{\lambda\sigma}$$

Exchange matrix:

$$K_{\mu\nu} = \sum_{\lambda\sigma} (\mu\lambda|\nu\sigma) P_{\lambda\sigma}$$

In NumPy:

```

1 J = np.einsum("ijkl,k1->ij", eri, P, optimize=True)
2 K = np.einsum("ikjl,k1->ij", eri, P, optimize=True)

```

Energy reconstruction:

$$E_{\text{elec}} = \text{tr}\{\mathbf{P}\mathbf{h}\} + \frac{1}{2} \text{tr}\left\{\mathbf{P}\left(\mathbf{J} - \frac{1}{2}\mathbf{K}\right)\right\}$$

Test System: H₂O/STO-3G with geometry O at origin, H at ($\pm 0.7586, 0, 0.5043$) Å.

Expected Results:

- $\|\mathbf{J} - \mathbf{J}_{\text{PySCF}}\|_F < 10^{-12}$
- $\|\mathbf{K} - \mathbf{K}_{\text{PySCF}}\|_F < 10^{-12}$
- Energy agreement: $|E_{\text{tot}} - E_{\text{PySCF}}| < 10^{-10}$ Hartree
- PySCF reference: $E_{\text{tot}} \approx -74.9459$ Hartree

Warning: Common Errors

1. **Wrong einsum for K:** The exchange has “crossed” indices. Using "ijkl,jk->il" is incorrect.
2. **Missing factor of 1/2:** The energy has $\frac{1}{2}$ on the two-electron term.
3. **Using K instead of $\frac{1}{2}\mathbf{K}$:** In RHF, the Fock matrix is $\mathbf{F} = \mathbf{h} + \mathbf{J} - \frac{1}{2}\mathbf{K}$.

6.6 Exercise 5.6: Schwarz Screening Toy Study [Advanced]

Schwarz Inequality

The Schwarz bound states:

$$|(\mu\nu|\lambda\sigma)| \leq \sqrt{(\mu\nu|\mu\nu)} \cdot \sqrt{(\lambda\sigma|\lambda\sigma)}$$

This enables screening: if the bound is below threshold τ , skip the quartet.

Procedure:

1. Compute all diagonal ERIs $(\mu\nu|\mu\nu)$ for shell pairs
2. For each threshold $\tau \in \{10^{-8}, 10^{-10}, 10^{-12}\}$:
 - Count pairs where $\sqrt{(\mu\nu|\mu\nu)} < \tau$
 - Estimate screened quartets

Expected Results:

Basis	N_{ao}	Screening at 10^{-10}	Notes
STO-3G	7	$\sim 5\%$	Compact, minimal screening
6-31+G*	19	$\sim 25\%$	Diffuse functions help

Physical Interpretation:

- Compact basis functions (STO-3G): Large overlaps, few negligible integrals

- Diffuse functions (aug-cc-pVXZ, 6-31+G*): Smaller self-overlaps for diffuse-diffuse pairs
- Screening effectiveness increases with molecule size and diffuse character

6.7 Exercise 5.7: Hankel Matrix Conditioning [Advanced]

Conditioning Analysis

The Hankel matrix \mathbf{H} with entries $H_{ij} = m_{i+j}$ becomes increasingly ill-conditioned as n_r grows.

Condition number: $\kappa(\mathbf{H}) = \|\mathbf{H}\| \cdot \|\mathbf{H}^{-1}\|$

Use `np.linalg.cond(H)` for numerical evaluation.

Expected Results at $T = 1.0$:

n_r	$\kappa(\mathbf{H})$	Cholesky Status
1	1.0	Stable
2	$\sim 10^2$	Stable
3	$\sim 10^4$	Stable
4	$\sim 10^6$	Marginally stable
5	$\sim 10^8$	May show issues
6	$\sim 10^{10}+$	Often fails

Dependence on T :

- **Small T (< 0.1):** Moments become similar, worse conditioning
- **Large T (> 10):** Moments decay rapidly, may help slightly
- Conditioning generally worsens for all T as n_r increases

Implications:

- High angular momentum quartets need many roots (e.g., $(ff|ff)$ needs $n_r = 7$)
- Double precision limits practical use of Algorithm 5.1 to $n_r \lesssim 5$ –6
- Production codes (libcint) use polynomial approximations or extended precision

6.8 Exercise 5.8: Obara–Saika Recursion [Research/Challenge]

OS Vertical Recurrence

The Obara–Saika (OS) method builds higher angular momentum integrals recursively:

$$(a + 1_i b | c d) = (P_i - A_i)(a b | c d) + \frac{1}{2p} [N_i(a)(a - 1_i b | c d) + N_i(b)(a b - 1_i | c d)]$$

plus ket-side terms.

For $(ps|ss)$: Start from $(ss|ss)^{(m)}$ auxiliary integrals (involving F_m), then apply the recurrence.

Implementation Steps:

1. Research the OS formulas (Obara & Saika, JCP 84, 3963, 1986)
2. Implement auxiliary integrals $(ss|ss)^{(m)} \propto F_m(T)$

3. Apply vertical recurrence to get $(ps|ss)$
4. Compare with derivative-based formula

Validation:

- OS-based $(p_zs|ss)$ should match derivative-based result to $< 10^{-12}$
- Both require F_0 and F_1
- Computational structure differs: OS builds up from lower angular momentum

Comparison: Rys vs OS

Aspect	Rys Quadrature	Obara–Saika
Approach	Quadrature for Boys	Recurrence relations
Auxiliary quantities	Nodes, weights	Auxiliary integrals $(\cdot \cdot)^{(m)}$
Numerical stability	Issues at high n_r	Generally stable
Efficiency	Good for mixed ℓ	Good for high ℓ

Warning: Research Note

This exercise is open-ended and intended for students interested in exploring alternative integral algorithms. The OS method is widely used in modern quantum chemistry codes (e.g., Gaussian, Q-Chem) alongside Rys quadrature. Understanding both approaches provides insight into the design choices in integral libraries.