Available online at www.sciencedirect.com

ScienceDirect

Journal of Symbolic Computation 42 (2007) 265–289

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Mathscape and molecular integrals

Michael P. Barnett*

Meadow Lakes, East Windsor, NJ 08520, United States

Received 7 June 2005; accepted 13 July 2006
Available online 20 September 2006

Abstract

I have used MATHEMATICA to solve several problems that relate to the symbolic calculation of the 'molecular integrals' that are a mainstay of computational chemistry. This work has provided many new results of chemical and mathematical interest, and it has led to a powerful programming methodology that I call MATHSCAPE that uses a novel open ended set of macros. Some further work on molecular integrals is presented here, largely as an introduction to MATHSCAPE. I discuss (1) the immediate mathematical problem of the 'J' integrals, (2) the key features of MATHSCAPE, (3) a novel reduction of the J integrals, (4) the chemical context of this work, and (5) the computer science context.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: High level symbolic languages; Functional composition; Symbolic integration; Mechanized proof procedures; Computational chemistry; Special functions of mathematical physics

1. Introduction

In the course of applying MATHEMATICA to computational chemistry, I have developed some programming methods that can be transferred to symbolic calculation in many further areas of research. I write the mathematical derivations in a style that I call MATHSCAPE. A large part of my work deals with 'molecular integrals' that require $3n$-fold integration over the Euclidean space of n electrons ('atomic' and 'molecular' are chemical terms in this paper). The integrals are an ongoing topic of research by many authors around the world. Recent papers include Barnett (2003b), Cesco et al. (2005), Gumus (2005), Guseinov and Mamedov (2005), Harris (2002), Quiney and Wilson (2005), Rico et al. (2005) and Safouhi and Bouferguene (2006). These contain extensive bibliographies.

* Tel.: +1 609 426 6266.
E-mail address: michaelb@princeton.edu.

0747-7171/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jsc.2006.07.002

My reasons for developing the MATHSCAPE style are discussed in Section 2 and its principles are listed in Section 3. Then these are explained in Sections 4–6 using pieces of some scripts that evaluate 'overlap integrals' symbolically. These are molecular integrals that relate to a system of two nuclei and a single electron. The mathematical problem is stated in Section 4 together with a set of algorithms to deal with it. These use ellipsoidal coordinates and feed directly into the built-in integration resources of MATHEMATICA and other symbolic calculation platforms. Most of my past work on molecular integrals has used polar coordinates and the 'molecular ζ-functions' that I introduced in Barnett and Coulson (1951) and discussed most recently in Barnett (2003b). These are needed for several kinds of molecular integral that are beyond the scope of the ellipsoidal coordinate method. This paper contains new material on integration in ellipsoidal coordinates, which is easier to explain for integrals that it covers than the ζ-function method.

Several MATHSCAPE scripts that are used in the symbolic calculation of the overlap integrals are explained in Sections 5 and 6. Some further background of the molecular integral problem is given in Section 7. The work is put in a broader computer science context, particularly in relation to THEOREMA, in Section 8.

Further details of the calculations reported here are on http://www.princeton.edu/~allengrp/ms/mmi.

2. Reasons for mathscape

In the course of over 50 years of intermittent work on molecular integrals, starting in 1948, I handwrote, typed and proofread dozens of accounts of the derivations that contained different notations and different amounts of detail for numerous papers and reports (see *e.g.* Barnett (1963) and Barnett and Coulson (1951)). In 1990, MATHEMATICA presented an opportunity to reconstruct all the derivations in machine readable files that could be adapted mechanically for variant needs without ever writing or typing any of the hundreds of definitions and intermediate steps and final results by hand again. This was a major motivation for the MATHSCAPE style. It has paid off on several occasions when I found, after coding the construction of a long sequence of formulas, that I needed to redefine the objects in some of the early steps of a derivation, and I was able to make the systematic variations with mechanical ease.

Another motivation came from my early experience of numerical computation. I was strongly influenced in the 1950s by leading makers of numerical tables, who placed paramount emphasis on the possibility of transcription errors and the necessity for checks. Years of experience of numerical computations gave continuing reinforcement of this need. So did my early encounters with MATHEMATICA — among other surprises this factored $xy/2$ into x, y and $1/4$. Symbolic calculation software has improved since then, but I continue to make mistakes and the need to check remains. Although this need is rarely mentioned, other authors consider it essential, too (Gracey, 1998; Wang and Kuppermann, 2006).

I was first alerted to the prospect of symbolic calculation by S.F. Boys in the 1950s. He programmed the EDSAC computer to manipulate arrays of coefficients that represent molecular integrals containing Gaussian factors e^{-kr^2} in the integrands. I applied array manipulation and syntactic methods to specialized symbolic calculations in the 1960s. Then, in the early 1990s, I began to reconstruct all the formulas that I had developed for work on molecular integrals that had not been superceded. In this project, I aimed:

1. to type as few mathematical identities as practical;
2. to validate these by tabulating special cases that were easy to check;
3. never to type a formula that could be obtained from formulas already in the files;

4. to rederive results that were in the literature and to check for agreement, before going on to results that could not be derived by hand.

This had several benefits:

1. it reduced the incidence of errors — I continued to make mistakes that checking detected, but these were less frequent than when I typed elementary formulas from memory and constructed 'obvious' transpositions in my head;
2. it brought a variety of software needs to light;
3. it brought mathematical details to light that had been glossed over previously, in some cases causing errors;
4. it identified elementary processes involved in mathematical reasoning.

I started to work this way as a matter of convenience, without a systematic methodology in mind. But as time progressed, I found that I had fallen into a programming style that used certain constructions and clichés repeatedly, and I started to write short scripts to encapsulate these. After a couple of years, I organized these scripts in a MATHEMATICA package called `bilo` and reported this (Barnett, 1993; Barnett and Perry, 1994a). I have supplemented the package with further files for successive projects since then. These can be downloaded from my website `www.princeton.edu/~allengrp/ms`. I provided a typesetting interface and called the entire body of material MATHSCAPE (Barnett, 1998a; Barnett and Perry, 1994b). The typesetting is entirely subordinate, however, to the mathematical manipulation that MATHSCAPE supports.

3. Mathscape principles

MATHSCAPE scripts are strongly mnemonic. The following general principles, conventions and notations provide the framework for an open ended collection of representations and functions, and give the basic information needed to read and to modify MATHSCAPE scripts.

P1. Mathematica platform: MATHSCAPE scripts are written in the MATHEMATICA language and run in MATHEMATICA sessions. MATHSCAPE makes particular use of certain features of MATHEMATICA and it facilitates the use of the full repertoire of MATHEMATICA functions.

P2. Postfix notation: MATHSCAPE scripts make heavy use of the MATHEMATICA postfix notation $x//f$ for the function f of argument x, that is usually typeset as $f(x)$.

P3. Composition: The **pipe operation**: x // pipe[$a_1, a_2, \ldots, a_n$] constructs x // $\bar{a}_1$ // $\bar{a}_2$ $\ldots$// $\bar{a}_n$ where s // $\bar{a}$ is

1. s // a, when a is a unary function that is not of the form b^ℓ;
2. s/.u->v, when a is the replacement u->v;
3. s // $\bar{b}$ // $\bar{b}^{\ell-1}$, when a has the form b^ℓ and ℓ is an explicit integer greater than 1;
4. s // $\bar{b}$ when a has the form $\bar{b}^1$;
5. FixedPoint[b, a] (see Wolfram (2003)), when a has the form $b^{\texttt{infinity}}$;
6. s // pipe[$\alpha_1, \ldots, \alpha_m$], when a is the list {$\alpha_1, \ldots, \alpha_m$}.

P4. Unary wrappers: MATHSCAPE wraps many binary MATHEMATICA functions in Curry style, *e.g.* `Collect[s,u]` as `collect[u][s]` and `Take[s,{m,n}]` as `take[{m,n}][s]`.

P5. Targeting: This focuses action on a piece (or n-tuple of pieces) of an expression. The name of a targeting function begins with the word `to`. Many of the targeting functions wrap the built-in MATHEMATICA `MapAt` function, but they are much easier to read. Also, the repertoire is very easy to extend. The names of the main kinds of targeting function are as follows:

1. `toTheLhs`, `toTheRhs`, `toTheIntegrand`, `toTheSummand`, `toTheDerivative`, `toTheIntegral`, `toTheSum`, `toTheCos`, `toTheExp`, `toBothSides` and other functions that refer to a piece of an expression, or a pair of pieces, by a coalesced phrase containing a recognizable name. A name that begins `toThe` implies that the target is unique. The first occurrence is selected if the target is not unique. The full form of a targeting expression is: s `//` `to···[`$a_1, a_2, \ldots, a_n$`]`. This converts each targeted piece t in s to t `//` `pipe[`$a_1, a_2, \ldots, a_n$`]` *in situ*.

2. `toThe[`v`]`. This targets the first or only subexpression which matches v or v`[...]`. v may be a pattern. The function is extremely versatile.

3. `toEachElement`, `toEachFactor`, `toEachTerm` and other functions that refer to all the pieces with a particular role. Each `toThe`... function is paralleled by a `toEach`... function. The initial effects are independent, *e.g.* `a+b//toEachTerm[f]` gives `f[a]+f[b]`, but these may produce subexpressions that MATHEMATICA combines automatically. The pieces are modified consecutively from right to left. This is significant, *e.g.* in line `(* 42 *)` of `op[Jnaf]` in Section 6.1.

4. `toTheArgumentOfThe[`u`]`, `toTheCoefficientOfThe[`u`]` and other functions that target by context.

5. `toElements[{`*cursors*`}]`, `toFactor[`*predicate*`]`, and other functions with names of the form `to···[`*qualifier*`]`, where the qualifier is either a cursor list or a predicate or both, and illustrative predicates include `containing[`u`]`, `innermost` and *ad hoc* truth functions of the targeted expression, expressed in `Function` notation.

6. `collectivelyToTerms[`*cursorList*`]` and corresponding functions for factors and list elements. These combine a set of subexpressions. For example: `(a+b+c+d) // collectivelyToTerms[{1,2,3}][f]` gives `f[a,b,c]+d`.

P6. Eqn objects: Equations and identities comprise the bulk of mathematical discourse throughout the natural sciences. For well over 100 years, equations have been labelled for reference in such discourse. Accordingly, the assignment `eqn[`*id*`]=(`u`==`v`)` in a MATHSCAPE session makes *id* refer to the equation u`==`v in certain contexts that are described next.

P7. Implicit rule formation — literal: The objects with the heads that begin `brule` are replacement rules that are evaluated automatically as follows:

1. `bruleFor[`u`==`v`]` has the value u`->`v.
2. `bruleFor[{`u_1`==`v_1, u_2`==`$v_2, \ldots$`}]` has the value {u_1`->`v_1, u_2`->`$v_2, \ldots$}.
3. `brule[`*id*`]` has the value of `bruleFor[eqn[`*id*`]]`.
4. `bruleReverseFor[`u`==`v`]` has the value v`->`u.
5. `bruleReverseFor[{`u_1`==`v_1, u_2`==`$v_2, \ldots$`}]` has the value {v_1`->`u_1, v_2`->`$u_2, \ldots$}.
6. `bruleReverse[`*id*`]` has the value of `bruleReverseFor[eqn[`*id*`]]`.

P8. Implicit rule formation — generic: The objects with the heads that begin `grule` are **pattern** replacement rules that are evaluated automatically as follows:

1. `gruleFor[`u`==`v`]` has the value $\hat{u}$`:>`v, where the pattern symbol _ is attached to each variable in u to form $\hat{u}$.
2. `grule[`*id*`]` has the value of `gruleFor[eqn[`*id*`]]`.
3. `gruleReverseFor[`*id*`]`has the value $\hat{v}$`:>`u.
4. `gruleReverse[`*id*`]` has the value of `gruleReverseFor[eqn[`*id*`]]`.
5. the `grule` object of a list is the list of `grule` objects of the members.

For consistency of notation, a statement `grule[`*id*`]=`$\hat{u}$`:>`v is used at times to introduce a pattern replacement directly, in the absence of a supporting `eqn[`*id*`]`, *e.g.* when $\hat{u}$ is conditionalized. The pattern type is changed by a function with usage illustrated by `changeRule[x_][x_.]` that changes `x_` to `x_.` (which connotes optional presence).

P9. Latentized action: MATHSCAPE wraps many MATHEMATICA keywords in names that begin in lower case or end in `$` to suppress the properties that MATHEMATICA provides automatically. For example, `sqrt`, `exp`, `sin`, `pi`, `infinity` and `D$` wrap `Sqrt`, `Exp`, `Sin`, `Pi`, `Infinity` and `D`, respectively. `useElementary`, `useBuiltinIntegrate` and other functions with similar mnemonic names replace the MATHSCAPE names by the MATHEMATICA names that provide the built-in properties when needed. `disuseElementary` and related functions perform the inverse operations.

P10. Range qualified operators: $\sum_{i=j}^{k}(s_i)$ and $\int_{x=a}^{b} s(x)\,\mathrm{d}x$ are written `sum[i,j,k][s[i]]` and `integral[x,a,b][s]`. Multiple sums and integrals are usually nested, as in `integral[x,a,b][integral[y,c,d][s]]`. Similar notations are used for path and other integrals, and for indexed products, arrays and other aggregates.

P11. Reindexing: `reindex` changes the index of sums and other aggregates.

P12. Range splitting: `splitTheRange` splits sums, other aggregates and integrals at a specified point in the range of indexing or integration.

P13. Range joining: `joinTheRanges` has the converse effect.

P14. Range extension and expansion: `leftExpand` and `rightExpand` split the low and high order elements out of a sum or other indexed aggregate; `leftExtend` and `rightExtend` decrease and increase the range of the index; all with appropriate compensatory action. Also, `fullExpand` converts a `sum` or a `prod` to an explicit `Plus` or `Times` when the range is an explicit integer.

P15. Distribution of aggregates: `expandAndDistribute` expands a summand or integrand and distributes the operation across all the terms that result.

P16. Operand inclusion and exclusion: `moveCoefficientRight` moves the multiplier of a sum (integral) into the summand (integrand).
`moveCoefficientLeft` moves the factors in a summand (integrand) that are independent of the summation index (variable of integration) to precede the sum (integral).

P17. Elementary algebraic operations: `factor` removes factors that are common to the factor lists of successive terms in a `Plus`. `factorOut[`c`]` converts $(x_1 + \cdots + x_n)$ to $c * (x_1/c + \cdots + x_n/c)$ when $c \neq -1$. `factorOut[-1]` factors out `Hold[-1]`. This is needed because MATHEMATICA automatically restructures *e.g.* $-(n + 1)$ to $(-n - 1)$. `distribution` moves selected factors of a `Times` onto selected terms of another factor that is a `Plus`. Several `collect`... functions collect on powers, heads and arguments. Other functions perform elementary transpositions. `solveLinear[`w`][`u`==`v`]` solves the equation u==v that is linear in the subexpression w. Several built-in MATHEMATICA functions for elementary algebraic processes embody powerful algorithms, but these do not address quite a few practical needs that are relatively mundane.

P18. Compensatory operations: these include

```
a // multiplyAndDivideBy[b] -> a b/HoldForm[b],
a // divideAndMultiplyBy[b] -> a HoldForm[b]/b,
a // addAndSubtract[b] -> a+b-HoldForm[b],
a // subtractAndAdd[b] a-b+HoldForm[b].
```

P19. Integration: several functions (i) support integration by parts and by change of variable that are specified in convenient ways, and (ii) change the order of integration in multiple integrals where inner limits depend on outer variables.

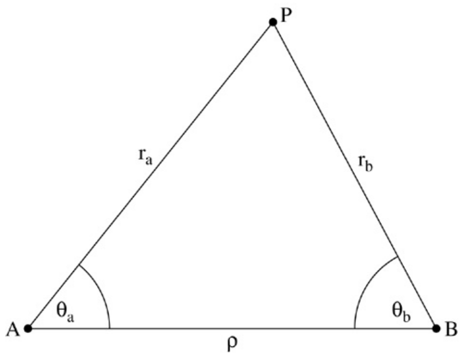

Fig. 1.

P20. Commutation: several functions and `grules` commute the linear operators of summation, integration, differentiation and limit formation.

P21. Assumptions and conditionality: these are supported by simple notations and functions that propagate assumptions into a conditional expression and resolve the conditions accordingly. Often, I write a predicate as `is[`*property*`][`*object*`]`, *e.g.* `is[even][n]` and provide `grules` to compute the truths of these expressions, *e.g.* `is[odd][n+m+2q]` from the truths of `is[odd][n]`, `is[odd][m]` and `is[integer][q]`.

P22. Local names: to make scripts more readable, local names are introduced trivially as wrappers for the functions described above. For example, the function defined by `toThePlusContaining[z_] := to[Plus][containing[z]]` is used in the script following (40).

P23. Unconventional objects: on several occasions, an analysis of the reasoning applied to a problem has been formalized in some syntactically well formed expressions that do not correspond to conventional mathematical notations. A simple tactic to capture and modify `pipe` expressions is illustrated at the end of Section 5. Whilst this is the essence of *n*-th order logic, I have not seen it formalized in work on special functions. A way to convert a very long equation to an abbreviated form that contains names in place of lengthy subexpressions, with accompanying equations that associate the names with these, is given at the end of Section 6.2. `pipe` representations of flow diagrams that support mechanized optimization are discussed in Barnett (2003b). These are mentioned, in large part, to prompt readers to develop further novel objects to facilitate mechanized mathematical reasoning.

4. The *J* integrals

The overlap integrals are linear combinations of certain 'J' integrals that involve the geometrical parameters shown in Fig. 1. These relate to a system of three points A, B and P. I refer to these, respectively, as two fixed atomic nuclei and an electron that moves through space. This does not lose mathematical generality.

The internuclear distance AB is denoted by ρ and the distances of the electron from the nuclei by r_a and r_b. The angles subtended at A and B by BP and AP are denoted by θ_a and θ_b. The out-of-plane coordinate is the azimuth ϕ, measured counterclockwise around the polar axis AB from the plane of the paper. The general form of the J integral is:

$$J_{n_a,m_a,\ell_a,n_b,m_b,\ell_b}(\alpha,\beta,\rho) = \int_{\mathbb{R}^3} \mathrm{e}^{-\alpha r_a}\mathrm{e}^{-\beta r_b} r_a^{n_a} r_b^{n_b} \cos^{m_a}\theta_a \cos^{m_b}\theta_b \sin^{\ell_a}\theta_a \sin^{\ell_b}\theta_b \,\mathrm{d}V, \tag{1}$$

where:

$$n_b \geq \ell_b + m_b - 1, \quad n_a > \ell_a + m_a - 1, \quad \ell_a + \ell_b \text{ is even.} \tag{2}$$

The arguments (α, β, ρ) are implied, when omitted. The symbol $\mathrm{d}V$ denotes the space element of the Euclidean space $\mathbb{R}^3$ of the electron. The closed formulas consist of exponentials and polynomials when $n_a \geq \ell_a + m_a$ and, additionally, exponential integrals and logarithms when $n_a = \ell_a + m_a - 1$. The first case provides enough examples for present purposes. Formulas for about 50 Js with $n_a + n_b \leq 3$ were constructed manually by Charles Coulson in the early 1940s (Coulson, 1942). He was my thesis advisor a few years later, and I think that he originated the idea of using ellipsoidal coordinates for these integrals. In Section 6.3, I derive a general formula that covers the Js with $n_a \geq \ell_a + m_a$. This has not been done before. It would have been impossible (and unusable) a few years ago. My work with MATHEMATICA has shown that the Js in Coulson (1942) contain errors and that the numerical values of overlap integrals which were computed electronically by several authors contain errors, too. The most serious is the value that rounds to -5×10^{-8} in Öztekin et al. (2001). My rounded value for this integral is -8×10^{-78} (Barnett, 2002). I computed this by a scheme that involves a hierarchy of closed formulas, using unrestricted precision arithmetic. This particular integral is a linear combination of Js with $n_a = n_b = 75, \ell_a = 0, 2, \ldots, 30, \ell_b = 0, 2, \ldots, 20$. The results in the present paper agree completely with my results in Barnett (2002). The new formulas in Section 6 avoid the recursions that earlier methods require.

The ellipsoidal coordinates λ and μ are defined by:

$$\lambda = \frac{r_a + r_b}{\rho}, \tag{3}$$

$$\mu = \frac{r_a - r_b}{\rho}. \tag{4}$$

From elementary considerations,

$$r_a = \frac{\rho}{2}(\lambda + \mu), \tag{5}$$

$$r_b = \frac{\rho}{2}(\lambda - \mu), \tag{6}$$

$$r_a \cos\theta_a = \frac{\rho}{2}(1 + \lambda\mu), \tag{7}$$

$$r_b \cos\theta_b = \frac{\rho}{2}(1 - \lambda\mu), \tag{8}$$

$$r_a \sin\theta_a = \frac{\rho}{2}\sqrt{(\lambda^2 - 1)(1 - \mu^2)}, \tag{9}$$

$$r_b \sin\theta_b = \frac{\rho}{2}\sqrt{(\lambda^2 - 1)(1 - \mu^2)}. \tag{10}$$

$\mathbb{R}^3$ is covered by $\{1 \leq \lambda < \infty, -1 \leq \mu \leq 1, 0 \leq \phi \leq 2\pi\}$. Hence the Jacobian with the non-negative value:

$$\frac{\partial(x, y, z)}{\partial(\lambda, \mu, \phi)} = \frac{\rho^3}{8}(\lambda^2 - \mu^2) \tag{11}$$

and the space integral:

$$\int_{\mathbb{R}^3} s\,\mathrm{d}V = \frac{\rho^3}{8}\int_{\lambda=1}^{\infty}\int_{\mu=-1}^{1}\int_{\phi=0}^{2\pi} s \times (\lambda^2-\mu^2)\,\mathrm{d}\lambda\,\mathrm{d}\mu\,\mathrm{d}\phi. \tag{12}$$

Then any J integral can be evaluated symbolically by Algorithm I below. To illustrate its operation, consider the simplest example, $J_{-1,0,0,-1,0,0}$:

$$J_{-1,0,0,-1,0,0} = \int_{\mathbb{R}^3} \frac{\exp[-\alpha r_a]\exp[-\beta r_b]}{r_a r_b}\,\mathrm{d}V. \tag{13}$$

Replace r_a, r_b and $\mathrm{d}V$ using (5), (6), (12). This converts the integral to:

$$\frac{\rho^3}{8}\int_{\lambda=1}^{\infty}\int_{\mu=-1}^{1}\int_{\phi=0}^{2\pi} \frac{\left[\exp\left(\dfrac{-\alpha\rho(\lambda+\mu)}{2}\right)\times\exp\left(\dfrac{-\beta\rho(\lambda-\mu)}{2}\right)\right]}{\dfrac{\rho(\lambda+\mu)}{2}\times\dfrac{\rho(\lambda-\mu)}{2}} \times (\lambda^2-\mu^2)\,\mathrm{d}\lambda\,\mathrm{d}\mu\,\mathrm{d}\phi. \tag{14}$$

Simplifying the integrand converts this to:

$$\frac{\rho}{2}\int_{\lambda=1}^{\infty}\int_{\mu=-1}^{1}\int_{\phi=0}^{2\pi}\exp\left[-\left(\frac{\rho(\alpha+\beta)}{2}\lambda+\frac{\rho(\alpha-\beta)}{2}\mu\right)\right]\mathrm{d}\lambda\,\mathrm{d}\mu\,\mathrm{d}\phi. \tag{15}$$

Elementary methods then give:

$$\frac{4\pi}{\alpha^2-\beta^2}\left[\mathrm{e}^{-\beta\rho}-\mathrm{e}^{-\alpha\rho}\right] \quad \text{when } \alpha\neq\beta,\ \text{and } 2\pi\rho\mathrm{e}^{-\beta\rho} \text{ when } \alpha=\beta. \tag{16}$$

When any of the indexes m_a, ℓ_a, m_b, ℓ_b is nonzero, $\cos\theta_a$, $\sin\theta_a$, $\cos\theta_b$ and $\sin\theta_b$ are replaced using (7)–(10). All cases are covered, accordingly, by

Algorithm I:

1. write the integral in the form (12),
2. in the innermost integrand:
   (a) use (7)–(10) to replace each trigonometric function by an expression containing λ and μ;
   (b) use (5) and (6) to replace residual occurrences of r_a, r_b;
   (c) expand the resulting product of polynomials of λ and μ;
   (d) coalesce the product of exponentials using the identity $\mathrm{e}^u \times \mathrm{e}^v = \mathrm{e}^{u+v}$;
   (e) expand the integrand into terms of form:
   $$\lambda^p\mu^q\exp\left[-\frac{\rho\{\alpha(\lambda+\mu)+\beta(\lambda-\mu)\}}{2}\right]; \tag{17}$$
   (f) expand the argument of the exponential, collect coefficients of λ and μ, and factor the exponential into separate functions of λ and μ;
3. distribute the integration over the terms:
   $$\lambda^p\mu^q\exp\left[-\frac{\rho(\alpha+\beta)}{2}\lambda\right]\exp\left[-\frac{\rho(\alpha-\beta)}{2}\mu\right]; \tag{18}$$
4. integrate over ϕ in each of the resulting terms, using $\int_0^{2\pi}\mathrm{d}\phi = 2\pi$;
5. integrate over μ using a formula for $\int_{-1}^{1} x^p\exp(-kx)\mathrm{d}x$;
6. integrate over λ using a formula for $\int_1^{\infty} x^p\exp(-kx)\mathrm{d}x$.

The built-in `Integrate` function of MATHEMATICA integrates the result of step 2(b) in about 40 s for $J_{-1,0,0,-1,0,0}$ and in about 3 min for $J_{3,0,0,3,0,0}$ on a Sun Solaris (UltraSPARC). These times make the direct application of the algorithm to individual integrals prohibitively slow for high indexes. A faster algorithm factors each triple integral of a term that has the form (18) into three independent integrals over λ, μ and ϕ, respectively, and uses:

$$\int_{x=1}^{\infty} x^n \mathrm{e}^{-kx} \mathrm{d}x = \frac{\mathrm{e}^{-k}}{k^{n+1}} \sum_{\ell=0}^{n} \left[\frac{n! k^\ell}{\ell!} \right] \tag{19}$$

$$\int_{x=-1}^{1} x^n \mathrm{e}^{-kx} \mathrm{d}x = \frac{1}{k^{n+1}} \sum_{\ell=0}^{n} \left[\frac{n! k^\ell \left((-1)^k \mathrm{e}^k - \mathrm{e}^{-k} \right)}{\ell!} \right] \tag{20}$$

$$\int_{0}^{2\pi} \mathrm{d}x = 2\pi. \tag{21}$$

Algorithm II:

1. same as step 1 in Algorithm I;
2. same as step 2 in Algorithm I;
3. same as step 3 in Algorithm I;
4. factor the integral of each term of the form (18) into three one-dimensional integrals;
5. replace these one-dimensional integrals using (19)–(21).

Algorithms I and II can be applied to any specific J integral with explicit integer subscripts that meet the constraints (2). Step 2(c) of Algorithm I, 'expand the product of polynomials of λ and μ', assumes the ability to expand binomial expressions such as $(\lambda + \mu)^n$ for **explicit** positive integer n. Now assume the ability to use the binomial theorem for **symbolic** positive integer n. This allows the following extension:

Algorithm III:

1. same as step 1 in Algorithm I;
2. in the innermost integrand:
   (a) same as step 2(a) in Algorithm I;
   (b) same as step 2(b) in Algorithm I;
   (c) expand the resulting product of powers of polynomials of λ and μ using the **symbolic** binomial theorem for positive integer n;
   (d) coalesce the product of exponentials;
   (e) expand the integrand into terms of the form (17) by converting a 6-fold product of sums into a 6-deep nest of sums
   $(\sum[\cdots]) \times (\sum[\cdots]) \times \cdots \times (\sum[\cdots]) \longrightarrow \sum[\sum[\cdots \sum[\cdots]]]]]]$;
   (f) expand the argument of the exponential;
   (g) collect the coefficients of λ and μ in the exponential
3. commute the triple integral with the nest of sum
   $\int\int\int \sum[\sum[\cdots \sum[\cdots]]]]]] \, \mathrm{d}\lambda \mathrm{d}\mu \mathrm{d}\phi \longrightarrow \sum[\sum[\cdots \sum[\int\int\int \cdots \, \mathrm{d}\lambda \mathrm{d}\mu \mathrm{d}\phi]]]]]]$;
4. distribute the integration over the terms of form (18);
5. factor each 3-dimensional integral into separate one- dimensional integrals;
6. replace these using (19)– (21) given in Algorithm II.

The application of Algorithm III to the definition (1) gives a a **general** formula for the Js that satisfy the constraints (2). Hence

Algorithm IV:

1. set some or all of the indexes that characterize a J integral to explicit values;

2. apply the general formula from Algorithm III;
3. simplify the result.

Hence closed formulas for Js with some or all of the indexes set to explicit integers.

5. An inductive proof

The integrals over λ and μ are very close to the incomplete gamma function that is discussed in standard texts on mathematics for physics and engineering, and in Abramowitz and Stegun (1964), Gradshteyn and Ryzhik (1979) and other reference works. When $k = 1$, the integral (19) is called $\Gamma(n+1, a)$. The closed formula is given in Gradshteyn and Ryzhik (1979) as entry 8.352(2). This leads trivially to the integral containing arbitrary k. The inductive proof of this result introduces several key elements of MATHSCAPE. Also, it is a prototype of scores of inductive proofs that I have coded over the past 15 years.

I write the generalization of (19) by:

$$\int_{x=a}^{\infty} x^n \mathrm{e}^{-kx} \mathrm{d}x = \frac{\mathrm{e}^{-ak}}{k^{n+1}} \sum_{\ell=0}^{n} \left[\frac{n!(ak)^\ell}{\ell!} \right] \tag{22}$$

and represent this by:

```
eqn[incompleteGamma, closed] =
 integral[x, a, infinity][x^n exp[-k x]] ==
  exp[-a k]/k^(n+1) sum[ell, 0, n][n! (a k)^ell/ell!]
```

A referee suggested that I show the MATHEMATICA and MAPLE reductions of these. MATHEMATICA gives:

$$\frac{\frac{\Gamma(n+1)}{k^n} + \frac{a^n(\Gamma(n+1,ka) - n\Gamma(n))}{(ka)^n}}{k} \quad \text{if } \mathrm{Re}(k) > 0. \tag{23}$$

I can force this by asserting $\mathrm{Re}(k) > 0$, and simplify using $n\Gamma(n) = \Gamma(n+1)$ when $a = 1$, but this simply tells me that I am dealing with the incomplete Γ function, which I knew already. MAPLE GIVES

$$-\frac{\pi \csc(n\pi)}{k^{n+1} + \Gamma(-n)} - \frac{k^{\frac{1}{2n}}}{n+1} \exp\left(-\frac{1}{2k}\right) M_{\frac{1}{2n}, \frac{1}{2n+1}+\frac{1}{2}}(k) \tag{24}$$

where M is the Whittaker M function, which does not help, either.

The basis of the induction is established by:

```
integrateAssumptions = Re[k] > 0

check[basis, gammaIncomplete] =
 eqn[incompleteGamma, closed] //
   pipe[
(* 1 *) toBothSides[n -> 0],
(* 2 *) toTheLhs[useIntegrate],
(* 3 *) toTheRhs[fullExpand]]
```

The successive lines of this `pipe` expression act on the representation of (22) as follows:

1 specializes the integral to the case $n = 0$.

2 evaluates the right hand side by converting it to

```
Integrate[E^(-k x),{x,a,Infinity},Assumptions->Re[k]>0]
```

which MATHEMATICA reduces automatically to e^{-ak}/k.

3 evaluates the right hand side by converting the sum to the explicit value 1.

These actions convert both sides of the Equal expression to exp[-k a]/k, and the statement collapses to True.

The induction is extended by the following statement.

```
check[gammaIncomplete] = eqn[incompleteGamma, closed] // pipe[
(*  1 *) toTheRhs[moveCoefficientRight],
(*  2 *) toBothSides[D$[k]],
(*  3 *) toTheLhs[
(*  4 *)  grule[commuteD$integral],
(*  5 *)  useElementary, useD$, disuseElementary ,
(*  6 *)  toTheIntegral[moveCoefficientLeft]],
(*  7 *) toTheRhs[
(*  8 *)  grule[commuteD$sum],
(*  9 *)  PowerExpand,
(* 10 *)  useElementary, useD$, disuseElementary,
(* 11 *)  toTheSum[Distribute],
(* 12 *)  toSum[containing[k^(ell-n-1)]][reindex[ell, ell+1]],
(* 13 *)  toSum[containing[ell!]][leftExpand],
(* 14 *)  toSum[containing[ell!]][rightExtend],
(* 15 *)  joinTheSummands,
(* 16 *)  toEach[n!][grule[factorialUp]], Simplify,
(* 17 *)  toEach[(ell-1)!][grule[factorialUp]], Simplify,
(* 18 *)  toTheSum[leftExtend]
(* 19 *)   ],
(* 20 *) toTheLhs[
(* 21 *)  grule[incompleteGamma, closed],
(* 22 *)  moveCoefficientRight,
(* 23 *)  PowerExpand]]
```

The successive lines of this pipe expression act on the representation of (22) as follows:

1 changes the right hand side to

$$\sum_{\ell=0}^{n} \frac{e^{-ak}}{k^{n+1}} \left[\frac{n!(ak)^{\ell}}{\ell!}\right]. \tag{25}$$

2 prepends the operator $\partial/\partial k$ represented by D$[k] to both sides of the equation but **does not** perform the differentiation.

3 focuses on the left hand side.

4 commutes $\partial/\partial k \int_x (\cdots)\,dx$ to $\int_x \partial(\cdots)/\partial k\,dx$.

5 uses MATHEMATICA differentiation on the left hand side, converting it to

$$\int_{x=a}^{\infty} \left[-x^{n+1}e^{-kx}\right] dx. \tag{26}$$

6 moves the constant, *i.e.* -1, out of the integral. Hence

$$-\int_{x=a}^{\infty} \left[x^{n+1}e^{-kx}\right] dx. \tag{27}$$

7 focuses attention on the right hand side.

8–11 convert the right hand side, in succession, to

$$\sum_{\ell=0}^{n} \frac{\partial}{\partial k}\left[\frac{n!(ka)^{\ell}\mathrm{e}^{-ka}}{\ell!k^{n+1}}\right], \tag{28}$$

$$\sum_{\ell=0}^{n} \frac{\partial}{\partial k}\left[\frac{n!a^{\ell}\mathrm{e}^{-ka}}{\ell!k^{n-\ell+1}}\right], \tag{29}$$

$$\sum_{\ell=0}^{n}\left[-\frac{n!a^{\ell+1}\mathrm{e}^{-ka}}{\ell!k^{n-\ell+1}} + \frac{(\ell-n-1)n!a^{\ell}\mathrm{e}^{-ka}}{\ell!k^{n-\ell+2}}\right], \tag{30}$$

$$\sum_{\ell=0}^{n}\left[-\frac{n!a^{\ell+1}\mathrm{e}^{-ka}}{\ell!k^{n-\ell+1}}\right] + \sum_{\ell=0}^{n}\left[\frac{(\ell-n-1)n!a^{\ell}\mathrm{e}^{-ka}}{\ell!k^{n-\ell+2}}\right]. \tag{31}$$

12 converts the first term to

$$\sum_{\ell=1}^{n+1}\left[-\frac{n!a^{\ell}\mathrm{e}^{-ka}}{(\ell-1)!k^{n-\ell+2}}\right]. \tag{32}$$

13–14 convert the second term in succession to

$$\frac{(-n-1)n!\mathrm{e}^{-ka}}{k^{n+2}} + \sum_{\ell=1}^{n}\left[-\frac{(\ell-n-1)n!a^{\ell}\mathrm{e}^{-ka}}{\ell!k^{n-\ell+2}}\right], \tag{33}$$

$$\frac{(-n-1)n!\mathrm{e}^{-ka}}{k^{n+2}} + \sum_{\ell=1}^{n+1}\left[-\frac{(\ell-n-1)n!a^{\ell}\mathrm{e}^{-ka}}{\ell!k^{n-\ell+2}}\right]. \tag{34}$$

The term that is subtracted to compensate for the right extension of the sum in the last of these steps is identically zero because of the factor $\ell - n - 1$.

15–19 use `grule[factorialUp] = n_!:>(n+1)!/(n+1)` and several functions described in Section 3 to convert the right hand side in succession to

$$\frac{(-n-1)n!\mathrm{e}^{-ka}}{k^{n+2}} + \sum_{\ell=1}^{n+1}\left[-\frac{n!a^{\ell}\mathrm{e}^{-ka}}{(\ell-1)!k^{n-\ell+2}} + \frac{(\ell-n-1)n!a^{\ell}\mathrm{e}^{-ka}}{\ell!k^{n-\ell+2}}\right], \tag{35}$$

$$-\frac{(n+1)!\mathrm{e}^{-ka}}{k^{n+2}} + \sum_{\ell=1}^{n+1}\frac{-\left((n-\ell+1)(\ell-1)!+\ell!\right)(n+1)!a^{\ell}\mathrm{e}^{-ka}}{(n+1)(\ell-1)!\ell!k^{n-\ell+2}}, \tag{36}$$

$$-\frac{(n+1)!\mathrm{e}^{-ka}}{k^{n+2}} + \sum_{\ell=1}^{n+1}\frac{-(n+1)!a^{\ell}\mathrm{e}^{-ka}}{\ell!k^{n-\ell+2}}, \tag{37}$$

$$\sum_{\ell=0}^{n+1}\frac{-(n+1)!a^{\ell}\mathrm{e}^{-ka}}{\ell!k^{n-\ell+2}}. \tag{38}$$

20 focuses attention on the left hand side.

21–22 convert the left hand side in succession to

$$-\frac{\mathrm{e}^{-ka}}{k^{n+2}}\sum_{\ell=0}^{n+1}\left[-\frac{(n+1)!(ka)^{\ell}}{\ell!}\right], \tag{39}$$

$$\sum_{\ell=0}^{n+1}\left[-\frac{(n+1)!(ka)^{\ell}\mathrm{e}^{-ka}}{\ell!k^{n+2}}\right]. \tag{40}$$

23 converts the left hand side to an expression that is identical to (38) and the system reduces the entire `Equal` expression to `True`.

The formulas for the integrals over λ and μ that are used in the algorithms of Section 4 are formed by:

```
eqn[lambdaIntegral] =
 eqn[incompleteGamma, closed] /. a-> 1

eqn[muIntegral] =
 eqn[incompleteGamma, closed] //
  pipe[
   toBothSides[a -> -1],
   toTheLhs[splitTheRange[1]],
   transposeFromLhsTerm[2],
   toTheRhs[
    grule[incompleteGamma, closed],
    Factor,
    toThePlusContaining[exp][
     toEachTerm[moveCoefficientRight],
     joinTheSummands,
     toTheSummand[
      PowerExpand,
      Factor,
      toThePlus[factorOut[-1]], ReleaseHold]]]]
```

Hence:

```
eqn[lambdaIntegral] =
 integral[x, 1, infinity][x^n*exp[-k x]] ==
  exp[-k]/k^(n+1) sum[ell, 0, n][n! k^ell/ell!]

eqn[muIntegral] =
integral[x, -1, 1][x^n exp[-k x]] ==
 1/k^(n+1) sum[ell, 0, n][n! k^ell ((-1)^ell exp[k] - exp[-k])/ell!]
```

The `pipe` expression that extends the induction contains two `toTheRhs` subexpressions separated by a `toTheLhs` subexpression. Because actions that are targeted on the left hand side cannot affect the right hand side, the separated expressions can be coalesced. The situation is recognized and streamlined mechanically by:

```
oldExtension =
 betwobj["check[gammaIncomplete]", "eqn[incompleteGamma, closed]"]

newExtension = oldExtension /.
 pipe[$1___?(Head[(Not[MatchQ[#, toTheRhs[___]]]&),
          $2_toTheRhs, $3_toTheLhs, $4_toTheRhs, $5___] :>
  pipe[$1, $3, hold[$2, $4], $5] /.
   hold[toTheRhs[$6___], toTheRhs[$7___]] :> toTheRhs[$6, $7]
```

The `betwobj` retrieves the operator from the history of the current run, using `InString/@Range[$Line]`. The pattern transformations are standard MATHEMATICA coding. These statements provide a further illustration of the benefit of having the repertoire of MATHEMATICA resources to invoke.

6. Constructing the general formulas

6.1. Constructing a prototype

To construct a closed formula for (1), I omit sines and cosines of θ_a and θ_b from the integrand, use the following MATHSCAPE statements to construct the formula for this special case, and then generalize.

```
op[Jnaf] =
pipe[
(*  1 *)  ma -> 0, mb -> 0, ella -> 0, ellb -> 0,
(*  2  *) toTheRhs[
(*  3  *)  brule /@ {raToElliptic, rbToElliptic},
(*  4  *)  grule[volumeIntegral, elliptic],
(*  5  *)  useElementary, PowerExpand, disuseElementary,
(*  6  *)  toTheArgumentOfTheExp[
(*  7  *)   collect[lambda, mu],
(*  8  *)   to[Plus][innermost][Factor]],
(*  9  *)  gruleReverse[expProd],
(* 10 *) grule[binomialExpansion],
(* 11 *)  toSum[1, containing[i]][i->i1],
(* 12 *)  toSum[1, containing[i]][i->i2],
(* 13 *)  to[_ sum[__][_]][1][moveCoefficientRight],
(* 14 *)  to[_ sum[__][_]][1][moveCoefficientRight],
(* 15 *)  toSums[{1,2}][expandAndDistribute],
(* 16 *)  toIntegrals[{1,2,3}][expandAndDistribute],
(* 17 *)  replaceRepeated[grule[commuteIntegralSum]],
(* 18 *)  toThePlus[
(* 19 *)   toEachTerm[
(* 20 *)   toIntegrals[{1,2,3}][moveCoefficientLeft],
(* 21 *)   toIntegral[containing[mu]][
(* 22 *)    PowerExpand, moveCoefficientLeft],
(* 23 *)   toTheArgumentOfEachExp[
(* 24 *)    divideAndMultiplyBy[-1],
(* 25 *)    collectivelyToFactors[
(* 26 *)    notContainingAny[HoldForm, lambda, mu]][Hold],
(* 27 *)    HoldForm -> Identity
(* 28 *)                 ],
(* 29 *)  grule[lambdaIntegral],
(* 30 *) grule[muIntegral],
(* 31 *)  toIntegral[containing[phi]][useIntegrate],
(* 32 *)  Hold -> Identity
(* 33 *)    ],
(* 34 *) replaceRepeated[grule[joinTheSummands]]],
(* 35 *) toSum[innermost][expandAndDistribute],
(* 36 *) toSummand[2][Expand],
(* 37 *) toSum[innermost][moveCoefficientLeft],
(* 38 *) toSummand[2][
(* 39 *)  grule[expProd],
(* 40 *)  toEachExp[Simplify],
(* 41 *)  collectByArguments[exp]],
(* 42 *) toSum[{1, 2}][Distribute],
(* 43 *) toEachSummand[Factor],
(* 44 *) toEach[sum[i2,__][_]][moveCoefficientLeft],
(* 45 *) toEach[sum[i1,__][_]][moveCoefficientLeft],
(* 46 *) Factor]]

eqn[Jnaf] = eqn[Jdefn] // op[Jnaf]
```

Hence the result of the form:

$$J_{n_a,0,0,n_b,0,0} = \frac{4n_a!n_b!\pi\rho^{n_a+n_b-1}}{(\alpha-\beta)^3(\alpha+\beta)^3} \times \left[Z^{(\alpha)}\mathrm{e}^{-\alpha\rho} + Z^{(\beta)}\mathrm{e}^{-\beta\rho}\right] \tag{41}$$

where

$$\begin{aligned} Z^{(\alpha)} &= -\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\Big[(-1)^{n_b-i_2}(Z_1^{(\alpha)}+Z_2^{(\alpha)}+\cdots+Z_6^{(\alpha)}) \\ &\quad \div \Big(i_1!(n_a-i_1)!i_2!(n_b-i_2)!\,((\alpha-\beta)\rho)^{n_a+n_b-i_1-i_2}\,((\alpha+\beta)\rho)^{i_1+i_2}\Big)\Big] \qquad (42) \\ Z_1^{(\alpha)} &= (2+i_1+i_2)!(n_a+n_b-i_1-i_2)!\alpha^2 \\ &\quad \times\left(\sum_{\ell=0}^{i_1+i_2+2}\frac{((\alpha+\beta)\rho)^\ell}{2^\ell \ell!}\right)\times\left(\sum_{\ell=0}^{n_a+n_b-i_1-i_2}\frac{((\alpha-\beta)\rho)^\ell}{2^\ell \ell!}\right), \qquad (43) \end{aligned}$$

with similar expressions for $Z_j^{(\alpha)}$, $j = 2, \ldots, 6$, and corresponding results for the coefficient of $\mathrm{e}^{-\beta\rho}$. The successive lines of the pipe expression act as follows (bold face symbols are surrogates):

1 specializes (1) to

$$J_{n_a,0,0,n_b,0,0} = \int_{\mathbb{R}^3} r_a^{n_a} r_b^{n_b} \mathrm{e}^{-\alpha r_a}\mathrm{e}^{-\beta r_b}\,\mathrm{d}V. \tag{44}$$

2 focuses on the right hand side.

3–4 convert this in turn to expressions of forms (45) and (46), where $\mathbf{p}(\lambda, \mu)$ stands for a polynomial in (λ, μ) and $\mathbf{a}$ stands for $\rho^3/8$.

$$\int_{\mathbf{R}^3}\mathbf{p}(\lambda,\mu)\mathrm{e}^{-\beta(\lambda-\mu)\rho/2}\mathrm{e}^{-\alpha(\lambda+\mu)\rho/2}\,\mathrm{d}V, \tag{45}$$

$$\mathbf{a}\int_\lambda\int_\mu\int_\phi(\lambda^2-\mu^2)\mathbf{p}(\lambda,\mu)\mathrm{e}^{-\beta(\lambda-\mu)\rho/2}\mathrm{e}^{-\alpha(\lambda+\mu)\rho/2}\,\mathrm{d}\lambda\,\mathrm{d}\mu\,\mathrm{d}\phi. \tag{46}$$

5, 6–8 and then 9 convert the product of exponentials in succession to the following forms.

$$\mathrm{e}^{-\beta(\lambda-\mu)\rho/2-\alpha(\lambda+\mu)\rho/2},\ \mathrm{e}^{-(\alpha+\beta)\rho\lambda/2-(\alpha-\beta)\rho\mu/2},\ \mathrm{e}^{-(\alpha+\beta)\rho\lambda/2}\mathrm{e}^{-(\alpha-\beta)\rho\mu/2}. \tag{47}$$

10–12 convert the right hand side to the following form, where $\mathbf{b}$ stands for $\rho^{n_a+n_b}/2^{n_a+n_b}$, the exponential factors are unchanged, and $\mathbf{s}_{i_1}(\lambda)$, $\mathbf{t}_{i_2}(\mu)$ stand for the terms in the binomial expansions of the (λ, μ) expressions for $r_a^{n_a}$ and $r_b^{n_b}$.

$$\mathbf{a}\int_\lambda\int_\mu\int_\phi \mathbf{b}\times(\lambda^2-\mu^2)\times \mathrm{e}^{(\dots)}\mathrm{e}^{(\dots)}\left(\sum_{i_1=0}^{n_a}\mathbf{s}_{i_1}(\lambda)\right)\left(\sum_{i_2=0}^{n_b}\mathbf{t}_{i_2}(\mu)\right)\mathrm{d}\lambda\,\mathrm{d}\mu\,\mathrm{d}\phi. \tag{48}$$

13–15 rearrange the integrand to the forms (49)–(51) in succession. The product $\mathbf{b}\,\mathbf{s}_{i_1}(\lambda)$ is restructured automatically to the form $\mathbf{b}'\,\mathbf{s}'_{i_1}(\lambda)$, where every factor of $\mathbf{s}_{i_1}(\lambda)$ depends on λ and $\mathbf{b}'$ is free of λ. Then the product $\mathbf{b}'\,\mathbf{t}_{i_2}(\mu)$ is restructured automatically to the form $\mathbf{b}''\,\mathbf{t}'_{i_2}(\mu)$, where every factor of $\mathbf{t}_{i_2}(\mu)$ depends on μ and $\mathbf{b}''$ is free of λ and μ. The product $\lambda^2\mathbf{s}'_{i_1}(\lambda)$ coalesces to a `Times` that is written $\mathbf{s}''_{i_1}(\lambda)$ here.

$$\left(\sum_{i_1=0}^{n_a}\mathbf{b}'\times(\lambda^2-\mu^2)\mathrm{e}^{(\dots)}\mathrm{e}^{(\dots)}\mathbf{s}'_{i_1}(\lambda)\right)\left(\sum_{i_2=0}^{n_b}\mathbf{t}_{i_2}(\mu)\right), \tag{49}$$

$$\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\left[\mathbf{b}''\times(\lambda^2-\mu^2)\mathrm{e}^{(\dots)}\mathrm{e}^{(\dots)}\mathbf{s}'_{i_1}(\lambda)\mathbf{t}'_{i_2}(\mu)\right], \tag{50}$$

$$\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\left[\mathbf{b}''\times\left(\mathrm{e}^{(\dots)}\mathrm{e}^{(\dots)}\mathbf{s}''_{i_1}(\lambda)\mathbf{t}_{i_2}(\mu)\right)+\dots\right]. \tag{51}$$

16–17 convert the triple integral to expressions of the following forms, in turn, where the ... in each + stand for a second term with $-\mu^2$ in place of λ^2.

$$\int_\lambda\int_\mu\int_\phi\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\mathbf{b}''\mathrm{e}^{(\dots)}\mathrm{e}^{(\dots)}\mathbf{s}''_{i_1}(\lambda)\mathbf{t}'_{i_2}(\mu)\mathrm{d}\lambda\,\mathrm{d}\mu\,\mathrm{d}\phi+\int_\lambda\dots\mathrm{d}\phi, \tag{52}$$

$$\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\int_\lambda\int_\mu\int_\phi\mathbf{b}''\mathrm{e}^{(\dots)}\mathrm{e}^{(\dots)}\mathbf{s}''_{i_1}(\lambda)\mathbf{t}'_{i_2}(\mu)\mathrm{d}\lambda\,\mathrm{d}\mu\,\mathrm{d}\phi+\sum_{i_1=0}^{n_a}\dots\mathrm{d}\phi. \tag{53}$$

18–28 act as follows on the `Plus`. (Targeting the entire right hand side was sufficiently focused for the preceding operations.) In particular,

18 focuses on the `Plus`.

19 makes lines 20–33 act in parallel on both terms.

20 converts the entire right hand side to the form

$$\mathbf{a}\left[\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\mathbf{b}''\times\left(\int_{\lambda=1}^{\infty}\lambda^{i_1+i_2+2}\mathrm{e}^{-(\alpha+\beta)\rho\lambda/2}\mathrm{d}\lambda\right)\right.$$
$$\left.\times\left(\int_{\mu=-1}^{1}(-\mu)^{n_b-i_2}\mu^{n_a-i_1}\mathrm{e}^{-(\alpha-\beta)\rho\lambda/2}\mathrm{d}\mu\right)\times\left(\int_{\phi=0}^{2\pi}\mathrm{d}\phi\right)+\dots\right]. \tag{54}$$

21–22 convert $\int_\mu(-\mu)^{n_b-i_2}\mu^{n_a-i_1}\dots\mathrm{d}\mu$ to $(-1)^{n_b-i_2}\int_\mu\mu^{n_a+n_b-i_1-i_2}\dots\mathrm{d}\mu$.

23–28 convert the exponentials to $\mathrm{e}^{-\mathtt{Hold}[(\alpha+\beta)\rho/2]\lambda}$ and $\mathrm{e}^{-\mathtt{Hold}[(\alpha-\beta)\rho/2]\mu}$ using **P5**(4, 6), **P20** to force the arguments of the exponentials into the form matched by `-(a_)*lambda` and `-(a_)*mu`. Knowledge of the MATHEMATICA representations was needed to write this piece of code, but this need is not a reflection on MATHEMATICA — the exigencies of symbolic calculation require canonical representations of expressions that have to be used in different forms.

29–32 change the right hand side to the form

$$\mathbf{a}\left[\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\mathbf{u}_1+\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\mathbf{u}_2\right], \tag{55}$$

where the factors of $\mathbf{u}_1$ include

1. a sum over ℓ containing $\mathrm{e}^{-(\alpha+\beta)\rho/2}$ as a factor of the summand, and
2. another sum over ℓ containing $\mathrm{e}^{-(\alpha-\beta)\rho/2}+(-1)^\ell\mathrm{e}^{(\alpha-\beta)\rho/2}$ as a factor of its summand.

The factors of $\mathbf{u}_2$ fit this pattern, too.

33 ends the scope of the `toEachTerm` in 19.

34 converts the right hand side to the form

$$\mathbf{a}\sum_{i_1=0}^{n_a}\left[\sum_{i_2=0}^{n_b}\mathbf{u}_1+\sum_{i_2=0}^{n_b}\mathbf{u}_2\right], \tag{56}$$

and then to the form

$$\mathbf{a}\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\mathbf{u}, \tag{57}$$

where $\mathbf{u}=\mathbf{u}_1+\mathbf{u}_2$. The addition shortens the expression by 30% due to mutual annihilation of terms of identical magnitude and opposite sign.

35 converts each sum over ℓ that results from the μ integration to the `Plus` of two sums.

36 expands $\mathbf{u}$ into the following form, where $\mathbf{c}_\ell$ and $\mathbf{d}_\ell$ stand for algebraic expressions in $(n_a, n_b, i_1, i_2, \ell)$.

$$\left[\sum_\ell \mathbf{c}_\ell \mathrm{e}^{-(\alpha+\beta)\rho/2}\right]\times\left[\sum_\ell\left(-\mathbf{d}_\ell\mathrm{e}^{-(\alpha-\beta)\rho/2}\right)\right]+$$
$$\left[\sum_\ell \mathbf{c}_\ell \mathrm{e}^{-(\alpha+\beta)\rho/2}\right]\times\left[\sum_\ell\left((-1)^\ell\mathbf{d}_\ell\mathrm{e}^{(\alpha-\beta)\rho/2}\right)\right]. \tag{58}$$

37 converts the first term to the form

$$\left[\sum_\ell -\mathbf{c}_\ell \mathrm{e}^{-(\alpha+\beta)\rho/2}\mathrm{e}^{-(\alpha-\beta)\rho/2}\right]\times\left[\sum_\ell \mathbf{d}_\ell\right], \tag{59}$$

and converts the second term to the corresponding form with $\mathrm{e}^{(\alpha-\beta)\rho/2}$ as the second exponential factor.

38–41 combine the exponentials, reducing the right hand side to the following form where $\mathbf{v}_1$ and $\mathbf{v}_2$ are polynomial in α, β, ρ.

$$\mathbf{a}\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\left[\mathbf{v}_1\mathrm{e}^{-\beta\rho}-\mathbf{v}_2\mathrm{e}^{-\alpha\rho}\right]. \tag{60}$$

42–46 convert this in succession to

$$\mathbf{a}\sum_{i_1=0}^{n_a}\left[\sum_{i_2=0}^{n_b}\left(\mathbf{v}_1\mathrm{e}^{-\beta\rho}\right)-\sum_{i_2=0}^{n_b}\left(\mathbf{v}_2\mathrm{e}^{-\alpha\rho}\right)\right], \tag{61}$$

$$\mathbf{a}\times\left[\mathbf{w}\mathrm{e}^{-\beta\rho}\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\mathbf{v}_1'-\mathbf{w}\mathrm{e}^{-\alpha\rho}\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\mathbf{v}_2'\right], \tag{62}$$

$$\mathbf{g}\times\left[\mathrm{e}^{-\beta\rho}\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\mathbf{v}_1'-\mathrm{e}^{-\alpha\rho}\sum_{i_1=0}^{n_a}\sum_{i_2=0}^{n_b}\mathbf{v}_2'\right]. \tag{63}$$

where $\mathbf{v}_1'$ and $\mathbf{v}_2'$ are polynomial in α, β, ρ and $\mathbf{g}$ is algebraic in $(n_a, n_b, \alpha, \beta, \rho)$. Note the right to left action of the `Distribute` functions in line 42, in accordance with the targeting principle **P5**(3) in Section 3.

I constructed the pipe expression interactively. At each stage of the development, I inspected the heads and lengths and short forms of pieces of the most recent intermediate result, using some further MATHSCAPE tools, and went on to the next step without bothering to record the details that I had found. Often, the inspection of one step was helped by scrolling back to see how I had analyzed the previous step. The entire process went quite rapidly. Writing the explanation, with surrogates for subexpressions, was far more arduous and time consuming than developing the statement. Several MATHSCAPE functions help the analysis of scripts. For example, the cumulative effects of `op[Jnaf]` through line `(* 42 *)` is shown by

```
eqn[Jdefn] //
 (op[Jnaf] // toThe[toTheRhs][toArguments[Range[-4, -1]][_->Identity]])
```

Note that in this expression, `toTheRhs` is the **head** of a piece of an operator that refers to the right hand side of an operand — the operator itself does not contain a right hand side.

6.2. Exploring the prototype

The byte count of the MATHEMATICA form of the explicit formula for $J_{n_a,0,0,n_b,0,0}$ is 18,968. It is particularized to $J_{0,0,0,0,0,0}$ by

```
eqn[fromJnaf[0,0]] =
 eqn[Jnaf] //
  pipe[
   na->0, nb->0, toEachSum[fullExpand]^2, toTheRhs[Simplify]]
```

Hence:

$$J_{0,0,0,0,0,0}=\frac{8\pi}{(\alpha^2-\beta^2)^3\rho}\left[\beta(4\alpha+\alpha^2\rho-\beta^2\rho)\mathrm{e}^{-\alpha\rho}+\alpha(-4\beta+\alpha^2\rho-\beta^2\rho)\mathrm{e}^{-\beta\rho}\right]. \tag{64}$$

For consistency with the existing literature, define X and transpose to equations for α^2 and β^2 by

```
eqn[J, X] =  X == alpha^2-beta^2;
eqn[alphaSqd] = eqn[J, X] // solveLinear[alpha^2];
eqn[betaSqd] = eqn[J, X] // solveLinear[beta^2];
eqn[J[0,0]] =
 eqn[fromJnaf[0,0]] //
  toTheRhs[
   brule[alphaSqd], Simplify,
   to[Plus][containing[-4], innermost][factorOut[-1]],
ReleaseHold]
```

This reduces the integral to

$$\frac{8\pi}{\rho X^3}\left[\beta(4\alpha+\rho X)\mathrm{e}^{-\alpha\rho}-\alpha(4\beta-\rho X)\mathrm{e}^{-\beta\rho}\right] \tag{65}$$

in agreement with the result in Coulson (1942). This final reduction that enables visual comparison with published formulas prompted the initial development of targeting functions. Several of Coulson's colleagues and students contributed to Coulson (1942), and their styles of simplification differed slightly. To convert the direct results of an algorithm (that, by its nature, is systematic) to the published formulas I had to throw the emphasis onto powers of α in some subexpressions and onto powers of β in other subexpressions without a consistent pattern. Targeting functions grew out of clichés to facilitate the necessary transformations.

Eqs. (41)–(43) are formed *via* the unconventional expression corresponding to (63) with $\left[(Z^{(\alpha)}=\cdots)\times \mathrm{e}^{-\alpha\rho})+(Z^{(\beta)}=\cdots)\times \mathrm{e}^{-\beta\rho})\right]$ in place of the bracketed `Plus`, where the $\cdots$ stand for the two coefficient expressions. The embedded equations are extracted by the MATHSCAPE `instancesOf` function that wraps MATHEMATICA `Cases`. Then the pieces of $Z^{(\alpha)}$ and $Z^{(\beta)}$ are formed *via* similar smaller embedded equations. The actual MATHSCAPE statements that extract $Z^{(\alpha)}$ and its pieces are

```
Thread[
 set[
  eqn[Jnaf, #]& /@ {Za, Zb},
  eqn[Jnaf][[2]] //
   pipe[
    toTheCoefficientOf[exp[-alpha rho]][Za==#&],
    toTheCoefficientOf[exp[-beta rho]][Zb==#&],
    instancesOf[lhs_==rhs_]]]] /. set -> Set;

eqn[Jnaf, toZs] = eqn[Jnaf] /. (bruleReverse[Jnaf, #]& /@ {Za,Zb})

eqn[Jnaf, Zaa] =
 eqn[Jnaf, Za][[2]] //
  pipe[
   toSummand[2][
    toFactor[containing[sum]][
     numberTheTerms,
     term[n_][v_] :> Zaa[n] == v]],
    instancesOf[lhs_ == rhs_]];

eqn[Jnaf, ZaFromZaa] =
 eqn[Jnaf, Za] /. bruleReverse[Jnaf, Zaa]
```

6.3. The generalization

The general formula for any $(n_a, m_a, \ell_a, n_b, m_b, \ell_b)$ consistent with (2) is constructed by

```
eqn[forJ, ells] = ellav == (ella + ellb)/2

op[Jclosed] =
toTheRhs[
(*  3x  *)  brule /@
             {cosAtoElliptic, cosBtoElliptic, sinAtoElliptic,
               sinBtoElliptic, raToElliptic, rbToElliptic},
(*  4    *)  grule[volumeIntegral, elliptic],
(*  5    *)  useElementary, PowerExpand, disuseElementary,
(*  5x  *)  brule[forJ, ells],
(*  6    *)  toTheArgumentOfTheExp[
...
(* 10  *)   grule[binomialExpansion],
(* 11x *)  toSum[1, containing[i]][i->i[#]]& /@ Range[6],
(* 12x *)  to[_ sum[__][_]][1][moveCoefficientRight] ^6,
(* 15x *)  toSums[Range[6] ][expandAndDistribute],
(* 16  *)  toIntegrals[{1,2,3}][expandAndDistribute],
...
(* 35  *) toSum[innermost][expandAndDistribute],
(* 36x *) toSummand[6][Expand],
(* 37  *) toSum[innermost][moveCoefficientLeft],
(* 38x *) toSummand[6][
(* 39  *)  grule[expProd],
(* 40  *)  toEachExp[Simplify],
(* 41  *)  collectByArguments[exp],
(* 42x *) toTheCoefficientOfEach[exp][hold]],
(* 42y *) toSum[Range[6]][Distribute],
(* 42z *) hold -> Identity,
(* 43  *) toEachSummand[Factor],
(* 45x *) toEach[sum[i[#],__][_]][
              moveCoefficientLeft]& /@ Reverse[Range[6]],
(* 46 *)  Factor,
(* 47x *) brule[forJ, ells]];

eqn[Jclosed] = eqn[Jdefn] // op[Jclosed]
```

Lines 4, 5, 6–10, 16–35, 37, 39–41, 43 and 46 are kept unchanged from the prototype. The indexes `i1` and `i2` have been changed to `i[1]` and `i[2]`, to allow indexed names up to `i[6]`for the summation indexes. Line 3x extends line 3 in `op[Jnaf]` to allow for sines and cosines. Lines 5x and 47x allow for the presence of `ellav`. Lines 11x, 12x, 15x, 36x, 38x, 42y and 45x allow for six binomial expansion indexes. Lines 42x and 42z protect the result of collecting the coefficients of the exponentials from the distribution of summation. Line 45x moves the minus sign from the summand over `i[6]` outward, to annihilate a minus sign at the start of the overall expression. This is done to give a final result that collapses to the prototype when $m_a = \ell_a = m_b = \ell_b = 0$. The byte count of the general formula is 62,160. The overall structure of the general formula is similar to (42) with a six-fold instead of two-fold sum and six $Z^{(\alpha)}$ coefficients again.

The general formula for the case $\alpha = \beta$ is produced by a simple adaptation of `op[Jclosed]`.

6.4. Checking

Several identities connect the J integrals. The symmetry of (1) is expressed by

$$J_{n_a,m_a,\ell_a,n_b,m_b,\ell_b}(\alpha, \beta, \rho) = J_{n_b,m_b,\ell_b,n_a,m_a,\ell_a}(\beta, \alpha, \rho). \tag{66}$$

Differentiation with respect to α and β raise the power of r_a and r_b, respectively, in the integrand.

$$\frac{\partial}{\partial\alpha} J_{n_a,m_a,\ell_a,n_b,m_b,\ell_b}(\alpha, \beta, \rho) = -J_{n_a+1,m_a,\ell_a,n_b,m_b,\ell_b}(\alpha, \beta, \rho), \tag{67}$$

$$\frac{\partial}{\partial\beta} J_{n_a,m_a,\ell_a,n_b,m_b,\ell_b}(\alpha, \beta, \rho) = -J_{n_a,m_a,\ell_a,n_b+1,m_b,\ell_b}(\alpha, \beta, \rho). \tag{68}$$

The limit when ρ goes to zero is a one-center integral. The (r_a, θ_a, ϕ) and (r_b, θ_b, ϕ) coordinate systems coalesce. Drop the subscripts. Then the limit factors into elementary integrals that the built-in `Integrate` of MATHEMATICA evaluates.

$$\begin{aligned}&\lim_{\rho->0} J_{n_a,m_a,\ell_a,n_b,m_b,\ell_b}(\alpha,\beta,\rho)\\&\quad=\int_{r=0}^{\infty} r^{n_a+n_b+2}\mathrm{e}^{-(\alpha+\beta)r}\,\mathrm{d}r\int_{\theta=0}^{\pi}\cos^{m_a+m_b}\theta\,\sin^{\ell_a+\ell_b+1}\theta\,\mathrm{d}\theta\int_{\phi=0}^{2\pi}\mathrm{d}\phi.\end{aligned}\tag{69}$$

The identity $\cos^2\theta+\sin^2\theta=1$ gives two more identities between the Js.

$$\begin{aligned}&J_{n_a+2,m_a,\ell_a,n_b,m_b,\ell_b}(\alpha,\beta,\rho)\\&\quad=J_{n_a+2,m_a+2,\ell_a,n_b,m_b,\ell_b}(\alpha,\beta,\rho)+J_{n_a+2,m_a,\ell_a+2,n_b,m_b,\ell_b}(\alpha,\beta,\rho),\end{aligned}\tag{70}$$

$$\begin{aligned}&J_{n_a,m_a,\ell_a,n_b+2,m_b,\ell_b}(\alpha,\beta,\rho)\\&\quad=J_{n_a,m_a,\ell_a,n_b+2,m_b+2,\ell_b}(\alpha,\beta,\rho)+J_{n_a,m_a,\ell_a,n_b+2,m_b,\ell_b+2}(\alpha,\beta,\rho).\end{aligned}\tag{71}$$

Elementary trigonometry also gives $r_a\cos\theta_a+r_b\cos\theta_b=\rho$ and $r_a\sin\theta_a=r_b\sin\theta_b$. Hence

$$\begin{aligned}&J_{n_a,m_a,\ell_a,n_b+1,m_b+1,\ell_b}(\alpha,\beta,\rho)\\&\quad=\rho J_{n_a,m_a,\ell_a,n_b,m_b,\ell_b}(\alpha,\beta,\rho)-J_{n_a+1,m_a+1,\ell_a,n_b,m_b,\ell_b}(\alpha,\beta,\rho),\end{aligned}\tag{72}$$

$$J_{n_a,m_a,\ell_a,n_b+1,m_b,\ell_b+1}(\alpha,\beta,\rho)=J_{n_a+1,m_a,\ell_a+1,n_b,m_b,\ell_b}(\alpha,\beta,\rho).\tag{73}$$

In principle, replacing the Js in all of these identities by the general formula gives an expression that can be reduced to `True`. This is challenging. For present purposes, I show plausibility by evaluating the identities for a succession of Js with explicit indexes. The following function opens up the nested sums:

```
vJ[na$_, ma$_, ella$_, nb$_, mb$_, ellb$_] :=
 (eqn[Jdefn] // op[Jclosed]) //
  pipe[
   na->na$, ma->ma$, ella->ella$, nb->nb$, mb->mb$, ellb->ellb$,
   toSum[outermost][fullExpand]^8];
```

Then, typically, the following function applies the symmetry test.

```
symmetryTest[na$_, ma$_, ella$_, nb$_, mb$_, ellb$_] :=
((vJ[na$, ma$, ella$, nb$, mb$, ellb$][[2]] -
(vJ[nb$, mb$, ellb$, na$, ma$, ella$] [[2]] /.
  {alpha->beta, beta->alpha})) // Simplify) === 0
```

The statement

```
(Table[symmetryTest[na, ma, ella, nb, mb, ellb],
 {na, 0, nhat}, {ma, 0, na}, {ella, 0, na-ma},
 {nb, 0, na}, {mb, 0, nb}, {ellb, 0, nb-mb}] //
  Flatten) /. List -> Plus
```

evaluates the test for all distinct Js with $n_a\le\hat{n}$ and returns $\bar{n}\times$`True`, where $\bar{n}$ is the number of integrals that are tested.

The most exacting test of the general formula has been its use to compute large numbers of overlap integrals that match the results of my earlier work.

7. Molecular integrals

Most of my work using MATHSCAPE has dealt with the molecular integrals of computational chemistry. The overlap integrals (which are closely related to the Js) and the other kinds of molecular and atomic integral are used in work on electronic structure that starts from the Schrödinger equation. The integrals comprise the elements of massive matrices that are fed into eigenvalue software. These '*ab initio*' calculations usually approximate the 'wave functions' (eigenfunctions) by sums of products of 'atomic orbitals'. Each of these is a function of the coordinates of an electron relative to an atomic nucleus. The nuclei are considered to be at rest, relative to the movement of the electrons (Born–Oppenheimer approximation). The best approximation for a given number of terms uses Slater orbitals, also called E(lectronic) T(ype) O(rbital)s, that comprise a complete orthogonal set enumerated by the quantum numbers (n, ℓ, m)

$$\sqrt{\frac{2^{2n-\delta_m} k^{2n+1} (2\ell+1)(\ell-m)!}{(2n)!\pi(\ell+m)!}}\, \mathrm{r}^{n-1} \mathrm{e}^{-k\mathrm{r}} P_\ell^m(\cos\vartheta) \begin{Bmatrix}\cos\\ \sin\end{Bmatrix} (m\varphi) \tag{74}$$

where δ_m is the Kronecker delta (1 when $m = 0$ and 0 otherwise). The quantum numbers are integers that satisfy the constraints $n \geq 1, 0 \leq \ell \leq n, 0 \leq m \leq \ell$. P_ℓ^m is the associated Legendre polynomial. The solid spherical harmonic $r^\ell P_\ell^m(\cos\theta)\{\cos \mid \sin\}(m\phi)$ is a product of non-negative powers of (x, y, z) with total exponent ℓ. The overlap integral is a measure of the spatial overlap of two orbitals of a single electron

$$\int_{\mathbb{R}_1^3} \Psi(A, 1, q_1)\, \Psi(B, 1, q_2)\, \mathrm{d}V_1 \tag{75}$$

where $\Psi(X, i, q)$ denotes an atomic orbital of electron i associated with nucleus X, with triple quantum number q. $\mathbb{R}_i^3$ is the Euclidean space of electron i and $\mathrm{d}V_i$ is its space element.

The distance between electrons i and j is written r_{ij} and the potential energy function in the Schrödinger equation contains a term $1/r_{ij}$ for every pair of electrons in the model. Hence integrals of the form:

$$\int_{\mathbb{R}_1^3} \int_{\mathbb{R}_2^3} \Psi(A, 1, q_1)\, \Psi(B, 1, q_2) \frac{1}{r_{12}}\, \Psi(C, 2, q_3)\, \Psi(D, 2, q_4)\, \mathrm{d}V_1 \mathrm{d}V_2. \tag{76}$$

Denote the association of electrons and nuclei in (76) by $(abcd)$. The main kinds of two-electron integral are Coulomb $(aabb)$, hybrid $(aaab)$, exchange $(abab)$, 3-centre Coulomb $(aabc)$, 3-centre exchange $(abac)$ and 4-centre $(abcd)$.

As mentioned earlier in this paper, I worked on the problem of molecular integrals from 1948 until the mid 1960s. I resumed the work in about 1990 at the suggestion of two members of Leland Allen's group who used the John von Neuman National Supercomputer Laboratory where I spent part of my time as a Visiting Scientist, developing some models of biological information processing. I found a way to circumvent a convergence problem in certain 3-centre integrals (Barnett, 1989, 1990, 1991) that other authors had found in the 1970s. Then, as a Visiting Research Collaborator in the Allen group at Princeton University, I developed new formulas for the auxiliary functions needed to compute overlap, $(aabb)$ and $(aaab)$ integrals (Barnett, 2000a). I developed efficient computational procedures for the $(aabb)$ and $(aaab)$ integrals, and constructed a table of closed formulas for a considerable number of these Barnett (2000b). Then I constructed a table of formulas for nearly 10,000 overlap integrals (Barnett, 2003b) and found significant errors in work of some other authors for further overlap integrals with very high quantum numbers (Barnett, 2002). I developed a method for transforming surface harmonics and constructed an extensive table of the relevant coefficients (Barnett, 2003a). A key stage in the reduction of the $(abab)$ and $(abac)$ was resolved (Barnett, 1998b). All this work used the so-called molecular ζ-function method (Barnett and Coulson, 1951; Barnett, 1963). It is dominated by special functions of mathematical physics and manipulation of surface harmonics. Related work on electronic structure of small atoms and molecules includes (Barnett et al., 2001; Barnett and Capitani, 2006a,b). Mechanized optimization is discussed in Barnett (2003b).

Whilst work on integrals over Slater orbitals is a necessary and ongoing topic of research, mention must be made that the overwhelming body of electronic structure calculations use Gaussian wave functions that contain $\exp(-kr^2)$ instead of $\exp(-kr)$ to simplify the evaluation of the multi-centre integrals.

8. Discussion

An account of MATHSCAPE that discusses its application to the proof of mathematical formulas must take note of THEOREMA and other systems that perform proofs in more highly automated ways. The website (Theorema publications, 2004) maintains a current list of publications of the THEOREMA group of Bruno Buchberger at the University of Linz. These report work on automatic theorem proving in Zermelo–Frankel set theory, predicate logic, computational origami, merge sorting and Buchberger's algorithm. The main link to the pedagogic examples presented at ISSAC 2002 http://www.risc.uni-linz.ac.at/research/theorema/software/demos/issac) leads to further work of a similar nature. Given the power of THEOREMA and the level of interest that it engenders, there is a need to connect it to the concerns of natural scientists who use symbolic calculation. The work (Windsteiger, 2003) on the Neville polynomial interpolation algorithm provides the beginning of a bridge. Engaging natural scientists in accounts of this kind of material would be very beneficial.

Although the proof in Section 5 is directed very tightly by the user, the mechanized use of analogy in MATHSCAPE proofs that is illustrated in Section 6 and in some material accessible on the web (Barnett, 2005a) does bring MATHSCAPE applications a little towards the field of automated theorem proving — and interactivity does feature in the title of the THEOREMA work (Piroi and Kutsia, 2005). My MATHSCAPE work overlaps several projects that have links in the 'Automated deduction systems and groups' and 'Strategies in automatic deduction' websites (Automated deduction systems and groups, 2003; Systems incorporating flexible strategy-based reasoning, 2004). High level mnemonics feature in PROVERB (The PROVERB project, 2000) as well as THEOREMA and MATHSCAPE. My efforts to build a body of material concerning special functions of mathematical physics is in the spirit of MKM (Adams and Davenport, 2004).

Most mathematical discourse in the natural sciences is dominated by the use of equations and identities. Discourse in algebra, geometry and logic, however, is dominated by predicates and assertions and rules of inference. I have started to explore extensions of MATHSCAPE to deal with these (Barnett, 2005a). I find the analysis of proof methods by Pölya (Pólya, 1954) extremely helpful.

A major recent development in the THEOREMA project has been the introduction of logicographic symbols that depict mathematical entities and concepts graphically (Buchberger, 2001). An anonymous referee has suggested that I incorporate this tactic in future developments of MATHSCAPE. I do use one graphical tactic already — a 'topological' description of the computational path to recurrence formulas for sets of integrals characterized by two integer indexes (Barnett, 2003c). A possibility that comes to mind immediately is to use (nested) rectangles around a summand to indicate (multiple) summation. Then:

$$\boxed{\boxed{x_1 + x_2}} \longrightarrow \boxed{\boxed{x_1} + \boxed{x_2}}$$

depicts

$$\sum\sum(x_1 + x_2) \longrightarrow \sum\left(\sum x_1 + \sum x_2\right).$$

This convention can be extended to a variety of operations on sums, integrals and other functionals, using different line styles to distinguish different operators, allowing simple depiction of commutation.

Logicographic symbols provide a very natural approach when calculating the properties of physical, chemical, biological, geological and engineering systems that (1) are defined conveniently by diagrams, *e.g.* molecules of organic substances, and metabolic pathways, and/or (2) produce diagrams, *e.g.* spin coupling in nuclear magnetic resonance, and Feynman diagrams in high energy physics. Recently, I started to explore notations analogous to the `pipe` function to model the cumulative effect of physical processes that take place sequentially over a period of time (Barnett, 2005b). The use of logicographic symbols in this work needs careful consideration, too.

The National Research Council report 'Mathematical challenges from theoretical/computational chemistry' called on mathematicians, computer scientists, chemists and chemical biologists to overcome cultural differences in reaching out to the languages, mindsets and interests of each other (National Research Council, 1995). I was trained as a chemist but I have taught both natural science and computer science and I am vividly aware of these differences. I hope that this paper will help the common cause of bringing the subjects together.

Acknowledgements

Because a large part of this paper encapsulates material developed over the past 15 years, I thank L.C. Allen and J.F. Capitani for their support, advice and encouragement throughout this time, K.M. Perry for his essential participation in the early stages of the work, M. Minimair for help in crafting the present paper, and a referee for several valuable comments.

References

Abramowitz, M., Stegun, I., 1964. Handbook of Mathematical Functions. National Bureau of Standards, Washington, DC.

Adams, A.A., Davenport, J.H., 2004. Mathematical knowledge management. In: Lecture Notes in Computer Science, vol. 3119. Springer.

Automated deduction systems and groups, 2003. http://www-unix.mcs.anl.gov/AR/others.html.

Barnett, M.P., 1963. In: Alder, B. (Ed.), Methods of Computational Physics, vol. 2. Academic Press, New York, pp. 95–153.

Barnett, M.P., 1989. Partial fraction formulas to sum slowly convergent series. SIGSAM Bulletin 23 (3), 13–86.

Barnett, M.P., 1990. Molecular integrals over Slater orbitals. Chem. Phys. Lett. 166 (1), 65–70.

Barnett, M.P., 1991. Summing $P_n(\cos\theta)/p(n)$ for certain polynomials $p(n)$. Comput. Math. Appl. 21 (10), 79–86.

Barnett, M.P., 1993. Implicit rule formation in symbolic computation. Comput. Math. Appl. 26 (1), 35–50.

Barnett, M.P., 1998a. Combining Mathematica and TeX. TUGboat 19 (2), 147–158.

Barnett, M.P., 1998b. A prototype three-center integral. http://www.princeton.edu/~allengrp/ms/other/axcat.pdf.

Barnett, M.P., 2000a. Symbolic calculation of auxiliary functions for molecular integrals over Slater orbitals. Int. J. Quant. Chem. 76 (3), 464–472.

Barnett, M.P., 2000b. Two-center non-exchange integrals over Slater orbitals. J. Chem. Phys. 113 (21), 9419–9428.
Barnett, M.P., 2002. Digital erosion in the evaluation of molecular integrals. Theor. Chem. Acc. 107 (4), 241–245.
Barnett, M.P., 2003a. Transformation of harmonics for molecular calculations. J. Chem. Inf. Sci. 43 (4), 1158–1165.
Barnett, M.P., 2003b. Molecular integrals and information processing. Int. J. Quant. Chem. 95 (6), 791–805.
Barnett, M.P., 2003c. Symbolic calculation of integrals by recurrence. SIGSAM Bulletin 37 (2), 49–63.
Barnett, M.P., 2005a. Symbolic calculation and functional programming. http://www.princeton.edu/~allengrp/ms/mscp/scfpAll.pdf.
Barnett, M.P., 2005b. Symbolic calculation in the life sciences — some trends and prospects. In: Anai, H., Horimoto, K. (Eds.), Proceedings of the Conference Algebraic Biology (Tokyo, 2005), Universal Academy Press, Tokyo, pp. 1–18. ISSN 1880-6694. http://www.princeton.edu/~allengrp/ms/annobib/mb.pdf.
Barnett, M.P., Capitani, J.F., 2006a. Modular chemical geometry and symbolic calculation. Int. J. Quant. Chem. 106 (1), 215–227.
Barnett, M.P., Capitani, J.F., 2006b. Interfacing GAUSSIAN with MATHEMATICA. http://www.princeton.edu/~allengrp/ms/igm/igmAll.pdf.
Barnett, M.P., Coulson, C.A., 1951. Evaluation of integrals occurring in the theory of molecular structure parts I and II. Phil. Trans. R. Soc. London, Ser. A 243, 221.
Barnett, M.P., Decker, T., Krandick, W., 2001. Power series expansion of the roots of a secular equation containing symbolic elements: Computer algebra and Moseley's law. J. Chem. Phys. 114 (23), 10265–10269.
Barnett, M.P., Perry, K.R., 1994a. Hierarchical addressing in symbolic computation. Comput. Math. Appl. 28 (8), 17–35.
Barnett, M.P., Perry, K.R., 1994b. Symbolic computation for electronic publishing. TUGboat 15 (3), 285–292.
Buchberger, B., 2001. Logicographic symbols: A new feature in THEOREMA. http://www.risc.uni-linz.ac.at/people/buchberg/papers/2001-06-25-A.pdf.
Cesco, J.C., Perez, J.E., Denner, C.C., et al., 2005. Rational approximants to evaluate four-center electron repulsion integrals for 1s hydrogen Slater type functions. App. Num. Math. 55 (2), 173–190.
Coulson, C.A., 1942. Two-centre integrals occurring in the theory of molecular structure. Proc. Camb. Phil. Soc. 38, 210–223.
Gracey, J.A., 1998. Computation of perturbative renormalization group functions: The large Nf algorithm. Comp. Phys. Comm. 115 (2–3), 113–123.
Gradshteyn, I.I., Ryzhik, I.M., 1979. Tables of Integrals, Series and Products. Academic Press.
Gumus, S., 2005. On the computation of two–center Coulomb integrals over Slater type orbitals using the Poisson equation. Z. Naturforsc. A 60 (7), 477–483.
Guseinov, I.I., Mamedov, B.A., 2005. Fast evaluation of molecular auxiliary functions $A(\alpha)$ and B_n by analytical relations. J. Math. Chem. 38 (1), 21–26.
Harris, F.E., 2002. Analytic evaluation of two-center STO electron repulsion integrals via ellipsoidal expansion. Int. J. Quant. Chem. 88 (6), 701–734.
National Research Council, 1995. Mathematical challenges from theoretical/computational chemistry. National Academy Press, Washington, DC.
Öztekin, E., Yavuz, M., Atalya, S., 2001. Theor. Chem. Acta 106, 264–271.
Piroi, F., Kutsia, T., 2005. The THEOREMA environment for interactive proof development. In: Sutcliffe, G., Voronkov, A. (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. In: Lecture Notes in Artificial Intelligence, vol. 3835. Springer, pp. 261–275.
Pólya, G., 1954. Mathematics and Plausible Reasoning. Princeton University Press.
The PROVERB project, presentation of machine found proofs, 2000. http://www.ags.uni-sb.de/~afiedler/proverb/detailed.html.
Quiney, H.M., Wilson, S., 2005. Literate programming in quantum chemistry: A collaborative approach to the development of theory and computer code. Mol. Phys. 103 (2–3), 389–399.
Rico, J.F., Lopez, R., Ema, I., et al., 2005. Translation of STO charge distributions. J. Comp. Chem. 26 (8), 846–855.
Systems incorporating flexible strategy-based reasoning, 2004. http://www.logic.at/strategies/home-systems.html.
Safouhi, H., Bouferguene, A., 2006. Symbolic programming language in molecular multicenter integral problem. Int. J. Quant. Chem. 106 (1), 65–78.
http://www.risc.uni-linz.ac.at/research/theorema/software/demos/issac.
Theorema publications, 2004. http://www.theorema.org/publication.
Wang, D., Kuppermann, A., 2006. The use of symbolic algebra in the calculation of hyperspherical harmonics. Int. J. Quant. Chem. 106 (1), 152–166.
Windsteiger, W., 2003. Exploring an algorithm for polynomial interpolation in the THEOREMA system. In: Proceedings of Calculemus'03, 10–12 September 2003, Rome, Italy.
Wolfram, S., 2003. The Mathematica Book, 5th ed. Wolfram Media.