TEACHING MATHEMATICS USING *MATHEMATICA*

PAUL C. ABBOTT
Department of Physics, University of Western Australia
Nedlands, WA 6907, Australia

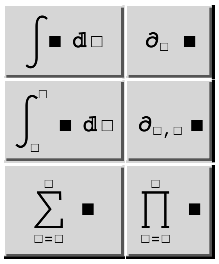
The *Mathematica* computer algebra system (CAS) is an attractive medium for teaching mathematics. New features in *Mathematica* 3.0 include editable typeset mathematical expressions; customizable palette interface; automatic arbitrary−precision control of numbers; numerical partial differential equations; and integrated hyperlinked documentation.

Mathematica is introduced through examples: antisymmetric operators; Gram−Schmidt orthogonalization (vectors and orthogonal polynomials); contour integration (residue theorem) and line integrals; the Kepler equation (series methods); and spheroidal harmonics (eigenfunctions of partial differential equations). These examples highlight the power and versatility of *Mathematica* and indicate its application to teaching mathematics.

1 Introduction

Mathematica [1] was designed as a system for doing mathematics by computer. It includes arbitrary precision and exact numerical computation, symbolic computation, graphics, sound, hyperlinked documentation, and interprocess communication—all integrated together in one easy−to−use package. More so than any other discipline, mathematics teaching has embraced the use of CAS: In particular, *Mathematica* is used extensively for teaching mathematics as a look at the *MathSource* [2] World Wide Web site reveals.

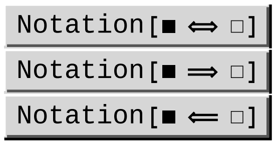
When using a CAS for teaching it is important that complicated algorithms and difficult concepts can be handled in an elegant and intuitive fashion and that their implementation and generalization is straightforward. Too often the language, syntax, and interface of the CAS overpower the pedagogy. Students and teachers can find it difficult and confusing to translate mathematical expressions into CAS syntax. The designers of *Mathematica* are well aware of this difficulty and, with version 3.0, have addressed these concerns by making the interface fully customizable. Palettes, such as the *Basic Input* palette, simplify the input of functions. For example, the sub−palette:



simplifies the input of (partial) derivatives, integrals, sums, and products. The *Notation* package, loaded via the command

```
<< Utilities‘Notation‘
```

enables users to define their own input and output syntax using palette such as:



Using this palette to define a general mapping in terms of *Mathematica*’s `Function` operator:

```
SetOptions[Notation, WorkingForm → TraditionalForm];

Notation[ f_ : x_ ↦ y_ ⟹ f_ := Function[x_ , Evaluate[y_]]]
```

mappings can then be entered using ordinary mathematical notation:

$$\boldsymbol{h : x \mapsto x\cos(x^2)}$$

$\boldsymbol{h(y^2)}$

$y^2\cos(y^4)$

This paper, typeset using *Mathematica*, presents solutions to a selected set of mathematical problems typical of those covered in undergraduate mathematics, physics, and engineering courses. The reader is invited to judge how well the *Mathematica* solutions balance syntax and pedagogy. Note that `input`, `output`, and graphics are interspersed between lines of text—exactly as generated by *Mathematica.*

2 Antisymmetric Operators

Using the high−level *signature function*, `Signature`, and the `Permutations` operator, both built into *Mathematica*, it is easy to construct completely antisymmetric functions, *i.e.*, functions that change sign under the interchange of *any* pair of arguments. `Signature` can be thought of as the totally antisymmetric Levi−Civita tensor ϵ. For example,

$\epsilon_{i\_\_}$:= Signature[{*i*}]

$\epsilon_{3,2,1}$

-1

The following simple code antisymmetrizes an arbitrary function:

Antisymmetrize[*f*_] := Module[{*p* = Permutations[*f*]}, Signature[*f*] Signature/@ *p* . *p*]

For example,

Antisymmetrize(*f*(*a*, *b*, *c*))

$f(a, b, c) - f(a, c, b) - f(b, a, c) + f(b, c, a) + f(c, a, b) - f(c, b, a)$

After swapping arguments `a` and `b` in the previous expression (denoted `%` in *Mathematica*):

% /. {*a* → *b*, *b* → *a*}

$-f(a, b, c) + f(a, c, b) + f(b, a, c) - f(b, c, a) - f(c, a, b) + f(c, b, a)$

and adding to the original function:

% + %%

0

we see that the function has indeed been antisymmetrized.

3 Gram−Schmidt Orthogonalization

Starting with the set $\{u_1, u_2, \ldots, u_n\}$, along with the *projection operator*

Projection(*f*_, *g*_) := *f* − ⟨*f*, *g*⟩ *g*

where $\langle f, g\rangle$ is any suitable *inner product*, and *norm*

$\|f\_\| := \sqrt{\langle f, f\rangle}$

Gram−Schmidt orthogonalization produces the *orthonormal set* $\{v_1, v_2, \ldots, v_n\}$ by the iterative construction: $v_1 = u_1$, $v_1 = \frac{v_1}{\|v_1\|}$, $v_2 = u_2 - < u_2, v_1 > v_1$, $v_2 = \frac{v_2}{\|v_2\|}$, $v_3 = u_3 - < u_3, v_1 > v_1 - < u_3, v_2 > v_2$, $v_3 = \frac{v_3}{\|v_3\|}$, and so on ([3], §17.4.2).

The above sequence of operations can be recognized as the pair−wise "folding" of the projection operator, followed by normalization. This *functional representation*, leads to a neat and powerful implementation of Gram−Schmidt orthogonalization using *functional programming*:

GramSchmidt := Fold$\Big[$Append$\Big[$#1, $\left(\frac{\#1}{\|\#1\|}\&\right)$[Fold[Projection, #2, #1]]$\Big]$&, {}, #1$\Big]$&;

The heart of `GramSchmidt` is the iterative projection of each vector—implemented using the functional operation `Fold`. Importantly, this code is a good example of *object−oriented programming* (OOP) as it is "re−usable" in the sense that it works for *arbitrary* projection operator, inner product, and norm. In other words, this simple piece of code can be used to orthogonalize vectors, matrices, polynomials, and functions.

Readers might feel that, even though this code is short, its workings may not be readily understood by students who do not have a solid grounding in *Mathematica* or in functional programming; the syntax could easily overpower the pedagogy. One can, if desired, "hide" the details from the student by saving `GramSchmidt` in an external package which is automatically loaded as required. In fact, the `Orthogonalization` package contains a similar routine for Gram−Schmidt orthogonalization.

3.1 Vectors

Gram−Schmidt orthogonalization can be used to construct an orthonormal set of vectors. To apply `GramSchmidt` to a set of *n*−dimensional vectors (entered as matrix), say

$$\boldsymbol{u} = \begin{pmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{3} & \mathbf{2} & \mathbf{1} \end{pmatrix};$$

we define the inner−product to be the usual `Dot` product:

$$\langle f\_, g\_\rangle = f \,.\, g;$$

The resulting orthonormal vectors are

v = GramSchmidt(u) // ExpandAll

$$\begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & -\sqrt{\frac{2}{3}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \end{pmatrix}$$

as is easily verified using `Outer` (the ordinary *outer product*) to form all possible pair−wise inner−product (*i.e.*, `⟨#1,#2⟩&`) combinations:

Outer[⟨#1, #2⟩&, v, v, 1] // MatrixForm

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

3.2 Orthogonal Polynomials

In most undergraduate mathematics courses, it is mentioned that Gram−Schmidt orthogonalization can be used to construct an orthonormal set of polynomials. However, because of the tedious computation of integrals involved, usually only very simple examples are considered. Consider constructing an orthonormal set of polynomials, $u_n[x]$, of degree n, over the interval $[-1, 1]$ with unit weight function such that $u_n'[\pm 1] == 0$. Noting that $(1 - x^2)\, x^{n-1}$ vanishes at ±1, we start with the basis

$$\boldsymbol{u_0}\,[\mathbf{x\_}] = \mathbf{1};$$

$$\boldsymbol{u_{n\_}}\,[\mathbf{x\_}] = \int (\mathbf{1} - \boldsymbol{x}^{\mathbf{2}})\, \boldsymbol{x}^{\boldsymbol{n}-\mathbf{1}}\, \boldsymbol{d}\boldsymbol{x}$$

$$x^n \left(\frac{1}{n} - \frac{x^2}{n+2}\right)$$

Note that in integrals *Mathematica* uses the special symbol ⅆ (*i.e.*, a double−struck *d*) to denote the differential element. This is not just a syntactical device to tell *Mathematica* where the integrand ends and what is the integration variable; the designers of *Mathematica* were attempting to make clearer the important distinction between *dx*—often interpreted by students as the letter *d* "times" the letter *x*—and ⅆ*x*. Many students find *dx* confusing and they are not helped by physicists and engineers who persist in "canceling" *d*'s in expressions involving differentials. Moreover, *Mathematica* uses this same syntactical device for the exponential *e*, ⅇ, the imaginary *i*, ⅈ, and so on which, unlike ordinary mathematical notation, highlights that these symbols are special.

It is easily verified that this set of polynomials, $u_n[x]$, satisfy $u_n'[\pm 1] == 0$:

$$\{u_0'(-1),\ u_0'(1),\ u_n'(-1),\ u_n'(1)\}\ //\ \mathbf{Simplify}$$

{0, 0, 0, 0}

Gram−Schmidt orthogonormalization of the set

$$u = \mathbf{Table}[u_n[x], \{n, 0, 4\}]$$

$$\left\{1,\ x\left(1-\frac{x^2}{3}\right),\ x^2\left(\frac{1}{2}-\frac{x^2}{4}\right),\ x^3\left(\frac{1}{3}-\frac{x^2}{5}\right),\ x^4\left(\frac{1}{4}-\frac{x^2}{6}\right)\right\}$$

with the inner product

$$\langle f\_,\ g\_\rangle := \int_{-1}^{1} f\, g\, ⅆx$$

yields

$$v = \mathbf{GramSchmidt}(u)\ //\ \mathbf{Simplify}$$

$$\left\{\frac{1}{\sqrt{2}},\ -\frac{1}{2}\sqrt{\frac{35}{34}}\ x\,(x^2-3),\ -\frac{1}{8}\sqrt{\frac{7}{6}}\ (15\,x^4-30\,x^2+7),\right.$$
$$\left.-\frac{1}{16}\sqrt{\frac{77}{221}}\ x\,(153\,x^4-290\,x^2+105),\ -\frac{1}{32}\sqrt{\frac{65}{159}}\ (693\,x^6-1365\,x^4+651\,x^2-43)\right\}$$

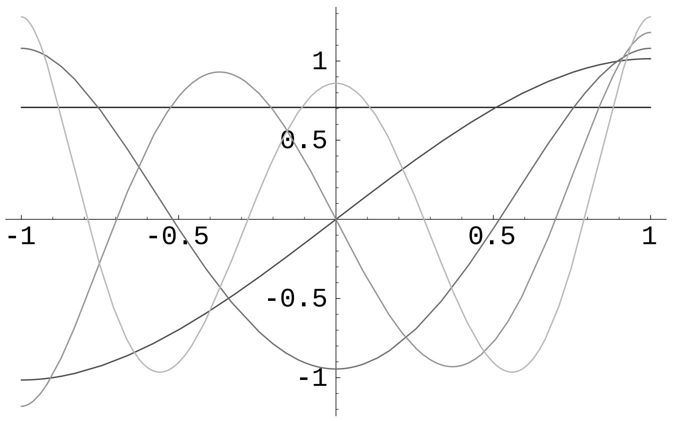
This set is orthonormal:

$$\mathbf{Outer}[\langle \#1, \#2\rangle \&,\ v,\ v,\ 1]\ //\ \mathbf{Simplify}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and visualization clearly demonstrates the behavior of these polynomials:

$$\mathbf{Plot}\left[\mathbf{Evaluate}[v],\ \{x, -1, 1\},\ \mathbf{PlotStyle} \to \mathbf{Table}\left[\mathbf{GrayLevel}\left[\frac{i}{\mathbf{Length}[v]+2}\right],\ \{i, \mathbf{Length}[v]\}\right]\right];$$



Most students observe that the resulting polynomials look somewhat similar to trigonometric functions. Moreover, the majority of science and engineering undergraduates should not find it difficult to modify this code so as to construct other sets of orthogonal polynomials.

4 Contour Integration, Line Integrals, and the Residue Theorem

Contour integrals are usually presented in mathematics courses after an introduction to complex function theory. Students are told about the great range and power of contour integration methods—yet some courses are lacking when it comes to a sufficiently wide range of examples which students can actually work through by themselves as an aid to understanding the mechanics of contour integration. Related topics such as the *residue theorem* are quite advanced and students need a good set of examples and a tool which can assist with the necessary and often complicated algebraic manipulation.

At an early stage in most undergraduate physics and engineering courses, students are introduced to *line integrals* such as $\int_\gamma \boldsymbol{F}(\boldsymbol{s})\cdot d\boldsymbol{s}$ when taught about the work done by a force, $\boldsymbol{F}$, displacing a mass, m, along a path γ. However, it is fair to say that most students do not fully appreciate how to compute such integrals in general—or even what one means by a line integral since, for conservative forces, one avoids computing such integrals directly by using energy arguments instead.

Many CAS do not have the necessary tools for students to investigate line and contour integrals by themselves. Also, to avoid confusion, if such tools do exist it is particularly important that the notations used are self−consistent and consistent with the ordinary mathematical usage. This section indicates ways of introducing line and contour integration using *Mathematica*.

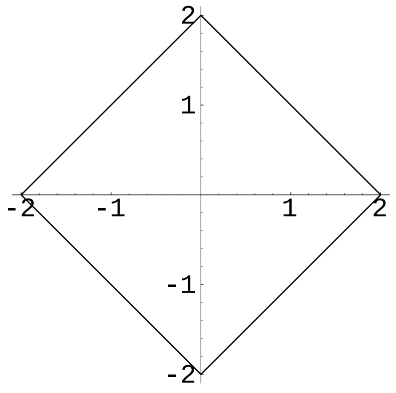
4.1 Numerical Contour Integration

Students often feel more comfortable initially if they are presented with numerical examples. Consider the closed polygonal contour, γ, specified by

$$\boldsymbol{\gamma = 2\,\{1, i, -1, -i, 1\};}$$

γ can be visualized using `ListPlot` by computing the real and imaginary parts of each vertex and transposing the resulting matrix:

$$\textbf{ListPlot}\left[\{\textbf{Re}(\gamma), \textbf{Im}(\gamma)\}^T, \textbf{PlotJoined} \to \textbf{True}, \textbf{AspectRatio} \to \textbf{Automatic}\right];$$



Using the `Notation` package it is straightforward to define an interpretation rule in natural syntax for numerical contour integrals using `NIntegrate`:

$$\texttt{Notation}\left[\oint_{\gamma\_} \texttt{f\_}\, d\texttt{z\_} \Longrightarrow \texttt{NIntegrate[f\_, Evaluate[Join[\{z\_\}, }\gamma\texttt{\_]]]}\right]$$

The numerical value of $\frac{1}{2\pi i}\oint_\gamma \frac{e^z}{z}\, dz$ is easily computed:

$$\frac{\mathbf{1}}{\mathbf{2}\,\pi\, i}\oint_\gamma \frac{e^{z}}{z}\, dz \;\textbf{// Chop}$$

1.

Note that `Chop` removes the small imaginary part. The result is the *residue* of the integrand at $z = 0$:

$$\mathbf{res}\left(\frac{e^z}{z}, \{z, 0\}\right)$$

1

For an analytic function $f[z]$, the contour integral $n = \frac{1}{2\pi i}\oint_\gamma \frac{f'[z]}{f[z]}\, dz$ gives the number of zeros, n, within the contour γ. For example, the polynomial

$$f(z\_) = z^5 - z^2 - \frac{z^3}{2} + \frac{1}{2};$$

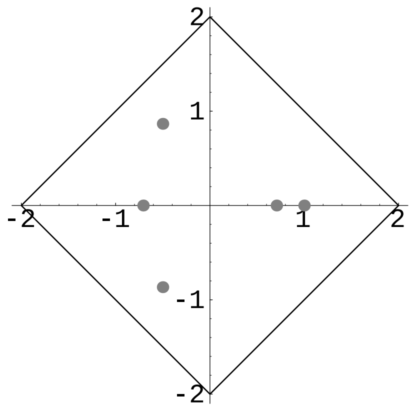
has the roots

$$r = z \;/.\; \mathbf{Solve}[f(z) == 0, z]$$

$$\left\{1, -\sqrt[3]{-1}, (-1)^{2/3}, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right\}$$

For an *arbitrary* closed contour containing these roots (we use `Epilog` to show the locations of the zeros as `Point` primitives):

ListPlot[{Re(γ), Im(γ)}^{T}, PlotJoined → True, AspectRatio → Automatic,
Epilog → {GrayLevel[0.5], PointSize[0.03], Point/@{Re(r), Im(r)}^{T}}];



the integral taken around γ confirms that there are 5 zeros inside the contour:

$$\frac{1}{2\pi i}\oint_\gamma \frac{f'(z)}{f(z)}\, dz \;//\; \mathbf{Chop}$$

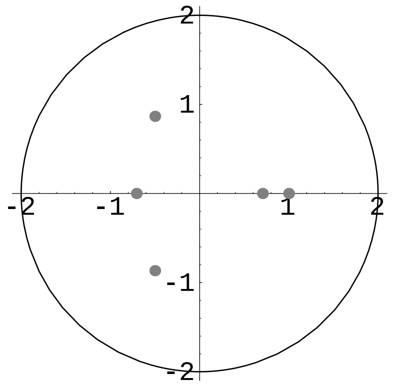
5.

It is straightforward to do contour integration over parametrically defined paths:

```
Notation[ ∫_{θ_==a_}^{θ_==b_} f_ dz_{z_→g_}  ⟹  NIntegrate[
    Evaluate[Simplify[ (f_ Dt[z_] /. z_ → g_) / Dt[θ_] ]], {θ_, a_, b_}]]
```

where we use `Dt` to change variables. For example, a circle in the complex plane can be parameterized using $r\, e^{i\theta}$. Here is a circle of radius 2:

ParametricPlot[{Re(2 $e^{i\theta}$), Im(2 $e^{i\theta}$)}, {θ, 0, 2 π}, AspectRatio → Automatic,
Epilog → {GrayLevel[0.5], PointSize[0.03], Point /@ {Re(r), Im(r)}^{T}}];



The integral of $\frac{f'[z]}{f[z]}$ around this contour again confirms that there are 5 zeros inside the contour:

$$\frac{1}{2\pi i}\int_{\theta==0}^{\theta==2\pi}\frac{f'(z)}{f(z)}\,dz_{z\to 2e^{i\theta}}\ \ \text{// Chop}$$

5.

4.2 Line Integrals

Line integrals often arise when dealing with scalar, vector, and tensor fields. In general, these integrals take the form $\int_c \boldsymbol{P}(\boldsymbol{x})\cdot d\boldsymbol{x}$, where c denotes the integration path. In three−dimensions, an explicit form is $\int_c P(x, y, z)\,dx + Q(x, y, z)\,dy + R(x, y, z)\,dz$ where x, y, and z are the coordinates along the path c.

One way of dealing with line integrals is to parameterize the path by specifying the coordinates in terms of a parameter t: $x = f(t)$, $y = g(t)$, $z = h(t)$, and the range of t−values which traces out the desired path, c. A direct implementation [4]

LineIntegrate(f : Literal[_. d_ | +(_. d_..)], c_Equal, p : {t_, _, _}, x : {__}) :=
$\int$(f /. First[Solve[c, x]] /. {dt → 1, Literal[Dt][_] → 0}) dp

might appear somewhat obtuse, but its application is straightforward. The pattern f : Literal[_. d_ | +(_. d_..)] uses `Alternatives` (*i.e.*, `|`) to test that the argument corresponds to a one−dimensional line integral $P(x)\,dx$ or to a multi−dimensional line integral of the form $P(x, y, \ldots)\,dx + Q(x, y, \ldots)\,dy + \ldots$, *etc.* The use of dx ensures that the appropriate change of variables takes place automatically. For example

dx /. x → sin(θ)

cos(θ) $d\theta$

`Solve` is used to express the variables in terms of the parameter, and the substitutions enable the parametric integral to be computed using `Integrate`.

Here is a particular three−dimensional example. Integrate the vector function

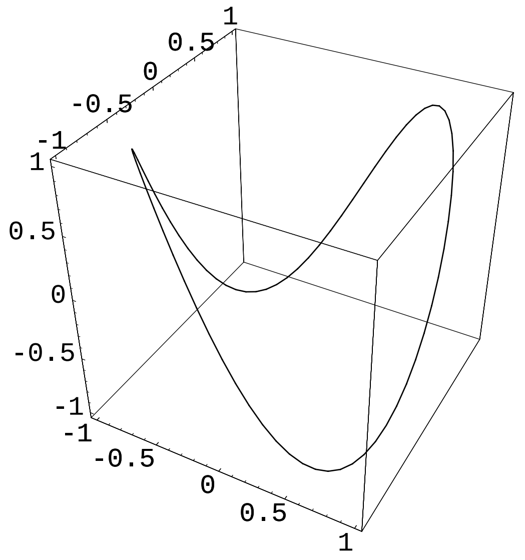
f({x_, y_, z_}) := {x^2 y, y z, z x^3}

along the path c:

c(θ_) = {cos(θ), sin(θ), sin(2 θ)};

Visualizing the parametric path is easy:

ParametricPlot3D[Evaluate[$c(\theta)$], {θ, 0, 2 π}];



Defining the cartesian variables:

x = {x, y, z};

the line integral is immediate:

LineIntegrate($f(x) . dx$, x == $c(\theta)$, {θ, 0, 2 π}, x) // Expand

$$\frac{\pi}{4}$$

4.3 Residue Theorem

Consider computing the integral $I_n = \frac{1}{2\pi i}\int_{-i\infty}^{i\infty}\frac{\mathcal{D}_n[z]}{\mathcal{A}_n[z]\,\mathcal{A}_n[-z]}\,dz$, where $\mathcal{A}_n[z]$ is a Hurwitz polynomial

$$\mathcal{A}_{n\_}[z\_] := \sum_{i=0}^{n} \alpha_i \, z^{n-i}$$

and

$$\mathcal{D}_{n\_}[z\_] := \sum_{i=0}^{n-1} \delta_i \, z^{2(-i+n-1)}$$

It is not trivial for a CAS (or a student) to compute I_n directly. However, from the *residue theorem* (see §§4.1−3 of[5]), if g and h are analytic at z_i and $g[z_i] \neq 0$, $h[z_i] == 0$, and $h'[z_i] \neq 0$, then $f[z] = \frac{g[z]}{h[z]}$ has a *simple pole* at z_i, the residue of f at z_i is Res(f, z_i) $= \frac{g[z_i]}{h'[z_i]}$, and $\frac{1}{2\pi i}\int_\gamma \frac{g[z]}{h[z]}\,dz = \sum_i \frac{g[z_i]}{h'[z_i]}$, where γ is a simple closed curve enclosing the z_i. Hence, it can be seen that the integral I_n reduces to computing a sum of $\frac{\mathcal{D}_n[z]}{(\partial_z \mathcal{A}_n[z])\,\mathcal{A}_n[-z]}$ over the simple poles, *i.e.*, the roots of $\mathcal{A}_n[z]$. It should be apparent that this is how one would compute such integrals by hand. Moreover, using the built−in `RootSum` function, one can compute I_n directly as follows:

$$\mathcal{I}_{n\_} := \text{RootSum@@Function/@}\left\{\mathcal{A}_n[\#],\ \frac{\mathcal{D}_n[\#]}{\frac{\partial \mathcal{A}_n[\#]}{\partial \#}\,\mathcal{A}_n[-\#]}\right\} \text{ // Simplify}$$

For example,

$\mathcal{I}_4$

$$\frac{\alpha_1\,\alpha_4\,(\alpha_0\,\delta_2 - \alpha_4\,\delta_0) + \alpha_0\,\alpha_3\,(\alpha_0\,\delta_3 - \alpha_4\,\delta_1) + \alpha_2\,(\alpha_3\,\alpha_4\,\delta_0 - \alpha_0\,\alpha_1\,\delta_3)}{2\,\alpha_0\,\alpha_4\,(\alpha_4\,\alpha_1^2 + \alpha_0\,\alpha_3^2 - \alpha_1\,\alpha_2\,\alpha_3)}$$

This example indicates how one can add extra knowledge, such as the residue theorem, to a CAS.

5 Kepler Equation

Methods involving power series are fundamental in mathematics, science, and engineering courses. However, because of algebraic complexity, most courses usually only cover trivial examples which hide the true power of such methods from students. It is important that the implementation of power series in a CAS be elegant, powerful, general, automatic, easy to use, and recognizably similar to ordinary mathematical notation—for otherwise the student will not see the relationship to the mathematical method being taught.

The Kepler equation, $s = u + e \sin s$, arises in celestial mechanics. Regarding e as a small quantity, this transcendental equation can be solved using power series. *Mathematica* syntax naturally lends itself to the solution of this problem. Defining $s = u + a$, the equation reduces to $a = e \sin(u + a)$, which can be solved by introducing the power series recurrence relation:

$$a_{k\_} := a_k = e \sin(u + a_{k-1}) + O(e)^{k+1};$$

Since e is a small parameter, as $k \to \infty$, $a_{k-1} \to a_k \to a$, and hence the Kepler equation is formally satisfied. Note that *dynamic programming* (*i.e.*, the syntax $a_{k\_} := a_k = \ldots$) is used to store the intermediate computations. Defining the initial condition:

$$a_0 = 0;$$

we immediately obtain

$$a_2$$

$$\sin(u)\, e + \cos(u) \sin(u)\, e^2 + O(e^3)$$

Here is the sixth approximation converted into a *Fourier series* expansion:

$$a_6 \;//\; \text{Normal} \;//\; \text{TrigReduce}$$

$$\frac{1}{1920}(40 \sin(2u)\, e^6 - 512 \sin(4u)\, e^6 + 648 \sin(6u)\, e^6 + 10 \sin(u)\, e^5 - 405 \sin(3u)\, e^5 + 625 \sin(5u)\, e^5 - 320 \sin(2u)\, e^4 + 640 \sin(4u)\, e^4 - 240 \sin(u)\, e^3 + 720 \sin(3u)\, e^3 + 960 \sin(2u)\, e^2 + 1920 \sin(u)\, e)$$

`Normal` converts the power series into a polynomial, `TrigReduce` rewrites products and powers of trigonometric functions in terms of trigonometric functions with combined arguments, and `//` indicates *postfix* action of an operator (similar to the Unix `pipe`). It is straightforward to collect together all the terms involving `Sin` (using the pattern `Sin[_]` with the *generic symbol*, `_`, and *pattern−matching*) in the previous expression:

```
Collect[%, sin(_)]
```

$$\frac{27}{80} \sin(6u)\, e^6 + \frac{125}{384} \sin(5u)\, e^5 + \left(\frac{e^5}{192} - \frac{e^3}{8} + e\right) \sin(u) + \left(\frac{e^6}{48} - \frac{e^4}{6} + \frac{e^2}{2}\right) \sin(2u) + \left(\frac{3e^3}{8} - \frac{27e^5}{128}\right) \sin(3u) + \left(\frac{e^4}{3} - \frac{4e^6}{15}\right) \sin(4u)$$

The self−consistency of this solution (*i.e.*, whether $a = e \sin(u+a)$) is easily checked:

$$a_8 - e \sin(u + a_8) == O(e)^9$$

True

6 Spheroidal Harmonics

Computing the spheroidal harmonics—something not normally covered in undergraduate mathematics courses— demonstrates hybrid numerical−symbolic computation and is a nice example of the power of *Mathematica*. Importantly, because the syntax does not dominate the message and the example can be easily modified to handle a wide range of differential equations, such hybrid methods for solving differential equations could be presented to undergraduate students.

6.1 Legendre Functions

Traditional mathematical notation is often ambiguous. Hence computer algebra systems cannot always correctly interpret arbitrary mathematical expressions. One solution is to define your own interpretation rules. Here we use the `Notation` package to attach input and output formatting rules to `LegendreP` enabling automatic interpretation of associated Legendre functions using the syntax $P_n^m(z)$. After entering the desired format:

```
Notation[𝒫ₙ_^m_[z_] ⟺ LegendreP[n_, m_, z_], WorkingForm → StandardForm]
```

then, for suitable parameters, closed−form expressions result:

$$\boldsymbol{P_{11}^{5}(a)}$$

$$-\frac{135135}{16}\sqrt{1-a^2}\,(a^2-1)^2\,(2261\,a^6-1615\,a^4+255\,a^2-5)$$

Using this notation, it is straightforward to implement a Legendre function *recurrence relation* (§8.5.3 of [6]), here called $\mathcal{R}$, which automatically reduces sums of products of (integral) powers of z^k multiplied by $P_\nu^\mu(z)$:

$$\boldsymbol{\mathcal{R} = z\_^{k\_.}\; P_{\nu\_}^{\mu\_}(z\_) \to \frac{(\mu+\nu)\,P_{\nu-1}^{\mu}(z)\,z^{k-1} + (\nu-\mu+1)\,P_{\nu+1}^{\mu}(z)\,z^{k-1}}{2\,\nu+1};}$$

Mathematica knows some basic properties of the associated Legendre functions, such as their derivatives. Hence, using the recurrence relation above, it is straightforward to verify that the $P_\nu^\mu(z)$ satisfy the differential equation

$$\boldsymbol{(1-z^2)\,\frac{\partial^2\,P_\nu^\mu(z)}{\partial z^2} - \left(2\,z\,\frac{\partial P_\nu^\mu(z)}{\partial z}\right) + \left(\nu\,(\nu+1) - \frac{\mu^2}{1-z^2}\right) P_\nu^\mu(z) == 0;}$$

After multiplying the left−hand side (*i.e.*, the `First` part) of the differential equation by $(1-z^2)$:

$$\boldsymbol{(1-z^2)\,\mathrm{First}[\%]\;//\;\mathrm{Together}}$$

$$P_{\nu-2}^{\mu}(z)\,\mu^2 - P_\nu^\mu(z)\,\mu^2 + 2\,\nu\,P_{\nu-2}^{\mu}(z)\,\mu - P_{\nu-2}^{\mu}(z)\,\mu + z\,P_{\nu-1}^{\mu}(z)\,\mu - 2\,z\,\nu\,P_{\nu-1}^{\mu}(z)\,\mu + \nu^2\,P_{\nu-2}^{\mu}(z) - \nu\,P_{\nu-2}^{\mu}(z) - 2\,z\,\nu^2\,P_{\nu-1}^{\mu}(z) + z\,\nu\,P_{\nu-1}^{\mu}(z) + \nu^2\,P_\nu^\mu(z)$$

then repeated application (using `//.`) of the recurrence relation ($\mathcal{R}$) and simplification (using `Collect` and `Factor`) yields

$$\boldsymbol{\mathrm{Factor}\,/@\,\mathrm{Collect}[\%\;//.\;\mathcal{R},\;P_{\_}^{\_}(z)]}$$

0

6.2 Spheroidal Harmonics

Spheroidal harmonics typically arise when solving classes of partial differential equations using separation of variables in spheroidal coordinates (see §17.4 of [7] and chapter 21 of [6]). The differential equation for the (angular prolate) spheroidal harmonics is

$$\boldsymbol{(1-\eta^2)\,\frac{\partial^2\,S_{n,m}[c][\eta]}{\partial\eta^2} - 2\,\eta\,\frac{\partial S_{n,m}[c][\eta]}{\partial\eta} + \left(\lambda_{m,n} - c^2\,\eta^2 - \frac{m^2}{1-\eta^2}\right) S_{n,m}[c][\eta] == 0;}$$

where m is an integer, c is the *oblateness* parameter, and $\lambda_{m,n}$ is the eigenvalue. Despite the notation, c^2 can be positive or negative. The equation has singular points at $\eta = \pm 1$ and the boundary conditions require the solution to be regular at $\eta = \pm 1$.

From a *Mathematica* perspective, the notation $S_{n,m}[c][\eta]$ is preferable to $S_{m,n}[c, \eta]$ because it leads to expressions only containing *ordinary* derivatives. One can think of $S_{m,n}[c]$ as a *pure function* and η as a *dummy argument*. Such notation is advantageous because it is easily generalized and is useful when working with operators, pure functions, and compiled functions.

One method for solving differential equations is to expand the solution into a suitable basis. Rewriting the differential equation as

$$\mathcal{D} = (1-\eta^2)\frac{\partial^2 S_{n,m}[c][\eta]}{\partial \eta^2} - 2\eta \frac{\partial S_{n,m}[c][\eta]}{\partial \eta} + \left((k+m)(k+m+1) - \frac{m^2}{1-\eta^2}\right) S_{n,m}[c][\eta] == (c^2\eta^2 + (k+m+1)(k+m) - \lambda_{m,n}) S_{n,m}[c][\eta];$$

reveals that it is very similar to the associated Legendre differential equation (see §6.1). In fact, since $P_{m+k}^{m}(z)$ satisfies the left−hand side of the differential equation,

$$(1-\eta^2)\,\text{First}[\mathcal{D}] \,/.\, S_{n,m}[c] \to \text{Function}[z, P_{m+k}^{m}(z)] \,//\, \text{Together};$$

$$\text{Collect}[\% \,//.\, \mathcal{R}, P_{\_}^{\_}(z\_), \text{Factor}]$$

0

it makes sense to use the *eigenfunction expansion* $S_{n,m}[c][\eta] = \sum_{k=0}^{\infty} d_k^{m,n}[c]\, P_{m+k}^{m}(\eta)$. The recurrence relation satisfied by $d_k^{m,n}[c]$ is easily generated. First, we substitute a *generic* term of the eigenfunction expansion (using a pure function denoted by `Function`) into the right−hand side of the differential equation:

$$\text{Last}[\mathcal{D}] \,/.\, S_{n,m}[c] \to \text{Function}[z, d_k\, P_{m+k}^{m}(z)];$$

Note that we have used the shorthand d_k to represent $d_k^{m,n}[c]$. We then (repetitively) apply the associated Legendre function recurrence relation to simplify the resulting expression:

$$(\text{Expand}[\%] \,//.\, \mathcal{R}) \,//\, \text{Expand};$$

Since k is a dummy summation index, we can use *pattern matching* to find the recursion relationship that must be satisfied by the d_k:

$$r = \frac{\% \,/.\, P_{m+k+a\_.}^{m}(z\_)\, d_k\, c\_ :\to (P_{m+k+a}^{m}(z)\, d_k\, c \,/.\, k \to k-a)}{P_{m+k}^{m}(\eta)};$$

The recurrence relation can be written in the form (§27.7.3 of [6]) $\alpha_k\, d_{k+2} + (\beta_k - \lambda_{m,n})\, d_k + \gamma_k\, d_{k-2} == 0$, where the coefficients are easily determined as follows:

$$\alpha_{k\_} = \text{Collect}[\text{Coefficient}[r, d_{k+2}], c, \text{Factor}]$$

$$\frac{c^2\,(k+2m+1)(k+2m+2)}{(2k+2m+3)(2k+2m+5)}$$

$$\beta_{k\_} = \text{Collect}[\text{Coefficient}[r, d_k], c, \text{Factor}] + \lambda_{m,n}$$

$$\frac{(2k^2+4mk+2k+2m-1)\,c^2}{(2k+2m-1)(2k+2m+3)} + k^2 + m^2 + k + 2km + m$$

$$\gamma_{k\_} = \text{Collect}[\text{Coefficient}[r, d_{k-2}], c, \text{Factor}]$$

$$\frac{c^2\,(k-1)\,k}{(2k+2m-3)(2k+2m-1)}$$

By truncating the infinite sum, $\sum_{k=0}^{\infty} d_k^{m,n}[c]\, P_{m+k}^{m}(\eta)$, to $N+1$ terms, the recurrence relation $\alpha_k\, d_{k+2} + \beta_k\, d_k + \gamma_k\, d_{k-2} == \lambda_{m,n}\, d_k$ leads to the tri−diagonal matrix eigenvalue problem, $\mathcal{A}_{\mathcal{N}} \,.\, \boldsymbol{d} == \lambda_{m,n}\, \boldsymbol{d}$, where

$$\mathcal{A}_{\mathcal{N}\_}[m\_][c\_] := \text{Table}[\text{Switch}[j-k, 2, \alpha_k, 0, \beta_k, -2, \gamma_k, \_, 0] \,/.\, \{m \to m, c \to c\}, \{k, 0, \mathcal{N}\}, \{j, 0, \mathcal{N}\}]$$

For example, with symbolic parameters and $N = 2$, the matrix reads

$\mathcal{A}_2[m][c]$ // MatrixForm

$$\begin{pmatrix} \frac{c^2}{2m+3}+m^2+m & 0 & \frac{c^2\,(2m+1)(2m+2)}{(2m+3)(2m+5)} \\ 0 & \frac{(6m+3)c^2}{(2m+1)(2m+5)}+m^2+3m+2 & 0 \\ \frac{2c^2}{(2m+1)(2m+3)} & 0 & \frac{(10m+11)c^2}{(2m+3)(2m+7)}+m^2+5m+6 \end{pmatrix}$$

If we supply (exact) numerical parameters, say $m = 0$ and $c = 1$, and set $N = 8$, the matrix becomes

$\mathcal{A}_8[0][1]$ // MatrixForm

$$\begin{pmatrix} \frac{1}{3} & 0 & \frac{2}{15} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{13}{5} & 0 & \frac{6}{35} & 0 & 0 & 0 & 0 & 0 \\ \frac{2}{3} & 0 & \frac{137}{21} & 0 & \frac{4}{21} & 0 & 0 & 0 & 0 \\ 0 & \frac{2}{5} & 0 & \frac{563}{45} & 0 & \frac{20}{99} & 0 & 0 & 0 \\ 0 & 0 & \frac{12}{35} & 0 & \frac{1579}{77} & 0 & \frac{30}{143} & 0 & 0 \\ 0 & 0 & 0 & \frac{20}{63} & 0 & \frac{3569}{117} & 0 & \frac{14}{65} & 0 \\ 0 & 0 & 0 & 0 & \frac{10}{33} & 0 & \frac{7013}{165} & 0 & \frac{56}{255} \\ 0 & 0 & 0 & 0 & 0 & \frac{42}{143} & 0 & \frac{12487}{221} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{56}{195} & 0 & \frac{20663}{285} \end{pmatrix}$$

It is apparent that the recurrence relation links only even or odd values of k. Hence, one could separate the solution (and the matrix) into two cases. However, in the following, both cases are treated together. This has the benefit that, for a different differential equation (where the recurrence relation may contain more than three terms or link even and odd values of k), the same method still works.

Now we compute and sort (using `Sort` and `Transpose`) the set of eigenvectors (d) in order of increasing eigenvalue ($\lambda_{m,n}$) :

$\{\lambda, d\}$ = Sort$\left[\text{Eigensystem}[\% \,//\, N]^T\right]^T$;

NumberForm[λ, 8]

```
{0.31900006, 2.5930846, 6.5334718, 12.514462, 20.508274, 30.505405, 42.503818,
  56.504696, 72.503857}
```

These eigenvalues agree perfectly with Table 21.1 of [6]. Nevertheless, they are only approximate because we have truncated the infinite sum. Increasing N leads to a better approximation.

We need to normalize the spheroidal harmonics. A number of normalization conventions exist in the literature; perhaps the simplest one is the Stratton−Morse−Chu−Little−Corbato scheme, $\sum_{r\geq 0} \frac{(r+2m)!}{r!}\, d_r = \frac{(n+m)!}{(n-m)!}$, which has the effect that $S_{n,m}[c][\eta] \to P_n^m(\eta)$ as $\eta \to 1$.

In the following we truncate the system by setting $N = 25$. Combining the steps above, and using *list manipulation* (*Mathematica* can work directly with list or vector operations applied to functions), the normalization can be simultaneously applied to all the eigenvectors, yielding the *whole set* of eigenfunctions (and eigenvalues) for fixed m and c:

$S[m\_][c\_] := S[m][c] = \text{Module}\big[\{d, k, n, r, \eta, \mathcal{N} = 25\},$

$\{\lambda[m][c], d\}$ = Reverse/@Eigensystem[$\mathcal{A}_{\mathcal{N}}[m][c]$ // N] // Chop;

$$d = d\,\frac{\text{Table}\left[\frac{(n+m)!}{(n-m)!}, \{n, m, \mathcal{N}+m\}\right]}{d\,.\,\text{Table}\left[\frac{(r+2m)!}{r!}, \{r, 0, \mathcal{N}\}\right]};$$

Compile[η, Evaluate[d . Table[$P_{m+k}^m(\eta)$, $\{k, 0, \mathcal{N}\}$]]]$\big]$

The syntax is quite elegant: Pure function notation coupled with *dynamic programming* is used to record the spheroidal harmonics (and their corresponding eigenvalues) as they are computed; List manipulation enables the whole *set* of harmonics (for a given m and c) to be computed in one go; the associated Legendre functions are built−in and exact closed−form expressions are available (so their evaluation is rapid and accurate); and `Compile` makes the computation fast. Because *Mathematica* supports arbitrary precision arithmetic, it is simple to modify this code to work with any desired precision. Furthermore, the relationship between this code and the algebraic expressions presented in the literature is immediate thus greatly reducing the chance of coding error. As an aside, efficient computation of the derivatives of the spheroidal harmonics is trivially implemented by modifying the above code to include derivatives of the associated Legendre functions which are computed exactly.

To project out a particular eigenfunction, we use the notation

$$S_{n\_}[m\_][c\_][\eta\_] /; m \le n := S[m][c][\eta][\![n - m + 1]\!]$$

The eigenvalues are produced as a *side−effect* of the computation of the related eigenfunction:

$$\lambda_{m\_,n\_}[c\_] /; m \le n := (S[m][c];\ \lambda[m][c][\![n - m + 1]\!])$$

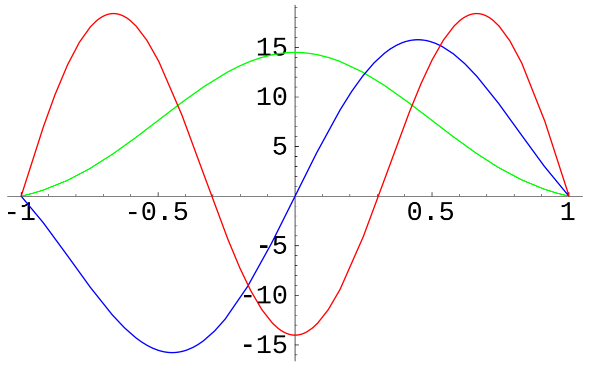
To give a concrete example, with $m = 2$ and $c = 5$, the first three eigenvalues are

$$\{\lambda_{2,2}[5], \lambda_{2,3}[5], \lambda_{2,4}[5]\}$$

```
{8.74767, 19.136, 29.6288}
```

with corresponding eigenfunctions:

$$\text{Plot}\left[\{S_2[2][5][x], S_3[2][5][x], S_4[2][5][x]\}, \{x, -1, 1\}, \text{PlotStyle} \to \text{Table}\left[\text{Hue}\left[\frac{i}{3}\right], \{i, 3\}\right]\right];$$



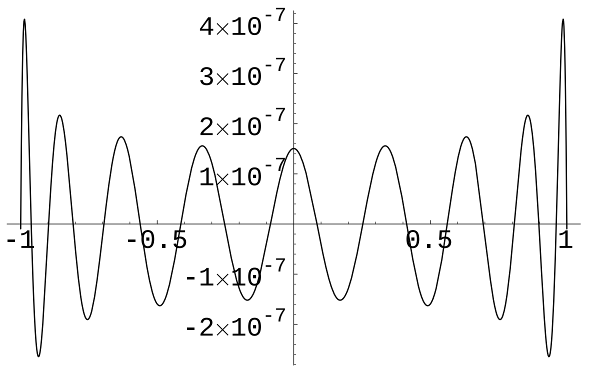
Writing the spheroidal differential equation as

$$\mathcal{D}_{m\_}[n\_][c\_][\eta\_] = \partial_\eta\,((1-\eta^2)\,\partial_\eta\, S_m[n][c][\eta]) + \left(\lambda_{m,n}[c] - c^2\,\eta^2 - \frac{m^2}{1-\eta^2}\right) S_m[n][c][\eta];$$

it is then straightforward to see how well each numerical solution satisfies the differential equation:

Off[CompiledFunction::cfr]

$$\text{Plot}[\text{Evaluate}[\mathcal{D}_2[2][5][\eta]], \{\eta, -1, 1\}];$$



Conclusions

I hope that these examples have demonstrated that *Mathematica* is an important and useful tool for teaching mathematics. The Notebook interface—which includes two−dimensional typeset mathematical input and output, hyperlinked text, and palettes—is, by itself, an excellent presentation, teaching, and research tool. I have found that students enjoy using *Mathematica* and, more importantly, they use it not just for the topics I teach them, but for a wide range of subjects including mathematics, geophysics, and chemistry. Of course, students usually try to use the tools that they are taught—which makes it especially important that we teach them using the best tools available. In our teaching, rather than using a collection of small programs designed to elucidate one or two concepts, it is advantageous if the tools we offer have wide application and I argue that *Mathematica* definitely qualifies in this respect.
In summary:

Antisymmetric Operators showed that, because the *Mathematica* programming language is very high−level, operations such as symmetry and permutation are built−in and easy to use. Hence the teacher can program at the desired level of abstraction. Functional programming is closely related to the way mathematics is actually done—*e.g.*, working with operations such as permutation—rather than focusing on implementation details like "do−loops", "for−loops", or "if−then statements", as required in procedural programming languages. Since the functional code is direct and simple it is easy for students to understand and generalize.

Gram−Schmidt Orthogonalization indicated how object−oriented programming is useful for implementing and working with abstract entities such as projection operators, inner products, and norms. One small piece of functional code orthogonalizes vectors, matrices, polynomials, and functions. Importantly, students and teachers do not need to know *Mathematica* well, nor understand functional programming, to be able to use the `GramSchmidt` routine.

In *Contour Integration, Line Integrals, and the Residue Theorem*, several examples of contour integration were presented. *Mathematica* includes tools enabling students to investigate line and contour integrals by themselves. Also, it is possible to introduce a notation and corresponding syntax for contour integration that is self−consistent and consistent with the ordinary mathematical usage.

Since most real−world problems are not exactly soluble in closed−form, approximation methods are usually required. *Kepler Equation* showed how symbolic series solution of a transcendental equation is straightforward. Although exact solution is possible for this example, series methods are generally useful and it is easy to use such series to achieve any desired numerical accuracy.

The *Spheroidal Harmonics* section showed how *Mathematica* can be used to obtain the recurrence relation, eigenvalues, and spheroidal harmonics *directly* from the differential equation. Using list manipulation, the whole set of eigenvalues and harmonics are produced and manipulated as a single object. The *Mathematica* code is quite simple because the implementation is directly and easily related to formulae presented in the literature. This reduces the chance of coding error and makes it easier for students to follow.

References

[1] Wolfram, S, *The Mathematica Book*, 3rd edition, Wolfram Media, Champaign, 1996.

[2] *MathSource*, http://www.wolfram.com/mathsource

[3] Pearson, CE (ed.), *Handbook of Applied Mathematics*, Van Nostrand Reinhold, 1990.

[4] Bahder, T, *Mathematica for Scientists and Engineers*, Addison−Wesley, 1995.

[5] Mardsen, JE, *Basic Complex Analysis*, W. H. Freeman, 1973.

[6] Abramowitz, M and Stegun, IA, *Handbook of Mathematical Functions*, Dover, 1970.

[7] Press, WH, Teukolsky, SA, Vetterling, WT, and Flannery, BP, *Numerical Recipes in C*, CUP,1992.