Tricks of the Trade

Edited by Paul Abbott

This is a column of programming tricks and techniques, most of which, we hope, will be contributed by our readers, either directly as submissions to *The Mathematica Journal* or as an edited answer to a question posted in the *Mathematica* newsgroup, comp.soft-sys.math.mathematica.

■ Interpolation with Noise

Vittorio G. Caffa

caffa@iabg.de

In "Interpolation," *TMJ*, **9**(2), 2004, pp. 306–310, a method was presented for computing derivatives of a sampled function. This method only works if the data is unaffected by measuring error. If models of the data source and the measurement noise are available, you can use (for example) Kalman filtering techniques for computing an optimal estimation (see mathworld.wolfram.com/KalmanFilter.html). Alternatively, you can achieve good results using a simple filtering technique.

Consider again the exact function $y(x) = x \sin(x) - x/2$.

```
In[1]:= y(x_) = x Sin[x] - x/2;
```

Take x from a uniform random distribution over the interval $(a, b) = (1, 12)$, and append the endpoints.

```
In[2]:= {a, b} = {1, 12};
        SeedRandom[4321];
        xs = Join[{a}, Table[Random[Real, {a, b}], {25}], {b}];
```

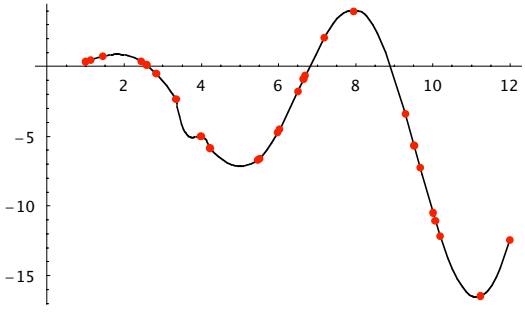
Now sample $y(x)$ over the interval adding Gaussian noise corresponding to measurement error.

```
In[3]:= Needs["Statistics`ContinuousDistributions`"];
        f(x_) := {x, y(x) + Random[NormalDistribution[0, 0.02]]};
        data = f /@ xs;
```

Superimpose the sampled data over a plot of its interpolated function to visualize the $\{x, y\}$ data. It is clear that the interpolation is not smooth, so computing derivatives will be problematic.

```
In[6]:= y_int = Interpolation[data];
```

In[7]:= **Plot[$y_{\text{int}}(x)$, {x, a, b}, Epilog → {Hue[1], AbsolutePointSize[3], Point /@ data}]**



A simple filtering technique involves computing a truncated Fourier series of the function $y(x)$, that is,

$$y_n(x) = a_0 + \sum_{k=1}^{n} (a_k \cos(k\,\omega\,x) + b_k \sin(k\,\omega\,x)).$$

Requiring the wavelength λ to satisfy $\lambda = 2\,\pi/\omega > 2\,(b-a)$ avoids wrap-around effects. As only the frequencies up to $n\,\omega$ are considered, the method acts like a low-pass filter. The value of the parameter n should be tuned to obtain the best results. This approach should be compared with that used in "Spectral Analysis of Irregularly Sampled Data," *TMJ*, **7**(1), 1997, pp. 27–28.

Here is the Fourier basis set for arbitrary n and λ.

In[8]:= **basis[n_, λ_] :=**

$$\textbf{With}\left[\left\{\omega = \frac{2\,\pi}{N[\lambda]},\ k = \textbf{Range}[n]\right\},\ \textbf{Join}[\{1\},\ \cos(k\,\omega\,x),\ \sin(k\,\omega\,x)]\right]$$

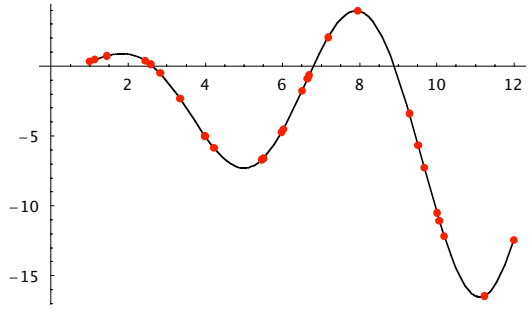
The best truncated Fourier expansion, y_n, for $\lambda = 30 > 2\,(b-a) = 22$ and $n = 5$ is computed using **Fit**.

In[9]:= **y_5 = Function[x, Evaluate[Fit[data, basis[5, 30], x]]]**

Out[9]= Function[x, 69.3068 cos(0.20944 x) + 95.0578 cos(0.418879 x) −
45.9663 cos(0.628319 x) + 5.88829 cos(0.837758 x) − 5.53855 cos(1.0472 x) +
186.779 sin(0.20944 x) − 79.2066 sin(0.418879 x) − 23.3035 sin(0.628319 x) +
8.88347 sin(0.837758 x) + 1.3561 sin(1.0472 x) − 118.99]

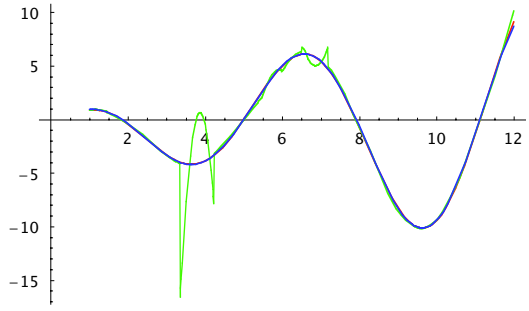
Here is the sampled data superimposed over a plot of y_5.

In[10]:= **Plot[$y_5(x)$, {x, a, b}, Epilog → {Hue[1], AbsolutePointSize[3], Point /@ data}]**



Compare the first derivatives; the red curve denotes the exact value (—), the green curve denotes the interpolated function (—), and the blue curve denotes the truncated Fourier fit y_5 (—). The agreement between the exact function and the truncated Fourier fit is excellent.

In[11]:= **Plot[{$y'(x)$, $y'_{\text{int}}(x)$, $y'_5(x)$}, {x, a, b}, PlotStyle → {Hue[0], Hue[$\frac{1}{3}$], Hue[$\frac{2}{3}$]}]**



■ Using Reduce To Solve The Kuhn–Tucker Equations

Frank J. Kampas

fkampas@msn.com

The functions **Maximize** and **Minimize** give exact solutions to simple nonlinear optimization problems with equality and inequality constraints. Another approach is to set up Lagrange multipliers for equality constraints and the Kuhn–Tucker equations for inequality constraints (see mathworld.wolfram.com/Kuhn-TuckerTheorem.html) and solve using **Reduce**. In some cases, this approach can solve problems which cannot directly be solved with **Maximize** and **Minimize**. The following function demonstrates this approach.

```
In[1]:= KuhnTucker(obj_, cons_List, vars_, domain___) :=
  Module[{consconvrule = {x_ ≥ y_ → y − x ≤ 0,
        x_ > y_ → y − x < 0, x_ == y_ → x − y == 0, x_ ≤ y_ → x − y ≤ 0},
    stdcons, eqcons, ineqcons, lambdas, mus, numequals,
    numinequals, lagrangian, eqs1, eqs2, eqs3},
   stdcons = cons /. consconvrule; eqcons = Cases[stdcons, x_ == 0 :> x];
   ineqcons = Cases[stdcons, x_ ≤ 0 :> x];
   numequals = Length[eqcons]; numinequals = Length[ineqcons];
   lambdas = Array[λ, numequals]; mus = Array[μ, numinequals];
   lagrangian = obj + lambdas.eqcons + mus.ineqcons;
   eqs1 = (∂lagrangian/∂#1 == 0 &) /@ vars; eqs2 = Thread[mus ≥ 0];
   eqs3 = Table[mus[[i]] ineqcons[[i]] == 0, {i, numinequals}];
   Reduce[Join[eqs1, eqs2, eqs3, cons],
    Join[vars, lambdas, mus], domain, Backsubstitution → True]]
```

Here is an example with an equality constraint in which both the maximum and minimum are obtained. Defining the objective function with a constraint and listing it first in the list of variables places the objective function first in the results.

```
In[2]:= KuhnTucker[objf, {objf == x + y + z, x^2 + y^2 + z^2 == 1}, {objf, x, y, z}]
```

$$\text{Out[2]}= \left(\text{objf} = -\sqrt{3} \bigwedge x = -\frac{1}{\sqrt{3}} \bigwedge y = -\frac{1}{\sqrt{3}} \bigwedge z = -\frac{1}{\sqrt{3}} \bigwedge \lambda(1) = -1 \bigwedge \lambda(2) = \frac{\sqrt{3}}{2}\right) \bigvee \left(\text{objf} = \sqrt{3} \bigwedge x = \frac{1}{\sqrt{3}} \bigwedge y = \frac{1}{\sqrt{3}} \bigwedge z = \frac{1}{\sqrt{3}} \bigwedge \lambda(1) = -1 \bigwedge \lambda(2) = -\frac{\sqrt{3}}{2}\right)$$

Equivalent results are obtained by using **Maximize** and **Minimize**.

```
In[3]:= Maximize[objf, {objf == x + y + z, x^2 + y^2 + z^2 == 1}, {objf, x, y, z}] // Simplify
```

$$\text{Out[3]}= \left\{\sqrt{3}, \left\{\text{objf} \to \sqrt{3}, x \to \frac{1}{\sqrt{3}}, y \to \frac{1}{\sqrt{3}}, z \to \frac{1}{\sqrt{3}}\right\}\right\}$$

```
In[4]:= Minimize[objf, {objf == x + y + z, x^2 + y^2 + z^2 == 1}, {objf, x, y, z}] // Simplify
```

$$\text{Out[4]}= \left\{-\sqrt{3}, \left\{\text{objf} \to -\sqrt{3}, x \to -\frac{1}{\sqrt{3}}, y \to -\frac{1}{\sqrt{3}}, z \to -\frac{1}{\sqrt{3}}\right\}\right\}$$

KuhnTucker can solve the problem when the righthand side of the constraint is changed from a number to a symbol.

In[5]:= **KuhnTucker[objf, {objf == x + y + z, x^2 + y^2 + z^2 == c}, {objf, x, y, z}]**

Out[5]=
$$\left(\text{objf} = -\sqrt{3}\ \sqrt{c} \bigwedge x = -\frac{\sqrt{c}}{\sqrt{3}} \bigwedge y = -\frac{\sqrt{c}}{\sqrt{3}} \bigwedge z = -\frac{\sqrt{c}}{\sqrt{3}} \bigwedge \lambda(1) = -1 \bigwedge c \neq 0 \bigwedge \lambda(2) = \frac{\sqrt{3}}{2\sqrt{c}}\right) \bigvee \left(\text{objf} = \sqrt{3}\ \sqrt{c} \bigwedge x = \frac{\sqrt{c}}{\sqrt{3}} \bigwedge y = \frac{\sqrt{c}}{\sqrt{3}} \bigwedge z = \frac{\sqrt{c}}{\sqrt{3}} \bigwedge \lambda(1) = -1 \bigwedge c \neq 0 \bigwedge \lambda(2) = -\frac{\sqrt{3}}{2\sqrt{c}}\right)$$

Maximize and **Minimize** give an error message in this situation.

In[6]:= **Maximize[objf, {objf == x + y + z, x^2 + y^2 + z^2 == c}, {objf, x, y, z}]**

> Maximize::consv : The constraint $x^2 + y^2 + z^2 = c$ contains a nonconstant expression c independent of variables {objf, x, y, z}. More…

Out[6]= Maximize[objf, {objf = x + y + z, x^2 + y^2 + z^2 = c}, {objf, x, y, z}]

In the following example, the constraint $x \leq b$ is only active if b is less than 1. Therefore, the solution obtained covers $b \leq 1$ and $b \geq 1$ separately.

In[7]:= **KuhnTucker[objf, {objf == x^2 − 2 x, x ≤ b}, {objf, x}]**

Out[7]=
$$(b \leq 1 \bigwedge \text{objf} = b^2 - 2b \bigwedge x = b \bigwedge \lambda(1) = -1 \bigwedge \mu(1) = -2(b-1)) \bigvee (b \geq 1 \bigwedge \text{objf} = -1 \bigwedge x = 1 \bigwedge \lambda(1) = -1 \bigwedge \mu(1) = 0)$$

When the solution is degenerate, all the results can be obtained.

In[8]:= **KuhnTucker[objf, {objf == x + y, x + y == 1}, {objf, x, y}]**

Out[8]= $\text{objf} = 1 \bigwedge y = 1 - x \bigwedge \lambda(1) = -1 \bigwedge \lambda(2) = -1$

The following problem can be solved quite easily if the objective function is not defined as a constraint. Adding the extra constraint greatly increases the solution time. The problem cannot be solved by **Minimize** or **NMinimize** unless a change of variables is used to eliminate the square roots.

In[9]:= **KuhnTucker[1/2 (t_b − Sqrt[q_b] + t_l − Sqrt[q_l]), {t_l ≥ 20 q_l, t_b ≥ 2 q_b, t_l − 20 q_l ≥ t_b − 20 q_b, t_b − 2 q_b ≥ t_l − 2 q_l}, {q_l, q_b, t_l, t_b}]**

Out[9]=
$$q_l = \frac{1}{5776} \bigwedge q_b = \frac{1}{16} \bigwedge t_l = \frac{5}{1444} \bigwedge t_b = \frac{185}{1444} \bigwedge \mu(1) = 1 \bigwedge \mu(2) = 0 \bigwedge \mu(3) = 0 \bigwedge \mu(4) = \frac{1}{2}$$

Using **Reduce** with domain ℝ specified eliminates complex solution candidates.

```
In[10]:= KuhnTucker[objf, {objf == x + y, x^3 + y^3 == 1}, {objf, x, y}, Reals] //
           FullSimplify
```

$$\text{Out[10]= objf} = 2^{2/3} \bigwedge x = \frac{1}{\sqrt[3]{2}} \bigwedge y = \frac{1}{\sqrt[3]{2}} \bigwedge \lambda(1) = -1 \bigwedge \sqrt[3]{2}\ \lambda(2) = -\frac{2}{3}$$

Note, however, that using Lagrange multipliers and the Kuhn–Tucker equations relies on assumptions that the code does not check. For instance, the solution set of equational constraints should be a manifold of dimension $v - e$, where v is the number of variables and e is the number of equations. If this is not satisfied, **KuhnTucker** may not find the extremum.

```
In[11]:= KuhnTucker[x + y, {z == x^2 - x y, z == y^2 + x y, z == x^2 + y^2}, {x, y, z}]
Out[11]= False
```

Maximize and **Minimize** work in this situation.

```
In[12]:= Minimize[x + y, {z == x^2 - x y, z == y^2 + x y, z == x^2 + y^2}, {x, y, z}]
Out[12]= {0, {x → 0, y → 0, z → 0}}

In[13]:= Maximize[x + y, {z == x^2 - x y, z == y^2 + x y, z == x^2 + y^2}, {x, y, z}]
Out[13]= {0, {x → 0, y → 0, z → 0}}
```

■ Legendre–Gauss Quadrature

Gaussian quadrature gives the best estimate of an integral by picking optimal abscissas, x_i, at which to evaluate the function $f(x)$. It is optimal because the n-point formula is exact for polynomials of degree $\leq 2n - 1$. Legendre–Gauss quadrature (see mathworld.wolfram.com/Legendre-GaussQuadrature.html) is a Gaussian quadrature over the interval $[-1, 1]$ with weighting function $W(x) = 1$, that is,

$$\int_{-1}^{1} f(x)\, dx \approx \sum_{i=1}^{n} w_i\, f(x_i),$$

where the abscissas, x_i, for quadrature order n, satisfy $P_n(x_i) = 0$, and the weights are

$$w_i = \frac{2\,(1 - {x_i}^2)}{(n+1)^2\, P_{n+1}(x_i)^2}.$$

Define the abscissas as the roots of $P_n(x)$ using **Root** and save them as they are computed. This is an *exact* expression for the roots.

```
In[1]:= x /: x_i_,n_ := x /: x_i,n = Root[Function[x, P_n(x)], i]
```

Define the weights. **RootReduce** expresses the weights as a *single* **Root** object. Save the weights as they are computed.

```
In[2]:= w /: w_{i_,n_} := w /: w_{i,n} = 2 (1 - x_{i,n}^2) / ((n + 1)^2 P_{n+1}(x_{i,n})^2) // RootReduce
```

Here are the x_i and w_i for $n = 3$.

```
In[3]:= With[{n = 3}, Table[{x_{i,n}, w_{i,n}}, {i, n}]]
```

$$\text{Out[3]=}\quad \begin{pmatrix} -\sqrt{\frac{3}{5}} & \frac{5}{9} \\ 0 & \frac{8}{9} \\ \sqrt{\frac{3}{5}} & \frac{5}{9} \end{pmatrix}$$

Here is x_2 and w_2 for $n = 5$, expressed as radicals.

```
In[4]:= With[{n = 5}, {x_{2,n}, w_{2,n}} // ToRadicals]
```

$$\text{Out[4]=}\quad \left\{-\frac{1}{3}\sqrt{\frac{1}{7}\left(35 - 2\sqrt{70}\right)},\ \frac{1}{900}\left(322 + 13\sqrt{70}\right)\right\}$$

Abscissas and weights for higher n are better left as **Root** objects (see "Root versus Radicals," *TMJ*, **9**(3), 2005, pp. 535–537). Note that, for fixed n, each w_i can be expressed as a particular root of a polynomial of degree $\lfloor \frac{n}{2} \rfloor$.

Here are the weights for $n = 6$.

```
In[5]:= With[{n = 6}, Table[w_{i,n}, {i, n}]]
```

```
Out[5]= {Root[2025000 #1^3 - 2025000 #1^2 + 629325 #1 - 58564 &, 1],
  Root[2025000 #1^3 - 2025000 #1^2 + 629325 #1 - 58564 &, 2],
  Root[2025000 #1^3 - 2025000 #1^2 + 629325 #1 - 58564 &, 3],
  Root[2025000 #1^3 - 2025000 #1^2 + 629325 #1 - 58564 &, 3],
  Root[2025000 #1^3 - 2025000 #1^2 + 629325 #1 - 58564 &, 2],
  Root[2025000 #1^3 - 2025000 #1^2 + 629325 #1 - 58564 &, 1]}
```

The weights satisfy $\sum_{i=1}^{n} w_i = 2$. Here we check this for $n = 6$.

```
In[6]:= Tr[%] // RootReduce
```

```
Out[6]= 2
```

Now consider computing the integral of a polynomial of degree m, denoted $p_m(x)$.

```
In[7]:= p_{m_}(x_) := a_0 + Sum[a_j x^j, {j, 1, m}];
```

Compute the integral of $p_7(x)$ over $-1 \le x \le 1$.

```
In[8]:= Integrate[p_7(x), {x, -1, 1}]
```

$$\text{Out[8]=}\quad 2\,a_0 + \frac{2\,a_2}{3} + \frac{2\,a_4}{5} + \frac{2\,a_6}{7}$$

As expected, an identical answer is obtained using Legendre–Gauss quadrature with $n = 4$ since $2n - 1 = 7$.

```
In[9]:= With[{n = 4}, Sum[w_{i,n} p_7(x_{i,n}), {i, 1, n}]] // FullSimplify
```

$$\text{Out[9]=}\quad 2\,a_0 + \frac{2\,a_2}{3} + \frac{2\,a_4}{5} + \frac{2\,a_6}{7}$$

■ Pascal Matrices

Writing Pascal's triangle as a lower triangular matrix generates one type of *Pascal matrix* (see mathworld.wolfram.com/PascalMatrix.html). Such matrices have many interesting properties (web.mit.edu/18.06/www/pascal-work.pdf).

Define the lower triangular matrix L_n by $\{L_n\}_{i,j} = \binom{i}{j}$, where $i, j = 0, 1, \ldots, n-1$.

```
In[1]:= L_n_ := Table[Binomial[i, j], {i, 0, n - 1}, {j, 0, n - 1}]
```

Here is L_5.

```
In[2]:= L_5
```

$$\text{Out[2]=}\quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 1 & 3 & 3 & 1 & 0 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

And here is its inverse.

```
In[3]:= L_5^-1
```

$$\text{Out[3]=}\quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ -1 & 3 & -3 & 1 & 0 \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix}$$

In general, the inverse of L_n has $\{L_n^{-1}\}_{i,j} = (-1)^{i+j}\binom{i}{j}$.

Clearly $\{L^p\}_{i,j} = p^{i-j}\binom{i}{j}$.

In[4]:= **MatrixPower[L_5, p]**

Out[4]=
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ p & 1 & 0 & 0 & 0 \\ p^2 & 2\,p & 1 & 0 & 0 \\ p^3 & 3\,p^2 & 3\,p & 1 & 0 \\ p^4 & 4\,p^3 & 6\,p^2 & 4\,p & 1 \end{pmatrix}$$

Define $v_n(x) = \{x^i\}_{i=0,1,\ldots,n}$.

In[5]:= **$v_{n\_}$ (x_) := $x^{\text{Range}[n]-1}$**

Using the binomial theorem, we can show that $L_n^p\, v_n(x) = v_n(p + x)$, that is,

$$\begin{pmatrix} 1 & & & & \\ p & 1 & & & \\ p^2 & 2\,p & 1 & & \\ p^3 & 3\,p^2 & 3\,p & 1 & \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \\ \vdots \end{pmatrix} = \begin{pmatrix} 1 \\ p+x \\ (p+x)^2 \\ (p+x)^3 \\ \vdots \end{pmatrix}.$$

"Prove" the binomial theorem.

In[6]:= **Simplify$\left[(p+x)^i = \sum_{j=0}^{i} \binom{i}{j} x^j\, p^{i-j},\ i \in \mathbb{Z}\right]$**

Out[6]= True

Verify that $L_n^p\, v_n(x) = v_n(p + x)$ for $n = 5$.

In[7]:= **MatrixPower[L_5, p].$v_5(x)$ // Factor**

Out[7]= $\{1,\ p + x,\ (p + x)^2,\ (p + x)^3,\ (p + x)^4\}$

In[8]:= **% == $v_5(p + x)$**

Out[8]= True

Writing $L_n^p = e^{A_n\, p}$, it follows that $A_n = \log(L_n) = \lim_{p\to 0}\ (L_n^p - I_n)/p$. Here is A_5.

In[9]:= **$A_5 = \lim_{p\to 0} \dfrac{\text{MatrixPower}[L_5,\ p] - \text{IdentityMatrix}[5]}{p}$**

Out[9]=
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \end{pmatrix}$$

In general, it can be shown that $\{A_n\}_{i,j} = i\,\delta_{i,j+1}$.

Check that $L_5 = e^{A_5}$.

In[10]:= **MatrixExp[A_5]**

Out[10]=
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 1 & 3 & 3 & 1 & 0 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

Define the upper triangular matrix $\mathcal{U}_n$ by alternating the sign of the rows in L_n^T.

In[11]:= $\mathcal{U}_{n\_} := (-1)^{\text{Range}[n]-1}\, L_n{}^T$

Here is $\mathcal{U}_5$.

In[12]:= $\mathcal{U}_5$

Out[12]=
$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & -1 & -2 & -3 & -4 \\ 0 & 0 & 1 & 3 & 6 \\ 0 & 0 & 0 & -1 & -4 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$\mathcal{U}_n$ is its own inverse (it is *involutory*).

In[13]:= $\mathcal{U}_5 . \mathcal{U}_5$

Out[13]=
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Multiplying L_n by its transpose yields a symmetric, positive-definite matrix, denoted S_n, now with Pascal's triangle entries along each skew diagonal (see mathworld.wolfram.com/SkewDiagonal.html).

In[14]:= $S_{n\_} := L_n . L_n{}^T$

In[15]:= S_4

Out[15]=
$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 6 & 10 \\ 1 & 4 & 10 & 20 \end{pmatrix}$$

Since $S_n = L_n \; . \; L_n^T$ and $\{L_n^T\}_{k,j} = \{L_n\}_{j,k}$, we obtain that

$$\{S_n\}_{i,j} = \sum_{k=0}^{i} \{L_n\}_{i,k} \{L_n\}_{j,k} = \binom{i+j}{i} = \binom{i+j}{j}.$$

In[16]:=
$$\sum_{k=0}^{i} \binom{i}{k}\binom{j}{k}$$

Out[16]=
$$\frac{\Gamma(i+j+1)}{\Gamma(i+1)\,\Gamma(j+1)}$$

In[17]:= **FullSimplify[% == Binomial[i + j, i] == Binomial[i + j, j]]**

Out[17]= True

The inverse of S_n has integer entries.

In[18]:= **S_5^{-1}**

Out[18]=
$$\begin{pmatrix} 5 & -10 & 10 & -5 & 1 \\ -10 & 30 & -35 & 19 & -4 \\ 10 & -35 & 46 & -27 & 6 \\ -5 & 19 & -27 & 17 & -4 \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix}$$

It can be shown that $S_n^{-1} = \mathcal{U}_n^{-1} \,.\, S_n \,.\, \mathcal{U}_n$. That is S_n^{-1} is similar to S_n (see mathworld.wolfram.com/SimilarMatrices.html).

Verify that $S_n^{-1} = \mathcal{U}_n^{-1} \,.\, S_n \,.\, \mathcal{U}_n$ for $n = 5$.

In[19]:= **S_5^{-1} == $\mathcal{U}_5^{-1}$. S_5 . $\mathcal{U}_5$**

Out[19]= True

Since similar matrices have the same eigenvalues, the eigenvalues of S_n must come in reciprocal pairs, λ and λ^{-1}. Moreover, if n is odd, then the "middle" eigenvalue must be unity.

Here are the eigenvalues of S_3.

In[20]:= **Eigenvalues[S_3]**

Out[20]= $\{4 + \sqrt{15},\ 1,\ 4 - \sqrt{15}\}$

Confirm that the eigenvalues come in reciprocal pairs.

In[21]:= **1 / %**

Out[21]= $\left\{\frac{1}{4+\sqrt{15}},\ 1,\ \frac{1}{4-\sqrt{15}}\right\}$

In[22]:= **RootReduce[%]**

Out[22]= $\{4 - \sqrt{15},\ 1,\ 4 + \sqrt{15}\}$

■ Generalized Padé Approximation

It is often the case that, while an exact solution is difficult or impossible, series or asymptotic solutions to a particular problem can be obtained without too much difficulty. As is well known, Padé approximations (see mathworld.wolfram.com/PadeApproximant.html) are usually superior to Taylor expansions when functions contain poles, because the use of rational functions allows them to be well represented.

Define $P_N(x)$ and $Q_M(x)$ as polynomials of degrees N and M, with $Q_M(0) = 1$.

```
In[1]:= P_n_(x_) := Sum_{i=0}^{n} p_i x^i
```

```
In[2]:= Q_m_(x_) := 1 + Sum_{i=1}^{m} q_i x^i
```

The coefficients in the $[N, M]$ Padé approximant to $f(x)$ are determined by requiring that the Taylor expansion of $f(x) - P_N(x)/Q_M(x)$ is of order $O(x^{N+M+1})$.

Suppose that you have obtained the following Taylor series up to $O(x^5)$ for a function of interest.

```
In[3]:= taylor[x_] = 1 - 2 x/3 + 4 x^2/15 - 8 x^3/105 + 16 x^4/945;
```

Here is the [2, 2] Padé approximant for this series.

```
In[4]:= << Calculus`
```

```
In[5]:= pade[x_] = Pade[taylor[x], {x, 0, 2, 2}]
```

```
Out[5]= (32 x^2/945 - 2 x/9 + 1) / (4 x^2/63 + 4 x/9 + 1)
```

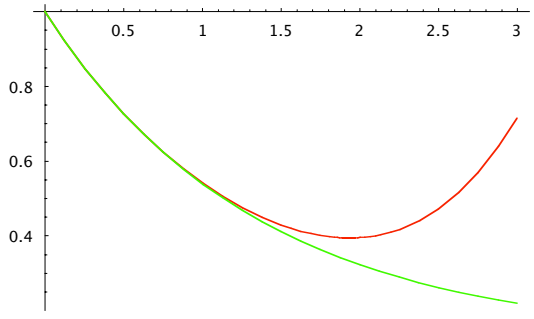
Verify that the Taylor expansion of $f(x) - P_2(x)/Q_2(x)$ is of order $O(x^5)$.

```
In[6]:= taylor[x] - pade[x] + O[x]^5
```

```
Out[6]= O(x^5)
```

A plot shows that the Taylor series (—) diverges from the Padé approximant (—) near $x = 1$.

```
In[7]:= small = Plot[{taylor[x], pade[x]}, {x, 0, 3}, PlotStyle → {Hue[1], Hue[1/3]}]
```



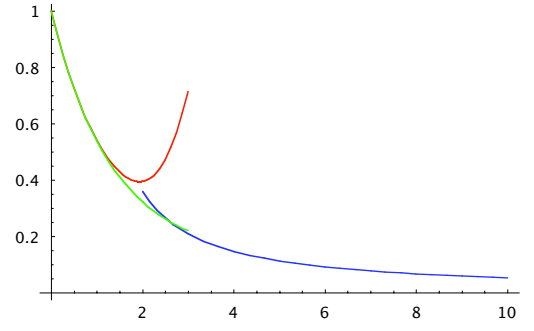
Suppose now that you have also computed the following asymptotic series for the same function.

```
In[8]:= asymptotic[x_] = 1/(2 x) + 1/(4 x^2) + 3/(8 x^3);
```

A plot shows that the Padé approximant about $x = 0$ (—) and the asymptotic expansion (—) intersect for $x \approx 2.5$.

```
In[9]:= << Graphics`
```

```
In[10]:= large = DisplayTogether[
           Plot[asymptotic[x], {x, 2, 10}, PlotStyle → Hue[2/3]], small]
```



How can you incorporate information from expansions of a function about two or more points? One approach is to generalize the idea of computing Padé approximants [1]. Suppose $f(x)$ has the asymptotic expansions

$$f(x) \sim \sum_{i=0}^{\infty} a_i (x - x_0)^i, \; x \to x_0,$$

$$f(x) \sim \sum_{i=0}^{\infty} b_i (x - x_1)^i, \; x \to x_1,$$

in the neighborhoods of distinct points x_0 and x_1, respectively. To compute a two-point Padé approximant to $f(x)$, we require that the rational function $P_N(x)/Q_M(x)$ agrees with $f(x)$ up to order $\mathcal{J}$ about x_0 and to order K about x_1, where $\mathcal{J} + K = N + M + 1$.

To use all the information at hand, note that for $x_0 = 0$, $\mathcal{J} = 5$, and for $x_1 = \infty$, $K = 4$.

```
In[11]:= j = Exponent[taylor[x], x] + 1
Out[11]= 5
In[12]:= k = Exponent[asymptotic[x], 1 / x] + 1
Out[12]= 4
```

The diagonal Padé approximant then has $N = M = 4$.

```
In[13]:= n = m = 1/2 (j + k - 1)
Out[13]= 4
```

It is straightforward to determine the coefficients p_i and q_i in terms of a_i and b_i. Here we obtain these coefficients using series arithmetic via **Solve**.

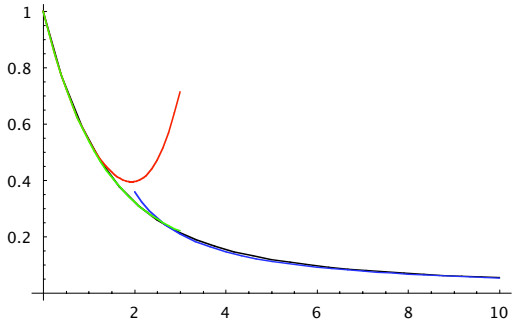
```
In[14]:= Solve[
           {taylor[x] - P_n(x)/Q_m(x) + O[x]^j == 0, asymptotic[x] - P_n(x)/Q_m(x) + O[x, ∞]^k == 0},
           Join[Table[p_i, {i, 0, n}], Table[q_i, {i, m}]]]
Out[14]= {{p_2 → 52/945, p_3 → 8/945, p_1 → 2/9, p_4 → 0,
           p_0 → 1, q_2 → 8/21, q_3 → 32/315, q_1 → 8/9, q_4 → 16/945}}
```

Here is the two-point Padé approximant to $f(x)$.

```
In[15]:= twopoint[x_] = P_n(x)/Q_m(x) /. First[%]
Out[15]= (8 x^3/945 + 52 x^2/945 + 2 x/9 + 1) / (16 x^4/945 + 32 x^3/315 + 8 x^2/21 + 8 x/9 + 1)
```

The two-point Padé approximant (—) agrees with the Taylor series about $x = 0$ (—), the one-point Padé approximant about $x = 0$ (—), and the asymptotic expansion (—) about $x = \infty$.

In[16]:= **DisplayTogether[Plot[twopoint[*x*], {*x*, 0, 10}], large]**



References

[1] C. M. Bender and S. A. Orszag, *Advanced Mathematical Methods for Scientists and Engineers*, New York: McGraw-Hill, 1978, pp. 393–395.

Paul Abbott
School of Physics, M013
University of Western Australia
35 Stirling Highway
Crawley WA 6009, Australia
tmj@physics.uwa.edu.au
physics.uwa.edu.au/~paul