Hybrid Symbolic-Numeric Computation: A Personal View

Robert M. Corless

July 2026

Editor-in-Chief, Maple Transactions
Western University, Canada

Dedicated to the memory of Hans J. Stetter (1935–2025)
and to the memory of Cleve B. Moler (1939–2026)
These slides available at rcorless.github.io

Overview of this ICMS Session

I'll be giving a *personal* view of hybrid symbolic-numeric computation in this talk, but our session and all its speakers will give a fuller view. In particular, other speakers will demonstrate

- Applications
- Series methods, where the symbolic or automatic methods support numerical computation
- Certifications, where formal or symbolic methods are used to verify the truth conveyed by numerical computation
- Replacing polynomial problems by eigensystems
- Numerical methods for tackling the notoriously difficult task of solving symbolic systems with parameters
- Other questions difficult to classify

Maple Transactions

Announcing Maple Transactions
an open access journal with no page charges or other fees
mapletransactions.org

Several hybrid symbolic-numeric papers have been published in this journal; indeed there was already one in the first issue, by Nick Trefethen and the late Peter Baddoo on Log-lightning computation of capacity and Green's function.

I also mention some papers on Bohemian Matrices, a field in which hybrid symbolic-numeric thinking is useful.

Definitions (many choices are possible)

- **Algebraic computation**: exact deterministic computation with symbols (including numerals), starting with exact data. Continuity is often unimportant because infinitesimal changes might be impossible ("rigidity"). Arbitrary precision floats are permitted, sometimes.
- **Numerical computation**: approximate computation with approximate values (possibly represented using fixed-precision floating-point), starting with potentially inaccurate or imprecise data. Continuity is important because changes (hopefully only small changes) are inevitable.
- **Hybrid computation**: any computation that has aspects of both the above. Structured perturbations and structured continuity can be important.

Hans Stetter said (in his book *Numerical Polynomial Algebra*) that coefficients that were not allowed to change were "intrinsic" while coefficients that were allowed to change were "empiric." This is a nice distinction.

Another point of view

Symbolic computation is when you want a formula as the answer. *Numeric* computation is when you want a number (or a plot) as the answer. When you use numerics to get a formula, it's hybrid.

A symbolic example: the steady-state solution to $\ddot{y} + 2\beta y + y = \cos \Omega t$ is frequently written symbolically using the formula

$$y = \frac{1}{\sqrt{(\Omega^2 - 1)^2 + 4\Omega^2\beta^2}} \cos(\Omega t - \phi) \,, \tag{1}$$

where the phase ϕ is chosen to combine the sine and cosine terms:

$$\phi = \arctan(2\Omega\beta, 1 - \Omega^2) \,. \tag{2}$$

Formulæ may be flawed

That such formulæ may have lacunæ where they do not apply was perhaps first noted explicitly by Cauchy in his 1821 *Cours d'Analyse* (English translation presented at the MacTutor site):

> *We must even note that they suggest that algebraic formulas have an unlimited generality, whereas in fact the majority of these formulas are valid only under certain conditions and for certain values of the quantities they contain. By determining these conditions and these values, and by fixing precisely the sense of all the notations I use, I make all uncertainty disappear.*

Cleve Moler (1939–2026)

Cleve Moler taught me that you can do “symbolic” computation with vectors of floating-point numbers. But you can sometimes miss critical points that way, too.

Vale, Cleve.

A cut-down version of Cleve's example

$$\begin{bmatrix} \frac{1}{e} & \frac{1}{e} & \frac{1}{e} & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & e & e & e \end{bmatrix} \tag{3}$$

where e "is not necessarily the base of the natural logarithm." Originally a 32-by-32 matrix that Kenton Yee of BNL wished to solve (factor) and which Cleve and I discussed eigenvalues/eigenvectors of an 8-by-8 version. Special cases here: $e = 1$, $e = \alpha$ where $\alpha^2 + 6\alpha + 1 = 0$, and $e = \beta$ where $\beta^4 + 4\beta^3 - 6\beta^2 + 4\beta + 1 = 0$.

The eigenvalues, done in Cleve's style

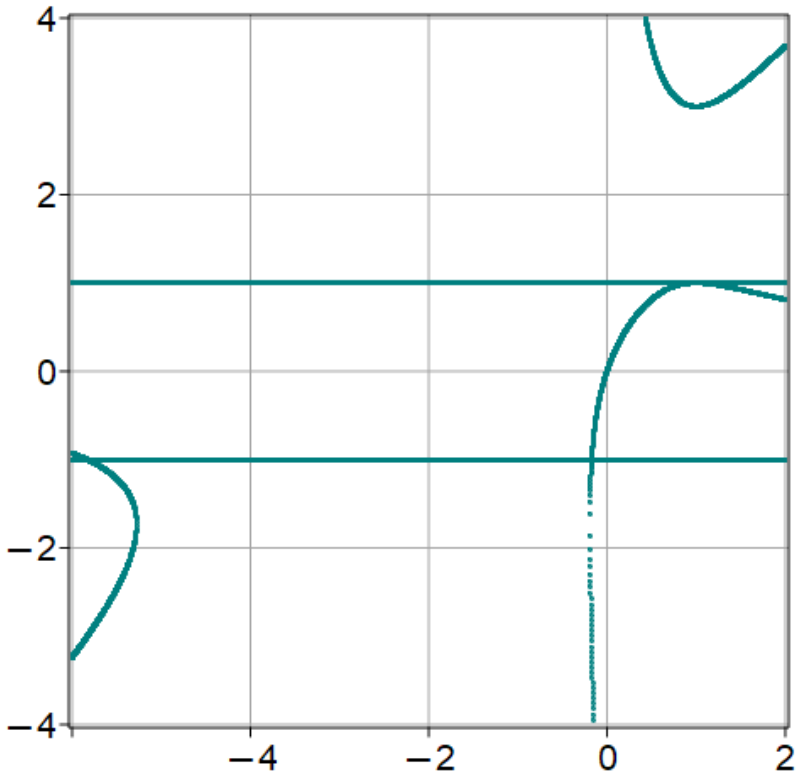

Figure 1: Eigenvalues of that 4 by 4 matrix, as e varies

A plot, computed numerically from a formula

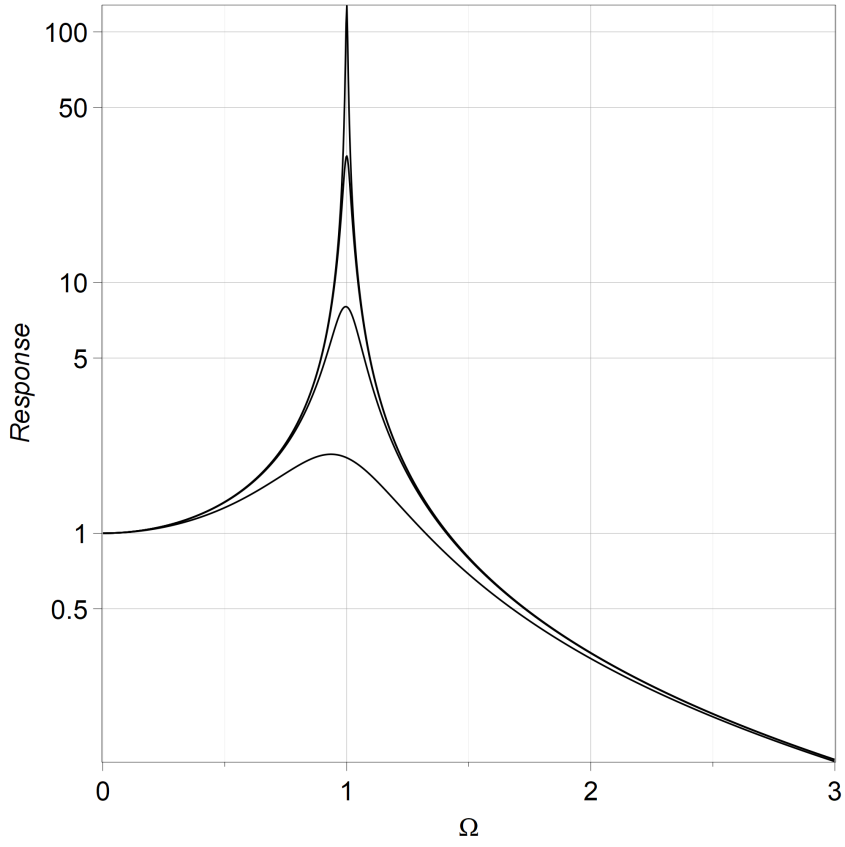

Figure 2: Amplitude of the steady-state solution of $\ddot{y} + 2\beta\dot{y} + y = \cos\Omega t$, which is $\left((\Omega^2 - 1)^2 + 4\Omega^2\beta^2\right)^{-1/2}$, for different damping coefficients $\beta = 4^{-k}$ for $k = 1, 2, 3$, and 4, plotted on $0 \leq \Omega < 3$.

Another, again numerical

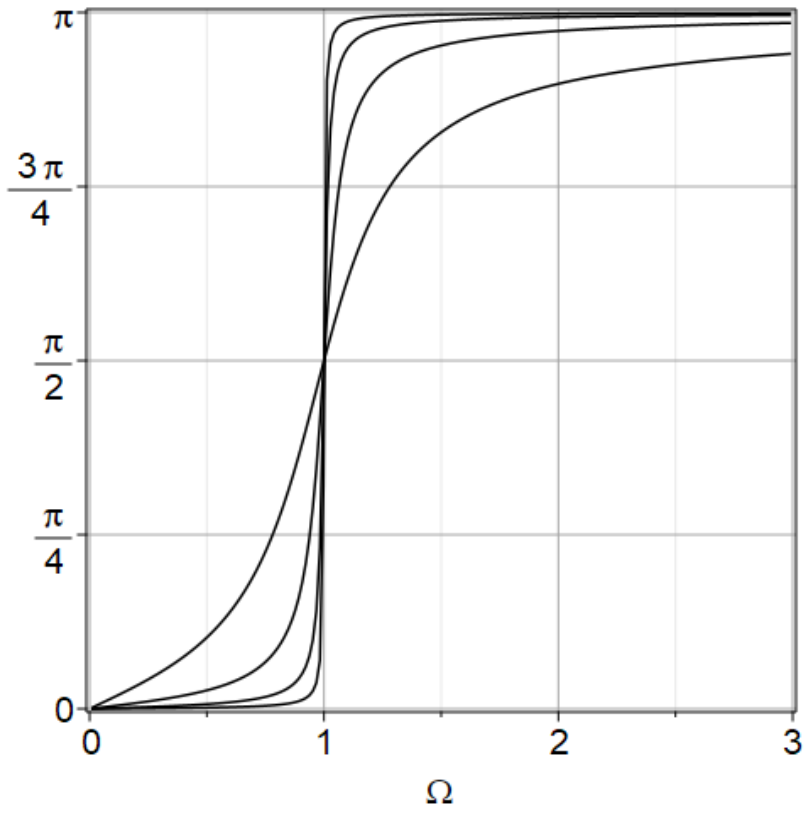

Figure 3: Phase $\arctan(2\Omega\beta, 1-\Omega^2)$ for different damping coefficients $\beta = 4^{-k}$ for $k = 1, 2, 3$, and 4, plotted on $0 \leq \Omega < 3$.

Cauchy smirks

The case $\beta = 0$ and $\Omega^2 = 1$ is not covered by those symbolic formulæ. One needs a secular term. See also RMC, D.J. Jeffrey, & D.R. Stoutemyer, Integrals of Functions Containing Parameters, Mathematical Gazette, vol. 104, no. 561, 2020, pp. 412–426.

A hybrid example

Cornelius Lanczos (1893–1974) is equally famous for work in relativity and work in computational mathematics. He was the one who first pointed out how useful Chebyshev polynomials were for computation*. He called his style of work "parexic analysis" (this name did not catch on). Let's solve $\ddot{y} + y = 0$ in his style. What will happen is that we will *actually* solve

$$\ddot{y} + y = \tau f(t) \tag{4}$$

for some small τ. Lanczos preferred integrating Chebyshev polynomials over differentiating them because there is a very short formula:

$$\int T_n(t)\,dt = \frac{1}{2}\left(\frac{1}{n+1}T_{n+1}(t) - \frac{1}{n-1}T_{n-1}(t)\right) \tag{5}$$

(Cauchy's observation applies if $n = 1$) but we can differentiate easily enough.

* John Von Neumann also pointed this out, and possibly did so first.

The tau method

Put $z = \sum_{k=0}^{5} c_k T_K(t)$, say. We then compute the residual $r = \ddot{z} + z$ as a sum of Chebyshev polynomials:

$$r = (c_0 + 4c_2 + 32c_4)\,T_0(t) + (c_1 + 24c_3 + 120c_5)\,T_1(t) + (c_2 + 48c_4)\,T_2(t) + (c_3 + 80c_5)\,T_3(t) + c_4 T_4(t) + c_5 T_5(t) \tag{6}$$

We set the first three of these to zero, and apply initial conditions $y(0) = 1$ and $\dot{y}(0) = 0$ (say) to make a square system of linear equations. The solution is

$$z = \frac{1}{209}\left(160T_0(t) - 20T_2(t) + T_4(t)\right) \tag{7}$$

and the residual is $\tau T_4(t)$ where $\tau = 1/209$.

That is, we have the *exact* solution of $\ddot{y} + y = \tau T_4(t)$ and, on $-1 \le t \le 1$ the residual is uniformly smaller than $1/209$, or less than 0.005.

The case $\beta = 0$ and $\Omega = 1$

Rewriting $\ddot{y} + y$ as $D^2 y + y$ with a differentiation matrix D appropriate for Chebyshev polynomials, and approximating $\cos t$ by a sum $C(t)$ of Chebyshev polynomials, we arrive at $(D^2 + I)y = C$ which is to be solved for y.

$$\begin{aligned} & 0.9852229794304 33 T_0(t) - 0.0195633539826686 T_2(t) \\ & - 0.00470352019800869 T_4(t) + 0.000082251026192 1204 T_6(t) \\ & - 5.60091400176361 \times 10^{-7} T_8(t) + 2.09251203145322 \times 10^{-9} T_{10}(t) \\ & - 4.98044654649350 \times 10^{-12} T_{12}(t) + 8.21482135308179 \times 10^{-15} T_{14}(t) \end{aligned} \tag{8}$$

This has residual error about double precision machine epsilon, and a similar forward error compared to $\cos(t) + t \sin(t)/2$.

Enter Chebfun

Nick Trefethen and his co-workers made this idea really practical, starting in 2001 with a first release in 2004 of the system Chebfun. Chebfun is (IMO) truly a system for hybrid computation: every object is expanded automatically into a piecewise Chebyshev approximation, which makes algebra and analysis both easy and effective.

Ten years later Sheehan Olver and Alex Townsend introduced the Julia package ApproxFun which uses piecewise rational approximations similarly.

Since the advent of the AAA algorithm, this approach is still undergoing rapid development and we should all be paying attention.

The AAA algorithm

The 2018 paper The AAA Algorithm for Rational Approximation by Nakatsukasa, Sète, and Trefethen, on the Adaptive Antoulas–Anderson algorithm, upended several long-held beliefs about rational approximation by being simple, robust, and efficient all at once. Even better methods are being developed now, but already as it was in the first paper, AAA is very impressive.

See also the 150 page (!) paper on Applications of AAA by Trefethen and Nakatsukasa. In spite of its size, the paper is extremely readable.

The basic idea is to play "whack-a-mole" with the largest error at any one stage. My Maple implementation takes 50ms to approximate $W(x)$ with $x = -\exp(-1)(1 - s^2/2)$ on $0 \le s \le \sqrt{2}$ to double precision; the algorithm in numapprox[minimax] in contrast requires working in 40 Digits and takes about 20 seconds. In the end the results are comparable.

Approximation of special functions by AAA may lead to much better support for special functions in PSE.

Some Results using AAA for Lambert *W*

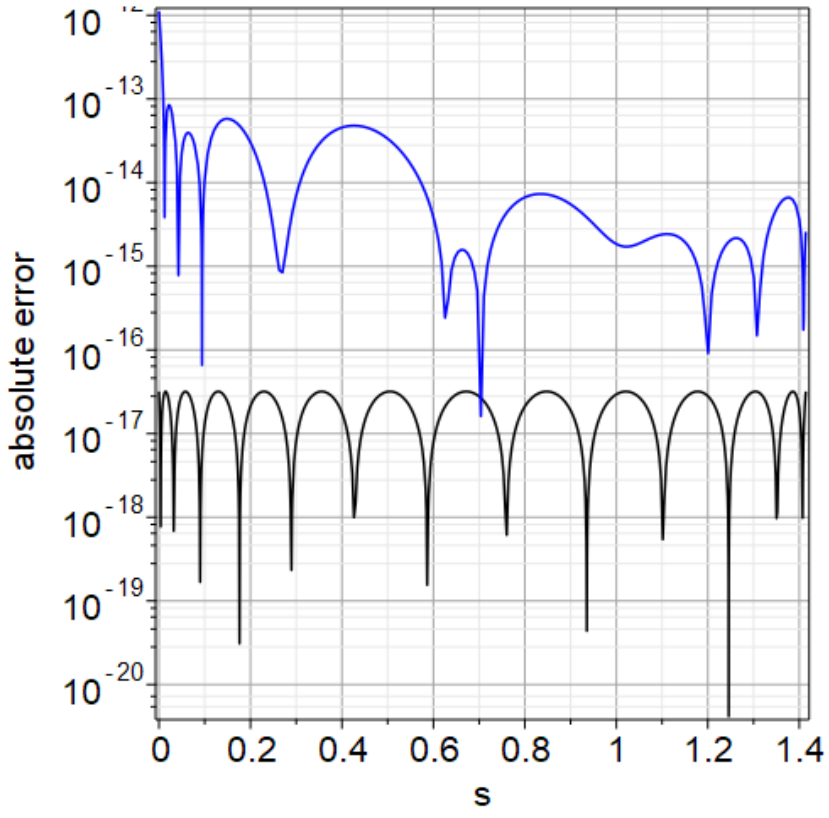

Figure 4: The error in the AAA approximation (blue) vs the best possible rational approximation of the same degree (black). Yes, the minimax approximant is better, but it took about a thousand times as long to compute (and required guessing how many Digits to use).

AAA is quite powerful

If

$$f(s) = \begin{cases} W_0(-e^{-1}(1 - s^2/2)) & 0 \le s \le \sqrt{2} \\ W_{-1}(-e^{-1}(1 - s^2/2)) & -\sqrt{2} < s < 0 \end{cases} \tag{9}$$

then $f(s) = -1 + s - s^2/3 + 11s^3/72 + \cdots$, a series which converges for $|s| < \sqrt{2}$ (has a log singularity at $-\sqrt{2}$), we can do better: AAA gets, nearly every time, rational approximations with errors about double precision all across $-\sqrt{2} < s < \sqrt{2}$ (almost: it uses weak poles to approximate the log singularity). This winds up approximating both W_0 and W_{-1}.

The minimax algorithm cannot do this at all. The log singularity defeats it.

One instance

After removing spurious poles and Froissart doublets (shades of approximate GCD!) we get

$$f(s) = a_0 + \sum_{i=1}^{17} \frac{a_i}{s - p_i} + \delta(s) \tag{10}$$

where $|\delta(s)| \leq 1.7 \times 10^{-12}$ on $-\sqrt{2} + O(10^{-3}) < s \leq \sqrt{2}$. The a_i range from about 251 for the farthest pole $p_1 \approx -56.8$ to $O(10^{-3})$ and $O(10^{-4})$ for the last six poles, which are all between -1.423 and $-\sqrt{2}$. $x = -\exp(-1)(1 - s^2/2)$ and the branch of W is the 0th branch if $s \geq 0$ and the -1st branch if $s \leq 0$.

Location of singularities

Going back to the first attempt: The function $W(x)$ has a branch cut on $x < -\exp(-1)$ and thus a branch point at $s = 0$ (remember $x = -\exp(-1)(1 - s^2/2)$). The AAA approximant automatically puts poles on that branch cut (on $s < -\sqrt{2}$, except one quite near to the negative of one of the nodes, which may be suboptimal). Similarly, if we instead approximate $W_{-1}(x)$, it chooses poles on $s > \sqrt{2}$ and thus $x < -\exp(-1)$.

We use a generalized eigenvalue problem (RMC, EACA 2004) to compute those poles, directly from the 2nd barycentric form of the AAA approximant. See Nakatsukasa and Trefethen, Acta Numerica 2025

A low-precision example

Using random sample points and a tolerance of 5.0×10^{-4} for AAA, and converting the barycentric form to partial fractions form to show the location of the singularities, we get (showing only the first five decimal places of each coefficient)

$$f(s) \approx 3.6178 - \frac{26.587}{s + 7.1294} - \frac{1.4799}{s + 1.9546} - \frac{0.17154}{s + 1.4823} - \frac{0.020090}{s + 1.4221} - \frac{0.0022369}{s + 1.4149} \tag{11}$$

and the error is (except just bigger than $-\sqrt{2}$) less than 1.5×10^{-3}. In contrast, a Padé approximant of the same degree has vastly better error near $s = 0$ and much worse error near the edges. If we approximate its coefficients, though, it's comparable (or better) for $s > -1$. Both may be used to start iterations for higher-precision values of W.

Adaptive Thiele Continued Fractions

There is an even newer method, lighter weight and nearly as powerful (just as powerful?): use the greedy algorithm to select interpolation points (to attack the "worst error" each time), in a Thiele Continued Fraction. This is work of Oliver Salazar Celis and Annie Cuyt, and Toby Driscoll and Yuxing Zhou. Oliver Salazar Celis took the three-term recurrences of the continuant and found a new pair of *tridiagonal* generalized eigenvalue problems to compute the poles and zeros.

Changing gears: “quasi” polynomials vs approximate polynomials

Schönhage’s paper on Quasi-GCD computations has a very careful notion of a polynomial with coefficients that are only imperfectly known, to some finite precision at any given moment, but which can be improved by asking an “oracle” for more digits. This is a kind of symbolic-numeric computation, but it doesn’t quite match many genuine applications in that in many situations, only a few digits are known, and some of them are wrong and there are no oracles to consult.

Approximate Polynomials

Instead in 1995 we introduced the idea of an approximate polynomial $f(z) = \sum_{k=0}^{N} c_k \phi_k(z)$ where the coefficients c_k had some uncertainty. We used bounds for error or relative error and not probability.

Then the GCD of f and g is better represented by asking "are there small perturbations of the coefficients for which the GCD is nontrivial?" This line of research has been quite productive; our algorithm was proved to be of polynomial time complexity by Karmarkar and Lakshman in 1996, for instance. Stetter and others examined the multivariate situation. Research continues to this day.

Special Functions and Spectral Methods

Expanding the unknown function in terms of special functions is an old technique, but revitalized by work arising from the Fast Multipole Method of Greengard and Rokhlin, which is arguably a symbolic-numeric method.

See also A fast and well-conditioned spectral method by Sheehan Olver and Alex Townsend (2013) and Riemann–Hilbert Problems, Their Numerical Solution, and the Computation of Nonlinear Special Functions (2015) by Tom Trogdon and Sheehan Olver. Perhaps also my own paper Special Functions in PSEs Maple Transactions Vol 3 No 2 (2023).

Hybrid computation in education

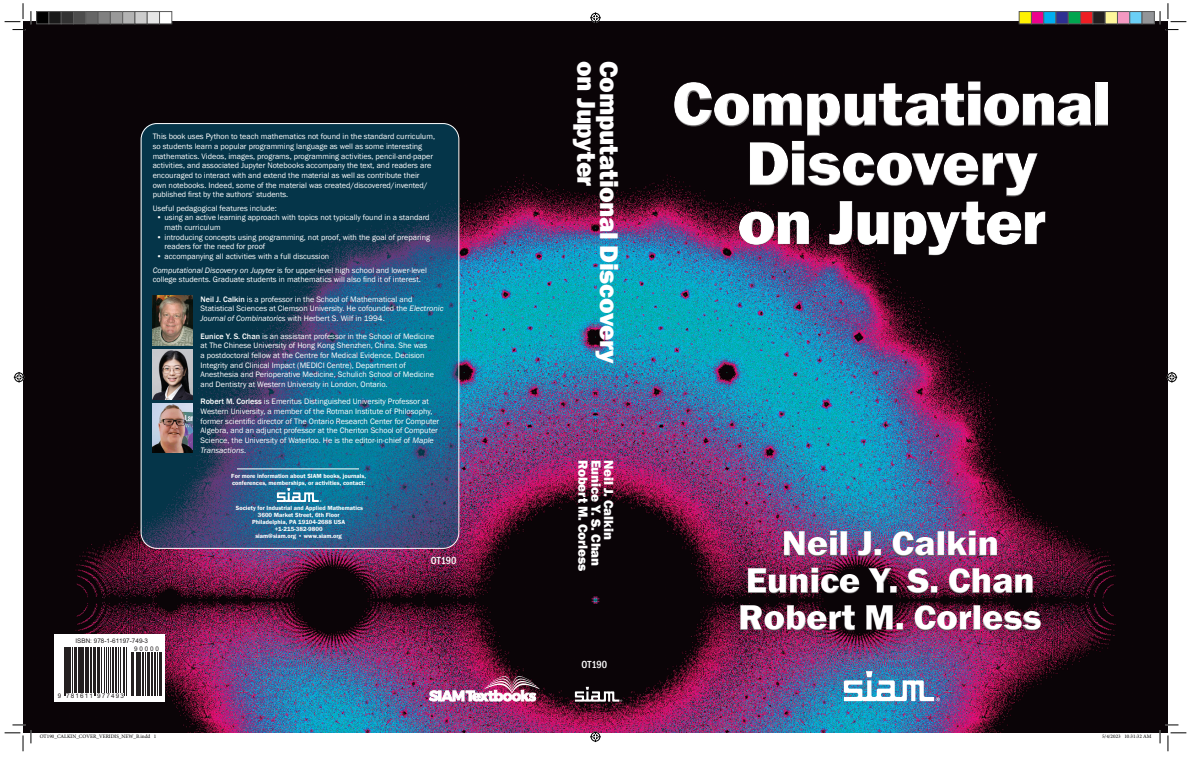

Figure 5: A 2023 book from SIAM: Calkin, Chan, & Corless, "Computational Discovery on Jupyter"

Hybrid computation in perturbation

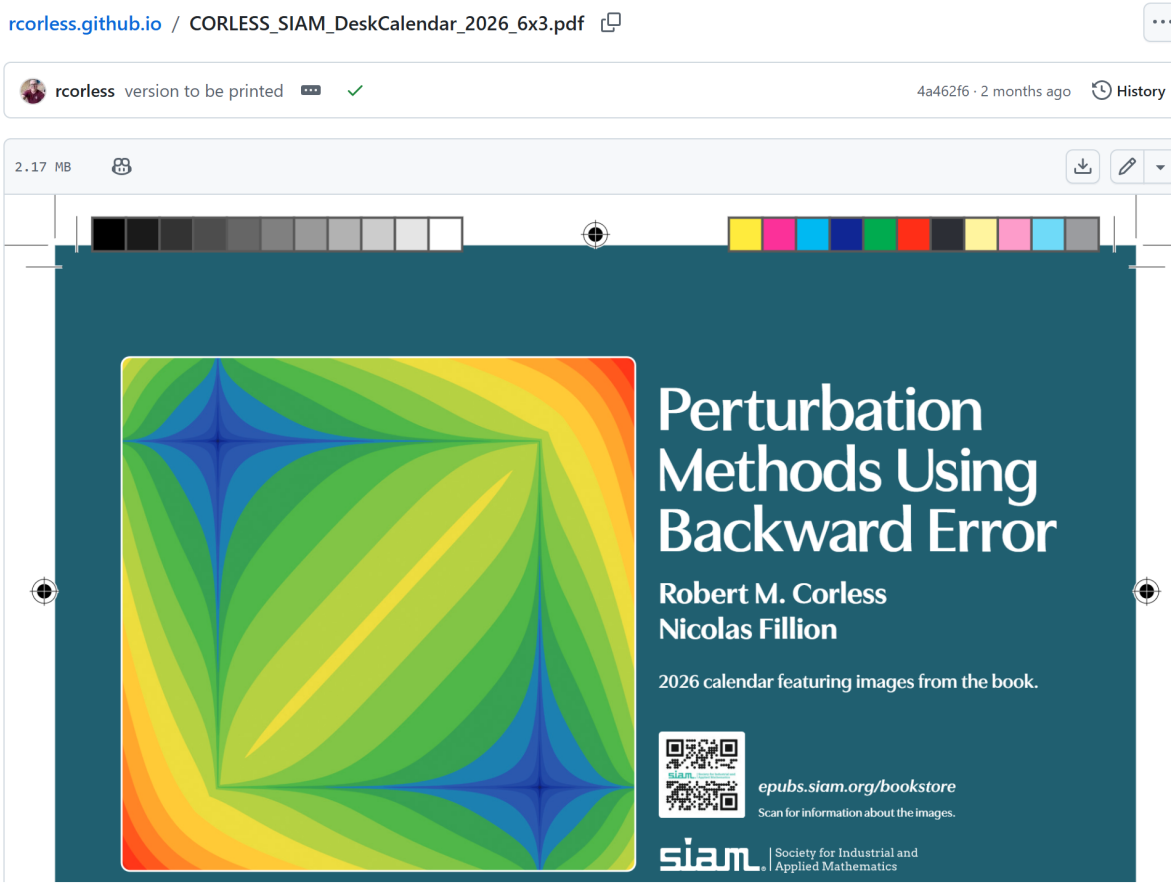

Figure 6: A Hybrid Symbolic-Numeric Book: Corless & Fillion "Perturbation Methods Using Backward Error", and a calendar with images from the book

So much more to say

Importing notions from analysis (e.g. a metric) into computer algebra is very productive. The idea of “backward error” involves more than just mathematics, because the problem context becomes important.

But I will stop here.

Thank you!

Happy to take questions!

This work was partially supported by NSERC grant RGPIN-2020-06438, and will continue to be supported by its extension for the next five years. I also thank my co-authors, all of them everywhere.