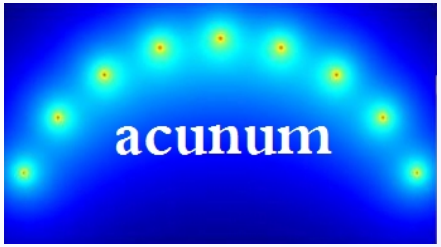
Numerical Laplace Transform Inversion Methods with Selected Applications

Patrick O. Kano

November 4, 2011



Acunum Algorithms and Simulations, LLC
Acute Numerical Algorithms And Efficient Simulations

Outline

This presentation is organized as follows:

I. Fundamental concepts and issues

1. basic definitions
2. relationship of numerical to analytic inversion
3. sensitivity and accuracy issues

II. Selected methods and applications

1. Weeks' Method – optical beam propagation & matrix exponentiation
2. Post's Formula – optical pulse propagation
3. Talbot's Method – matrix exponentiation with Dempster- Shafer evidential reasoning

III. Current work & future directions

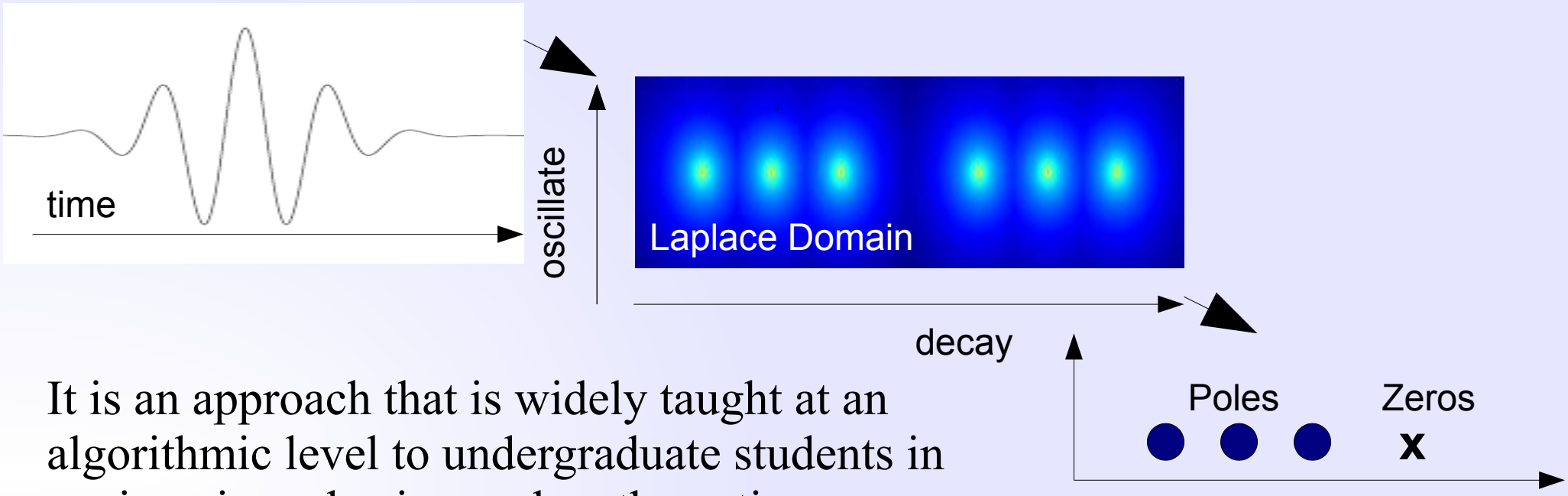
Basic Definitions

The Laplace Transform is tool to convert a difficult problem into a simpler one.

Difficult Time Dependent Problem	Solve Simpler Laplace Space Problem	Invert to a Time Dependent Solution

$$P\{F(t)\} \rightarrow L \rightarrow F(s) \rightarrow L^{-1} \rightarrow f(t)$$

It transforms a time dependent signal into its oscillating and exponentially decaying components.



It is an approach that is widely taught at an algorithmic level to undergraduate students in engineering, physics, and mathematics.

Laplace Transform Definitions

The forward Laplace transform is defined as an infinite integral over time (t).

$$F(s) = L(f(t))$$

$$F(s) = \int_0^\infty e^{-st} f(t) dt$$

Sufficient conditions for the integral's existence are that f(t) :

1. Is piecewise continuous

2. Of exponential order

There exist nonnegative σ and M, such that $|f(t)| \leq Me^{\sigma t}$ for all $0 \leq t$.

The Laplace transform can be viewed as the continuous analog of a power series.

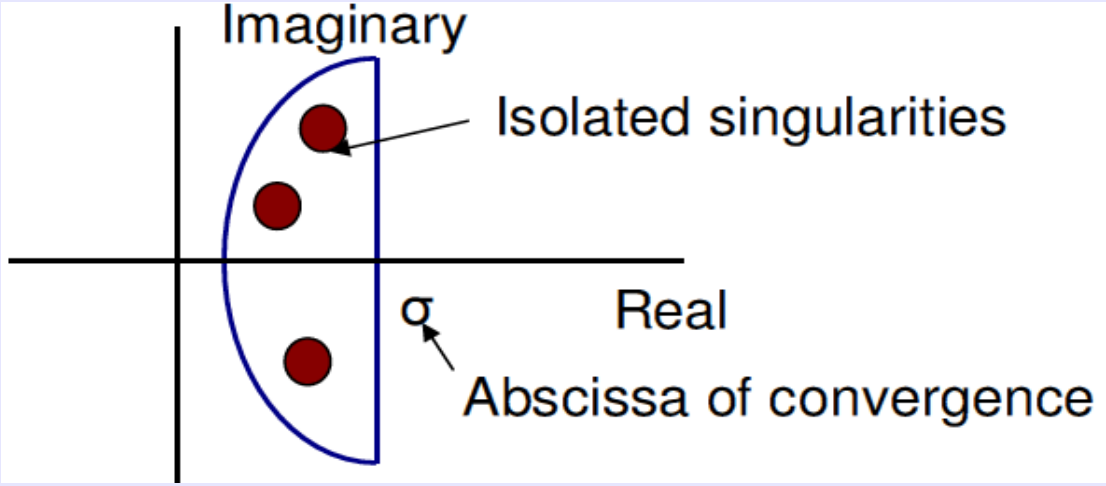
$$\sum_{n=0}^{\infty} a_n x^n \Rightarrow \begin{bmatrix} \sum \rightarrow \int \\ n \rightarrow t \\ a_n \rightarrow f(t) \\ x = e^{-s} \end{bmatrix} \Rightarrow \int_0^\infty f(t) e^{-st} dt$$

Inverse Laplace Transform Definitions

Analytic inversion of the Laplace transform is defined as an contour integration in the complex plane.

The *Bromwich* contour is commonly chosen.

$$f(t) = \frac{1}{2\pi i} \int_{\sigma - i\infty}^{\sigma + i\infty} F(s) e^{st} ds$$



For simple F(s), Cauchy's residue theorem can be employed.

$$\int_{\sigma - i\infty}^{\sigma + i\infty} F(s) e^{st} ds = 2\pi i \sum_j r_j$$

$$r_j = \frac{1}{(m-1)!} \lim_{s \to s_j} \frac{d^{m-1}}{ds^{m-1}} \left[(s - s_j)^m F(s) e^{st} \right]$$

→ f(t) is sum of the residues

For complicated F(s), this approach can be too cumbersome to perform even in symbolic software (Maple or Mathematica).

Numerical Laplace Transform Inversion

A numerical inversion approach is an obvious alternative.

How does one numerically invert a complicated F(s)?

We can alleviate some of the suspense at the very beginning by cheerfully confessing that there is no single answer to this question.

Instead, there are many particular methods geared to appropriate situations.

This is the usual situation in mathematics and science and, hardly necessary to add, a very fortunate situation for the brotherhood.

Richard Bellman
Numerical inversion of the Laplace transform: applications to biology, economics, engineering, and physics

The inversion integral is inherently sensitivity.

The exponential term leads to a large increase in the total error from even small numerical and finite precision errors.

There are multiple, distinctly different, inversion algorithms which are efficacious for various classes of functions.

Selected Numerical Inversion Methods

Of the numerous numerical inversion algorithms, my own research has focused on three of the more well known:

1. Weeks' Method
   - "Application of Weeks method for the numerical inversion of the Laplace transform to the matrix exponential", P. Kano, M. Brio, published 2009
   - "C++/CUDA implementation of the Weeks method for numerical Laplace transform inversion", P. Kano, M. Brio, Acunum white paper 2011
2. Post's Formula
   - "Application of Post's formula to optical pulse propagation in dispersive media", P. Kano, M. Brio, published 2010
3. Talbot's Method
   - "Dempster-Shafer evidential theory for the automated selection of parameters for Talbot's method contours and application to matrix exponentiation", P. Kano, M. Brio, P. Dostert, J. Cain, in review 2011

In the remaining slides, I introduce each of the algorithms and discuss my own applications.

Numerical Inversion Methods Timeline

The development of accurate numerical inversion Laplace transform methods is a long standing problem.

Post's Formula (1930)

- Based on asymptotic expansion (Laplace's method) of the forward integral
- Post (1930), Gaver (1966), Valko-Abate (2004)

Weeks Method (1966)

- Laguerre polynomial expansion method
- Ward (1954), Weeks (1966), Weideman (1999)

Talbot's Method (1979)

- Deformed contour method
- Talbot (1979), Weideman & Trefethen (2007)

Weeks' Method

The Weeks' method is one of the most well known algorithms for the numerical inversion of a Laplace space function.

It returns an explicit expression for the time domain function as an expansion in Laguerre polynomials.

$$f(t) = \lim_{N\to\infty} f_N(t)$$

$$f_N(t) = e^{\sigma t} \sum_{n=0}^{N-1} a_n e^{-bt} L_n(2bt)$$

$$L_n(x) = \frac{e^x}{n!}\frac{d^n}{dx^n}\left(e^{-x}x^n\right)$$

The coefficients $\{a_n\}$

1. contain the information particular to the Laplace space function
2. may be complex scalars, vectors, or matrices
3. time independent

Two free scaling parameters σ and *b*, must be selected according to the constraints that:

- *b>0 [Time scale factor]* ensures that the Laguerre polynomials are well behaved for large t
- $\sigma > \sigma_0$ *[Exponential factor]* at least as large as the abscissa of convergence

Laguerre Polynomials Expansion

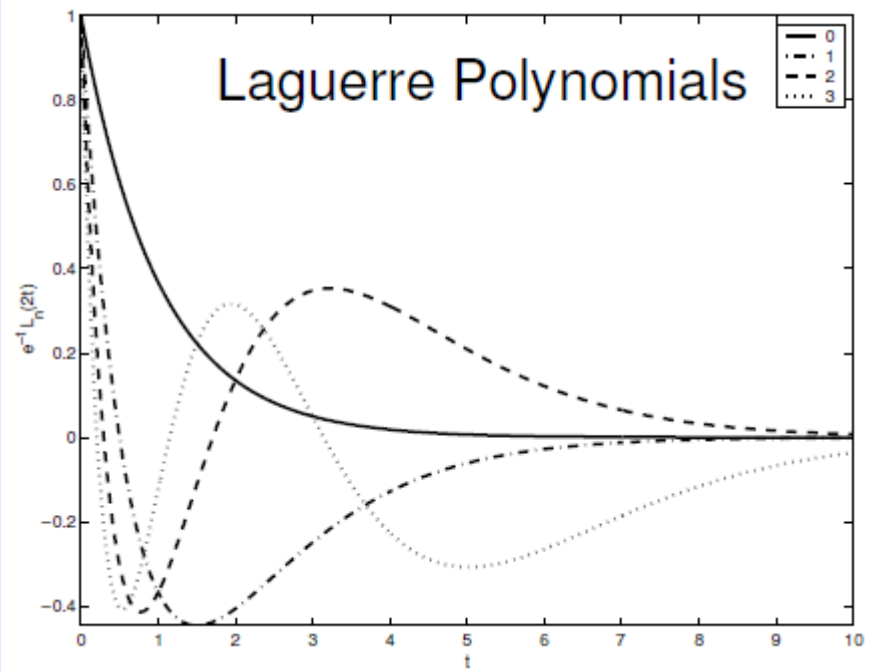
Weeks' contribution is an insightful algorithm for the coefficients.

The algorithm is based on the use of a Möbius transformation to remap the Bromwich line-contour to a circular contour.

The computation of the coefficients begins with a Bromwich integration in the complex plane.

$$s = \sigma + iy, \sigma > \sigma_0, -\infty < y < \infty$$

$$f(t) = \frac{e^{\sigma t}}{2\pi} \int_{-\infty}^{\infty} e^{iyt} F(\sigma + iy) dy$$



Assume the expansion

$$f(t) = e^{\sigma t} \sum_{n=0}^{\infty} a_n e^{-bt} L_n(2bt)$$

Equate the two expressions

$$\sum_{n=0}^{\infty} a_n e^{-bt} L_n(2bt) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{iyt} F(\sigma + iy) dy$$

Laguerre Polynomials Fourier Representation

Use the fact that
the weighted Laguerre polynomials have a nice Fourier representation:

$$e^{-bt}L_n(2bt) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{iyt}\frac{(iy-b)^n}{(iy+b)^{n+1}}dy$$

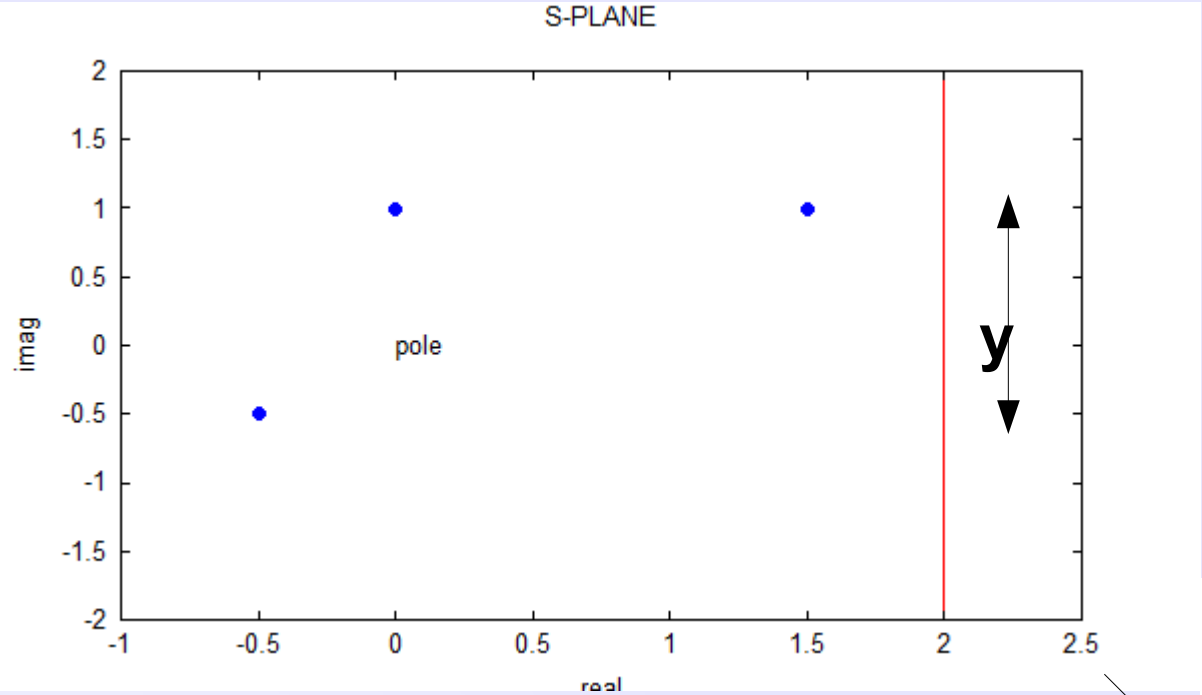
1. substitute
2. assume it is possible to interchange the sum and integral
3. equating integrands

$$\sum_{n=0}^{\infty} a_n e^{-bt}L_n(2bt) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{iyt}F(\sigma+iy)dy$$

$$\sum_{n=0}^{\infty} a_n \frac{(iy-b)^n}{(iy+b)^{n+1}} = F(\sigma+iy)$$

Almost a power series.

Möbius Transformation



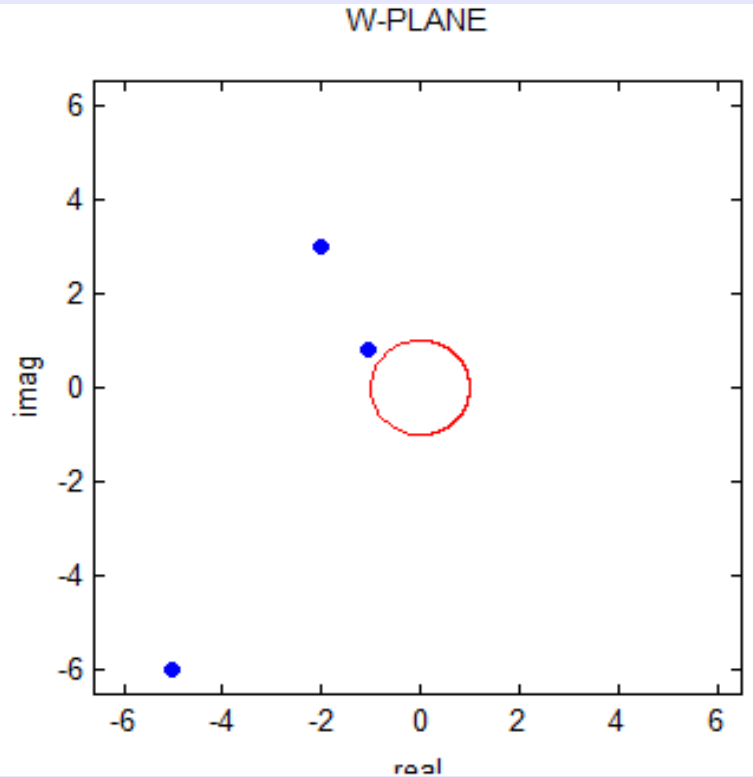
$$s = \sigma + iy$$

$$f(z) = \frac{az+b}{cz+d}$$

$$w = \frac{s-\sigma-b}{s-\sigma+b}$$

$$w = \frac{iy-b}{iy+b}$$

$$y = ib\frac{w+1}{w-1}$$



Isolated singularities of *F(s)* are mapped to the exterior of the unit circle in the w-plane.

Instead of integration on the y-line of *s*, *integrate on the circular contour* in *w*.

W-Plane Representation

With the change of variables, one obtains a **power series** in *w*.

$$\sum_{n=0}^{\infty} a_n w^n = \frac{2b}{1-w} F\left(\sigma - b\frac{w+1}{w-1}\right)$$

Radius of convergence is greater than 1.

The unit circle parametrized by θ as an integration path. $w = e^{i\theta}$

The coefficients are obtained by multiplying by both sides and integrating.

$$a_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{-in\theta}\frac{2b}{1-e^{i\theta}} F\left(\sigma - b\frac{e^{i\theta}+1}{e^{i\theta}-1}\right) d\theta$$

Integration is accurately estimated via the mid-point rule on the circle.

$$a_n \approx \frac{e^{-in\pi/(2M)}}{2M}\sum_{m=-M}^{M} e^{-in\theta_m}\frac{2b}{1-e^{i\theta_{m+1/2}}} F\left(\sigma - b\frac{e^{i\theta_{m+1/2}}+1}{e^{i\theta_{m+1/2}}-1}\right)$$

$$\theta = \frac{m\pi}{M}$$

Clenshaw Algorithm

Direct numerical Laguerre polynomial summation is not robust.

$$L_0(x) = 1 \; L_1(x) = 1 - x$$
$$(n+1)L_{n+1}(x) = (2n+1-x)L_n(x) - nL_n(x)$$

The backward Clenshaw algorithm can be used to perform the final sum.

Given

$$f(x) = \sum_{k=0}^{N} c_k F_k(x)$$
$$F_{n+1}(x) = \alpha_n(x)F_n(x) + \beta_n(x)F_{n-1}(x)$$

Define

$$y_{N+2} = y_{N+1} = 0$$
$$y_k = a_k(x)y_{k+1} + \beta_{k+1}(x)y_{k+2} + c_k$$

Solve back to y_1 and y_2

$$f(x) = c_0F_0(x) + y_1F_1(x) + \beta_1(x)y_2(x)F_0(x)$$

$$y_2 = 0; y_1 = a_N$$
$$\text{for } k = N-1 : -1 : 1$$
$$y_0 = \frac{2k-1-2bt}{k}y_1 + \frac{-k}{k+1}y_2 + a_k$$
$$y_2 = y_1; y_1 = y_0;$$
$$\text{end}$$
$$f(t) = e^{(\sigma-b)t}y_0$$

MATLAB

Weeks' Method Error Estimate

A straight forward error estimate yields three contributions:

1. Discretization (D_E) – Finite integral sampling
2. Truncations (T_E) – Finite number of Laguerre polynomials
3. Round-off (R_E) – Finite computation precision

- The integration on the circular w-space contour converges quickly.
- The discretization error can be neglected when compared to the truncation and round-off errors.

$$E_{total} \leq e^{\sigma t}(T_E + R_E)$$

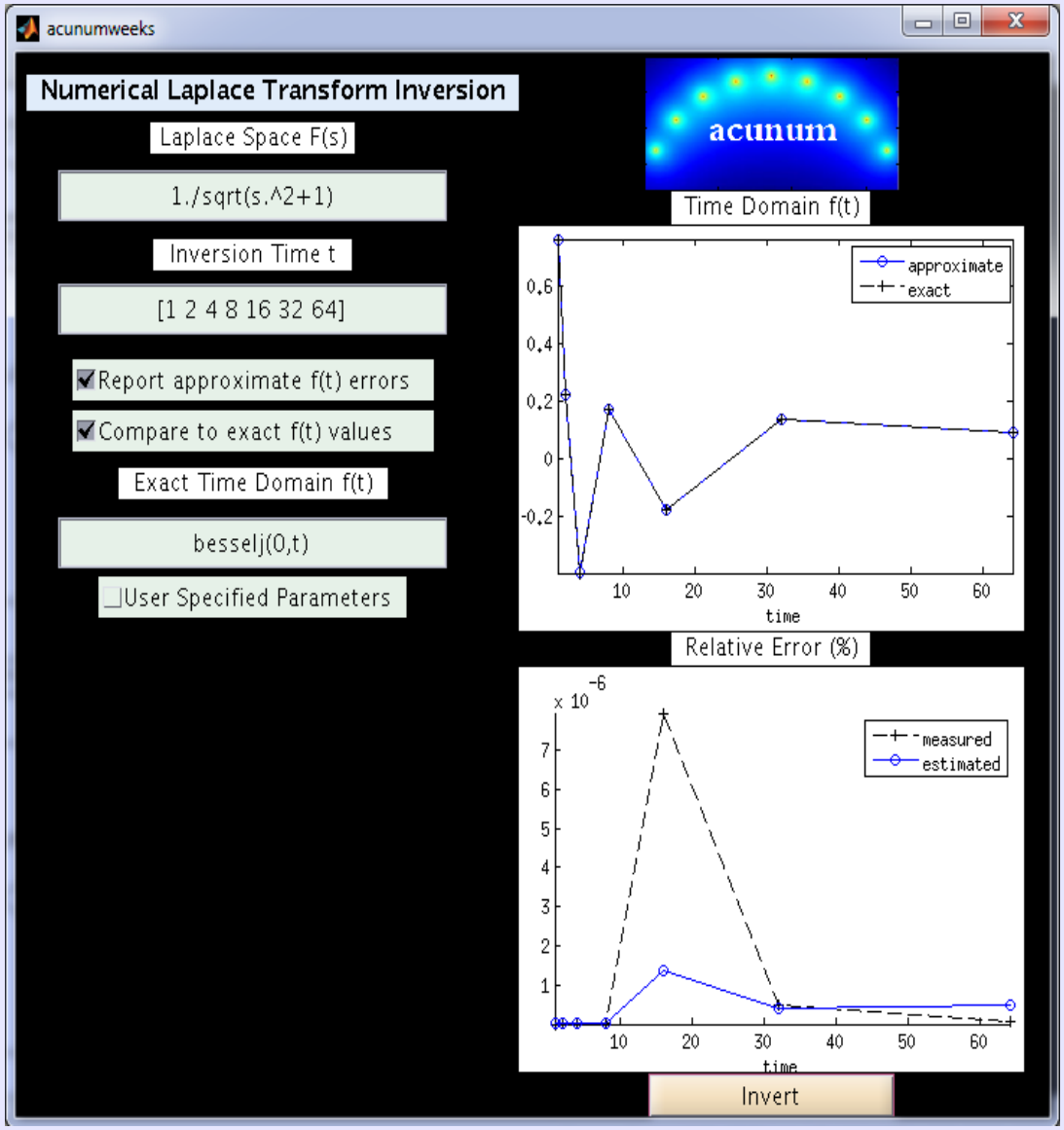
$$E_{total} \leq e^{\sigma t}\left(\sqrt{\sum_{n=N}^{\infty} ||a_n||^2} + \epsilon\sqrt{\sum_{n=0}^{N-1} ||a_n||^2}\right)$$

$$||a_n||^2 \leq \frac{||F(s)||^2}{r^n}$$

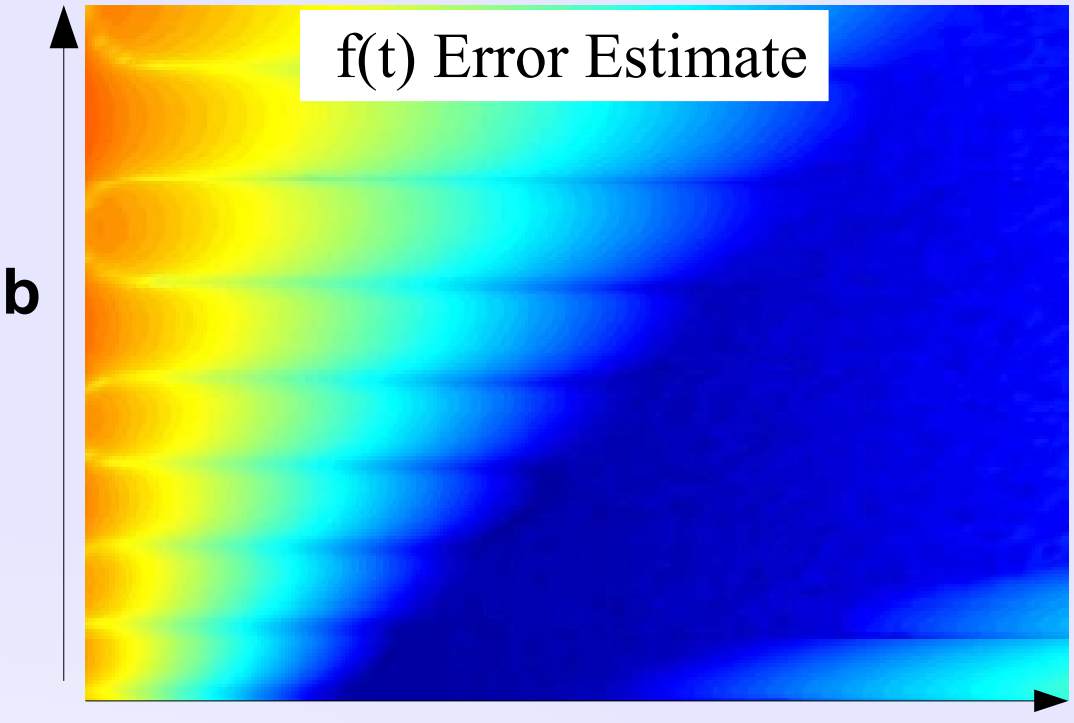
Weeks Method GPU Accelerated Tool

Acunum has posted to the MATLAB file exchange an implementation of the Weeks method.

- The codes are freely available under a [BSD] license in:
  - MATLAB using JACKET from AccelerEyes, Inc. on the MATLAB file exchange
  - C/C++ on the Acunum website
- The tool includes a GUI 'acunumweeks' or can be run from the MATLAB environment.
- 'acunumweeks' tool relieves the user of the very difficult problem of choosing optimal method parameters (σ,b).

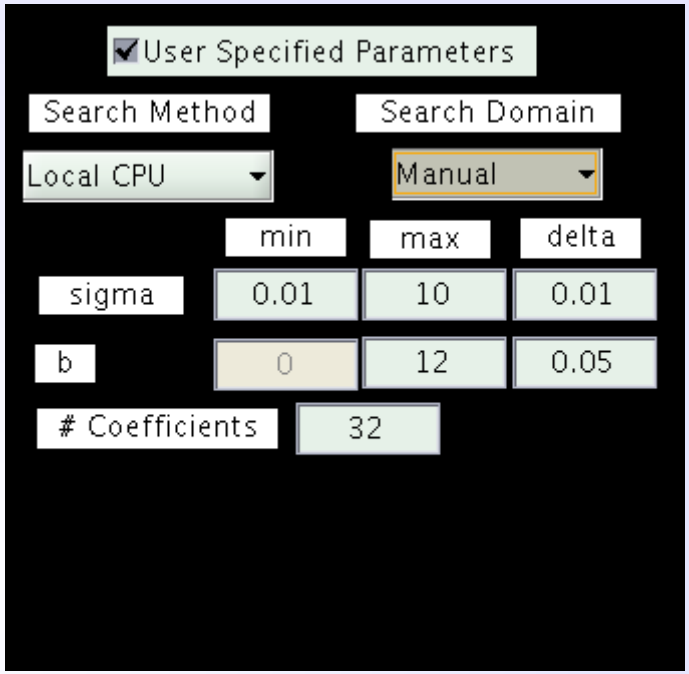


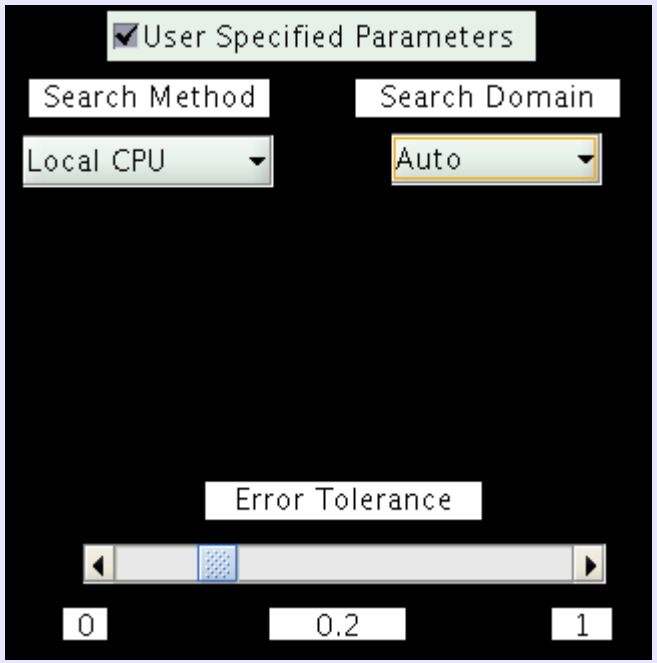
Weeks Method GPU Accelerated Tool



- Fully automated but with flexibility for users to control parameters.
- Minimizes an error estimate to obtain optimal (σ,b) parameters.
- Uses graphics processing unit [GPU] parallelization to quickly perform a global minimization.

Manual (σ,b) Ranges



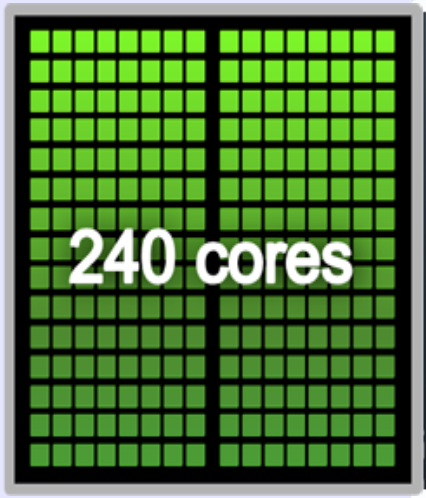


Auto (σ,b) Ranges

Technology behind Acunum Applications

- General purpose graphics processing unit [GPGPU] computing is the application of the parallelism in GPU technology to perform scientific or engineering computations.
- Arithmetically intense algorithms are often orders of magnitude ***faster*** on a GPU than a CPU.

Central Processing Unit [CPU]



Graphics Processing Unit [GPU]

Technology behind Acunum Applications

Acunum offers C++ & MATLAB tools that perform computations on NVIDIA graphics processors.

MATLAB

Dominant environment for scientific computing and algorithm development.

Interfaces are being developed for GPU processing in MATLAB.

NVIDIA Graphics Processing Units

Low cost and widely distributed graphics processors

Compute Unified Device Architecture [CUDA] allows for general purpose [GPGPU] on NVIDA products.

ACUNUM Software

Technology behind Acunum Applications

NVIDIA
Graphics Processing Units

ACUNUM
Matlab/Jacket & C/C++
Numerical
Laplace Transform Inversion
Toolbox

Acunum released a numerical inversion tool to the web for public use.

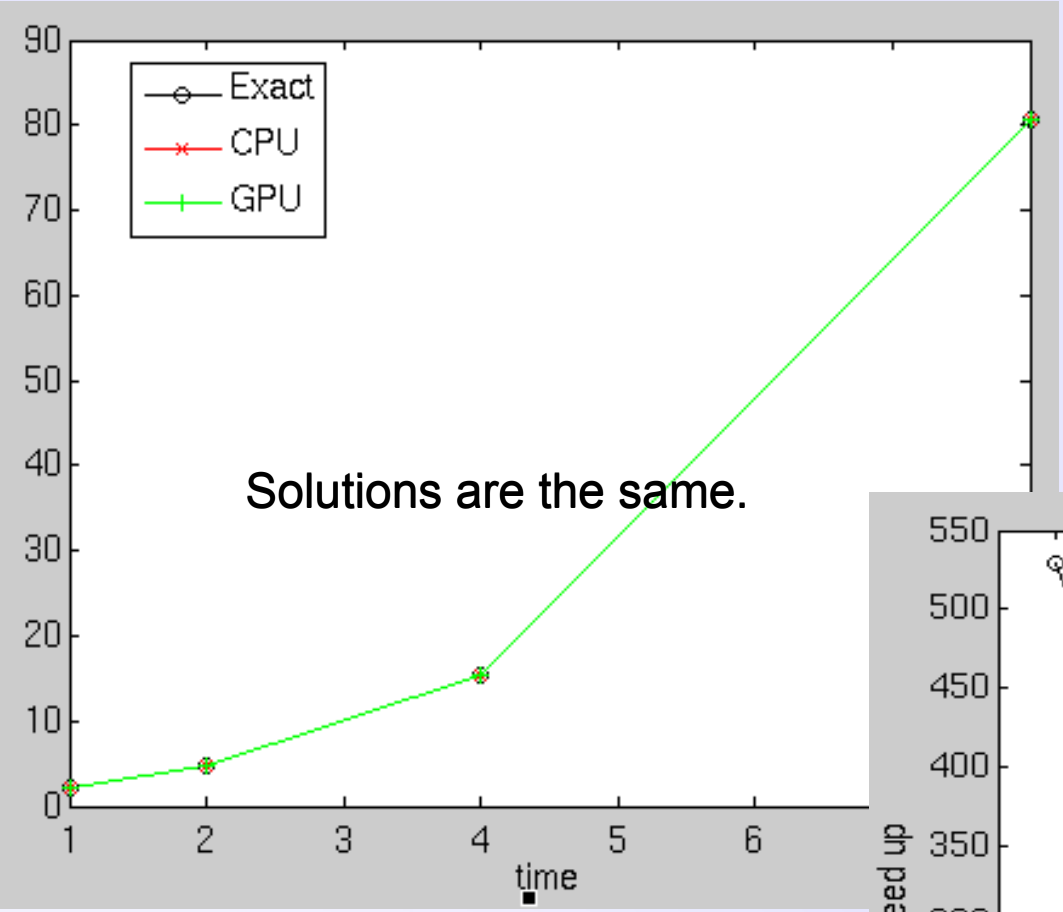
ACUNUM
C/C++
Dempster-Shafer
Data Fusion

Acunum is developing a fast GPU accelerated algorithm for sensor data fusion and object classification.

http://www.accelereyes.com/examples/academia

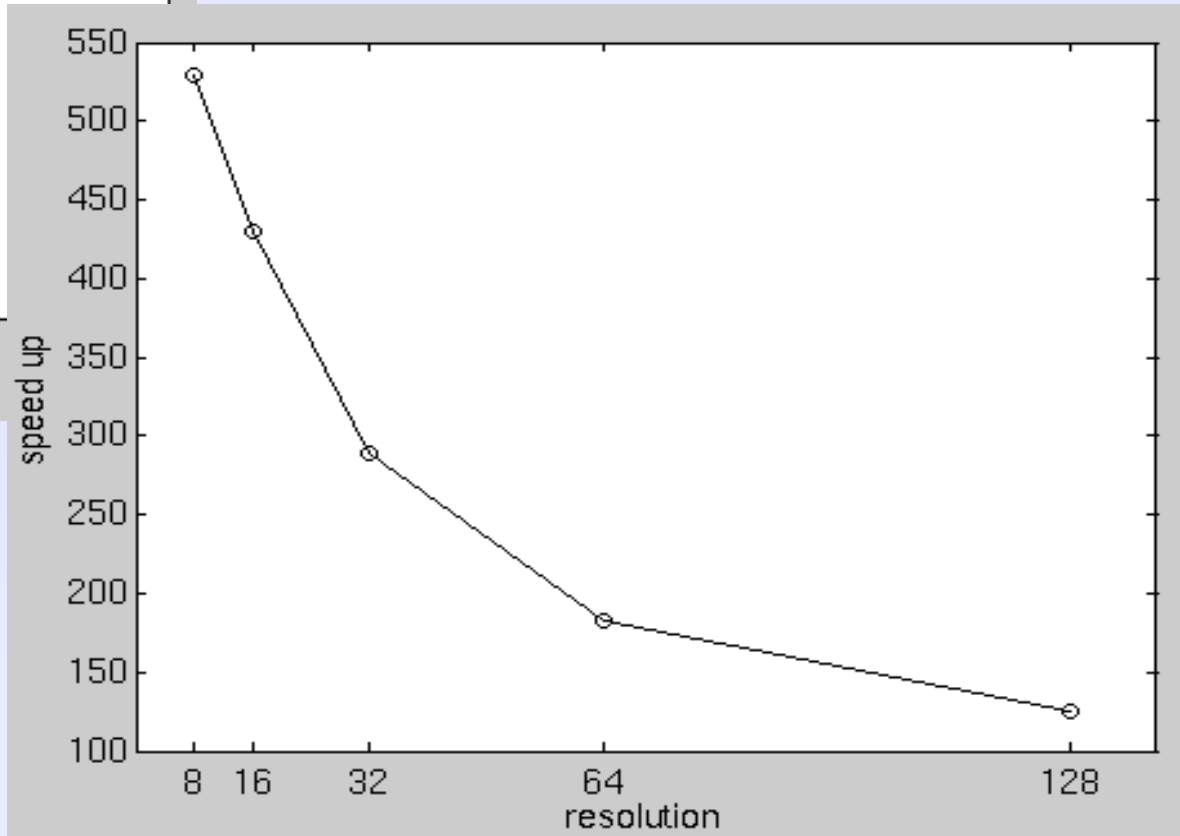
Our tools uses JACKET from AccelerEyes, Inc. for the MATLAB/GPU interface.

Acunum Laplace Transform Toolbox



Solutions are the same.

The time to run the global search using the graphics processor is a fraction of the time of that using the main central processor.



Weeks Method Matrix Exponential

Matrix Exponential =
Inverse Laplace Transform of
the Resolvent Matrix $(sI-A)^{-1}$

$$e^{At} = \frac{1}{2\pi i} \int [sI - A]^{-1} e^{st} ds$$

It is method #12 in the SIAM reviews on matrix exponentiation:

- "Nineteen Dubious Ways to Compute the Exponential of a Matrix", SIAM Review 20, Moler & Van Loan, 1978.
- "Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later", SIAM Review 45, Moler & Van Loan, 2003.

The Pade' -scaling-squaring method (#3) is a commonly used alternative (MATLAB expm).

$$e^M = \left(e^{M/2n}\right)^{2n} = \left(e^B\right)^{2n}$$

$$e^B \approx \frac{N_{pq}(B)}{D_{pq}(B)}$$

Pade' approximations are useful to compare with the Laplace transform values.

Optical Beam Propagation Equation

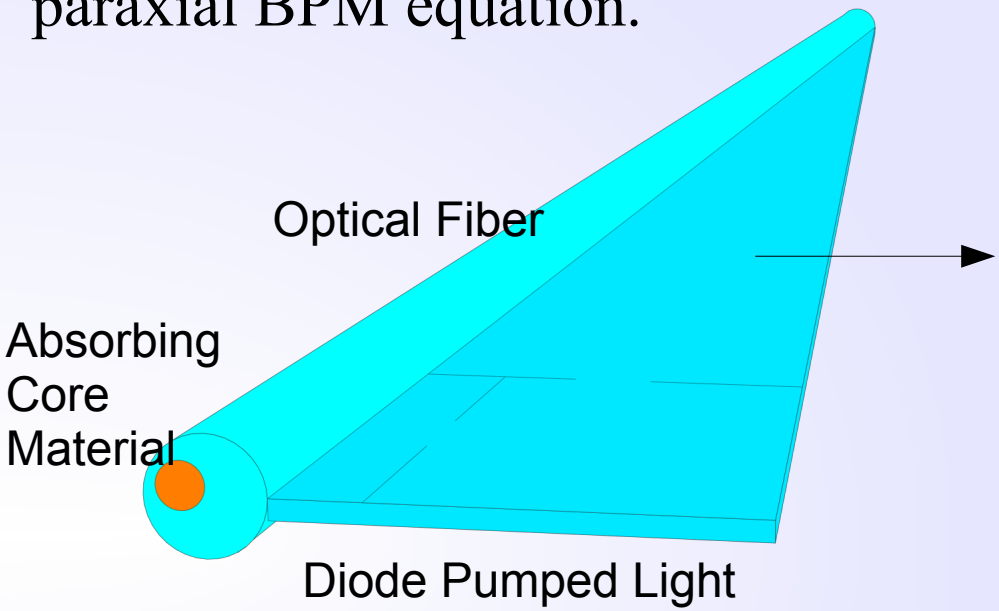
The matrix exponential work was motivated by the desire to accurately solve the **non-paraxial** optical beam propagation method [BPM] equation.

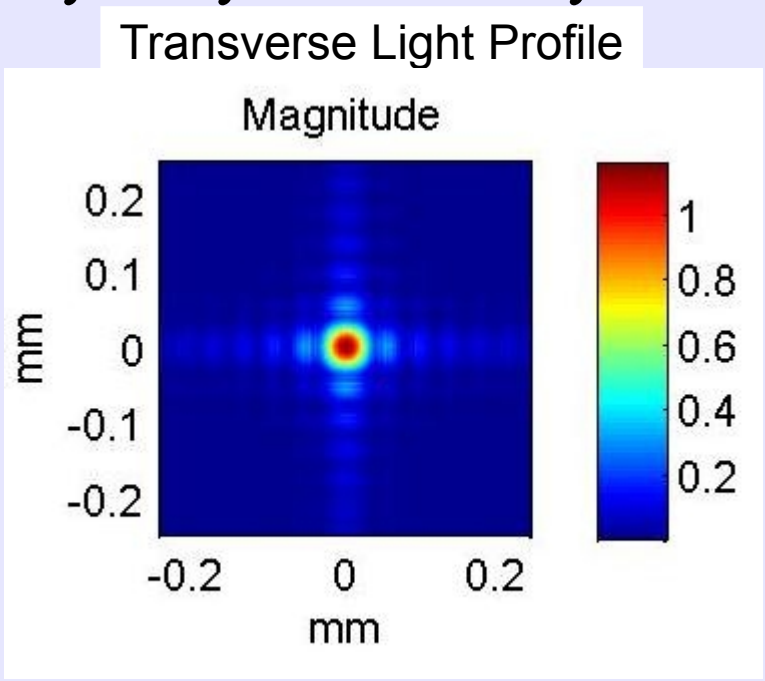
$$\frac{\partial u}{\partial z} = i\beta u - i\sqrt{(\nabla^2 + k_0^2 n^2)u}$$

u = scalar component of the electric field

The equation describes the propagation of an optical beam through an object with spatially dependent refractive index n(x,y,z).

The square root is commonly approximated by a Taylor-series to yield the paraxial BPM equation.





GPU Accelerated BPM

Beam Propagation is typically performed using a split-step method:
a) Finite difference alternating direction implicit method [ADI]
b) Fourier transform [FFT] in space.

Optical Beam Propagation Equation

Alternative - Laplace-Transform Weeks method BPM approach

Spacial discretization in the transverse direction yields a set of ODEs:

$$\frac{d\vec{u}}{dz} = i\beta\vec{u} - i\sqrt{(D^2 + k_0^2 n^2)}\vec{u}$$

Laplace Transform in Z

$$\hat{u} = \left[sI - i\beta(I - \sqrt{I + A})\right]^{-1}\vec{u}_0$$

$$A = \frac{D + k_0^2\bar{n}^2 - \beta^2 I}{\beta^2}$$

A general Weeks method matrix exponential code was written:
www.math.arizona.edu/~brio/

The Laplace space function is of a matrix exponential.

$$\hat{u}(s) = [sI - M]^{-1}\vec{u}(0)$$

$$M = i\beta(I - \sqrt{I + A})$$

Weeks Method Based Matrix Exponentiation

Direct matrix inversions are considerably slower than solving the system of linear equations. → 2 implementations

$$e^{i\beta(I-\sqrt{I+A})t} \text{ or } e^{i\beta(I-\sqrt{I+A})t}\vec{u}_0$$

This code considered 3 approaches to choose the parameters (σ,b):

1. Weeks' original suggestions

$$\sigma = max\,(0, \sigma_0 + 1/t)$$

$$b = \frac{2N}{t} \text{ where } N = \text{ expansion coefficients}$$

2. Error-Estimate Motivated Approach
   a) Min-Max: minimize truncation error only/maximize the radius of convergence as a function of σ and b
   b) Weideman: minimize the total error estimate as a function of σ and b

$$E_{total} \leq e^{\sigma t}\left(\sqrt{\sum_{n=N}^{\infty} ||a_n||_F^2} + \epsilon\sqrt{\sum_{n=0}^{N-1} ||a_n||_F^2}\right)$$

$$||a_n||_F^2 \leq \frac{||(sI-M)^{-1}||_F^2}{r^n}$$

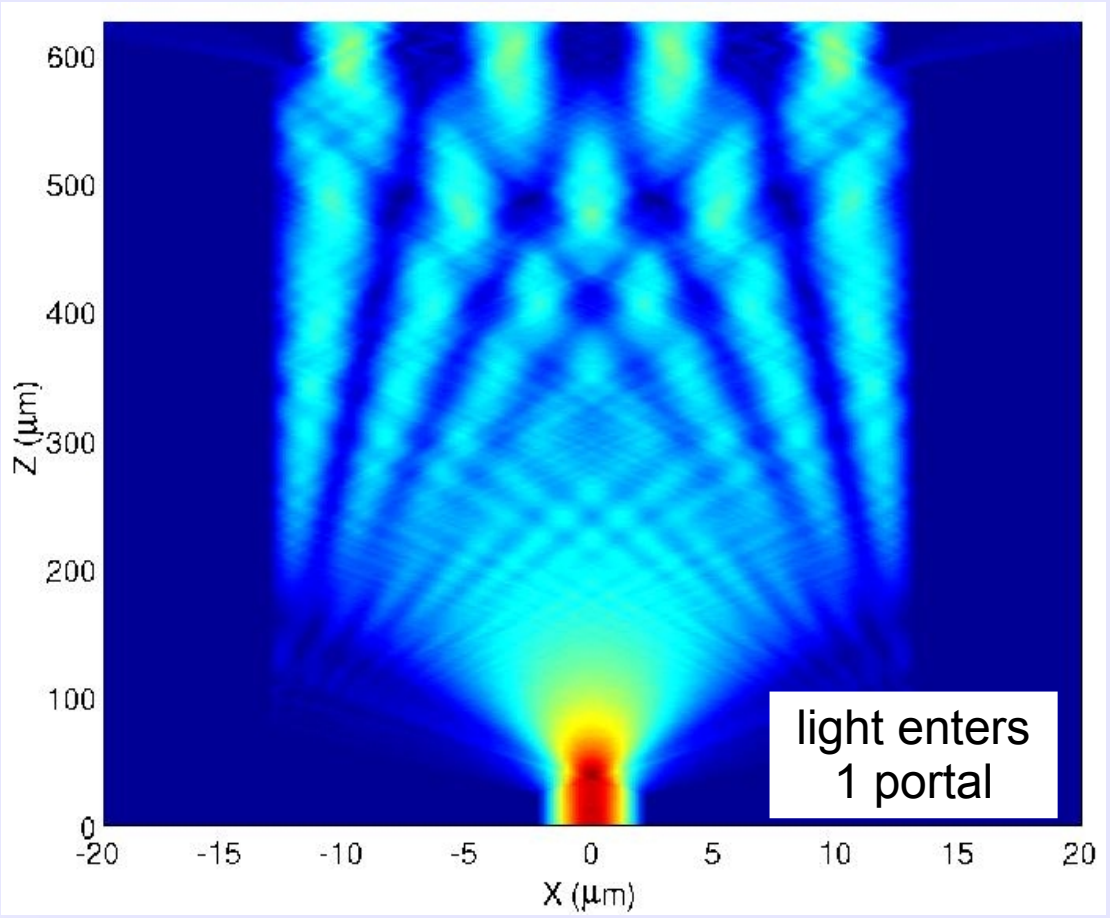
Weeks Method for BPM

	σ	b	Maximum Absolute Error
Weeks	**1**	**32**	**12.28**
Min-Max	**20**	**10**	**0.00119**
Weideman	**11.84**	**16.79**	**0.000425**

- Weideman's Method of minimizing the truncation and round-off errors works impressively.

- It is the approach in both the matrix exponential Weeks code & the scalar inversion tool on the file exchange.

By proper selection of coupler dimension, the light exits the 4 portals.



Multi-mode interference coupler simulated with Weeks-BPM

Post's Formula

Provides an alternative to the Bromwich contour:

$$f(t) = \lim_{q\to\infty} \frac{(-1)^q}{q!}\left(\frac{q}{t}\right)^{q+1}\left(\frac{d^q F(s)}{ds^q}\right)|_{s=q/t}$$

1. Only parameter is the maximum derivative order.
2. The inversion is performed using:
   - Only real values for *s*
   - Without prior knowledge of singularities

Emil Post

$$F(s) = \frac{1}{s^2} \Rightarrow \text{2nd order pole } s_0$$

$$L^{-1}(F(s)) = \frac{1}{(2-1)!}\lim_{s\to 0}\frac{d^{(2-1)}}{ds^{(2-1)}}\left[s^2\frac{e^{st}}{s^2}\right] = \lim_{s\to 0} te^{st} = t$$

Method of Residues

Post's Formula

$$\frac{d^q F(s)}{ds^q} = (-1)^q(q+1)!s^{-(q+2)}$$

$$L^{-1}(F(s)) = \lim_{q\to\infty}\frac{(-1)^q}{q!}\left(\frac{q^{q+1}}{t^{q+1}}\right)\cdot(-1)^q(q+1)!\left(\frac{t^{q+2}}{q^{q+2}}\right)$$

$$= \lim_{q\to\infty}(q+1)\frac{t}{q} = t$$

Post's Formula

Laplace's method for the asymptotic expansion of the forward integral
→ Post's Formula

if $\frac{dg}{d\tau}|_{\tau_0} = 0$ and if $\frac{d^2g}{d\tau^2}|_{\tau_0} \equiv \beta \neq 0$

then as $k \to \infty$, the integral

$$I(k) = \int_0^\infty e^{-kg(\tau)} h(\tau) d\tau$$

has the asymptotic behaviour

$$I(k, \tau_0) \sim \sqrt{\frac{2\pi}{k\beta}} e^{-kg(\tau_0)} h(\tau_0)$$

$$F(s) = \int_0^\infty e^{-s\tau} f(\tau) d\tau$$

Take the k^{th} derivative with respect to s

$$F^k(s) = (-1)^k \int_0^\infty \tau^k e^{-s\tau} f(\tau) d\tau$$

Rearrange and evaluate at $s = \frac{k}{t}$

$$(-1)^k F^k\left(\frac{k}{t}\right) = \int_0^\infty e^{k(\ln\tau - \tau/t)} f(\tau) d\tau$$

Assign $\tau_0 = t, h(\tau) = f(\tau)$,

$$g(\tau) = \tau/t - \ln(\tau), \frac{d^2g}{d\tau^2}|_{\tau_0} = t^{-2}$$

and apply Laplace's method to yield

$$(-1)^k F^k\left(\frac{k}{t}\right) \sim \sqrt{\frac{2\pi t^2}{k}} e^{-k(1-\ln t)} f(t)$$

$$\frac{k}{t^{k+1}} (-1)^k F^k(\frac{k}{t}) \sim \sqrt{2\pi k} e^{-k} f(t)$$

With Stirling's formula

$$k! = \sqrt{2\pi k} e^{-k} k^k \left(1 + \frac{1}{12k} + \frac{1}{288k^2} + \dots\right)$$

$$f(t) \sim \frac{(-1)^k}{k!} \left(\frac{k}{t}\right)^{k+1} F^k\left(\frac{k}{t}\right) \left(1 + \frac{1}{12k} + \frac{1}{288k^2} + \dots\right)$$

Practical Application Impediments

1. Errors are amplified by a factor that grows quickly with q.

$$f(t) = \lim_{q\to\infty} \frac{(-1)^q}{q!}\left(\frac{q}{t}\right)^{q+1}\left(\frac{d^q F(s)}{ds^q}\right)|_{s=q/t}$$

2. The method converges slowly.

$$|f(t) - f_q(t)| \sim 1/q$$

For example:

$$F(s) = \frac{1}{s^2}$$

$$f_q(t) = \frac{(q+1)}{q}t$$

$$\text{absolute error} = t(q+1)/q - t = t/q$$

$$\text{relative error} = 1/q$$

3. Expressions (or approximations) for the higher order derivatives of F(s) are required.

Gaver-Post Functionals

- Finite differences commonly used to approximate the high order derivatives in Post's formula.

$$f_q(t) = (-1)^q q \frac{ln(2)}{t} \binom{2q}{q} \triangle^q F\left(q\frac{ln(2)}{t}\right)$$

$$\triangle F(nx) = F((n+1)x) - F(nx)$$

Gaver-Post Formula 1966

$$f_q(t) = q\frac{ln(2)}{t}\binom{2q}{q}\sum_{j=0}^{q}(-1)^j\binom{q}{j}F\left((q+j)\frac{\ln(2)}{t}\right)$$

$$f(t) = \lim_{q\to\infty} f_q(t)$$

- The 'Gaver functionals' can be computed recursively to yield the approximate inverse:

$$G_0^{(n)} = n\frac{\ln 2}{t}F\left(n\frac{\ln 2}{t}\right) \quad 1 \le n$$

$$f_q(t) = G_q^q$$

$$G_p^{(n)} = \left(1+\frac{n}{p}\right)G_{p-1}^{(n)} - \left(\frac{n}{p}\right)G_{p-1}^{(n+1)} \quad 1 \le p, p \le n$$

Post's Formula Rate of Convergence

- Series acceleration (sequence transform) methods are used to mitigate the slow rate of convergence.
- A comparative study has shown the utility of the Wynn-ρ algorithm.

$$f(t) \approx \rho_{N-2}^{0}$$

where ρ is an element of the matrix

$$\begin{array}{llllll}
\rho_{-1}^{0} = 0 & \rho_{0}^{0} = \phi_0 & \rho_{1}^{0} & \rho_{2}^{0} & \cdots & \rho_{N-2}^{0} \\
\rho_{-1}^{1} = 0 & \rho_{0}^{1} = \phi_1 & \rho_{1}^{1} & \rho_{2}^{1} & \cdots & \rho_{N-2}^{1} \\
\rho_{-1}^{2} = 0 & \rho_{0}^{2} = \phi_2 & \rho_{1}^{2} & \rho_{2}^{2} & & \\
\vdots & \vdots & \vdots & \vdots & & \\
\vdots & \vdots & \vdots & \rho_{2}^{N-3} & & \\
\vdots & \vdots & \rho_{1}^{N-2} & & & \\
\rho_{-1}^{N-1} = 0 & \rho_{0}^{N-1} = \phi_{N-1} & & & &
\end{array}$$

The Post approximations are in the second column.
For even N , the elements are computed from the rule:

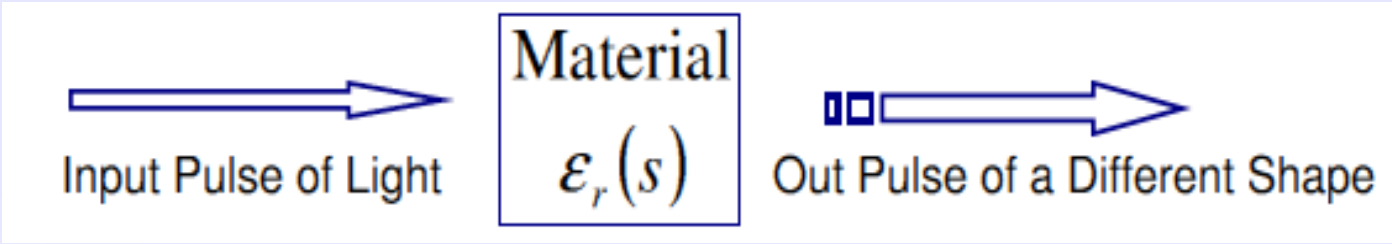
$$\rho_{k+1}^{n} = \frac{k+1}{\rho_{k}^{n+1} - \rho_{k}^{n}} + \rho_{k-1}^{n+1}$$

$$k = 0, \ldots, N-3$$

$$n = 0, \ldots, N-1-k$$

Application to Optical Pulse Propagation

NSF Grant ITR-0325097
An Integrated Simulation Environment for High-Resolution Computational Methods in Electromagnetics with Biomedical Applications
Moysey Brio, et. al.



Problem

Biological materials often have a dielectric constant which is a nontrivial function of wavelength.

Goal

Generate a numerical method for rapid computation of the
- distribution of an initial optical pulse
- in a fixed dielectric medium
- with a nontrivial material dispersion relation.

Concept

Create databases of pre-computed tables which can be used by devices which must operate in real-time.

Laplace Transforms for Maxwell's Equations

Laplace transforms for optical pulse propagation in dispersive media is a well known application:

Maxwell's Equations

$$\nabla \cdot D(\vec{x},t) = 0$$
$$\nabla \cdot H(\vec{x},t) = 0$$
$$\nabla \times H(\vec{x},t) = \frac{\partial D(\vec{x},t)}{\partial t}$$
$$\nabla \times E(\vec{x},t) = -\mu_0 \frac{\partial H(\vec{x},t)}{\partial t}$$

electric field strength $E(\vec{x},t)$
and the electric displacement $D(\vec{x},t)$
related by a temporal convolution,
the polariziation $P(\vec{x},t)$

$$D(\vec{x},t) = \epsilon_0 E(\vec{x},t) + P(\vec{x},t)$$
$$P(\vec{x},t) = \epsilon_0 \int_0^t \Phi(t-\tau) E(\vec{x},\tau) d\tau$$

In the Laplace space, the convolution and derivatives become multiplications.

$$D(\vec{x},s) = \epsilon_0 (1+\phi(s)) E(\vec{x},s)$$
$$\nabla \cdot E(\vec{x},s) = 0$$
$$\nabla \cdot H(\vec{x},s) = 0$$
$$\nabla \times H(\vec{s},s) = s\epsilon_0 (1+\phi(s)) E(\vec{x},s) - \epsilon_0 E(\vec{x},t=0)$$
$$\nabla \times E(\vec{x},s) = -\mu_0 (sH(\vec{x},s) - H(\vec{x},t=0))$$

Laplace Transforms for Maxwell's Equations

Maxwell's equations can now be concisely stated in a form which involves only the electric field:

$$s^2\epsilon_r(s)E(\vec{x},s) - c^2\nabla^2 E(\vec{x},s) = sE(\vec{x},0) + \frac{\partial E}{\partial t}(\vec{x},0) \equiv V(\vec{x},0)$$

$$\epsilon_r(s) = \epsilon_0\left(1+\phi(s)\right)$$

Applying the Fourier transform in space, yields a compact joint Fourier-Laplace solution:

$$E(\vec{k},s) = \frac{V(\vec{k},s)}{s^2\epsilon_r(s) + c^2|\vec{k}|^2}$$

$$= \frac{s}{s^2\epsilon_r(s) + c^2|\vec{k}|^2}E(\vec{k},0) + \frac{1}{s^2\epsilon_r(s) + c^2|\vec{k}|^2}\frac{\partial E(\vec{k},0)}{\partial t}|_0$$

$$\equiv \beta(\vec{k},s)E(\vec{k},0) + \alpha(\vec{k},s)\frac{\partial E(\vec{k},0)}{\partial t}|_0$$

This solution is true for all dispersion relations based on a temporal convolution.

Post's Formula For Pulse Propagation

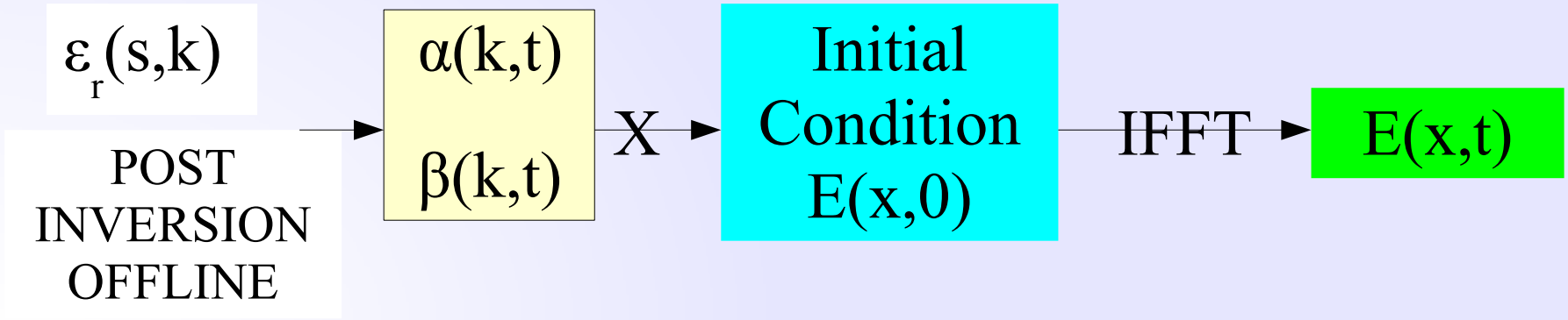
Concept

Create databases of pre-computed tables which can be used by devices which must operate in real-time.

Approach

For a given disperison relation $\varepsilon(k,s)$, numerically invert $\alpha(k,s)$ & $\beta(k,s)$ at times 't' via Post's formula.

Space-time solution from inverse Fourier transform of coefficients and initial conditions products.



Post's Formula For Pulse Propagation

Three approaches attempted for the derivatives.

$$\beta = s\alpha$$

$$\frac{\partial^q \beta}{\partial s^q} = s\frac{\partial^q \alpha}{\partial s^q} + q\frac{\partial^{q-1} \alpha}{\partial s^{q-1}}$$

$$\frac{\partial^q \alpha}{\partial s^q} = \frac{\partial^q}{\partial s^q}(s^2\epsilon_r(s) + c^2|\vec{k}|^2)^{-1}$$

1. Standard Gaver-Wynn-Rho
   a. Finite Differences + Wynn-Rho Acceleration
   b. Brute force application of the Gaver-Functionals for each α(s,k).
   c. Dispersion relation ε for each s & k.

2. Gaver-Post
   a. Finite Differences + Wynn-Rho Acceleration
   b. Store ε(s) and recall for each kth α(s,k) computation

3. Bell-Post
   a. Explicit Derivatives + Wynn-Rho Acceleration
   b. Store ε(s) and recall for each kth α(s,k) computation
   c. Use Faa' di Bruno's formula & Bell polynomials for explicit derivatives

Bell-Post Method

In the Bell-Post method,
Faa' di Bruno's formula is used to express the inversion
in terms of the susceptibility ε(s) and its arbitrary order derivatives.

$$\frac{d^q}{dx^q} f(g(x)) = \sum_{p=0}^{q} \frac{d^p f}{dx^p}(g(x)) B_{q,p}\left(\frac{dg}{dx}, \frac{d^2 g}{dx^2}, \ldots, \frac{d^{q-p+1} g}{dx^{q-p+1}}\right)$$

Faa' di Bruno's Formula

Formula expressed in terms of Bell polynomials of the 2nd kind ($B_{q,p}$)

www.mathworks.com/matlabcentral/fileexchange/14483

$$f(s) = 1/s$$

$$g(s) = s^2 \epsilon_r(s) + c^2 |k|^2$$

$$\frac{\partial^q \alpha}{\partial s^q} \equiv D^q \alpha$$

$$D^q \alpha = \sum_{p=0}^{q} \frac{p!(-1)^p}{g^{p+1}} B_{q,p}(Dg(s), D^2 g(s), \ldots, D^{q-p+1} g(s))$$

$$D^n g(s) = s^2 D^n \epsilon_r(s) + 2sn D^{n-1} \epsilon_r(s) + n(n-1) D^{n-2} \epsilon_r(s)$$

Leibniz Rule

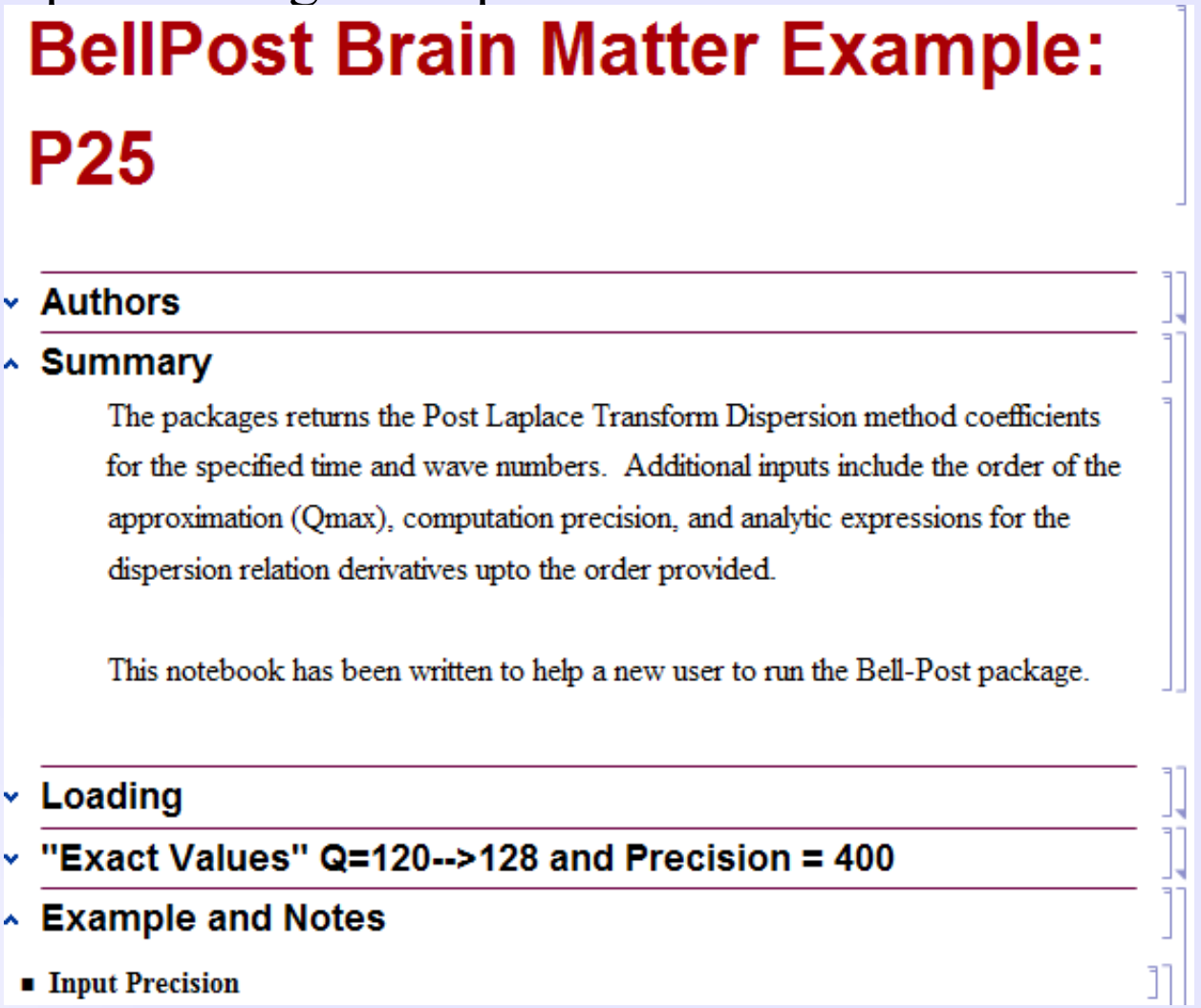
The pulse description is reduced to evaluating $\varepsilon_r(s)$ & its derivatives.

Post Mathematica Implementation

- The round-off errors can not be neglected.
- They can be mitigated by using fixed high-precision variables.
- There are multiple tools for performing fixed precision arithmetic.

ARPREC
An **Ar**bitrary
Precision
Computation Package
Lawrence Berkeley
National Laboratory

GMP
GNU
Multiple
Precision
Arithmetic Library



- Mathematica allows for wider distribution and modularity.

Brain-White Matter Example

- The fractional exponents prohibit analytic inversion.

$$\epsilon_\infty = 4.0 \qquad \sigma = 0.02 \qquad \{b\} = 1$$

$$\epsilon_r(s) = \epsilon_\infty + \sum_n \frac{\delta\epsilon_n}{\left(1 + (s\tau_n)^{1-a_n}\right)^{b_n}} + \frac{\sigma}{s\epsilon_0}$$

$$\delta\epsilon = (32.0, 100.0, 4.0 \cdot 10^4, 3.5 \cdot 10^7)$$
$$\tau = (7.96(ps), 7.96(ns), 53.05(ns), 7.958(ms))$$
$$a = (0.10, 0.10, 0.30, 0.02)$$

- The nontrivial dispersion relation expressions in high precisions is time consuming → storing ε(s) and recalling it for each k^{th} α

$$\frac{d^q\epsilon_r(s)}{ds^q} = \frac{\sigma}{\epsilon_0}\frac{(-1)^q q!}{s^{q+1}} + \sum_n \delta\epsilon_n \frac{d^q}{ds^q} f_n(g_n(s))$$

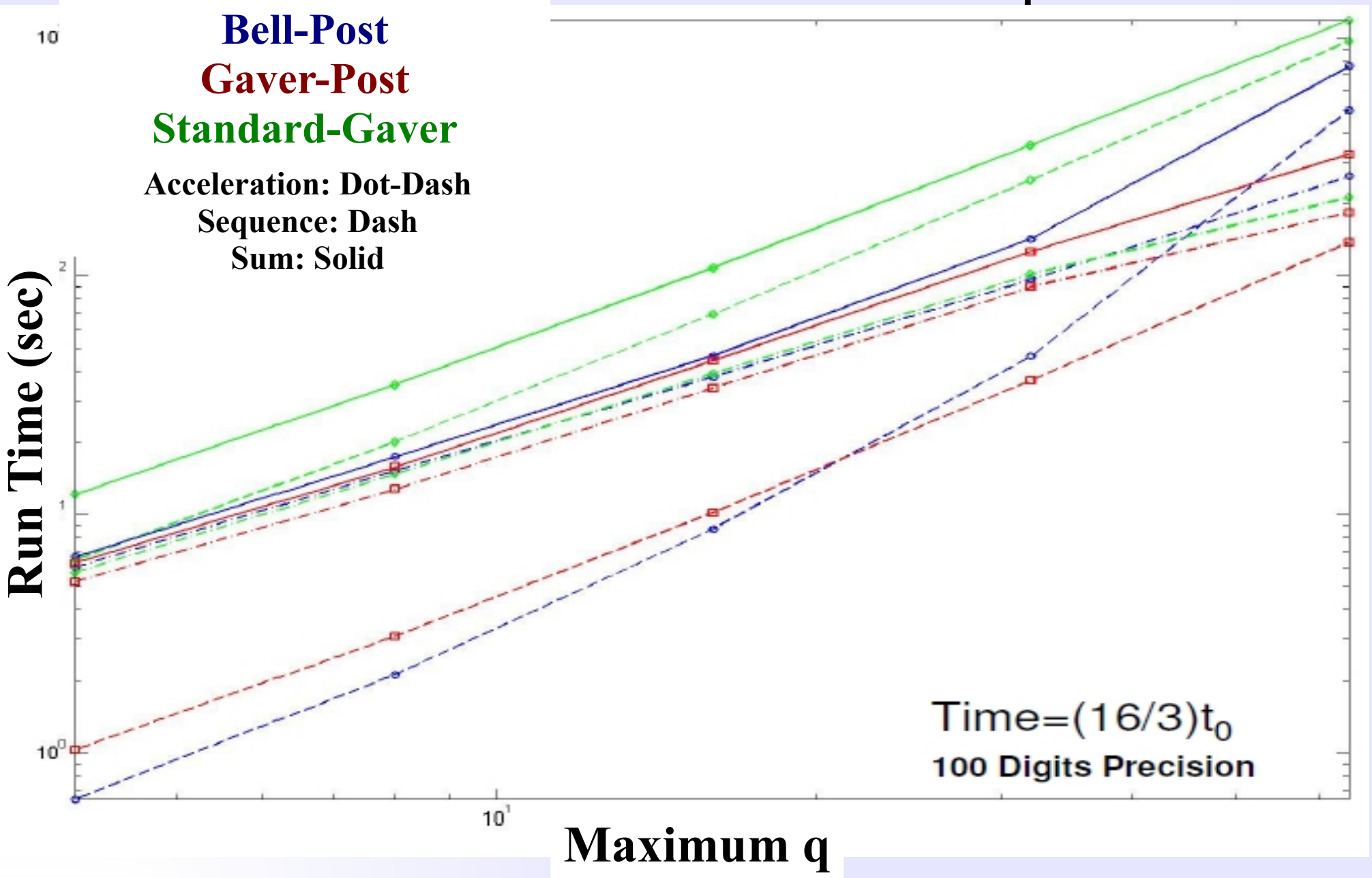
$$g(s) = (\tau s)^{1-a}$$
$$f(s) = (1+s)^{-b}$$

$$\frac{d^q f(g(s))}{ds^q} = \sum_{k=0}^{q} f^k(g(s)) B_{q,k}\left(g^1, g^2, \ldots, g^{q-k+1}(s)\right)$$

$$g^p(s) = \left[\tau^{1-a}\prod_{j=0}^{p-1} 1 - a - j\right] s^{1-a-p}$$

$$f^k(g) = \left[(-1)^k \prod_{j=0}^{k-1} b + j\right] (1+g)^{-(b+k)}$$

Brain White Matter Example



The sequence acceleration times dominated over the sequence generation time.

Brain White Matter-Bell Post

α relative error percentage for the largest k-value

q	2/3τ	4/3τ	8/3τ	16/3τ
Precision 25				
1→4	3.682e-2	5.952e-2	0.7637	0.3708
1→8	9.891e-2	6.692e-3	1.004	0.2810
1→16	7.612e-4	8.735e-4	9.471e-2	8.872e-2
1→32	6.217e-5	1.151e-4	7.445e-3	2.629e-2
1→64	4.13e-3	-	-	-
Precision 100				
1→4	3.682e-2	5.952e-2	0.7637	0.3708
1→8	1.972e-7	2.225e-5	1.470e-3	1.846
1→16	7.303e-12	9.037e-13	7.625e-11	1.034e-4
1→32	1.456e-17	5.413e-15	3.655e-12	9.873e-10
1→64	1.457e-17	5.413e-15	3.655e-12	9.873e-10

The computed coefficients can be accurately computed with sufficient precision & derivative order.

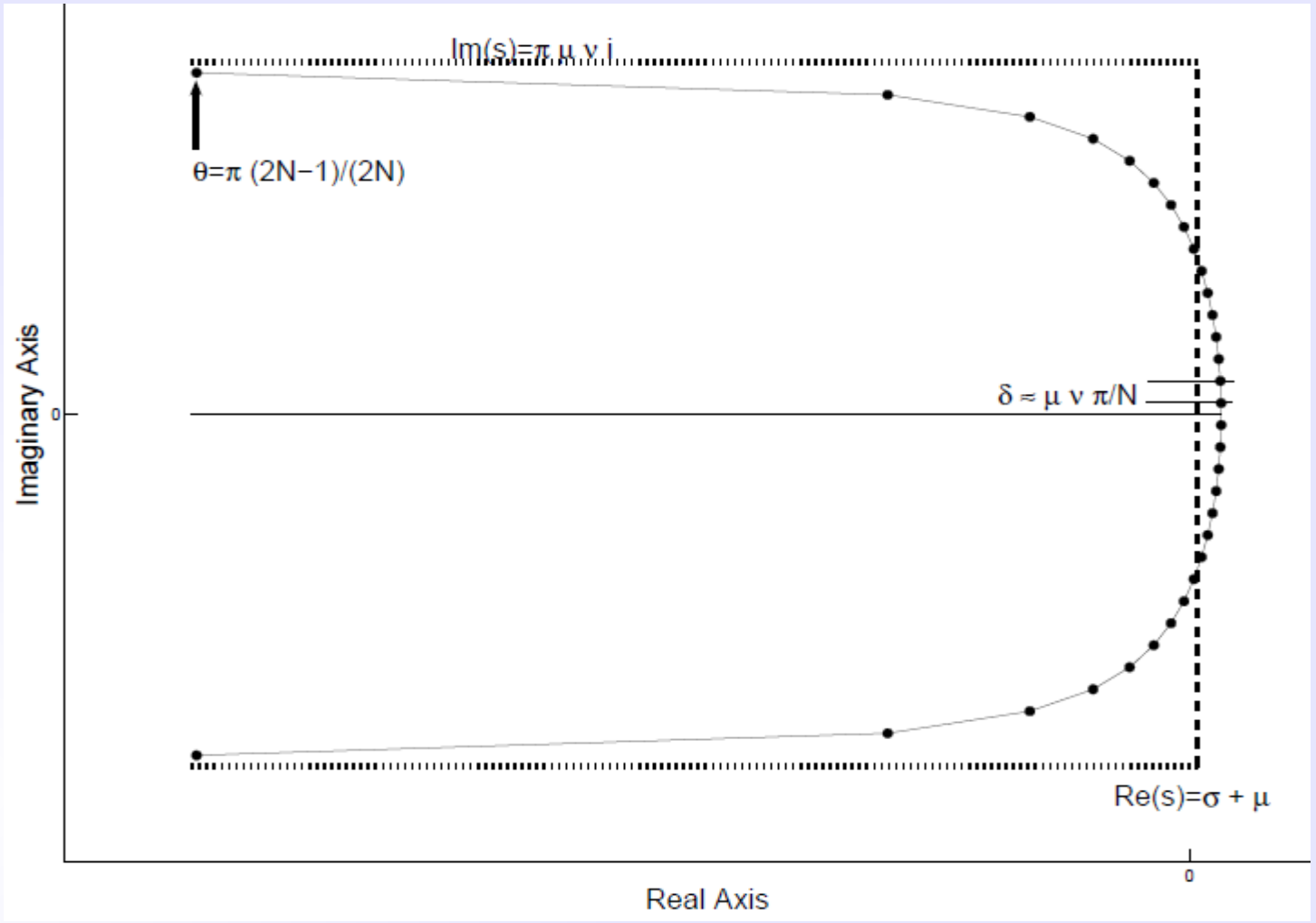
Talbot's Method

Talbot's method is based on a deformation of the Bromwich contour.

$$s(\theta) = \sigma + \mu\theta(i\nu + \cot(\theta))$$

$$\sigma \in (-\infty, \infty), \mu \in [0, \infty),$$
$$\nu \in [0, \infty), \theta \in (-\pi, \pi)$$

Replace the contour with one which opens towards the negative real axis
→ damping of highly oscillatory terms



Talbot's Method

Talbot's method is easily implemented in Mathematica.

$$s(\theta) = \sigma + \mu\theta(i\nu + \cot(\theta))$$

$$\frac{ds}{d\theta} = i\nu\mu + \mu\left[\cot(\theta) - \theta\csc^2(\theta)\right]$$

$$f(t) = \frac{1}{2\pi i}\int_{-\pi}^{\pi} e^{st}F(s)\frac{ds}{d\theta}d\theta$$

Precision	Run Time (sec)
10	0.047
20	0.141
40	0.391
80	1.625

```
Timeval = 1;
Rval = 1/2;

Flap[s_]=Exp[-2*Sqrt[s]];
Tfunexact[t_] =Exp[-1/t]/Sqrt[Pi*t*t*t];
Valexact = N[Tfunexact[Timeval],1000]

STalbot[r_,a_]=r*a*Cot[a]+I*r*a;
dsda[r_,a_]=I*r*(1+I*(a+Cot[a]*(a*Cot[a]-1)));
TimeDfun[r_,t_]:=1/(2*Pi*I)*NIntegrate[Exp[STalbot[r,a]*t]*Flap[STalbot[r,a]]*dsda[r,a],{a,-Pi,Pi},WorkingPrecision→20];
{Timeval,Approxval}=Timing[TimeDfun[Rval,Timeval]]
RelError = Abs[Approxval-Valexact]/Valexact
```

$$F(s) = e^{-2\sqrt{s}} \Longleftrightarrow f(t) = \frac{e^{-1/t}}{\sqrt{\pi t^3}}$$

$$\sigma = 0, \mu = 1/2, \nu = 1$$

Talbot's Method

The key issue in Talbot's method is the choice of the parameters (σ,μ,ν).

$$F(s) = \frac{s}{s^2+2} \Longleftrightarrow f(t) = \cos\left(\sqrt{2}t\right)$$

$$\sigma = 0, \mu = 1/2, \nu = 1$$

→ Complete Failure for the Same Parameter Values

Attempts have been made to automate the parameter selection:

- "Algorithm 682: Talbot's method of the Laplace inversion problems", Murli & Rizzardi, 1990.
- "Optimizing Talbot's contours for the inversion of the Laplace transform" A. Weideman, 2006
- "Parabolic and Hyperbolic contours for computing the Bromwich integral" A. Weideman & L.N. Trefethen, 2007

- "Dempster-Shafer Evidential Theory for the Automated Selection of Parameters for Talbot's Method Contours and Application to Matrix Exponentiation"
  P. Kano, M. Brio, P. Dostert, J. Cain, submitted 2011

Dempster-Shafer Talbot

Dempster-Shafer theory of evidential reasoning is applied to the problem of optimal contour parameters selection in Talbot's method.

Concept

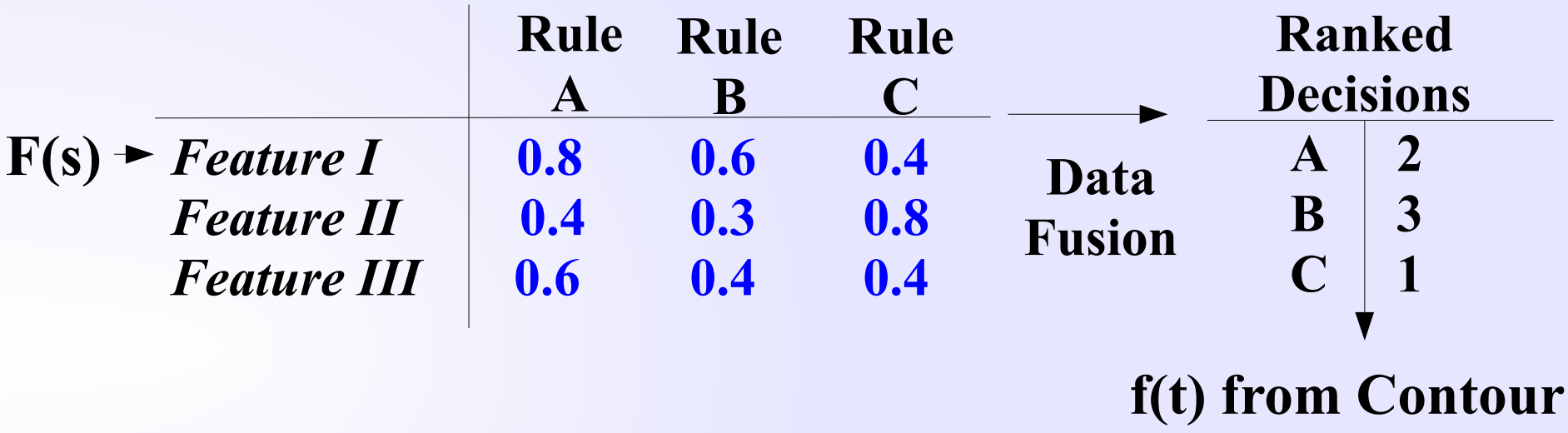
Discriminate between rules for the parameters that define the shape of the contour based on the features of the function to invert.

Example

Matrix exponential via numerical inversion of the corresponding resolvent matrix.

Implementation

MATLAB with results are compared with those from the rational approximation from 'expm'.



Dempster-Shafer Evidential Theory

Dempter-Shafer [DS] theory is routinely used in the defense industry to differentiate between possible threats or targets.

It assigns mass to sets of classes as opposed to individual elements as in standard probability theory.

Class A

Class B

→ Color, Height, Width, Mass → Weights Per Class & Feature based on Prior Training → **DS Data Fusion** → **Friend** or **Foe**

Two Step Implementation

Identical Features & Classes

Off-line Algorithm Training on Test Data Sets

Real Time Feature Extraction & Data Fusion

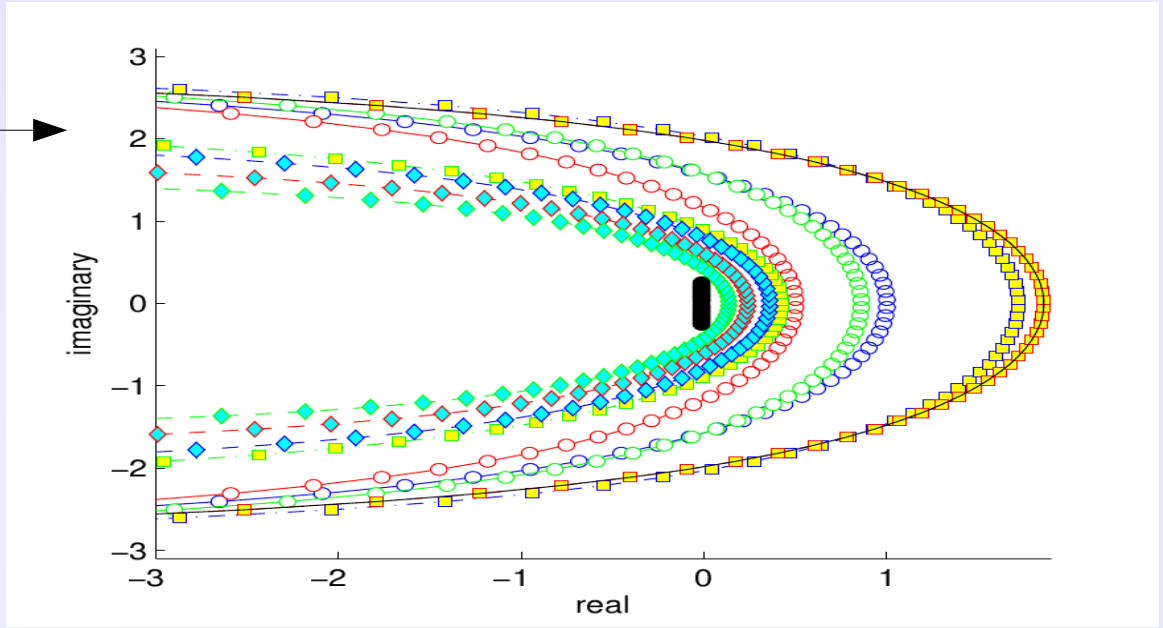
Dempster-Shafer Formalism

sources of evidence: $F(s)$
decision states: rules for the contour parameter values Θ
frame of discernment: power set 2^{Θ} of the set of decision states
features: the information extracted from $F(s)$ that allows one to choose a set of contour parameters over another
mass function: the mass m is assigned to an frame of discernment element based on the feature values
belief function: the final metric from the fusion of the masses on a decision state. It is used for a contour selection.

decision states

Class	σ	ν	μ
1	0	1	1
2	$\mathscr{R}(\text{trace}(A))$	1	1
3	0	$\frac{1}{\|A\|}$	$\|A\|$
4	$\|A\|$	$\frac{1}{\|A\|}$	$\|A\|$
5	$\|A\|$	1	1
6	$\mathscr{R}(\text{trace}(A))$	$\|A\|$	$\sqrt{\nu}$
7	$\mathscr{R}(\text{trace}(A))$	$\|A\|$	ν
8	$\mathscr{R}(\text{trace}(A))$	$\|A\|$	ν^2
9	$\mathscr{R}(\text{trace}(A))$	$\|A\|$	ν^3

sources of evidence $F(s) = (sI - A)^{-1}$



Features

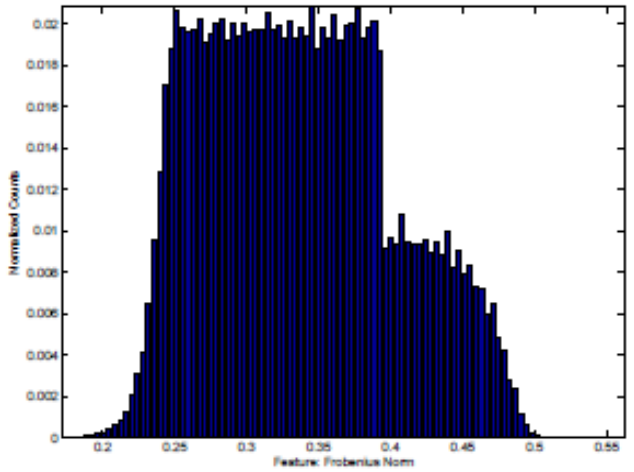
Features for inversion of the resolvent matrix $F(s)=(sI-A)^{-1}$ are based on the properties of A.

Feature	Motivation
Trace Real Part	Sum of Eigenvalues
One Norm	Maximum Absolute Column Sum
Frobenius Norm	Spectral Radius if Hermetian
Infinity Norm	Maximum Absolute Row Sum

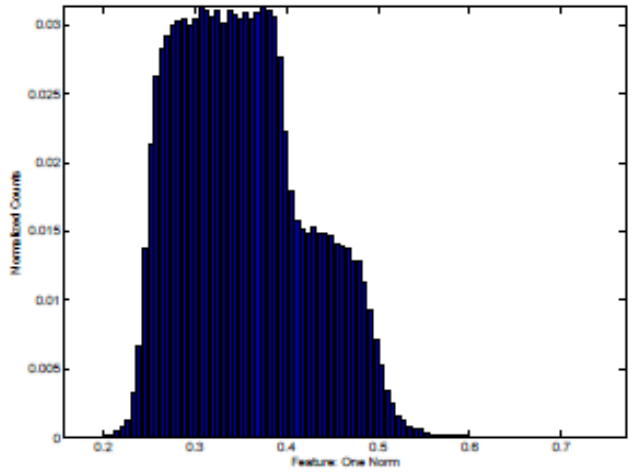
Training Set

Uniform Random Matrices

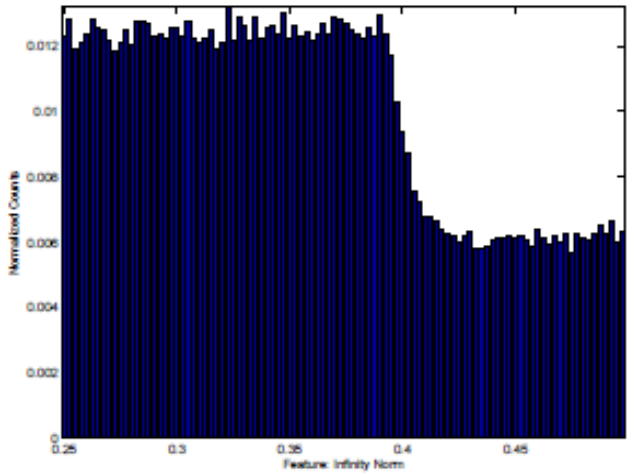
Parameter	Value
Dimension	2,3,...,100
#/dimension	1500
Real Values	[-100,100]
Imaginary Values	[-100,100]
Seed	1234



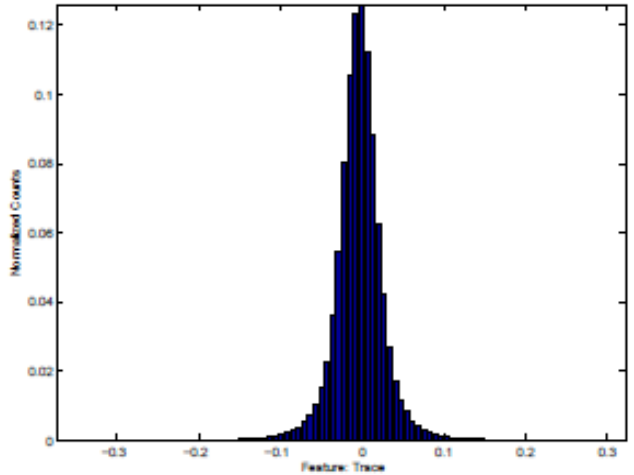
(a) Frobenius Norm



(b) One Norm



(c) Infinity Norm



(d) Trace Real Part

Mass

mass functions

- from comparison of the matrix exponential from Talbot's method & MATLAB's *expm* Pade' approximation
- defined using the exponential of the Frobenius norm

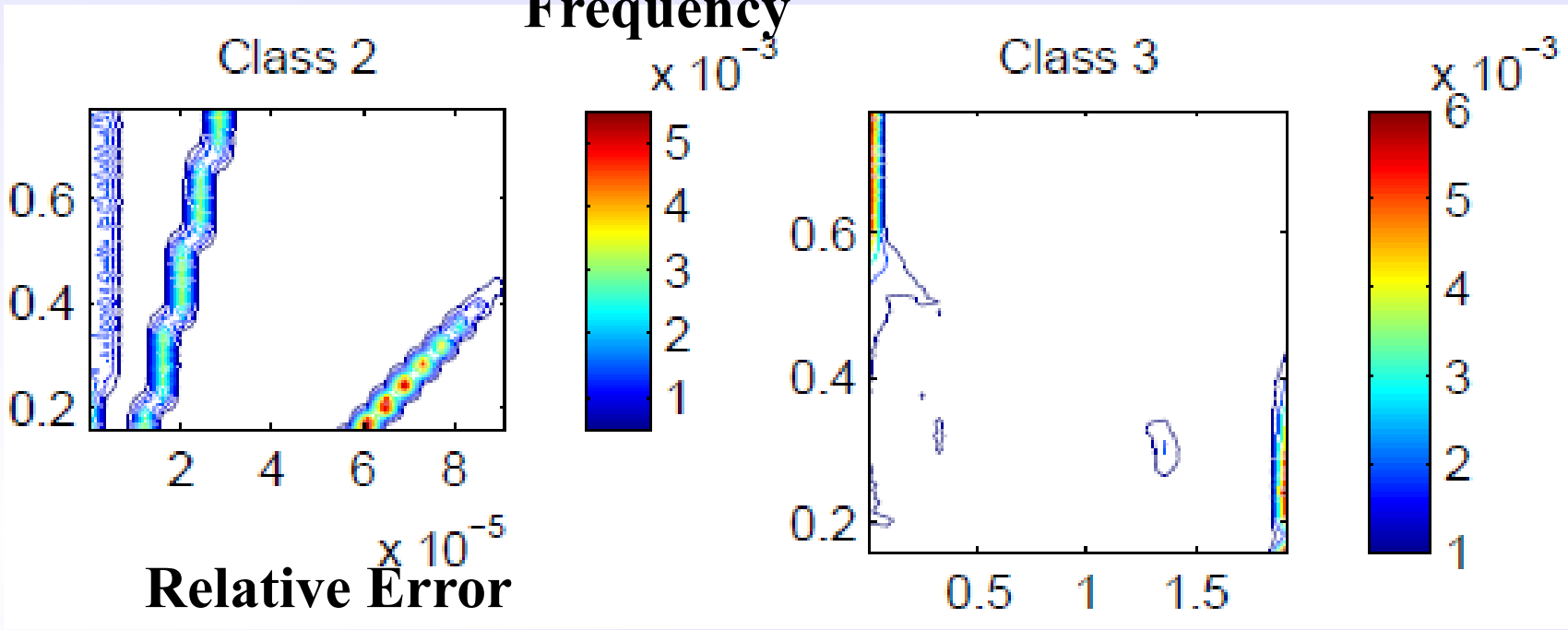
$$f_n = \text{feature value of matrix } A_n$$

$$T_A^c = \text{Talbot exponential of } A \text{ on contour } c$$

$$P_A = \text{Pade' exponential of } A$$

$$E_{nc} = \| \frac{T_A^c - P_A}{P_A} \|_{Frobenius}$$

$$m_{nc} = e^{-E_{nc}}$$



Mass Function

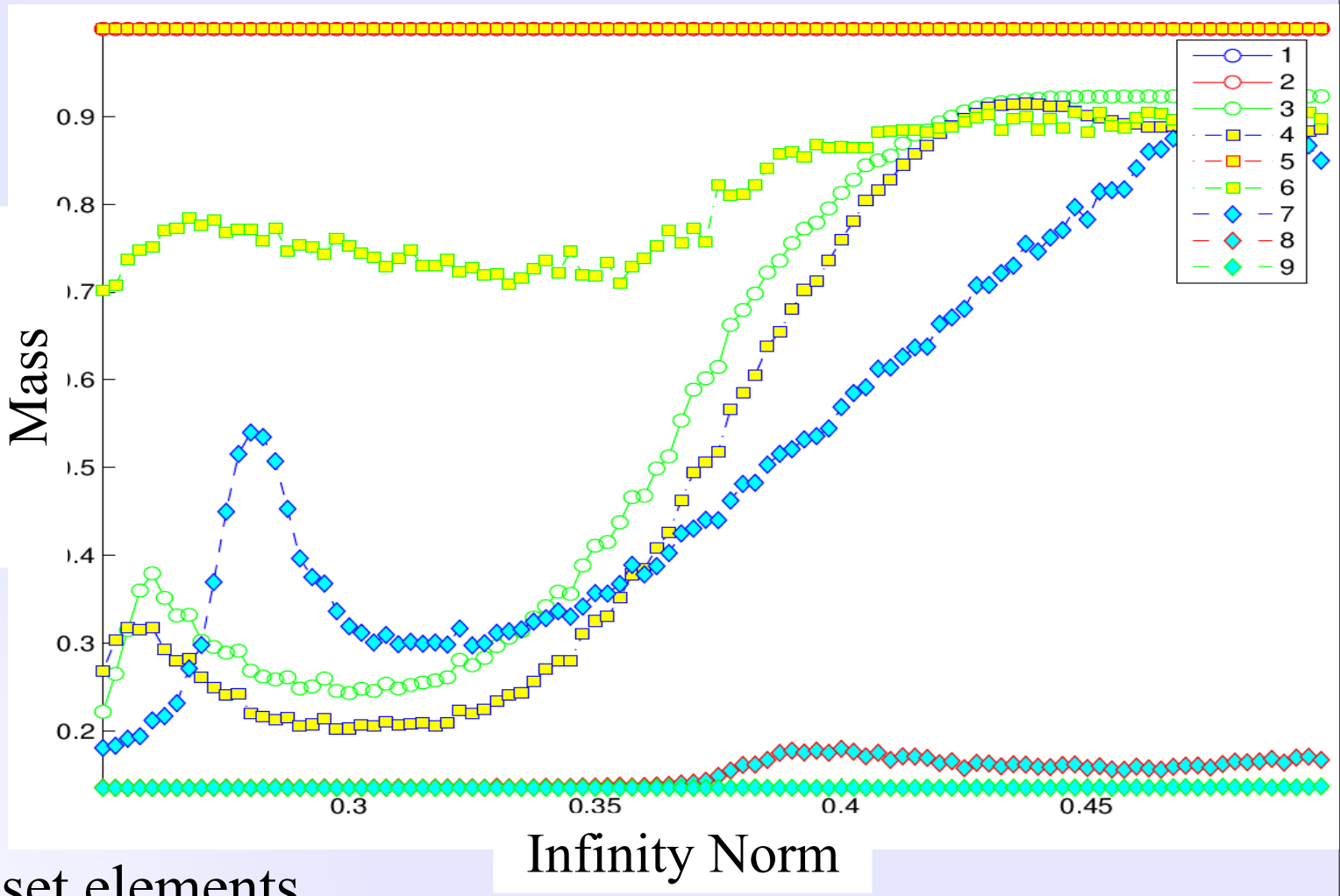
$$m_{kc} = \max_n \{e^{-E_{nc}}, f_n \in f_k \pm \delta f\}$$

$m : 2^{\Theta} \rightarrow [0, 1]$
has the properties:

- $m(\varnothing) = 0$
- $\sum m(2^{\Theta}) = 1$

The feature values are binned.

Mass is assigned to each class & k^{th} bin center value.



Mass on the power set elements is defined as the sum of the masses on the singleton sets:

$m(\{a,b\}) = m(\{a\}) + m(\{b\})$

a, b = contour parameter rules

Data Fusion & Belief

For every power set 2^{Θ} element **C**, fusion of the masses occurs via the Dempster-Shafer fusion rule:

$$m^{1,2}(C) = \frac{\sum_{A \cap B = C} m^1(A) m^2(B)}{1 - \sum_{A \cap B = \emptyset} m^1(A) m^2(B)}$$

- The fused mass assigned to a set **C** is a normalized sum of the masses from sets **A** and **B** supporting **C.**

- The normalization removes the sum of the masses from **A** and **B** for features 1 & 2 which are in conflict.

- One now returns to original set of decision states Θ and calculates *belief* on each.

$$b(A) = \sum_{B \subseteq A} m(B)$$

$$b(\emptyset) = 0$$

$$b(\Theta) = 1$$

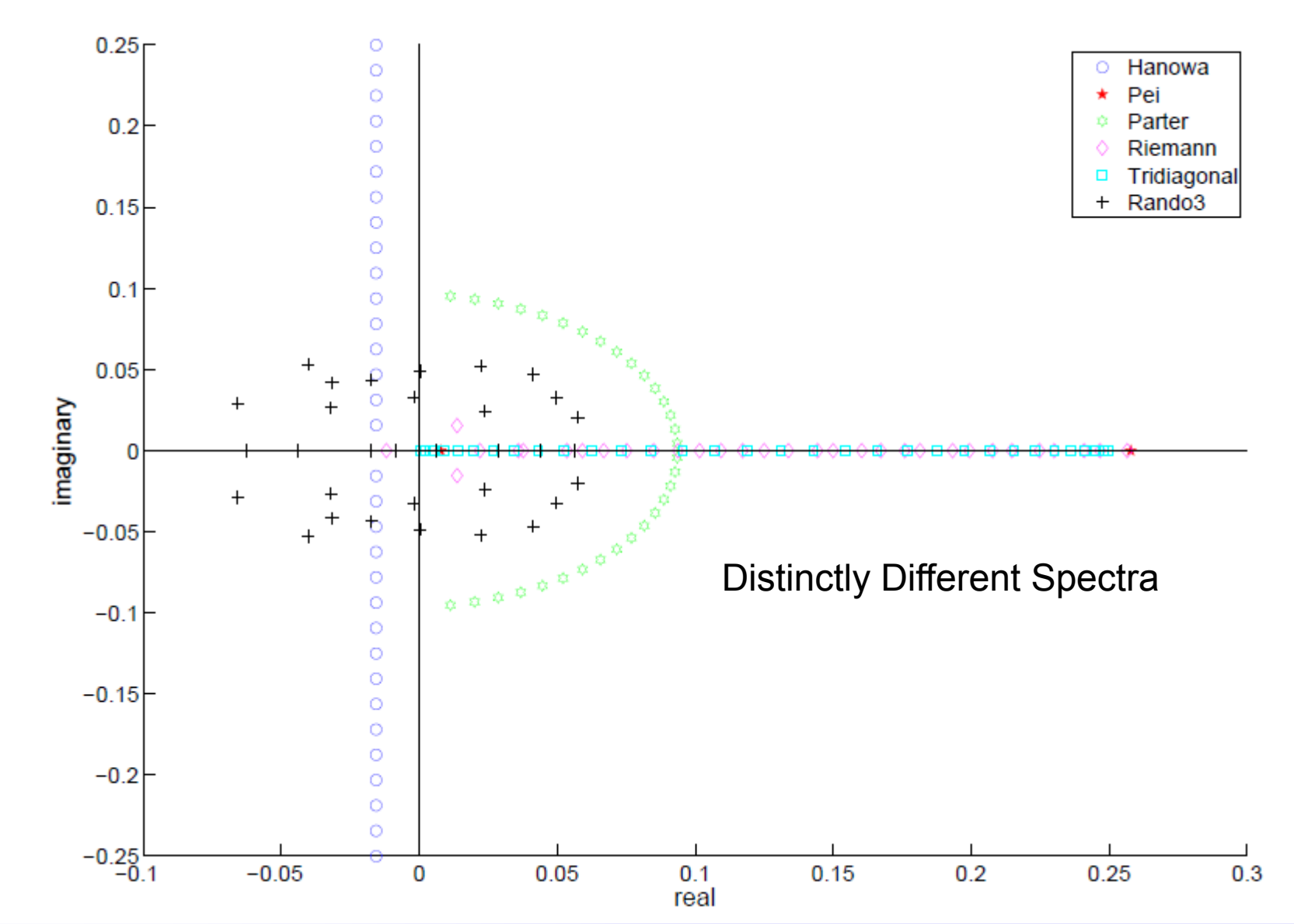
- The contour is selected based on the maximum belief.

$$\text{if } A, B \subseteq \Theta \text{ and } b(A \cap B) = \emptyset$$

$$b(A \cup B) = b(A) + b(B)$$

Test Cases

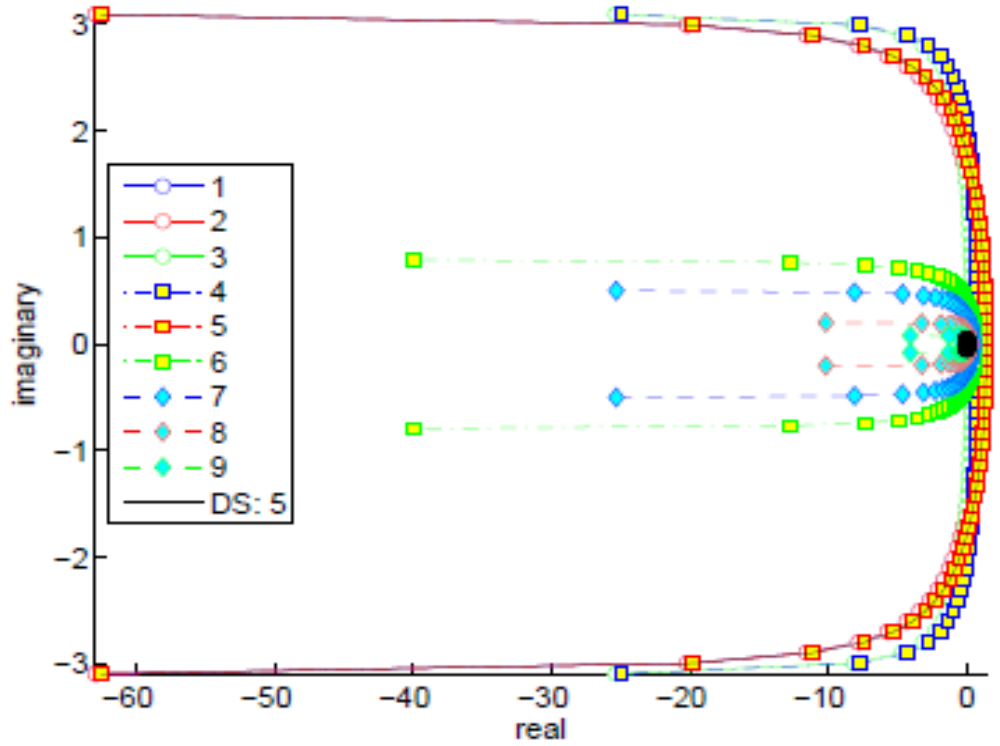
The matrix exponential of six square matrices from the MATLAB matrix gallery were used for illustration.

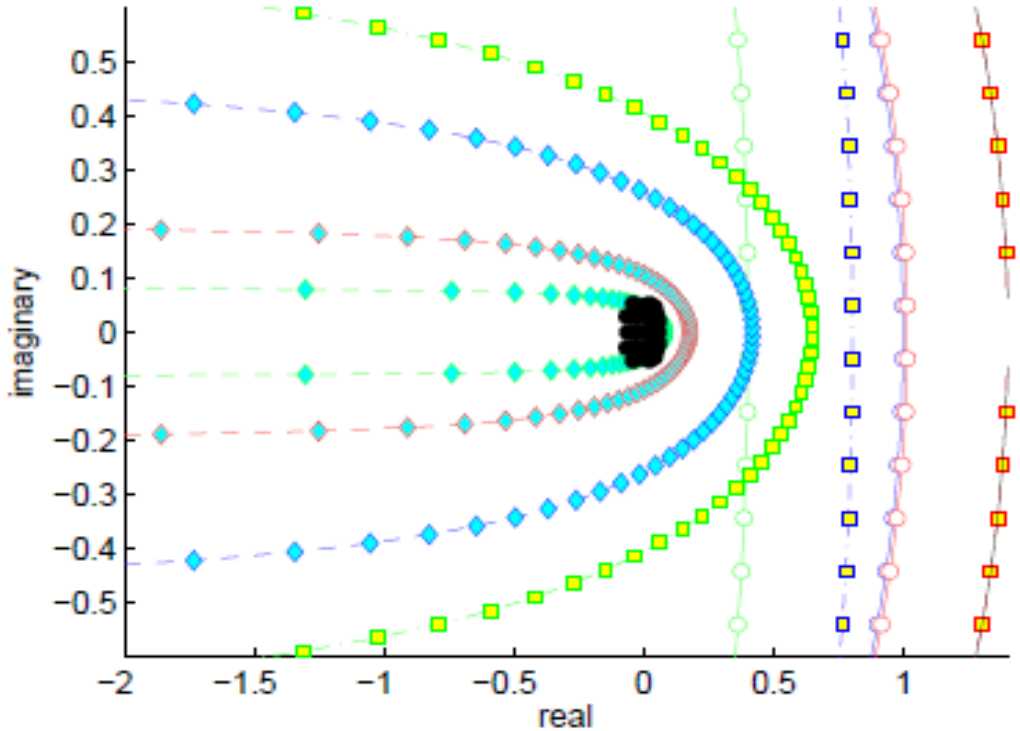


Test Case:
32x32 Rando3

The DS-Talbot implementation correctly associates the belief with the relative error.

Contour	Belief	Δ Error	Contour Rank by Belief
C-5	0.17429	0.0	1
C-1	0.17429	9.12e-10	2
C-2	0.17429	9.83e-10	3
C-6	0.15913	9.05e-5	4
C-3	0.14410	2.49e-4	5
C-4	0.14020	2.85e-4	6
C-7	0.12761	1.55e-3	7
C-8	0.093395	6.23e-2	8
C-9	0.091523	4.92e2	9





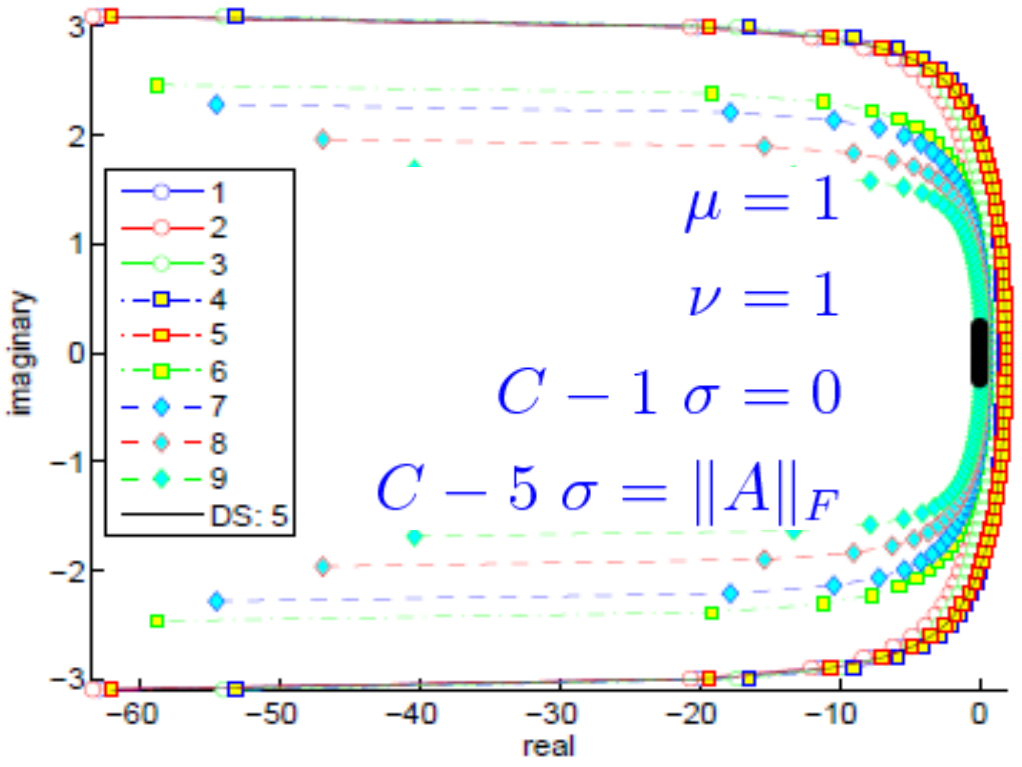
Test Case: 32x32 Hanowa

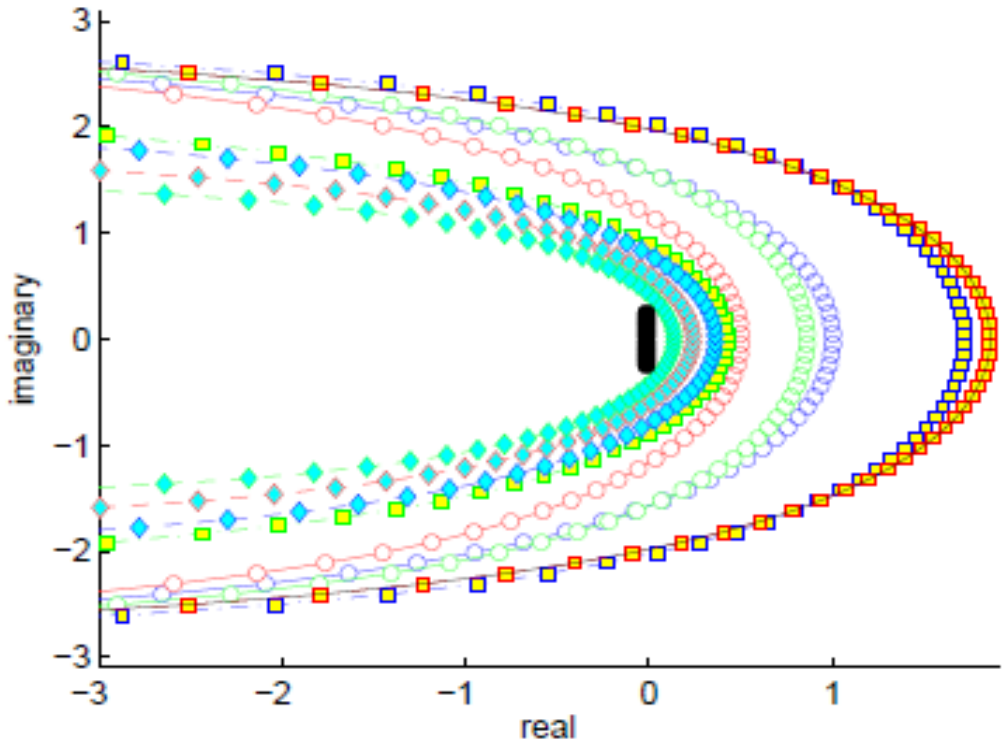
That Talbot's method on contours 1,2,5 is likely due to the matrix scaling used.

$$e^A = (e^{A/q})^q$$

$$q = \max(0, 1 + log_2(\|A\|_\infty))$$

Contour	Belief	Δ Error	Contour Rank by Belief
C-1	0.32816	0.0	2
C-2	0.32816	2.06e-8	3
C-4	0.23646	1.29e-7	6
C-5	0.32816	2.39e-7	1
C-3	0.24252	3.72e-7	5
C-6	0.29859	2.37e-6	4
C-7	0.23489	1.10e-5	7
C-8	0.22113	2.45e-5	8
C-9	0.22112	2.20e-3	9





Varying Matrix Size & # Integration Points

Rando3

	16	32	64	128
16	[5] 2.57e-3	[5] 1.11e-2	[2] 4.48e-2	[1] 1.78e-1
32	[5] 1.02e-5	[5] 4.21e-5	[2] 1.73e-4	[1] 6.90e-4
64	[5] 1.48e-8	[5] 6.24e-8	[2] 2.54e-7	[1] 1.02e-6
128	[5] 1.00e-12	[5] 4.62e-12	[5] 1.70e-10	[1] 1.40e-10

Hanowa

	16	32	64	128
16	[1] 1.01e-3	[1] 2.87e-3	[1] 8.04e-3	[1] 2.27e-2
32	[2] 3.40e-6	[1] 1.10e-5	[3] 1.96e-5	[1] 8.71e-5
64	[2] 5.10e-9	[1] 1.62e-8	[9] 6.09e-9	[9] 5.83e-12
128	[2] 2.89e-13	[1] 1.30e-12	[9] 1.25e-12	[8] 1.28e-12

Pei

	16	32	64	128
16	[2] 5.33e-3	[2] 2.12e-2	[2] 8.47e-2	[2] 3.38e-1
32	[1] 2.10e-5	[2] 8.39e-5	[2] 3.35e-4	[2] 1.34e-3
64	[2] 3.06e-8	[2] 1.22e-7	[2] 4.87e-7	[2] 1.95e-6
128	[1] 2.14e-12	[1] 8.56e-12	[1] 2.27e-11	[2] 6.23e-10

 # Integration Points

Square Matrix Size

Current and Future Work

- Current and future work is focused on using GPU acceleration for numerical inversion.
- The MATLAB tool on the file exchange currently only includes a GPU accelerated Weeks method.
- Dempster-Shafer evidential reasoning between the three inversion methods themselves is also direction for future work.

Laplace Transform Methods are an active area of research.

Numerical inverse Laplace transform methods will increase in popularity as computing capability increases.