COMPUTING GREEN'S FUNCTION INTEGRALS VIA RECURRENCE RELATIONSHIPS

Sheehan Olver
Imperial

Supported by EPSRC EP/T022132/1

How can we compute the Newtonian potential

$$\iint_{\Omega} f(\mathbf{t}) \log \|\mathbf{t} - \mathbf{x}\| d\mathbf{x}$$

it's gradient, and more general Green's function integrals?

Idea: when $\mathbf{\Omega}$ is a square we can expand f in tensor product of Legendre polynomials.

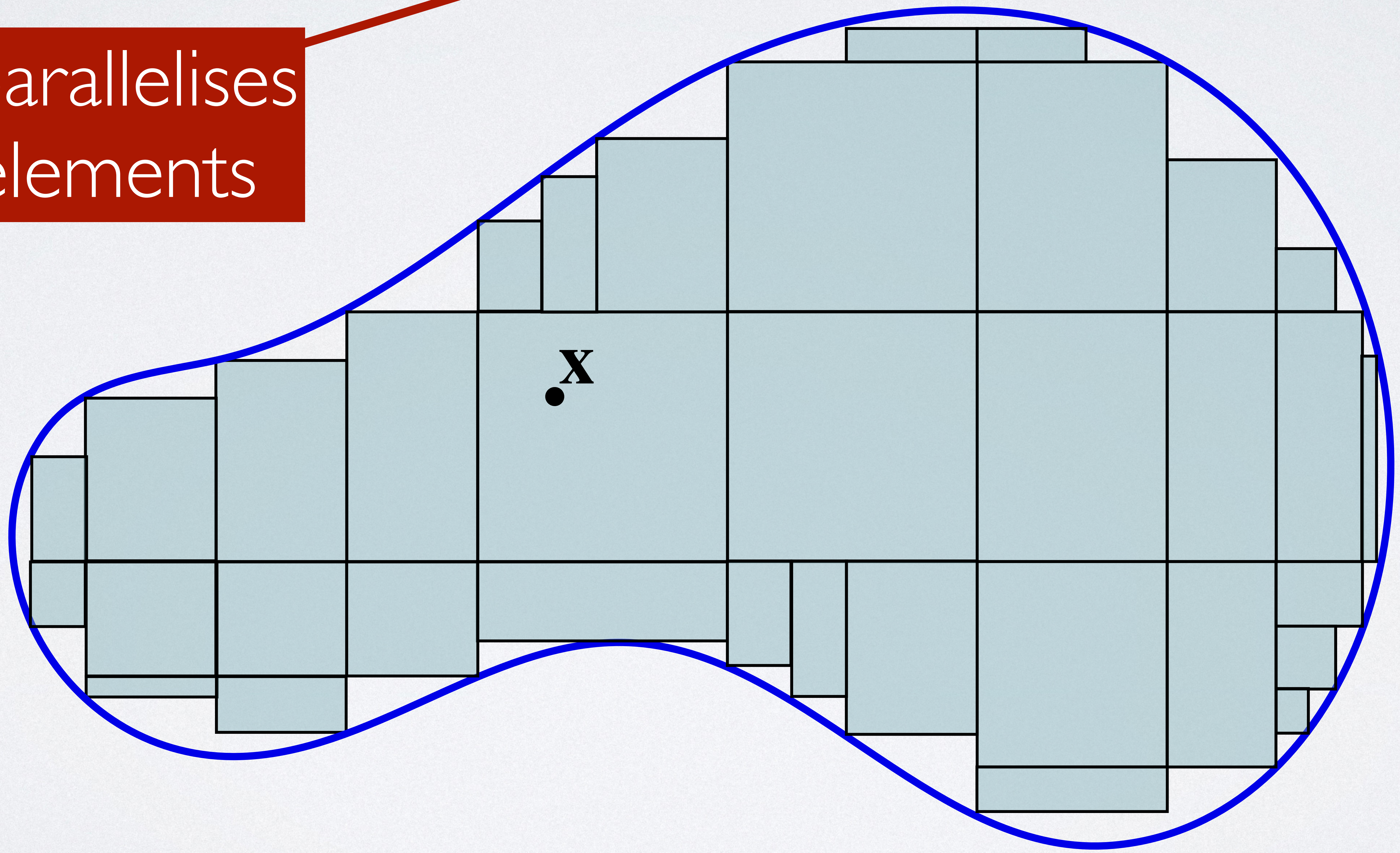
We show the Newtonian potential (and its gradient) of a tensor product of Legendre polynomials can be expressed exactly in terms of a simple recurrence.

This can be used for more general Green's functions on rectangles.

Motivation: solving inhomogenous PDEs

$$(\Delta + k^2)u = f \qquad \Rightarrow \qquad u(\mathbf{x}) = \iint_\Omega K(\|\mathbf{x} - \mathbf{t}\|)f(\mathbf{t})\mathrm{d}\mathbf{t} + v(\mathbf{x})$$

Trivially parallelises across elements



Many related numerical methods! See

Atkinson (1985)
Greengard & Lee (1996)
Barnett (2014)
Klinteberg & Tornberg (2018)
Greengard, O'Neil, Rachh & Vico (2021)
Shen & Serkh (2024)
Anderson, Bonnet, Faria & Pérez-Arancibia (2024)

and many more!

I won't pretend what I'm talking about is currently competitive, but perhaps in the future.

- From Cauchy to Log kernel on an interval
- Log kernels on a rectangle
- Bessel kernels on a rectangle

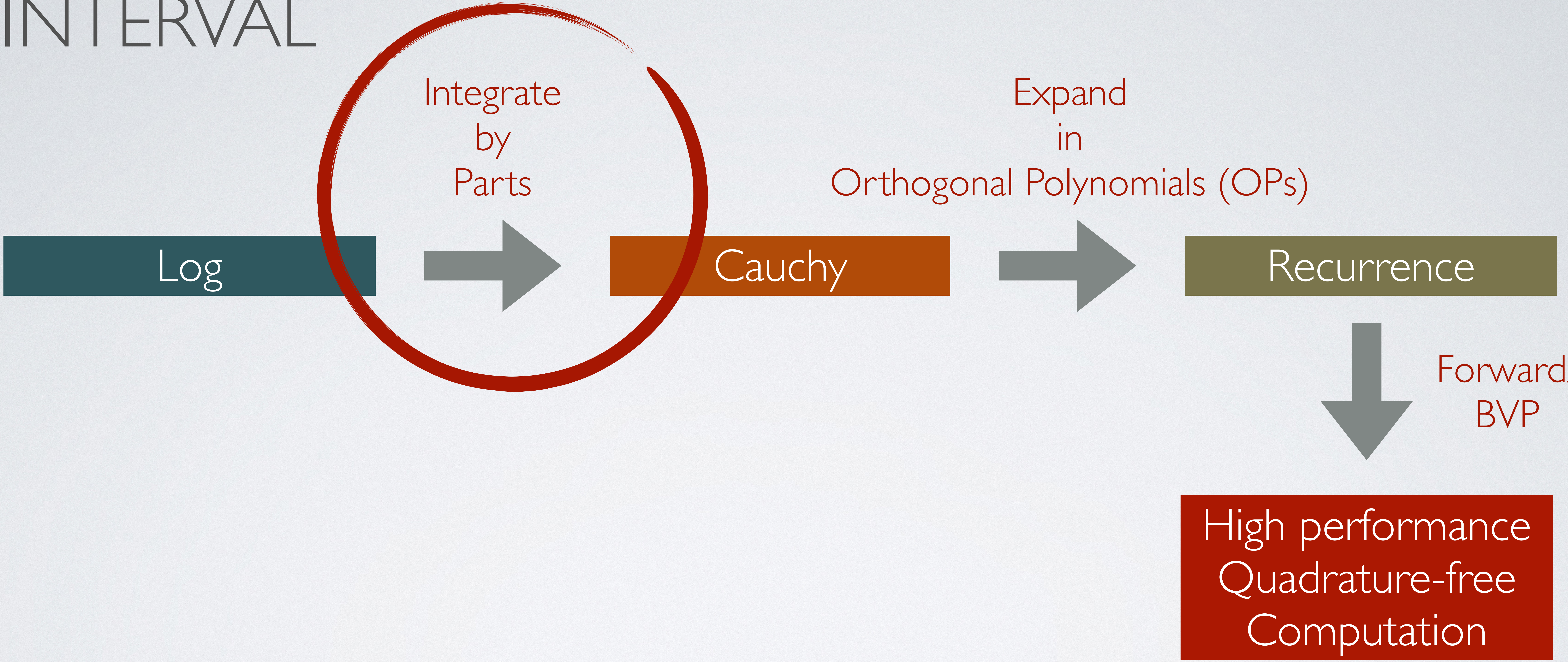
INTERVAL

Log $\xrightarrow{\text{Integrate by Parts}}$ Cauchy $\xrightarrow{\text{Expand in Orthogonal Polynomials (OPs)}}$ Recurrence

$\downarrow$ Forward/BVP

High performance Quadrature-free Computation

INTERVAL


Integrate
by
Parts
Expand
in
Orthogonal Polynomials (OPs)
Log
Cauchy
Recurrence
Forward/
BVP
High performance
Quadrature-free
Computation

Using complex variables $z = x + \mathrm{i}y$:

$$\int_{-1}^{1} \log|z - t| u(t) \mathrm{d}t = \Re \int_{-1}^{1} \log(z - t) u(t) \mathrm{d}t$$

$$= \Re \left\{ \int_{-1}^{1} u(t) \mathrm{d}t \log(z + 1) + \underline{\int_{-1}^{1} \frac{U(t)}{z - t} \mathrm{d}t} \right\}$$

Cauchy/Stieltjes transform

where $U(x) := \int_{1}^{x} u(t) \mathrm{d}t$

Using complex variables $z = x + \mathrm{i}y$:

$$\int_{-1}^{1} \log|z-t|u(t)\mathrm{d}t = \Re \int_{-1}^{1} \log(z-t)u(t)\mathrm{d}t$$

$$= \Re\left\{\int_{-1}^{1} u(t)\mathrm{d}t \log(z+1) + \int_{-1}^{1} \frac{U(t)}{z-t}\mathrm{d}t\right\}$$

Weak singularity $\longrightarrow$ Strong singularity

Bad for quadrature,
Good for deriving recurrence

INTERVAL

Log $\xrightarrow{\text{Integrate by Parts}}$ Cauchy $\xrightarrow{\text{Expand in Orthogonal Polynomials (OPs)}}$ Recurrence

Recurrence $\xrightarrow{\text{Forward/ BVP}}$ High performance Quadrature-free Computation

For Legendre polynomials:

$$\int_{-1}^{1} \log(z-t)P_0(t)\mathrm{d}t = 2\log(z+1) + \int_{-1}^{1} \frac{t-1}{z-t}\mathrm{d}t$$

$$\int_{-1}^{1} \log(z-t)P_n(t)\mathrm{d}t = -\frac{1}{2n}\underline{\int_{-1}^{1} \frac{(1-t^2)P_{n-1}^{(1,1)}(x)}{z-t}\mathrm{d}t}$$

Cauchy/Stieltjes transform
of
weighted Jacobi/Ultraspherical polynomials

Consider Cauchy transforms of weighted OPs $q_n(z) := \int_{-1}^{1} \frac{w(t)p_n(t)}{z-t} \mathrm{d}t$

$$
\begin{aligned}
zq_n(z) &= \int_{-1}^{1} \frac{zp_n(t)w(t)}{z-t}\mathrm{d}t \\
&= \int_{-1}^{1} \frac{z-t}{z-t} p_n(t)w(t)\mathrm{d}t + \int_{-1}^{1} \frac{tp_n(t)w(t)}{z-t}\mathrm{d}t \\
&= \int_{-1}^{1} w(t)\mathrm{d}t\delta_{n0} + \int_{-1}^{1} \frac{tp_n(t)w(t)}{z-t}\mathrm{d}t
\end{aligned}
$$

Classic observation: since OPs satisfy 3-term recurrence

$$xp_0(x) = a_0 p_0(x) + b_0 p_1(x)$$
$$xp_n(x) = c_{n-1} p_{n-1}(x) + a_n p_n(x) + b_n p_n(x)$$

Then so do their weighted Cauchy transforms (w/ RHS)

$$zq_0(z) = \int_{-1}^{1} w(t)\mathrm{d}t + a_0 q_0(z) + b_n q_1(z)$$
$$zq_n(z) = c_{n-1} q_{n-1}(z) + a_n q_n(z) + b_n q_{n+1}(z)$$

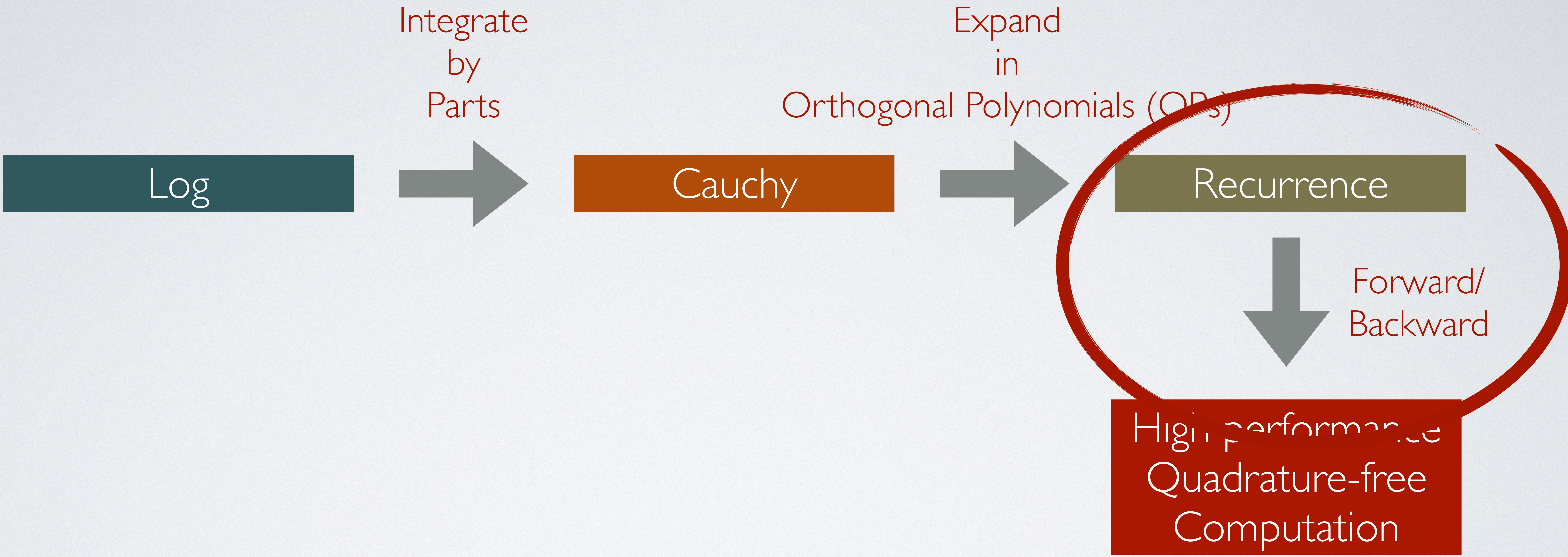
where $q_n(z) := \int_{-1}^{1} \frac{w(t)p_n(t)}{z-t}\mathrm{d}t$

Gives simple 3-term recurrence for Log Kernel of Legendre:

$$zL_0(z) = -L_0(z) + 2L_1(z) + 2(z+1)\log(z+1) - 2$$
$$zL_n(z) = \frac{n+2}{2n+1}L_{n-1}(z) + \frac{n-1}{2n+1}L_{n-1}(z)$$

where $L_n(z) := \int_{-1}^{1} \log(z-t)P_n(t)\mathrm{d}t$

INTERVAL


Log
Integrate by Parts
Cauchy
Expand in Orthogonal Polynomials (OPs)
Recurrence
Forward/ Backward
High performance Quadrature-free Computation

Recurrences, two ways

Assume we know

Forward:

$$\begin{bmatrix} 1 & & & & \\ a_0 - z & b_0 & & & \\ c_0 & a_1 - z & b_1 & & \\ & c_1 & a_2 - z & b_2 & \\ & & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} q_0(z) \\ q_1(z) \\ q_2(z) \\ q_3(z) \\ \vdots \end{bmatrix} = \begin{bmatrix} q_0(z) \\ -\int_{-1}^{1} w(t)\mathrm{d}t \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

BVP:

$$\begin{bmatrix} \cancel{1} & & & & \\ a_0 - z & b_0 & & & \\ c_0 & a_1 - z & b_1 & & \\ & c_1 & a_2 - z & b_2 & \\ & & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} q_0(z) \\ q_1(z) \\ q_2(z) \\ q_3(z) \\ \vdots \end{bmatrix} = \begin{bmatrix} \cancel{q_0(z)} \\ -\int_{-1}^{1} w(t)\mathrm{d}t \\ 0 \\ 0 \\ \vdots \end{bmatrix}$$

Methods for Backward/BVP/minimal solution

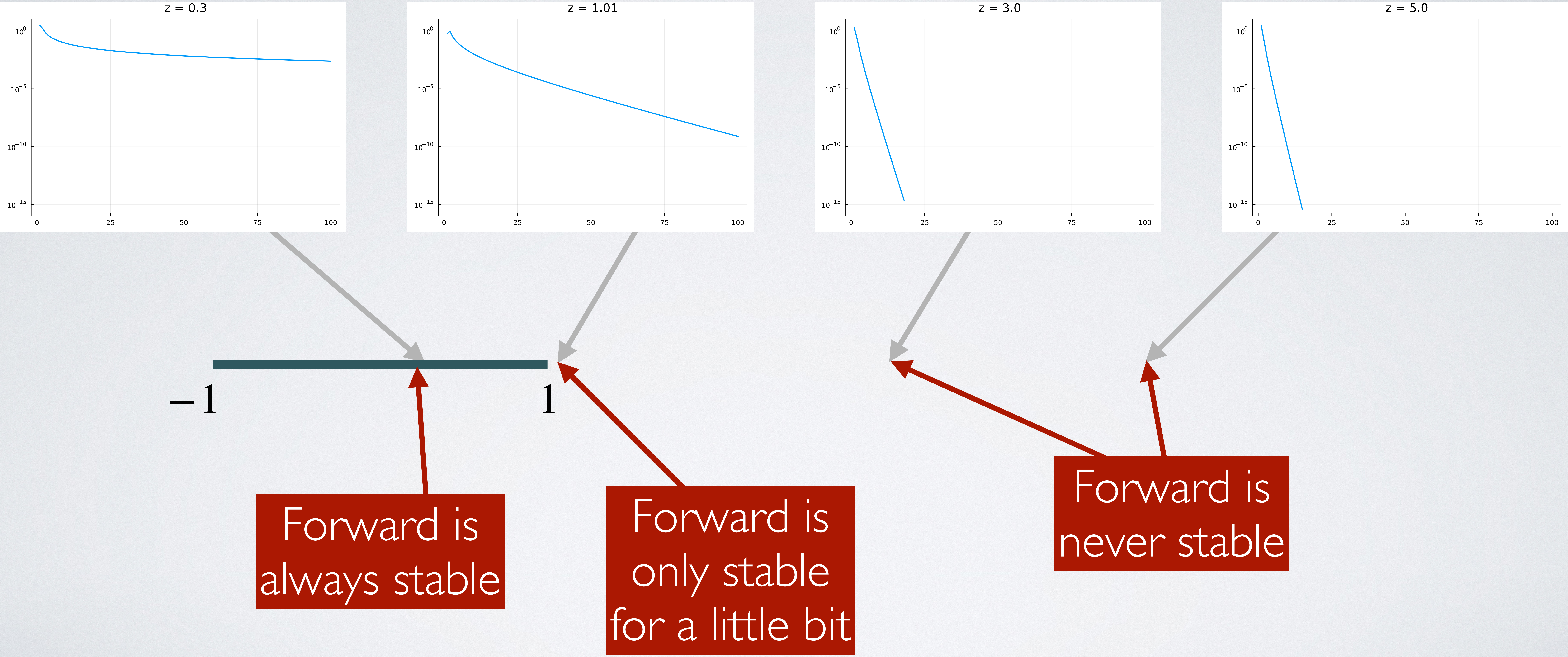
Miller (1952): shooting method

(F.W.J.) Olver (1967): adaptive Gaussian elimination

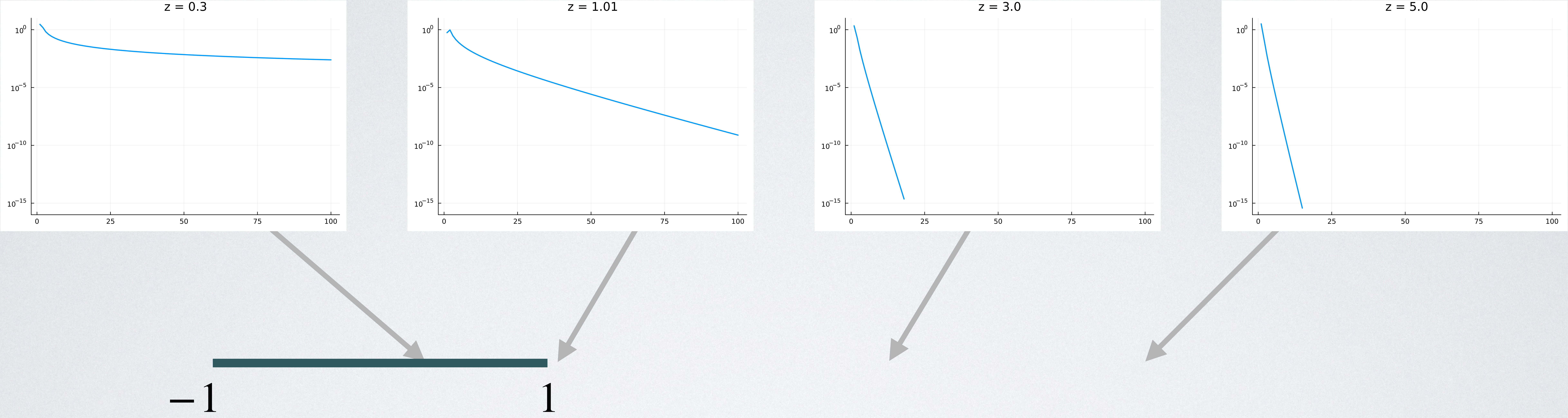
Gautschi (1981): continued fraction

All 3 break down in computational cost
as z approaches the element $[-1,1]$

Decay depends on how close to element



Decay depends
on how close to element

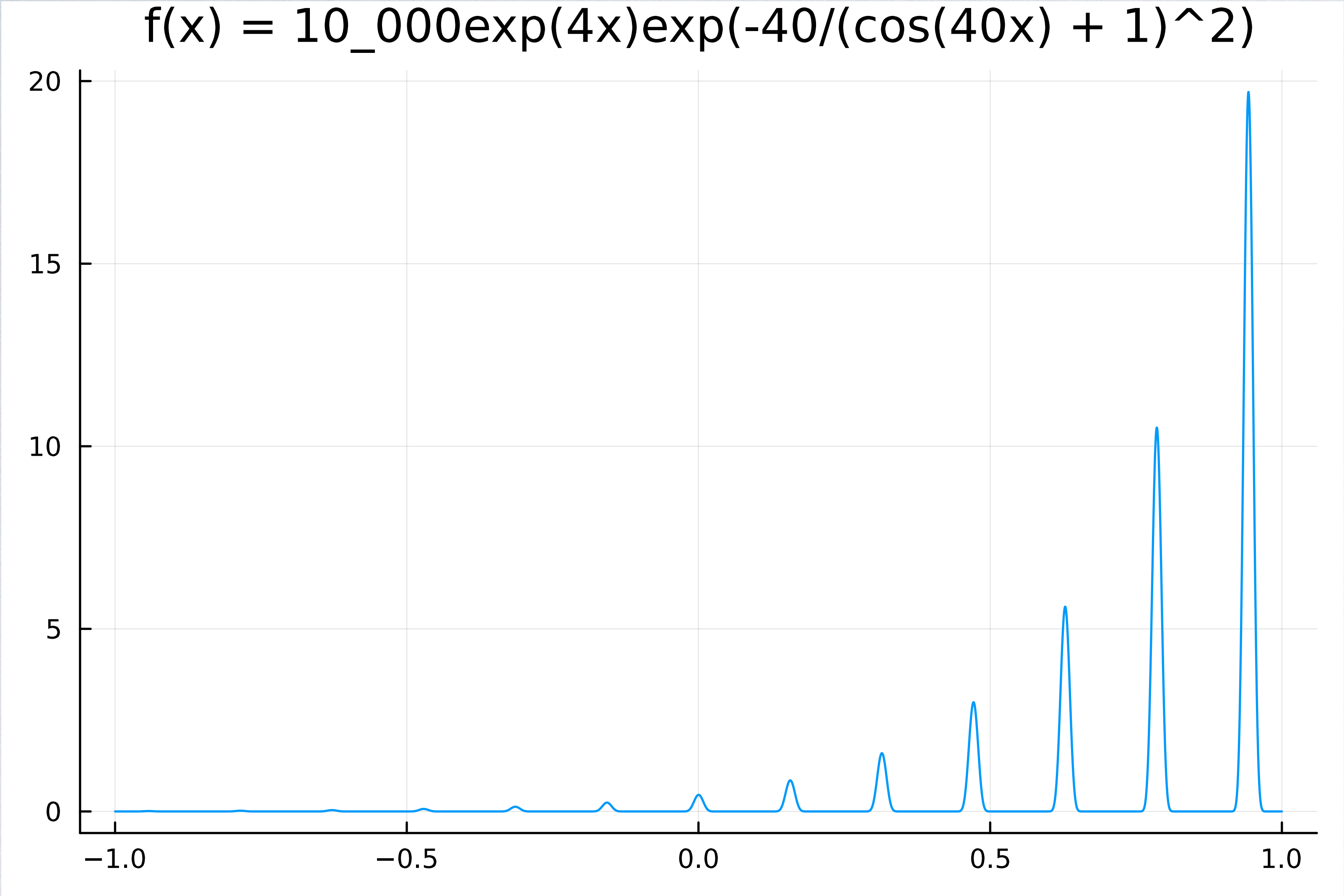


Idea: combine Forward + Olver's algorithm
using simple error estimate for forward recurrence to switch
See also Keller (1999)

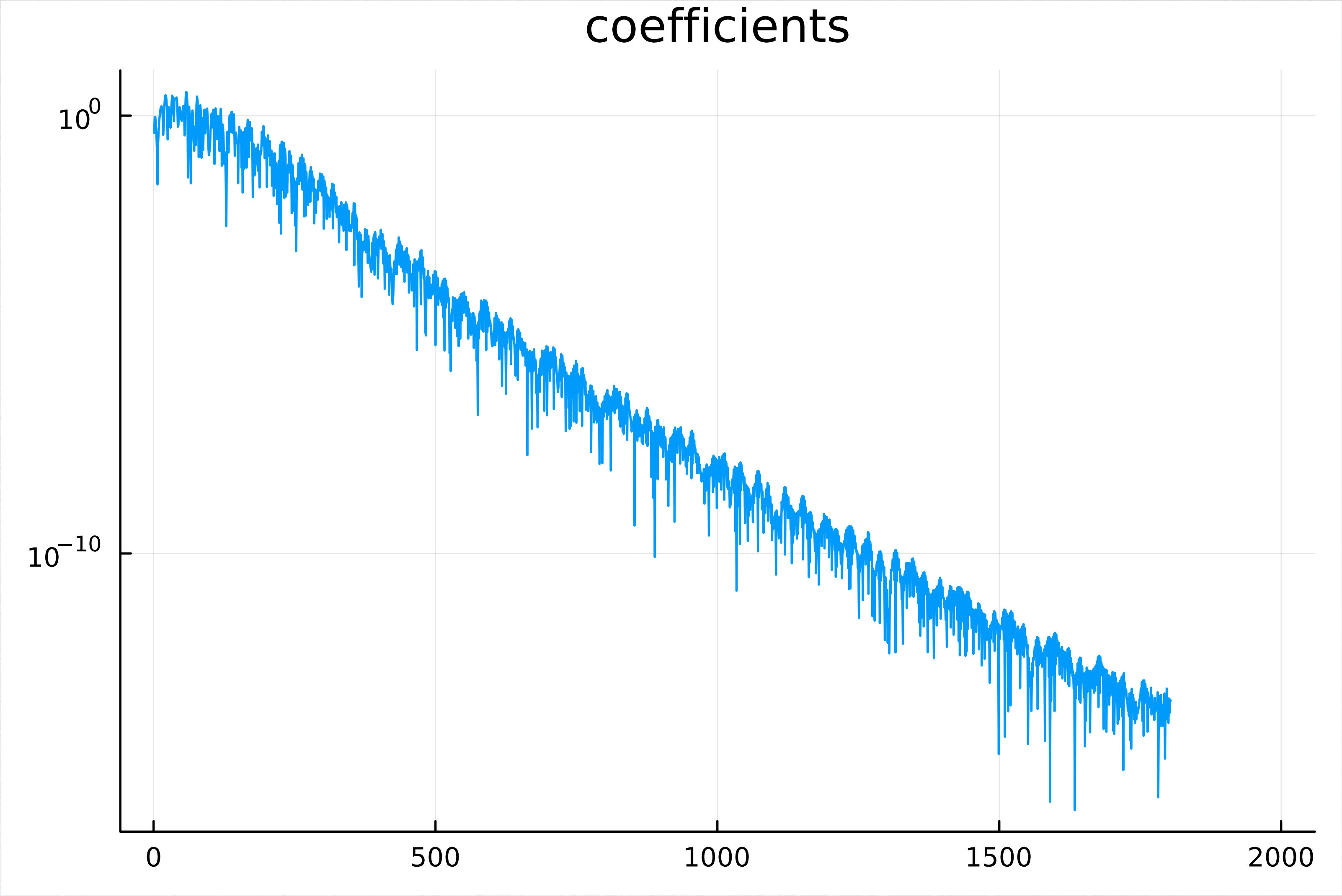
```
julia> n = 1000; x = ChebyshevGrid{1}(n); # 1000 Chebyshev points

julia> @btime log.(3.0 .- x); # evaluate complex log kernel at 1000 quadrature points
  36.028 μs (5 allocations: 8.06 KiB)

julia> @btime complexlogkernel(P[:,1:n], 3.0); # evaluate log kernel of 1000 Legendre polynomials, far from element (Olver's)
  1.915 μs (13 allocations: 24.28 KiB)

julia> @btime complexlogkernel(P[:,1:n], 0.1+0im); # evaluate log kernel of 1000 Legendre polynomials, on element (Forward)
  12.845 μs (12 allocations: 32.09 KiB)

julia> @btime complexlogkernel(P[:,1:n], 1.00001); # evaluate log kernel of 1000 Legendre polynomials, near element (Forward+Olver's)
  7.077 μs (12 allocations: 16.41 KiB)
```

Faster than even quadrature!
+
auto-detects "low rank"
(no need for FMM)

Needs Legendre coefficients
so use (one-off)
fast Legendre transform:
Alpert & Rokhlin (1991)
Townsend, Webb & SO (2018)

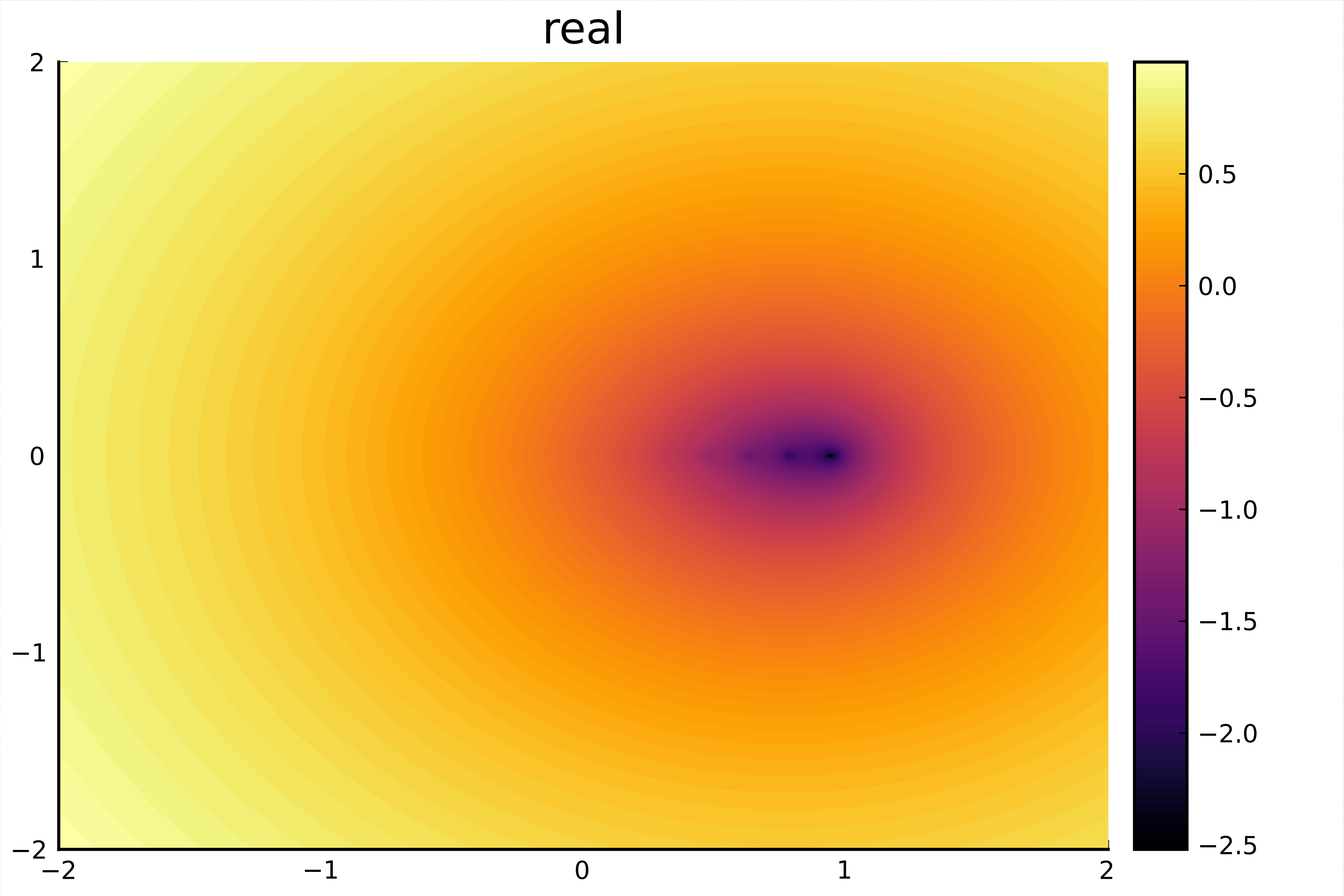


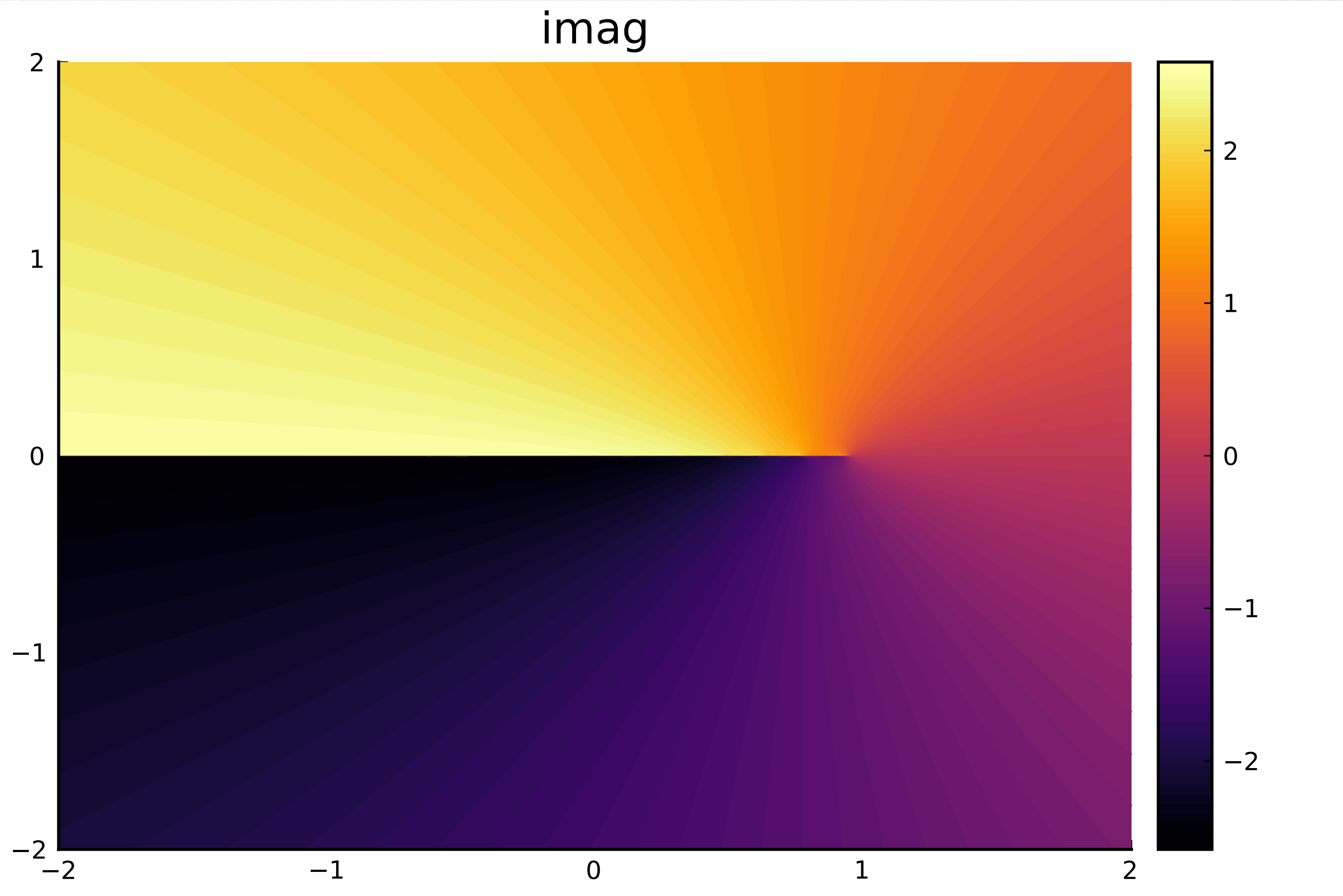
Fast Legendre Transform



Recurrences

$$\int_{-1}^{1} \log(z - t) f(t) \mathrm{d}t$$





- From Cauchy to Log kernel on an interval
- Log kernels on a rectangle
- Bessel kernels on a rectangle

Instead of

$$\iint_{\Omega} \log \|\mathbf{x} - \mathbf{t}\| f(\mathbf{t}) \mathrm{d}\mathbf{t}$$

Write in complex variables

$$\iint_{\Omega} \log |z - (s + \mathrm{i}t)| f(s,t) \mathrm{d}s\mathrm{d}t = \Re \iint_{\Omega} \log(z - (s + \mathrm{i}t)) f(s,t) \mathrm{d}s\mathrm{d}t$$

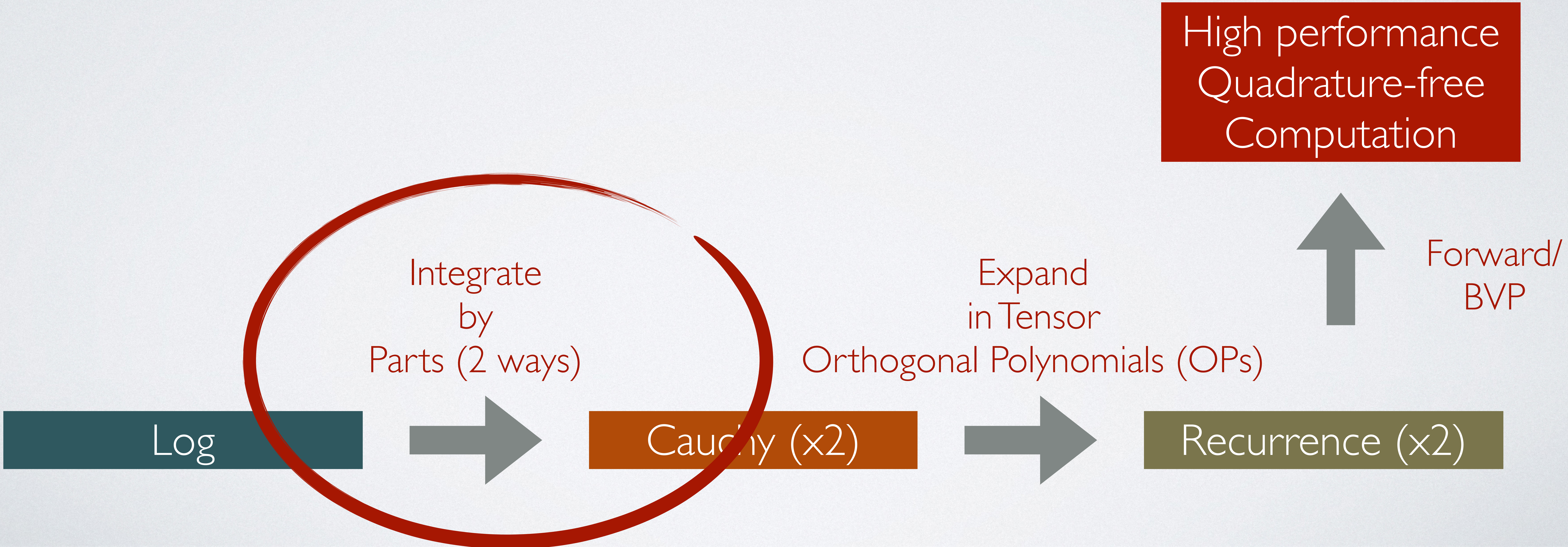
Expanding in Legendre reduces to computing

$$L_{kj}(z) := \iint_{\Omega} \log(z - (s + \mathrm{i}t)) P_k(s) P_j(t) \mathrm{d}s\mathrm{d}t$$

Rectangles

Log → Integrate by Parts (2 ways) → Cauchy (x2) → Expand in Tensor Orthogonal Polynomials (OPs) → Recurrence (x2)

RECTANGLES


High performance Quadrature-free Computation
Forward/ BVP
Integrate by Parts (2 ways)
Expand in Tensor Orthogonal Polynomials (OPs)
Log
Cauchy (x2)
Recurrence (x2)

Assume $k, j > 0$:

1D Log Kernel

$$\int_{-1}^{1}\int_{-1}^{1} \log(z-(s+\mathrm{i}t))P_k(s)P_j(t)\mathrm{d}s\mathrm{d}t = \int_{-1}^{1} P_j(t)\int_{-1}^{1} \log(z-\mathrm{i}t-s)P_k(s)\mathrm{d}s\mathrm{d}t$$

Assume $k, j > 0$:

$$\int_{-1}^{1}\int_{-1}^{1} \log(z-(s+\mathrm{i}t))P_k(s)P_j(t)\mathrm{d}s\mathrm{d}t = \int_{-1}^{1} P_j(t)\int_{-1}^{1} \log(z-\mathrm{i}t-s)P_k(s)\mathrm{d}s\mathrm{d}t$$

$$= -\frac{1}{2k}\int_{-1}^{1}\int_{-1}^{1} \frac{(1-s^2)P_{k-1}^{(1,1)}(s)P_j(t)}{z-(s+\mathrm{i}t)}\mathrm{d}s\mathrm{d}t$$

2D Cauchy/Stieltjes transform
of
weighted Jacobi ⊗ Legendre

Assume $k, j > 0$:

1D Log Kernel (not quite)

$$\int_{-1}^{1}\int_{-1}^{1} \log(z-(s+\mathrm{i}t))P_k(s)P_j(t)\mathrm{d}s\mathrm{d}t = \int_{-1}^{1} P_k(s)\int_{-1}^{1} \log(z-\mathrm{i}t-s)P_j(t)\mathrm{d}t\mathrm{d}s$$

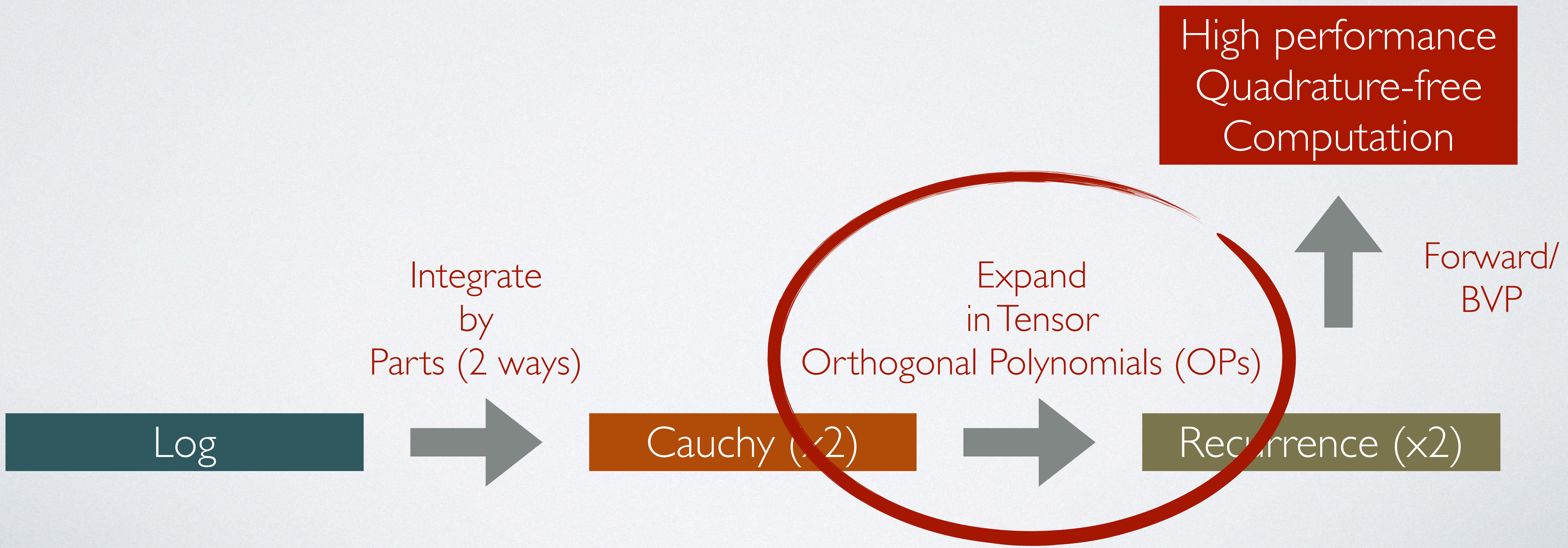
Assume $k, j > 0$, $\Re z > 1$:

$$\int_{-1}^{1}\int_{-1}^{1}\log(z-(s+\mathrm{i}t))P_k(s)P_j(t)\mathrm{d}s\mathrm{d}t = \int_{-1}^{1}P_k(s)\int_{-1}^{1}\log(z-\mathrm{i}t-s)P_j(t)\mathrm{d}t\mathrm{d}s$$
$$= \int_{-1}^{1}P_k(s)\int_{-1}^{1}\log(\mathrm{i}(s-z)-t)P_j(t)\mathrm{d}t\mathrm{d}s$$
$$= -\frac{\mathrm{i}}{2j}\int_{-1}^{1}\int_{-1}^{1}\frac{P_k(s)(1-t^2)P_{j-1}^{(1,1)}(t)}{z-(s+\mathrm{i}t)}\mathrm{d}s\mathrm{d}t$$

If $\Re z < 1$
then extra integral terms
appear, expressible
in terms of Jacobi polynomials,
treated as "forcing" term in
recurrence

2D Cauchy/Stieltjes transform
of
Legendre ⊗ weighted Jacobi

RECTANGLES


High performance
Quadrature-free
Computation
Forward/
BVP
Integrate
by
Parts (2 ways)
Expand
in Tensor
Orthogonal Polynomials (OPs)
Log
Cauchy (x2)
Recurrence (x2)

Assume $k, j > 0$:

$$zL_{kj}(z) = \int_{-1}^{1}\int_{-1}^{1} \frac{z(1-s^2)P_{k-1}^{(1,1)}(t)P_j(t)}{z-(s+\mathrm{i}t)}\mathrm{d}t$$

$$= \int_{-1}^{1}\int_{-1}^{1} \frac{(1-s^2)sP_{k-1}^{(1,1)}(s)P_j(t)}{z-(s+\mathrm{i}t)}\mathrm{d}s\mathrm{d}t + \mathrm{i}\int_{-1}^{1}\int_{-1}^{1} \frac{(1-s^2)P_{k-1}^{(1,1)}(s)tP_j(t)}{z-(s+\mathrm{i}t)}\mathrm{d}s\mathrm{d}t$$

Use Jacobi
3-term recurrence

Use Legendre
3-term recurrence

where $L_{kj}(z) := \int_{-1}^{1}\int_{-1}^{1} \log(z-(s+\mathrm{i}t))P_k(s)P_j(t)\mathrm{d}s\mathrm{d}t$

THEOREM 3.3. *The complex log transform of tensor-product Legendre polynomials satisfies two "5-point stencil"-like recurrence relations:*

$$\begin{aligned} zL_{kj} &= \frac{k-1}{2k+1}L_{k-1,j} + \frac{k+2}{2k+1}L_{k+1,j} + \mathrm{i}\left(\frac{j}{2j+1}L_{k,j-1} + \frac{j+1}{2j+1}L_{k,j+1}\right) + F_{kj}^{(1)}(z), \\ &= \frac{k}{2k+1}L_{k-1,j} + \frac{k+1}{2k+1}L_{k+1,j} + \mathrm{i}\left(\frac{j-1}{2j+1}L_{k,j-1} + \frac{j+2}{2j+1}L_{k,j+1}\right) + F_{kj}^{(2)}(z). \end{aligned}$$

where we take $L_{-1,j} = L_{k,-1} = 0$, *for*

$$M_k(z) := \int_{-1}^{1} P_k(t)\log(z - \mathrm{i}t)\mathrm{d}t$$

$$\begin{aligned} F_{0j}^{(1)}(z) &= \mathrm{i}\frac{M_{j+1}(z-1) + M_{j+1}(z+1) - M_{j-1}(z-1) - M_{j-1}(z+1)}{2j+1} \\ &\quad + \mu_j(z-1) + \mu_j(z+1), \\ F_{10}^{(1)}(z) &= -4/3, F_{kj}^{(1)}(z) = 0, \\ F_{k0}^{(2)}(z) &= \frac{L_{k+1}(z-\mathrm{i}) + L_{k+1}(z+\mathrm{i}) - L_{k-1}(z-\mathrm{i}) - L_{k-1}(z+\mathrm{i})}{2k+1} \\ &\quad + \lambda_k(z-\mathrm{i}) + \lambda_k(z+\mathrm{i}) + \beta_{kj}(z) \\ F_{01}^{(2)}(z) &= -4\mathrm{i}/3 + \beta_{kj}(z), F_{kj}^{(2)}(z) = \beta_{kj}(z) \end{aligned}$$

$$L_k(z) := \int_{-1}^{1} P_k(t)\log(z - t)\mathrm{d}t$$

where

$$\beta_{kj}(z) = 2\mathrm{i}\pi C_{j+1}^{(-1/2)}(x)\begin{cases} C_{k+2}^{(-3/2)}(x)/3 - \delta_{k0}x + \delta_{k1}/3 & z \in \Omega \\ 2x\delta_{k0} - 2\delta_{k1}/3 & x < -1 \text{ and } -1 < y < 1 \\ 0 & \text{otherwise} \end{cases}.$$

Two Sylvester equations:

$$zL = AL(z) + \mathrm{i}L(z)B^\top + F^{(1)}(z)$$
$$zL = BL(z) + \mathrm{i}L(z)A^\top + F^{(2)}(z)$$

Simple expressions in terms of 1D Log Kernel of Legendre and recurrences

where $L = (L_{kj}(z))_{k=0:\infty, j=0:\infty}$ and

$$A = \begin{pmatrix} 0 & 2 & & & \\ 0 & 0 & 1 & & \\ & 1/5 & 0 & 4/5 & \\ & & 2/7 & 0 & \ddots \\ & & & \ddots & \ddots \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 1 & & & \\ 1/3 & 0 & 2/3 & & \\ & 2/5 & 0 & 3/5 & \\ & & 3/7 & 0 & \ddots \\ & & & \ddots & \ddots \end{pmatrix}$$

RECTANGLES

High performance Quadrature-free Computation

Forward/ BVP

Integrate by Parts (2 ways)

Expand in Tensor Orthogonal Polynomials (OPs)

Log → Cauchy (x2) → Recurrence (x2)

Have two “5-point stencil” equations:

$$
\begin{array}{ccccc}
 & & A_{k,k-1}L_{k-1,j} & & \\
zL_{kj} = & \mathrm{i}B_{j,j-1}L_{k,j-1} & + & \mathrm{i}B_{j,j+1}L_{k,j+1} & +F^{(1)}_{kj}(z) \\
 & & A_{k,k+1}L_{k+1,j} & &
\end{array}
$$

Can cancel terms w/ linear combo of recurrences

$$
\begin{array}{ccccc}
 & & B_{k,k-1}L_{k-1,j} & & \\
zL_{kj} = & \mathrm{i}A_{j,j-1}L_{k,j-1} & + & \mathrm{i}A_{j,j+1}L_{k,j+1} & +F^{(2)}_{kj}(z) \\
 & & B_{k,k+1}L_{k+1,j} & &
\end{array}
$$

Forward: Take linear combination of recurrences to remove degrees of freedom. Eg to express $L_{k+1,j}$ or $L_{k,j+1}$ in terms of $L_{k-1,j}$, $L_{k,j-1}$ and $L_{k,j}$

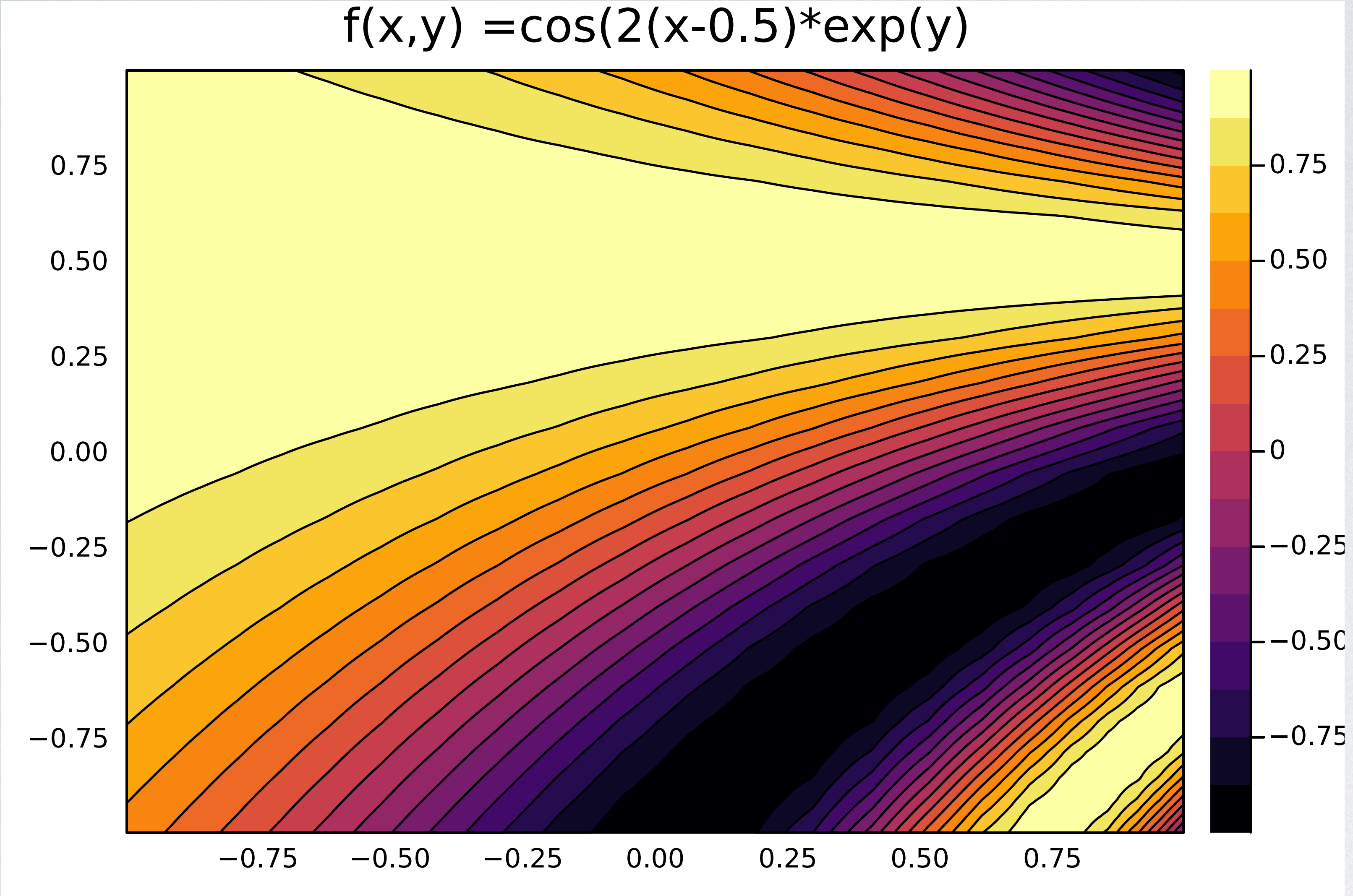
Have two “5-point stencil” equations:

Can cancel terms w/
linear combo of recurrences

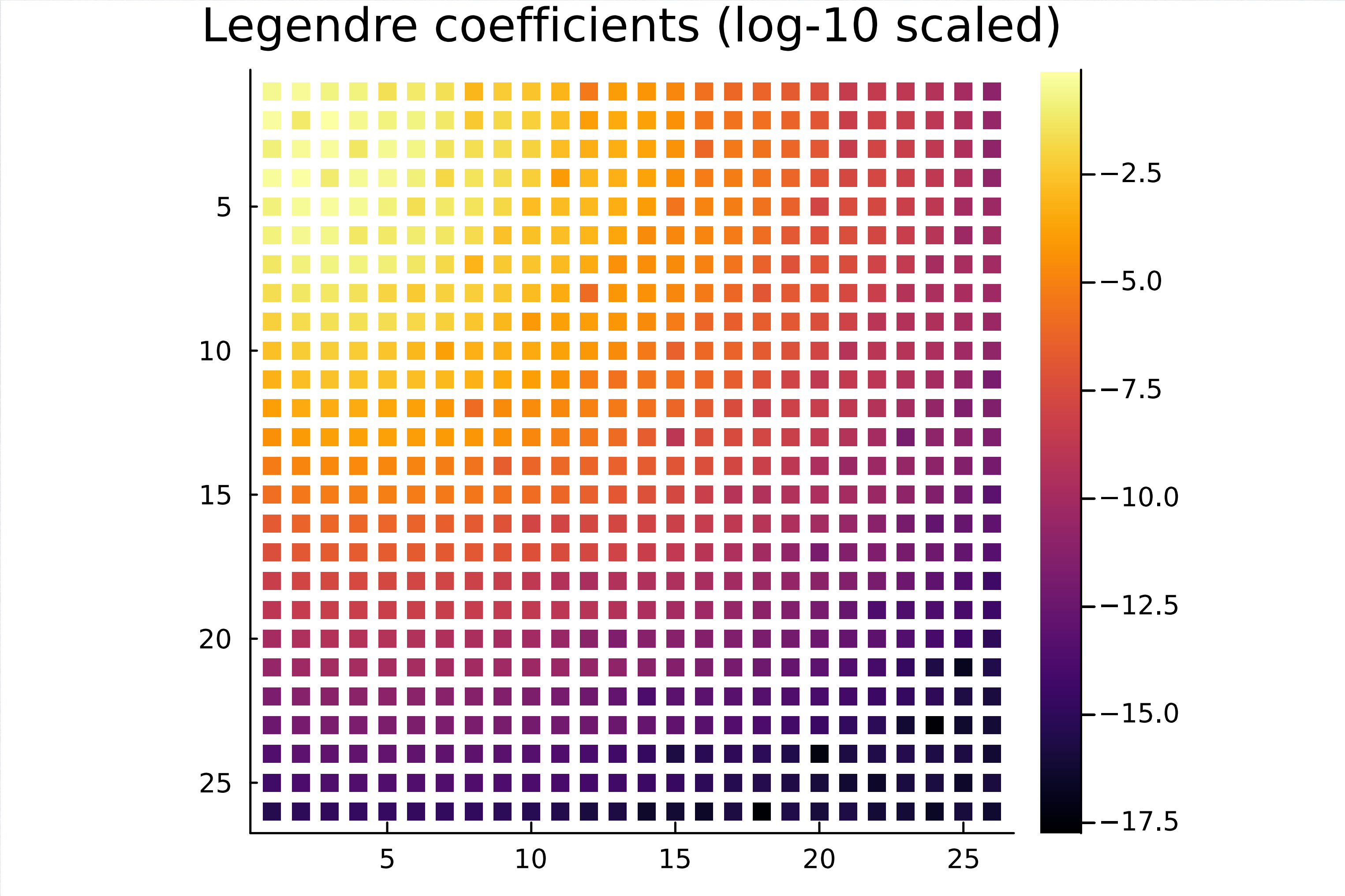
$$\begin{array}{ccccc} & & A_{k,k-1}L_{k-1,j} & & \\ zL_{kj} = & \mathrm{i}B_{j,j-1}L_{k,j-1} & + & \mathrm{i}B_{j,j+1}L_{k,j+1} & +F^{(1)}_{kj}(z) \\ & & A_{k,k+1}L_{k+1,j} & & \end{array}$$

$$\begin{array}{ccccc} & & B_{k,k-1}L_{k-1,j} & & \\ zL_{kj} = & \mathrm{i}A_{j,j-1}L_{k,j-1} & + & \mathrm{i}A_{j,j+1}L_{k,j+1} & +F^{(2)}_{kj}(z) \\ & & B_{k,k+1}L_{k+1,j} & & \end{array}$$

Forward: Take linear combination of recurrences to remove degrees of freedom.
Eg to express $L_{k+1,j}$ or $L_{k,j+1}$ in terms of $L_{k-1,j}$, $L_{k,j-1}$ and $L_{k,j}$

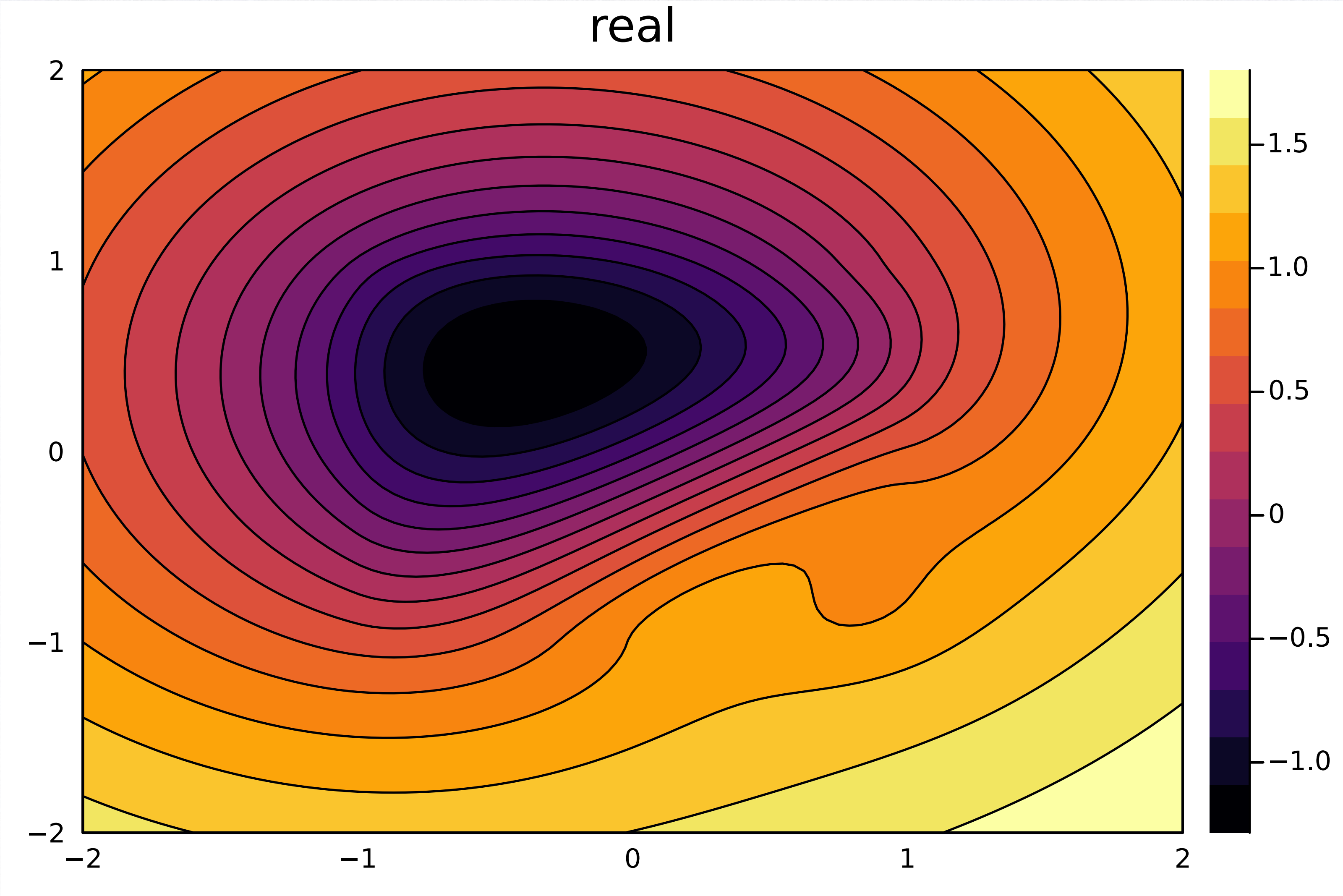


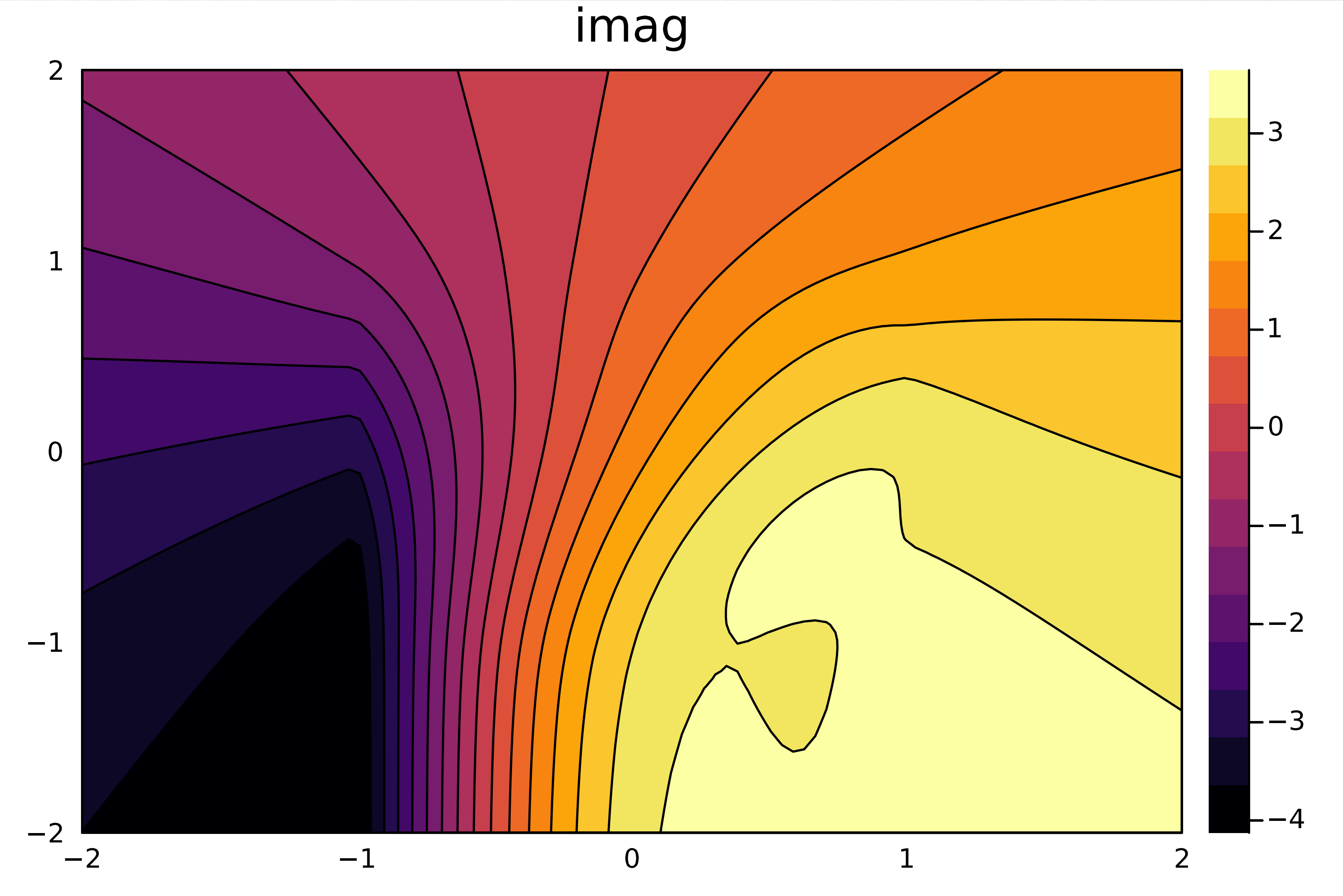
Fast Legendre Transform



Recurrences

$$\int_{-1}^{1}\int_{-1}^{1} \log(z-(s+\mathrm{i}t))f(s,t)\mathrm{d}s\mathrm{d}t$$





Forward substitution timing:

```
julia> n = 40; x = ChebyshevGrid{1}(n); # 40 Chebyshev points

julia> z = 1.01 + im/2; # point of evaluation

julia> @btime log.(z .- (x .+ im .* x')); # evaluate kernel at 40^2 quadrature points;
  112.513 μs (11 allocations: 25.38 KiB)

julia> @btime logkernelsquare(z, n); # evaluate log kernel of 40^2 tensor Legendre polymomials;
  59.827 μs (40 allocations: 85.62 KiB)
```

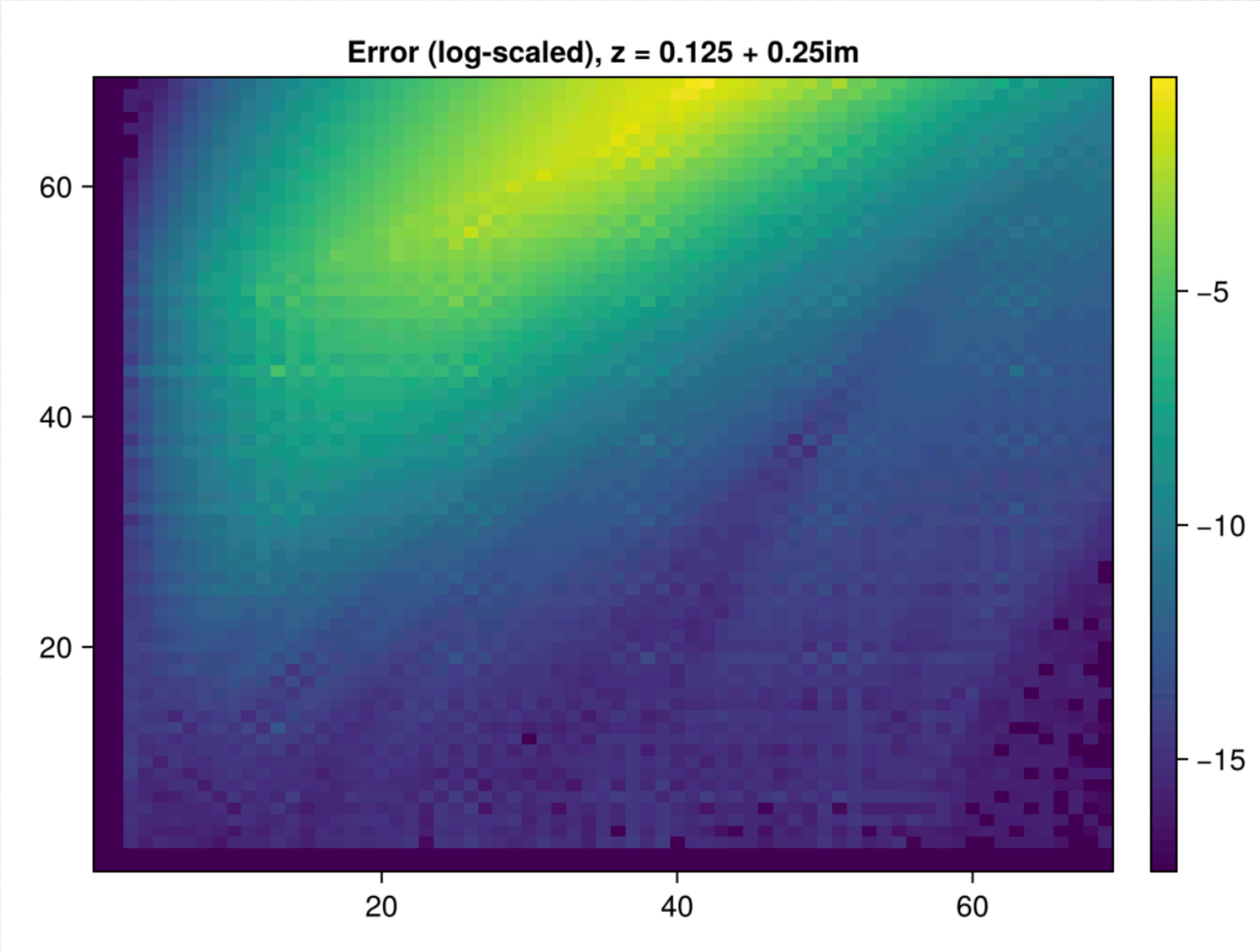
Faster than even quadrature!

But is it stable?

Good news! It even works with High-precision (BigFloat)

Bad news… it needs to at high order

Forward recurrence only numerically accurate for low orders ($p < 40$)



- From Cauchy to Log kernel on an interval
- Log kernels on a rectangle
- Bessel kernels on a rectangle

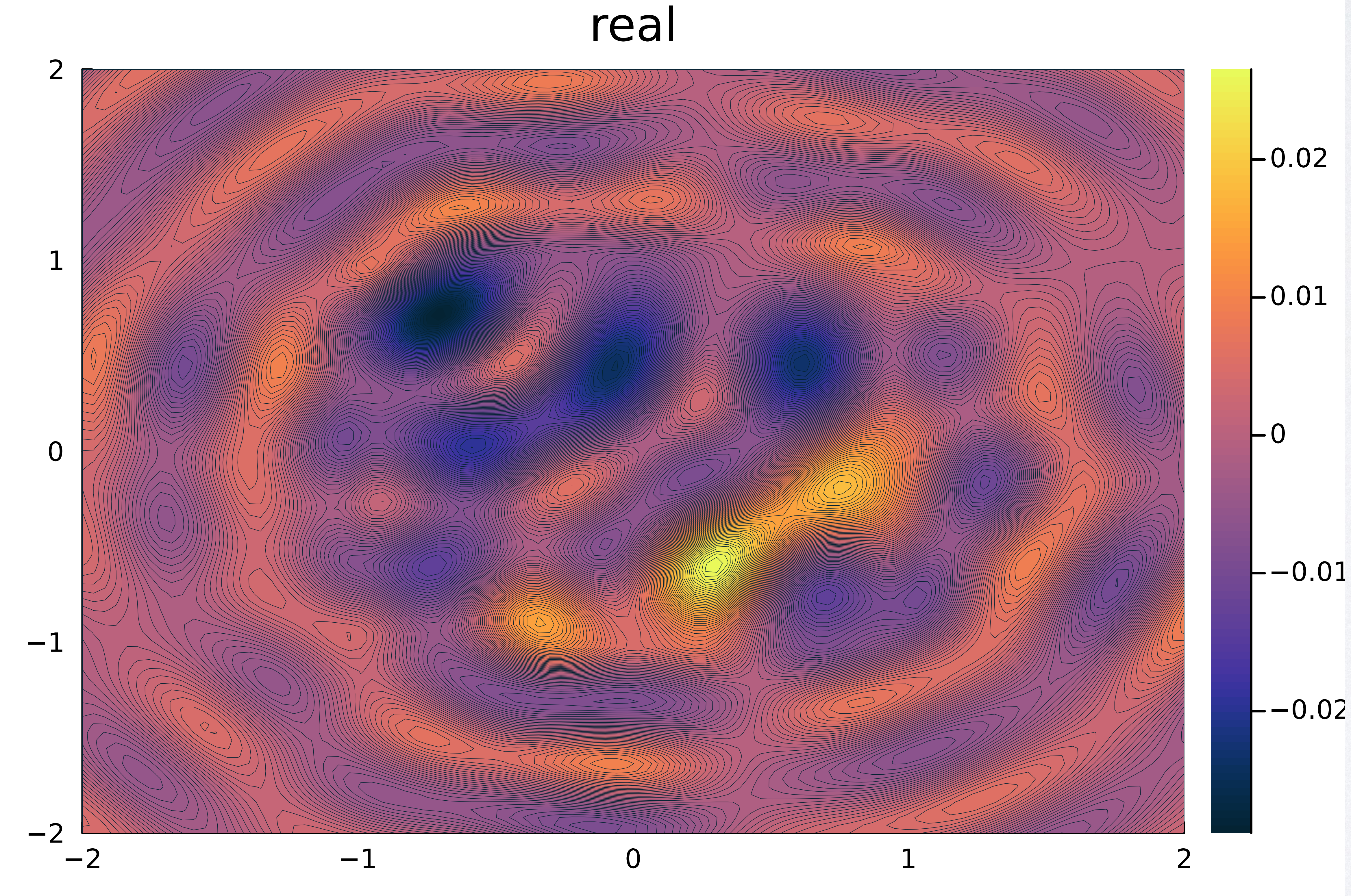
Rectangles

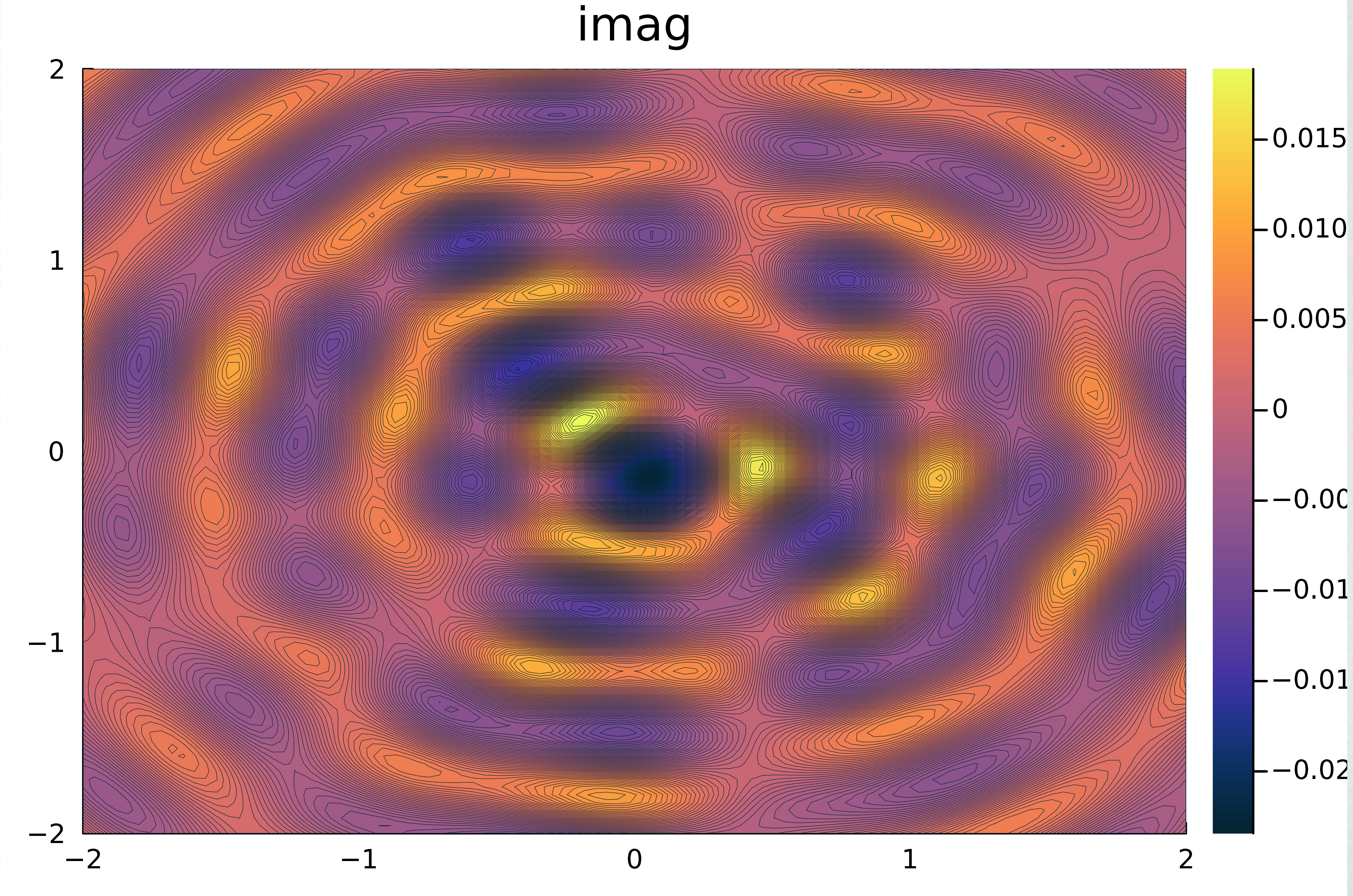

High performance
Quadrature-free
Computation
Bessel/other kernels
Riemann
functions
Integrate
by
Parts (2 ways)
Expand
in Tensor
Orthogonal Polynomials (OPs)
Forward/
BVP
Log
Cauchy (x2)
Recurrence (x2)

We can expand (a la Daan's talk):

$$\frac{\mathrm{i}}{4}H_0^{(1)}(k\|\mathbf{x}-\mathbf{t}\|) = -\frac{1}{2\pi}J_0(k\|\mathbf{x}-\mathbf{t}\|)\log\|\mathbf{x}-\mathbf{t}\| + B(\mathbf{x},\mathbf{t})$$

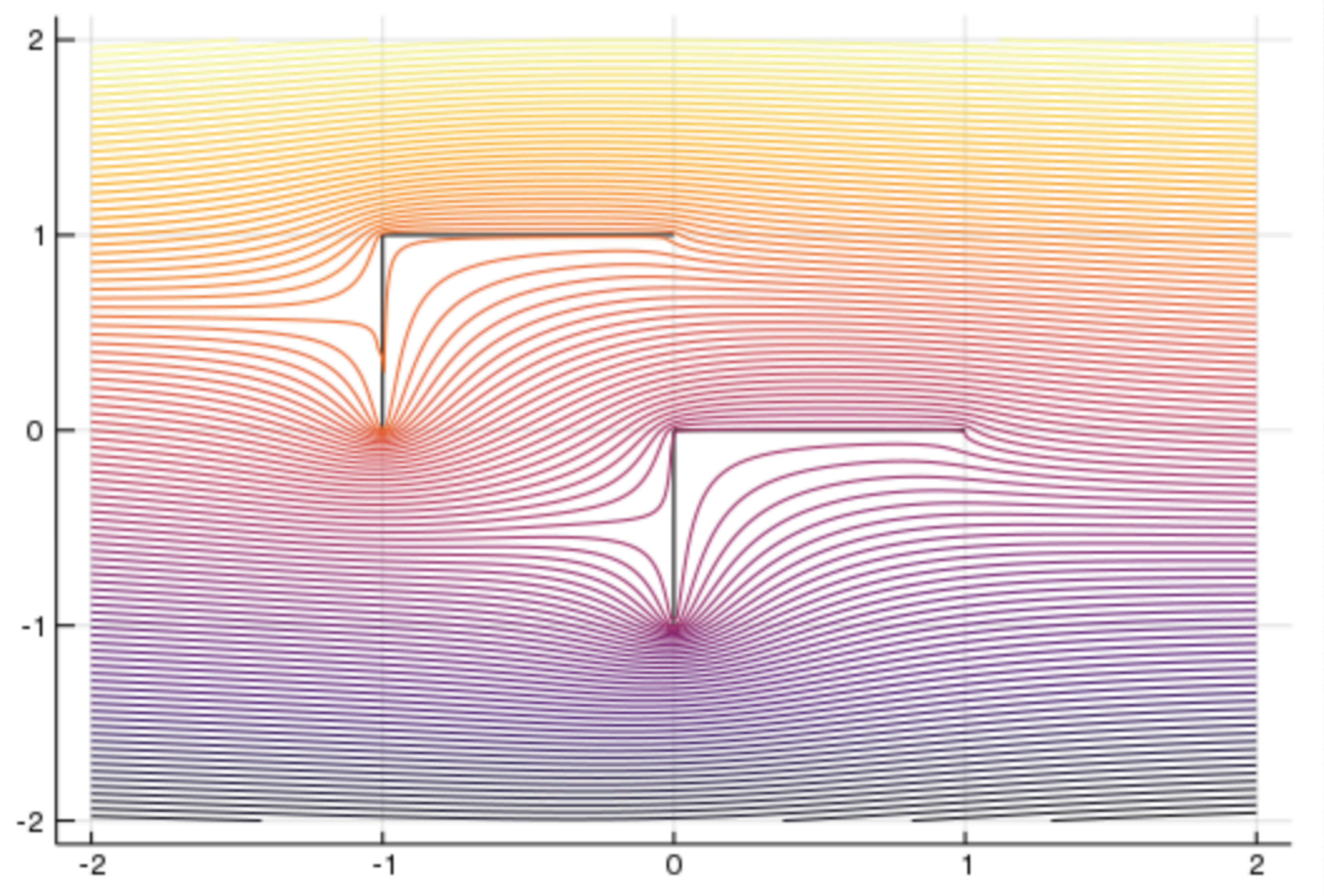
$$k = 10$$

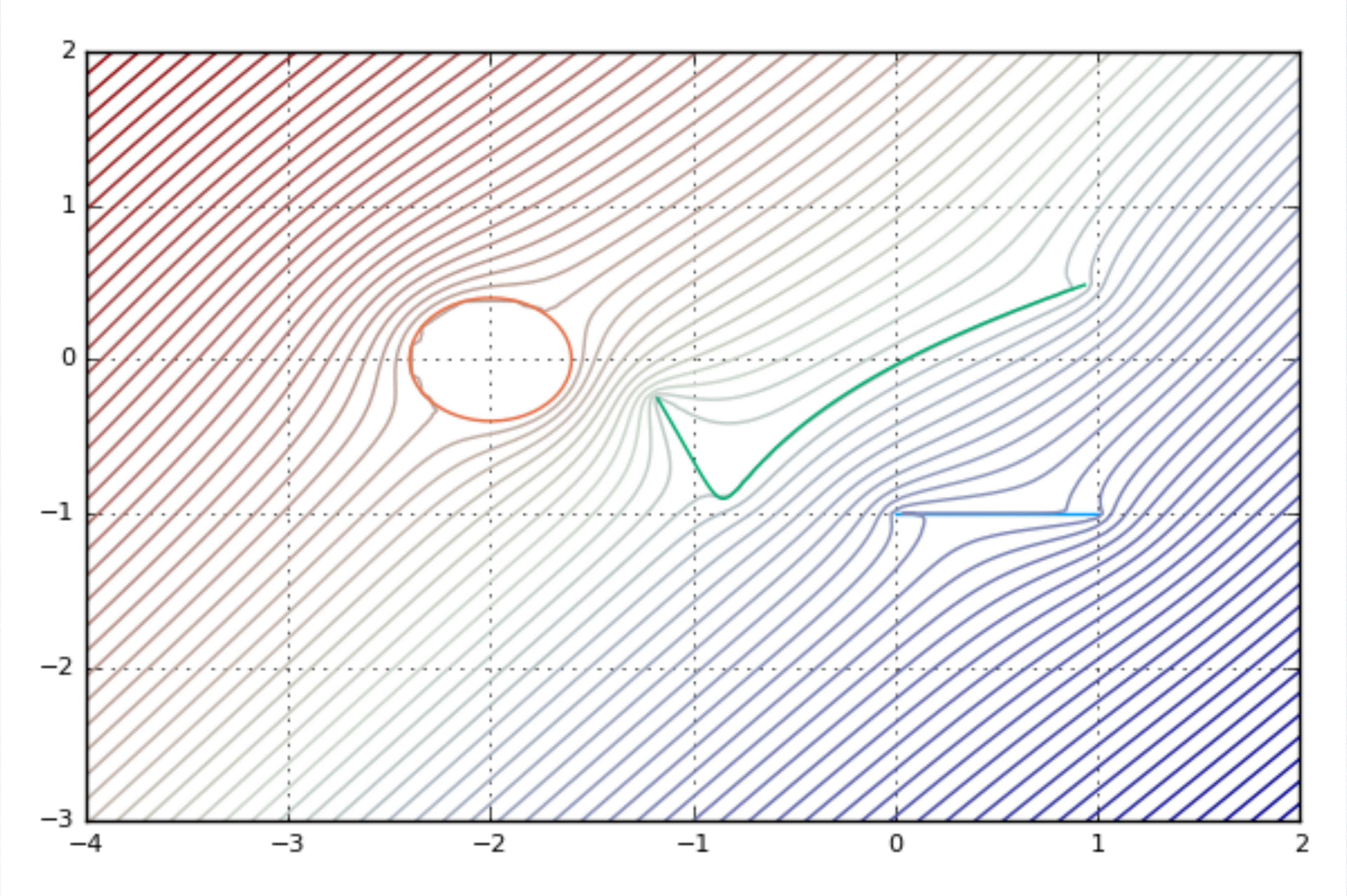




Big question: how to do mapped rectangles (kites/quads/etc.)?

In 1D we can use Change-of-Variable formulae
to compute on mapped intervals (smooth arcs/wedges/etc.):





Can this be extended to 2D?

Bigger question: 3D?