Orthogonal polynomials, singular integrals, and solving Riemann–Hilbert problems

Sheehan Olver, Imperial College, London

August 6, 2019

A *Riemann-Hilbert problem* (RH problem) is a boundary value problem in the complex plane, typically consisting of finding Φ from functions G and F that satisfies

1. *Analyticity*: $\Phi : (\mathbb{C} \backslash \Gamma) \to \mathbb{C}^{d \times d}$ is analytic off a contour Γ
2. *Asymptotic behaviour*: $\Phi(z) \sim I$ as $z \to \infty$
3. *Jump condition*:

$$\Phi_+(s) - G(s)\Phi_-(s) = F(s) \qquad \text{for} \qquad s \in \Gamma \tag{1}$$

where Φ_+ is the limit from the left and Φ_- is the limit from the right.

They arise in a number of applications including special functions, inverse spectral theory, integrable systems, asymptotics of orthogonal polynomials, and random matrices. Important to this workshop, Wiener-Hopf problems can be recast as RH problems.

In many applications, G depends on a parameter and asymptotic expansions of the solution to a RH problem can be found via a procedure called Deift-Zhou steepest descent, which consists of a sequence of jump factorisations, contour deformations, and local parametrices.

What if we are not in the asymptotic regime? Or if the rather involved steps of Deift-Zhou steepest descent are yet to be derived? We can instead solve RH problems numerically to find the solution to high accuracy.

In these lecture notes we explain how to do this, based on [SO 2011] and subsequent publications by Olver and Trogdon, see especially the book [Trogdon & SO 2015]. On the software side, this was originally implemented in RHPackage in Mathematica, but recently translated to Julia in RiemannHilbert.jl, which has led to a number of improvements. We demonstrate this software framework throughout the notes.

There are 5 parts:

1. Cauchy transforms and Plemelj theorem. We review the Plemelj theorem and how it can be used to explicitly compute Cauchy transforms. We also discuss how it leads to change-of-variables formulae for Cauchy transforms, that do not follow from integral change-of-variables.

2. Orthogonal polynomials and Cauchy transforms. We discuss how Cauchy transforms of orthogonal polynomials satisfy a simple recurrence that makes them easy to compute. This leads to an effective scheme for calculating Cauchy transforms.

3. Riemann-Hilbert problems. We show how RH problems can be reduced to singular integral equations involving Cauchy transforms and then solved using collocation. Special treatment happens for contours with junction points.

4. Deift-Zhou steepest descent. We show how oscillatory RH problems can be deformed in the complex plane to remove oscillations, and thereby retain asymptotic accuracy.

5. Sommerfeld radiation on the half-line. Following [Llewellyn-Smith & Luca 2019] show how these techniques can be used to numerically solve the classical problem of Sommerfeld radiation on the half-line, which is typically solved analytically via Wiener-Hopf methods. These techniques apply to wider classes of Wiener-Hopf problems not directly solvable.

1 Cauchy transforms and Plemelj theorem

Definition (Cauchy transform) The Cauchy transform over a contour Γ is denoted

$$\mathcal{C}_\Gamma f(z) := \frac{1}{2\pi \mathrm{i}} \int_\Gamma \frac{f(t)}{t-z} \mathrm{d}t. \tag{2}$$

The Cauchy transform is analytic for $z \notin \Gamma$. For reasonable f it has well-defined left and right limits

$$\begin{aligned} \mathcal{C}_\Gamma^+ f(x) &= \lim_{\substack{z \to x \\ \text{from the left}}} \mathcal{C}_\Gamma f(z) \\ \mathcal{C}_\Gamma^- f(x) &= \lim_{\substack{z \to x \\ \text{from the right}}} \mathcal{C}_\Gamma f(z) \end{aligned}$$

The subscript Γ will be omitted when implied from context.

Cauchy transforms are (of course) integrals, and so can be calculated via quadrature. that is, if

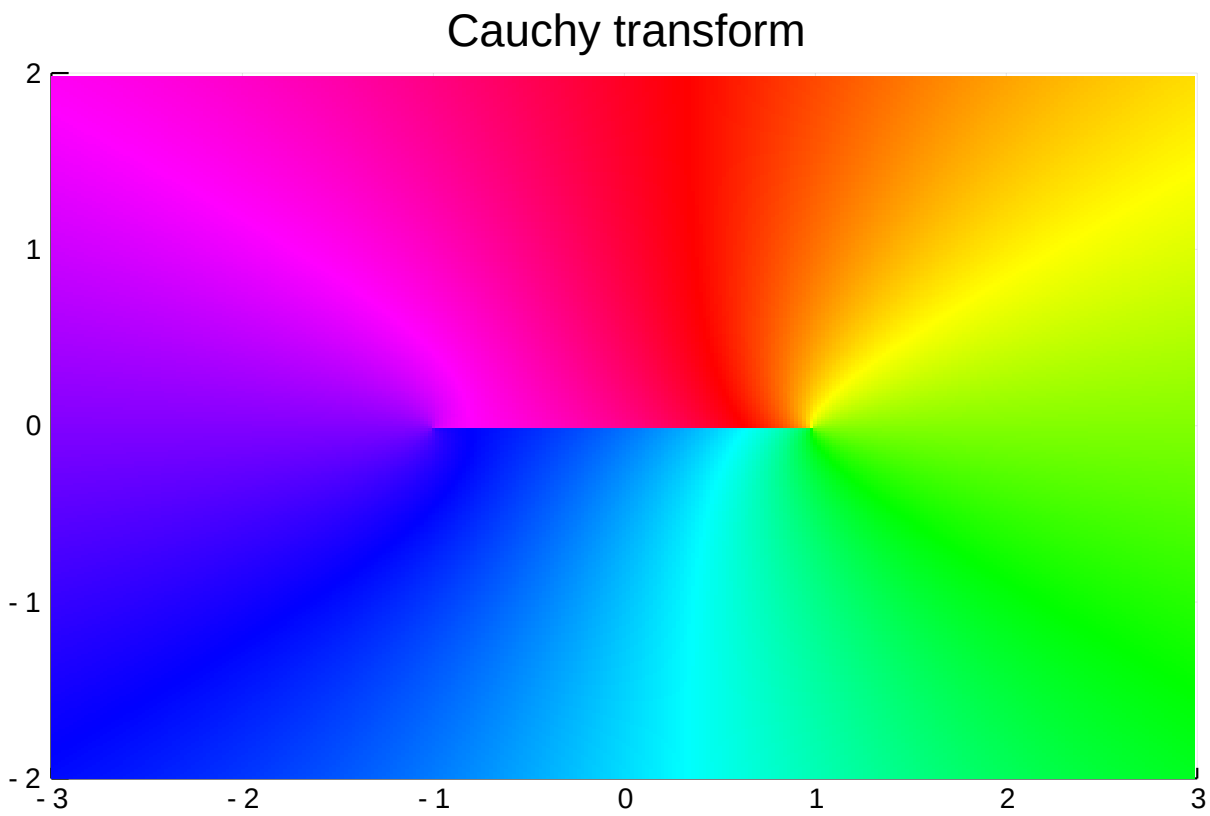
$$\int_\Gamma f(t)\mathrm{d}t \approx \sum_{j=1}^n w_j f(t_j)$$

then

$$\mathcal{C}_\Gamma f(z) \approx \frac{1}{2\pi \mathrm{i}} \sum_{j=1}^n w_j \frac{f(t_j)}{t_j - z} \tag{3}$$

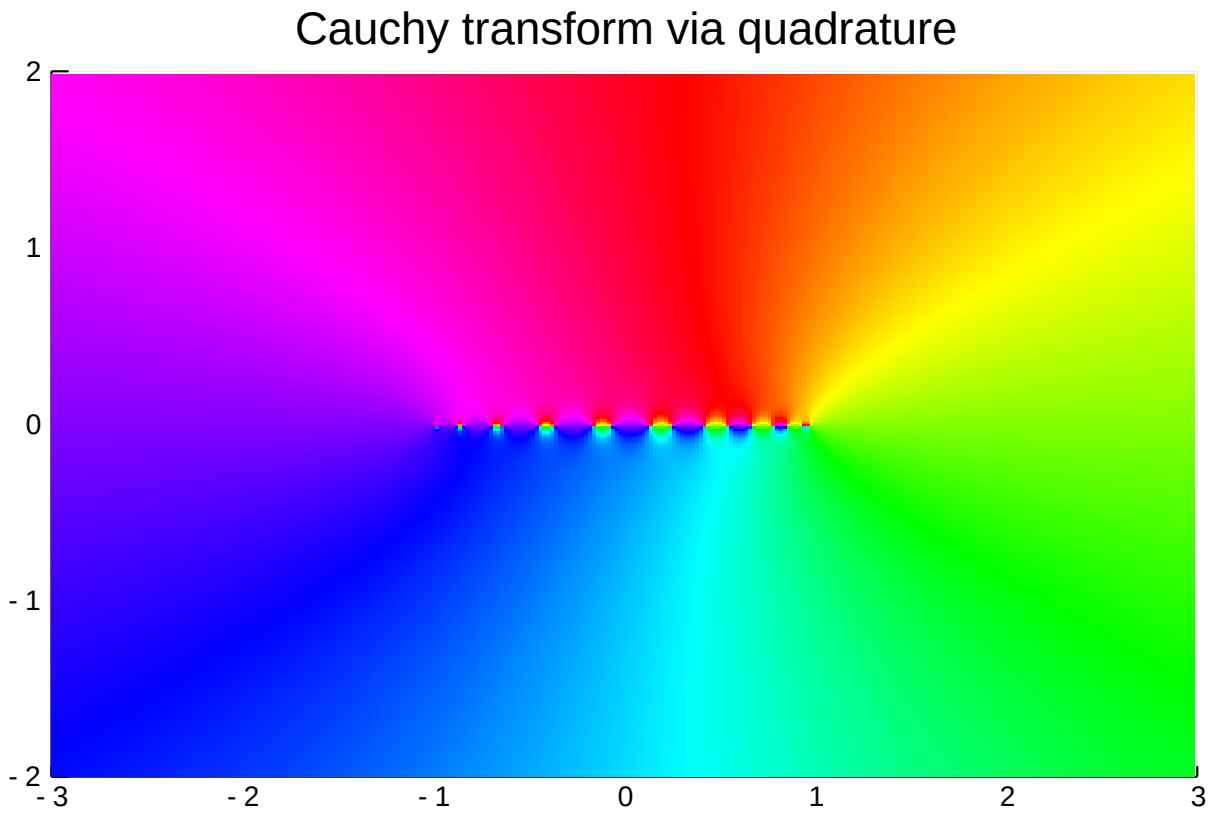
This is not a good idea: we're approximating something with a branch cut (left-hand side) by something with poles (right-hand side). This will be inaccurate on or near Γ. We can visualise this conveniently using a phase portrait. First we see that the true Cauchy transform (as approximated by the techniques we will develop) has a branch cut on the interval:

```
using ApproxFun, SingularIntegralEquations, ComplexPhasePortrait, FastGaussQuadrature,
    Plots
x = Fun(-1..1)
f = exp(x)
xx = range(-3,3;length=300)
yy = range(-2,2;length=300)
phaseplot(xx, yy, z -> cauchy(f,z); title="Cauchy transform")
```



On the other hand, Gauss quadrature results in poles along the branch cut:

```
s,w = gausslegendre(10)
phaseplot(xx, yy, z -> w'*(exp.(s)./(s .- z)); title="Cauchy transform via quadrature")
```



1.1 Plemelj Theorem

We will forgo quadrature and use the Plemelj theorem to derive numerical schemes for the Cauchy transform. A typical statement of Plemelj's theorem is the following (restricted to an interval for simplicity):

Theorem (Plemelj on the interval I) Suppose $(b-x)^\alpha(x-a)^\beta f(x)$ is differentiable on $[a,b]$, for $\alpha,\beta<1$. Then the Cauchy transform has the following properties:

1. It is analytic in $\bar{\mathbb{C}}\backslash[a,b]$.
2. It decays at infinity: $\mathcal{C}_{[a,b]}f(\infty)=0$.
3. It has weaker than pole singularities at a and b.
4. It has the subtractive jump:

$$\mathcal{C}^+_{[a,b]}f(x)-\mathcal{C}^-_{[a,b]}f(x)=f(x) \qquad \text{for} \qquad a<x<b.$$

What is less typically stated explicitly but is well-known is that these 4 properties of the Cauchy transform *uniquely* determine it:

Theorem (Plemelj on the interval II) Suppose $\phi(z)$ satsfies the following properties:

1. It is analytic in $\bar{\mathbb{C}}\backslash[a,b]$.
2. It has weaker than pole singularities at a and b.
3. It decays at infinity.
4. For f such that $(b-x)^\alpha(x-a)^\beta f(x)$ is differentiable in $[a,b]$ where $\alpha,\beta<1$, it has the subtractive jump:

$$\phi^+(x)-\phi^-(x)=f(x) \qquad \text{for} \qquad a<x<b.$$

Then $\phi(z)=\mathcal{C}_{[a,b]}f(z)$.

Sketch of Proof Consider $A(z)=\phi(z)-\mathcal{C}_{[a,b]}f(z)$ This is continuous (hence analytic) on (a,b) as $A^+(x)-A^-(x)=\phi^+(x)-\phi^-(x)-\mathcal{C}^+_{[a,b]}f(x)+\mathcal{C}^-_{[a,b]}f(x)=f(x)-f(x)=0$ Also, A has weaker than pole singularities at a and b, hence is analytic there as well: it's entire. Liouville's theorem therefore guarantees A is constant, and since it vanishes at ∞ the constant must be zero.

Q.E.D.

We now demostrate how Plemelj can be used to to re-derive several example Cauchy transforms.

Examples

$$\mathcal{C}1(z)=\frac{\log(z-1)-\log(z+1)}{2\pi\mathrm{i}}$$

since

$$\phi(z):=\frac{\log(z-1)-\log(z+1)}{2\pi\mathrm{i}}$$

satisfies the following:

1. It is clearly analytic off $(-\infty, 1]$ and is also analytic on $(-\infty, 0)$ since it is continuous

$$\phi_+(x) = \frac{\log_+(x-1) - \log_+(1+x)}{2\pi\mathrm{i}} = \frac{\log|x-1| - \log|1+x|}{2\pi\mathrm{i}} = \phi_-(x)$$

It is also analytic at infinity since it is analytic in a neighbourhood and has weaker than polynomial growth.

2. It has only log singularities which are weaker than poles.

3. It vanishes as $x \to +\infty$ due to $\log(x+c) = \log x + o(1)$,

implying

$$\phi(x) = \frac{\log(x-1) - \log(x+1)}{2\pi\mathrm{i}} = o(1).$$

Since $\phi(z)$ is analytic at infinity the result is valid any direction of approach.

4. It has the right jump:

$$\phi_+(x) - \phi_-(x) = \frac{\log_+(x-1) - \log_-(x-1)}{2\pi\mathrm{i}} = 1.$$

Similar arguments show the following:

$$\mathcal{C}[1/\sqrt{1-\diamond^2}](z) = \frac{\mathrm{i}}{2\sqrt{z-1}\sqrt{z+1}},$$
$$\mathcal{C}[\sqrt{1-\diamond^2}](z) = \frac{\sqrt{z-1}\sqrt{z+1} - z}{2\mathrm{i}}.$$

1.2 Hilbert transforms

There is also a well-known connection between Cauchy transforms and Hilbert transforms as part of Plemelj. We won't use this but state it here for completeness.

Definition (Hilbert transform) For $a < x < b$, define

$$\mathcal{H}_{[a,b]}f(x) = \frac{1}{\pi}⨍_a^b \frac{f(t)}{t-x}\mathrm{d}t.$$

Note sometimes the Hilbert transform is defined with opposite signs.

Theorem (Plemelj on the interval III) The Cauchy transform has the additive jump

$$\mathcal{C}^+_{[a,b]}f(x) + \mathcal{C}^-_{[a,b]}f(x) = -\mathrm{i}\mathcal{H}f(x).$$

Examples Recall (using $\diamond$ for the dummy variable)

$$\frac{1}{\sqrt{z-1}\sqrt{z+1}} = -2\mathrm{i}\mathcal{C}\left[\frac{1}{\sqrt{1-\diamond^2}}\right](z)$$

Therefore:

$$\mathcal{H}\left[\frac{1}{\sqrt{1-\diamond^2}}\right](x) = \mathrm{i}(\mathcal{C}^+ + \mathcal{C}^-)\left[\frac{1}{\sqrt{1-\diamond^2}}\right](x)$$
$$= \frac{1}{2\sqrt{x-1}_+\sqrt{x+1}} + \frac{1}{2\sqrt{x-1}_-\sqrt{x+1}} = 0$$

Similarly, recall

$$\sqrt{z-1}\sqrt{z+1} = z + 2\mathrm{i}\mathcal{C}[\sqrt{1-\diamond^2}](z)$$

Therefore,

$$\mathcal{H}\left[\sqrt{1-\diamond^2}\right](x) = \mathrm{i}(\mathcal{C}^+ + \mathcal{C}^-)\left[\sqrt{1-\diamond^2}\right](x) = -x.$$

One more example: since

$$\mathcal{C}1(z) = \frac{\log(z-1) - \log(z+1)}{2\pi\mathrm{i}},$$

we have

$$\mathcal{H}1(x) = \frac{\log_+(x-1) + \log_-(x-1) - 2\log(x+1)}{2\pi} = \frac{\log(1-x) - \log(x+1)}{\pi}.$$

1.3 Change-of-variable formulae

Plemelj allows us to deduce Cauchy transforms on simple rationally mapped domains. Examples include geometries like arcs, circles, and polynomial contours. Furhermore, such maps have the ability to simplify singular behaviour. Note that these formulae are *not* typically equivalent to standard integration change-of-variables.

The simplest case is an affine transform:

Proposition (Affine transformation) Let $p(z) = a + bz$ and $\Gamma = p([-1,1]) = \{p(x) : -1 \le x \le 1\}$. Then

$$\mathcal{C}_\Gamma f(z) = \mathcal{C}_{[-1,1]}[f \circ p](p^{-1}(z))$$

Proof The right-hand side (1) is analytic off Γ, (2) has weaker than pole singularities, (3) decays at infinity and (4) satisfies

$$\mathcal{C}^+_{[-1,1]}[f \circ p](p^{-1}(s)) - \mathcal{C}^-_{[-1,1]}[f \circ p](p^{-1}(s)) = f(p(p^{-1}(s))).$$

Q.E.D.

This can be extended to a Mobius transformation by subtracting the behaviour at infinity:

Proposition (Mobius transformation) Let

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

be a Mobius tranformation and $\Gamma = r([-1,1]) = \{r(x) : -1 \le x \le 1\}$. Then

$$\mathcal{C}_\Gamma f(z) = \mathcal{C}_{[-1,1]}[f \circ r](r^{-1}(z)) - \mathcal{C}_{[-1,1]}[f \circ r](r^{-1}(\infty))$$

Polynomial maps represent a more substantial leap, in part due to them having existing multiple inverses. By summing over all inverses we obtain the result:

Proposition (Polynomial transformation) Let

$$p(x) = p_0 + \cdots + p_d x^d$$

be a degree d polynomial and denote its d inverses as $p_j^{-1}(z)$: that is,

$$p(p_j^{-1}(z)) = z.$$

Assume p on $[-1,1]$ is one-to-one. For $\Gamma = p([-1,1]) = \{p(x) : -1 \le x \le 1\}$, we have

$$\mathcal{C}_\Gamma f(z) = \sum_{j=1}^{d} \mathcal{C}_{[-1,1]}[f \circ p](p_j^{-1}(z))$$

Sketch of Proof Define the $d \times d$ companion matrix associated with $p(x) - z$ as

$$C(z) = \begin{pmatrix} 0 & 1 & & \\ & & \ddots & 1 \\ (z-p_0)/p_d & -p_1/p_d & \cdots & -p_{d-1}/p_d \end{pmatrix}$$

for which $p_j^{-1}(z)$ are the d eigenvalues. Then for any contour γ surrounding the eigenvalues functional calculus gives

$$\phi(z) := \sum_{j=1}^{d} \mathcal{C}_{[-1,1]}[f \circ p](p_j^{-1}(z)) = \frac{1}{2\pi \mathrm{i}} \mathrm{Tr} \int_\gamma \mathcal{C}[f \circ p](\zeta)(\zeta I - C(z))^{-1} \mathrm{d}\zeta$$

This relationship ensures $\phi(z)$ is analytic off Γ. The assumption that p is one-to-one means for $s \in \Gamma$ there is a unique root $p_1^{-1}(s) \in [-1,1]$. We therefore have:

$$\phi_+(s) - \phi_-(s) = \mathcal{C}[f \circ p]^+(p_1^{-1}(s)) - \mathcal{C}[f \circ p]^-(p_1^{-1}(s)) = f(p(p_1^{-1}(s))) = f(s).$$

Q.E.D.

This construction extends to rational maps. In particular, Llewellyn-Smith & Luca 2019 use this construction with a special map to evaluate Cauchy transforms of functions on the real line that only decay like $x^{-1/2}$ by using a rational map to $[-1,1]$ in a way that transforms sqrt-decay into smoothness.

Proposition (Llewellyn-Smith & Luca 2019) Let

$$r(x) = \frac{x + x^3}{(1-x^2)^2}$$

and define its four inverses

$$r_1^{-1}(\alpha) = \frac{1+\sqrt{1+16\alpha^2} - sqrt(2)\sqrt{1+\sqrt{1+16\alpha^2}}}{4\alpha}$$
$$r_2^{-1}(\alpha) = 1/r_1^{-1}(\alpha), r_3^{-1}(\alpha) = \frac{-c+\sqrt{c^2-4}}{2}, r_4^{-1}(\alpha) = \frac{-c-\sqrt{c^2-4}}{2},$$

where $c = r_1^{-1}(\alpha) + 1/r_1^{-1}(\alpha) - 1/\alpha$. Then assuming f decays sufficiently fast and is sufficiently regular,

$$C_{\mathbb{R}} f(z) = \sum_{j=1}^{4} C_{[-1,1]}[f \circ r](r_j^{-1}(z)) - 2C_{[-1,1]}[f \circ r](1) - 22C_{[-1,1]}[f \circ r](-1).$$

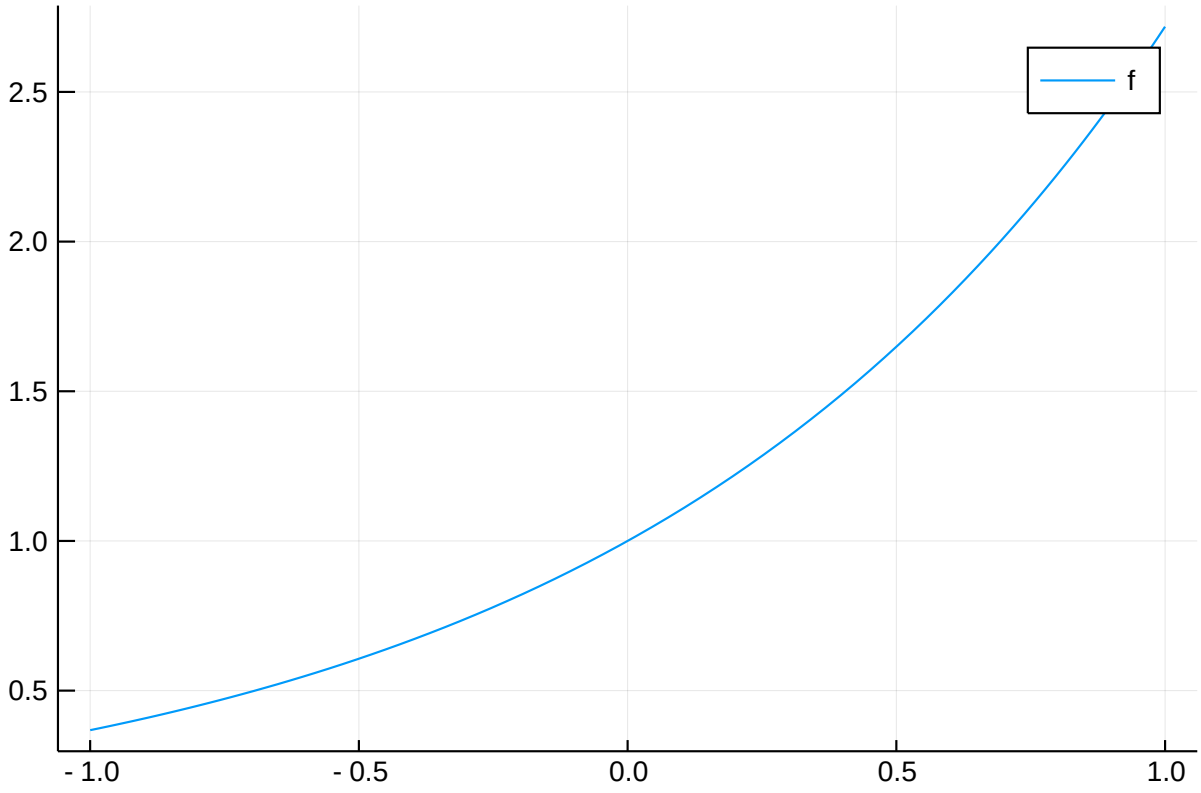
Here we are using the fact that if $f(r(x))$ decays at least algebraically at ± 1 then the Cauchy transform exists at these points in a classical integral.

1.4 Software demo

SingularIntegralEquations.jl is a Julia package for computing Cauchy transforms and other singular integrals, as well as solving singular integral equations. It builds on the ApproxFun.jl package for manipulating functions and solving differential equations, which is similar in spirit to Chebfun. Here we demonstrate how the package can be used for calculating and plotting Cauchy transforms, to get a visual demonstration of above.

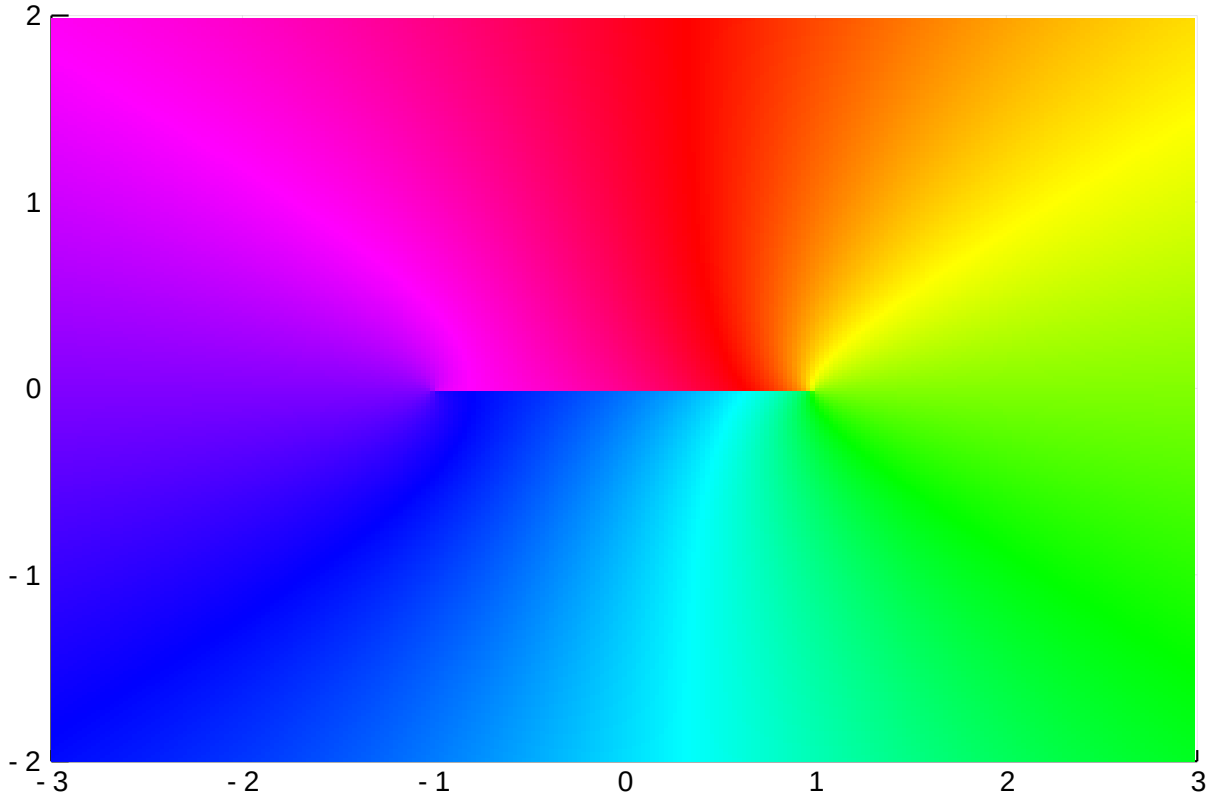
We start with an arbitrary function:

```
x = Fun(-1..1)              # Create the identity function on the unit interval -1..1
f = exp(x)                  # exp(x) on -1..1
plot(f; label="f")          # plot f
```



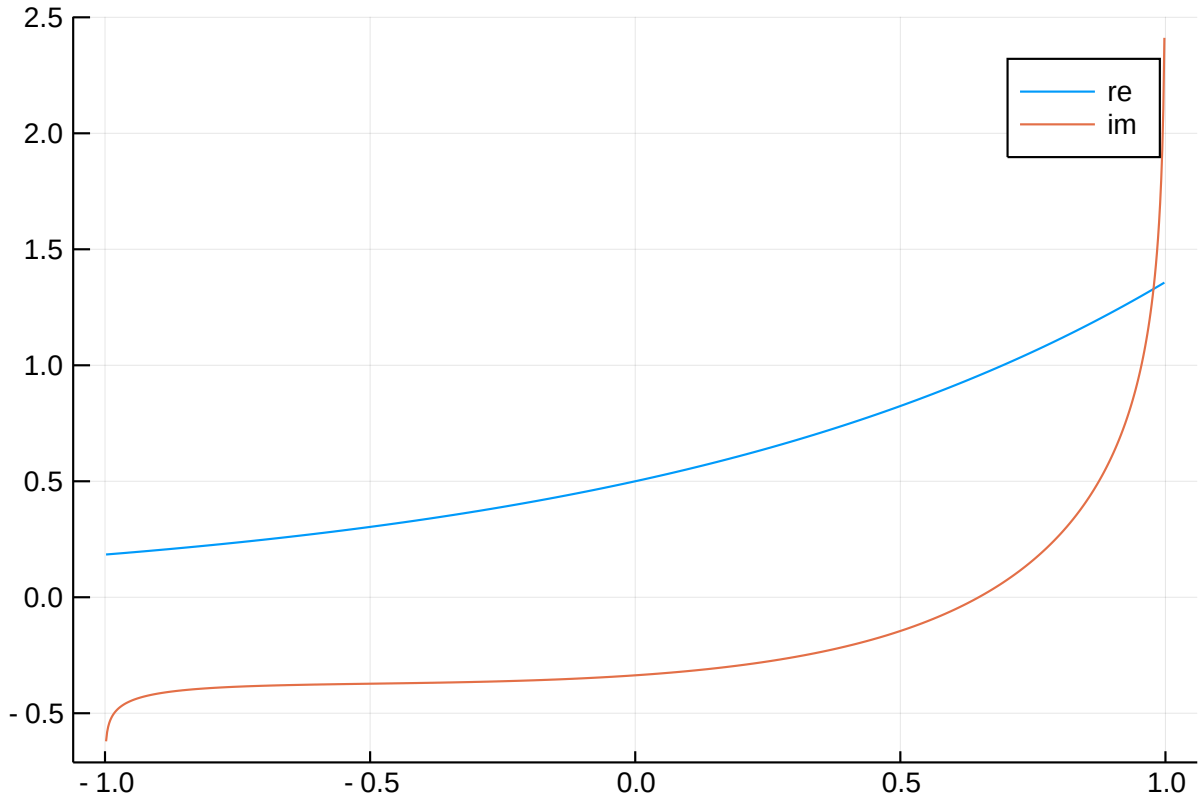
It's Cauchy transform can be visualised using the `cauchy` command from SingularIntegralEquations.jl. We depict it as a phase plot a la Wegert, as implemented in ComplexPhasePortrait.jl:

```
xx,yy = range(-3,3;length=250), range(-2,2;length=250) # plotting grid
phaseplot(xx, yy, z -> cauchy(f,z))
```



As seen above, the Cauchy transform has a branch cut on $[-1, 1]$. We can plot the left/right limits on $[-1, 1]$ via:

```
xx = range(-1,1;length=1000) # plotting grid
plot(xx, real.(cauchy.(f, xx .+ 0im)); label="re")
plot!(xx, imag.(cauchy.(f, xx .+ 0im)); label="im")
```



Using the change-of-variable formulae we can calculate Cauchy transforms over other domains. Here we approximate the Cauchy transform over a simple smooth arc that is approximated by a polynomial:

```
Γ = IntervalCurve(Fun(x -> sqrt(x^2+1) + im*x, -1.0..1))
s = Fun(Γ)
f = exp(s)
z = 2.0+0.1im
cauchy(f,z) , sum(f/(s-z))/(2π*im) # compare with quadrature
```

```
(-0.7515318502119724 + 0.025862882280999772im, -0.7515318502119764 + 0.0258
6288228100456im)
```

We finally demonstrate the Llewellyn-Smith-Luca map on a function with only sqrt-decay. This is implemented in the experimental (and largely empty) package WienerHopf.jl as `SqrtLine`:

```
using WienerHopf
f = Fun(x -> 1/(x^2+1) + 1/sqrt(x-2im), SqrtLine()) # arbitray function decaying like
    sqrt(x)
cauchy(f,1+im) # Matches Mathematica's NIntegrate of 0.2+0.1im (and _not_ Mathematica's
    wrong Integrate answer...)
```

```
0.1999999999999949 + 0.09999999999998438im
```

2 Orthogonal polynomials and Cauchy transforms

In this section we see how expansion of a function into a suitable orthogonal polynomial basis leads to an effective scheme for calculating Cauchy transforms.

Definition (Orthogonal polynomial) Let $p_0(x), p_1(x), p_2(x), \ldots$ be a sequence of polynomials such that $p_n(x)$ is exactly degree n, that is,

$$p_n(x) = k_n x^n + O(x^{n-1}) \tag{4}$$

where $k_n \neq 0$. Let $w(x)$ be a continuous weight function on a (possibly infinite) interval (a, b): that is $w(x) \geq 0$ for all $a < x < b$. This induces an inner product

$$\langle f, g \rangle := \int_a^b f(x)g(x)w(x)\mathrm{d}x \tag{5}$$

We say that $\{p_0, p_1, \ldots\}$ are *orthogonal with respect to the weight* w if

$$\langle p_n, p_m \rangle = 0 \qquad \text{for} \qquad n \neq m. \tag{6}$$

Theorem (three-term recurrence) Suppose $\{p_n(x)\}$ are a family of orthogonal polynomials w.r.t. a weight $w(x)$. Then there exists constants a_n, $b_n \neq 0$ and $c_n \neq 0$ such that

$$\begin{aligned} xp_0(x) &= a_0 p_0(x) + b_0 p_1(x) \\ xp_n(x) &= c_n p_{n-1}(x) + a_n p_n(x) + b_n p_{n+1}(x) \end{aligned}$$

We will use *classical orthogonal polynomials* which have additional properties that make them particularly easy to work with. Examples are

1. Legendre polynomials $P_n(x)$ ortho. w.r.t. $w(x) = 1$ on $[-1, 1]$
2. Chebyshev polynomials (1st kind) $T_n(x)$ ortho. w.r.t. $w(x) = 1/\sqrt{1-x^2}$ on $[-1, 1]$
3. Chebyshev polynomials (2nd kind) $U_n(x)$ ortho. w.r.t. $w(x) = \sqrt{1-x^2}$ on $[-1, 1]$
4. Jacobi polynomials $P_n^{(a,b)}(x)$ ortho. w.r.t. $w(x) = (1-x)^a(1+x)^b$ on $[-1, 1]$

Orthogonal polynomials form a natural basis for expanding functions, and in this context we consider weighted approximations:

$$f(x) \approx w(x) \sum_{k=0}^{n-1} f_k p_k(x)$$

Such expansions are easily calculated for classical orthogonal polynomials, either via Gaussian quadrature, interpolation, or using fast transforms. Once an expansion is known, we have a natural approximation to the Cauchy transform:

$$\mathcal{C}f(z) \approx \sum_{k=0}^{n-1} f_k \mathcal{C}[wp_k](x)$$

The approximation on the right has a jump on $[-1,1]$ matching the true Cauchy transform. In fact, in a certain sense we achieve uniform convergence (though we omit details) we can approximate the Cauchy transform up to and on the jump.

The key feature is that we have reduced calculation of general Cauchy transforms to that of weighted orthogonal polynomials. It turns out that Cauchy transforms satisfy a simple recurrence that makes this extremely straightforward:

Theorem (Three-term recurrence Cauchy transform of weighted OPs)

$$C_n(z) := \mathcal{C}[p_n w](z)$$

satisfy the same recurrence relationship as $p_n(x)$ for $n = 1, 2, \ldots$:

$$\begin{aligned} zC_0(z) &= a_0 C_0(z) + b_0 C_1(z) - \frac{1}{2\pi \mathrm{i}} \int_a^b w(x) \mathrm{d}x \\ zC_n(z) &= c_n C_{n-1}(z) + a_n C_n(z) + b_n C_{n+1}(z) \end{aligned}$$

Proof

$$\begin{aligned} zC_n(z) &= \frac{1}{2\pi \mathrm{i}} \int_a^b \frac{z p_n(x) w(x)}{x - z} \mathrm{d}x = \frac{1}{2\pi \mathrm{i}} \int_a^b \frac{(z - x) p_n(x) w(x)}{x - z} \mathrm{d}x + \int_a^b \frac{x p_n(x) w(x)}{x - z} \mathrm{d}x \\ &= -\frac{1}{2\pi \mathrm{i}} \int_a^b p_n(x) w(x) \mathrm{d}x + \int_a^b \frac{c_n p_{n-1}(x) + a_n p_n(x) + b_n p_{n+1}(x) w(x)}{x - z} \mathrm{d}x \\ &= -\frac{1}{2\pi \mathrm{i}} \int_a^b p_n(x) w(x) \mathrm{d}x + c_n C_{n-1}(z) + a_n C_n(z) + b_n C_{n+1}(z) \end{aligned}$$

when $n > 0$, the integral term disappears.

Q.E.D.

Thus if we know $C_0(z) = \mathcal{C} w(z)$ and $\int_a^b w(x) \mathrm{d}x$, we can determine $C_n(z)$ by solving the lower triangular system:

$$\begin{pmatrix} 1 & & & & \\ a_0 - z & b_0 & & & \\ c_1 & a_1 - z & b_1 & & \\ & c_2 & a_2 - z & b_2 & \\ & & c_3 & a_3 - z & \ddots \\ & & & \ddots & \ddots \end{pmatrix} \begin{pmatrix} C_0(z) \\ C_1(z) \\ C_2(z) \\ C_3(z) \\ \vdots \end{pmatrix} = \begin{pmatrix} C_0(z) \\ \frac{1}{2\pi \mathrm{i}} \int_a^b w(x) \mathrm{d}x \\ 0 \\ 0 \\ \vdots \end{pmatrix} \tag{7}$$

Example (Chebyshev Cauchy transform)

Consider the Chebyshev case $w(x) = \frac{1}{\sqrt{1-x^2}}$, which satisfies $\int_{-1}^1 w(x) \mathrm{d}x = \pi$. Recall that

$$C_0(z) = \mathcal{C} w(z) = \frac{\mathrm{i}}{2\sqrt{z-1}\sqrt{z+1}} \tag{8}$$

Further, we have

$$xT_0(x) = T_1(x)$$
$$xT_n(x) = \frac{T_{n-1}(x)}{2} + \frac{T_{n+1}(x)}{2}$$

hence

$$zC_0(z) = C_1(z) - \frac{1}{2\mathrm{i}}$$
$$zC_n(z) = \frac{C_{n-1}(z)}{2} + \frac{C_{n+1}(z)}{2}.$$

In other words, we can compute $C_0(z), C_1(z), \ldots$ by solving

$$\begin{pmatrix} 1 & & & & \\ -z & 1 & & & \\ 1/2 & -z & 1/2 & & \\ & 1/2 & -z & 1/2 & \\ & & 1/2 & -z & \ddots \\ & & & \ddots & \ddots \end{pmatrix} \begin{pmatrix} C_0(z) \\ C_1(z) \\ C_2(z) \\ C_3(z) \\ \vdots \end{pmatrix} = \begin{pmatrix} \frac{\mathrm{i}}{2\sqrt{z-1}\sqrt{z+1}} \\ \frac{1}{2\mathrm{i}} \\ 0 \\ 0 \\ \vdots \end{pmatrix} \tag{9}$$

with forward substitution.

2.1 Hilbert transform of weighted orthogonal polynomials

Now consider the Hilbert transform of weighted orthogonal polynomials:

$$H_n(x) = \mathcal{H}_{[a,b]}[p_n w](x) = \frac{1}{\pi} \int_a^b \frac{p_n(t)w(t)}{t-x} \mathrm{d}t \tag{10}$$

Just like Cauchy transforms, the Hilbert transforms have

Corollary (Hilbert transform recurrence)

$$xH_0(x) = a_0 H_0(x) + b_0 H_1(x) - \frac{1}{\pi} \int_a^b w(x) \mathrm{d}x$$
$$xH_n(x) = c_n H_{n-1}(x) + a_n H_n(x) + b_n H_{n+1}(x)$$

Proof Recall

$$\mathcal{C}^+ f(x) + \mathcal{C}^- f(x) = -\mathrm{i}\mathcal{H} f(x). \tag{11}$$

Therefore, we have

$$C_n^+(x) + C_n^-(x) = -\mathrm{i}\mathcal{H}[wp_n](x). \tag{12}$$

Hence we have

$$xH_0(x) = \mathrm{i}x(C_0^+(x) + C_0^-(x)) = \mathrm{i}\left[a_0(C_0^+(x) + C_0^-(x)) + b_0(C_1^+(x) + C_1^-(x)) - \frac{1}{\pi\mathrm{i}}\int_a^b w(x)\mathrm{d}x\right]$$
$$= a_0H_0(x) + b_0H_1(x) - \frac{1}{\pi}\int_a^b w(x)\mathrm{d}x.$$

Other n follows by a similar argument. Q.E.D.

Example (weighted Chebyshev) For

$$H_n(x) = \frac{1}{\pi}\int_{-1}^{1} \frac{T_n(t)}{(t-x)\sqrt{1-t^2}}\mathrm{d}t \tag{13}$$

the recurrence gives us

$$xH_0(x) = H_1(x) - 1 \qquad \text{and} \qquad xH_n(x) = \frac{H_{n-1}(x)}{2} + \frac{H_n(x)}{2}.$$

In this case, we have $H_0(x) = \mathcal{H}[w](x) = 0$. Therefore, we can rewrite this recurrence as

$$H_1(x) = 1, \qquad xH_1(x) = \frac{H_2(x)}{2} \qquad \text{and} \qquad xH_n(x) = \frac{H_{n-1}(x)}{2} + \frac{H_n(x)}{2}$$

This is precisely the three-term recurrence satisfied by U_{n-1}! We therefore have

$$H_n(x) = U_{n-1}(x) \tag{14}$$

2.2 (F.W.J.) Olver's algorithm

While forward recurrence is an effective means of calculating Cauchy transforms, it runs into numerical stability issues if we are far from $[-1, 1]$: Cauchy transforms and OPs satisfy the same recurrence relationship (they are the *minimal* and *maximal* solutions) apart from the initial conditions, and any small perturbation to the initial condition for the minimal solution will pick up the maximal solution, which blow up as $n \to \infty$. On the other hand, $C_n(z) \to 0$ as $n \to \infty$ for z off the support of w.

There are effective methods for calculating minimal solutions to recurrence relationships, including Miller's algorithm (effectively a shooting method) and (F.W.J.) Olver's algorithm (effectively treating it as a discrete 2-point boundary value problem). We present the latter in a slightly simplified form. Here the idea is to only specify one initial condition, leading to a tridiagonal system:

$$\begin{pmatrix} 1 & & & & \\ c_1 & a_1 - z & b_1 & & \\ & c_2 & a_2 - z & b_2 & \\ & & c_3 & a_3 - z & \ddots \\ & & & \ddots & \ddots \end{pmatrix} \begin{pmatrix} C_0(z) \\ C_1(z) \\ C_2(z) \\ C_3(z) \\ \vdots \end{pmatrix} = \begin{pmatrix} C_0(z) \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix} \tag{15}$$

The operator on the left is invertible in ℓ^2 (this is seen by doing a row elimination and noting the connection with the resolvent of the Jacobi operator), hence the Cauchy transforms are

the unique solution. An effective way to solve this is finite section and then using Gaussian elimination; F.W.J. Olver made the observation that the truncation size did not impact the actual Gaussian elimination steps hence this can be performed adaptively. Two simple convergence tests are available: the error in residual is available exactly, but also a backward error estimate can be found.

We note that this idea can also be performed using Given rotations / Householder reflections leading to the adaptive QR method introduced in [SO & Townsend 2013]. For parameters we consider Gaussian elimination is stable and more efficient.

When do we use forward recurrence versus Olver's algorithm? This depends on how many terms of $C_n(z)$ are needed to be computed. Based on heuristics we use forward recurrence if z is within a Bernstein ellipse with radii proportional $1/n$, otherwise we use Olver's algorithm. Note that evaluating left/right limits of the Cauchy transform on the interval itself is accurate with forward recurrence.

2.3 Software demo

ApproxFun is built out of orthogonal polynomials: functions are represented by a space and coefficients, and the space dictates the basis. Here is a simple Example showing that $\cos(n * \operatorname{acos} x)$ is one in the $(n+1)$-th coefficient:

```
using ApproxFun, SingularIntegralEquations, ComplexPhasePortrait

n = 4
f = Fun(x -> cos(n*acos(x)), Chebyshev()) # determine the Chebyshev coefficients
    adaptively
f.coefficients ≈ [zeros(n); 1]
```

```
true
```

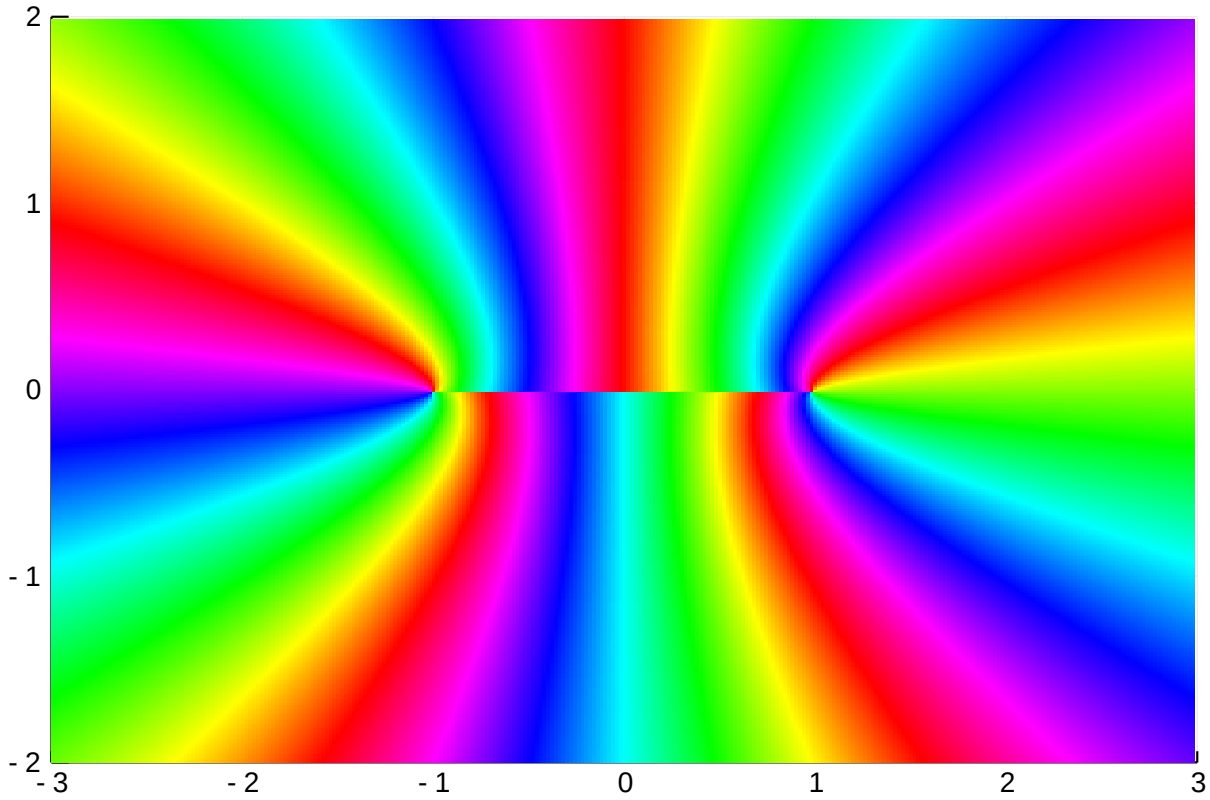
To incorporate the weight, we wrap the space in a `JacobiWeight`:

```
f = Fun(x -> cos(n*acos(x))/sqrt(1-x^2), JacobiWeight(-0.5,-0.5,Chebyshev()))
f.coefficients ≈ [zeros(n); 1; zeros(length(f.coefficients)-n-1)]
```

```
true
```

Cauchy transforms of OPs are special, for one they decay like $O(z^{-n-1})$. This can be seen easily with a phase portrait:

```
xx = range(-3,3; length=300)
yy = range(-2,2; length=300)
phaseplot(xx, yy, z -> cauchy(f,z))
```


2
1
0
- 1
- 2
- 3
- 2
- 1
0
1
2
3

3 Riemann-Hilbert problems

We now see how these tools can be applied to solve Riemann-Hilbert problems. Consider a matrix RH problem of find ϕ from G and F satisfying

1. *Analyticity*: $\Phi : (\mathbb{C}\backslash\Gamma) \to \mathbb{C}^{d\times d}$ is analytic off Γ,
2. *Asymptotic behaviour*: $\Phi(z) \sim I$ as $z \to \infty$,
3. *Jump condition*: the left/right limits $\Phi_\pm$ satisfy

$$\Phi_+(s) - G(s)\Phi_-(s) = F(s) \qquad \text{for} \qquad s \in \Gamma.$$

This is an example of a *left* RH problem as G is acting on the left. Most RH problems in applications are stated as *right* RH problems where the jump is

$$\Psi_+(s) - \Psi_-(s)G(s) = F(s) \qquad \text{for} \qquad s \in \Gamma.$$

Note that they can be easily transformed: if $\tilde{\Phi}(z) = \Psi(z)^\top$ then

$$\tilde{\Phi}_+(s) - G(s)^\top\tilde{\Phi}_-(s) = F(s)^\top \qquad \text{for} \qquad s \in \Gamma.$$

We reduce (left) RH problems to a singular integral equation on Γ involving Cauchy transforms by writing

$$\Phi(z) = I + \mathcal{C}_\Gamma V(z)$$

where $V : \Gamma \to \mathbb{C}^{d\times d}$. Then V satisfies

$$\mathcal{C}_+V(s) - G(s)\mathcal{C}_-V(s) = F(s) + G(s) - I.$$

To simplify things we use $\mathcal{C}_+ - \mathcal{C}_- = I$ to rewrite this as

$$V(s) - (G(s) - I)\mathcal{C}_-V(s) = F(s) + G(s) - I.$$

There are a number of possibilities for discretising such equations. For simplicity, we opt for collocation: choose a basis $\psi_k : \Gamma \to \mathbb{C}$ built out of orthogonal polynomials for which we know the Cauchy transform explicitly, and write

$$V(s) \approx \sum_{k=0}^{n-1} \mathbf{V}_k\psi_k(s)$$

where $\mathbf{V}_k \in \mathbb{C}^{d\times d}$ are determined by solving a linear system

$$\sum_{k=0}^{n} \mathbf{V}_k\psi(s_j) - (G(s_j) - I)\mathbf{V}_kC_-\psi(s_j) = F(s_j) + G(s_j) - I,$$

where $s_1, \ldots, s_n$ are *collocation* points. The situation is slightly more complicated when we include junction points, we'll come to that later.

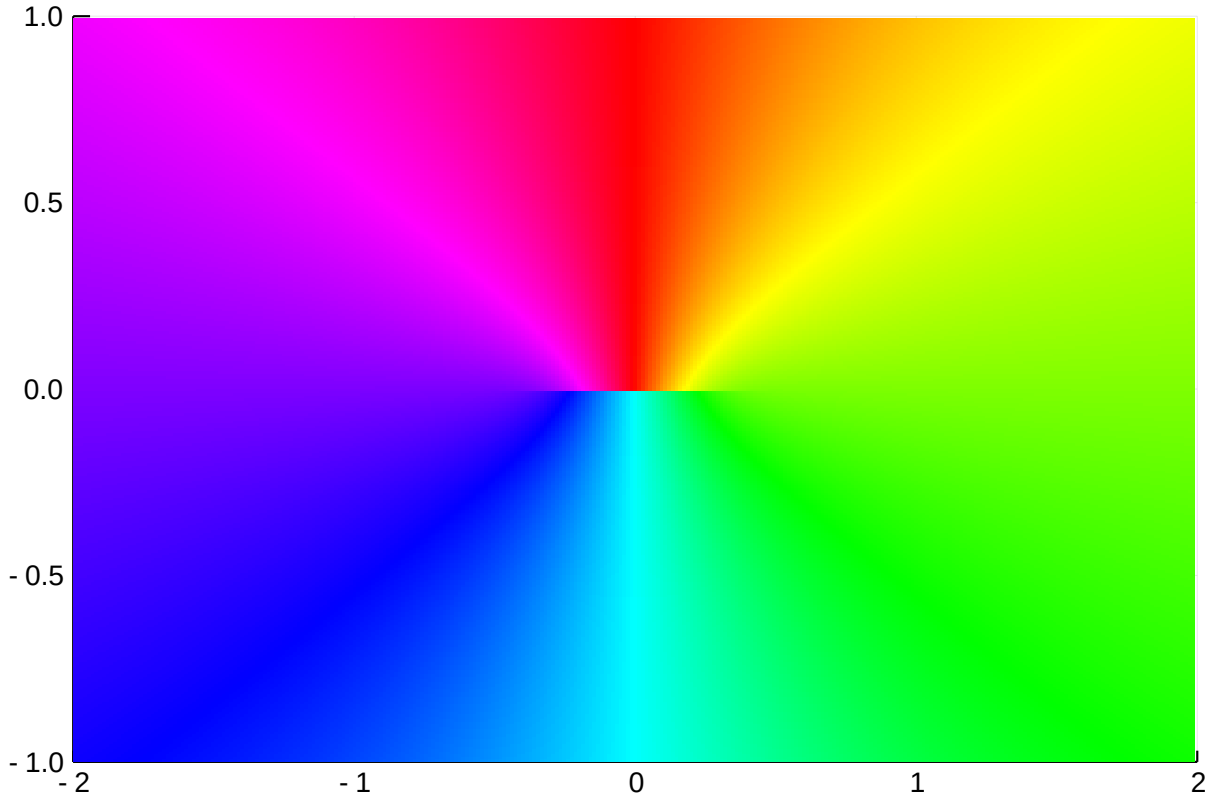
Example (Scalar RH problem) Let's look at a simple scalar example on $[-1, 1]$ with the jump

$$G(x) = 1 + \mathrm{e}^{-30x^2}.$$

Practically speaking this vanishes to all orders at ± 1. We can use Legendre or weighted Chebyshev bases: weighted Chebyshev bases are in some sense better since we know everything in closed form but for consistency in with the multiple segment case we use Legendre. The points we use are also flexible, we use Chebyshev points of the first kind $\cos \frac{2\pi(j+1/2)}{n}$.

Thus the following code solves the specified RH problem:

```
using ApproxFun, SingularIntegralEquations, RiemannHilbert, Plots, LinearAlgebra,
    ComplexPhasePortrait, BenchmarkTools
import RiemannHilbert: RiemannDual

n = 200
s = points(-1..1, n) # First kind Chebyshev points
G = x -> 1 + exp(-30x^2)

L = Array{ComplexF64}(undef, n, n) # collocation matrix
for k = 0:n-1
    p_k = Fun(Legendre(), [zeros(k); 1]) # k-th Legendre polynomial
    L[:,k+1] .= p_k.(s) .- (G.(s).-1).*cauchy.(p_k, (s)^-)
end

V_cfs = L \ (G.(s) .- 1) # Find Legendre coefficients match collocation system
V = Fun(Legendre(), V_cfs) # Interpret the coefficients as Legendre coefficients
xx = range(-2,2; length=300)
yy = range(-1,1; length=300)
phaseplot(xx, yy, z -> cauchy(V,z)) # plot cauchy transform to see cut clearly
```



Note that `V` encodes the solution to the RH problem which satisfies the specified jump to high accuracy:

```
Φ = z -> 1 + cauchy(V,z)    # Φ solves the RH problem
Φ(0.1⁺) - G(0.1)Φ(0.1⁻)
```

```
-2.220446049250313e-16 - 5.551115123125783e-17im
```

This whole solution technique is wrapped up in `rhsolve` from RiemannHilbert.jl. Thus we perform the above calculation as simply as writing:

```
x = Fun(-1..1)
G = 1 + exp(-30x^2)
Φ = rhsolve(G, 200)
Φ(0.1⁺) - G(0.1)Φ(0.1⁻)
```

```
-6.661338147750939e-16 - 8.326672684688674e-17im
```

Example (Matrix RH problem) The matrix-valued case works very similarly:

```
x = Fun(-1.0..1.0)
G = [1+exp(-30x^2) sech(30x) ;
      -sech(15x)^2    1.0   ]
Φ = @btime rhsolve(G, 1000) # use BenchmarkTools.jl to time
```

```
252.175 ms (1002882 allocations: 127.75 MiB)
```

```
Φ(0.1⁺) - G(0.1)Φ(0.1⁻)
```

```
2×2 Array{Complex{Float64},2}:
 -5.32907e-15+6.52256e-15im   2.95597e-15-1.29619e-14im
 -2.96291e-15+6.32827e-15im  -6.21725e-15+6.89379e-15im
```

Here we needed a 1000 collocation points, but as we can see the problem is still solvable in under a second.

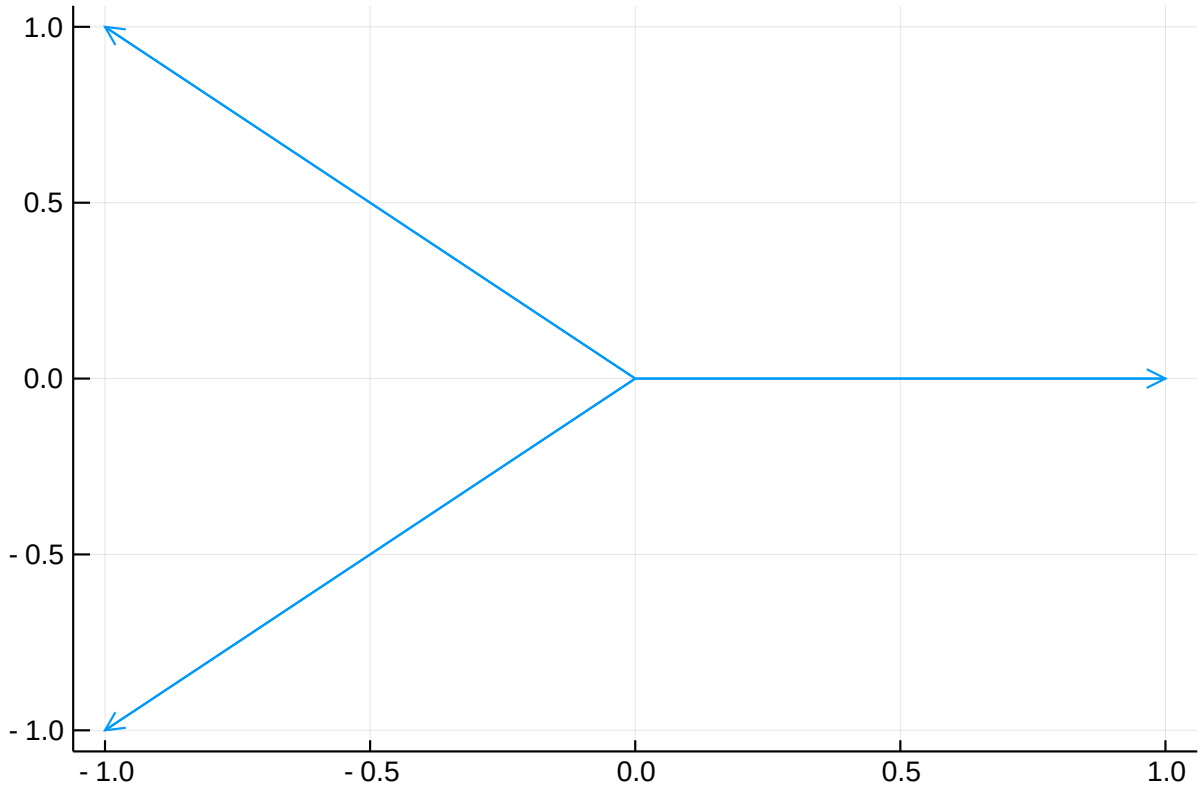
3.1 Junction points

We now come to the challenging part (and arguably the main contribution of [SO 2011]), junction points. Cauchy transforms of Legendre polynomials have logarithmic singularities, whereas the solution to the RH problems that arise in the applications we consider are nonsingular at junctions. Thus somehow we need the logarithmic singularities to cancel out. Here we will consider a single junction point on

$$\Gamma = \Gamma_1 \cup \cdots \cup \Gamma_M$$

where each of Γ_j are line-segments oriented outwards. Here is a simple example of three intervals with the origin as a junction point:

```
Γ_1, Γ_2, Γ_3 = Segment(0,1), Segment(0,-1+im), Segment(0,-1-im)
Γ = Γ_1 ∪ Γ_2 ∪ Γ_3
plot(Γ; arrow=:arrow, legend=false)
```



A key feature that makes this tractable is that we know the asymptotics near the endpoints precisely for Cauchy transforms of smooth functions:

$$\mathcal{C}_{[-1,1]}f(z) \sim_{z\to 1} \frac{f(1)}{2\pi \mathrm{i}} \log(z-1) + \text{const.}$$
$$\mathcal{C}_{[-1,1]}f(z) \sim_{z\to -1} \frac{-f(-1)}{2\pi \mathrm{i}} \log(-1-z) + \text{const.}$$

one way to see this is by expanding f in an orthogonal polynomial expansion and applying the 3-term recurrence of the Cauchy transform. That is, we know

$$\mathcal{C}_{[-1,1]}1(z) = \frac{\log(z-1) - \log(z+1)}{2\pi \mathrm{i}} = \frac{1}{2\pi \mathrm{i}} \log(z-1) - \frac{\log 2}{2\pi \mathrm{i}}$$

and then the 3-term recurrences gives the asymptotics of other terms in the expansion,

We want to get a handle on Cauchy transforms for bases supported on Γ_j evaluated on Γ_k as we approach a junction point. Thus consider

$$\mathcal{C}_{[-1,1]}f(1 + \varepsilon \mathrm{e}^{\mathrm{i}\theta}) \sim \frac{f(1)}{2\pi \mathrm{i}} (\log \varepsilon + \mathrm{i}\theta) + \text{const.}$$

The constant term changes depending on the angle of approach. We want a convenient way to capture this notion in software.

3.2 Dual and log numbers

Motivated by techniques from automatic differentiation we replicate the relationship between dual numbers and derivatives.

Definition (Dual number) A *dual number* is denoted

$$a + b\varepsilon$$

where $\varepsilon^2 = 0$.

Note that analytic functions have the feature that

$$f(a + b\varepsilon) = f(a) + bf'(a)\varepsilon$$

which follows from applying Taylor series of f simplied by the algebraic relationship $\varepsilon^2 = 0$. Dual numbers are a common tool of automatic differentiation, and in fact all relevant commands are implemented in DualNumbers.jl: functions can be composed, added, multiplied, etc. to automatically determine derivatives of functions to high accuracy. All that is needed is that the basic building blocks correctly support dual numbers.

Now consider

$$\mathcal{C}_{[-1,1]} f(1 + \mathrm{e}^{\mathrm{i}\theta}\varepsilon)$$

The Cauchy transform is not analytic at 1 so we cannot use the standard dual number function evaluation. Instead we introduce a related notion that encodes logarithmic blow-up:

Definition (Log number) A *log number* is a tuple (a, b) denoted by

$$a \log \varepsilon + b.$$

We refer to a as the *log part* and b as the *finite part*.

Unlike dual numbers which have an algebraic definition, we just treat log numbers as a book keeping device, with only addition and scalar multiplication defined. A nice feature is that this is all that is needed for forward recurrence, hence we can successfully calculate the local behaviour at the endpoints.

Example (two intervals)

A simple model problem is dividing a single interval $[-1, 1]$ into two intervals $[-1, 0] \cup [0, 1]$, thus artificially introducing a junction point at 0. We know that if f is smooth on $[-1, 1]$ then $C^{\pm} f(0)$ are well-defined. But if we break the interval in two and proceed naively then we have

$$C^{+}_{[-1,1]} f(0) = C_{[-1,0]} f(\mathrm{i}\varepsilon) + C_{[0,1]} f(\mathrm{i}\varepsilon) = \frac{1}{2\pi\mathrm{i}}(\infty - \infty)$$

Log numbers give a way to make sense of this.

We can see this in software. To avoid confusion with automatic differentiation, we use a `RiemannDual` type to encode direction, which also supports the case of z on the Riemann sphere: that is we can capture the angle of approach to ∞. (This is not discussed further in these notes.) Thus we have

```
x_1,x_2 = Fun(-1..0), Fun(0..1)
f_1,f_2 = exp(x_1), exp(x_2)
cauchy(f_1,RiemannDual(0,im)), cauchy(f_2,RiemannDual(0,im))
```

```
((-0.0 - 0.1591549430918951im)log ε + 0.2499999999999972 - 0.1267827638931
4774im, (0.0 + 0.15915494309189537im)log ε + 0.25000000000000006 - 0.209750
64191541176im)
```

here we see that the summing of two contributions cause the logarithmic terms to cancel (apart from round-off error), recovering the true Cauchy transform:

```
cauchy(f_1,RiemannDual(0,im)) + cauchy(f_2,RiemannDual(0,im))
```

```
(0.0 + 2.7755575615628914e-16im)log ε + 0.4999999999999998 - 0.336533405808
5595im
```

This matches the single interval case to high accuracy:

```
x = Fun()
f = exp(x)
cauchy(f,0.0⁺)
```

```
0.5000000000000002 - 0.33653340580856117im
```

3.3 Zero sum condition product condition and cyclic junction condition

For junction points, we use one collocation point at the junction point in each direction. That is, if ζ is a junction point and $\Gamma_1, \ldots, \Gamma_M$ are the adjoining contours eminating from ζ at angles $\theta_1, \ldots, \theta_M$, the collocation points contain

$$\zeta + \mathrm{e}^{\mathrm{i}\theta_1}\varepsilon, \ldots, \zeta + \mathrm{e}^{\mathrm{i}\theta_M}\varepsilon.$$

We use the notation $\zeta_j := \zeta + \mathrm{e}^{\mathrm{i}\theta_j}$. To build the collocation matrix, we ignore the log part and take only the finite part of the Cauchy transforms. This (generically) successfully results in a solution that does not blow up because the logarithmic singularities cancel, and has high accuracy.

To see this, we introduce three related conditions. Here we state the conditions assuming the contours Γ_j adjoining a junction point ζ are oriented away, where the alternative case follows from reversing the orientation with various sign changes.

Definition (zero sum condition) A function V satisfies the *zero sum condition* (at a junction point ζ) if

$$V(\zeta_1) + \cdots + V(\zeta_M) = 0,$$

assuming all contours are oriented out from ζ.

If V satisfies the zero sum condition then the logarithmic singularities cancel at ζ. Moreover, the Cauchy transform is continuous between each contour (that is, for $\theta_j < \theta < \theta_{j+1}$).

Definition (product condition) The jump G satisfies the *product condition* (at a junction point ζ) if the product of the jumps at the junction points is I:

$$G_M \cdots G_1 = I$$

where $G_j = G(\zeta_j)$, assuming all contours are oriented out from ζ and listed in counter-clockwise order.

Definition (cyclic junction condition) A function R satisfies the *cyclic junction condition* (at a junction point ζ) if

$$G_M \cdots G_2 R_1 + G_M \cdots G_3 R_2 + \cdots + G_M R_{M-1} + R_M = 0$$

If G satisfies the product condition (at ζ) and V satisfies the zero sum condition (at ζ) then

$$(I - (G - I)C^-)V = C^+V - GC^-V$$

satisfies the cyclic junction condition. This is due to a telescoping sum argument, using the continuity of the Cauchy transform. We demonstrate this on the three-contour setting. Denote

$$C_j^{\pm} := C^{\pm}V(\zeta_j)$$

where $R_j = C_j^+ - G_j C_j^-$. As the logarithmic terms have cancelled we have sectional continuity, that is, $C_1^+ = C_2^-$, $C_2^+ = C_3^-$ and $C_3^+ = C_1^-$. Therefore, we have

$$\begin{aligned} G_3G_2R_1 + G_3R_2 + R_3 &= G_3G_2(C_1^+ - G_1C_1^-) + G_3(C_2^+ - G_2C_2^-) + (C_3^+ - G_3C_3^-) \\ &= G_3G_2C_2^- - G_3G_2G_1C_1^- + G_3C_3^- - G_3G_2C_2^- + C_1^- - G_3C_3^- \\ &= 0. \end{aligned}$$

Generically (under another condition called the *nonsingular junction condition*), the collocation system resulting from only taking the finite part is invertible and the solution satisfies the zero sum condition whenever the product condition and cyclic junction conditions are satisfied. This extra condition can be removed by replacing one of the collocation conditions

$$V(\zeta_M) - (G(\zeta_M) - I)\mathcal{C}_- V(\zeta_M) = F(\zeta_M) + G(\zeta_M) - I$$

where $\zeta_j = \zeta + \mathrm{e}^{\mathrm{i}\theta_j}\epsilon$ with the zero sum condition. Doing so ensures that the removed condition is still satisfied.

To see this we return to the three contour case. Denote the solution to the n-point collocation system V_n. Note that the collocation conditions ensure that

$$C_{n1}^+ - G_1C_{n1}^- = R_1 \qquad \text{and} \qquad C_{n2}^+ - G_2C_{n2}^- = R_2$$

where $G_j := G(\zeta_j)$, $R_j := R(\zeta_j)$ for $R(\zeta) := F(\zeta) + G(\zeta) - I$, and $C_{nj}^{\pm} := C^{\pm}V_n(\zeta_j)$. Instead of imposing the condition $C_{n3}^+ - G_3C_{n3}^- = R_3$, we impose the zero sum condition

$$V_n(\zeta_1) + V_n(\zeta_2) + V_n(\zeta_3) = 0.$$

A consequence is that $\mathcal{C}V_n(\zeta)$ has no logarithmic singularity and therefore $C_{nj}^+ = C_{n,j+1}^-$. Thus we have

$$\begin{aligned} G_3C_{n3}^- + R_3 &= G_3C_{n2}^+ + R_3 = G_3G_2C_{n2}^- + R_3 + G_3R_2 \\ &= G_3G_2C_{n1}^+ + R_3 + G_3R_2 = G_3G_2G_1C_{n1}^- + R_3 + G_3R_2 + G_3G_2R_1 \\ &= C_{n3}^+ \end{aligned}$$

where we used the fact that R satisfies the cyclic junction condition. In other words, the condition we did not impose is still satisfied.

3.4 Software demo

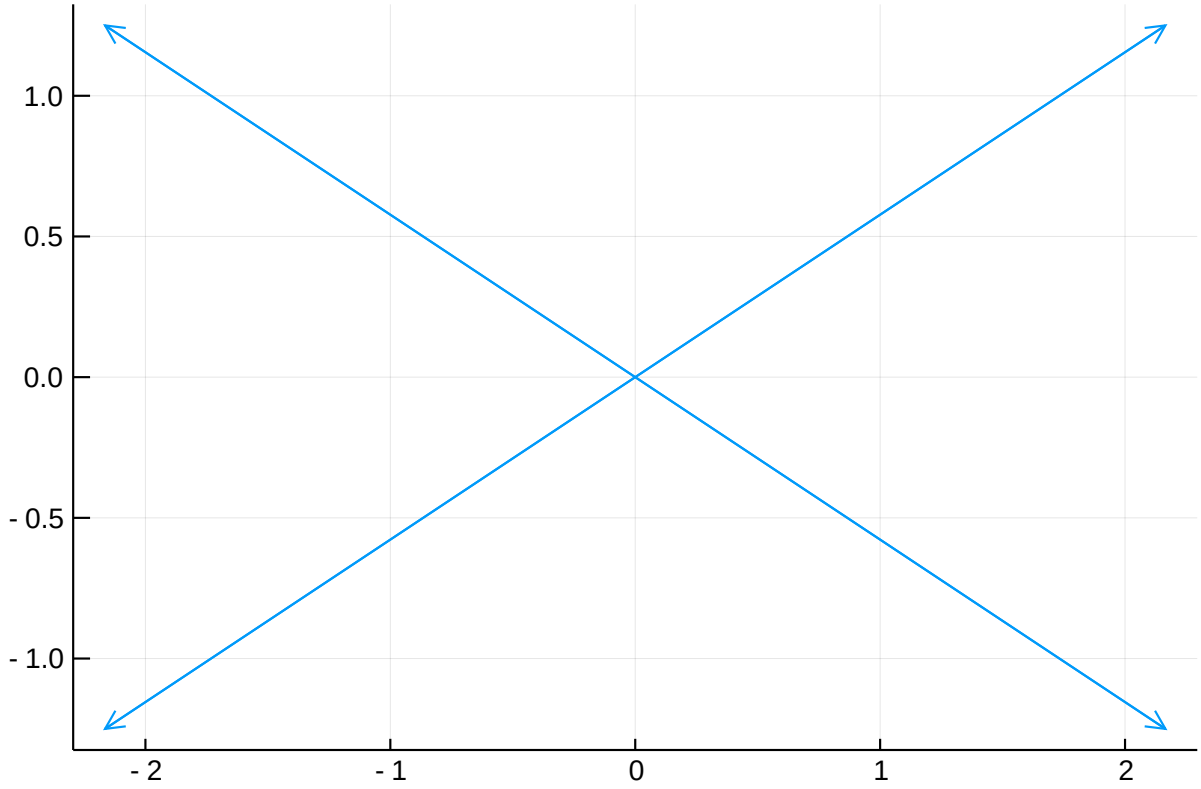
Here's an example of calculating the Hastings-McLeod solution, which is a special solution to homogeneous Painleve II (PII), that is, satisfying

$$u'' = xu + 2u^3.$$

We omit the details but PII can be recast as a RH problem where the point of evaluation x is a parameter in the jump.

Hasting-McLeod's RH problem simplifies to the following contour, where we truncate the infinite rays where the jump is within machine tolerance of I:

```
Γ = Segment(0, 2.5exp(im*π/6))   ∪
    Segment(0, 2.5exp(5im*π/6))  ∪
    Segment(0, 2.5exp(-5im*π/6)) ∪
    Segment(0, 2.5exp(-im*π/6))
plot(Γ; arrow=:arrow, legend=false)
```



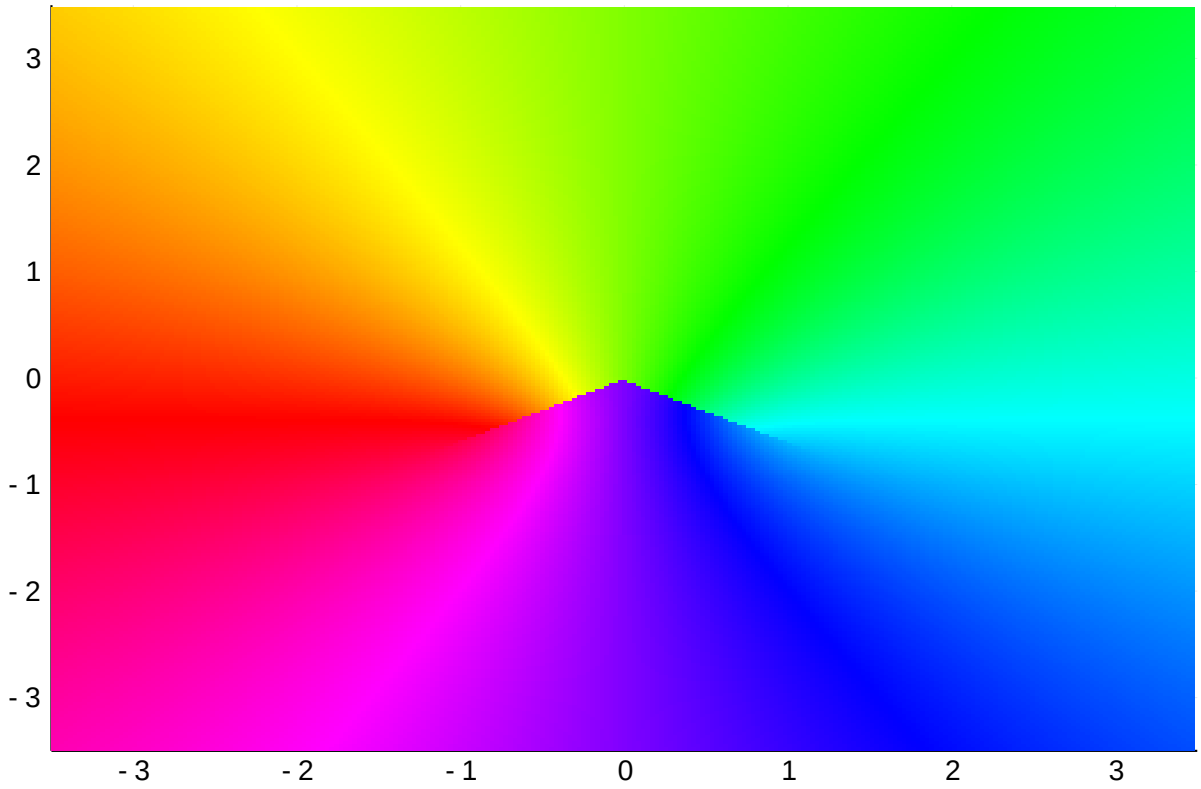
We define the jump:

```
s_1,s_3 = im, -im # Stokes' constants that determine which solution to PII
x = 0.0           # point of evaluation
G = Fun( z -> if angle(z) ≈ π/6
                  [1                  0;
                   s_1*exp(8im/3*z^3+2im*x*z) 1]
              elseif angle(z) ≈ 5π/6
                  [1                  0;
                   s_3*exp(8im/3*z^3+2im*x*z) 1]
              elseif angle(z) ≈ -π/6
                  [1                -s_3*exp(-8im/3*z^3-2im*x*z);
                   0                 1]
```

```
elseif angle(z) ≈ -5π/6
    [1                -s_1*exp(-8im/3*z^3-2im*x*z);
     0                 1]
end
    , Γ);
```

We can now solve the RH problem, which in this case is given as a left RH problem The default is to solve a left RH problem $\Phi_+ = G\Phi_-$, so we transpose to convert to $\Phi_+^\top = \Phi_-^\top G^\top$.

```
Φ = transpose(rhsolve(transpose(G), 800))
xx = yy = range(-3.5,3.5; length=250)
phaseplot(xx, yy, Φ[1,2]) # plot the (1,2) component
```



We can check that it does indeed satisfy the right jump:

```
ζ = exp(im*π/6)
norm(Φ((ζ)⁺) - Φ((ζ)⁻)*G(ζ))
```

```
5.943599078413926e-16
```

The value of Painleve II falls out of taking an asymptotic limit:

```
z = Fun(ℂ)
2(z*Φ[1,2])(Inf)
```

```
-0.3670615515480783 + 4.163336342344337e-17im
```

4 Deift-Zhou Steepest Descent

A powerful aspects of Riemann-Hilbert problems is that they can be deformed in the complex plane along paths of steepest descent to turn oscillations into exponential decay. This is useful for asymptotics with respect to large parameters: Provided a *parametrix* is known that locally solves the RH problem, the RH problem can be solved asymptotically, and thereby rigorous asymptotics with error bounds derived. This is also useful for numerics, which do not require the knowledge of parametrices to achieve asymptotic accuracy, while still achieving uniform computational cost.

Deforming oscillatory RH problems is very much in the same spirit as steepest descent for oscillatory integrals, which produces asymptotic results but can also be used for effective numerics (see e.g. [Huybrechs & Vandewalle 2006]). However, for RH problems we also need to factorise the jumps in order to effectively deform.

In what follows we work with right RH problems with no forcing terms, that is, the jump satisfies

$$\Phi_+(s) = \Phi_-(s)G(s) \qquad \text{for} \qquad s \in \Gamma.$$

4.1 Lensing

A key observation is that if we can factor $G(z) = A(z)B(z)C(z)$ where C/A can be analytically continued above/below we can *lense* the RH problem. For example, suppose the jump is on the interval $(-1, 1)$ and let Ω_+/Ω_- be the upper/lower half unit disk. Define a change of variables:

$$\Psi(z) := \begin{cases} \Phi(z)C(z)^{-1} & z \in \Omega_+ \\ \Phi(z)A(z) & z \in \Omega_- \\ \Phi(z) & \text{otherwise} \end{cases}$$

Then Ψ satisfies the lensed RH problem. That is, for Γ_+/Γ_- the upper/lower half circle (oriented from -1 to 1):

$$\begin{aligned} &\Psi_+(z) = \Phi(z) = \Psi_-(z)C(z) \qquad \text{for} \qquad z \in \Gamma_+ \\ &\Psi_+(x) = \Phi_+(x)C(x)^{-1} = \Phi_-(x)G(x)C(x)^{-1} = \Psi_-(x)B(x) \qquad \text{for} \qquad z \in (-1, 1) \\ &\Psi_+(z) = \Phi(z)A(z) = \Psi_-(z)A(z) \qquad \text{for} \qquad z \in \Gamma_-. \end{aligned}$$

4.2 Example: defocusing Nonlinear Schrödinger equation

We start with a simple example of an oscillatory RH problem posed only on the real line (which should tie closely with Wiener-Hopf problems). This arises from inverse scattering of the defocusing nonlinear Schrödinger equation on the real line

$$\mathrm{i}u_t + u_{xx} - 2|u|^2 u = 0, \qquad u(0, x) = u_0(x).$$

Finding associated scattering data allows us to *transform* u_0 into a *reflection coefficient* $r : \mathbb{R} \to \mathbb{C}$, this can be thought of as a nonlinear analogue of the Fourier transform. Provided u_0 is smooth and rapidly decaying, r can be assumed to be be a smooth, exponentially decaying function satisfying $|r(k)| < 1$. Moreover, like the Fourier transform it can be analytically continued in a strip surrounding the real axis.

The solution $u(t, x)$ can be recovered from r by solving the following RH problem

$$\Phi_+(k) = \Phi_-(k) \begin{pmatrix} 1 - |r(k)|^2 & -\bar{r}(k)\mathrm{e}^{-\mathrm{i}(4tk^2+2xk)} \\ r(k)\mathrm{e}^{\mathrm{i}(4tk^2+2xk)} & 1 \end{pmatrix}$$

and taking a limit:

$$u(t, x) = 2\mathrm{i} \lim_{z\to\infty} z\Phi(z)_{12}.$$

Here $\bar{r}(k)$ is the complex conjugate (at least for now). The importance of this formulation is that evolving the RH problem in time is trivial. This very much replicates the relationship between linear Schrödinger equation and its Fourier transform.

Let's see this in action. We take $r(k) = 0.5\mathrm{e}^{-k^2}$: this is arbitrary but solving the RH problem will tell us which $u_0(x)$ gives this reflection coefficient. For example, the following code recovers $u_0(0.0)$ from its reflection coefficient:

```
using RiemannHilbert, ApproxFun, Plots
x,t = 0.0,0.0
k = Fun(-5.5 .. 5.5) # truncate at 5.5 where 0.5exp(-5.5^2) is numerically zero
r = 0.5exp(-k^2)
θ = 4t*k^2+2x*k
G = [1-abs2(r)                      -conj(r)*exp(-im*θ);
        r*exp(im*θ)      1  ]

Φ = transpose(rhsolve(transpose(G),500))
z = Fun(ℂ)          # the Function z in the complex plane
2im*(z*Φ[1,2])(Inf)    # short-hand for taking limits
```

```
0.2890170986853543 - 2.862293735361732e-17im
```

We won't discuss the other direction (computing r from u_0) but this is a scattering problem which can be accomplished by solving ODEs.

As x or t become large the jump becomes increasingly oscillatory, and we need more and more coefficients to resolve it:

```
x,t = 1.0,1.0
r = exp(-k^2)
θ = 4t*k^2+2x*k
G = [1-abs2(r)                      -conj(r)*exp(-im*θ);
        r*exp(im*θ)      1  ]
ncoefficients(G)
```

```
795
```

This means we need more coefficients to resolve Φ:

```
Φ = transpose(rhsolve(transpose(G),1000))
z = Fun(ℂ)          # the Function z in the complex plane
2im*(z*Φ[1,2])(Inf)    # short-hand for taking limits
```

```
0.26746565554744944 - 0.14664527290803903im
```

We will avoid these oscillations by deforming in the complex plane. But first we lense. To lense, we use the UL and LDU decompositions. Let $\theta(z) := 4tz^2 + 2xz$. Then we have the decompositions

$$G(z) = \underbrace{\begin{pmatrix} 1 & -\bar{r}\mathrm{e}^{-\mathrm{i}\theta} \\ & 1 \end{pmatrix}}_{M(z)} \underbrace{\begin{pmatrix} 1 & \\ r\mathrm{e}^{\mathrm{i}\theta} & 1 \end{pmatrix}}_{P(z)}$$

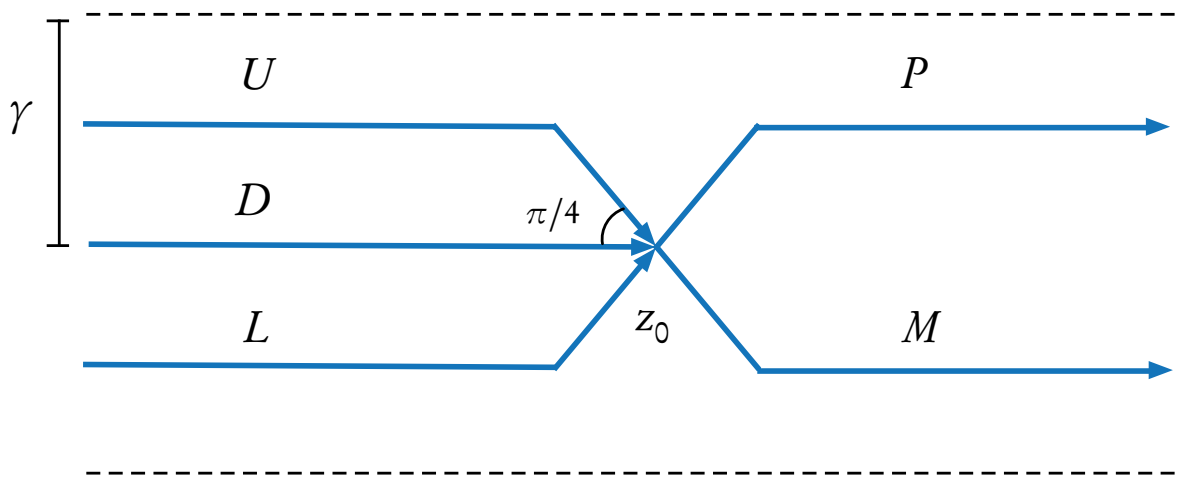
$$G(z) = \underbrace{\begin{pmatrix} 1 & \\ \frac{r\mathrm{e}^{\mathrm{i}\theta}}{1-|r|^2} & 1 \end{pmatrix}}_{L(z)} \underbrace{\begin{pmatrix} 1-|r|^2 & \\ & \frac{1}{1-|r|^2} \end{pmatrix}}_{D(z)} \underbrace{\begin{pmatrix} 1 & -\frac{\bar{r}\mathrm{e}^{-\mathrm{i}\theta}}{1-|r|^2} \\ & 1 \end{pmatrix}}_{U(z)}$$

Before we continue there's a slight technical detail: the complex conjugate of $\overline{r(z)}$ is *not* analytic in the complex plane, and the proposed lensing process falls apart. There's a trivial solution: $\overline{r(\bar{z})}$ is analytic and equals $\bar{r}(k)$ for real z. Thus in what follows we define (in a bit of slightly confusing notation)

$$\bar{r}(z) := \overline{r(\bar{z})}.$$

Similarly, the analytic continuation of $|r(z)|^2$ is given by $r(z)\bar{r}(z)$.

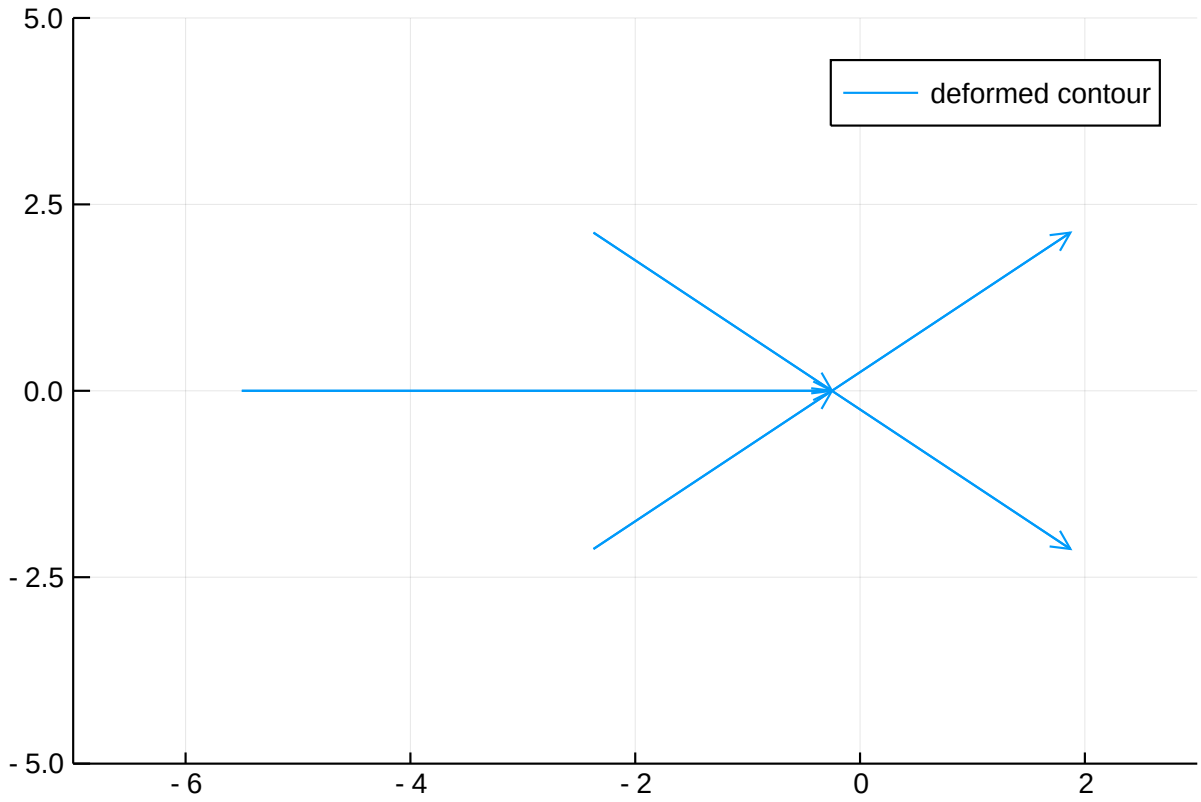
Following classical steepest descent, we deform through the stationary point of θ at $z_0 = -\frac{x}{4t}$. To ensure *descent* (and not *ascent*) we need to use the LDU decomposition to the left of z_0 and the UL decomposition (i.e. $G(z) = M(z)P(z)$) to the right of z_0. Here is a figure showing the deformation:



The deformed jump rapid decays to the identity, at which point we can truncate the contours. The place of truncation depends on x and t, choosing it optimally guarantees asymptotic accuracy [Olver & Trogdon 2014].

Thus we can solve the deformed RH problem on the following contour:

```
z_0 = -x/(4t)
Γ = Segment(z_0,z_0+3exp(im*π/4)) ∪ Segment(z_0,z_0+3exp(-im*π/4)) ∪
    Segment(z_0+3exp(3im*π/4),z_0) ∪ Segment(-5.5,z_0) ∪ Segment(z_0+3exp(-3im*π/4),z_0)
plot(Γ; xlims=(-7,3), ylims=(-5,5), label="deformed contour", arrow=:arrow)
```



The following constructs the deformed RH problem. We see a significant decrease in coefficients:

```
θ = z -> 4t*z^2+2x*z
r = z -> exp(-z^2)
r̄ = z -> conj(r(conj(z)))

L = z -> [1 0; r(z)/(1-r(z)r̄(z))*exp(im*θ(z)) 1]
D = z -> [1-r(z)r̄(z) 0; 0 1/(1-r(z)r̄(z))]
U = z -> [1 -r̄(z)/(1-r(z)r̄(z))*exp(-im*θ(z)); 0 1]

P = z -> [1 0; r(z)*exp(im*θ(z)) 1]
M = z -> [1 -r̄(z)*exp(-im*θ(z)); 0 1]


G = Fun(function(z)
        z ∈ component(Γ,1) && return P(z)
        z ∈ component(Γ,2) && return M(z)
        z ∈ component(Γ,3) && return U(z)
        z ∈ component(Γ,4) && return D(z)
        z ∈ component(Γ,5) && return L(z)
        error("Not in contour")
    end, Γ);
```

We can now solve this deformed RH problem numerically:

```
Φ = transpose(rhsolve(transpose(G),1000))
z = Fun(ℂ)
```

```
2im*(z*Φ[1,2])(Inf)
```

```
0.2674656555474456 - 0.14664527290805038im
```

This matches the undeformed RH problem to high accuracy, while avoiding oscillations. If x and t were larger, the deformed RH problem will continue to provide accurate results while the orginal undeformed RH problem becomes too oscillatory (and too ill conditioned) to be of use.

4.3 Asymptotic accuracy

This is not the end of the story. The remaining $D(z)$ jump is diagonal and solvable exactly, we can therefore remove it: if $Y_+(x) = Y_-(x)D(x)$ then writing

$$\Phi(z) = \Psi(z)Y(z)$$

we see that Ψ has no jump and on the remaining contours has the jump

$$\Psi_+(s) = \Psi_-(s)Y(s)G(s)Y(s)^{-1}.$$

As x or t become large, the contour can be shrunk closer to z_0, and the number of collocation points needed uniformly bounded.

Things become harder for degenerate RH problems, which normally arise when then jump functions LDU or UL decomposition become degenerate (for example, had $r(k) = 1$ at a point). This issue can usually be resolved by replacing θ with a so-called g-function that matches θ at infinity but has a branch cut, in a way that just happens to regularise the RH problem. This is needed for deforming the Hastings-McLeod solution to PII for large negative x, but is beyond the scope of these notes.

5 Sommerfeld radiation

Here we focus on the classical problem of scattering off a half-plate, based on [Llewellyn-Smith & Luca 2019]. That is, we want to solve Helmholtz

$$u_{xx} + u_{yy} + k^2 u = 0, u(x,y) \sim \mathrm{e}^{-\mathrm{i}kx\cos\theta_0 - \mathrm{i}ky\sin\theta_0}, u(x,0) = 0 \qquad \text{for} \qquad 0 \le x < \infty$$

I'll omit the details (as the audience knows this better than I do!) but we can recast the problem to a RH problem by taking the Fourier transform with respect to x:

$$\begin{aligned}
\Phi_+(\alpha, y) &:= \int_0^\infty \phi(x,y)\mathrm{e}^{\mathrm{i}\alpha x}\mathrm{d}x, \\
\Phi_-(\alpha, y) &:= \int_{-\infty}^0 \phi(x,y)\mathrm{e}^{\mathrm{i}\alpha x}\mathrm{d}x, \\
\Phi(\alpha, y) &:= \Phi_+(\alpha, y) + \Phi_-(\alpha, y).
\end{aligned}$$

We can clearly recover ϕ from Φ, and with a little work (see [Noble 1958]) it can be shown that we can recover Φ from

$$\Phi'_+(\alpha, 0) = \frac{\partial \Phi_+}{\partial y}(\alpha, y).$$

Let's make k a little bit complex: $\Im k > 0$. Then there is a RH problem that

$$\Psi(z) = \begin{cases} \Phi'_+(z,0) & \Im z > 0 \\ D_-(z) & \Im z < 0 \end{cases}$$

satisfies (with $\lim_{z\to\infty} \Psi(z) = 0$):

$$\Psi_+(\alpha) + \gamma(\alpha)\Psi_-(\alpha) = -k\frac{\sin\theta_0}{\alpha - k\cos\theta_0}$$

where $\gamma(\alpha) = \mathrm{i}\sqrt{k-\alpha}\sqrt{\alpha+k}$ and D_- is some suitably analytic, unknown function. The reason we need k to be complex is so that the singularities of γ and the right-hand side do not touch the contour we are solving the RH problem.

We can deform this in the complex plane to the line $\mathrm{e}^{-\mathrm{i}\pi/4}\mathbb{R}$ via:

$$\tilde{\Psi}(z) := \begin{cases} -k\frac{\sin\theta_0}{z - k\cos\theta_0} - \gamma(z)\Psi(z) & -\Re z < \Im z < 0 \cup 0 < \Im z < -\Re z \\ \Psi(z) & \text{otherwise} \end{cases}$$

Note that $D_-(z)$ (and thence $\Psi(z)$ in the lower half plane) decays like $O(z^{-3/2})$ and hence $\tilde{\Psi}(z)$ also decays: $\lim_{z\to\infty} \tilde{\Psi}(z) = 0$.

At this point we can allow k to become real again as we have deformed the contour to avoid the singularity. Thus we have our RHP

$$\tilde{\Psi}_+(\alpha) - \gamma(\alpha)\tilde{\Psi}_-(\alpha) = f(\alpha)$$

for $\alpha \in \mathrm{e}^{-\mathrm{i}\pi/4}\mathbb{R}$ and $f(\alpha) = -k\frac{\sin\theta_0}{\alpha - k\cos\theta_0}$.

We now consider numerical solution of this RH problem. Again we reduce it to a singular integral equation via

$$\tilde{\Phi}(z) = \mathcal{C}_{\mathrm{e}^{-\mathrm{i}\pi/4}\mathbb{R}} u(z).$$

Here $u(\alpha) = \tilde{\Phi}_+(\alpha) - \tilde{\Phi}_-(\alpha)$ only decays like $|\alpha|^{-1/2}$ and so we employ the Llewellyn-Smith-Luca map. That is, we represent

$$u(\alpha) \approx \sum_{k=0}^{n-1} u_k P_k(r_1^{-1}(\mathrm{e}^{\mathrm{i}\pi/4}\alpha))$$

where P_k are Legendre polynomials. We can evaluate

$$\mathcal{C}[P_k(r_1^{-1}(\mathrm{e}^{\mathrm{i}\pi/4}\diamond))](z)$$

by combining the Cauchy transform change-of-variables formulae and evaluation of the Cauchy transform of Legendre polynomials.

Thus we can set up a collocation for

$$\mathcal{C}^+ u(\alpha) + \gamma(\alpha)\mathcal{C}^- u(\alpha) = f(\alpha).$$

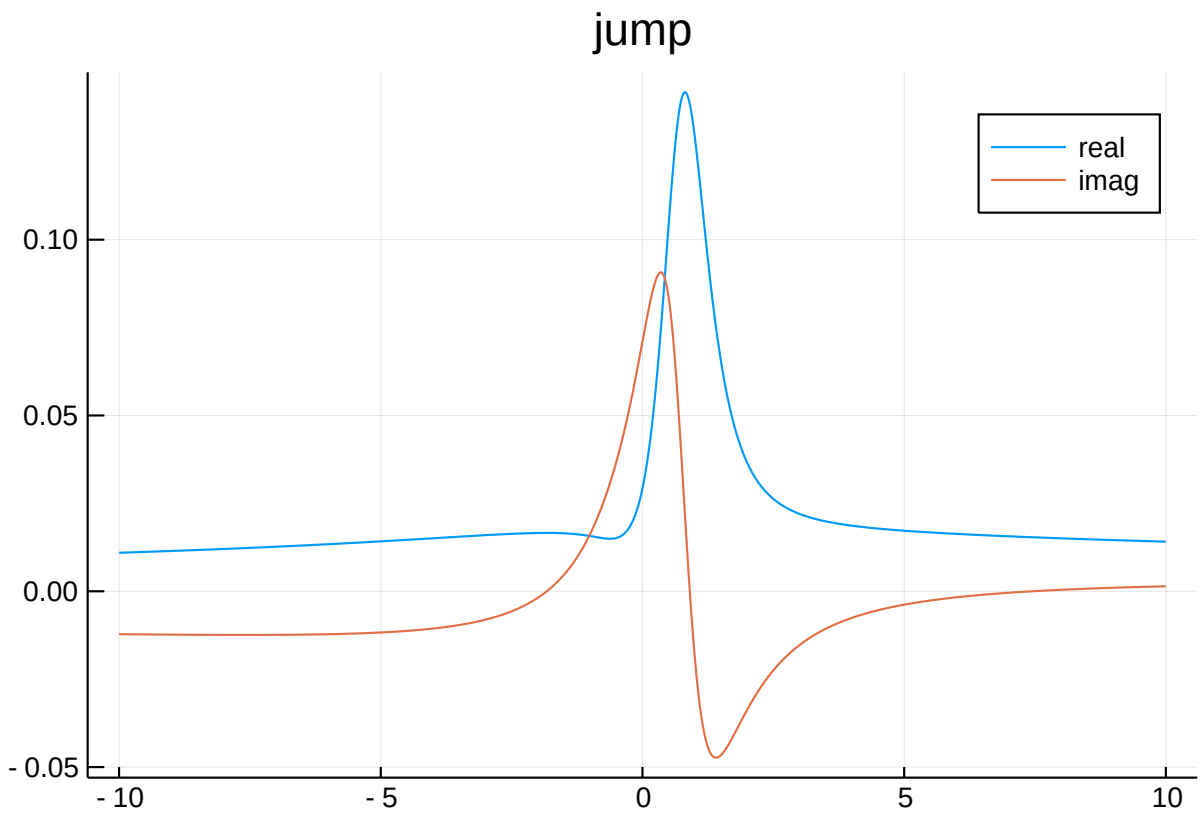
We do so using mapped Chebyshev points, but now we avoid ± 1. There we impose that u itself vanishes.

5.1 Software demo

This code is not meant for general usage, but will work. The WIP package WienerHopf.jl implements the map `SqrtLine{a}()` which is the real line rotated at angle `exp(im*π*a)`, using the Llewellyn-Smith-Luca map to transform the contour to the unit interval.

```
using WienerHopf, ApproxFun, RiemannHilbert, SingularIntegralEquations, LinearAlgebra,
    Plots

k = 1      # frequency
θ_0 = 0.1  # angle
S = Legendre(SqrtLine{-1/4}())             # Use Llewellyn-Smith-Luca mapped Legendre
    polynomial basis

γ = α -> isinf(α) ? complex(Inf) : im*sqrt(k-α)*sqrt(α+k)
f = α -> isinf(α) ? zero(α) : -k*sin(θ_0)/(α-k*cos(θ_0))

n = 256                                    # number of collocation points

pts = collocationpoints(S, n)[2:end-1]     # mapped Chebyshev points.  We drop ±∞.
Cm = fpcauchymatrix(S,n,n)[2:end-1,:]      # Evaluate the Cauchy transform at the
    collocation points.
E = evaluationmatrix(S,pts,n)              # Evaluate the basis at the collocation
    points.  This is the identity operator

L = E .+ Diagonal(γ.(pts) .+ 1)*Cm         # Discretisation of C^+ + γ*C^- == I +
    (γ+1)*C^-
A = [fill(1.0,1,n); L; ((-1.0).^(0:n-1))'] # Add top/bottom row to force vanshing at ±∞
u_cfs = A \ [0; f.(pts); 0]                # solve the discretised system
```

```
u = Fun(S, u_cfs)                              # wrap in mapped
xx = range(-10,10; length=1000)                # Plotting grid
plot(xx, real.(u.(exp(-im*π/4).*xx)); title="jump", label="real")
plot!(xx, imag.(u.(exp(-im*π/4).*xx)); label="imag")
```



We now compare with the exact solution:

```
Hp = α -> k*sin(θ_0)/(α-k*cos(θ_0)) * (1/sqrt(α+k) - 1/sqrt(k+k*cos(θ_0)))
Hm = α -> k*sin(θ_0) / (sqrt(k+k*cos(θ_0))*(α-k*cos(θ_0)))

Φdp = α -> -sqrt(α+k)Hp(α)
Dm = α -> -Hm(α)/(im*sqrt(k-α))

Φdp(1+im)-cauchy(u,1+im), Dm(-1-im)-cauchy(u,-1-im)
```

```
(-1.634126861604912e-12 + 3.9142296175131985e-12im, -1.8924410649656664e-12
 + 1.2427905926593041e-13im)
```

This is one simple example. See [Llewellyn-Smith & Luca 2019] for more complicated examples including matrix-valued cases.

6 References

1. D Huybrechs & S Vandewalle (2006), On the evaluation of highly oscillatory integrals by analytic continuation, *SIAM Journal on Numerical Analysis* 44, 1026-1048.

2. S. G. Llewelyn-Smith & E. Luca (2019) Numerical solution of scattering problems using a Riemann–Hilbert formulation, *Proc. Roy. Soc. A*, to appear.

3. B. Noble (1958), *Methods based on the Wiener-Hopf technique for the solution of partial differential equations*, Pergamon Press.

4. S. Olver (2014), Change of variable formul_ for regularizing slowly decaying and oscillatory Cauchy and Hilbert transforms, *Anal. Appl.*, 12: 369–384.

5. S. Olver (2012), A general framework for solving Riemann–Hilbert problems numerically, *Numer. Math.*, 122: 305–340.

6. S. Olver (2011), Numerical solution of Riemann–Hilbert problems: Painlevé II, *Found. Comput. Maths*, 11: 153–179.

7. S. Olver & T. Trogdon (2014), Nonlinear steepest descent and the numerical solution of Riemann–Hilbert problems, *Comm. Pure Appl. Maths*, 67: 1353–1389.

8. R.M. Slevinsky & S. Olver (2017), A fast and well-conditioned spectral method for singular integral equations, *J. Comp. Phys.*, 332: 290–315.

9. T. Trogdon & S. Olver (2015), *Riemann–Hilbert Problems, Their Numerical Solution and the Computation of Nonlinear Special Functions*, SIAM.

10. T. Trogdon & S. Olver (2013), Numerical inverse scattering for the focusing and defocusing nonlinear Schrödinger equations, *Proc. Royal Soc. A*, 469: 20120330.