0. A wrong in aaa and barycentric method in ACFlow

I find that for original ACFlow when we put in some noisy, this method works so bad. With debug I find the min sigularity of the chosen submatrix of the original Lowner matrix doesn't tend to be zero. It's impossible in theory.

The I realize that the svd() function in LinearAlgebra will throw rows (or columns, according to the definition of svd) in U and V corresponding to zero singular values. So even if the submatrix is not row full rank, $V[:, \mathrm{end}]$ got form svd() isn't corresponding to the smallest (that is to say, zero) singular value.

In fact, there is no need to do svd decomposition. we can just do Orthogonal decomposition on $Lsub' * Lsub$. I have done this with code and it worls well with adding noise.

1. How ACFlow brc algorithm find poles with delts type spectral density

The way barycentric algorithm get poles and corresponding amplitudes is function poles!().

poles!() :
Input : Barycentric Function

Output : reconstruct poles and amplitudes $\{\widetilde{P}_k, \widetilde{\gamma}_k\}_{k=1}^m$

(1). bc_poles() :
Find the places of poles of a barycentric rational approximation.
For such a function:

$$\frac{\sum_{k=1}^m \frac{w_k}{z-z_k}\overline{G}(z_k)}{\sum_{k=1}^m \frac{w_k}{z-z_k}},\ w_k \neq 0,\ z_k \neq z_j\ when\ k \neq j$$

It's poles are solutions of equation: (including Removable singularity)

$$\left| \begin{pmatrix} 0 & w_1 & .. & w_m \\ 1 & & & \\ ... & & \mathrm{diag}\{\mathrm{z}_1, .., \mathrm{z}_\mathrm{m}\} & \\ 1 & & & \end{pmatrix} - \lambda \begin{pmatrix} 0 & \\ & I_m \end{pmatrix} \right| = 0$$

And it can also be proven that all z_k are not solution, which is reasonable.

Input : $\{z_k, w_k\}_{k=1}^m$

Output : recontruct poles $\{\widetilde{P}_k\}_{k=1}^m$

(2). gradient_fd()

Get derivatives of the error function:

$$EA:\{\gamma_k\}_{k=1}^m \to \sum_{j=1}^N \left| \overline{G}(i\omega_j) - \sum_{k=1}^m \frac{\gamma_k}{i\omega_j - pole_k} \right|$$

by finite difference. EA means error of amplitudes.

$\text{Input}: \{\widetilde{P}_k\}_{k=1}^m$

$\text{Output}:\ \text{function}\ \nabla EA(\{\gamma_k\}_{k=1}^m)$

(3). optimize()

Find the minmum point of function EA by Newton's iterative method:

$$x_{k+1} = x_k - H^{-1}(x_k)\nabla EA(x_k)$$

In which H^{-1} is the Hessel matrix:

$$\frac{\partial^2 EA}{\partial x^2}$$

$\text{Input}:\ \text{function}\ EA(x),\ \nabla EA(x)$

$\text{Output}:\ \text{minmum points}$, that is to say, reconstruct amplitudes $\{\widetilde{\gamma}_k\}_{k=1}^m$

(4). Improvement

We find that this method can find the minimum pole with hight accuracy but much less accurate for farther poles. So we find poles on by one. That is to say, after find $\{\widetilde{P}_k, \widetilde{\gamma}_k\}$, we do a update

$$\bar{G}_j^{(k)} \to \bar{G}_j^{(k+1)} = \bar{G}_j^{(k)} - \frac{\gamma_k}{z_j - \widetilde{P}_k},\ \ j \geq k+1$$

And then input $\bar{G}_j^{(k+1)},\ j = k+1, .., m$ and calculate $\{\widetilde{P}_{k+1}, \widetilde{\gamma}_{k+1}\}$

And for the green function question, if we know all poles in advance, we don't need such a method to get poles and amplitudes at all.

Please see coding in examples/FindPolesByACFlow.jl

But it still reminds a problem how to get all amplitudes of poles with high accuracy?

2. Introdunction to AD

(1) ForwardDiff.jl : The way it realizes AD is to use dual number and store the derivatives of the basic functions in advance.

If you have a basic function $f(x)$, then ForwardDiff.jl apply (f, f') on dual number (a, b) to get $(f(a), f'(a)b)$.

3. Where difficult to apply AD

1. svd

   (1) Complex derivative

   Natutal way:

$$df = \frac{\partial f}{\partial z}dz + \frac{\partial f}{\partial z^*}dz^*$$

   When f is a real value function, it's easy to get that df is real sa well, that is to say:

$$df = \frac{1}{2}(u'_x - iu'_y)dz + c.c$$

   (2) Complex gradient

   How define complex gradient for a function? Denote a complex linear function:

$$\nabla_z f(z) = R(z) + iI(z)$$

   When $f(z) = u(z)$, we hope that the complex gradient is the direction of the fastest increase of u, which is to say:

$$\nabla u(z) = u'_x + iu'_y$$

   Now we consider analytic functions:

$$\nabla f(z) = \nabla u(z) + i\nabla v(z)$$
$$= u'_x + iu'_y + i(v'_x + iv'_y)$$
$$= u'_x + iu'_y + i(-u'_y + iu'_x) = 0$$

So we can define $\nabla f(z)$ on $\mathbb{H}(\mathbb{C})$ as

$$\nabla f(z) = k\frac{\partial f}{\partial z^*}$$

And consider $\nabla u(z)$ we get $k = 2$.
Continue this to all complex function, we get:

$$\nabla f(z) = 2\frac{\partial f}{\partial z^*}$$

Specifically, when $f(z)$ is a real value function, we have:

$$\nabla f(z) = 2\left(\frac{\partial f}{\partial z}\right)^*$$

(3) Complex derivative of on $\mathbb{C}^n$
When $f : \mathbb{C}^n \to \mathbb{C}$, we can define its complex derivative on $\mathbb{C}^n$ as:

$$\frac{\partial f}{\partial Z} = \left(\frac{\partial f}{\partial z_i}\right)_{1\le i\le n}$$
$$\nabla f(Z) = 2\left(\frac{\partial f}{\partial z_i^*}\right)_{1\le i\le n}$$

Here we have clear that for

$$Z = (z_{ij})_{1\le i\le n, 1\le j\le m}$$

We define $\frac{\partial f}{\partial Z}$ as:

$$\left(\frac{\partial f}{\partial z_{ij}}\right)_{ij}$$

But not

$$\left(\frac{\partial f}{\partial z_{ji}}\right)_{ij}$$

This also means that if Z is a colunm (row) vector, $\frac{\partial f}{\partial Z}$ is also a row (colunm) vector.

(4) Introduction to AD for svd

Denote

$$\bar{A} = \frac{\partial Loss}{\partial A}$$

Assume that $L(A)$ is a real gauge loss function. Gauge means for a svd composition

$$A = USV^\dagger$$

$$Loss(A) = Loss(U, S, V)$$

has nothing with the choose of U, V.

And assume that A is a matrix has no zero or same eigenvalue.

Then we have:

$$dLoss = \mathrm{Tr}(\bar{A}^T dA + c.c)$$

$$\Longrightarrow \nabla f(A) = 2(\bar{A})^* = 2\left(A_s + A_J + A_K + A_O\right)^*$$

$$A_s^* = U(\bar{S})^* V^\dagger$$
$$A_J^* = U(J^* + J^T)SV^\dagger$$
$$A_K^* = US(K^* + K^T)V^\dagger$$
$$A_O^* = \frac{1}{2}US^{-1}(O - O^\dagger)V^\dagger$$

$$J = F \circ (U^T \bar{U})$$
$$K = F \circ (V^T \bar{V})$$
$$O = I \circ (V^T \bar{V})$$

$$F = \frac{1}{s_j^2 - s_i^2}\chi_{i \neq j}$$

In formulas above, $\circ$ is:

$$(a_{ij})_{n\times n} \circ (b_{ij})_{n\times n} = (a_{ij}b_{ij})_{n\times n}$$

I is identity matrix.

(5) Theoretical validation of method effectiveness.

(a) Gauge freedom

Arbitrary given

$$\text{diag}\{e^{i\theta_1}, .., e^{i\theta_m}\},\ V = [v_1, .., v_m]$$

$$\Longrightarrow V\text{diag} = [.., e^{i\theta_m} v_m]$$

$$\Longrightarrow \|\text{Bary}(v_m e^{i\theta_m}) - A\|_2^2 = \|\text{Bary}(v_m) - A\|_2^2 = Loss$$

(b) Different eigenvalues

In practice, input green function values take some noise and therefore we can assume that all submatrices of L without zero eigenvalues are non-singular with probability 1.

(c) non-zero eigenvalues

Denote size of L is $N \times N$ and size of L_{sub} is $(N - m) \times m$.

For equation:

$$L_{sub} w = G_{sub}$$

In practice, G_{sub} has noise so if $m \leq N/2$ and not full column rank, this equation has no solution with probability 1.

So if $m \leq N/2$, we can think that L_{sub} is full column and therefore has no zero eigenvalue.

1. greedy algorithm

   (1) The perturbation does not affect the choice.

   AD obviously works. Refer an example in examples/ADforGreedy.jl

   (2) The perturbation affects the choice.