Improving performance of contour integral-based nonlinear eigensolvers with infinite GMRES

Yuqi Liu
School of Mathematical Sciences, Fudan University

Joint work with Jose E. Roman and Meiyue Shao

8th ALAMA: Liner Algebra, Matrix Analysis and Applications, Jun, 2024

- Preliminaries
  - Nonlinear eigenvalue problems
  - Contour integral-based nonlinear eigensolvers
  - Infinite GMRES
- Accelerate Beyn's method with infinite GMRES
- Implementation details
  - Weighting
  - Compact representation of the Arnoldi basis
- Numerical experiments

Preliminaries

- Nonlinear eigenvalue problem (NEP):

$$T(\lambda)v = 0, \qquad v \in \mathbb{C}^n \setminus \{0\}, \qquad \lambda \in \Omega.$$

  - Ω: a connected region with a smooth boundary.
  - $T(\xi)$: a holomorphic ξ-dependent matrix in Ω.
  - λ: eigenvalue.
  - v: (right) eigenvector.
- Our goal:
  - To find all eigenvalues $\lambda_1, \ldots, \lambda_k$ lying in Ω,
    as well as their corresponding (right) eigenvectors $v_1, \ldots, v_k$.

- Generalized eigenvalue problems: $(A - \lambda B)v = 0$. ($T(\xi) = A - \xi B$)
- Polynomial eigenvalue problems: $(A_0 + \lambda A_1 + \lambda^2 A_1 + \cdots + \lambda^p A_p)v = 0$.
- Quasi normal mode (QNM) analysis for nanophotonic devices:

$$\sum_{j=1}^{p} r_j(\lambda) L_j v = 0, \qquad r_j(\cdot) : \mathbb{C} \to \mathbb{C}, \qquad L_j \in \mathbb{C}^{n\times n}, \qquad j = 1, \dots, p.$$

  - $r_j(\cdot)$: rational functions.
- Acoustic analysis of the Helmholtz wave equation:

$$(A + \lambda B + \lambda^2 C)v = e^{\mathrm{i}\lambda\tau} S v, \qquad \tau \in \mathbb{R}, \qquad A, B, C, S \in \mathbb{C}^{n\times n}.$$

- Approximate the moments

$$\mathcal{M}_j = \frac{1}{2\pi \mathrm{i}} \int_{\partial\Omega} \xi^j T(\xi)^{-1} Z \, \mathrm{d}\xi,$$

by quadrature rules (for example, the mid point rule)

$$\mathcal{M}_{j,N} = \frac{1}{\mathrm{i}N} \sum_{j=0}^{N-1} \varphi^j(\theta_j)\varphi'(\theta_j) T\big(\varphi(\theta_j)\big)^{-1} Z.$$

And solve some small-size problems with these moments to approximate the eigenpairs.

  - The nonlinear Sakurai-Sugiura algorithm.
  - The nonlinear FEAST algorithm.
  - Beyn's method.

- Drawback: A lot of large, sparse linear systems have to be solved!

- If $T(\xi) = I - \xi A$, we can solve all $T(\xi_j)^{-1} z$, $j = 0, 1, \ldots, N-1$, with a single Arnoldi process.
  - The Krylov subspaces of $I - \xi_j A$ are the same.
  $$\mathcal{K}_m(I - \xi_j A, z) = \mathcal{K}_m(A, z), \qquad j = 0, \ldots, N-1.$$
  - If we have
  $$AU_m = U_{m+1} \underline{H}_m$$
  by the Arnoldi process, we also have
  $$(I - \xi_j A) U_m = U_{m+1} (\underline{I}_m - \xi_j \underline{H}_m).$$
  - The approximate solution can be obtained by solving N small least squares problems
  $$U_m \cdot \arg\min_y \left\| (\underline{I}_m - \xi_j \underline{H}_m) y - \|z\| e_1 \right\|_2.$$

- To solve $T(\xi_0)^{-1}z$ for some ξ_0 close to 0, we can transform it into

$$T(\xi_0)^{-1}z \xrightarrow[(1)]{} \left(\sum_{j=0}^{p} \frac{\xi_0^j}{j!} T^{(j)}(0)\right)^{-1} z \xrightarrow[(2)]{} (\mathcal{L}_0 - \xi_0 \mathcal{L}_1)^{-1}\hat{z} = \mathcal{L}_0^{-1}(I - \xi_0 \mathcal{L}_1 \mathcal{L}_0^{-1})^{-1}\hat{z},$$

where

$$\mathcal{L}_0 = \begin{bmatrix} T(0) & \frac{T^{(1)}(0)}{1!} & \frac{T^{(2)}(0)}{2!} & \cdots & \frac{T^{(p)}(0)}{p!} \\ & I & & & \\ & & I & & \\ & & & \ddots & \\ & & & & I \end{bmatrix}, \quad \mathcal{L}_1 = \begin{bmatrix} 0 & & & & \\ I & 0 & & & \\ & I & \ddots & & \\ & & \ddots & 0 & \\ & & & I & 0 \end{bmatrix}, \quad \hat{z} = \begin{bmatrix} z \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

(1). Taylor expansion at $\xi = 0$.

(2). Companion linearization.

- We can solve several different ξ_0's with a single Arnoldi process on $\mathcal{L}_1 \mathcal{L}_0^{-1}$.
- Advantage: only one factorization on $T(0)$ is needed.

Input: Maximum iteration m, the parameter-dependent matrix $T(\xi): \mathbb{C} \to \mathbb{C}^{n\times n}$, the right-hand side $z \in \mathbb{C}^n$ and the points to be solved ξ_j for $j = 0, \dots, N-1$

Output: Approximations $x_{0,j} \approx T(\xi_j)^{-1}z$ for $j = 0, \dots, N-1$

1: Linearize T to

$$\mathcal{L}_0 = \begin{bmatrix} T(0) & \frac{T^{(1)}(0)}{1!} & \cdots & \frac{T^{(p)}(0)}{p!} \\ & I & & \\ & & \ddots & \\ & & & I \end{bmatrix}, \qquad \mathcal{L}_1 = \begin{bmatrix} 0 & & & \\ I & 0 & & \\ & \ddots & \ddots & \\ & & I & 0 \end{bmatrix}, \qquad p > m$$

2: Perform m iterations of Arnoldi process on $\left(\mathcal{L}_1\mathcal{L}_0^{-1}, \hat{z}\right)$ to obtain $\mathcal{L}_1\mathcal{L}_0^{-1}\mathcal{U}_m = \mathcal{U}_{m+1}\underline{H}_m$

3: Set $y_j \leftarrow \arg\min_y \left\| (\underline{I}_m - \xi_j \underline{H}_m)y - \|z\| e_1 \right\|_2$ for $j = 0, \dots, N-1$

4: Set $x_j \leftarrow L_0^{-1}\mathcal{U}_m y_j$ for $j = 0, \dots, N-1$

5: Set $x_{0,j} \leftarrow x_j(1:n)$ for $j = 0, \dots, N-1$

- Remark: when taking $p > m$, we can assume there is no truncate error introduced by Taylor expansion. In other words, the parameter-dependent matrix $T(\xi)$ is expanded infinitely.

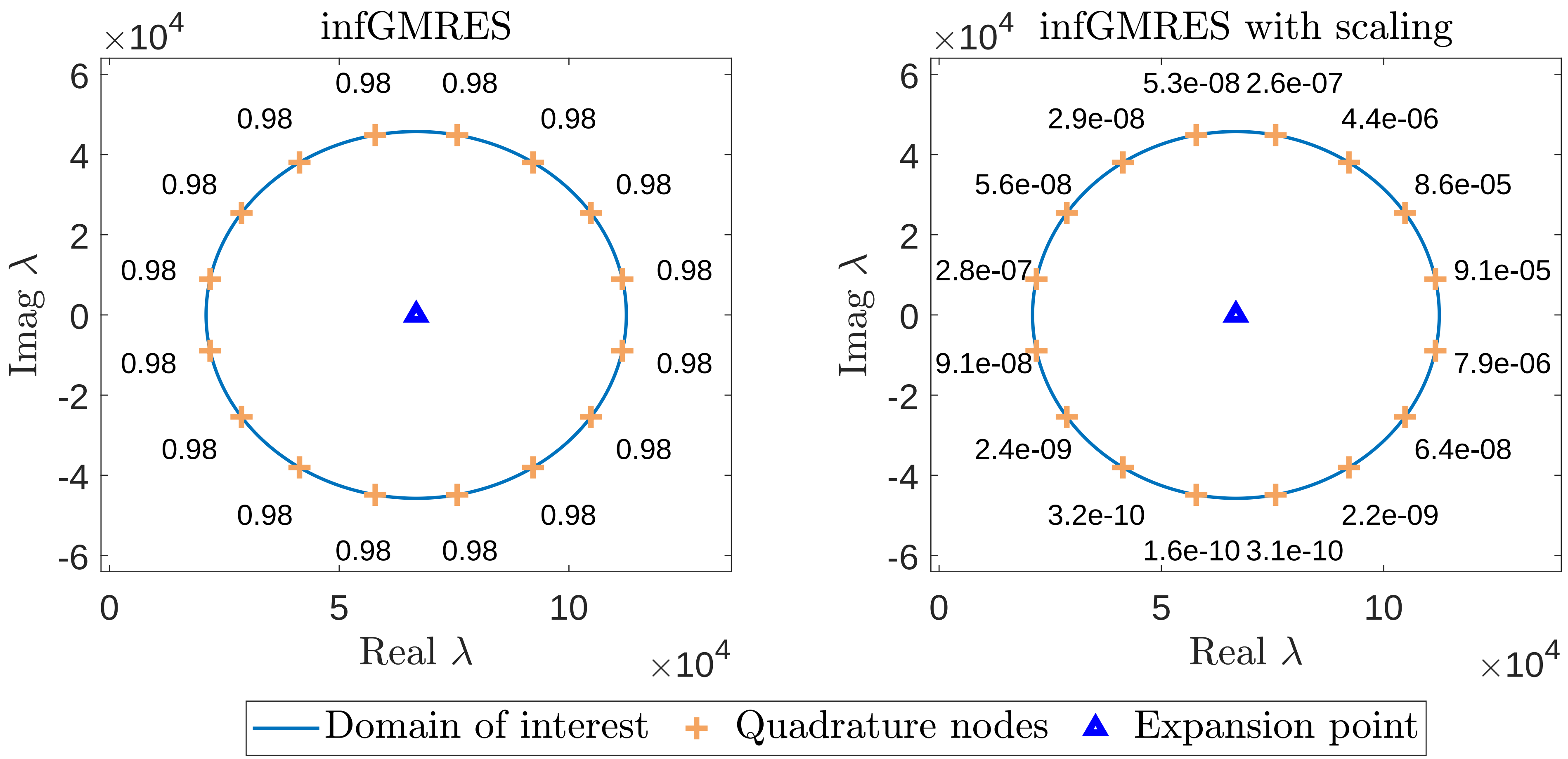
Accelerate Beyn’s method with infinite GMRES

Input: The parameter-dependent matrix $T(\xi)\colon \mathbb{C} \to \mathbb{C}^{n\times n}$, the initial guess $Z = [z_1, \ldots, z_k] \in \mathbb{C}^{n\times k}$, the contour φ and quadrature nodes θ_j's for $j = 0, \ldots, N-1$

Output: Approximate eigenvalues Λ and eigenvectors V

for $s = 1, \ldots, k$ **do**
  Use infGMRES to solve $T\big(\varphi(\theta_j)\big)^{-1} z_s$ for $j = 0, \ldots, N-1$ simultaneously
end for
Set $\mathcal{M}_{0,N} \leftarrow \frac{1}{\mathrm{i}N} \sum_{j=0}^{N-1} \varphi'(\theta_j) T\big(\varphi(\theta_j)\big)^{-1} Z$
Set $\mathcal{M}_{1,N} \leftarrow \frac{1}{\mathrm{i}N} \sum_{j=0}^{N-1} \varphi(\theta_j)\varphi'(\theta_j) T\big(\varphi(\theta_j)\big)^{-1} Z$
Singular value decomposition $\mathcal{M}_{0,N} = V_0 \Sigma_0 W_0^*$
Set $\breve{\mathcal{M}}_{1,N} \leftarrow V_0^* \mathcal{M}_{1,N} W_0 \Sigma_0^{-1}$
Eigenvalue decomposition $\breve{\mathcal{M}}_{1,N} = S \Lambda S^{-1}$
Set $V \leftarrow V_0 S$

- The simple implementation is not efficient enough!
  - High accuracy can not be reached with a modest GMRES iteration m.
  - Storing Arnoldi subspace $\mathcal{U}_m$ requires $\mathcal{O}(m^2 n)$ memory.



- The Taylor expansion is employed on the center of a circular contour.
- The relative residuals of 16 linear parameterized systems of the gun problem are shown.
- The accuracy is obviously higher when scaling $T(a\tilde{\xi} + b)$ is used (right).

- On the scaling technique:
  - Reveal the relationship between NEP and their scaled problems.
    (It does not make sense that a scaling will increase the accuracy of the Taylor expansion.)
  - Provide a novel weighting strategy to accelerate the convergence of infGMRES.
- On the memory usage:
  - Adopt the technique of TOAR to represent the Arnoldi basis compactly (Omitted).

A weighting technique

- Solving $T(\xi)^{-1}z$ at $\xi=\xi_0$ is equivalent to solving $\tilde{T}(\xi)^{-1}z$ at $\xi=\xi_0/\rho$, where $\tilde{T}(\xi)=T(\rho\xi)$.
- The linearization becomes $\left(\tilde{\mathcal{L}}_0-\frac{\xi_0}{\rho}\tilde{\mathcal{L}}_1\right)^{-1}\hat{z}$, where

$$\tilde{\mathcal{L}}_0=\begin{bmatrix} T(0) & \rho\frac{T^{(1)}(0)}{1!} & \rho^2\frac{T^{(2)}(0)}{2!} & \cdots & \rho^p\frac{T^{(p)}(0)}{p!} \\ I & & & & \\ & I & & & \\ & & \ddots & & \\ & & & I & \end{bmatrix}, \qquad \tilde{\mathcal{L}}_1=\mathcal{L}_1.$$

- We notice that

$$\tilde{\mathcal{L}}_0=D_\rho^{-1}\mathcal{L}_0D_\rho, \qquad \frac{1}{\rho}\tilde{\mathcal{L}}_1=D_\rho^{-1}\mathcal{L}_1D_\rho, \qquad \tilde{\mathcal{L}}_0-\frac{\xi_0}{\rho}\tilde{\mathcal{L}}_1=D_\rho^{-1}(\mathcal{L}_0-\xi_0\mathcal{L}_1)D_\rho,$$

where

$$D_\rho=\begin{bmatrix} I & & & \\ & \rho I & & \\ & & \ddots & \\ & & & \rho^pI \end{bmatrix}.$$

- Actually, more degrees of freedom can be introduced.

Lemma 1 *Suppose* $d_j \in \mathbb{C}\backslash\{0\}$ *and* $T_j \in \mathbb{C}^{n\times n}$ *for* $j = 0, \ldots, p$, *and*

$$D = \begin{bmatrix} d_0 I & & & \\ & d_1 I & & \\ & & \ddots & \\ & & & d_p I \end{bmatrix}, \quad \mathcal{L}_0 = \begin{bmatrix} T_0 & T_1 & T_2 & \cdots & T_p \\ & I & & & \\ & & I & & \\ & & & \ddots & \\ & & & & I \end{bmatrix}, \quad \mathcal{L}_1 = \begin{bmatrix} 0 & & & \\ I & 0 & & \\ & \ddots & \ddots & \\ & & I & 0 \end{bmatrix}.$$

Then, for any scalar $\xi \in \mathbb{C}$ *and vector* $z \in \mathbb{C}^n$,

$$\left(D^{-1}(\mathcal{L}_0 - \xi\mathcal{L}_1)D\right)^{-1}\hat{z} = \begin{bmatrix} \left(\sum_{j=0}^{p} \xi^j T_j\right)^{-1} z \\ * \\ \vdots \\ * \end{bmatrix}.$$

- Therefore, the scaling in infGMRES can be regarded as a balanced companion linearization $(D^{-1}\mathcal{L}_0 D, D^{-1}\mathcal{L}_1 D)$.

- If the Krylov subspace $\mathcal{K}_m\big(I-\xi_0\mathcal{L}_1\mathcal{L}_0^{-1},\hat{z}\big)$ is spanned by $\mathcal{U}_m$, by induction, we can prove that $\mathcal{K}_m\big(D^{-1}(I-\xi_0\mathcal{L}_1\mathcal{L}_0^{-1})D,\hat{z}\big)$ is spanned by $D^{-1}\mathcal{U}_m$.
- At the mth iteration, infGMRES will give the solution by

$$\begin{aligned} y_* &= \arg\min_y \left\| \big(D^{-1}(I-\xi_0\mathcal{L}_1\mathcal{L}_0^{-1})D\big)\big(D^{-1}\mathcal{U}_m\big)y-\hat{z} \right\|_2 \\ &= \arg\min_y \left\| D^{-1}\big((I-\xi_0\mathcal{L}_1\mathcal{L}_0^{-1})\mathcal{U}_m y-\hat{z}\big) \right\|_2 . \end{aligned}$$

- Idea: we can choose D appropriately to guide GMRES to pick a more accurate solution from the search space!
  - Our goal is to minimize $\|r_{\mathrm{N}}\|_2 = \left\|T(\xi_0)x_0 - z\right\|_2$, or $\|r_{\mathrm{P}}\|_2 = \left\|\sum_{j=0}^{p}\xi_0^j T_j x_0 - z\right\|_2$, where $x_0 = (D^{-1}\mathcal{L}_0^{-1}\mathcal{U}_m y_*)(1:n)$.

Lemma 2 *Suppose vector $y_* \in \mathbb{C}^m$ is the solution for some weighting matrix D. Then, the polynomial-wise residual $r_{\mathrm{P}} = \sum_{j=0}^{p} \xi_0^j T_j x_0 - z$ can be represented as*

$$r_{\mathrm{P}} = \left[I, -\sum_{j=1}^{p} \xi_0^{j-1} T_j, -\sum_{j=2}^{p} \xi_0^{j-2} T_j, \ldots, -T_p \right] \begin{bmatrix} r_0 \\ r_1 \\ \vdots \\ r_p \end{bmatrix},$$

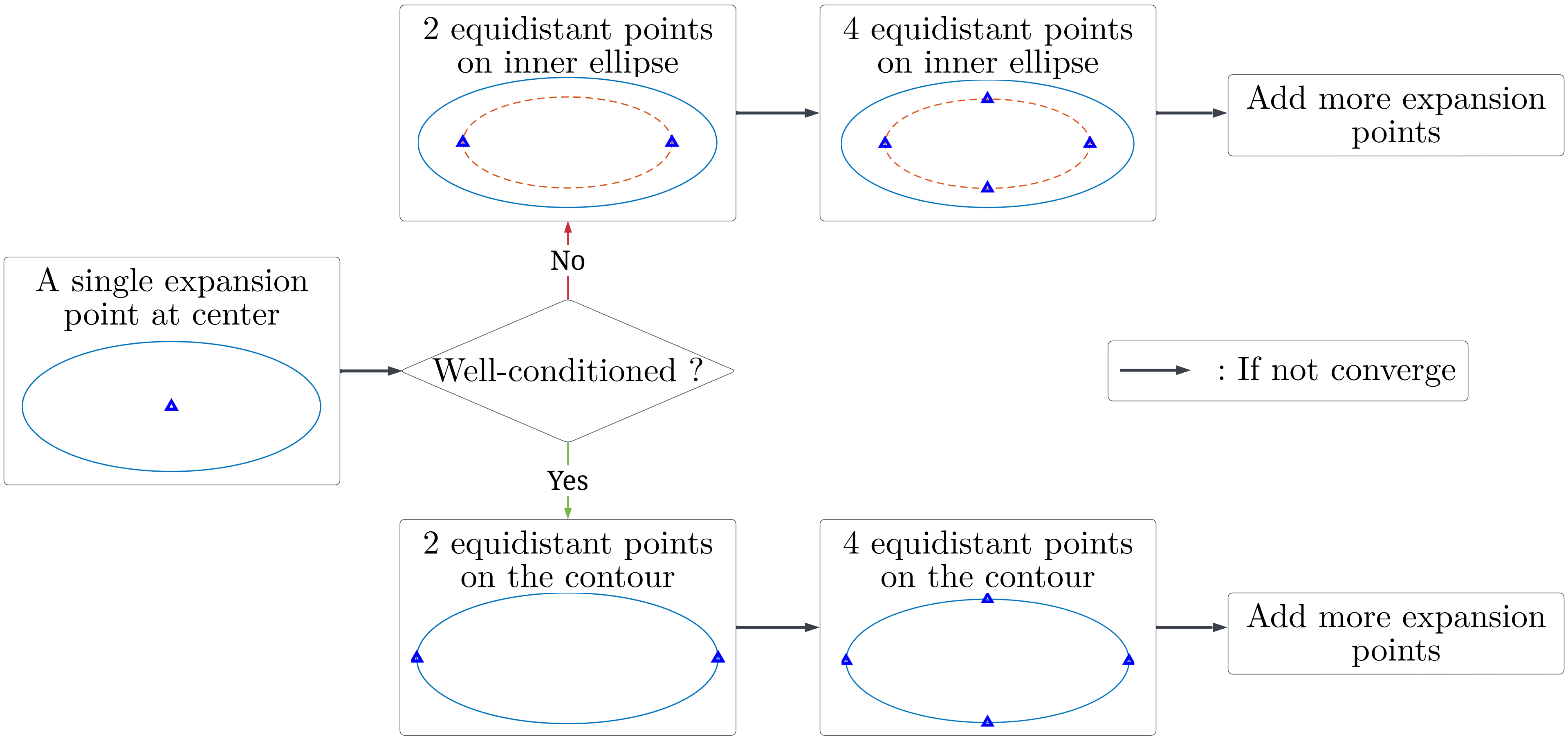
where

$$\begin{bmatrix} r_0 \\ r_1 \\ \vdots \\ r_p \end{bmatrix} = (I - \xi_0 \mathcal{L}_1 \mathcal{L}_0^{-1}) \mathcal{U}_m y_* - \hat{z}.$$

- r_j can be adjusted by D.
- We can decrease the residual by balancing the sum

$$\|r_{\mathrm{P}}\| \leq \|r_0\|_2 + \left\| \sum_{j=1}^{p} \xi_0^{j-1} T_j \right\|_2 \|r_1\|_2 + \cdots + \|T_p\|_2 \|r_p\|_2.$$

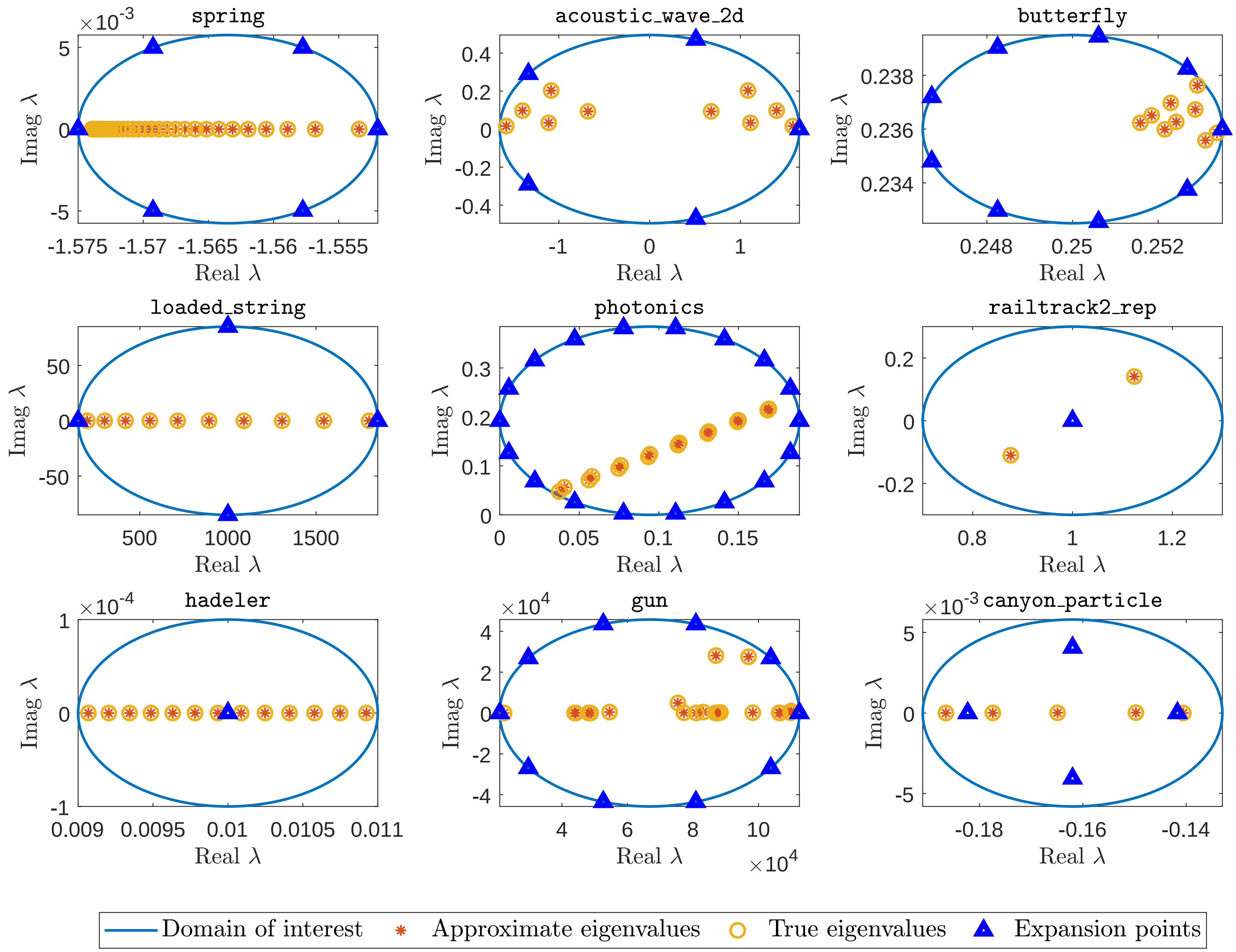
How to choose expansion points

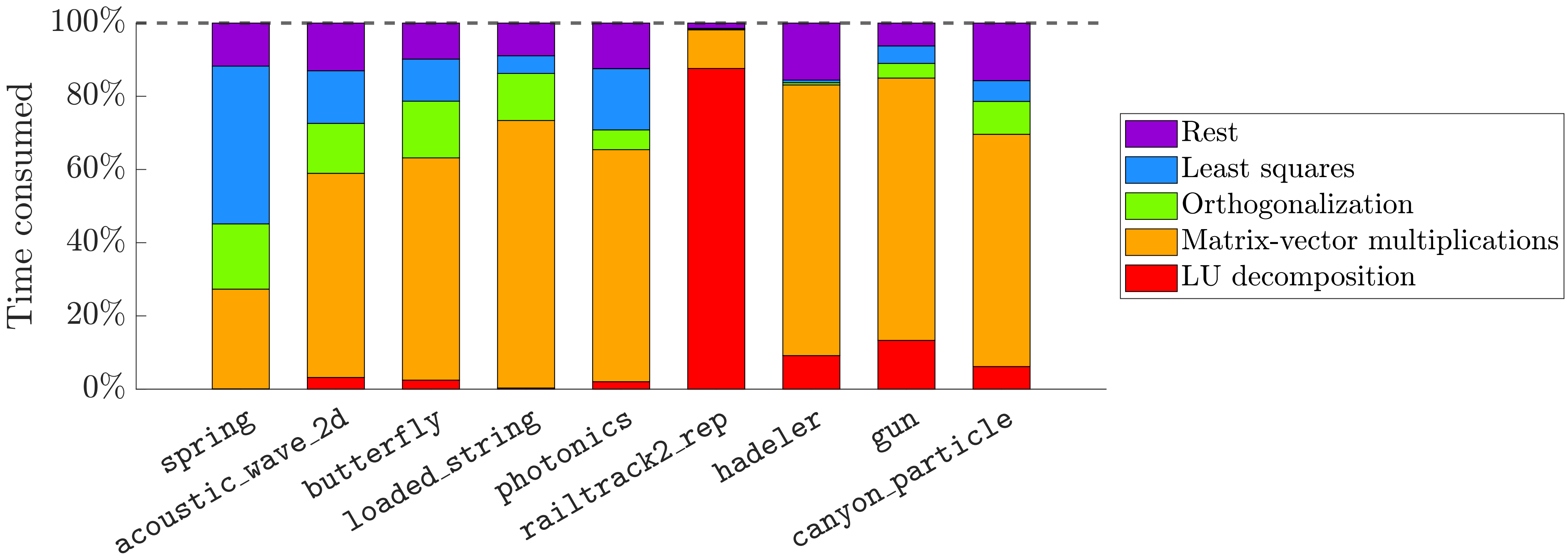

2 equidistant points on inner ellipse
4 equidistant points on inner ellipse
Add more expansion points
A single expansion point at center
No
Well-conditioned ?
: If not converge
Yes
2 equidistant points on the contour
4 equidistant points on the contour
Add more expansion points

Numerical Experiments

- Information of test problems.
  - n: the size of the problem.
  - k: the number of the eigenvalues to be computed.
  - N: the number of quadrature nodes.
  - n_{ep}: the number of expansion points.
  - t_{s}, t_{iG}: the times consumed by Beyn's method with MATLAB backslash and with infGMRES.

Problem	Type	n	k	N	n_{ep}	t_{s}(s)	t_{iG}(s)	$(t_{\text{s}} - t_{\text{iG}})/t_{\text{s}}$
spring	QEP	3000	32	1024	6	3.472	15.72	−353%
acoustic_wave_2d	QEP	9900	10	512	5	28.14	9.475	66%
butterfly	PEP	5000	9	512	9	11.54	8.453	27%
loaded_string	REP	20000	10	128	4	1.07	9.805	−816%
photonics	REP	20363	16	3060	18	600	209.7	65%
railtrack2_rep	REP	35955	2	128	1	533.4	71.95	87%
hadeler	NEP	5000	13	32	1	40.2	46.6	−16%
gun	NEP	9956	21	1024	10	501.2	148.9	70%
canyon_particle	NEP	16281	5	256	4	40.94	7.652	81%


spring
acoustic_wave_2d
butterfly
loaded_string
photonics
railtrack2_rep
hadeler
gun
canyon_particle
Real λ
Imag λ
Domain of interest
Approximate eigenvalues
True eigenvalues
Expansion points



- Most of time is consumed by the Arnoldi process (matrix–vector multiplications and orthogonalization).
- When the size of the matrices grow higher, the time consumed by matrix factorization increases dramatically.

- Summary
  - infGMRES can be employed in contour integral-based nonlinear eigensolvers to reduce the cost of solving linear systems.
  - We propose the convergence-accelerating weighting strategy, the memory-friendly TOAR-like trick, and the selection strategies of the parameters, making infGMRES robust and efficient in practice.
  - Our algorithm can achieve a speedup up to 7x on the test examples.

- Ongoing work
  - Develop a machine learning-based adaptive strategy for selecting expansion points automatically.
  - Implement a block variant of infGMRES.

Thank you for your attention!

Contact: yuqliu21@m.fudan.edu.cn

Appendix

Compact representation of the Arnoldi basis

- The Arnoldi process on $\mathcal{L}_1\mathcal{L}_0^{-1}$ can be rearranged into

$$\mathcal{L}_1\mathcal{L}_0^{-1}\mathcal{U}_m = \mathcal{L}_1\mathcal{L}_0^{-1}\begin{bmatrix} Q_m\breve{U}_{m,0} \\ \vdots \\ Q_m\breve{U}_{m,p}\end{bmatrix} = \begin{bmatrix} Q_m\breve{U}_{m,0} & [Q_m, q_{m+1}]\breve{u}_{m+1,0} \\ \vdots & \vdots \\ Q_m\breve{U}_{m,p} & [Q_m, q_{m+1}]\breve{u}_{m+1,p}\end{bmatrix}\underline{H}_m = \mathcal{U}_{m+1}\underline{H}_m,$$

where

$$Q_m \in \mathbb{C}^{n\times m}, \quad Q_m^*Q_m = I, \quad \breve{U}_{m,j} \in \mathbb{C}^{m\times m}, \quad j = 0, \ldots, p.$$

- Then, we can set

$$Q_{m+1} \leftarrow [Q_m, q_{m+1}], \qquad \breve{U}_{m+1,j} \leftarrow \left[\begin{array}{c|c} \breve{U}_{m,j} & \breve{u}_{m+1,j-1} \\ 0 & \end{array}\right].$$

- To store $\mathcal{U}_m$ in this way, we need to store Q_m by $\mathcal{O}(mn)$ memory and $\breve{U}_{m,j}$ for $j = 0$, $\ldots$, $m-1$ by $\mathcal{O}(m^3)$ memory.
- When n is very large, $\mathcal{O}(mn + m^3) \ll \mathcal{O}(m^2n)$.

Comparison
on different weighting strategies

- No weight: $D = I$.
- Scalar weight from [Betcke 2009]:

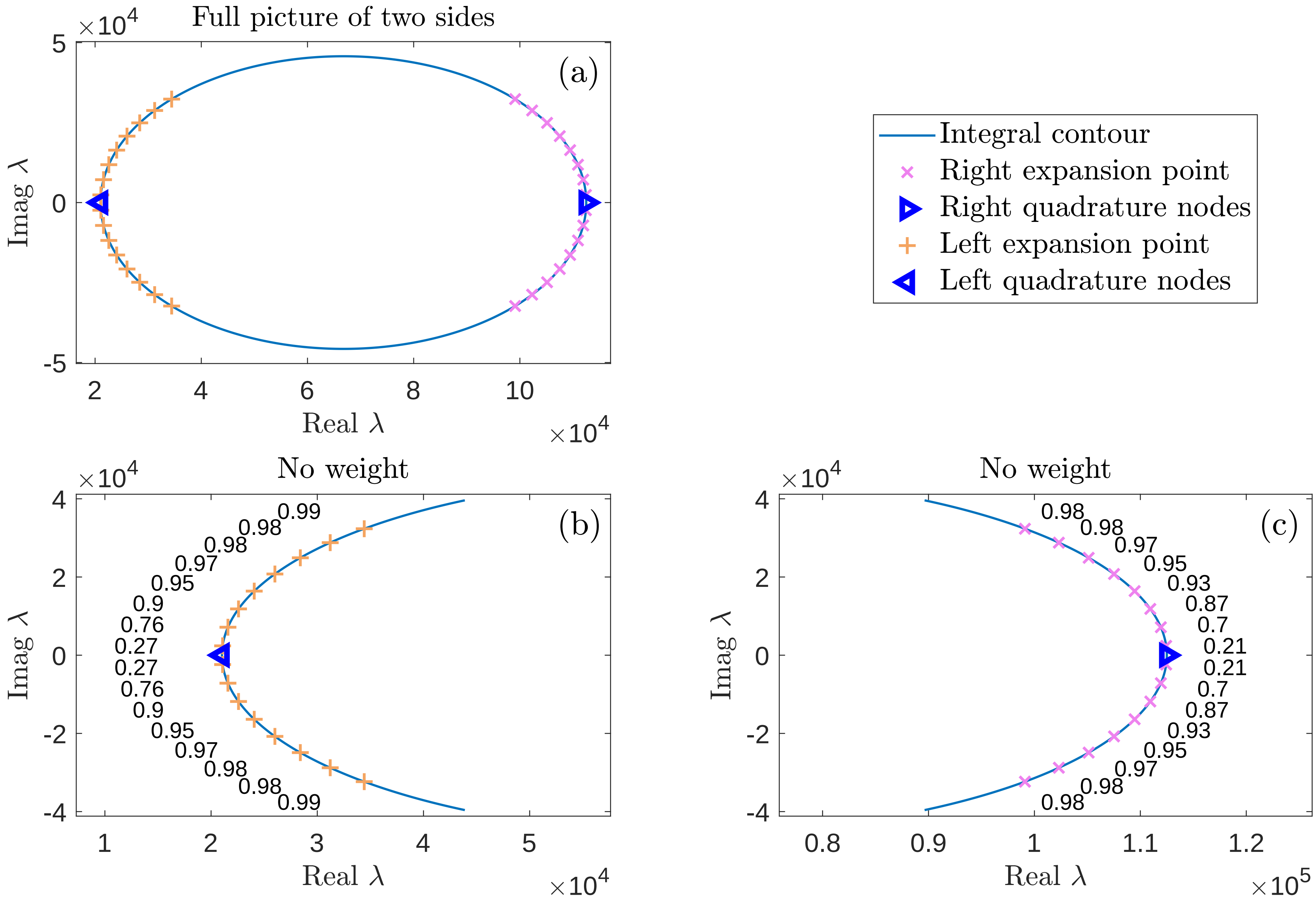
$$\rho = \left(\frac{\|T(0)\|_2}{\|T^{(p)}(0)/p!\|_2} \right)^{1/p}, \qquad D = \begin{bmatrix} I & & & \\ & \rho I & & \\ & & \ddots & \\ & & & \rho^p I \end{bmatrix}.$$

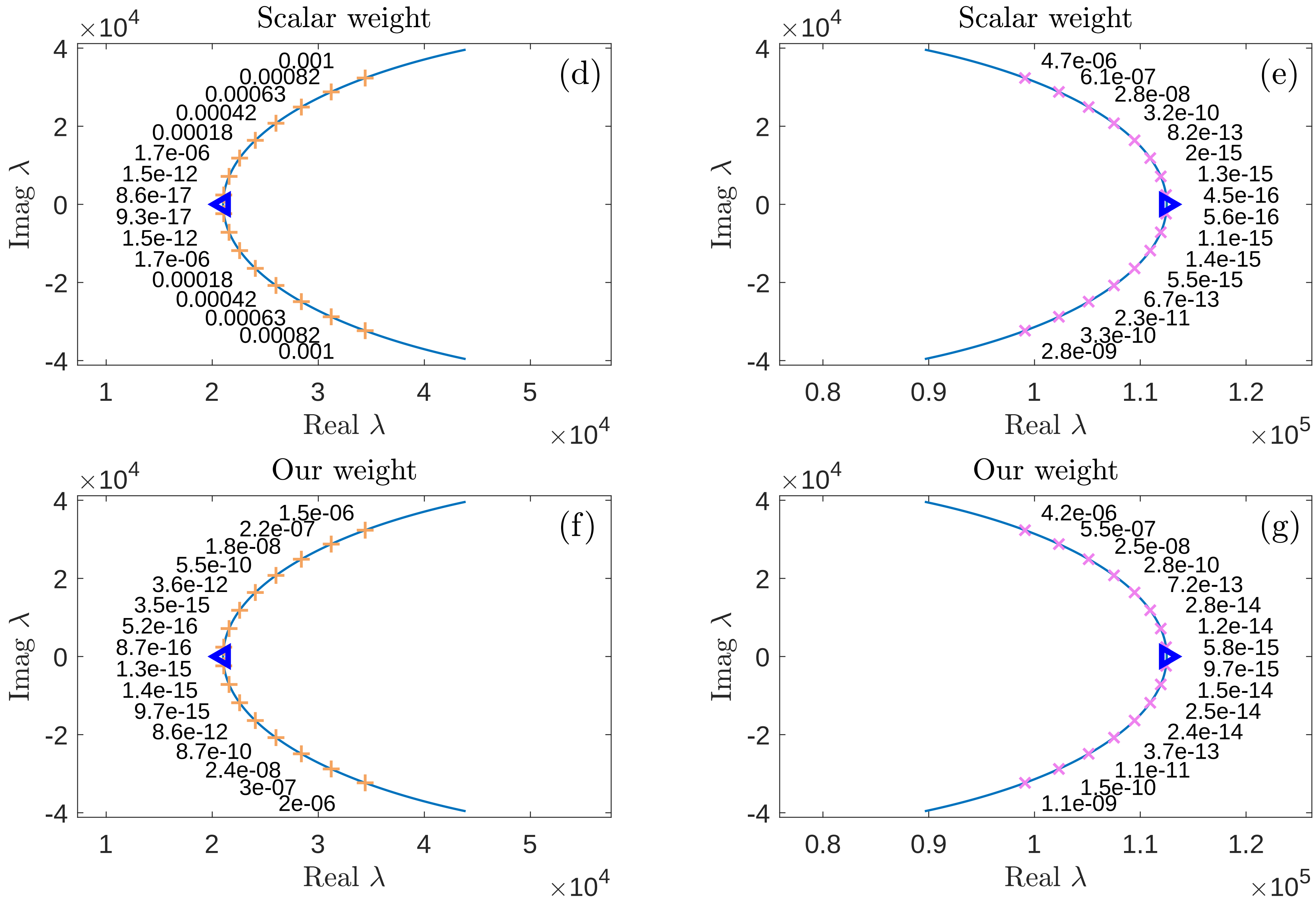
- Our weight:

$$d_0 = 1, \quad d_s = \frac{\gamma}{\|\sum_{j=s}^{p} \xi_0^{j-s} T_j\|_2}, \quad \gamma = \frac{\|\sum_{j=2}^{p} \xi_0^{j-2} T_j\|_2^2}{\|\sum_{j=3}^{p} \xi_0^{j-3} T_j\|_2}, \quad s = 1, \ldots, p,$$

$$D = \begin{bmatrix} d_0 I & & & \\ & d_1 I & & \\ & & \ddots & \\ & & & d_p I \end{bmatrix}.$$

Comparison on different weighting strategies




Scalar weight
(d)
Imag λ
Real λ
×10^4
0.001
0.00082
0.00063
0.00042
0.00018
1.7e-06
1.5e-12
8.6e-17
9.3e-17
1.5e-12
1.7e-06
0.00018
0.00042
0.00063
0.00082
0.001
Scalar weight
(e)
×10^5
4.7e-06
6.1e-07
2.8e-08
3.2e-10
8.2e-13
2e-15
1.3e-15
4.5e-16
5.6e-16
1.1e-15
1.4e-15
5.5e-15
6.7e-13
2.3e-11
3.3e-10
2.8e-09
Our weight
(f)
1.5e-06
2.2e-07
1.8e-08
5.5e-10
3.6e-12
3.5e-15
5.2e-16
8.7e-16
1.3e-15
1.4e-15
9.7e-15
8.6e-12
8.7e-10
2.4e-08
3e-07
2e-06
Our weight
(g)
4.2e-06
5.5e-07
2.5e-08
2.8e-10
7.2e-13
2.8e-14
1.2e-14
5.8e-15
9.7e-15
1.5e-14
2.5e-14
2.4e-14
3.7e-13
1.1e-11
1.5e-10
1.1e-09