An introduction to fractional calculus

Fundamental ideas and numerics

Fabio Durastante

Università di Pisa

fabio.durastante@unipi.it

fdurastante.github.io

October, 2022

The curse of dimensionality

In the last lesson we saw that:

- Circulant preconditioners **cannot cluster** cases with variable coefficients,

The curse of dimensionality

In the last lesson we saw that:

- Circulant preconditioners **cannot cluster** cases with variable coefficients,
- Multilevel circulant preconditioners **cannot cluster** multilevel Toeplitz systems,

The curse of dimensionality

In the last lesson we saw that:

- Circulant preconditioners **cannot cluster** cases with variable coefficients,
- Multilevel circulant preconditioners **cannot cluster** multilevel Toeplitz systems,
- Preconditioner based on matrix algebras with fast simultaneous diagonalization **cannot cluster** multilevel Toeplitz systems.

The curse of dimensionality

In the last lesson we saw that:

- Circulant preconditioners **cannot cluster** cases with variable coefficients,
- Multilevel circulant preconditioners **cannot cluster** multilevel Toeplitz systems,
- Preconditioner based on matrix algebras with fast simultaneous diagonalization **cannot cluster** multilevel Toeplitz systems.
- Can we **reduce the dimensionality** of the problem to reuse the information and good results we have in the 1D case?

The curse of dimensionality

In the last lesson we saw that:

- Circulant preconditioners **cannot cluster** cases with variable coefficients,
- Multilevel circulant preconditioners **cannot cluster** multilevel Toeplitz systems,
- Preconditioner based on matrix algebras with fast simultaneous diagonalization **cannot cluster** multilevel Toeplitz systems.
- Can we **reduce the dimensionality** of the problem to reuse the information and good results we have in the 1D case?
  - Alternating-direction implicit method,

The curse of dimensionality

In the last lesson we saw that:

- Circulant preconditioners **cannot cluster** cases with variable coefficients,
- Multilevel circulant preconditioners **cannot cluster** multilevel Toeplitz systems,
- Preconditioner based on matrix algebras with fast simultaneous diagonalization **cannot cluster** multilevel Toeplitz systems.
- Can we **reduce the dimensionality** of the problem to reuse the information and good results we have in the 1D case?
  - Alternating-direction implicit method,
  - reformulation as *matrix equations*,

The curse of dimensionality

In the last lesson we saw that:

- Circulant preconditioners **cannot cluster** cases with variable coefficients,
- Multilevel circulant preconditioners **cannot cluster** multilevel Toeplitz systems,
- Preconditioner based on matrix algebras with fast simultaneous diagonalization **cannot cluster** multilevel Toeplitz systems.
- Can we **reduce the dimensionality** of the problem to reuse the information and good results we have in the 1D case?
  - Alternating-direction implicit method,
  - reformulation as *matrix equations*,
  - reformulation as *tensor problems*.

Matrix equation reformulations

The simplest way of introducing this reformulation is to go back to the 1D problem (now with a *source term*):

$$\begin{cases} \frac{\partial W}{\partial t} = \theta \, {}^{RL}D^{\alpha}_{[0,x]} W(x,t) + (1-\theta)\, {}^{RL}D^{\alpha}_{[x,1]} W(x,t) + f(x,t), \qquad \theta \in [0,1], \\ W(0,t) = W(1,t) = 0, \qquad W(x,t) = W_0(x). \end{cases}$$

Matrix equation reformulations

The simplest way of introducing this reformulation is to go back to the 1D problem (now with a *source term*):

$$\begin{cases} \frac{\partial W}{\partial t} = \theta\, {}^{GL}D^{\alpha}_{[0,x]} W(x,t) + (1-\theta)\, {}^{GL}D^{\alpha}_{[x,1]} W(x,t) + f(x,t), & \theta \in [0,1], \\ W(0,t) = W(1,t) = 0, \qquad W(x,t) = W_0(x). \end{cases}$$

To solve everything we have to solve the **sequence of linear systems**

$$\frac{1}{\Delta t}\left(\mathbf{W}^{(m+1)} - \mathbf{W}^{(m)}\right) = \frac{1}{h^{\alpha}}\left(\theta G_N + (1-\theta) G_N^T\right) \mathbf{W}^{(m+1))} + \mathbf{f}^{(m+1)}, \; m = 0, \ldots, M-1.$$

Matrix equation reformulations

The simplest way of introducing this reformulation is to go back to the 1D problem (now with a *source term*):

$$\begin{cases} \frac{\partial W}{\partial t} = \theta \, {}^{GL}D^{\alpha}_{[0,x]} W(x,t) + (1-\theta) {}^{GL}D^{\alpha}_{[x,1]} W(x,t) + f(x,t), & \theta \in [0,1], \\ W(0,t) = W(1,t) = 0, \qquad W(x,t) = W_0(x). \end{cases}$$

To solve everything we have to solve the **sequence of linear systems**

$$\frac{1}{\Delta t}\left(\mathbf{W}^{(m+1)} - \mathbf{W}^{(m)}\right) = \frac{1}{h^{\alpha}}\left(\theta G_N + (1-\theta) G_N^T\right) \mathbf{W}^{(m+1))} + \mathbf{f}^{(m+1)}, \; m = 0, \ldots, M-1.$$

Do we really have to solve this sequentially?

Matrix equation reformulations

Following (Breiten, Simoncini, and Stoll 2016), we can **collect the time steps altogether**:

$$\left(B_M \otimes I_N - \frac{\Delta t}{h^\alpha} I_M \otimes T_N \right) \hat{\mathbf{W}} = \mathbf{F},$$

since

$$\begin{bmatrix} I_N - \frac{\Delta t}{h^\alpha} T_N & & & \\ -I_N & I_N - \frac{\Delta t}{h^\alpha} T_N & & \\ & \ddots & \ddots & \\ & & -I_N & I_N - \frac{\Delta t}{h^\alpha} T_N \end{bmatrix} \begin{bmatrix} \mathbf{W}^{(1)} \\ \mathbf{W}^{(2)} \\ \vdots \\ \mathbf{W}^{(M-1)} \end{bmatrix} = \begin{bmatrix} \mathbf{W}^{(0)} + \Delta t \mathbf{f}^{(1)} \\ \Delta t \mathbf{f}^{(2)} \\ \vdots \\ \Delta t \mathbf{f}^{(M)}, \end{bmatrix}$$

for $T_N = \left(\theta G_N + (1-\theta) G_N^T\right)$, $B_M = T_M(1 - e^{i\theta})$.

Matrix equation reformulations

Following (Breiten, Simoncini, and Stoll 2016), we can **collect the time steps altogether**:

$$\left(B_M \otimes I_N - \frac{\Delta t}{h^\alpha} I_M \otimes T_N\right) \hat{\mathbf{W}} = \mathbf{F},$$

since

$$\begin{bmatrix} I_N - \frac{\Delta t}{h^\alpha} T_N & & & \\ -I_N & I_N - \frac{\Delta t}{h^\alpha} T_N & & \\ & \ddots & \ddots & \\ & & -I_N & I_N - \frac{\Delta t}{h^\alpha} T_N \end{bmatrix} \begin{bmatrix} \mathbf{W}^{(1)} \\ \mathbf{W}^{(2)} \\ \vdots \\ \mathbf{W}^{(M-1)} \end{bmatrix} = \begin{bmatrix} \mathbf{W}^{(0)} + \Delta t \mathbf{f}^{(1)} \\ \Delta t \mathbf{f}^{(2)} \\ \vdots \\ \Delta t \mathbf{f}^{(M)}, \end{bmatrix}$$

for $T_N = \left(\theta G_N + (1-\theta) G_N^T\right)$, $B_M = T_M(1-e^{i\theta})$.

This is now a **coupled system** of size $MN \times MN$, that is larger and uglier than before...

Matrix equation reformulations

❓ Where is the advantage in dealing with

$$\left(B_M \otimes I_N - \frac{\Delta t}{h^{\alpha}} I_M \otimes T_N \right) \hat{\mathbf{W}} = \mathbf{F}?$$

Matrix equation reformulations

❓ Where is the advantage in dealing with

$$\left(B_M \otimes I_N + I_M \otimes A_N\right) \hat{\mathbf{W}} = \mathbf{F}, \qquad A_N = -\frac{\Delta t}{h^\alpha} T_N?$$

Matrix equation reformulations

Where is the advantage in dealing with

$$(B_M \otimes I_N + I_M \otimes A_N)\,\hat{\mathbf{W}} = \mathbf{F}, \qquad A_N = -\frac{\Delta t}{h^\alpha} T_N?$$

Let's call $W = [\mathbf{W}^{(1)}|\cdots|\mathbf{W}^{(M)}]_{N\times M}$, $F = [\mathbf{W}^{(0)} + \Delta t\mathbf{f}^{(1)}|\cdots|\Delta t\mathbf{f}^{M}]_{N\times M}$, and rewrite our problem as

Compute $W \in \mathbb{R}^{N\times M}$ s.t. $A_N W + W B_M^T = F$.

Matrix equation reformulations

Where is the advantage in dealing with

$$(B_M \otimes I_N + I_M \otimes A_N)\,\hat{\mathbf{W}} = \mathbf{F}, \qquad A_N = -\frac{\Delta t}{h^\alpha} T_N?$$

Let's call $W = [\mathbf{W}^{(1)}|\cdots|\mathbf{W}^{(M)}]_{N\times M}$, $F = [\mathbf{W}^{(0)} + \Delta t \mathbf{f}^{(1)}|\cdots|\Delta t \mathbf{f}^{M}]_{N\times M}$, and rewrite our problem as

Compute $W \in \mathbb{R}^{N\times M}$ s.t. $A_N W + W B_M^T = F$.

This is a well-know object called **Sylvester equation**!

Matrix equation reformulations

Where is the advantage in dealing with

$$(B_M \otimes I_N + I_M \otimes A_N)\,\hat{\mathbf{W}} = \mathbf{F}, \qquad A_N = -\frac{\Delta t}{h^\alpha} T_N?$$

Let's call $W = [\mathbf{W}^{(1)}|\cdots|\mathbf{W}^{(M)}]_{N\times M}$, $F = [\mathbf{W}^{(0)} + \Delta t\mathbf{f}^{(1)}|\cdots|\Delta t\mathbf{f}^{M}]_{N\times M}$, and rewrite our problem as

Compute $W \in \mathbb{R}^{N\times M}$ s.t. $A_N W + W B_M^T = F$.

This is a well-know object called **Sylvester equation**!

Did we gain anything?

Matrix equation reformulations

❓ Where is the advantage in dealing with

$$(B_M \otimes I_N + I_M \otimes A_N)\,\hat{\mathbf{W}} = \mathbf{F}, \qquad A_N = -\frac{\Delta t}{h^\alpha} T_N?$$

Let's call $W = [\mathbf{W}^{(1)}|\cdots|\mathbf{W}^{(M)}]_{N\times M}$, $F = [\mathbf{W}^{(0)} + \Delta t\mathbf{f}^{(1)}|\cdots|\Delta t\mathbf{f}^{M}]_{N\times M}$, and rewrite our problem as

$$\text{Compute } W \in \mathbb{R}^{N\times M} \text{ s.t. } A_N W + W B_M^T = F.$$

- This is a well-know object called **Sylvester equation**!
- ❓ Did we gain anything? Back to this in a few moment...

Matrix equation reformulations

❓ Where is the advantage in dealing with

$$(B_M \otimes I_N + I_M \otimes A_N)\,\hat{\mathbf{W}} = \mathbf{F}, \qquad A_N = -\frac{\Delta t}{h^{\alpha}} T_N?$$

Let's call $W = [\mathbf{W}^{(1)}|\cdots|\mathbf{W}^{(M)}]_{N\times M}$, $F = [\mathbf{W}^{(0)} + \Delta t\mathbf{f}^{(1)}|\cdots|\Delta t\mathbf{f}^{M}]_{N\times M}$, and rewrite our problem as

Compute $W \in \mathbb{R}^{N\times M}$ s.t. $A_N W + W B_M^T = F$.

- This is a well-know object called **Sylvester equation**!
- ❓ Did we gain anything? Back to this in a few moment...
- ❓ Since we are accumulating all the time steps in one step, is it appropriate to simply use one of the methods we already know (e.g. Euler, BDFs, Adams', etc.) or can we do better? Next lecture!

What about the 2D problem?

What happens if we want then to reformulate:

$$\begin{cases} \frac{\partial W}{\partial t} = \left(\theta\, {}^{RL}D^{\alpha}_{[0,x]} \cdot + (1-\theta)^{RL}D^{\alpha}_{[x,1]} \cdot\right) W(x,y,t) \\ \qquad\qquad + \left(\theta\, {}^{RL}D^{\alpha}_{[0,y]} \cdot + (1-\theta)^{RL}D^{\alpha}_{[y,1]} \cdot\right) W(x,y,t) + f(x,y,t), \qquad \theta \in [0,1], \\ W(0,t) = W(1,t) = 0, \qquad W(x,t) = W_0(x). \end{cases}$$

What about the 2D problem?

What happens if we want then to reformulate:

$$\begin{cases} \frac{\partial W}{\partial t} = \left(\theta \, {}^{GL}D^{\alpha}_{[0,x]} \cdot + (1-\theta) \, {}^{GL}D^{\alpha}_{[x,1]} \cdot \right) W(x,y,t) \\ \qquad\qquad + \left(\theta \, {}^{GL}D^{\alpha}_{[0,y]} \cdot + (1-\theta) \, {}^{GL}D^{\alpha}_{[y,1]} \cdot \right) W(x,y,t) + f(x,y,t), \qquad \theta \in [0,1], \\ W(0,t) = W(1,t) = 0, \qquad W(x,t) = W_0(x). \end{cases}$$

What about the 2D problem?

What happens if we want then to reformulate:

$$\begin{cases} \frac{\partial W}{\partial t} = \left(\theta\, {}^{GL}D^{\alpha}_{[0,x]} \cdot + (1-\theta)\, {}^{GL}D^{\alpha}_{[x,1]} \cdot \right) W(x,y,t) \\ \qquad\qquad + \left(\theta\, {}^{GL}D^{\alpha}_{[0,y]} \cdot + (1-\theta)\, {}^{GL}D^{\alpha}_{[y,1]} \cdot \right) W(x,y,t) + f(x,y,t), \qquad \theta \in [0,1], \\ W(0,t) = W(1,t) = 0, \qquad W(x,t) = W_0(x). \end{cases}$$

By the usual procedure

$$\frac{1}{\Delta t}\left(\mathbf{W}^{(m+1)} - \mathbf{W}^{(m)}\right) = \frac{1}{h^{\alpha}}\left((\theta G_{N_x} + (1-\theta) G_{N_x}^T) \otimes I_{N_y} + I_{N_x} \otimes (\theta G_{N_y} + (1-\theta) G_{N_y}^T)\right)\mathbf{W}^{(m+1)}$$
$$+ \mathbf{f}^{(m+1)}, \quad m = 0, \ldots, M-1.$$

What about the 2D problem?

What happens if we want then to reformulate:

$$\begin{cases} \frac{\partial W}{\partial t} = \left(\theta\, {}^{GL}D^{\alpha}_{[0,x]} \cdot + (1-\theta)\, {}^{GL}D^{\alpha}_{[x,1]} \cdot\right) W(x,y,t) \\ \qquad\qquad + \left(\theta\, {}^{GL}D^{\alpha}_{[0,y]} \cdot + (1-\theta)\, {}^{GL}D^{\alpha}_{[y,1]} \cdot\right) W(x,y,t) + f(x,y,t), \qquad \theta \in [0,1], \\ W(0,t) = W(1,t) = 0, \qquad W(x,t) = W_0(x). \end{cases}$$

By the usual procedure

$$\frac{1}{\Delta t}\left(\mathbf{W}^{(m+1)} - \mathbf{W}^{(m)}\right) = \left(\tilde{G}_{N_x} \otimes I_{N_y} + I_{N_x} \otimes \tilde{G}_{N_y}\right)\mathbf{W}^{(m+1)} + \mathbf{f}^{(m+1)}, \quad m = 0, \ldots, M-1.$$

What about the 2D problem?

What happens if we want then to reformulate:

$$\begin{cases} \frac{\partial W}{\partial t} = \left(\theta \, {}^{GL}D^{\alpha}_{[0,x]} \cdot + (1-\theta)\, {}^{GL}D^{\alpha}_{[x,1]} \cdot\right) W(x,y,t) \\ \qquad\qquad + \left(\theta \, {}^{GL}D^{\alpha}_{[0,y]} \cdot + (1-\theta)\, {}^{GL}D^{\alpha}_{[y,1]} \cdot\right) W(x,y,t) + f(x,y,t), \qquad \theta \in [0,1], \\ W(0,t) = W(1,t) = 0, \qquad W(x,t) = W_0(x). \end{cases}$$

By the usual procedure

$$\frac{1}{\Delta t}\left(\mathbf{W}^{(m+1)} - \mathbf{W}^{(m)}\right) = \underbrace{\left(\tilde{G}_{N_x} \otimes I_{N_y} + I_{N_x} \otimes \tilde{G}_{N_y}\right)}_{G_{N_x N_y}} \mathbf{W}^{(m+1)} + \mathbf{f}^{(m+1)}, \quad m = 0, \ldots, M-1.$$

What about the 2D problem?

What happens if we want then to reformulate:

$$\begin{cases} \frac{\partial W}{\partial t} = \left(\theta\, {}^{GL}D^{\alpha}_{[0,x]} \cdot + (1-\theta)\, {}^{GL}D^{\alpha}_{[x,1]} \cdot\right) W(x,y,t) \\ \qquad\quad + \left(\theta\, {}^{GL}D^{\alpha}_{[0,y]} \cdot + (1-\theta)\, {}^{GL}D^{\alpha}_{[y,1]} \cdot\right) W(x,y,t) + f(x,y,t), \qquad \theta \in [0,1], \\ W(0,t) = W(1,t) = 0, \qquad W(x,t) = W_0(x). \end{cases}$$

By the usual procedure

$$\left(I_{N_x N_y} - \Delta t G_{N_x N_y}\right) \mathbf{W}^{(m+1)} = \mathbf{W}^{(m)} + \Delta \mathbf{f}^{m+1}, \quad m = 0, \ldots, M-1.$$

What about the 2D problem?

What happens if we want then to reformulate:

$$\begin{cases} \frac{\partial W}{\partial t} = \left(\theta\, {}^{GL}D^{\alpha}_{[0,x]} \cdot + (1-\theta)\, {}^{GL}D^{\alpha}_{[x,1]} \cdot\right) W(x,y,t) \\ \qquad\quad + \left(\theta\, {}^{GL}D^{\alpha}_{[0,y]} \cdot + (1-\theta)\, {}^{GL}D^{\alpha}_{[y,1]} \cdot\right) W(x,y,t) + f(x,y,t), \qquad \theta \in [0,1], \\ W(0,t) = W(1,t) = 0, \qquad W(x,t) = W_0(x). \end{cases}$$

By the usual procedure

$$\left(I_{N_x N_y} - \Delta t G_{N_x N_y}\right) \mathbf{W}^{(m+1)} = \mathbf{W}^{(m)} + \Delta \mathbf{f}^{m+1}, \quad m = 0, \ldots, M-1.$$

The **clever observation** is now that

$$I_{N_x N_y} - \Delta t G_{N_x N_y} = I_{N_y} \otimes \left(\frac{1}{2} I_{N_x} - \Delta t \tilde{G}_{N_x}\right) + \left(\frac{1}{2} I_{N_y} - \Delta t \tilde{G}_{N_y}\right) \otimes I_{N_x}.$$

What about the 2D problem?

What happens if we want then to reformulate:

$$\begin{cases} \frac{\partial W}{\partial t} = \left(\theta\, {}^{GL}D^{\alpha}_{[0,x]} \cdot + (1-\theta)\, {}^{GL}D^{\alpha}_{[x,1]} \cdot \right) W(x,y,t) \\ \qquad\qquad + \left(\theta\, {}^{GL}D^{\alpha}_{[0,y]} \cdot + (1-\theta)\, {}^{GL}D^{\alpha}_{[y,1]} \cdot \right) W(x,y,t) + f(x,y,t), \qquad \theta \in [0,1], \\ W(0,t) = W(1,t) = 0, \qquad W(x,t) = W_0(x). \end{cases}$$

By the usual procedure

$$\left(\frac{1}{2} I_{N_x} - \Delta t \tilde{G}_{N_x}\right) \tilde{W}^{(m+1)} + \tilde{W}^{(m+1)} \left(\frac{1}{2} I^{N_y} - \Delta t \tilde{G}_{N_y}\right)^T = \tilde{W}^{(m)} + \Delta t F^{(m+1)}, \quad m = 0, \ldots, M-1.$$

The **clever observation** is now that

$$I_{N_x N_y} - \Delta t G_{N_x N_y} = I_{N_y} \otimes \left(\frac{1}{2} I_{N_x} - \Delta t \tilde{G}_{N_x}\right) + \left(\frac{1}{2} I_{N_y} - \Delta t \tilde{G}_{N_y}\right) \otimes I_{N_x}.$$

What about the 2D problem?

What happens if we want then to reformulate:

$$\begin{cases} \frac{\partial W}{\partial t} = \left(\theta \, {}^{GL}D^{\alpha}_{[0,x]} \cdot + (1-\theta) \, {}^{GL}D^{\alpha}_{[x,1]} \cdot \right) W(x,y,t) \\ \qquad\quad + \left(\theta \, {}^{GL}D^{\alpha}_{[0,y]} \cdot + (1-\theta) \, {}^{GL}D^{\alpha}_{[y,1]} \cdot \right) W(x,y,t) + f(x,y,t), \qquad \theta \in [0,1], \\ W(0,t) = W(1,t) = 0, \qquad W(x,t) = W_0(x). \end{cases}$$

By the usual procedure

$$\left(\frac{1}{2} I_{N_x} - \Delta t \tilde{G}_{N_x}\right) \tilde{W}^{(m+1)} + \tilde{W}^{(m+1)} \left(\frac{1}{2} I^{N_y} - \Delta t \tilde{G}_{N_y}\right)^T = \tilde{W}^{(m)} + \Delta t F^{(m+1)}, \quad m = 0, \ldots, M-1.$$

- We now have a **sequence of Sylvester equations** for $m = 0, \ldots, M-1$.
- The matrix coefficients are related to *rescaled* 1D problems.

Solving Sylvester equations (Simoncini 2016)

This rewriting effort will be worth it only if we can **efficiently solve** Sylvester equations:

$$AX + XB = C, \quad A \in \mathbb{R}^{N\times N},\ B \in \mathbb{R}^{M\times M}, \quad C \in \mathbb{R}^{N\times M}.$$

Solving Sylvester equations (Simoncini 2016)

This rewriting effort will be worth it only if we can **efficiently solve** Sylvester equations:

$$AX + XB = C, \quad A \in \mathbb{R}^{N\times N},\ B \in \mathbb{R}^{M\times M}, \quad C \in \mathbb{R}^{N\times M}.$$

The solution can be expressed in **closed form** in a number of ways, e.g., as *integrals of resolvents*

$$X = -\frac{1}{4\pi^2}\int_{\Gamma_1}\int_{\Gamma_2} \frac{(\gamma I_N - A)^{-1} C (\mu I_M - B)^{-1}}{\lambda + \mu}\,\mathrm{d}\mu\mathrm{d}\lambda,$$

for Γ_1, Γ_2 contours containing and sufficiently close to the spectra of A and B, respectively.

Solving Sylvester equations (Simoncini 2016)

This rewriting effort will be worth it only if we can **efficiently solve** Sylvester equations:

$$AX + XB = C, \quad A \in \mathbb{R}^{N\times N},\ B \in \mathbb{R}^{M\times M}, \quad C \in \mathbb{R}^{N\times M}.$$

The solution can be expressed in **closed form** in a number of ways, e.g., as *integrals of exponentials*

$$X = -\int_0^{+\infty} e^{At} C e^{Bt}\, \mathrm{d}t,$$

for A and B with a spectra separated by a vertical line.

Solving Sylvester equations (Simoncini 2016)

This rewriting effort will be worth it only if we can **efficiently solve** Sylvester equations:

$$AX + XB = C, \quad A \in \mathbb{R}^{N\times N},\ B \in \mathbb{R}^{M\times M}, \quad C \in \mathbb{R}^{N\times M}.$$

The solution can be expressed in **closed form** in a number of ways, e.g., in the *diagonalizable case*, by means of *similarity transformations*

$$U^{-1}AU = \mathrm{diag}(\lambda_1, \dots, \lambda_N), \qquad V^{-1}BV = \mathrm{diag}(\mu_1, \dots, \mu_M),$$

then

$$X = U\tilde{X}V, \quad \tilde{x}_{i,j} = \frac{1}{\lambda_i + \mu_j}(U^{-1}CV)_{i,j}.$$

Solving Sylvester equations (Simoncini 2016)

This rewriting effort will be worth it only if we can **efficiently solve** Sylvester equations:

$$AX + XB = C, \quad A \in \mathbb{R}^{N\times N},\ B \in \mathbb{R}^{M\times M}, \quad C \in \mathbb{R}^{N\times M}.$$

The solution can be expressed in **closed form** in a number of ways, e.g., in the *diagonalizable case*, by means of *similarity transformations*

$$U^{-1}AU = \mathrm{diag}(\lambda_1,\ldots,\lambda_N), \qquad V^{-1}BV = \mathrm{diag}(\mu_1,\ldots,\mu_M),$$

then

$$X = U\tilde{X}V, \quad \tilde{x}_{i,j} = \frac{1}{\lambda_i + \mu_j}(U^{-1}CV)_{i,j}.$$

Numerical mehtods

These formulations can exploited to devise numerical methods, to avoid a very long detour, we are going to just mention a couple of them; read (Simoncini 2016) for the full story.

The Bartels and Stewart 1972 algorithm

Input: A, B, C
Compute Schur factorizations
$URU^H = A^H$ and $B = VSV^H$;
Solve $R^H Y + YS = U^H CV$ for Y;
Compute $X = UYV^H$;

The Bartels and Stewart 1972 algorithm

- The first step costs $O(N^3)$ and $O(M^3)$ operations by **QR algorithm** for general A and B,

Input: A, B, C
Compute Schur factorizations $URU^H = A^H$ and $B = VSV^H$;
Solve $R^H Y + YS = U^H CV$ for Y;
Compute $X = UYV^H$;

The Bartels and Stewart 1972 algorithm

- The first step costs $O(N^3)$ and $O(M^3)$ operations by **QR algorithm** for general A and B,
- The last step is just two matrix-matrix multiplications.

Input: A, B, C
Compute Schur factorizations $URU^H = A^H$ and $B = VSV^H$;
Solve $R^H Y + YS = U^H CV$ for Y;
Compute $X = UYV^H$;

The Bartels and Stewart 1972 algorithm

- The first step costs $O(N^3)$ and $O(M^3)$ operations by **QR algorithm** for general A and B,
- The last step is just two matrix-matrix multiplications.

Input: A, B, C
Compute Schur factorizations $URU^H = A^H$ and $B = VSV^H$;
Solve $R^H Y + YS = U^H CV$ for Y;
Compute $X = UYV^H$;

We can solve the system with triangular coefficients by substitution

$$R^H Y + YS = U^H CV$$

The Bartels and Stewart 1972 algorithm

- The first step costs $O(N^3)$ and $O(M^3)$ operations by **QR algorithm** for general A and B,
- The last step is just two matrix-matrix multiplications.

Input: A, B, C
Compute Schur factorizations $URU^H = A^H$ and $B = VSV^H$;
Solve $R^H Y + YS = U^H CV$ for Y;
Compute $X = UYV^H$;

We can solve the system with triangular coefficients by substitution

$$\begin{bmatrix} \spadesuit & & \\ \spadesuit & \spadesuit & \\ \spadesuit & \spadesuit & \spadesuit \end{bmatrix} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} + \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \begin{bmatrix} \clubsuit & \clubsuit & \clubsuit \\ & \clubsuit & \clubsuit \\ & & \clubsuit \end{bmatrix} = \begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}$$

The Bartels and Stewart 1972 algorithm

- The first step costs $O(N^3)$ and $O(M^3)$ operations by **QR algorithm** for general A and B,
- The last step is just two matrix-matrix multiplications.

Input: A, B, C
Compute Schur factorizations $URU^H = A^H$ and $B = VSV^H$;
Solve $R^H Y + YS = U^H CV$ for Y;
Compute $X = UYV^H$;

We can solve the system with triangular coefficients by substitution

$$\begin{bmatrix} \spadesuit & & \\ \spadesuit & \spadesuit & \\ \spadesuit & \spadesuit & \spadesuit \end{bmatrix} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} + \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \begin{bmatrix} \clubsuit & \clubsuit & \clubsuit \\ & \clubsuit & \clubsuit \\ & & \clubsuit \end{bmatrix} = \begin{bmatrix} \bigstar & \bigstar & \bigstar \\ \bigstar & \bigstar & \bigstar \\ \bigstar & \bigstar & \bigstar \end{bmatrix}$$

$(Y)_{1,1}$ element is readily obtained by solving: $(\spadesuit + \clubsuit)(Y)_{11} = \bigstar$.

The Bartels and Stewart 1972 algorithm

- The first step costs $O(N^3)$ and $O(M^3)$ operations by **QR algorithm** for general A and B,
- The last step is just two matrix-matrix multiplications.

Input: A, B, C
Compute Schur factorizations $URU^H = A^H$ and $B = VSV^H$;
Solve $R^H Y + YS = U^H CV$ for Y;
Compute $X = UYV^H$;

We can solve the system with triangular coefficients by substitution

$$\begin{bmatrix} \spadesuit & & \\ \spadesuit & \spadesuit & \\ \spadesuit & \spadesuit & \spadesuit \end{bmatrix} \begin{bmatrix} y_{1,1} & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} + \begin{bmatrix} y_{11} & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \begin{bmatrix} \clubsuit & \clubsuit & \clubsuit \\ & \clubsuit & \clubsuit \\ & & \clubsuit \end{bmatrix} = \begin{bmatrix} \bigstar & \bigstar & \bigstar \\ \bigstar & \bigstar & \bigstar \\ \bigstar & \bigstar & \bigstar \end{bmatrix}$$

Then we proceed with the first row...

The Bartels and Stewart 1972 algorithm

- The first step costs $O(N^3)$ and $O(M^3)$ operations by **QR algorithm** for general A and B,
- The last step is just two matrix-matrix multiplications.

Input: A, B, C
Compute Schur factorizations $URU^H = A^H$ and $B = VSV^H$;
Solve $R^H Y + YS = U^H CV$ for Y;
Compute $X = UYV^H$;

We can solve the system with triangular coefficients by substitution

$$\begin{bmatrix} \spadesuit & & \\ \spadesuit & \spadesuit & \\ \spadesuit & \spadesuit & \spadesuit \end{bmatrix} \begin{bmatrix} y_{1,1} & y_{12} & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} + \begin{bmatrix} y_{11} & y_{12} & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \begin{bmatrix} \clubsuit & \clubsuit & \clubsuit \\ & \clubsuit & \clubsuit \\ & & \clubsuit \end{bmatrix} = \begin{bmatrix} \bigstar & \bigstar & \bigstar \\ \bigstar & \bigstar & \bigstar \\ \bigstar & \bigstar & \bigstar \end{bmatrix}$$

Then we proceed with the first row...

The Bartels and Stewart 1972 algorithm

- The first step costs $O(N^3)$ and $O(M^3)$ operations by **QR algorithm** for general A and B,
- The last step is just two matrix-matrix multiplications.

Input: A, B, C
Compute Schur factorizations $URU^H = A^H$ and $B = VSV^H$;
Solve $R^H Y + YS = U^H CV$ for Y;
Compute $X = UYV^H$;

We can solve the system with triangular coefficients by substitution

$$\begin{bmatrix} \spadesuit & & \\ \spadesuit & \spadesuit & \\ \spadesuit & \spadesuit & \spadesuit \end{bmatrix} \begin{bmatrix} y_{1,1} & y_{12} & y_{1,3} \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} + \begin{bmatrix} y_{11} & y_{12} & y_{1,3} \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \begin{bmatrix} \clubsuit & \clubsuit & \clubsuit \\ & \clubsuit & \clubsuit \\ & & \clubsuit \end{bmatrix} = \begin{bmatrix} \bigstar & \bigstar & \bigstar \\ \bigstar & \bigstar & \bigstar \\ \bigstar & \bigstar & \bigstar \end{bmatrix}$$

Then we proceed with the first row... and every other row.

The Bartels and Stewart 1972 algorithm

- The first step costs $O(N^3)$ and $O(M^3)$ operations by **QR algorithm** for general A and B,
- The last step is just two matrix-matrix multiplications.

Input: A, B, C
Compute Schur factorizations $URU^H = A^H$ and $B = VSV^H$;
Solve $R^H Y + YS = U^H CV$ for Y;
Compute $X = UYV^H$;

We can solve the system with triangular coefficients by substitution

$$\begin{bmatrix} \spadesuit & & \\ \spadesuit & \spadesuit & \\ \spadesuit & \spadesuit & \spadesuit \end{bmatrix} \begin{bmatrix} y_{1,1} & y_{12} & y_{1,3} \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} + \begin{bmatrix} y_{11} & y_{12} & y_{1,3} \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \begin{bmatrix} \clubsuit & \clubsuit & \clubsuit \\ & \clubsuit & \clubsuit \\ & & \clubsuit \end{bmatrix} = \begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}$$

Then we proceed with the first row... and every other row.

The Bartels and Stewart 1972 algorithm

- The first step costs $O(N^3)$ and $O(M^3)$ operations by **QR algorithm** for general A and B,
- The last step is just two matrix-matrix multiplications.

Input: A, B, C
Compute Schur factorizations $URU^H = A^H$ and $B = VSV^H$;
Solve $R^H Y + YS = U^H CV$ for Y;
Compute $X = UYV^H$;

We can solve the system with triangular coefficients by substitution

$$\begin{bmatrix} \spadesuit & & \\ \spadesuit & \spadesuit & \\ \spadesuit & \spadesuit & \spadesuit \end{bmatrix} \begin{bmatrix} y_{1,1} & y_{12} & y_{1,3} \\ y_{2,1} & y_{2,2} & y_{2,3} \\ \times & \times & \times \end{bmatrix} + \begin{bmatrix} y_{11} & y_{12} & y_{1,3} \\ y_{2,1} & y_{2,2} & y_{2,3} \\ \times & \times & \times \end{bmatrix} \begin{bmatrix} \clubsuit & \clubsuit & \clubsuit \\ & \clubsuit & \clubsuit \\ & & \clubsuit \end{bmatrix} = \begin{bmatrix} \bigstar & \bigstar & \bigstar \\ \bigstar & \bigstar & \bigstar \\ \bigstar & \bigstar & \bigstar \end{bmatrix}$$

Then we proceed with the first row... and every other row.

The Bartels and Stewart 1972 algorithm

- The first step costs $O(N^3)$ and $O(M^3)$ operations by **QR algorithm** for general A and B,
- The last step is just two matrix-matrix multiplications.

Input: A, B, C
Compute Schur factorizations $URU^H = A^H$ and $B = VSV^H$;
Solve $R^H Y + YS = U^H CV$ for Y;
Compute $X = UYV^H$;

We can solve the system with triangular coefficients by substitution

$$\begin{bmatrix} \spadesuit & & \\ \spadesuit & \spadesuit & \\ \spadesuit & \spadesuit & \spadesuit \end{bmatrix} \begin{bmatrix} y_{1,1} & y_{12} & y_{1,3} \\ y_{2,1} & y_{2,2} & y_{2,3} \\ y_{3,1} & y_{3,2} & y_{3,3} \end{bmatrix} + \begin{bmatrix} y_{11} & y_{12} & y_{1,3} \\ y_{2,1} & y_{2,2} & y_{2,3} \\ y_{3,1} & y_{3,2} & y_{3,3} \end{bmatrix} \begin{bmatrix} \clubsuit & \clubsuit & \clubsuit \\ & \clubsuit & \clubsuit \\ & & \clubsuit \end{bmatrix} = \begin{bmatrix} \star & \star & \star \\ \star & \star & \star \\ \star & \star & \star \end{bmatrix}$$

Then we proceed with the first row... and every other row.
$\Rightarrow$ The **leading cost** is the Schur factorization $O(N^3 + M^3)$! ☹ only **small matrices**.

The Bartels and Stewart 1972 algorithm

- The first step costs $O(N^3)$ and $O(M^3)$ operations by **QR algorithm** for general A and B,
- The last step is just two matrix-matrix multiplications.

Input: A, B, C
Compute Schur factorizations $URU^H = A^H$ and $B = VSV^H$;
Solve $R^H Y + YS = U^H CV$ for Y;
Compute $X = UYV^H$;

We can solve the system with triangular coefficients by substitution

$$\begin{bmatrix} \spadesuit & & \\ \spadesuit & \spadesuit & \\ \spadesuit & \spadesuit & \spadesuit \end{bmatrix} \begin{bmatrix} y_{1,1} & y_{12} & y_{1,3} \\ y_{2,1} & y_{2,2} & y_{2,3} \\ y_{3,1} & y_{3,2} & y_{3,3} \end{bmatrix} + \begin{bmatrix} y_{11} & y_{12} & y_{1,3} \\ y_{2,1} & y_{2,2} & y_{2,3} \\ y_{3,1} & y_{3,2} & y_{3,3} \end{bmatrix} \begin{bmatrix} \clubsuit & \clubsuit & \clubsuit \\ & \clubsuit & \clubsuit \\ & & \clubsuit \end{bmatrix} = \begin{bmatrix} \bigstar & \bigstar & \bigstar \\ \bigstar & \bigstar & \bigstar \\ \bigstar & \bigstar & \bigstar \end{bmatrix}$$

Then we proceed with the first row... and every other row.
$\Rightarrow$ The **leading cost** is the Schur factorization $O(N^3 + M^3)$! only **small matrices**.
We may gain something if A and B are in **upper Hessenberg form**...

The small case scenario

There are a number of variations that we can apply for the case of small matrices

The small case scenario

There are a number of variations that we can apply for the case of small matrices

- We can use **real Schur form** instead of the complex one, avoids complex arithmetic, but now for in the second step we have to solve some Sylvester equation with 2×2 coefficients. We do it by going back to a small linear system.

The small case scenario

There are a number of variations that we can apply for the case of small matrices

- We can use **real Schur form** instead of the complex one, avoids complex arithmetic, but now for in the second step we have to solve some Sylvester equation with 2×2 coefficients. We do it by going back to a small linear system.
- We can go directly for the Hessenberg form instead of Schur (Golub, Nash, and Van Loan 1979).

The small case scenario

There are a number of variations that we can apply for the case of small matrices

- We can use **real Schur form** instead of the complex one, avoids complex arithmetic, but now for in the second step we have to solve some Sylvester equation with 2×2 coefficients. We do it by going back to a small linear system.
- We can go directly for the Hessenberg form instead of Schur (Golub, Nash, and Van Loan 1979).
- If A is much larger than B then we can work on the block case

$$\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} B = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}.$$

The small case scenario

There are a number of variations that we can apply for the case of small matrices

- We can use **real Schur form** instead of the complex one, avoids complex arithmetic, but now for in the second step we have to solve some Sylvester equation with 2×2 coefficients. We do it by going back to a small linear system.
- We can go directly for the Hessenberg form instead of Schur (Golub, Nash, and Van Loan 1979).
- If A is much larger than B then we can work on the block case

$$\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} B = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}.$$

The small case scenario

There are a number of variations that we can apply for the case of small matrices

- We can use **real Schur form** instead of the complex one, avoids complex arithmetic, but now for in the second step we have to solve some Sylvester equation with 2×2 coefficients. We do it by going back to a small linear system.
- We can go directly for the Hessenberg form instead of Schur (Golub, Nash, and Van Loan 1979).
- If A is much larger than B then we can work on the block case.
- If $B = -A$ and C is *small rank*, then we are falling back to our "fast small-displacement-rank solver" scenario, e.g., (Gohberg, Kailath, and Olshevsky 1995).

The small case scenario

There are a number of variations that we can apply for the case of small matrices

- We can use **real Schur form** instead of the complex one, avoids complex arithmetic, but now for in the second step we have to solve some Sylvester equation with 2×2 coefficients. We do it by going back to a small linear system.
- We can go directly for the Hessenberg form instead of Schur (Golub, Nash, and Van Loan 1979).
- If A is much larger than B then we can work on the block case.
- If $B = -A$ and C is *small rank*, then we are falling back to our "fast small-displacement-rank solver" scenario, e.g., (Gohberg, Kailath, and Olshevsky 1995).

But our cases are not small...

If only we knew a way to from a large matrix setting, to a small one made of Hessenberg matrices...

The small case scenario

There are a number of variations that we can apply for the case of small matrices

- We can use **real Schur form** instead of the complex one, avoids complex arithmetic, but now for in the second step we have to solve some Sylvester equation with 2×2 coefficients. We do it by going back to a small linear system.
- We can go directly for the Hessenberg form instead of Schur (Golub, Nash, and Van Loan 1979).
- If A is much larger than B then we can work on the block case.
- If $B = -A$ and C is *small rank*, then we are falling back to our "fast small-displacement-rank solver" scenario, e.g., (Gohberg, Kailath, and Olshevsky 1995).

☺ But our cases are not small. . .

If only we knew a way to from a large matrix setting, to a small one made of Hessenberg matrices. . . wait a second, we may know a trick or two for this! ☺

When in doubt: project!

When we have to solve **linear systems** with a **large matrix**, we have seen that a good solution is represented by the Krylov projection methods.

❓ Can we do something similar for this problem too?

Theorem (Simoncini 2016, Theorem 4)

Let A and B be stable[1] and real symmetric, with spectra contained in $[a, b]$ and $[c, d]$, respectively. Define $\eta = 2(b-a)(d-c)/((a+c)(b+d))$. Assume C is of **rank** p. Then the singular values $\sigma_1 \geq \cdots \geq \sigma_{\min\{M,N\}}$ of the solution X to the Sylvester equation satisfy

$$\frac{\sigma_{pr+1}}{\sigma_1} \leq \left(\frac{1-\sqrt{k_r'}}{1+\sqrt{k_r'}}\right)^2, \; 1 \leq pr < n, \; k_r' = \frac{1}{1+\eta+\sqrt{\eta(\eta+2)}}.$$

[1] A matrix is called stable (or sometimes *Hurwitz*) if every eigenvalue has strictly negative real part.

When in doubt: project!

When we have to solve **linear systems** with a **large matrix**, we have seen that a good solution is represented by the Krylov projection methods.

❓ Can we do something similar for this problem too? *sometimes*, it is a **matter of rank**.

Theorem (Simoncini 2016, Theorem 4)

Let A and B be stable[1] and real symmetric, with spectra contained in $[a, b]$ and $[c, d]$, respectively. Define $\eta = 2(b-a)(d-c)/((a+c)(b+d))$. Assume C is of **rank** p. Then the singular values $\sigma_1 \geq \cdots \geq \sigma_{\min\{M,N\}}$ of the solution X to the Sylvester equation satisfy

$$\frac{\sigma_{pr+1}}{\sigma_1} \leq \left(\frac{1-\sqrt{k_r'}}{1+\sqrt{k_r'}} \right)^2, \; 1 \leq pr < n, \; k_r' = \frac{1}{1+\eta+\sqrt{\eta(\eta+2)}}.$$

[1]A matrix is called stable (or sometimes *Hurwitz*) if every eigenvalue has strictly negative real part.

Projection methods for low-rank right-hand sides

Let us **assume** that $\mathrm{rank}\,(C) = p \ll \min\{n, m\}$.

Projection methods for low-rank right-hand sides

Let us **assume** that $\mathrm{rank}\,(C) = p \ll \min\{n, m\}$.

- Decompose $C = C_1 C_2^H$;

Projection methods for low-rank right-hand sides

Let us **assume** that $\mathrm{rank}\,(C) = p \ll \min\{n, m\}$.

- Decompose $C = C_1 C_2^H$;
- Select $\mathcal{V} = \mathrm{span}(V_k)$ and $\mathcal{W} = \mathrm{span}(W_j)$ subspaces of $\mathbb{C}^N$ and $\mathbb{C}^M$;

Projection methods for low-rank right-hand sides

Let us **assume** that $\mathrm{rank}\,(C) = p \ll \min\{n, m\}$.

- Decompose $C = C_1 C_2^H$;
- Select $\mathcal{V} = \mathrm{span}(V_k)$ and $\mathcal{W} = \mathrm{span}(W_j)$ subspaces of $\mathbb{C}^N$ and $\mathbb{C}^M$;
- Basis V_k, $k \ll N$, and W_j, $j \ll M$, are *orthonormal* and such that $\mathcal{V}$ and $\mathcal{W}$ are not orthogonal to $\mathrm{range}(C_1)$ and $\mathrm{range}(C_2)$ respectively;

Projection methods for low-rank right-hand sides

Let us **assume** that $\operatorname{rank}(C) = p \ll \min\{n, m\}$.

- Decompose $C = C_1 C_2^H$;
- Select $\mathcal{V} = \operatorname{span}(V_k)$ and $\mathcal{W} = \operatorname{span}(W_j)$ subspaces of $\mathbb{C}^N$ and $\mathbb{C}^M$;
- Basis V_k, $k \ll N$, and W_j, $j \ll M$, are *orthonormal* and such that $\mathcal{V}$ and $\mathcal{W}$ are not orthogonal to $\operatorname{range}(C_1)$ and $\operatorname{range}(C_2)$ respectively;
- Build an approximation $\tilde{X} = V_k Y W_j^H \approx X$ with residual $R = C_1 C_2^H - A\tilde{X} - \tilde{X}B$.

Projection methods for low-rank right-hand sides

Let us **assume** that $\operatorname{rank}(C) = p \ll \min\{n, m\}$.

- Decompose $C = C_1 C_2^H$;
- Select $\mathcal{V} = \operatorname{span}(V_k)$ and $\mathcal{W} = \operatorname{span}(W_j)$ subspaces of $\mathbb{C}^N$ and $\mathbb{C}^M$;
- Basis V_k, $k \ll N$, and W_j, $j \ll M$, are *orthonormal* and such that $\mathcal{V}$ and $\mathcal{W}$ are not orthogonal to $\operatorname{range}(C_1)$ and $\operatorname{range}(C_2)$ respectively;
- Build an approximation $\tilde{X} = V_k Y W_j^H \approx X$ with residual $R = C_1 C_2^H - A\tilde{X} - \tilde{X}B$.

Galerkin (orthogonality) condition

Call $\tilde{\mathbf{x}} = \operatorname{vec}(\tilde{X}) = (W_j \otimes V_k)\operatorname{vec}(Y)$, then we want V_k and W_k to be selected as

$$(W_j \otimes V_k)^H(\mathbf{c} - \mathcal{A}\mathbf{x}) = 0 \Leftrightarrow V_k^H R W_j = 0 \text{ with } \mathcal{A} = B^T \otimes I + I \otimes A,\ \mathbf{c} = \operatorname{vec}(C).$$

Projection methods for low-rank right-hand sides

Let us **assume** that $\operatorname{rank}(C) = p \ll \min\{n, m\}$.

- Decompose $C = C_1 C_2^H$;
- Select $\mathcal{V} = \operatorname{span}(V_k)$ and $\mathcal{W} = \operatorname{span}(W_j)$ subspaces of $\mathbb{C}^N$ and $\mathbb{C}^M$;
- Basis V_k, $k \ll N$, and W_j, $j \ll M$, are *orthonormal* and such that $\mathcal{V}$ and $\mathcal{W}$ are not orthogonal to $\operatorname{range}(C_1)$ and $\operatorname{range}(C_2)$ respectively;
- Build an approximation $\tilde{X} = V_k Y W_j^H \approx X$ with residual $R = C_1 C_2^H - A\tilde{X} - \tilde{X}B$.

Galerkin (orthogonality) condition

Call $\tilde{\mathbf{x}} = \operatorname{vec}(\tilde{X}) = (W_j \otimes V_k)\operatorname{vec}(Y)$, then we want V_k and W_k to be selected as

$$(W_j \otimes V_k)^H(\mathbf{c} - \mathcal{A}\mathbf{x}) = 0 \Leftrightarrow V_k^H R W_j = 0 \text{ with } \mathcal{A} = B^T \otimes I + I \otimes A,\ \mathbf{c} = \operatorname{vec}(C).$$

- To compute Y, solve the **small Sylvester equation**:

$$V_k^H A V_k Y + Y W_j^H B W_j = V_k^H C_1 (W_j^H C_2)^H.$$

Projection methods for low-rank right-hand sides

Let us **assume** that $\operatorname{rank}(C) = p \ll \min\{n, m\}$.

- Decompose $C = C_1 C_2^H$;
- Select $\mathcal{V} = \operatorname{span}(V_k)$ and $\mathcal{W} = \operatorname{span}(W_j)$ subspaces of $\mathbb{C}^N$ and $\mathbb{C}^M$;
- Basis V_k, $k \ll N$, and W_j, $j \ll M$, are *orthonormal* and such that $\mathcal{V}$ and $\mathcal{W}$ are not orthogonal to $\operatorname{range}(C_1)$ and $\operatorname{range}(C_2)$ respectively;
- Build an approximation $\tilde{X} = V_k Y W_j^H \approx X$ with residual $R = C_1 C_2^H - A\tilde{X} - \tilde{X}B$.

Galerkin (orthogonality) condition

Call $\tilde{\mathbf{x}} = \operatorname{vec}(\tilde{X}) = (W_j \otimes V_k)\operatorname{vec}(Y)$, then we want V_k and W_k to be selected as

$$(W_j \otimes V_k)^H(\mathbf{c} - \mathcal{A}\mathbf{x}) = 0 \Leftrightarrow V_k^H R W_j = 0 \text{ with } \mathcal{A} = B^T \otimes I + I \otimes A,\ \mathbf{c} = \operatorname{vec}(C).$$

- To compute Y, solve the **small Sylvester equation**:

$$V_k^H A V_k Y + Y W_j^H B W_j = V_k^H C_1 (W_j^H C_2)^H \Rightarrow \text{Bartels and Stewart.}$$

Projection methods for low-rank right-hand sides

Existence of the solution

If $V_k^H A V_k$ and $-W_j^H B W_j$ have **disjoint spectra** we can solve

$$V_k^H A V_k Y + Y W_j^H B W_j = V_k^H C_1 (W_j^H C_2)^H \qquad \forall\, C = C_1 C_2^H.$$

To enforce it, is sufficient to have A and $-B$ with disjoint field of values.

Projection methods for low-rank right-hand sides

The cost of **one iteration** for $m > n$ and $p = \mathrm{rank}(C)$ is then given by

Input: A, B, C_1 and C_2
Orthogonalize columns of C_1 to get $\mathbf{v}_1 = V_1$;
Orthogonalize columns of C_2 to get $\mathbf{v}_2 = W_1$;
for $k = 1, 2, \ldots,$ **do**
  Compute Y_k solution to
  $V_k^H A V_k Y + Y W_k^H B W_k - V_k^H C_1 (W_k^H C_2)^H = 0$;
  if *converged* **then**
    Return V_k, Y_k and W_k such that
    $X_k = V_k Y_k W_k^*$ and **stop**.
  end
  `/* Compute next bases blocks */`
  Compute $\tilde{v}$ and $\hat{w}$ from the **approximate space**;
  Make $\hat{v}$ orthogonal w.r.t. $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$;
  Make $\hat{w}$ orthogonal w.r.t. $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$;
  Orthogonalize col.s of $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$ for $\mathbf{v}_{k+1}$ and $\mathbf{w}_{k+1}$;
  Update: $V_{k+1} = [V_k, \mathbf{v}_{k+1}]$, $W_{k+1} = [W_k, \mathbf{w}_{k+1}]$;
end

Projection methods for low-rank right-hand sides

The cost of **one iteration** for $m > n$ and $p = \mathrm{rank}(C)$ is then given by

- $O((kp)^3)$ flops for the solution of the projected problem,

Input: A, B, C_1 and C_2
Orthogonalize columns of C_1 to get $\mathbf{v}_1 = V_1$;
Orthogonalize columns of C_2 to get $\mathbf{v}_2 = W_1$;
for $k = 1, 2, \ldots,$ **do**
    Compute Y_k solution to
    $V_k^H A V_k Y + Y W_k^H B W_k - V_k^H C_1 (W_k^H C_2)^H = 0$;
    if *converged* **then**
        Return V_k, Y_k and W_k such that
        $X_k = V_k Y_k W_k^*$ and **stop**.
    end
    `/* Compute next bases blocks */`
    Compute $\tilde{v}$ and $\hat{w}$ from the **approximate space**;
    Make $\hat{v}$ orthogonal w.r.t. $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$;
    Make $\hat{w}$ orthogonal w.r.t. $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$;
    Orthogonalize col.s of $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$ for $\mathbf{v}_{k+1}$ and $\mathbf{w}_{k+1}$;
    Update: $V_{k+1} = [V_k, \mathbf{v}_{k+1}]$, $W_{k+1} = [W_k, \mathbf{w}_{k+1}]$;
end

Projection methods for low-rank right-hand sides

The cost of **one iteration** for $m > n$ and $p = \mathrm{rank}(C)$ is then given by

- $O((kp)^3)$ flops for the solution of the projected problem,
- Orthogonalization of the new basis vectors with respect to the older vectors: $O(mkp^2)$,

Input: A, B, C_1 and C_2
Orthogonalize columns of C_1 to get $\mathbf{v}_1 = V_1$;
Orthogonalize columns of C_2 to get $\mathbf{v}_2 = W_1$;
for $k = 1, 2, \ldots,$ **do**
  Compute Y_k solution to
  $V_k^H A V_k Y + Y W_k^H B W_k - V_k^H C_1 (W_k^H C_2)^H = 0$;
  if *converged* **then**
    Return V_k, Y_k and W_k such that
    $X_k = V_k Y_k W_k^*$ and **stop**.
  end
  `/* Compute next bases blocks */`
  Compute $\tilde{v}$ and $\hat{w}$ from the **approximate space**;
  Make $\hat{v}$ orthogonal w.r.t. $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$;
  Make $\hat{w}$ orthogonal w.r.t. $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$;
  Orthogonalize col.s of $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$ for $\mathbf{v}_{k+1}$ and $\mathbf{w}_{k+1}$;
  Update: $V_{k+1} = [V_k, \mathbf{v}_{k+1}]$, $W_{k+1} = [W_k, \mathbf{w}_{k+1}]$;
end

Projection methods for low-rank right-hand sides

The cost of **one iteration** for $m > n$ and $p = \mathrm{rank}(C)$ is then given by

- $O((kp)^3)$ flops for the solution of the projected problem,
- Orthogonalization of the new basis vectors with respect to the older vectors: $O(mkp^2)$,
- Orthogonalization of the new block: $O(mp^2)$.

Input: A, B, C_1 and C_2
Orthogonalize columns of C_1 to get $\mathbf{v}_1 = V_1$;
Orthogonalize columns of C_2 to get $\mathbf{v}_2 = W_1$;
for $k = 1, 2, \ldots,$ **do**
  Compute Y_k solution to
  $V_k^H A V_k Y + Y W_k^H B W_k - V_k^H C_1 (W_k^H C_2)^H = 0$;
  if *converged* **then**
    Return V_k, Y_k and W_k such that
    $X_k = V_k Y_k W_k^*$ and **stop**.
  end
  `/* Compute next bases blocks */`
  Compute $\tilde{v}$ and $\hat{w}$ from the **approximate space**;
  Make $\hat{v}$ orthogonal w.r.t. $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$;
  Make $\hat{w}$ orthogonal w.r.t. $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$;
  Orthogonalize col.s of $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$ for $\mathbf{v}_{k+1}$ and $\mathbf{w}_{k+1}$;
  Update: $V_{k+1} = [V_k, \mathbf{v}_{k+1}]$, $W_{k+1} = [W_k, \mathbf{w}_{k+1}]$;
end

Projection methods for low-rank right-hand sides

The cost of **one iteration** for $m > n$ and $p = \mathrm{rank}(C)$ is then given by

- $O((kp)^3)$ flops for the solution of the projected problem,
- Orthogonalization of the new basis vectors with respect to the older vectors: $O(mkp^2)$,
- Orthogonalization of the new block: $O(mp^2)$.

Loss of rank

If the generated basis experiences loss of rank, deflation procedures can be applied to remove redundant columns.

Input: A, B, C_1 and C_2
Orthogonalize columns of C_1 to get $\mathbf{v}_1 = V_1$;
Orthogonalize columns of C_2 to get $\mathbf{v}_2 = W_1$;
for $k = 1, 2, \ldots,$ **do**
  Compute Y_k solution to
  $V_k^H A V_k Y + Y W_k^H B W_k - V_k^H C_1 (W_k^H C_2)^H = 0$;
  if *converged* **then**
    Return V_k, Y_k and W_k such that
    $X_k = V_k Y_k W_k^*$ and **stop**.
  end
  `/* Compute next bases blocks */`
  Compute $\tilde{v}$ and $\hat{w}$ from the **approximate space**;
  Make $\hat{v}$ orthogonal w.r.t. $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$;
  Make $\hat{w}$ orthogonal w.r.t. $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$;
  Orthogonalize col.s of $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$ for $\mathbf{v}_{k+1}$ and $\mathbf{w}_{k+1}$;
  Update: $V_{k+1} = [V_k, \mathbf{v}_{k+1}]$, $W_{k+1} = [W_k, \mathbf{w}_{k+1}]$;
end

Projection methods for low-rank right-hand sides

The cost of **one iteration** for $m > n$ and $p = \mathrm{rank}(C)$ is then given by

- $O((kp)^3)$ flops for the solution of the projected problem,
- Orthogonalization of the new basis vectors with respect to the older vectors: $O(mkp^2)$,
- Orthogonalization of the new block: $O(mp^2)$.

Loss of rank

If the generated basis experiences loss of rank, deflation procedures can be applied to remove redundant columns.

Input: A, B, C_1 and C_2
Orthogonalize columns of C_1 to get $\mathbf{v}_1 = V_1$;
Orthogonalize columns of C_2 to get $\mathbf{v}_2 = W_1$;
for $k = 1, 2, \ldots,$ **do**
  Compute Y_k solution to
  $V_k^H A V_k Y + Y W_k^H B W_k - V_k^H C_1 (W_k^H C_2)^H = 0$;
  if *converged* **then**
    Return V_k, Y_k and W_k such that
    $X_k = V_k Y_k W_k^*$ and **stop**.
  end
  `/* Compute next bases blocks */`
  Compute $\tilde{v}$ and $\hat{w}$ from the **approximate space**;
  Make $\hat{v}$ orthogonal w.r.t. $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$;
  Make $\hat{w}$ orthogonal w.r.t. $\{\mathbf{w}_1, \ldots, \mathbf{w}_k\}$;
  Orthogonalize col.s of $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$ for $\mathbf{v}_{k+1}$ and $\mathbf{w}_{k+1}$;
  Update: $V_{k+1} = [V_k, \mathbf{v}_{k+1}]$, $W_{k+1} = [W_k, \mathbf{w}_{k+1}]$;
end

Selection of $\mathcal{V}$ and $\mathcal{W}$

How do we select the approximation spaces $\mathcal{V}$ and $\mathcal{W}$?

Standard **block** Krylov subspace

$$\mathcal{V} = \mathrm{range}\{[C_1, AC_1, A^2 C_2, \dots]\}, \quad \mathcal{W} = \mathrm{range}\{[C_2, B^H C_1, (B^H)^2 C_2, \dots]\},$$

Rational **block** Krylov subspace

$$\begin{aligned} \mathcal{V} &= \mathrm{range}\{[(A + \sigma_1 I)^{-1} C_1, (A + \sigma_2 I)^{-1}(A + \sigma_1 I)^{-1} C_1, \dots]\}, \\ \mathcal{W} &= \mathrm{range}\{[(B^H + \eta_1 I)^{-1} C_2, (B^H + \eta_2 I)^{-1}(B^H + \eta_1 I)^{-1} C_2, \dots]\}, \end{aligned}$$

Global Krylov subspace:

$$\mathcal{V} = \left\{ \sum_{i \geq 0} A^i C_i \gamma_i, \quad \gamma_i \in \mathbb{R} \right\} = \mathrm{span}\{C_1, AC_1, A^2 C_2, \dots\}$$

where the linear combination is performed blockwise, and analogously for $\mathcal{W}$.

Stopping criterions

To change the "**if** *converged*" in the algorithm we have to monitor the residual, e.g.,

$$\|R\|_2 = \|A\tilde{X} + \tilde{X}B - C_1C_2^*\|_2 \text{ or } \|R\|_F = \|A\tilde{X} + \tilde{X}B - C_1C_2^*\|_F.$$

Stopping criterions

To change the "**if** *converged*" in the algorithm we have to monitor the residual, e.g.,

$$\|R\|_2 = \|A\tilde{X} + \tilde{X}B - C_1C_2^*\|_2 \text{ or } \|R\|_F = \|A\tilde{X} + \tilde{X}B - C_1C_2^*\|_F.$$

⚠ R is **dense** and **large**: we should avoid assembling it!

Stopping criterions

To change the "**if** *converged*" in the algorithm we have to monitor the residual, e.g.,

$$\|R\|_2 = \|A\tilde{X} + \tilde{X}B - C_1C_2^*\|_2 \text{ or } \|R\|_F = \|A\tilde{X} + \tilde{X}B - C_1C_2^*\|_F.$$

- R is **dense** and **large**: we should avoid assembling it!
- If we are using Krylov subspaces, we can employ Arnoldi-like relations to this end:

$$AV_k = [V_k, \hat{v}_k]\underline{H}_k \text{ and } B^H W_j = [W_j, \hat{w}_j]\underline{K}_j,$$

  with $[V_k, \hat{v}_k]$ and $[W_j, \hat{w}_j]$ having orthonormal columns.
- If $\exists\, C_1^{(k)}$ and $C_2^{(j)}$ s.t. $C_1 = [V_k, \hat{v}_k]C_1^{(k)}$ and $C_2 = [W_j, \hat{w}_j]C_2^{(j)}$

$$\begin{aligned}
\|R\|_F &= \|AV_kYW_j^H + V_kYW_j^HB - \hat{V}_kC_1^{(k)}(\hat{W}_jC_2^{(j)})^H\|_F \\
&= \left\|[V_k\hat{v}_k]\left(\underline{H}_kY[I,0] + [I;0]Y\underline{K}_j^H - C_1^{(k)}(C_2^{(j)})^H\right)[W_j,\hat{w}_j]^H\right\|_F \\
&= \|\underline{H}_kY[I,0] + [I;0]Y\underline{K}_j^H - C_1^{(k)}(C_2^{(j)})^H\|_F.
\end{aligned}$$

Stopping criterions

To change the "**if** *converged*" in the algorithm we have to monitor the residual, e.g.,

$$\|R\|_2 = \|A\tilde{X} + \tilde{X}B - C_1C_2^*\|_2 \text{ or } \|R\|_F = \|A\tilde{X} + \tilde{X}B - C_1C_2^*\|_F.$$

- R is **dense** and **large**: we should avoid assembling it!
- If we are using Krylov subspaces, we can employ Arnoldi-like relations to this end:

$$AV_k = [V_k, \hat{v}_k]\underline{H}_k \text{ and } B^H W_j = [W_j, \hat{w}_j]\underline{K}_j,$$

  with $[V_k, \hat{v}_k]$ and $[W_j, \hat{w}_j]$ having orthonormal columns.
- If $\exists\, C_1^{(k)}$ and $C_2^{(j)}$ s.t. $C_1 = [V_k, \hat{v}_k]C_1^{(k)}$ and $C_2 = [W_j, \hat{w}_j]C_2^{(j)}$

$$\|R\|_F = \|\underline{H}_k Y[I, 0] + [I; 0]Y\underline{K}_j^H - C_1^{(k)}(C_2^{(j)})^H\|_F.$$

- The matrix in the last norm is small **if** k and j are small, if we are under the conditions on the spaces we can **monitor the residual along the way**.

Variants: as many as for standard Krylov methods

- We can use different approximation space dimensions for A and B.

Variants: as many as for standard Krylov methods

- We can use different approximation space dimensions for A and B.
- We can use different approximation spaces of the same dimension.

Variants: as many as for standard Krylov methods

- We can use different approximation space dimensions for A and B.
- We can use different approximation spaces of the same dimension.
- We could use nonsymmetric Lanczos (*oblique* subspaces) if $B = A^H$ and $C_1 C_2^*$ is nonsymmetric to build simultaneously $K_j(A, C_1)$ and $K_j(A, C_2)$.

Variants: as many as for standard Krylov methods

- We can use different approximation space dimensions for A and B.
- We can use different approximation spaces of the same dimension.
- We could use nonsymmetric Lanczos (*oblique* subspaces) if $B = A^H$ and $C_1 C_2^*$ is nonsymmetric to build simultaneously $K_j(A, C_1)$ and $K_j(A, C_2)$.
- We could use Krylov methods with restart to save memory.

Variants: as many as for standard Krylov methods

- We can use different approximation space dimensions for A and B.
- We can use different approximation spaces of the same dimension.
- We could use nonsymmetric Lanczos (*oblique* subspaces) if $B = A^H$ and $C_1 C_2^*$ is nonsymmetric to build simultaneously $K_j(A, C_1)$ and $K_j(A, C_2)$.
- We could use Krylov methods with restart to save memory.
- If A and B are symmetric (and not necessarily equal), we could use short-term block recurrences.

Variants: as many as for standard Krylov methods

- We can use different approximation space dimensions for A and B.
- We can use different approximation spaces of the same dimension.
- We could use nonsymmetric Lanczos (*oblique* subspaces) if $B = A^H$ and $C_1 C_2^*$ is nonsymmetric to build simultaneously $\mathcal{K}_j(A, C_1)$ and $\mathcal{K}_j(A, C_2)$.
- We could use Krylov methods with restart to save memory.
- If A and B are symmetric (and not necessarily equal), we could use short-term block recurrences.

The review by Simoncini 2016 has pointers to all the different strategies available.

Variants: as many as for standard Krylov methods

- We can use different approximation space dimensions for A and B.
- We can use different approximation spaces of the same dimension.
- We could use nonsymmetric Lanczos (*oblique* subspaces) if $B = A^H$ and $C_1 C_2^*$ is nonsymmetric to build simultaneously $\mathcal{K}_j(A, C_1)$ and $\mathcal{K}_j(A, C_2)$.
- We could use Krylov methods with restart to save memory.
- If A and B are symmetric (and not necessarily equal), we could use short-term block recurrences.

The review by Simoncini 2016 has pointers to all the different strategies available.

Where were we?

For the two equations we wanted to solve we have then the following questions:

- Is our C low-rank?
- What type of Krylov subspace should we select?
- Does any of this stuff converge at all?

Low-rank, regularity and separability

For the 1D+1D case we have to solve

$$A_N W + W B_M^T = F, \text{ with } F = [\mathbf{W}^{(0)} + \Delta t \mathbf{f}^{(1)} | \cdots | \Delta t \mathbf{f}^{M}]_{N \times M},$$

with $(\mathbf{f}^{(m)})_i = f(x_i, t_m)$.

Low-rank, regularity and separability

For the 1D+1D case we have to solve

$$A_N W + W B_M^T = F, \text{ with } F = [\mathbf{W}^{(0)} + \Delta t \mathbf{f}^{(1)} | \cdots | \Delta t \mathbf{f}^M]_{N \times M},$$

with $(\mathbf{f}^{(m)})_i = f(x_i, t_m)$.

For the 1D+2D case we have to solve for $m = 0, \ldots, M-1$

$$\left(\frac{1}{2} I_{N_x} - \Delta t \tilde{G}_{N_x}\right) \tilde{W}^{(m+1)} + \tilde{W}^{(m+1)} \left(\frac{1}{2} I^{N_y} - \Delta t \tilde{G}_{N_y}\right)^T = \tilde{W}^{(m)} + \Delta t F^{(m+1)},$$

with $(F^{(m+1)})_{i,j} = f(x_i, y_j, t_{m+1})$.

Low-rank, regularity and separability

For the 1D+1D case we have to solve

$$A_N W + W B_M^T = F, \text{ with } F = [\mathbf{W}^{(0)} + \Delta t \mathbf{f}^{(1)} | \cdots | \Delta t \mathbf{f}^{M}]_{N \times M},$$

with $(\mathbf{f}^{(m)})_i = f(x_i, t_m)$.

For the 1D+2D case we have to solve for $m = 0, \ldots, M-1$

$$\left(\frac{1}{2} I_{N_x} - \Delta t \tilde{G}_{N_x} \right) \tilde{W}^{(m+1)} + \tilde{W}^{(m+1)} \left(\frac{1}{2} I^{N_y} - \Delta t \tilde{G}_{N_y} \right)^T = \tilde{W}^{(m)} + \Delta t F^{(m+1)},$$

with $(F^{(m+1)})_{i,j} = f(x_i, y_j, t_{m+1})$.

Low-Rank

When is it that these matrices have a fixed, size-independent "small" rank?

Low-rank, regularity and separability

- If a function $f(x, y) = f_1(x) f_2(y)$ then

$$\begin{bmatrix} f(x_1, y_1) & f(x_1, y_2) & \cdots & f(x_1, y_n) \\ f(x_2, y_1) & f(x_2, y_2) & \cdots & f(x_2, y_n) \\ \vdots & \vdots & \ddots & \vdots \\ f(x_n, y_1) & f(x_n, y_2) & \cdots & f(x_n, y_n) \end{bmatrix}$$

Low-rank, regularity and separability

If a function $f(x, y) = f_1(x)f_2(y)$ then

$$\begin{bmatrix} f_1(x_1)f_2(y_1) & f_1(x_1)f_2(y_2) & \cdots & f_1(x_1)f_2(y_n) \\ f_1(x_2)f_2(y_1) & f_1(x_2)f_2(y_2) & \cdots & f_1(x_2)f_2(y_n) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_n)f_2(y_1) & f_1(x_n)f_2(y_2) & \cdots & f_1(x_n)f_2(y_n) \end{bmatrix}$$

Low-rank, regularity and separability

If a function $f(x, y) = f_1(x)f_2(y)$ then

$$\begin{bmatrix} f_1(x_1)f_2(y_1) & f_1(x_1)f_2(y_2) & \cdots & f_1(x_1)f_2(y_n) \\ f_1(x_2)f_2(y_1) & f_1(x_2)f_2(y_2) & \cdots & f_1(x_2)f_2(y_n) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_n)f_2(y_1) & f_1(x_n)f_2(y_2) & \cdots & f_1(x_n)f_2(y_n) \end{bmatrix} = \begin{bmatrix} f_1(x_1) \\ f_1(x_2) \\ \vdots \\ f_1(x_n) \end{bmatrix} \begin{bmatrix} f_2(y_1) & f_2(y_2) & \cdots & f_n(y_n) \end{bmatrix}$$

Low-rank, regularity and separability

If a function $f(x,y) = f_1(x)f_2(y)$ then

$$\begin{bmatrix} f_1(x_1)f_2(y_1) & f_1(x_1)f_2(y_2) & \cdots & f_1(x_1)f_2(y_n) \\ f_1(x_2)f_2(y_1) & f_1(x_2)f_2(y_2) & \cdots & f_1(x_2)f_2(y_n) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_n)f_2(y_1) & f_1(x_n)f_2(y_2) & \cdots & f_1(x_n)f_2(y_n) \end{bmatrix} = \begin{bmatrix} f_1(x_1) \\ f_1(x_2) \\ \vdots \\ f_1(x_n) \end{bmatrix} \begin{bmatrix} f_2(y_1) & f_2(y_2) & \cdots & f_n(y_n) \end{bmatrix}$$

To have a simple example:

```
n = 10;
f1 = @(x) exp(-2*x); f2 = @(y) sin(2*pi*y); f = @(x,y) f1(x).*f2(y);
x = linspace(0,1,n); y = linspace(0,1,n);
[X,Y] = meshgrid(x,y);
A = f(X.',Y.'); a1 = f1(x); a2 = f2(y);
norm(A-a1.'*a2)
```

that answers us `>> ans = 0`.

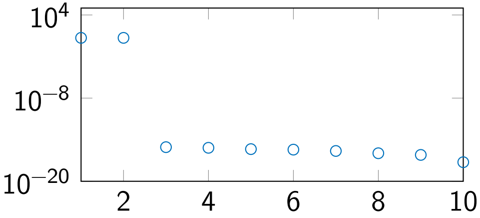
Low-rank, regularity and separability

If a function $f(x,y) = f_1(x) f_2(y)$ then

$$\begin{bmatrix} f_1(x_1)f_2(y_1) & f_1(x_1)f_2(y_2) & \cdots & f_1(x_1)f_2(y_n) \\ f_1(x_2)f_2(y_1) & f_1(x_2)f_2(y_2) & \cdots & f_1(x_2)f_2(y_n) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_n)f_2(y_1) & f_1(x_n)f_2(y_2) & \cdots & f_1(x_n)f_2(y_n) \end{bmatrix} = \begin{bmatrix} f_1(x_1) \\ f_1(x_2) \\ \vdots \\ f_1(x_n) \end{bmatrix} \begin{bmatrix} f_2(y_1) & f_2(y_2) & \cdots & f_n(y_n) \end{bmatrix}$$

What happens if $f(x,y)$ is not separable? E.g., if $f(x,y) = \sin(\pi(x+y))$?

```
n = 10;
f = @(x,y) sin(pi*(x+y));
x = linspace(0,1,n);
y = linspace(0,1,n);
[X,Y] = meshgrid(x,y); A = f(X.',Y.');
sv = svd(A);
```



Low-rank, regularity and separability

If a function $f(x,y) = f_1(x)f_2(y)$ then

$$\begin{bmatrix} f_1(x_1)f_2(y_1) & f_1(x_1)f_2(y_2) & \cdots & f_1(x_1)f_2(y_n) \\ f_1(x_2)f_2(y_1) & f_1(x_2)f_2(y_2) & \cdots & f_1(x_2)f_2(y_n) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_n)f_2(y_1) & f_1(x_n)f_2(y_2) & \cdots & f_1(x_n)f_2(y_n) \end{bmatrix} = \begin{bmatrix} f_1(x_1) \\ f_1(x_2) \\ \vdots \\ f_1(x_n) \end{bmatrix} \begin{bmatrix} f_2(y_1) & f_2(y_2) & \cdots & f_n(y_n) \end{bmatrix}$$

What happens if $f(x,y)$ is not separable? E.g., if $f(x,y) = \sin(\pi(x+y))$?

$$\sin(\pi(x+y)) = \sin(\pi x)\cos(\pi y) + \cos(\pi x)\sin(\pi y)$$

is the **sum of** two **separable functions**, i.e., we get a matrix that has rank equal to 2.

Low-rank, regularity and separability

If a function $f(x,y) = f_1(x)f_2(y)$ then

$$\begin{bmatrix} f_1(x_1)f_2(y_1) & f_1(x_1)f_2(y_2) & \cdots & f_1(x_1)f_2(y_n) \\ f_1(x_2)f_2(y_1) & f_1(x_2)f_2(y_2) & \cdots & f_1(x_2)f_2(y_n) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_n)f_2(y_1) & f_1(x_n)f_2(y_2) & \cdots & f_1(x_n)f_2(y_n) \end{bmatrix} = \begin{bmatrix} f_1(x_1) \\ f_1(x_2) \\ \vdots \\ f_1(x_n) \end{bmatrix} \begin{bmatrix} f_2(y_1) & f_2(y_2) & \cdots & f_n(y_n) \end{bmatrix}$$

What happens if $f(x,y)$ is not separable? E.g., if $f(x,y) = \sin(\pi(x+y))$?

$$\sin(\pi(x+y)) = \sin(\pi x)\cos(\pi y) + \cos(\pi x)\sin(\pi y)$$

is the **sum of** two **separable functions**, i.e., we get a matrix that has rank equal to 2.

We can try to **generalize** this **decomposition idea** to more general functions!

Low-rank, regularity and separability

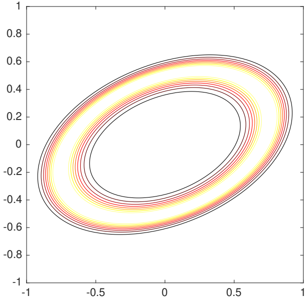
We can approximate a function of two variables as the sum of separable functions

$$f(x,y) = \sum_{k=1}^{K} f_k T_k(x) T_k(y), \quad \{T_k(\cdot)\}_k \text{ Čebyšëv polynomials.}$$

Example (Using Chebfun (Driscoll, Hale, and Trefethen 2014))

Consider $f(x,y) = \exp(-40(x^2 - xy + 2y^2 - 1/2)^2)$.

```
cheb.xy
ff=@(x,y)exp(-40*(x.^2-x.*y+2*y.^2-1/2).^2);
f=chebfun2(ff);
levels = 0.1:0.1:0.9;
contour(f,levels);
axis([-1 1 -1 1]);
axis square
```



Low-rank, regularity and separability

We can approximate a function of two variables as the sum of separable functions

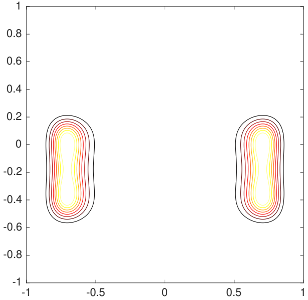
$$f(x,y) = \sum_{k=1}^{K} f_k T_k(x) T_k(y), \quad \{T_k(\cdot)\}_k \text{ Čebyšëv polynomials.}$$

Example (Using Chebfun (Driscoll, Hale, and Trefethen 2014))

Consider $f(x,y) = \exp(-40(x^2 - xy + 2y^2 - 1/2)^2)$.

Forcing a rank K approximation

```
levels = 0.1:0.1:0.9;
for K = 1:9
 contour(chebfun2(ff,K),levels)
 xlim([-1 1]), axis equal
end
```



Low-rank, regularity and separability

We can approximate a function of two variables as the sum of separable functions

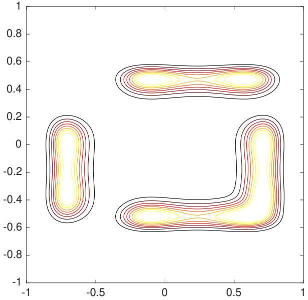
$$f(x,y) = \sum_{k=1}^{K} f_k T_k(x) T_k(y), \quad \{T_k(\cdot)\}_k \text{ Čebyšëv polynomials.}$$

Example (Using `Chebfun` (Driscoll, Hale, and Trefethen 2014))

Consider $f(x,y) = \exp(-40(x^2 - xy + 2y^2 - 1/2)^2)$.

Forcing a rank K approximation

```
levels = 0.1:0.1:0.9;
for K = 1:9
 contour(chebfun2(ff,K),levels)
 xlim([-1 1]), axis equal
end
```



Low-rank, regularity and separability

We can approximate a function of two variables as the sum of separable functions

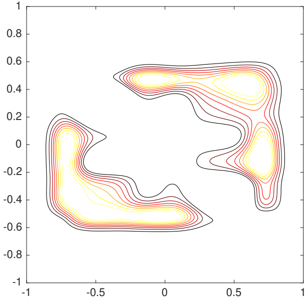
$$f(x, y) = \sum_{k=1}^{K} f_k T_k(x) T_k(y), \quad \{T_k(\cdot)\}_k \text{ Čebyšëv polynomials.}$$

Example (Using Chebfun (Driscoll, Hale, and Trefethen 2014))

Consider $f(x, y) = \exp(-40(x^2 - xy + 2y^2 - 1/2)^2)$.

Forcing a rank K approximation

```
levels = 0.1:0.1:0.9;
for K = 1:9
 contour(chebfun2(ff,K),levels)
 xlim([-1 1]), axis equal
end
```



Low-rank, regularity and separability

We can approximate a function of two variables as the sum of separable functions

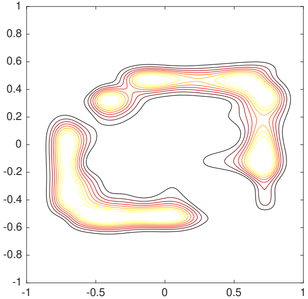
$$f(x,y) = \sum_{k=1}^{K} f_k T_k(x) T_k(y), \quad \{T_k(\cdot)\}_k \text{ Čebyšëv polynomials.}$$

Example (Using `Chebfun` (Driscoll, Hale, and Trefethen 2014))

Consider $f(x,y) = \exp(-40(x^2 - xy + 2y^2 - 1/2)^2)$.

Forcing a rank K approximation

```
levels = 0.1:0.1:0.9;
for K = 1:9
 contour(chebfun2(ff,K),levels)
 xlim([-1 1]), axis equal
end
```



Low-rank, regularity and separability

We can approximate a function of two variables as the sum of separable functions

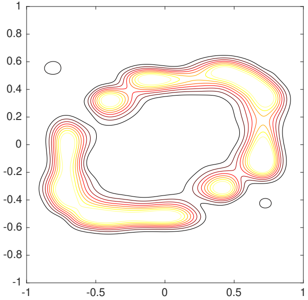
$$f(x,y) = \sum_{k=1}^{K} f_k T_k(x) T_k(y), \quad \{T_k(\cdot)\}_k \text{ Čebyšëv polynomials.}$$

Example (Using Chebfun (Driscoll, Hale, and Trefethen 2014))

Consider $f(x,y) = \exp(-40(x^2 - xy + 2y^2 - 1/2)^2)$.

Forcing a rank K approximation

```
levels = 0.1:0.1:0.9;
for K = 1:9
 contour(chebfun2(ff,K),levels)
 xlim([-1 1]), axis equal
end
```



Low-rank, regularity and separability

We can approximate a function of two variables as the sum of separable functions

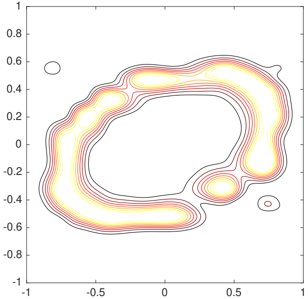
$$f(x,y) = \sum_{k=1}^{K} f_k T_k(x) T_k(y), \quad \{T_k(\cdot)\}_k \text{ Čebyšëv polynomials.}$$

Example (Using Chebfun (Driscoll, Hale, and Trefethen 2014))

Consider $f(x,y) = \exp(-40(x^2 - xy + 2y^2 - 1/2)^2)$.

Forcing a rank K approximation

```
levels = 0.1:0.1:0.9;
for K = 1:9
 contour(chebfun2(ff,K),levels)
 xlim([-1 1]), axis equal
end
```



Low-rank, regularity and separability

We can approximate a function of two variables as the sum of separable functions

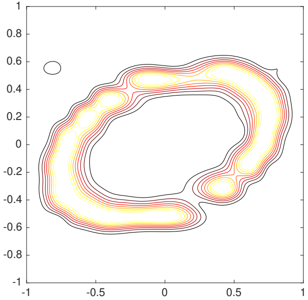
$$f(x,y) = \sum_{k=1}^{K} f_k T_k(x) T_k(y), \quad \{T_k(\cdot)\}_k \text{ Čebyšëv polynomials.}$$

Example (Using `Chebfun` (Driscoll, Hale, and Trefethen 2014))

Consider $f(x,y) = \exp(-40(x^2 - xy + 2y^2 - 1/2)^2)$.

Forcing a rank K approximation

```
levels = 0.1:0.1:0.9;
for K = 1:9
 contour(chebfun2(ff,K),levels)
 xlim([-1 1]), axis equal
end
```



Low-rank, regularity and separability

We can approximate a function of two variables as the sum of separable functions

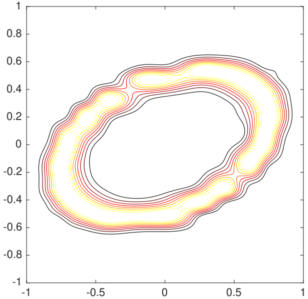
$$f(x, y) = \sum_{k=1}^{K} f_k T_k(x) T_k(y), \quad \{T_k(\cdot)\}_k \text{ Čebyšëv polynomials.}$$

Example (Using `Chebfun` (Driscoll, Hale, and Trefethen 2014))

Consider $f(x, y) = \exp(-40(x^2 - xy + 2y^2 - 1/2)^2)$.

Forcing a rank K approximation

```
levels = 0.1:0.1:0.9;
for K = 1:9
 contour(chebfun2(ff,K),levels)
 xlim([-1 1]), axis equal
end
```



Low-rank, regularity and separability

We can approximate a function of two variables as the sum of separable functions

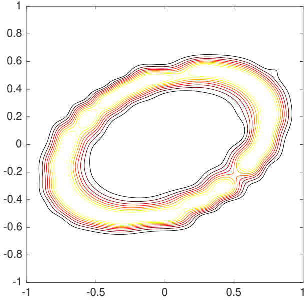
$$f(x,y) = \sum_{k=1}^{K} f_k T_k(x) T_k(y), \quad \{T_k(\cdot)\}_k \text{ Čebyšëv polynomials.}$$

Example (Using `Chebfun` (Driscoll, Hale, and Trefethen 2014))

Consider $f(x,y) = \exp(-40(x^2 - xy + 2y^2 - 1/2)^2)$.

Forcing a rank K approximation

```
levels = 0.1:0.1:0.9;
for K = 1:9
 contour(chebfun2(ff,K),levels)
 xlim([-1 1]), axis equal
end
```



Low-rank, regularity and separability

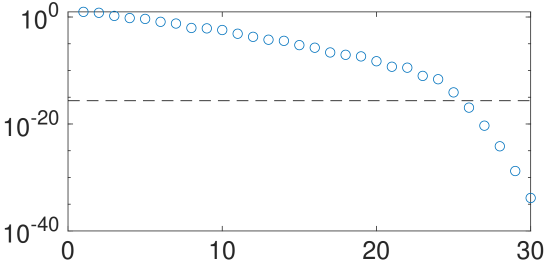
We can approximate a function of two variables as the sum of separable functions

$$f(x,y) = \sum_{k=1}^{K} f_k T_k(x) T_k(y), \quad \{T_k(\cdot)\}_k \text{ Čebyšëv polynomials.}$$

Example (Using `Chebfun` (Driscoll, Hale, and Trefethen 2014))

Consider $f(x,y) = \exp(-40(x^2 - xy + 2y^2 - 1/2)^2)$.

$F = (f(x_i, x_j))_{i,j} \qquad \mathrm{rank}(F) =$



$\approx 25.$

Low-rank, regularity and separability

- Not every right-hand side will have a small-enough rank!

Low-rank, regularity and separability

- Not every right-hand side will have a small-enough rank!
- Whenever we have **closed form expression** of the **involved functions** we can work with polynomial basis expansion to discover the rank.

Low-rank, regularity and separability

- Not every right-hand side will have a small-enough rank!
- Whenever we have **closed form expression** of the **involved functions** we can work with polynomial basis expansion to discover the rank.
- To actually compute the decomposition $C = C_1 C_2^H$ we need we can either

Low-rank, regularity and separability

- Not every right-hand side will have a small-enough rank!
- Whenever we have **closed form expression** of the **involved functions** we can work with polynomial basis expansion to discover the rank.
- To actually compute the decomposition $C = C_1 C_2^H$ we need we can either
  - assemble everything and use the **SVD**,

Low-rank, regularity and separability

- Not every right-hand side will have a small-enough rank!
- Whenever we have **closed form expression** of the **involved functions** we can work with polynomial basis expansion to discover the rank.
- To actually compute the decomposition $C = C_1 C_2^H$ we need we can either
  - assemble everything and use the **SVD**,
  - work with **polynomial expansion** and truncate it for small enough coefficients, e.g., (Beckermann and Townsend 2019; Carvajal, Chapman, and Geddes 2005; Townsend and Trefethen 2013),

Low-rank, regularity and separability

- Not every right-hand side will have a small-enough rank!
- Whenever we have **closed form expression** of the **involved functions** we can work with polynomial basis expansion to discover the rank.
- To actually compute the decomposition $C = C_1 C_2^H$ we need we can either
  - assemble everything and use the **SVD**,
  - work with **polynomial expansion** and truncate it for small enough coefficients, e.g., (Beckermann and Townsend 2019; Carvajal, Chapman, and Geddes 2005; Townsend and Trefethen 2013),
  - using algorithm that only need to compute *few* entries of A, such as Adaptive-Cross-Approximation (Tyrtyshnikov 2000), or RandSVD-type algorithms (Halko, Martinsson, and Tropp 2011).

Low-rank, regularity and separability

- Not every right-hand side will have a small-enough rank!
- Whenever we have **closed form expression** of the **involved functions** we can work with polynomial basis expansion to discover the rank.
- To actually compute the decomposition $C = C_1 C_2^H$ we need we can either
  - assemble everything and use the **SVD**,
  - work with **polynomial expansion** and truncate it for small enough coefficients, e.g., (Beckermann and Townsend 2019; Carvajal, Chapman, and Geddes 2005; Townsend and Trefethen 2013),
  - using algorithm that only need to compute *few* entries of A, such as Adaptive-Cross-Approximation (Tyrtyshnikov 2000), or RandSVD-type algorithms (Halko, Martinsson, and Tropp 2011).

Approximating approximating we could get where we wanted...

Let us remember that the approximation of the low-rank term must be done together with the approximation induced by the FDE solution method. We may not need to go as far as machine precision.

Selecting the Krylov subspace

If we are now in the case of a **low rank** right-hand side, we have to select Krylov subspaces for the spaces $\mathcal{V}$ and $\mathcal{W}$.

- From the work we have done in the last couple of lectures, we know how to solve linear systems involving discretization of 1D problems,

Selecting the Krylov subspace

If we are now in the case of a **low rank** right-hand side, we have to select Krylov subspaces for the spaces $\mathcal{V}$ and $\mathcal{W}$.

- From the work we have done in the last couple of lectures, we know how to solve linear systems involving discretization of 1D problems,
- Rational (block) Krylov subspace can therefore be a good choice!

$$\mathcal{V} = \operatorname{range}\{[(A + \sigma_1 I)^{-1} C_1, (A + \sigma_2 I)^{-1}(A + \sigma_1 I)^{-1} C_1, \ldots]\},$$
$$\mathcal{W} = \operatorname{range}\{[(B^H + \eta_1 I)^{-1} C_2, (B^H + \eta_2 I)^{-1}(B^H + \eta_1 I)^{-1} C_2, \ldots]\},$$

Selecting the Krylov subspace

If we are now in the case of a **low rank** right-hand side, we have to select Krylov subspaces for the spaces $\mathcal{V}$ and $\mathcal{W}$.

- From the work we have done in the last couple of lectures, we know how to solve linear systems involving discretization of 1D problems,
- Rational (block) Krylov subspace can therefore be a good choice!

$$\mathcal{V} = \mathrm{range}\{[(A+\sigma_1 I)^{-1}C_1, (A+\sigma_2 I)^{-1}(A+\sigma_1 I)^{-1}C_1, \ldots]\},$$
$$\mathcal{W} = \mathrm{range}\{[(B^H+\eta_1 I)^{-1}C_2, (B^H+\eta_2 I)^{-1}(B^H+\eta_1 I)^{-1}C_2, \ldots]\},$$

- ... but how do we **select the poles**?

Selecting the Krylov subspace

If we are now in the case of a **low rank** right-hand side, we have to select Krylov subspaces for the spaces $\mathcal{V}$ and $\mathcal{W}$.

- From the work we have done in the last couple of lectures, we know how to solve linear systems involving discretization of 1D problems,
- Rational (block) Krylov subspace can therefore be a good choice!

$$\mathcal{V} = \mathrm{range}\{[(A+\sigma_1 I)^{-1}C_1, (A+\sigma_2 I)^{-1}(A+\sigma_1 I)^{-1}C_1, \ldots]\},$$
$$\mathcal{W} = \mathrm{range}\{[(B^H+\eta_1 I)^{-1}C_2, (B^H+\eta_2 I)^{-1}(B^H+\eta_1 I)^{-1}C_2, \ldots]\},$$

- ... but how do we **select the poles**?
- This is not an easy problem in general! A maybe lazy (but surprisingly well behaving) choice is to set $\{\sigma_i, \eta_i\} \in \{0, \infty\} \Rightarrow$ if we choose the two values alternately, then we get the Extended Krylov Subspace.

The Extended Krylov Subspace approach

If $B = A^T$ and $C = C_1 C_2^T$ with $C_1 = C_2$, we can generate the space:

$$\mathbb{EK}(A, C_1) = \mathrm{range}([C_1, A^{-1}C_1, AC_1, A^{-2}C_1, A^2C_1, \ldots]) = \mathcal{V} = \mathcal{W}.$$

The resulting algorithm is the KPIK method by (Simoncini 2007), and can be easily extended to solve the general case, by building both

$$\mathcal{V} = \mathbb{EK}(A, C_1) = \mathrm{range}([C_1, A^{-1}C_1, AC_1, A^{-2}C_1, A^2C_1, \ldots]),$$
$$\mathcal{W} = \mathbb{EK}(B^T, C_2) = \mathrm{range}([C_2, B^{-T}C_2, B^TC_2, A^{-2T}C_2, A^{2T}C_2, \ldots]).$$

The Extended Krylov Subspace approach

If $B = A^T$ and $C = C_1 C_2^T$ with $C_1 = C_2$, we can generate the space:

$$\mathbb{EK}(A, C_1) = \mathrm{range}([C_1, A^{-1}C_1, AC_1, A^{-2}C_1, A^2C_1, \ldots]) = \mathcal{V} = \mathcal{W}.$$

The resulting algorithm is the KPIK method by (Simoncini 2007), and can be easily extended to solve the general case, by building both

$$\mathcal{V} = \mathbb{EK}(A, C_1) = \mathrm{range}([C_1, A^{-1}C_1, AC_1, A^{-2}C_1, A^2C_1, \ldots]),$$
$$\mathcal{W} = \mathbb{EK}(B^T, C_2) = \mathrm{range}([C_2, B^{-T}C_2, B^TC_2, A^{-2T}C_2, A^{2T}C_2, \ldots]).$$

For our two problems, we have to solve systems and do mat-vec with matrices

$$\text{1D: } A = \frac{-\Delta t}{h_N^\alpha}(\theta G_N + (1-\theta)G_N^T) \qquad B = T_M(1 - e^{i\theta})$$

$$\text{2D: } A = \frac{1}{2}I_{N_x} - \frac{\Delta t}{h_{N_x}^\alpha}(\theta G_{N_x} + (1-\theta)G_{N_x}^T) \qquad B = \frac{1}{2}I_{N_y} - \frac{\Delta t}{h_{N_y}^\alpha}(\theta G_{N_y} + (1-\theta)G_{N_y}^T)$$

A couple of examples - I

Let us start from the 1D+1D case

$$\begin{cases} \frac{\partial W}{\partial t} = \Gamma(3-\alpha)x^{\alpha}\,{}^{RL}D^{\alpha}_{[0,x]}W + \Gamma(3-\alpha)(2-x)^{\alpha}\,{}^{RL}D^{\alpha}_{[x,2]}W - x(x-2)e^{-t}, \\ W(0,t) = W(1,t) = 0, \quad W(x,0) = 5x(2-x); \end{cases}$$

We can **discretize it** in the usual way:

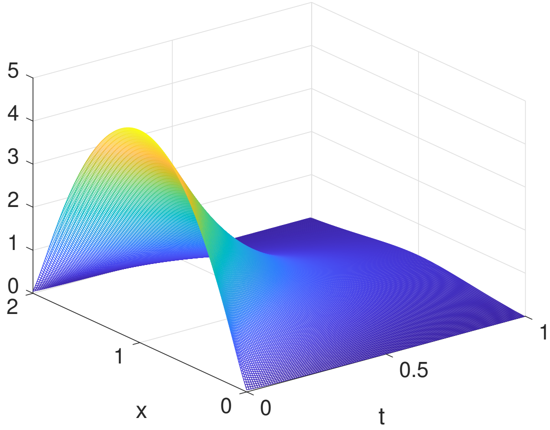
```
w0 = @(x) 5*x.*(2-x);
hN = 2/(N-1); x = 0:hN:2;
dt = hN; t = 0:dt:1; M = length(t);
dplus=@(x,t)gamma(3-alpha).*x.^alpha;
dmin=@(x,t)gamma(3-alpha).*(2-x).^alpha;
f= @(x,t) -x.*(x-2).*exp(-t);
G = glmatrix(N,alpha);
Gr = G; Grt = G.';
Dplus = diag(dplus(x,0));
Dminus = diag(dmin(x,0));
I = eye(N,N); e = ones(N,1);
```

```
A = -dt*(Dplus*Gr +
↪  Dminus*Grt)/hN^alpha;
B = spdiags([-e,e],-1:0,M,M);
[X,T] = meshgrid(x,t);
C = dt*f(X,T);
C(1,:) = w0(x) + C(1,:);
C = -C';
[U,S,V] = svd(C);
C1 = U(:,1:2)*sqrt(S(1:2,1:2));
C2 = (sqrt(S(1:2,1:2))*
↪  V(:,1:2).').';
```

A couple of examples - I

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪  B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```



$N = 2^8, \ M = 2^7, \ \alpha = 1.5$

- We are using LU-factorization and direct solutions;
- We are reassembling the solution!

A couple of examples - I

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪  B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

α	$N = 2M$	IT	Rel. Residual
1.2	2^5	7	4.982093e-10
	2^6	11	7.629176e-11
	2^7	15	3.721767e-10
	2^8	21	2.406077e-10
	2^9	28	4.726518e-10
	2^{10}	37	8.250742e-10
	2^{11}	50	5.928325e-10

- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!

A couple of examples - I

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪  B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

α	$N = 2M$	IT	Rel. Residual
1.3	2^5	8	7.473189e-41
	2^6	10	3.324155e-10
	2^7	14	1.876221e-10
	2^8	18	6.104754e-10
	2^9	24	4.098504e-10
	2^{10}	31	5.142375e-10
	2^{11}	40	6.702602e-10

- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!

A couple of examples - I

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪  B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

α	$N = 2M$	IT	Rel. Residual
1.4	2^5	7	4.900654e-10
	2^6	10	4.402728e-11
	2^7	13	1.970841e-10
	2^8	17	2.024635e-10
	2^9	22	5.120085e-10
	2^{10}	28	8.263324e-10
	2^{11}	36	8.596848e-10

- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!

A couple of examples - I

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪  B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

α	$N = 2M$	IT	Rel. Residual
1.5	2^5	7	1.235969e-10
	2^6	9	2.799035e-10
	2^7	13	1.007848e-10
	2^8	16	6.145733e-10
	2^9	21	7.639171e-10
	2^{10}	27	5.857467e-10
	2^{11}	34	8.065585e-10

- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!

A couple of examples - I

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪  B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

α	$N = 2M$	IT	Rel. Residual
1.6	2^5	7	2.480357e-11
	2^6	9	8.683894e-11
	2^7	13	7.692141e-11
	2^8	16	3.792143e-10
	2^9	21	3.991222e-10
	2^{10}	26	6.017048e-10
	2^{11}	33	6.133773e-10

- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!

A couple of examples - I

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪ B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

α	$N = 2M$	IT	Rel. Residual
1.7	2^5	7	5.588528e-12
	2^6	8	6.692127e-10
	2^7	12	8.189936e-10
	2^8	16	3.403250e-10
	2^9	20	9.093120e-10
	2^{10}	26	3.550244e-10
	2^{11}	32	7.478792e-10

- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!

A couple of examples - I

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪ B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

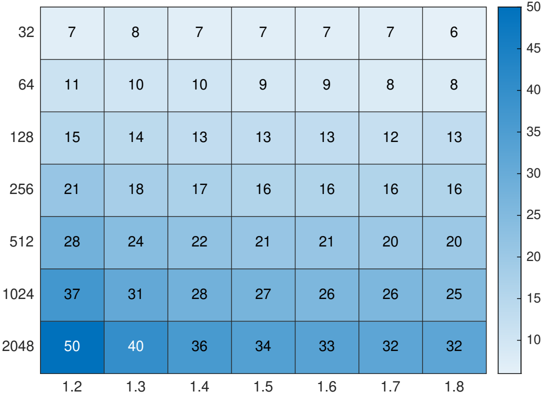
α	$N = 2M$	IT	Rel. Residual
1.8	2^5	6	6.097527e-10
	2^6	8	9.737670e-11
	2^7	13	6.202872e-11
	2^8	16	2.193864e-10
	2^9	20	7.469866e-10
	2^{10}	25	8.191797e-10
	2^{11}	32	5.086938e-10

- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!

A couple of examples - I

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪  B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```



- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!

A couple of examples - II

We can then try the 1D+2D case

$$\begin{cases} \frac{\partial W}{\partial t} = \Gamma(3-\alpha)x^{\alpha}\, {}^{RL}D^{\alpha}_{[0,x]} W + \Gamma(3-\alpha)(2-x)^{\alpha}\, {}^{RL}D^{\alpha}_{[x,2]} W \\ \qquad + \Gamma(3-\alpha)y^{\alpha}\, {}^{RL}D^{\alpha}_{[0,y]} W + \Gamma(3-\alpha)(2-y)^{\alpha}\, {}^{RL}D^{\alpha}_{[y,2]} W \\ \qquad + \sin(\pi x)\sin(\pi y)e^{-t}, \\ W(x,y,t) = 0, & (x,y) \in \partial[0,2]^2, \\ W(x,y,0) = 5x(2-x)y(2-y), \end{cases}$$

for which the discretization proceeds along the usual lines, i.e,

```
hN = 2/(N-1); x = 0:hN:2; y = 0:hN:2; [X,Y] = meshgrid(x,y);
dt = hN; t = 0:dt:1; M = length(t);
w0 = @(x,y) 5*x.*(2-x).*y.*(2-y);
dplus = @(x,t) gamma(3-alpha).*x.^alpha;
dminus = @(x,t) gamma(3-alpha).*(2-x).^alpha;
f = @(x,y,t) sin(pi*x).*sin(pi*y).*exp(-t);
```

A couple of examples - II

We can then try the 1D+2D case

$$
\begin{cases}
\frac{\partial W}{\partial t} = \Gamma(3-\alpha)x^{\alpha}\, {}^{RL}D^{\alpha}_{[0,x]} W + \Gamma(3-\alpha)(2-x)^{\alpha}\, {}^{RL}D^{\alpha}_{[x,2]} W \\
\qquad + \Gamma(3-\alpha)y^{\alpha}\, {}^{RL}D^{\alpha}_{[0,y]} W + \Gamma(3-\alpha)(2-y)^{\alpha}\, {}^{RL}D^{\alpha}_{[y,2]} W \\
\qquad + \sin(\pi x)\sin(\pi y)e^{-t}, \\
W(x,y,t) = 0, & (x,y) \in \partial[0,2]^2, \\
W(x,y,0) = 5x(2-x)y(2-y),
\end{cases}
$$

for which the discretization proceeds along the usual lines, i.e,

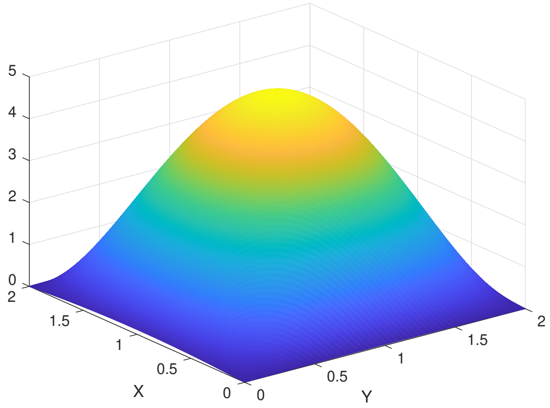
```
G = glmatrix(N,alpha); Gr = G; Grt = G.';
Dplus = diag(dplus(x,0)); Dminus = diag(dminus(x,0));
I = eye(N,N); e = ones(N,1);
A = 0.5*I -dt*(Dplus*Gr + Dminus*Grt)/hN^alpha; % Left-hand side
B = (0.5*I -dt*(Dplus*Gr + Dminus*Grt)/hN^alpha).';
C = w0(X,Y) + dt*f(X,Y,t(1)); C = -C'; [U,S,V] = svd(C);  % Right-hand side
C1 = U(:,1:2)*sqrt(S(1:2,1:2)); C2 = (sqrt(S(1:2,1:2))*V(:,1:2).').';
```

A couple of examples - II

And then use the kpik_sylv solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪  B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!



$N = 2^8,\ M = 2^8,\ \alpha = 1.5$

A couple of examples - II

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪  B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

α	$N = M$	IT	Rel. Residual
1.2	2^5	7	8.572314e-12
	2^6	9	1.035235e-10
	2^7	10	6.376925e-10
	2^8	11	4.294848e-10
	2^9	11	4.831316e-10
	2^{10}	11	3.340377e-10
	2^{11}	10	8.493637e-10

- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!

A couple of examples - II

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪ B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

α	$N = M$	IT	Rel. Residual
1.3	2^5	7	7.117681e-11
	2^6	9	7.410001e-11
	2^7	10	6.311608e-10
	2^8	11	6.629092e-10
	2^9	11	7.935697e-10
	2^{10}	11	5.256769e-10
	2^{11}	11	3.021361e-10

- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!

A couple of examples - II

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪  B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

α	$N = M$	IT	Rel. Residual
1.4	2^5	7	6.199844e-11
	2^6	9	5.440959e-11
	2^7	10	6.223106e-10
	2^8	12	2.743756e-10
	2^9	12	6.270319e-10
	2^{10}	12	4.310692e-10
	2^{11}	11	4.849822e-10

- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!

A couple of examples - II

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪ B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

α	$N = M$	IT	Rel. Residual
1.5	2^5	7	5.108938e-11
	2^6	8	7.696608e-10
	2^7	10	5.554438e-10
	2^8	12	3.501633e-10
	2^9	13	4.696907e-10
	2^{10}	13	5.839644e-10
	2^{11}	12	6.172378e-10

- We are using LU-factorization and direct solutions;
- We are reassembling the solution!

A couple of examples - II

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪  B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

α	$N = M$	IT	Rel. Residual
1.6	2^5	7	4.147318e-11
	2^6	9	1.120891e-10
	2^7	10	4.652358e-10
	2^8	12	3.624143e-10
	2^9	13	6.835564e-10
	2^{10}	14	5.920602e-10
	2^{11}	13	8.882506e-10

- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!

A couple of examples - II

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪ B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

α	$N = M$	IT	Rel. Residual
1.7	2^5	7	3.321348e-11
	2^6	9	9.437180e-11
	2^7	10	7.551800e-10
	2^8	12	3.268160e-10
	2^9	13	7.715645e-10
	2^{10}	14	8.954668e-10
	2^{11}	15	5.806398e-10

- We are using LU-factorization and direct solutions;
- We are reassembling the solution!

A couple of examples - II

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
↪  B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```

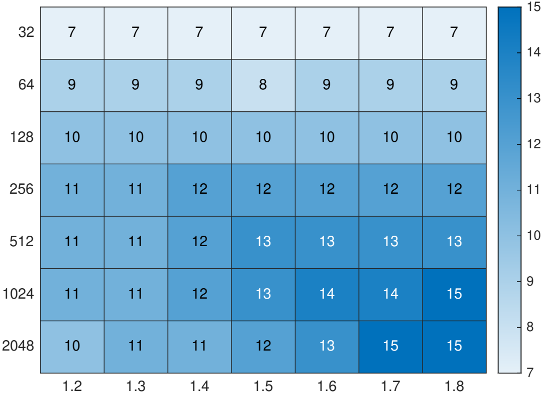
- We are using LU-factorization and direct solutions;
- We are reassembling the solution!

α	$N = M$	IT	Rel. Residual
1.8	2^5	7	2.639521e-11
	2^6	9	7.654578e-11
	2^7	10	6.909946e-10
	2^8	12	4.424195e-10
	2^9	13	7.255110e-10
	2^{10}	15	4.728355e-10
	2^{11}	15	8.400505e-10

A couple of examples - II

And then use the `kpik_sylv` solver from V. Simoncini's software:

```
m = 100;
tol = 1e-9;
[LA,UA] = lu(A); % Direct solutions!
[LB,UB] = lu(B);
[X1,X2,res]=kpik_sylv(A,LA,UA,
    B,LB,UB,C1,C2,m,tol);
SOL = X1*X2'; % Not clever al all!
```



- ⚠ We are using LU-factorization and direct solutions;
- ⚠ We are reassembling the solution!

Convergence

❓ What can we say about the convergence?

Convergence

What can we say about the convergence?

If A is symmetric and positive definite, and $B = A^T$, i.e., we are solving a Lyapunov equation, and using **polynomial Krylov subspace**:

Theorem (Simoncini and Druskin 2009, Proposition 3.1)

Let A be symmetric and positive definite, and let $\lambda_{\min}$ be the smallest eigenvalue of A. Let $\hat{\lambda}_{\min}$, $\hat{\lambda}_{\max}$ be the extreme eigenvalue of $A + \lambda_{\min} I$ and $\hat{\kappa} = \hat{\lambda}_{\max}/\hat{\lambda}_{\min}$. Then

$$\|X - X_m\| \leq 4 \frac{\sqrt{\hat{\kappa}} + 1}{\hat{\lambda}_{\min}\sqrt{\hat{\kappa}}} \left(\frac{\sqrt{\hat{\kappa}} - 1}{\sqrt{\hat{\kappa}} + 1} \right)^m .$$

Convergence

❓ What can we say about the convergence?

⚙ If A is symmetric and positive definite, and $B = A^T$, i.e., we are solving a Lyapunov equation, and using **polynomial Krylov subspace**:

Theorem (Simoncini and Druskin 2009, Proposition 3.1)

Let A be symmetric and positive definite, and let $\lambda_{\min}$ be the smallest eigenvalue of A. Let $\hat{\lambda}_{\min}, \hat{\lambda}_{\max}$ be the extreme eigenvalue of $A + \lambda_{\min} I$ and $\hat{\kappa} = \hat{\lambda}_{\max}/\hat{\lambda}_{\min}$. Then

$$\|X - X_m\| \leq 4 \frac{\sqrt{\hat{\kappa}}+1}{\hat{\lambda}_{\min}\sqrt{\hat{\kappa}}} \left(\frac{\sqrt{\hat{\kappa}}-1}{\sqrt{\hat{\kappa}}+1} \right)^m .$$

⚠ If $B = A^T$ but A is **no longer symmetric**, one then needs again bounds related to the Field-of-Values of A, see (Simoncini and Druskin 2009).

Convergence

If we have $B \neq A^T$ things are more involved and due to (Beckermann 2011), and we need preliminary work.

- First of all, we need a more manageable expression of the rational Krylov subspace, let us re-brand the poles in the extended complex plane $\overline{\mathbb{C}} = \mathbb{C} \cup \{\infty\}$ as

$$z_{A,1}, \dots, z_{A,m} \in \overline{\mathbb{C}} \setminus \Lambda(A), \qquad z_{B,1}, \dots, z_{B,n} \in \overline{\mathbb{C}} \setminus \Lambda(B),$$

  and introduce the polynomials

$$Q_A(z) = \prod_{\substack{j=1 \\ z_{A,j} \neq \infty}}^{m} (z - z_{A,j}) \text{ and } Q_B(z) = \prod_{\substack{j=1 \\ z_{B,j} \neq \infty}}^{n} (z - z_{A,j}).$$

Convergence

If we have $B \neq A^T$ things are more involved and due to (Beckermann 2011), and we need preliminary work.

- First of all, we need a more manageable expression of the rational Krylov subspace, let us re-brand the poles in the extended complex plane $\overline{\mathbb{C}} = \mathbb{C} \cup \{\infty\}$ as

$$z_{A,1}, \dots, z_{A,m} \in \overline{\mathbb{C}} \setminus \Lambda(A), \qquad z_{B,1}, \dots, z_{B,n} \in \overline{\mathbb{C}} \setminus \Lambda(B),$$

and introduce the polynomials

$$Q_A(z) = \prod_{\substack{j=1 \\ z_{A,j} \neq \infty}}^{m} (z - z_{A,j}) \text{ and } Q_B(z) = \prod_{\substack{j=1 \\ z_{B,j} \neq \infty}}^{n} (z - z_{A,j}).$$

- The two rational spaces can then be written as

$$\mathcal{V} = \{R_A(A)C_1 \,:\, R_A \in \mathbb{P}_{m-1}/Q_A\}, \qquad \mathcal{W} = \{R_B(B)^H C_2 \,:\, R_B \in \mathbb{P}_{n-1}/Q_B\}.$$

Convergence

Consider the **rational functions** for the projected matrices A_m and B_n on $\mathcal{V}$ and $\mathcal{W}$

$$R_A^G(z) = \frac{\det(zI - A)}{Q_A(z)} \in \mathbb{P}_m / Q_A, \qquad R_B^G(z) = \frac{\det(zI - B_n)}{Q_B(z)} \in \mathbb{P}_n / Q_B$$

Convergence

Consider the **rational functions** for the projected matrices A_m and B_n on $\mathcal{V}$ and $\mathcal{W}$

$$R_A^G(z) = \frac{\det(zI - A)}{Q_A(z)} \in \mathbb{P}_m/Q_A, \qquad R_B^G(z) = \frac{\det(zI - B_n)}{Q_B(z)} \in \mathbb{P}_n/Q_B$$

Theorem (Beckermann 2011, Theorem 2.1)

Let $\operatorname{rank}(C) = 1$. The rational Galerkin residual ρ can be decomposed into the sum

$$\rho = \rho_{1,2} + \rho_{2,1} + \rho_{2,2}, \qquad \|\rho\|_F^2 = \|\rho_{1,2}\|_F^2 + \|\rho_{2,1}\|_F^2 + \|\rho_{2,2}\|_F^2,$$

with, $C_{1,m} = U^H C_1$, $C_{2,n} = V^H C_2$, and

$$\rho_{1,2} U \frac{1}{R_B^G}(A_m) C_{1,m} C_2^H R_B^G(B), \;\; \rho_{2,1} = R_A^G(A) C_1 C_{2,n}^H \frac{1}{R_A^G}(B_n) V^H,$$

$$\rho_{2,2} = \frac{R_A^G(A) C_1 C_2^H R_B^G(B)}{R_A^G(\infty) R_B^G(\infty)}.$$

Convergence

Consider the **rational functions** for the projected matrices A_m and B_n on $\mathcal{V}$ and $\mathcal{W}$

$$R_A^G(z) = \frac{\det(zI - A)}{Q_A(z)} \in \mathbb{P}_m/Q_A, \qquad R_B^G(z) = \frac{\det(zI - B_n)}{Q_B(z)} \in \mathbb{P}_n/Q_B$$

Theorem (Beckermann 2011, Theorem 2.1)

Let $\operatorname{rank}(C) = 1$. The rational Galerkin residual ρ can be decomposed into the sum

$$\rho = \rho_{1,2} + \rho_{2,1} + \rho_{2,2}, \qquad \|\rho\|_F^2 = \|\rho_{1,2}\|_F^2 + \|\rho_{2,1}\|_F^2 + \|\rho_{2,2}\|_F^2,$$

with, $C_{1,m} = U^H C_1$, $C_{2,n} = V^H C_2$, and

$$\|\rho_{2,2}\|_F = \inf_{\substack{R_A \in \mathbb{P}_m/Q_A \\ R_B \in \mathbb{P}_n/Q_B}} \left\| \frac{R_A(A) C_1 C_2^H R_B(B)}{R_A(\infty) R_B(\infty)} \right\|_F = \|(I - UU^H) C_1 C_2^H (I - VV^H)\|_F,$$

Convergence

Consider the **rational functions** for the projected matrices A_m and B_n on $\mathcal{V}$ and $\mathcal{W}$

$$R_A^G(z) = \frac{\det(zI - A)}{Q_A(z)} \in \mathbb{P}_m/Q_A, \qquad R_B^G(z) = \frac{\det(zI - B_n)}{Q_B(z)} \in \mathbb{P}_n/Q_B$$

Theorem (Beckermann 2011, Theorem 2.1)

Let $\operatorname{rank}(C) = 1$. The rational Galerkin residual ρ can be decomposed into the sum

$$\rho = \rho_{1,2} + \rho_{2,1} + \rho_{2,2}, \qquad \|\rho\|_F^2 = \|\rho_{1,2}\|_F^2 + \|\rho_{2,1}\|_F^2 + \|\rho_{2,2}\|_F^2,$$

with, $C_{1,m} = U^H C_1$, $C_{2,n} = V^H C_2$, and

$$\|\rho_{1,2}\|_F = \min_{R_B \in \mathbb{P}_m/Q_B} \left[\|R_B(A_m) C_{1,m} C_2^H R_B(B)\|_F + c_0 \|\frac{1}{R_B}(A_m) C_{1,m} C_{2,n}^H R_B(B_n)\|_F \right],$$

for $c_0 = 2\operatorname{diam}(W(A), W(B))/\operatorname{dist}(W(A), W(B))$.

Convergence

Consider the **rational functions** for the projected matrices A_m and B_n on $\mathcal{V}$ and $\mathcal{W}$

$$R_A^G(z) = \frac{\det(zI - A)}{Q_A(z)} \in \mathbb{P}_m/Q_A, \qquad R_B^G(z) = \frac{\det(zI - B_n)}{Q_B(z)} \in \mathbb{P}_n/Q_B$$

Theorem (Beckermann 2011, Theorem 2.1)

Let $\operatorname{rank}(C) = 1$. The rational Galerkin residual ρ can be decomposed into the sum

$$\rho = \rho_{1,2} + \rho_{2,1} + \rho_{2,2}, \qquad \|\rho\|_F^2 = \|\rho_{1,2}\|_F^2 + \|\rho_{2,1}\|_F^2 + \|\rho_{2,2}\|_F^2,$$

with, $C_{1,m} = U^H C_1$, $C_{2,n} = V^H C_2$, and

$$\|\rho_{2,1}\|_F = \min_{R_A \in \mathbb{P}_m/Q_A} \left[\|R_A(A) C_1 C_{2,n}^H \frac{1}{R_A}(B_n)\|_F + c_0 \|R_A(A_m) C_{1,m} C_{2,n}^H \frac{1}{R_A}(B_n)\|_F \right],$$

for $c_0 = 2 \operatorname{diam}(W(A), W(B)) / \operatorname{dist}(W(A), W(B))$.

Convergence

- Consider the **rational functions** for the projected matrices A_m and B_n on $\mathcal{V}$ and $\mathcal{W}$

$$R_A^G(z) = \frac{\det(zI - A)}{Q_A(z)} \in \mathbb{P}_m / Q_A, \qquad R_B^G(z) = \frac{\det(zI - B_n)}{Q_B(z)} \in \mathbb{P}_n / Q_B$$

- Now we have a **representation of the residual** in the **orthogonal bases** associated to the given Krylov subspaces, and furthermore we know that $\rho_{2,2} = 0$ if at least one of the $z_{A,j}$ or $z_{B,j}$ is ∞, i.e., if either of the initial vectors are in the subspace.

Convergence

- Consider the **rational functions** for the projected matrices A_m and B_n on $\mathcal{V}$ and $\mathcal{W}$

$$R_A^G(z) = \frac{\det(zI - A)}{Q_A(z)} \in \mathbb{P}_m / Q_A, \qquad R_B^G(z) = \frac{\det(zI - B_n)}{Q_B(z)} \in \mathbb{P}_n / Q_B$$

- Now we have a **representation of the residual** in the **orthogonal bases** associated to the given Krylov subspaces, and furthermore we know that $\rho_{2,2} = 0$ if at least one of the $z_{A,j}$ or $z_{B,j}$ is ∞, i.e., if either of the initial vectors are in the subspace.
- The bounds are then obtained by having upper-bounds of the quantities

$$E_m(\spadesuit, Q_\spadesuit, z) = \min_{p \in \mathbb{P}_\heartsuit} \frac{\left\| \frac{P}{Q_\spadesuit}(\spadesuit) \right\|}{\left| \frac{P}{Q_\spadesuit}(z) \right|}, \text{ for } \spadesuit = \{A, B\},\ \heartsuit = \{m, n\}.$$

Convergence

- Consider the **rational functions** for the projected matrices A_m and B_n on $\mathcal{V}$ and $\mathcal{W}$

$$R_A^G(z) = \frac{\det(zI - A)}{Q_A(z)} \in \mathbb{P}_m / Q_A, \qquad R_B^G(z) = \frac{\det(zI - B_n)}{Q_B(z)} \in \mathbb{P}_n / Q_B$$

- Now we have a **representation of the residual** in the **orthogonal bases** associated to the given Krylov subspaces, and furthermore we know that $\rho_{2,2} = 0$ if at least one of the $z_{A,j}$ or $z_{B,j}$ is ∞, i.e., if either of the initial vectors are in the subspace.
- The bounds are then obtained by having upper-bounds of the quantities

$$E_m(\spadesuit, Q_\spadesuit, z) = \min_{p \in \mathbb{P}_\heartsuit} \frac{\left\| \frac{P}{Q_\spadesuit}(\spadesuit) \right\|}{\left| \frac{P}{Q_\spadesuit}(z) \right|}, \text{ for } \spadesuit = \{A, B\},\ \heartsuit = \{m, n\}.$$

$\Rightarrow$ This can be faced by using the upper bound given by Crouziex upper-bound for matrix-functions.

Convergence: potential theory

In order to obtain the bounds and the rate of convergence, we need to work with the **Green functions** of $\overline{\mathbb{C}} \setminus W(A)$ and $\overline{\mathbb{C}} \setminus W(B)$ with poles at $\zeta \in \mathbb{C}$ called $g_A(\cdot, \zeta)$ and $g_B(\cdot, \zeta)$ respectively; (Saff and Totik 1997).

Convergence: potential theory

In order to obtain the bounds and the rate of convergence, we need to work with the **Green functions** of $\overline{\mathbb{C}} \setminus W(A)$ and $\overline{\mathbb{C}} \setminus W(B)$ with poles at $\zeta \in \mathbb{C}$ called $g_A(\cdot, \zeta)$ and $g_B(\cdot, \zeta)$ respectively; (Saff and Totik 1997).

With this **potential functions** the bound can then be expressed in terms of the functions

$$u_{A,m}(z) = \exp\left(-\sum_{j=1}^{m} g_A(z, z_{A,j})\right), \text{ and } u_{B,n}(z) = \exp\left(-\sum_{j=1}^{n} g_B(z, z_{B,j})\right).$$

Convergence: potential theory

In order to obtain the bounds and the rate of convergence, we need to work with the **Green functions** of $\overline{\mathbb{C}} \setminus W(A)$ and $\overline{\mathbb{C}} \setminus W(B)$ with poles at $\zeta \in \mathbb{C}$ called $g_A(\cdot, \zeta)$ and $g_B(\cdot, \zeta)$ respectively; (Saff and Totik 1997).

With this **potential functions** the bound can then be expressed in terms of the functions

$$u_{A,m}(z) = \exp\left(-\sum_{j=1}^{m} g_A(z, z_{A,j})\right), \text{ and } u_{B,n}(z) = \exp\left(-\sum_{j=1}^{n} g_B(z, z_{B,j})\right).$$

A mad research idea

Given the case we are interested in, can we find **optimal poles**, i.e., the one minimizing the bounds and have both α robustness, and M and N independence?

Let's blow up the bridges

What do we do if the **space coefficients** are **not separable**?

Let's blow up the bridges

What do we do if the **space coefficients** are **not separable**?

We decompose

$$d^{\pm}(x,y) = \sum_{k=1}^{K} t_k^{\pm} T_k(x) T_k(y)$$

and substitute in our equation obtaining

$$\sum_{k=1}^{K} \left(\tilde{A}_k X + X \tilde{B}_k^T \right) = C_1 C_2^T.$$

Let's blow up the bridges

What do we do if the **space coefficients** are **not separable**?

We decompose

$$d^{\pm}(x,y) = \sum_{k=1}^{K} t_k^{\pm} T_k(x) T_k(y)$$

and substitute in our equation obtaining

$$\sum_{k=1}^{K} \left(\tilde{A}_k X + X \tilde{B}_k^T \right) = C_1 C_2^T.$$

We can try generalize the Galerkin projection

$$\sum_{k=1}^{2K} \widehat{A}_k X \widehat{B}_k = C_1 C_2^T \Rightarrow \sum_{k=1}^{2K} (V_m^T \widehat{A}_k V_m) X (W_m^T \widehat{B}_k W_m) = V_m C_1 (W_m^T C_2)^T,$$

Let's blow up the bridges

What do we do if the **space coefficients** are **not separable**?

We decompose

$$d^{\pm}(x,y) = \sum_{k=1}^{K} t_k^{\pm} T_k(x) T_k(y)$$

and substitute in our equation obtaining

$$\sum_{k=1}^{K} \left(\tilde{A}_k X + X \tilde{B}_k^T \right) = C_1 C_2^T.$$

We can try generalize the Galerkin projection

$$\sum_{k=1}^{2K} \hat{A}_k X \hat{B}_k = C_1 C_2^T \Rightarrow \sum_{k=1}^{2K} (V_m^T \hat{A}_k V_m) X (W_m^T \hat{B}_k W_m) = V_m C_1 (W_m^T C_2)^T,$$

- How do we select $\mathcal{V}$ and $\mathcal{W}$? How do we generate nested subspace? How do we solve the reduced multiterm equation?

Let's blow up the bridges

What do we do if the **space coefficients** are **not separable**?

We decompose

$$d^{\pm}(x,y) = \sum_{k=1}^{K} t_k^{\pm} T_k(x) T_k(y)$$

and substitute in our equation obtaining

$$\sum_{k=1}^{K} \left(\tilde{A}_k X + X \tilde{B}_k^T \right) = C_1 C_2^T.$$

We can try generalize the Galerkin projection

$$\sum_{k=1}^{2K} \hat{A}_k X \hat{B}_k = C_1 C_2^T \Rightarrow \sum_{k=1}^{2K} (V_m^T \hat{A}_k V_m) X (W_m^T \hat{B}_k W_m) = V_m C_1 (W_m^T C_2)^T,$$

- How do we select $\mathcal{V}$ and $\mathcal{W}$? How do we generate nested subspace? How do we solve the reduced multiterm equation? $\Rightarrow$ many more questions than answers... ☹.

Let's blow up the bridges

What if the **convergence rate** is **poor**?

Let's blow up the bridges

What if the **convergence rate** is **poor**?

Let's blow up the bridges

What if the **convergence rate** is **poor**?

Let's blow up the bridges

What if the **convergence rate** is **poor**?

Since convergence depends on the spectrum, we may be tempted to precondition the equation with a matrix P, i.e.,

$$(P^{-1}AP)P^{-1}XP^{-H} + P^{-1}XP^{-H}(P^{H}BP^{-H}) = P^{-1}CP^{-H},$$

that is of no use since $P^{-1}AP \sim A$ and $P^{-1}BP \sim B$.

Can we use the **Kronecker structure** to put together **1D+2D** case as a **single matrix equation** or, more generally, **1D+dD equations** as a **single matrix equation**?

Let's blow up the bridges

What if the **convergence rate** is **poor**?

Since convergence depends on the spectrum, we may be tempted to precondition the equation with a matrix P, i.e.,

$$(P^{-1}AP)P^{-1}XP^{-H} + P^{-1}XP^{-H}(P^{H}BP^{-H}) = P^{-1}CP^{-H},$$

that is of no use since $P^{-1}AP \sim A$ and $P^{-1}BP \sim B$.

Can we use the **Kronecker structure** to put together **1D+2D** case as a **single matrix equation** or, more generally, **1D+dD equations** as a **single matrix equation**?

This is a whole *different can of worms* and it's called *tensor equations*.

Let's blow up the bridges

❓ What if the **convergence rate** is **poor**?

Since convergence depends on the spectrum, we may be tempted to precondition the equation with a matrix P, i.e.,

$$(P^{-1}AP)P^{-1}XP^{-H} + P^{-1}XP^{-H}(P^{H}BP^{-H}) = P^{-1}CP^{-H},$$

that is of no use since $P^{-1}AP \sim A$ and $P^{-1}BP \sim B$.

$\Rightarrow$ the only possibility is **playing around with the poles**.

❓ Can we use the **Kronecker structure** to put together **1D+2D** case as a **single matrix equation** or, more generally, **1D+dD equations** as a **single matrix equation**?

🔧 This is a whole *different can of worms* and it's called *tensor equations*.

❓ What if the **right-hand side** is **not low rank**?

Let's blow up the bridges

What if the **convergence rate** is **poor**?

Since convergence depends on the spectrum, we may be tempted to precondition the equation with a matrix P, i.e.,

$$(P^{-1}AP)P^{-1}XP^{-H} + P^{-1}XP^{-H}(P^{H}BP^{-H}) = P^{-1}CP^{-H},$$

that is of no use since $P^{-1}AP \sim A$ and $P^{-1}BP \sim B$.

$\Rightarrow$ the only possibility is **playing around with the poles**.

Can we use the **Kronecker structure** to put together **1D+2D** case as a **single matrix equation** or, more generally, **1D+dD equations** as a **single matrix equation**?

This is a whole *different can of worms* and it's called *tensor equations*.

What if the **right-hand side** is **not low rank**?

We can use some **approximation strategy**, solve the matrix-equation **incompletely** and use it as a **preconditioner** inside a FGMRES method, or *turn to other structures*...

Rank-structured matrices

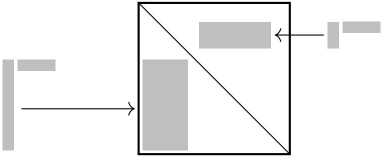
Global low-rank matrices is **not the only setting** in which computations can be spared!

Rank-structured matrices

Global low-rank matrices is **not the only setting** in which computations can be spared!

Quasiseparable matrix

A matrix A is *quasiseparable* of order k if the maximum of the ranks of all its submatrices contained in the strictly upper or lower part is less or equal than k.

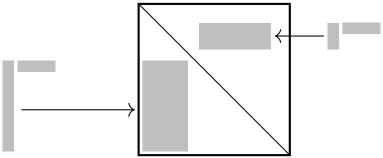


Rank-structured matrices

Global low-rank matrices is **not the only setting** in which computations can be spared!

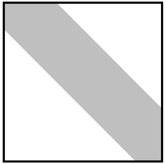
Quasiseparable matrix

A matrix A is *quasiseparable* of order k if the maximum of the ranks of all its submatrices contained in the strictly upper or lower part is less or equal than k.



Example: k-banded matrices

A banded matrix with bandwidth k is quasiseparable of order (at most) k. In particular, diagonal matrices are quasiseparable of order 0, tridiagonal matrices are quasiseparable of order 1, *etc.*

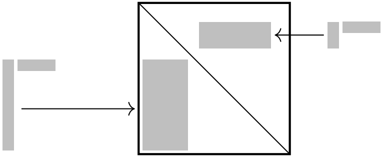


Rank-structured matrices

Global low-rank matrices is **not the only setting** in which computations can be spared!

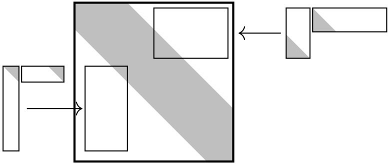
Quasiseparable matrix

A matrix A is *quasiseparable* of order k if the maximum of the ranks of all its submatrices contained in the strictly upper or lower part is less or equal than k.



Example: k-banded matrices

A banded matrix with bandwidth k is quasiseparable of order (at most) k. In particular, diagonal matrices are quasiseparable of order 0, tridiagonal matrices are quasiseparable of order 1, *etc.*



Rank-structured matrices

Theorem (Massei, Palitta, and Robol 2018, Theorem 2.7)

Let A and B be **symmetric positive definite** matrices of **quasiseparable** rank k_A and k_B, respectively, and suppose that the spectra of A and B are both contained in the interval $[a, b]$. Then, if X solves the Sylvester equation $AX + XB = C$, with C of **quasiseparable** rank k_C, a generic off-diagonal block Y of X satisfies

$$\frac{\sigma_{1+k\ell}(Y)}{\sigma_1(Y)} \leq 4\rho^{-2\ell},$$

where $k \triangleq k_A + k_B + k_C$, $\rho = \exp\left(\frac{\pi^2}{2\mu(\frac{b}{a})}\right)$ and $\mu(\cdot)$ the Grötzsch ring function

$$\mu(\lambda) \triangleq \frac{\pi}{2}\frac{K(\sqrt{1-\lambda^2})}{K(\lambda)}, \qquad K(\lambda) \triangleq \int_0^1 \frac{1}{(1-t^2)(1-\lambda^2 t^2)}\,\mathrm{d}t.$$

Rank-structured matrices

Theorem (Massei, Palitta, and Robol 2018, Theorem 2.7)

Let A and B be **symmetric positive definite** matrices of **quasiseparable** rank k_A and k_B, respectively, and suppose that the spectra of A and B are both contained in the interval $[a, b]$. Then, if X solves the Sylvester equation $AX + XB = C$, with C of **quasiseparable** rank k_C, a generic off-diagonal block Y of X satisfies

$$\frac{\sigma_{1+k\ell}(Y)}{\sigma_1(Y)} \leq 4\rho^{-2\ell},$$

where $k \triangleq k_A + k_B + k_C$, $\rho = \exp\left(\frac{\pi^2}{2\mu(\frac{b}{a})}\right)$ and $\mu(\cdot)$ the Grötzsch ring function

$$\mu(\lambda) \triangleq \frac{\pi}{2}\frac{K(\sqrt{1-\lambda^2})}{K(\lambda)}, \qquad K(\lambda) \triangleq \int_0^1 \frac{1}{(1-t^2)(1-\lambda^2 t^2)}\,\mathrm{d}t.$$

⚠ As usual, the **non-symmetric case** requires using the field-of-values!

Rank-structured matrices

Theorem (Massei, Palitta, and Robol 2018, Theorem 2.12)

Let A, B be matrices of quasiseparable rank k_A and k_B respectively and such that $W(A) \subseteq E$ and $W(-B) \subseteq F$. Consider the Sylvester equation $AX + XB = C$, with C of quasiseparable rank k_C. Then a generic off-diagonal block Y of the solution X satisfies

$$\frac{\sigma_{1+k\ell}(Y)}{\sigma_1(Y)} \leq \mathcal{C}^2 \cdot Z_\ell(E, F), \qquad k := k_A + k_B + k_C.$$

Where $Z_\ell(E, F)$ is the solution of the **Zolotarev problem**

$$Z_\ell(E, F) \triangleq \inf_{r(x) \in \mathcal{R}_{\ell,\ell}} \frac{\max_{x \in E} |r(x)|}{\min_{y \in F} |r(y)|}, \qquad \ell \geq 1,$$

for $\mathcal{R}_{\ell,\ell}$ is the set of rational functions of degree at most (ℓ, ℓ), and $\mathcal{C}$ is the Crouzeix universal constant.

The Zolotarev 3rd Problem

Zolotarev's **third problem** is exactly the computation of

$$Z_\ell(E,F) \triangleq \inf_{r(x)\in\mathcal{R}_{\ell,\ell}} \frac{\max_{x\in E}|r(x)|}{\min_{y\in F}|r(y)|}, \qquad \ell \geq 1,$$

for two given sets E, F and a degree ℓ, *informally*:

> *"Find a rational function that is as small as possible on a set E while being ≥ 1 in absolute value on another set F"*

The Zolotarev 3rd Problem

Zolotarev's **third problem** is exactly the computation of

$$Z_\ell(E,F) \triangleq \inf_{r(x)\in\mathcal{R}_{\ell,\ell}} \frac{\max_{x\in E}|r(x)|}{\min_{y\in F}|r(y)|}, \qquad \ell \geq 1,$$

for two given sets E, F and a degree ℓ, *informally*:

> *"Find a rational function that is as small as possible on a set E while being ≥ 1 in absolute value on another set F"*

- For general sets E and F the solution is not explicitly known.

The Zolotarev 3rd Problem

Zolotarev's **third problem** is exactly the computation of

$$Z_\ell(E,F) \triangleq \inf_{r(x)\in\mathcal{R}_{\ell,\ell}} \frac{\max_{x\in E}|r(x)|}{\min_{y\in F}|r(y)|}, \qquad \ell \geq 1,$$

for two given sets E, F and a degree ℓ, *informally*:

> *"Find a rational function that is as small as possible on a set E while being ≥ 1 in absolute value on another set F"*

- For general sets E and F the solution is not explicitly known.
- However, there are cases where a solution is known.

The Zolotarev 3rd Problem

Zolotarev's **third problem** is exactly the computation of

$$Z_\ell(E,F) \triangleq \inf_{r(x)\in\mathcal{R}_{\ell,\ell}} \frac{\max_{x\in E}|r(x)|}{\min_{y\in F}|r(y)|}, \qquad \ell \geq 1,$$

for two given sets E, F and a degree ℓ, *informally*:

> *"Find a rational function that is as small as possible on a set E while being ≥ 1 in absolute value on another set F"*

- For general sets E and F the solution is not explicitly known.
- However, there are cases where a solution is known.

Example: two equal intervals

One can prove that for $E = [-b, -1]$ and $F = [1, b]$ the solution is

$$\sup_{x\in[-b,1]\cup[1,b]} |R(x) - \mathrm{sgn}(x)| = \frac{\sqrt{Z_\ell(E,F)}}{1 + Z_\ell(E,F)}$$

The Zolotarev 3$^{\text{rd}}$ Problem

Zolotarev's **third problem** is exactly the computation of

$$Z_\ell(E,F) \triangleq \inf_{r(x)\in\mathcal{R}_{\ell,\ell}} \frac{\max_{x\in E}|r(x)|}{\min_{y\in F}|r(y)|}, \qquad \ell \geq 1,$$

for two given sets E, F and a degree ℓ, *informally*:

> *"Find a rational function that is as small as possible on a set E while being ≥ 1 in absolute value on another set F"*

- For general sets E and F the solution is not explicitly known.
- However, there are cases where a solution is known.

Example: two equal intervals

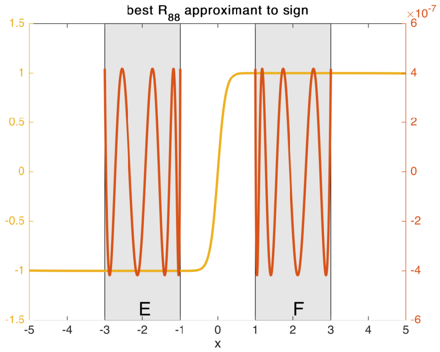
One can prove that for $E = [-b,-1]$ and $F = [1,b]$ the solution is

$$\sup_{x\in[-b,1]\cup[1,b]} |R(x) - \operatorname{sgn}(x)| = \frac{\sqrt{Z_\ell(E,F)}}{1+Z_\ell(E,F)} \;\Rightarrow\; \text{This is Zolotarev } 4^{\text{th}} \text{ problem!}$$

The Zolotarev 4^{th} Problem

A closed form solution, involving Jacobi elliptic functions, is available in the RKToolbox

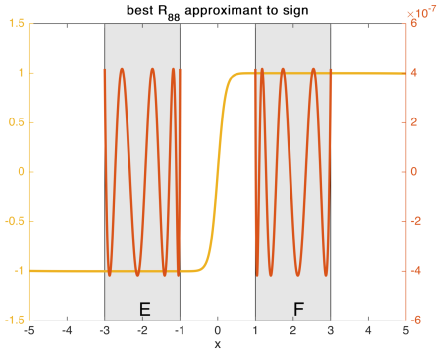
```
b = 3; % E = [-b,-1] and F = [1,b]
k = 8; % Degree of rational approximant to sign.
% Solution to Z's fourth problem:
r = rkfun.gallery('sign', k/2, b);
% Plot the computed rational function:
x = linspace(-5, 5, 1000);
y1 = linspace(-3, -1, 1000);
y2 = linspace(1, 3, 1000);
fill([-b -1 -1 -b -b], 1.5*[-1 -1 1 1 -1], .9*[1 1
↪  1] ),
hold on
fill([b 1 1 b b],1.5*[-1 -1 1 1 -1],.9*[1 1 1] )
[~,l1,l2] = plotyy(x,r(x),[y1 0 y2],[(1-abs(r(y1)))
↪  NaN (1-abs(r(y2)))]);
l1.LineWidth = 2; l2.LineWidth = 2;
hold off
```



The Zolotarev 4th Problem

A closed form solution, involving Jacobi elliptic functions, is available in the RKToolbox

```
b = 3; % E = [-b,-1] and F = [1,b]
k = 8; % Degree of rational approximant to sign.
% Solution to Z's fourth problem:
r = rkfun.gallery('sign', k/2, b);
% Plot the computed rational function:
x = linspace(-5, 5, 1000);
y1 = linspace(-3, -1, 1000);
y2 = linspace(1, 3, 1000);
fill([-b -1 -1 -b -b], 1.5*[-1 -1 1 1 -1], .9*[1 1
↪  1] ),
hold on
fill([b 1 1 b b],1.5*[-1 -1 1 1 -1],.9*[1 1 1] )
[~,l1,l2] = plotyy(x,r(x),[y1 0 y2],[(1-abs(r(y1)))
↪  NaN (1-abs(r(y2)))]);
l1.LineWidth = 2; l2.LineWidth = 2;
hold off
```



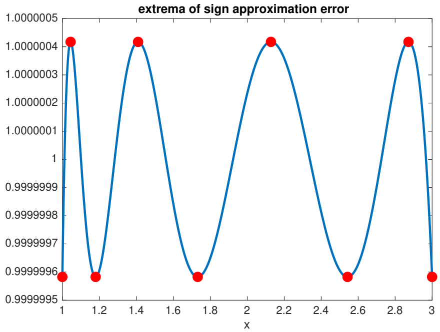
$\Rightarrow$ Zolotarev problem is then solved by solving for $Z_\ell(E, F)$.

The Zolotarev 3rd Problem

$$\text{Solve for } Z_\ell(E,F) \text{ s.t. } \sup_{x\in[-b,1]\cup[1,b]} |R(x) - \mathrm{sgn}(x)| = \frac{\sqrt{Z_\ell(E,F)}}{1+Z_\ell(E,F)}$$

```
% Extrema for [-1,-1/b]\cup [1/b,1]:
K = ellipke(1-1/b^2);
[sn, cn, dn] = ellipj((0:k)*K/k, 1-1/b^2);
% Transplant to [-b,-1]\cup [1,b]:
extrema = b*dn;
vals = 1-r(extrema);
c = mean( vals(1:2:end) );
e = eig( [ 2-4/c^2 1 ; 1 0 ] );
Zk = min(abs(e))
```

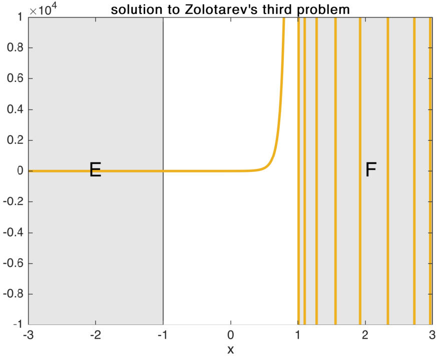
From which we obtain `Zk = 4.3542e-14`.



The Zolotarev 3rd Problem

To visualize the function realizing the extrema, one can use a Mobius transform to convert the best rational approximation to the sgn function that solves the 4th problem $r(x)$ to the extremal rational function $R_{\ell,\ell}(x)$ solving the 3rd:

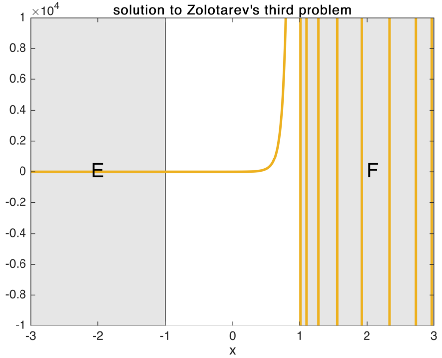
$$R_{\ell,\ell}(x) = \frac{\frac{1+Z_\ell(E,F)}{(1-Z_\ell(E,F))r(x)}}{\left(1 - \frac{1+Z_\ell(E,F)}{1-Z_\ell(E,F)} r(x)\right)}$$



The Zolotarev 3rd Problem

To visualize the function realizing the extrema, one can use a Mobius transform to convert the best rational approximation to the sgn function that solves the 4th problem $r(x)$ to the extremal rational function $R_{\ell,\ell}(x)$ solving the 3rd:

$$R_{\ell,\ell}(x) = \frac{\frac{1+Z_\ell(E,F)}{(1-Z_\ell(E,F))r(x)}}{\left(1 - \frac{1+Z_\ell(E,F)}{1-Z_\ell(E,F)} r(x)\right)}$$

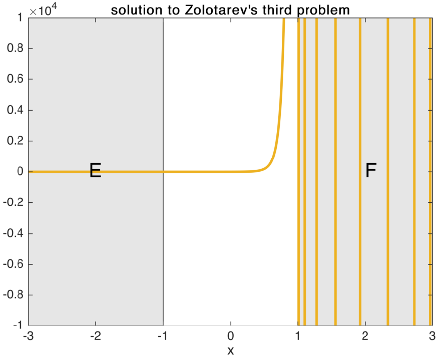


- There are **other cases** for which one can solve the 3rd problem, e.g., *unsymmetrical intervals*, or *rectangles* (Istace and Thiran 1995).

The Zolotarev 3rd Problem

To visualize the function realizing the extrema, one can use a Mobius transform to convert the best rational approximation to the sgn function that solves the 4th problem $r(x)$ to the extremal rational function $R_{\ell,\ell}(x)$ solving the 3rd:

$$R_{\ell,\ell}(x) = \frac{\frac{1+Z_\ell(E,F)}{(1-Z_\ell(E,F))r(x)}}{\left(1 - \frac{1+Z_\ell(E,F)}{1-Z_\ell(E,F)} r(x)\right)}$$



- There are **other cases** for which one can solve the 3rd problem, e.g., *unsymmetrical intervals*, or *rectangles* (Istace and Thiran 1995).
- If we are satisfied by the quasi-separability rank of the solution we can then attempt it!

Conclusion and summary

- ✅ We have reformulated several of our problems in terms of matrix equations,
- ✅ We have discussed projection methods for the solution of Sylvester equations,
- ✅ We have seen some limitations of the approach and shown a possible extension.

Next up

- 📋 More on rank-structured matrices and related solution strategies,
- 📋 All-at-once in time: using different methods to march in time than the standard ones,
- 📋 Still some other approaches with structured preconditioners.

Bibliography I

- Bartels, R. H. and G. W. Stewart (Sept. 1972). “Solution of the Matrix Equation $AX + XB = C$ [F4]”. In: *Commun. ACM* 15.9, pp. 820–826. ISSN: 0001-0782. DOI: 10.1145/361573.361582. URL: https://doi.org/10.1145/361573.361582.
- Beckermann, B. (2011). “An error analysis for rational Galerkin projection applied to the Sylvester equation”. In: *SIAM J. Numer. Anal.* 49.6, pp. 2430–2450. ISSN: 0036-1429. DOI: 10.1137/110824590. URL: https://doi.org/10.1137/110824590.
- Beckermann, B. and A. Townsend (2019). “Bounds on the singular values of matrices with displacement structure”. In: *SIAM Rev.* 61.2. Revised reprint of ”On the singular values of matrices with displacement structure” [MR3717820], pp. 319–344. ISSN: 0036-1445. DOI: 10.1137/19M1244433. URL: https://doi.org/10.1137/19M1244433.
- Breiten, T., V. Simoncini, and M. Stoll (2016). “Low-rank solvers for fractional differential equations”. In: *Electron. Trans. Numer. Anal.* 45, pp. 107–132.
- Carvajal, O. A., F. W. Chapman, and K. O. Geddes (2005). “Hybrid symbolic-numeric integration in multiple dimensions via tensor-product series”. In: *Proceedings of the 2005 international symposium on Symbolic and algebraic computation*, pp. 84–91.

Bibliography II

Driscoll, T. A., N. Hale, and L. N. Trefethen (2014). *Chebfun guide*.

Gohberg, I., T. Kailath, and V. Olshevsky (1995). “Fast Gaussian elimination with partial pivoting for matrices with displacement structure”. In: *Math. Comp.* 64.212, pp. 1557–1576. ISSN: 0025-5718. DOI: 10.2307/2153371. URL: https://doi.org/10.2307/2153371.

Golub, G., S. Nash, and C. Van Loan (1979). “A Hessenberg-Schur method for the problem AX + XB= C”. In: *IEEE Transactions on Automatic Control* 24.6, pp. 909–913. DOI: 10.1109/TAC.1979.1102170.

Halko, N., P. G. Martinsson, and J. A. Tropp (2011). “Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions”. In: *SIAM Rev.* 53.2, pp. 217–288. ISSN: 0036-1445. DOI: 10.1137/090771806. URL: https://doi.org/10.1137/090771806.

Istace, M.-P. and J.-P. Thiran (1995). “On the third and fourth Zolotarev problems in the complex plane”. In: *SIAM J. Numer. Anal.* 32.1, pp. 249–259. ISSN: 0036-1429. DOI: 10.1137/0732009. URL: https://doi.org/10.1137/0732009.

Bibliography III

- Massei, S., D. Palitta, and L. Robol (2018). "Solving rank-structured Sylvester and Lyapunov equations". In: *SIAM J. Matrix Anal. Appl.* 39.4, pp. 1564–1590. ISSN: 0895-4798. DOI: 10.1137/17M1157155. URL: https://doi.org/10.1137/17M1157155.
- Saff, E. B. and V. Totik (1997). *Logarithmic potentials with external fields*. Vol. 316. Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. Appendix B by Thomas Bloom. Springer-Verlag, Berlin, pp. xvi+505. ISBN: 3-540-57078-0. DOI: 10.1007/978-3-662-03329-6. URL: https://doi.org/10.1007/978-3-662-03329-6.
- Simoncini, V. (2007). "A new iterative method for solving large-scale Lyapunov matrix equations". In: *SIAM J. Sci. Comput.* 29.3, pp. 1268–1288. ISSN: 1064-8275. DOI: 10.1137/06066120X. URL: https://doi.org/10.1137/06066120X.
- — (2016). "Computational methods for linear matrix equations". In: *SIAM Rev.* 58.3, pp. 377–441. ISSN: 0036-1445. DOI: 10.1137/130912839. URL: https://doi.org/10.1137/130912839.

Bibliography IV

Simoncini, V. and V. Druskin (2009). "Convergence analysis of projection methods for the numerical solution of large Lyapunov equations". In: *SIAM J. Numer. Anal.* 47.2, pp. 828–843. ISSN: 0036-1429. DOI: 10.1137/070699378. URL: https://doi.org/10.1137/070699378.

Townsend, A. and L. N. Trefethen (2013). "An extension of Chebfun to two dimensions". In: *SIAM J. Sci. Comput.* 35.6, pp. C495–C518. ISSN: 1064-8275. DOI: 10.1137/130908002. URL: https://doi.org/10.1137/130908002.

Tyrtyshnikov, E. E. (2000). "Incomplete cross approximation in the mosaic-skeleton method". In: vol. 64. 4. International GAMM-Workshop on Multigrid Methods (Bonn, 1998), pp. 367–380. DOI: 10.1007/s006070070031. URL: https://doi.org/10.1007/s006070070031.