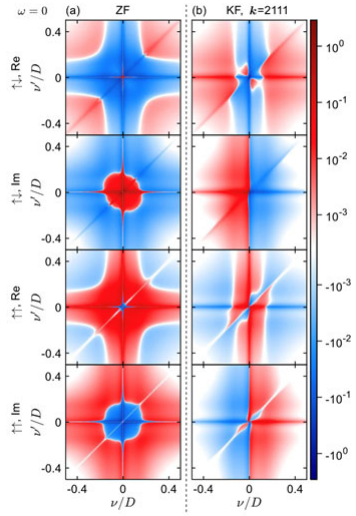
Tensor Cross Interpolation (TCI) I [Fernandez2024]

1. Motivation

4-point vertex functions of single-impurity Anderson model



Many branches of science deal with complicated functions of many variables (e.g. classical or quantum field theories…).

These functions often have structure of many different length scales.

(i) Long scales: demand large domain of definition.

(ii) Short length scales demand dense grid.

(i) and (ii) together imply: numerics requires exploding costs in memory and computation time = 'curse of dimensionality'

Can explosion of costs be avoided without loss of control of accuracy?

Yes, if functions are "compressible"!

Famous example: quantum many-body wavefunctions.

$$|\Psi\rangle = \sum_{\sigma} F_{\sigma_1 \dots \sigma_{\mathcal{L}}} |\sigma_1\rangle \otimes |\sigma_2\rangle \otimes \cdots \otimes |\sigma_{\mathcal{L}}\rangle, \quad \sigma = (\sigma_1, \dots, \sigma_{\mathcal{L}})$$

They can always be expressed as tensors:

$F_{\sigma} =$ 

'degree of tensor' = number of legs = $\mathcal{L}$

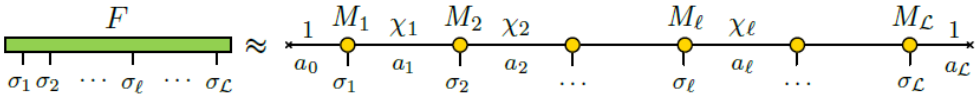
'local index' = $\sigma_\ell = 1, \dots, d$ 'local dimension' = d

Total number of elements of $F_{\vec{\sigma}}$: $d^{\mathcal{L}} \Rightarrow F_{\vec{\sigma}} \in \mathbb{C}^{d^{\mathcal{L}}}$

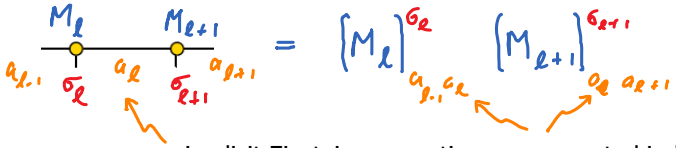
exponentially large in $\mathcal{L}$ $\Rightarrow$ 'curse of dimensionality'

Any tensor can be 'decomposed' or 'unfolded' into a 'matrix product state' (MPS) or 'tensor train' (TT):

$$F_{\sigma} \approx \widetilde{F}_{\sigma} = \prod_{\ell=1}^{\mathcal{L}} M_\ell^{\sigma_\ell} = [M_1]^{\sigma_1}_{1 a_1} [M_2]^{\sigma_2}_{a_1 a_2} \cdots [M_{\mathcal{L}}]^{\sigma_{\mathcal{L}}}_{a_{\mathcal{L}-1} 1}, \quad (1)$$



(2)



(3)

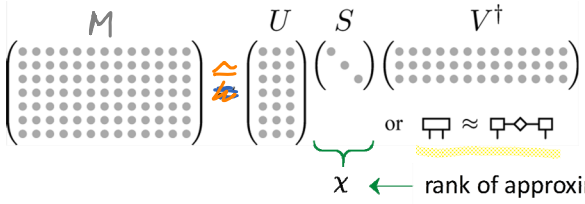
implicit Einstein summation over repeated indices: $\sum_{a_\ell=1}^{\chi_\ell}$

'bond index' = $a_\ell = 1, \dots \chi_\ell$

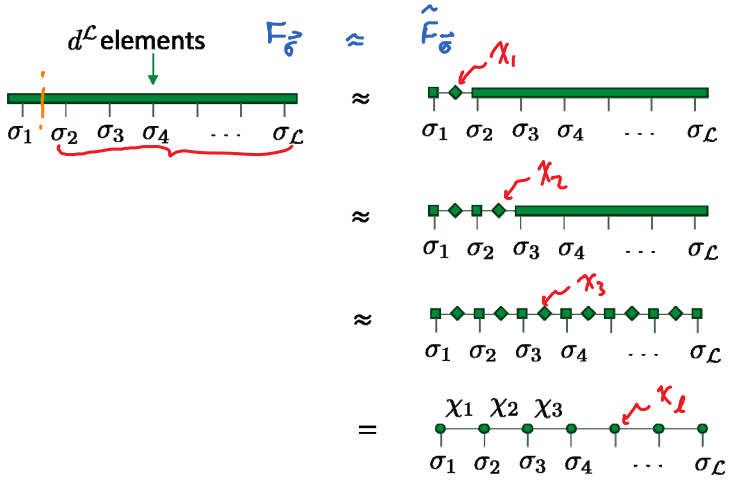
'bond dimension' = χ_ℓ

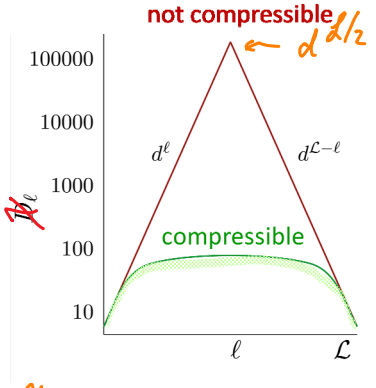
Standard tool for unfolding tensors via repeated singular value decomposition (SVD):

Matrix SVD: $M = U S V^\dagger$



Tensor SVD:





$\chi = \chi_{\max}$ is measure of maximal "bond entanglement"

$\mathcal{O}(d\chi^2\mathcal{L})$ elements $\ll d^{\mathcal{L}}$ if $F_{\vec{\sigma}}$ is compressible.

Tensor SVD is

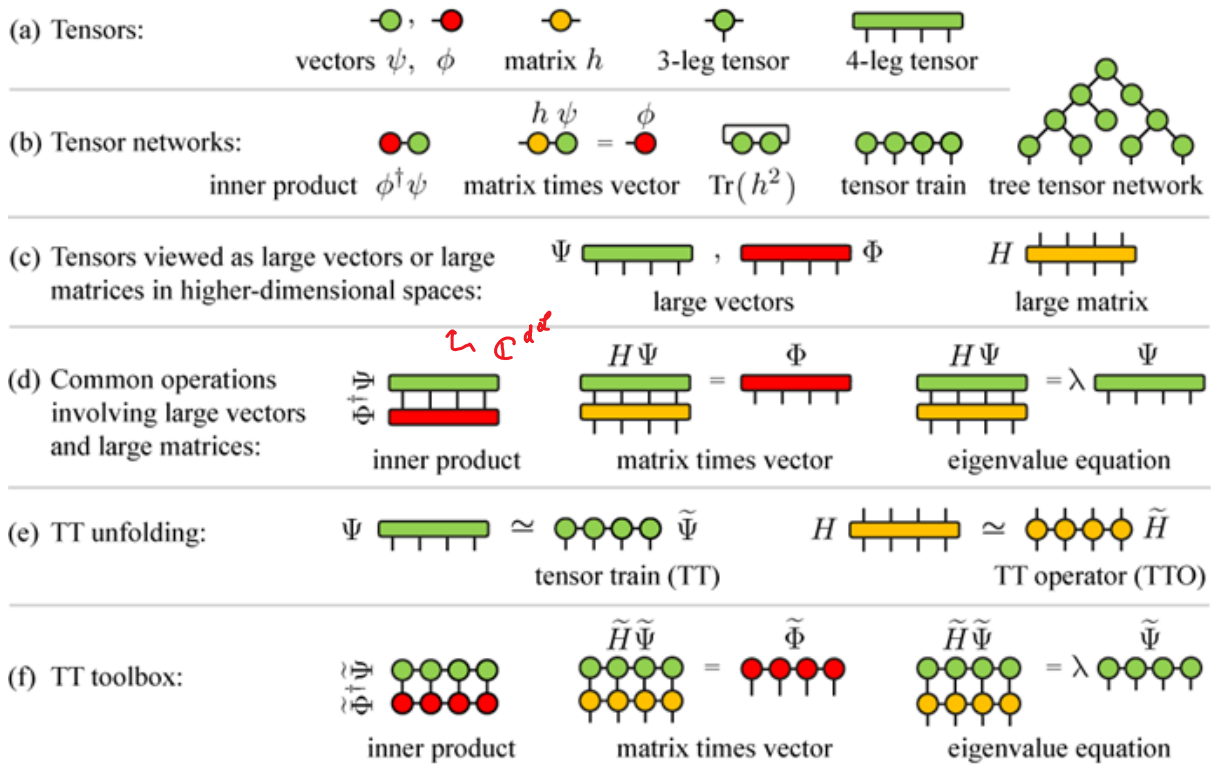
'optimal': minimizes $\| F_{\vec{\sigma}} - \tilde{F}_{\vec{\sigma}} \|$ in Frobenius norm.

'rank-revealing': reveals correct rank (bond dimension) of every bond.

'expensive': needs all $d^{\mathcal{L}}$ elements of $F_{\vec{\sigma}}$ (else decomposition could not be optimal)!

$F_{\vec{\sigma}}$ is strongly compressible if it can be approximated by TT $\tilde{\tilde{F}}_{\vec{\sigma}}$ with low bond dimension: $\chi_{\max} \ll d^{\mathcal{L}}$

Miraculous fact: ground-state wave-functions of 1-dimensional Hamiltonians with short-ranged interactions are compressible! This led to development of DMRG and a powerful 'standard toolbox' for manipulating TTs (and more general tensor networks, e.g. tree tensor networks, ...)



Recent (since ~2010) insight: compressible tensors arise not only in quantum many-body theory.

Any discretized function can be expressed as a tensor. Often, that tensor is compressible!

If so, standard function manipulations (addition, multiplication, integration, convolution, Fourier transform) can be performed using tensor network toolbox, with exponential savings in numerical cost (memory & time)!

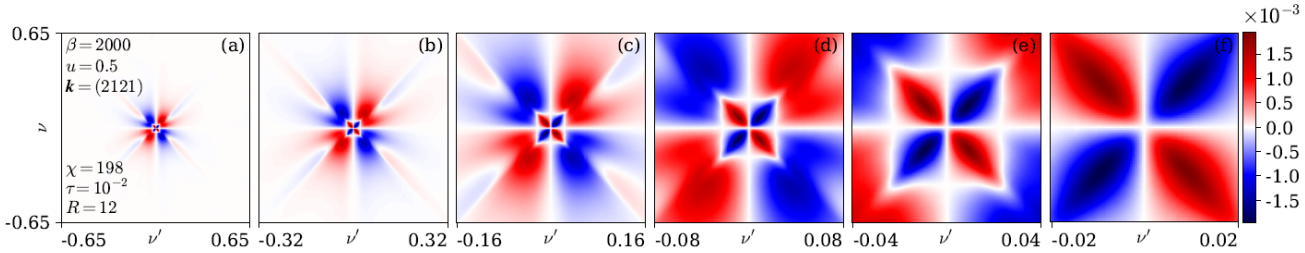
Bottleneck: standard strategy for unfolding tensor to a TT uses SVD, which requires knowledge of all d^L components of tensor. So, SVD-based unfolding is exponentially costly.

Recent progress: novel unfolding strategy based on tensor cross interpolation (TCI).

If tensor is compressible, TCI finds TT representations at runtime costs $\mathcal{O}(dL\chi^3)$
(If tensor is not compressible, TCI signals this by converging very slowly or not at all.)

TCI 'learns' compressed representation by interrogating only tiny subset of all elements of tensor.
This is akin to active machine learning, using a structured model (namely the TT) to represent the data.

TCI compression of 4-point vertex of Single-impurity Anderson model: allows zooming in without loss of resolution:



Historical remarks:

TCI pioneering papers (published in applied maths journals):
[Oseledets2010, Oseledets2011, Savostyanov2011, Savostyanov2014, Dolgov2020]

Early applications in physics (and beyond):

[Sozykin2022] Find minima of functions

[Fernandez2023] efficient (sign-problem-free) alternative to Monte Carlo sampling for calculating high-dimensional integrals arising in Feynman diagrams for quantum many-body systems

[Ritter2023] Compute topological invariants

[Jolly2023] Compute overlaps between atomic orbitals, solve Schrödinger equation of H_2^+ ion

[Sakurai2024] In mathematical finance, to speed up Fourier-transform-based option pricing

Open-source toolbox:

https://tensor4all.org/

analyze your favorite function → using our tools → obtain compressibility diagnosis

2. Integration of high-dimensional functions

[Fernandez2024, Sec. 2.2]

TCI.1

Consider integral over function of L variables: $I = \int_{\mathcal{U}} d^L x \, f(\vec{x})$ (1)

$f : \mathcal{U} = D^L \subset \mathbb{R}^L \to \mathbb{R}$, $\vec{x} = (x_1, .., x_l, ..., x_L) \mapsto f(\vec{x})$ (2)

(for simplicity, same integration domain for each variable)

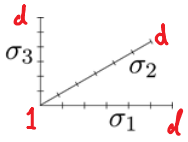
Standard approach: Monte-Carlo sampling of integrand.
Well-known issue: sign problem -- very slow convergence if integrand has many sign changes.

TT approach: discretize function, represent it as tensor, unfold it to TT: $f(\vec{x}) = F_{\vec{\sigma}} \simeq \widetilde{F}_{\vec{\sigma}} = \prod_l M^{\sigma_l}_l$ (3)
This yields L independent integrals that can be computed separately.

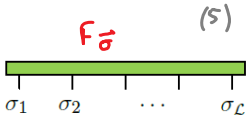
Discretization: (assume same grid for all variables)

Represent each variable on grid of d distinct points: $x_l \to \{p_1, .., p_{\sigma_l}, .., p_d\}$ (4)

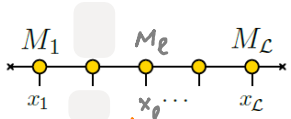
enumerated by an index $\sigma = 1, .., d$



'Natural tensor representation' of function: $F_{\vec{\sigma}} = f(p_{\sigma_1}, .., p_{\sigma_l}, .., p_{\sigma_L})$ (5)
$\vec{\sigma} = (\sigma_1, ..., \sigma_L)$



Unfold this tensor into a TT [using SVD (expensive) or TCI (much cheaper)]:
will be explained in detail later!

$f(x_1, .., x_L) \approx \widetilde{F}_{\vec{\sigma}} =$  $= M_1(x_1) \cdots M_l(x_l) \cdots M_L(x_L)$ (6)

shorthand for $x_l(\sigma_l)$ $= M^{\sigma_1}_1 \cdots M^{\sigma_l}_l \cdots M^{\sigma_L}_L$ (7)

When $\widetilde{F}$ is low rank (χ_{max} = small), f is almost separable' (it would be separable if $\chi = 1$)

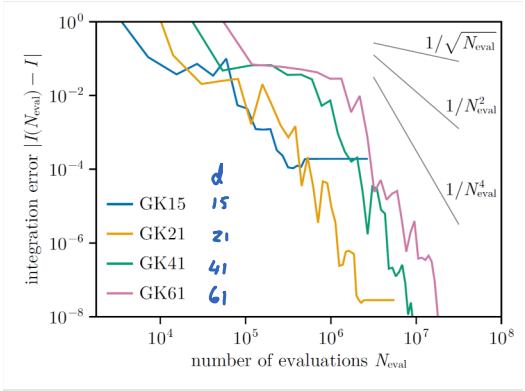
Integral separates into L one-dimensional integrals, which in practice are performed as discrete sums, followed by sequence of matrix-vector multiplications:

$$I = \int_{\mathcal{U}} d\vec{x} \, f(\vec{x}) \simeq \int_D dx_1 M_1(x_1) \cdots \int_D dx_l M_l(x_l) \cdots \int_D dx_L M_L(x_L) \quad (8)$$

$$= \left(\sum_{\sigma_1} M^{\sigma_1}_1\right) \cdot \left(\sum_{\sigma_l} M^{\sigma_l}_l\right) \cdot \left(\sum_{\sigma_L} M^{\sigma_L}_L\right) \quad (9)$$

Example: 10-dimensional integral of rapidly oscillating function:

$$I = 10^3 \int_{[-1,+1]^{10}} \mathrm{d}^{10}\mathbf{x} \cos\left(10 \textstyle\sum_{\ell=1}^{10} x_\ell^2\right) \exp\left[-10^{-3}\left(\textstyle\sum_{\ell=1}^{10} x_\ell\right)^4\right] \quad (10)$$

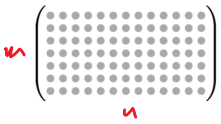


Convergence with number of function evaluations: TCI: $\sim N_{eval}^{-4}$ (fast!) Monte Carlo: $\sim N_{eval}^{-1/2}$ (slow!)

TCI unfolding gives huge speed-up for multi-dimensional integration (without sign problem)!

3. Matrix cross interpolation (MCI)

TCI.3

Consider $m \times n$ matrix $A = (\vec{a}_1, \vec{a}_2, \dots, \vec{a}_n) =$  (1)

with column vectors $\vec{a}_j \in \mathbb{C}^m$

'range of A' = $\mathrm{span}\{\vec{a}_1, \vec{a}_2, \dots, \vec{a}_n\}$ (2)

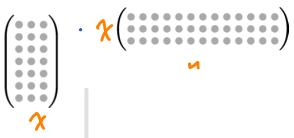
'rank of A' = dimension of range of A = number of linearly independent columns
= number of linearly independent ~~columns~~ rows (3)

If A has rank = χ, then every column can be represented

as linear combination of subset of χ columns, $\vec{a}_j = \sum_i \vec{b}_i C_{ij}$ (4)

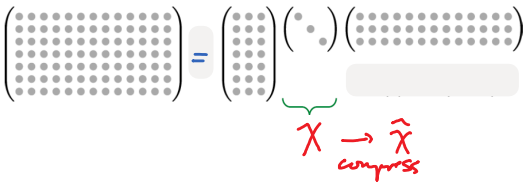
Let $\vec{B}$ be submatrix of $\vec{A}$ containing those columns: $\vec{B} = (\vec{b}_1, \dots, \vec{b}_\chi)$ (5)

Matrix elements of A: $A_{kj} = (\vec{a}_j)_k \overset{(4)}{=} \sum_{i=1}^{\chi} (\vec{b}_i)_k C_{ij} = \sum_{i=1}^{\chi} B_{ki} C_{ij}$ (6)

So, A can be factorized: $A_{m\times n} = B_{m\times\chi} \cdot C_{\chi\times n} =$  (7)

This is compressed representation:

Memory cost: $m \cdot n$ elements for A, $(m+n)\chi$ for B and C.

SVD reveals rank: $A = USV^+$:  (8)

but computing it requires entire matrix A

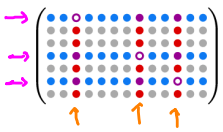
$\chi \to \tilde{\chi}$ compress

Alternative low-rank approximation techniques exist that require only subset of rows and columns of matrix:

(i) matrix cross interpolation (CI), and (ii) partial rank-revealing LU decompostion (prrLU)

this section — next ~~section~~ time

Matrix cross interpolation (CI)

Let A be $m \times n$ matrix of rank χ with elements A_{ij}.  (9)

$i = \mathbb{I} = \{1, \dots, m\}$ list of all row indices, $j \in \mathbb{J} = \{1, \dots, n\}$ list of column indices (10)

$\mathcal{I} = \{i_1, \dots, i_{\hat{\chi}}\} \subset \mathbb{I}$ sublist of row indices, $\mathcal{J} = \{j_1, \dots, j_{\hat{\chi}}\} \subset \mathbb{J}$ sublist of column indices (11)

$A[\mathcal{I}, \mathcal{J}]$ = 'slice' of A containing all intersections of $\mathcal{I}$-rows and $\mathcal{J}$-columns: (12)

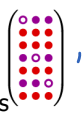
$A[\mathcal{I}, \mathcal{J}]_{\alpha\beta} = A_{i_\alpha j_\beta} \quad \forall\ \alpha, \beta \in \{1, \dots, \hat{\chi}\}$ In particular: $A[\mathbb{I}, \mathbb{J}] = A$ (13)

Assume $\hat{\chi} \leq \chi$, with $\mathcal{I}, \mathcal{J}$ chosen such that $A[\mathcal{I},\mathcal{J}]$ is non-singular: $\det\left(A[\mathcal{I},\mathcal{J}]\right) \neq 0$ (14)

Define: $P = A[\mathcal{I},\mathcal{J}] =$  ,

'pivot matrix', its elements are called 'pivots'

$\det P \neq 0$ (14) $\Rightarrow P^{-1}$ exists

$C = A[\mathbb{I},\mathcal{J}] =$  ,

gathers all columns containing pivots, i.e. all 'pivot columns'

$R = A[\mathcal{I},\mathbb{J}] =$  (15)

gathers all rows containing pivots, i.e. all 'pivot rows'

CI formula:

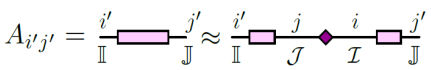
gives rank-$\hat{\chi}$ approximation $\hat{A}$ of A

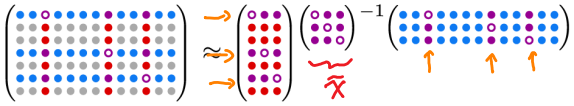
external indices i', j' are fixed,

internal bonds represent sums over pivot lists: $\sum_{i \in \mathcal{I}}$ $\sum_{j \in \mathcal{J}}$

$$A \approx C \cdot P^{-1} \cdot R = \hat{A} \quad (16)$$

$$A[\mathbb{I},\mathbb{J}] \approx A[\mathbb{I},\mathcal{J}] \cdot P^{-1} A[\mathcal{I},\mathbb{J}] \quad (17)$$

$A_{i'j'} =$  (18)

 (19)

Remark: C and R contain original columns and rows of A itself, not linear combinations of those. By contrast, for SVD, U and V^+ contain linear combinations of rows and columns of A.

Properties of CI formula:

(i) For $\tilde{\chi} \leq \chi$, $\tilde{A}$ yields an 'interpolation', i.e. it exactly reproduces all $\mathcal{I}$-rows and all $\mathcal{J}$-columns of A, (21)

'pivot rows' 'pivot columns'

i.e. 'all rows and columns from which $\tilde{A}$ was built'.

Proof: consider only pivot rows ($\mathcal{I}$-rows) or pivot columns ($\mathcal{J}$- columns):

$$\tilde{A}[\mathcal{I},\mathbb{J}] \overset{(16)}{=} \underbrace{A[\mathcal{I},\mathcal{J}] \cdot P^{-1}}_{1_{\hat{\chi} \to \hat{\chi}}} A[\mathcal{I},\mathbb{J}] = A[\mathcal{I},\mathbb{J}] \quad \rightrightarrows \left(\quad\right)\left(\quad\right)^{-1}\left(\quad\right) = \left(\quad\right) \quad (22)$$

$$\tilde{A}[\mathbb{I},\mathcal{J}] \overset{(16)}{=} A[\mathbb{I},\mathcal{J}] \underbrace{\cdot P^{-1} A[\mathcal{I},\mathcal{J}]}_{1_{\hat{\chi} \to \hat{\chi}}} = A[\mathbb{I},\mathcal{J}] \quad \left(\quad\right)\left(\quad\right)^{-1}\left(\quad\right) = \left(\quad\right) \quad (23)$$

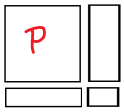
In general, $\tilde{A}$ has no information about the 'other' (non-pivot) rows and columns of A. If these are unrelated to the pivot rows and columns (e.g. because A is a random matrix), CI formula will get them completely wrong. However, if the non-pivot rows and columns are somewhat / very / fully related to the pivot rows and columns, i.e. if some / most / all of them can be expressed as linear combination of pivot rows and columns, the CI formula will reproduce them somewhat well / very well / exactly.

(ii) For $\tilde{\chi} = \chi$, $\tilde{A}$ exactly reproduces the entire matrix, $\tilde{A} = A$ (20)

'Plausibility argument': for $\tilde{\chi} = \chi$, all columns of A can be expressed as linear combinations of a subset of $\tilde{\chi}$ columns. Choose these to be pivot columns. Since CI formula reproduces all pivot columns exactly, it is plausible that it reproduces all columns of A exactly.

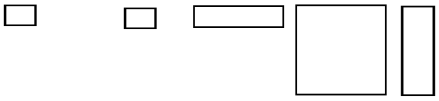
Caveat: does the CI for the non-pivot columns yield the correct linear combinations of pivot columns? Answer is yes. Proof follows further below.

(iii) Error of CI formula is determined by $[A/P]$, the Schur complement of P in A (21)

Definition: for block matrix: $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ =  with A_{11} assumed square and invertible,

the 'Schur complement of A_{11} in A is: $[A/A_{11}] \equiv A_{22} - A_{21}(A_{11})^{-1}A_{12}$ (22)

note index structure: 1 meets 1

block dimensions match: 

for future reference:

Schur determinant identity: $\det A = \det A_{11} \det[A/A_{11}]$ (23)

proof: see (TCI.4.3-5)

Consider arbitrary matrix A and pivot matrix P.

Permute rows and columns of A such that all pivots lie in first χ rows and columns, labeled

$\mathcal{I}_1 = \mathcal{J}_1 = \{1, \ldots, \chi\}$. Denote labels of remaining rows and columns by $\mathcal{I}_2 = \mathbb{I} \setminus \mathcal{I}_1$, $\mathcal{J}_2 = \mathbb{J} \setminus \mathcal{J}_1$

Permuted matrix:

$$A(\mathbb{I}, \mathbb{J}) = \begin{pmatrix} A(\mathcal{I}_1, \mathcal{J}_1) & A(\mathcal{I}_1, \mathcal{J}_2) \\ A(\mathcal{I}_2, \mathcal{J}_1) & A(\mathcal{I}_2, \mathcal{J}_2) \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \qquad P = A_{11} = A(\mathcal{I}_1, \mathcal{J}_1) \tag{24}$$

R

C

CI formula: $\tilde{A} = \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} (A_{11})^{-1} \begin{pmatrix} A_{11} & A_{12} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{21}(A_{11})^{-1}A_{12} \end{pmatrix}$ (25)

CI error: $A - \tilde{A} = \begin{pmatrix} 0 & 0 \\ 0 & [A/A_{11}] \end{pmatrix}$. (26)

Error depends on inverse of pivot matrix. So, strategy for reducing error is to choose pivots such that $|\det P|$ is maximal. This is known as 'maximum volume principle'.

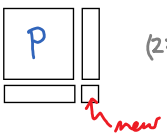
Finding 'best pivots', i.e. ones that satisfy maximum volume principle, is in general exponentially difficult; but good heuristics exist that get close to optimum in practice.

Proof of (ii): (rigorous version of plausibility argument given above)

For $\hat{\chi} = \chi$, pivot matrix $P = A[\mathcal{I}, \mathcal{J}]$ has dimension $\chi \times \chi$

Pick arbitrary 'new' element (x_0, y_0) of A that does not sit on a pivot row or pivot column.

Consider $(\chi+1) \times (\chi+1)$ submatrix of A, built from pivot matrix and new element:

$$A' = \begin{pmatrix} A'_{11} & A'_{12} \\ A'_{21} & A'_{22} \end{pmatrix} = \begin{pmatrix} A(\mathcal{I}, \mathcal{J}) & A(\mathcal{I}, y_0) \\ A(x_0, \mathcal{J}) & A(x_0, y_0) \end{pmatrix} = \text{  } \tag{27}$$

new

Recall Schur determinant identity: $\det A' = \det A'_{11} \det[A/A'_{11}]$ (28)

$A'_{22} - A'_{21}(A'_{11})^{-1}A'_{12}$

Applied to A':

$$0 = \det A' = \det[A(\mathcal{I}, \mathcal{J})] \det[A(x_0, y_0) - A(x_0, \mathcal{J})A^{-1}(\mathcal{I}, \mathcal{J})A(\mathcal{I}, y_0)$$

Applied to A' :

$$0 = \det A' = \underbrace{\det[A(\mathcal{I},\mathcal{J})]}_{\neq 0} \det[A(x_0,y_0) - A(x_0,\mathcal{J})A^{-1}(\mathcal{I},\mathcal{J})A(\mathcal{I},y_0)] \tag{29}$$

$= A(x_0,y_0)$

A' has $1+\chi$ columns, hence they are linearly dependent

pivot matrix is non-singular

Therefore:

$$A(x_0,y_0) = A(x_0,\mathcal{J})A^{-1}(\mathcal{I},\mathcal{J})A(\mathcal{I},y_0) \overset{(17)}{=} \tilde{A}(x_0,y_0) \tag{30}$$

Thus, CI formula exactly reproduces any 'new' element that dos not sit on pivot row or pivot column.

4. Properties of Schur complement

TCI.4

Consider block matrix: $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$ with A_{11} assumed square and invertible. (1)

Definition: 'Schur complement of A_{11} in A : $[A/A_{11}] \equiv A_{22} - A_{21}(A_{11})^{-1}A_{12}$ (2)

A can be factorized as follows:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} \mathbb{1}_{11} & 0 \\ A_{21}A_{11}^{-1} & \mathbb{1}_{22} \end{pmatrix} \begin{pmatrix} A_{11} & 0 \\ 0 & [A/A_{11}] \end{pmatrix} \begin{pmatrix} \mathbb{1}_{11} & A_{11}^{-1}A_{12} \\ 0 & \mathbb{1}_{22} \end{pmatrix} \quad (3)$$

check this by multiplying out:

$$= \begin{pmatrix} \mathbb{1}_{11} & 0 \\ A_{21}A_{11}^{-1} & \mathbb{1}_{22} \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ 0 & [A/A_{11}] \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{21}(A_{11})^{-1}A_{12} + [A/A_{11}] \end{pmatrix} \quad (4)$$

$$\overset{(2)}{=} A_{22}$$

Schur determinant identity: $\det A = \det A_{11} \det[A/A_{11}]$ (5)

follows from det(3)