Exercise session Numerical Linear Algebra: matrix compression in practice

1 Linear-time ACA

You have previously seen *Adaptive Cross Approximation* (ACA) as an effective compression scheme. In this section we will devise a linear-time version. **Recall the theoretical form of ACA from the lecture notes, and show its equivalence to the form given in algorithm 1.**
What is the complexity of algorithm 1? Be precise.

Algorithm 1: Linear-time ACA

input : Block $B(\mathbf{t}, \mathbf{s}) \in \mathbb{C}^{\mathbf{t}\times\mathbf{s}}$, tolerance **tol**
output: Low-rank approx $B(\mathbf{t}, \mathbf{s}) \approx B_r = XY^T$, $X \in \mathbb{C}^{\mathbf{t}\times r}, Y \in \mathbb{C}^{\mathbf{s}\times r}$

init $\epsilon := 1$, $k = 0$
while $\epsilon > tol$ **do**
  select pivot $(i_k, j_k) \in \mathbf{t} \times \mathbf{s}$;
  $\mathbf{x}_k := B(:, j_k)$, $\mathbf{y}_k := B(i_k, :)$;
  $\mathbf{x}_k := \mathbf{x}_k - \sum_{\mu<k} \mathbf{x}_\mu \mathbf{y}_\mu(j_k)$;
  $\mathbf{y}_k := \mathbf{y}_k - \sum_{\mu<k} \mathbf{x}_\mu(i_k)\mathbf{y}_\mu$;
  set $\mathbf{x}_k := \frac{1}{\mathbf{x}_k(i_k)}\mathbf{x}_k$ or $\mathbf{y}_k := \frac{1}{\mathbf{y}_k(j_k)}\mathbf{y}_k$;
  $X = [X, \mathbf{x}_k]$, $Y = [Y, \mathbf{y}_k]$;
  $\epsilon := \|\mathbf{x}_k\|\|\mathbf{y}_k\|/(\|\mathbf{x}_0\|\|\mathbf{y}_0\|)$;
  $k := k + 1$
end

Implement algorithm 1 on the matrices provided in `matrices.zip`. Compare the obtained rank to the numerical rank obtained using

the SVD. What do you observe? Can you explain this?

2 ACA-CUR

You have seen that, at least theoretically, if the ACA algorithm selects pivots $\{i_1, \ldots, i_k\}$ and $\{j_1, \ldots, j_k\}$, then

$$B(\mathbf{t}, \mathbf{s}) \approx B_r = XY^T = B(:, \mathbf{s})B(\mathbf{t}, \mathbf{s})^{-1}B(\mathbf{t}, :).$$

Verify this claim numerically. Use for the inverse both the matlab function 'inv' and '\'. What do you observe? Explain.
We will outline here a way we can stably, on the fly, factorize $B(\mathbf{t}, \mathbf{s})^{-1}$. To do so, we look first at theorem 1

Theorem 1. *Suppose algorithm 1 is used to factor the block $B(\mathbf{t}, \mathbf{s})$, and the pivots selected during its run are $(\mathbf{t}_r, \mathbf{s}_r) = ([i_1, \ldots, i_r], [j_1, \ldots, j_r])$. We adopt the convention that in line 7 $\mathbf{x}_k$ is set to $\mathbf{x}_k/d_k$ or $\mathbf{y}_k$ is set to $\mathbf{y}_k/d_k$, and let $\mathbf{t}_k, \mathbf{s}_k$ correspondingly denote the index sets selected up to step k of the algorithm. Then*

$$B_r = X_r Y_r^T = B(:, \mathbf{s}_r)U_{\mathbf{s}_r} D_r U_{\mathbf{t}_r}^T B(\mathbf{t}_r, :) \tag{1}$$

with $D_r := diag(d_1, \ldots, d_r)^{-1}$ and $U_{\mathbf{t}_r}, U_{\mathbf{s}_r} \in \mathbb{C}^{r \times r}$ upper triangular matrices defined by

$$U_{j_1} = 1, \quad U_{i_1} = 1$$

and

$$U_{\mathbf{s}_{k+1}} = \begin{bmatrix} U_{\mathbf{s}_k} & -U_{\mathbf{s}_k} D_k U_{\mathbf{t}_k}^T B(\mathbf{t}_k, :)\mathbf{e}_{j_{k+1}} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2}$$

and similarly for $U_{\mathbf{t}_{k+1}}$.

You have been provided a 'naive' implementation of this idea. **Compare this to algorithm 2. What is different?**
Implement algorithm 2. Compare the numerical accuracy of the naive implementation to the accuracy of algorithm 2, over decreasing user-supplied tolerance. What do you observe?

Algorithm 2: ACA-CUR

input : Matrix $B(\mathbf{t}, \mathbf{s}) \in \mathbb{C}^{\mathbf{t}\times\mathbf{s}}$, tolerance **tol**

output: Low-rank CUR approx. $B(\mathbf{t}, \mathbf{s}) \approx B(:, \mathbf{s}_r)U_{\mathbf{s}_r}D_rU_{\mathbf{t}_r}^TB(\mathbf{t}_r, :)$

init $\epsilon := \infty$, $k = 0$, $\mathbf{t}_0 = \mathbf{s}_0 = \emptyset$, $U_{\mathbf{t}_0} = U_{bt_0} = D_0 = [\,]$

while $\epsilon >$ ***tol*** **do**

  select pivot $(i_k, j_k) \in \mathbf{t} \times \mathbf{s}$;

  $d_k = B(i_k, j_k) - (B(i_k, \mathbf{s}_k)U_{\mathbf{s}_k})D_k(U_{\mathbf{t}_k}^TB(\mathbf{t}_k, j_k))$;

  $\mathbf{b}_k = D_k \cdot (U_{\mathbf{t}_k}^TB(\mathbf{t}_k, j_k))$;

  $\mathbf{a}_k^T = D_k \cdot (B(i_k, \mathbf{s}_k)U_{\mathbf{s}_k})$;

  $U_{\mathbf{s}_{k+1}} = \begin{bmatrix} U_{\mathbf{s}_k} & -U_{\mathbf{s}_k}\mathbf{b}_k \\ \mathbf{0}^T & 1 \end{bmatrix}$;

  $U_{\mathbf{t}_{k+1}} = \begin{bmatrix} U_{\mathbf{t}_k} & -U_{\mathbf{t}_k}^T\mathbf{a}_k \\ \mathbf{0}^T & 1 \end{bmatrix}$;

  $D_{k+1} = \begin{bmatrix} D_k & \\ & 1/d_k \end{bmatrix}$;

  $\mathbf{t}_{k+1} = \mathbf{t}_k \cup i_k, \mathbf{s}_{k+1} = \mathbf{s}_k \cup j_k$;

  $\mathbf{x}_k = B(:, \mathbf{s}_k)U_{\mathbf{s}_k}, \mathbf{y}_k^T = U_{\mathbf{t}_k}^TB(\mathbf{t}_k, :)$;

  $\epsilon := \|\mathbf{x}_k\|\|\mathbf{y}_k\|/(\|\mathbf{x}_0\|\|\mathbf{y}_0\|)$;

  $k := k + 1$;

end

3 Hierarchical matrices for the Helmholtz equation

The Helmholtz equation is an important equation in mathematics and physics. It models monochromatic wave scattering, both acoustic and electro-magnetic. We consider in this section the Helmholtz equation in an open domain $\Omega \subseteq \mathbb{R}^3$ with (weakly) Lipschitz boundary $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$, on which repectively Dirichlet and Neumann boundary conditions are imposed:

$$\begin{cases} \nabla^2 u + \kappa^2 u &= 0, \quad \text{in } \Omega \\ u &= g_D, \text{ on } \partial\Omega_D \\ \frac{\partial u}{\partial \mathbf{n}} &= g_N, \text{ on } \partial\Omega_N \end{cases} \tag{3}$$

and we require u to satisfy the *Sommerfeld radiation condition*, i.e.

$$\lim_{\mathbf{r}\to\infty} \mathbf{r}\cdot\left(\frac{\partial u}{\partial \mathbf{r}} - \imath\kappa u\right) = 0. \tag{4}$$

Using the method of Green's functions it can be shown that, for $\mathbf{x}\in\Omega$

$$u(\mathbf{x}) = -\underbrace{\int_{\partial\Omega_D}[\gamma_1 u]G(\mathbf{x},\mathbf{y})d\mathbf{y}}_{:=\mathrm{SLP}([\gamma_1 u])(\mathbf{x})} + \underbrace{\int_{\partial\Omega_N}[\gamma_0 u]\frac{\partial}{\partial \mathbf{n_y}}G(\mathbf{x},\mathbf{y})d\mathbf{y}}_{:=\mathrm{DLP}([\gamma_0 u])(\mathbf{x})} \tag{5}$$

in which γ_0, γ_1 denote the trace and conormal derivative respectively, $[\,.\,]$ denotes the jump across the boundary, and

$$G(\mathbf{x},\mathbf{y}) := \frac{\exp\left(\imath\kappa\|\mathbf{x}-\mathbf{y}\|\right)}{4\pi\|\mathbf{x}-\mathbf{y}\|}$$

is the *Green's kernel* for the 3D Helmholtz equation. Setting $[\gamma_1 u] = \varphi$ and $[\gamma_0 u] = \psi$, and taking the trace and conormal derivative of equation (5) we obtain, on $\partial\Omega$:

$$g_D(\mathbf{x}) = -\underbrace{\int_{\partial\Omega_D} G(\mathbf{x},\mathbf{y})\varphi(\mathbf{y})d\mathbf{y}}_{(\mathcal{S}\varphi)(\mathbf{x})} \pm\frac{1}{2}\psi(\mathbf{x}) + \underbrace{\int_{\partial\Omega_N}\frac{\partial}{\partial \mathbf{n_y}}G(\mathbf{x},\mathbf{y})\psi(\mathbf{y})d\mathbf{y}}_{(\mathcal{D}\psi)(\mathbf{x})}$$

$$g_N(\mathbf{x}) = \pm\frac{1}{2}\varphi(\mathbf{x}) - \underbrace{\int_{\partial\Omega_D}\frac{\partial}{\partial \mathbf{n_x}}G(\mathbf{x},\mathbf{y})\varphi(\mathbf{y})d\mathbf{y}}_{(\mathcal{D}'\varphi)(\mathbf{x})} + \underbrace{\gamma_1\int_{\partial\Omega_N}\frac{\partial}{\partial \mathbf{n_y}}G(\mathbf{x},\mathbf{y})\psi(\mathbf{y})d\mathbf{y}}_{-(\mathcal{W}\psi)(\mathbf{x})}$$

in which the 4 main boundary integral operators (BIOs) for the Helmholtz equation have been introduced: $\mathcal{S}, \mathcal{D}, \mathcal{D}', \mathcal{W}$, respectively the single-layer, double-layer, adjoint double-layer and hypersingular boundary operators. **You do not need to understand this fully, this is just some background for the curious.**
In particular, sound-soft scattering is modelled by setting

$$\mathcal{S}u_{sc} = -u_{in}$$

where u_{sc} is the scattered field and u_{in} is the incoming acoustic pressure field, typically a plane wave.

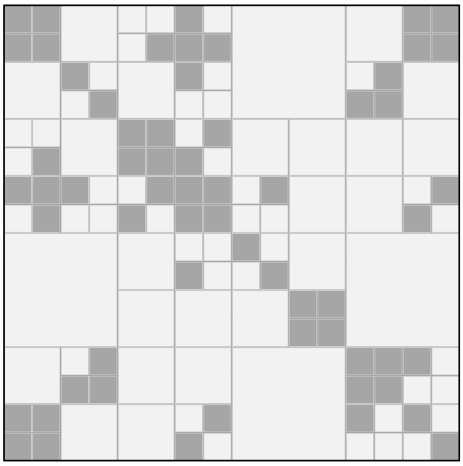

Figure 1: Illustration of the block structure of an $\mathcal{H}$-matrix. dark gray blocks correspond to non-admissible cluster-cluster interactions.

The integral kernel G is separable at long distances, meaning that upon Galerkin discretization of the operator $\mathcal{S}$, the blocks corresponding to well-separated clusters are low-rank. This gives rise to a so-called *Hierarchical Matrix* ($\mathcal{H}$-matrix). Large, off-diagonal blocks are low rank, while the diagonal blocks, and small off-diagonal blocks are dense, as in figure 1. Often, the low-rank blocks are compressed using ACA. **You have been provided code that computes an $\mathcal{H}$-matrix approximation to S. Look at the code provided. Re-write your ACA scheme to be of the form**

```
[A,B,flag]=aca(row,col,n,m,tol)
```

Where '`flag`' is a boolean that is 1 if the low-rank compression failed, and '`row`' and '`col`' are functions that, given i or j, return the ith row or the jth column, as (column) vectors. Run the $\mathcal{H}$-matrix compression scheme for some wavenumbers varying from $.1\kappa_{\max}$ to $.9\kappa_{\max}$. Look at the console output and compute the data sparsity for the far-field. What do you observe?