Numerical approximation of the Gradient

Let us consider a $N \times N$ matrix $\underline{\mathcal{C}}$ defined as

$$\underline{\mathcal{C}} = \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & -2 & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{pmatrix}$$

The aim here is to differentiate analytically and/or numerically the functional defined for $\boldsymbol{X} \in \mathbb{R}^N$ as

$$\mathcal{J}(\boldsymbol{X}) = \frac{1}{2}\boldsymbol{X} \cdot \Big[\mathcal{A}(\boldsymbol{X})\boldsymbol{X}\Big] + \boldsymbol{B}(\boldsymbol{X}) \cdot \boldsymbol{X} \quad \text{where} \quad \mathcal{A}(\boldsymbol{X}) = \underline{\Lambda}(\boldsymbol{X})\underline{\mathcal{C}} \in \mathbb{R}^N \times \mathbb{R}^N$$

We will investigate this differentiation for different formulations of $\underline{\Lambda}(\boldsymbol{X}) \in \mathbb{R}^N \times \mathbb{R}^N$ and $\boldsymbol{B}(\boldsymbol{X}) \in \mathbb{R}^N$ and for differents size of the problem : $N = 10, 100, 500, 1000, 10000, \cdots, 1000000$.

1) Constant $\underline{\Lambda}(\boldsymbol{X})$ and $\boldsymbol{B}(\boldsymbol{X})$: In this case $\underline{\Lambda}(\boldsymbol{X}) \equiv \underline{\Lambda}_0 \in \mathbb{R}^N \times \mathbb{R}^N$ and $\boldsymbol{B}(\boldsymbol{X}) \equiv \boldsymbol{B}_0 \in \mathbb{R}^N$. Use for example :

$$\underline{\Lambda}_0 = (N+1)\, Id \quad \text{and} \quad \boldsymbol{B}_0 = \Big[0, \cdots, 0,\ (N+1)\Big]^T$$

For $k = 1 \cdots N$, $\quad \varepsilon = 10^{-1},\ 10^{-2},\ 10^{-4},\ 10^{-8}$

- Compute numerically (for example $\boldsymbol{u} \equiv \frac{1}{N+1}$)

$$\mathcal{J}(\boldsymbol{u} + \varepsilon \boldsymbol{e}_k) \quad \text{and} \quad d_\varepsilon^k \mathcal{J}(\boldsymbol{u}) = \mathcal{J}(\boldsymbol{u} + \varepsilon \boldsymbol{e}_k) - \mathcal{J}(\boldsymbol{u})$$

- Compute numerically (for example $\boldsymbol{u} \equiv \frac{1}{N+1}$)

$$\frac{d_\varepsilon^k \mathcal{J}(\boldsymbol{u})}{\varepsilon} \quad \text{where} \quad (\boldsymbol{e}_k)_j = \delta_{jk}$$

- Verify that $\Big[\nabla_\varepsilon \mathcal{J}(\boldsymbol{u}) \cdot \boldsymbol{e}_k\Big]_{\varepsilon=10^{-8}} = \left(\frac{d_\varepsilon^k \mathcal{J}}{\varepsilon}\right)_{\varepsilon=10^{-8}} \simeq \lim\limits_{\varepsilon \to 0} \left(\frac{d_\varepsilon^k \mathcal{J}}{\varepsilon}\right) = \nabla \mathcal{J}(\boldsymbol{u}) \cdot \boldsymbol{e}_k.$
- Compute the approximated norm $\|\nabla_\varepsilon \mathcal{J}(\boldsymbol{u})\|$.
- Compute the approximated norm $\|\nabla_\varepsilon \mathcal{J}(\boldsymbol{u})\|$ when $\boldsymbol{u}_j = \dfrac{j}{N+1}$, for $j = 1 \cdots N$.

2) Linear $\underline{\Lambda}(\boldsymbol{X})$ and $\boldsymbol{B}(\boldsymbol{X})$: In this case, use for example :

$$\Big[\underline{\Lambda}(\boldsymbol{X})\Big]_{ij} = \alpha \boldsymbol{X}_i \delta_{ij} + (N+1)\,\delta_{ij} \quad \text{and} \quad \boldsymbol{B} = \beta \boldsymbol{X} + \boldsymbol{B}_0$$

where α and β are constants, in practice takes it small.

- Compute a numerical approximation $\nabla_\varepsilon \mathcal{J}(\boldsymbol{u})$ of the gradient, for $\varepsilon = 10^{-1},\ 10^{-2},\ 10^{-4},\ 10^{-8}$ when $\boldsymbol{u} \equiv 1/(N+1)$ and when $\boldsymbol{u}_j = \dfrac{j}{N+1}$.

3) Memory Consuming : For $N = 10^6$ how many non zero coeficients in the matrix $\mathcal{A}(\boldsymbol{X})$ compare it to $N \times N$. What can we do to reduce the memory. In this context, propose a memory optimized progammation.