Question 1: Quadratic Form

Parallellize with OpenMP the computation of the quadratic form

$$Q = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} v_i A_{ij} w_j \tag{1}$$

for a matrix $A \in \mathbb{R}^{N\times N}$ and two vectors $v, w \in \mathbb{R}^N$.

Question 2: GEneral Matrix Multiply (GEMM)

You are given a skeleton code that performs a naive general matrix multiplication $C = AB$ for matrices $A, B, C \in \mathbb{R}^{N\times N}$. Parallelize the computation with OpenMP. Then, perform the same computation with the BLAS library by calling the function *cblas_dgemm* and compare the performance of your implementation with it. On Euler, you will need to *module load gcc openblas.*

Question 3: Poisson Equation and Jacodi Method

We are interested in solving the Poisson equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) \tag{2}$$

for a scalar quantity u, in the rectangle $[0, L_x] \times [0, L_y]$ with homegeous Dirichlet boundary conditions

$$u(x, L_y, t) = u(x, 0, t) = u(0, y, t) = u(L_x, y, t) = 0, \tag{3}$$

and the following right-hand side

$$f(x, y) = \exp\left[-(x - L_x/2)^2 - (y - L_y/2)^2\right]. \tag{4}$$

To do so, we employ a grid of $N_x \times N_y$ points. For $i = 0, \dots, N_x - 1$ and $j = 0, \dots, N_y - 1$, we define

$$u_{i,j} = u\left(i\Delta x, j\Delta y\right) \quad , \quad f_{i,j} = f\left(i\Delta x, j\Delta y\right), \tag{5}$$

where $\Delta x = \frac{L_x}{N_x - 1}$ and $\Delta x = \frac{L_y}{N_y - 1}$ are the grid spacings in the x and y directions. Using this definition, we proceed to discretize the partial derivatives in eq. (2) as follows

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} \ , \ \frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}$$

resulting in the following discrete form

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} = f_{i,j} \tag{6}$$

which is valid for the inner points $i = 1, \dots, N_x - 2$ and $j = 1, \dots, N_y - 2$. For the points on the boundary, we set

$$u_{0,j} = u_{N_x-1,j} = u_{i,0} = u_{i,N_y} = 0 \quad , \quad \forall i = 0, \dots, N_x - 1 \quad , \quad \forall j = 0, \dots, N_y - 1. \tag{7}$$

Equation (6) corresponds to a linear system of $(N_x - 2) \times (N_y - 2)$ equations with an equal number of unknowns. We will solve this system with the iterative Jacobi method. First, we rewrite eq. (6) as

$$u_{i,j}^{m+1} = \frac{1}{2/\Delta x^2 + 2/\Delta y^2} \left[\frac{u_{i+1,j}^m + u_{i-1,j}^m}{\Delta x^2} + \frac{u_{i,j+1}^m + u_{i,j-1}^m}{\Delta y^2} - f_{i,j} \right] \tag{8}$$

where we simply solved for $u_{i,j}$ and added a superscript m. The iterative Jacobi method performs iterations with eq. (8) for $m = 0, 1, \dots$ until

$$E_m = \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} |u_{i,j}^{m+1} - u_{i,j}^m| < \epsilon \tag{9}$$

where $\epsilon = 10^{-6}$ is a user-defined threshold. Note that we set $u_{i,j}^0 = 0$.

You are given a skeleton code that solves the Poisson equation with the method described above. Parallelize the code with OpenMP.