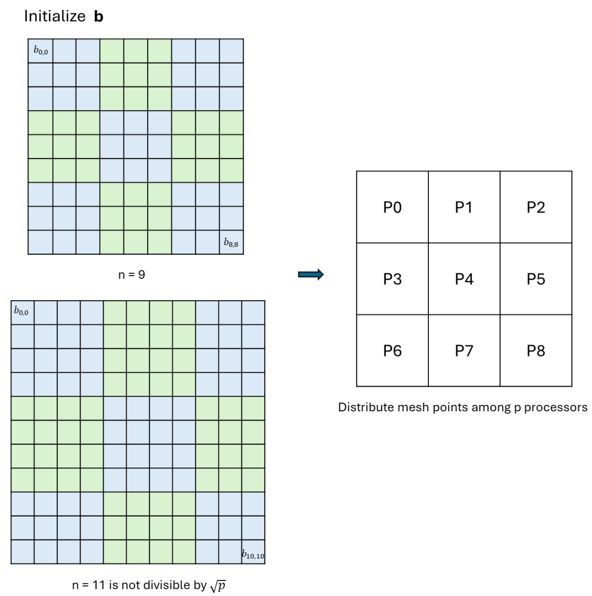
Parallel Conjugate Gradient Method
with Stencil-Based Matrix-Vector Multiplication

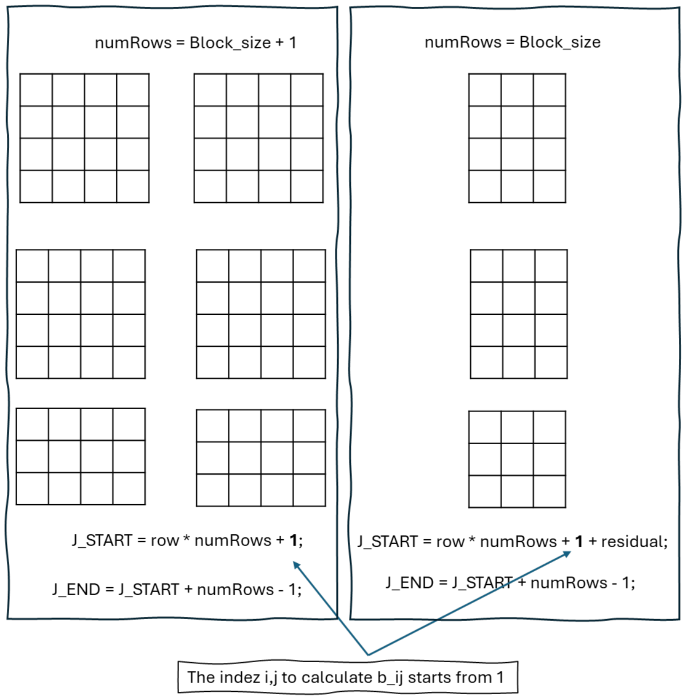
Algorithm The Conjugate Gradient (CG) Algorithm

Define A and b based on the input parameter n
Define maximum iterations $imax$ and tolerance error $rmax$
Initialize $u = 0$, $g = -b$, $d = b$
$q_0 = g^T g$
for $it = 1, 2, \ldots$ until $it == imax$ or $|b - Au| < rmax$ **do**
    $q = Ad$
    $\tau = q_0/(d^T q)$ ▷ Step length
    $u = u + \tau d$ ▷ Approximate solution
    $g = g + \tau q$ ▷ New residual
    $q_1 = g^T g$
    $\beta = q_1/q_0$ ▷ Improvement
    $d = -g + \beta d$ ▷ New search direction
    $q_0 = q_1$
end for
return u, the approximation solution



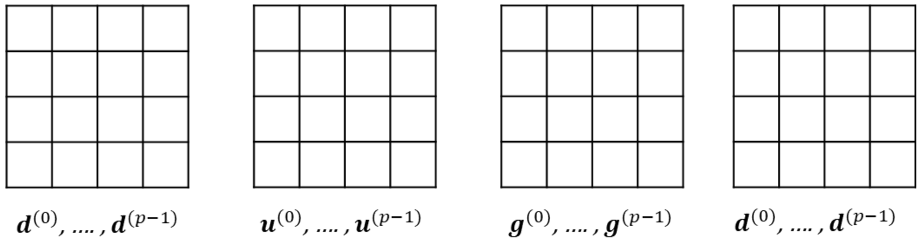
Initialization

block_size = n / $\sqrt{p}$
residual = n % $\sqrt{p}$
If row < residual:
get an extra row

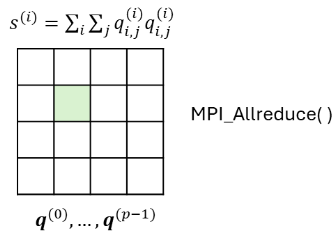



```
double x, y;
    for (int i = 0; i < numRows; i++) {
        for (int j = 0; j < numCols; j++) {
            x = (I_START + i) * h;
            y = (J_START + j) * h;
            b[i][j] = 2 * h * h * (x * (1-x) + y * (1-y));
        }
    }
```

Initialize **u** = **0**, **g** = −**b**, **d** = **b**, and $q_0 = \boldsymbol{g}^T\boldsymbol{g}$

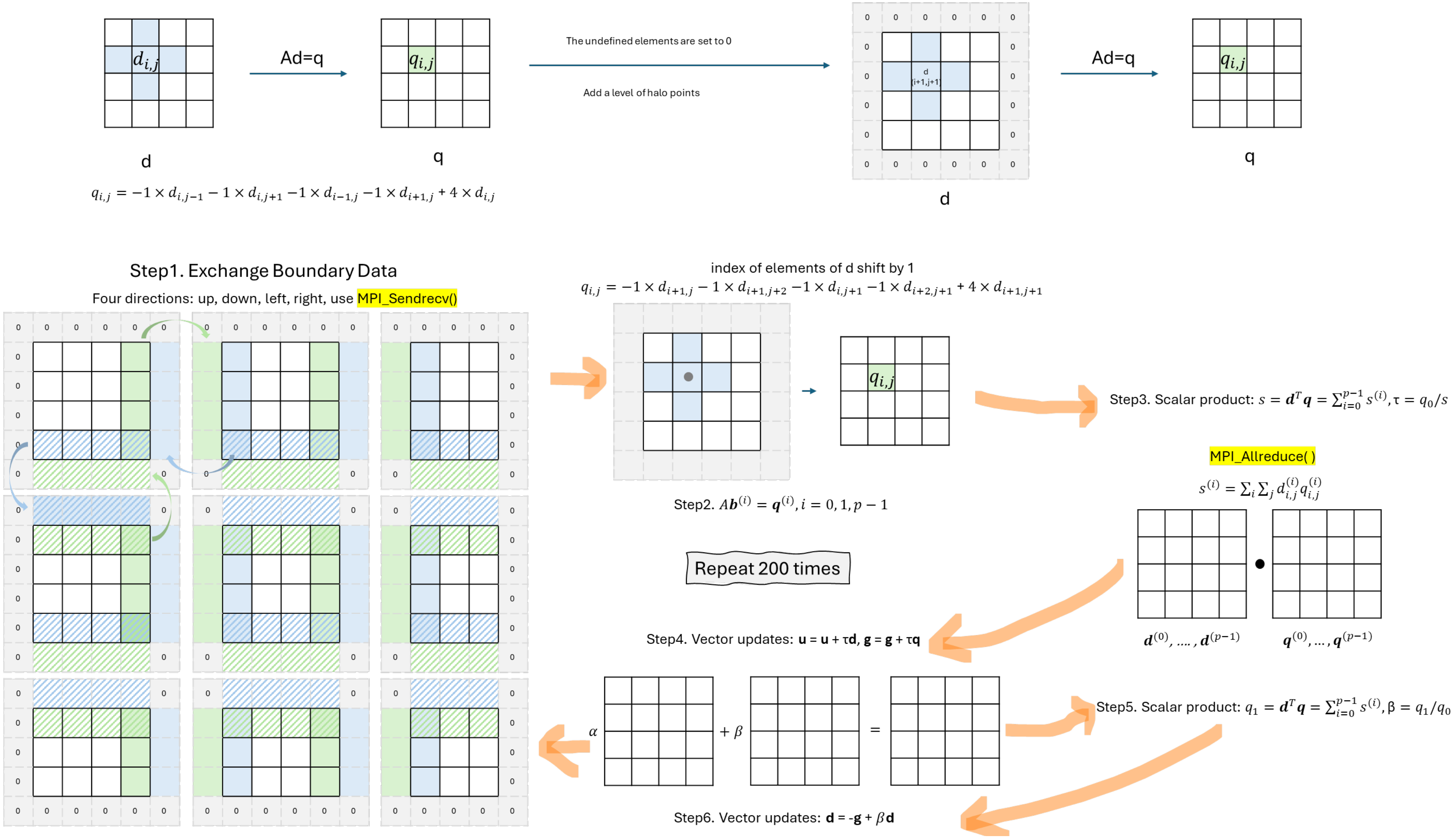


These vectors has the same shapes on each process, since all the basis operations are element-wise operation.

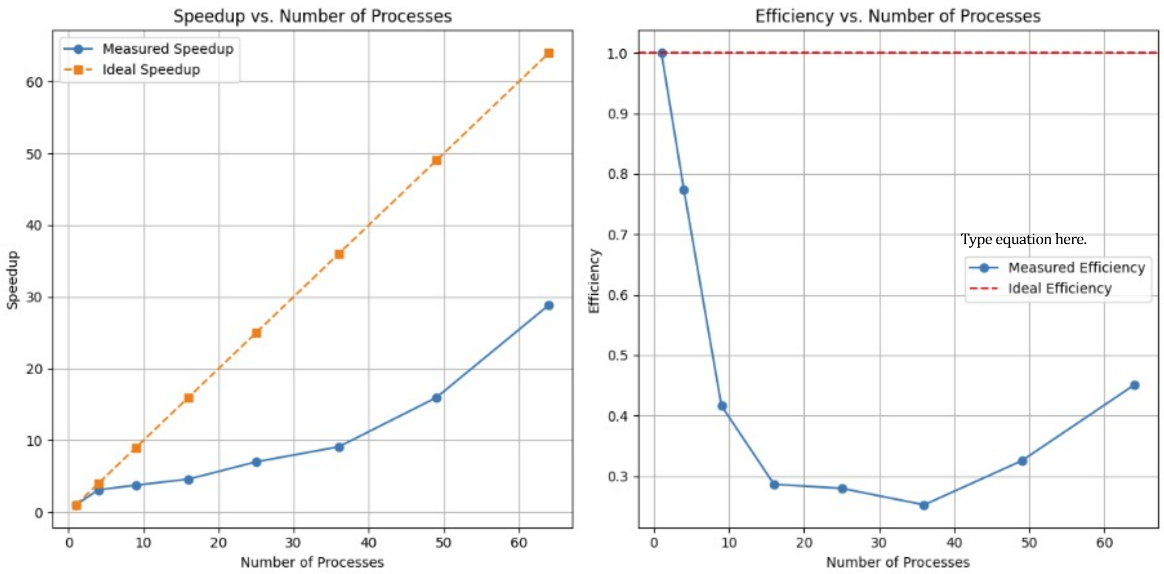


Scalar product: $q_0 = \boldsymbol{q}^T\boldsymbol{q} = \sum_{i=0}^{p-1} s^{(i)}$

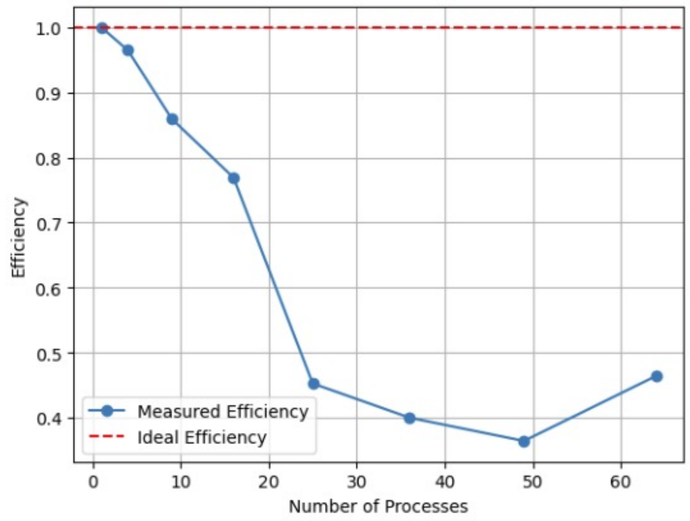
Iteration Process



Scalability Experiments



Strong Scalibility, n = 2048



Weak Scalibility, n = 256 per process

Outputs

n	residual ($\|g\|_2$)
256	3.82×10^{-5}
512	4.43×10^{-3}
768	6.34×10^{-3}
1024	7.28×10^{-3}
1280	7.62×10^{-3}
1536	7.90×10^{-3}
1972	8.14×10^{-3}
2048	8.25×10^{-3}

O(n^2)

n = 1024, itr = 722, r = 9.75e-6