

# HIGH-PERFORMANCE COMPUTING IN APPLIED MATHEMATICS

MTH 410/510

# Numerical solution to Poisson's equation

One of the most important applications of the iterative methods for linear systems is obtain the numerical solutions of partial differential equations that are discretized using finite differences or finite element methods.

To motivate our study, consider the 2D Poisson's equation on a square: the goal is to find a function

$$u(x, y), \quad 0 \leq x \leq 1, \quad 0 \leq y \leq 1$$

that satisfies the partial differential equation

$$u_{xx}(x, y) + u_{yy}(x, y) = f(x, y), \quad 0 < x, y < 1 \quad (1)$$

with boundary conditions

$$u(x, y) = g(x, y), \quad (x, y) \text{ boundary point}$$

The functions  $f(x, y)$  and  $g(x, y)$  are given, we must find  $u(x, y)$ .

For  $n > 1$  we define  $h = 1/n$  and the mesh

$$(x_j, y_k) = (j * h, k * h), \quad 0 \leq j, k \leq n$$

These are called the *grid points* or *mesh points*. A discrete problem is obtained by approximating the second order derivatives using the central difference formula:

$$G''(x) \approx [G(x + h) - 2G(x) + G(x - h)]/h^2 \quad (2)$$

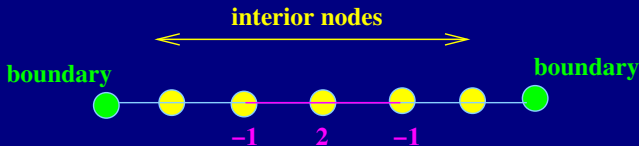
which is a second order accurate approximation (error of order  $h^2$ ). When (??) is used to approximate  $u_{xx}$  and  $u_{yy}$  at each *interior* grid point we obtain a linear system of equations of dimension  $(n - 1)^2$

$$4u_{j,k} - u_{j+1,k} - u_{j-1,k} - u_{j,k-1} - u_{j,k+1} = -h^2 f_{j,k}, \quad 1 \leq j, k \leq n - 1 \quad (3)$$

# Poisson's equation in 1D

The matrix of the linear system and the stencil of the central difference discretization in 1D are

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix},$$





# Iterative solution to linear systems

Consider the linear system

$$\mathbf{Ax} = \mathbf{b} \quad (4)$$

where  $\mathbf{A} \in \mathbb{R}^{n \times n}$  is a given matrix and  $\mathbf{b} \in \mathbb{R}^n$  is a given vector.

In many cases solving the system using direct methods (e.g., Gaussian elimination) may be too expensive since the order of computations required is  $n^3$ .

Iterative methods provide approximate solutions by performing a number of iterations, each of computational expense  $n^2$ .

The general framework of the iterative methods for solving linear systems is closely related to the fixed point theory.

- split the matrix  $\mathbf{A}$  into

$$\mathbf{A} = \mathbf{N} - \mathbf{P}$$

- write eq. (??) as

$$\mathbf{Nx} = \mathbf{b} + \mathbf{Px}$$

- define the iteration method

$$\mathbf{Nx}^{k+1} = \mathbf{b} + \mathbf{Px}^k$$

# Jacobi's method

In order to begin the iteration, an initial guess  $\mathbf{x}^0$  must be provided. The matrix  $\mathbf{N}$  must be selected such that the system

$$\mathbf{N}\mathbf{x}^{k+1} = \mathbf{b} + \mathbf{P}\mathbf{x}^k$$

is "easily" solvable in order to keep the cost per iteration low. In particular, *Jacobi's method* is obtained by taking

$$\mathbf{N} = \text{diag} [a_{11}, a_{22}, \dots, a_{nn}]$$

Componentwise, Jacobi's iteration is written:

$$x_i^{k+1} = \frac{1}{a_{ii}} \left( b_i - \sum_{j \neq i} a_{ij} x_j^k \right), \quad i = 1, 2, \dots, n \quad (5)$$

The iteration converges from any initial guess  $\mathbf{x}^0$  if the operator

$$\mathbf{x} \rightarrow \mathbf{N}^{-1}\mathbf{P}\mathbf{x}$$

is a "contraction", i.e.,

$$\|\mathbf{N}^{-1}\mathbf{P}\| < 1$$

for some matrix norm. In this case

$$\mathbf{b} = \mathbf{A}\mathbf{x} = (\mathbf{N} - \mathbf{P})\mathbf{x}$$

implies

$$\mathbf{x}^{k+1} - \mathbf{x} = \mathbf{N}^{-1}\mathbf{P}(\mathbf{x}^k - \mathbf{x})$$

such that

$$\|\mathbf{x}^{k+1} - \mathbf{x}\| \leq \|\mathbf{N}^{-1}\mathbf{P}\| \|\mathbf{x}^k - \mathbf{x}\| < \|\mathbf{x}^k - \mathbf{x}\|$$

Therefore,  $\mathbf{x}^{k+1}$  is "closer" to the solution than  $\mathbf{x}^k$ .

Notice that if  $\mathbf{A}$  is strictly diagonally dominant which means

$$|a_{ii}| > \sum_{j \neq i} |a_{i,j}|, \quad i = 1, 2, \dots, n$$

then  $\mathbf{N}^{-1}\mathbf{P}$  in the Jacobi's method is contractive in the "infinity norm" defined for a general matrix  $\mathbf{C}$

$$\|\mathbf{C}\|_{\infty} = \max_{1 \leq i \leq n} \sum_{j=1}^n |c_{ij}| \quad (6)$$

For such matrices, Jacobi's method converges from any initial guess  $\mathbf{x}^0$

# Serial Jacobi's algorithm

Input: initial guess  $\mathbf{x}_0$ , tolerance  $\epsilon$

$enorm = \epsilon + 1$

do while  $enorm > \epsilon$

$\mathbf{x}_1 = \text{Jacobi}(\mathbf{x}_0)$

$enorm = \max\{|x_1(i) - x_0(i)|, i = 1, 2, \dots, n\}$

$\mathbf{x}_0 = \mathbf{x}_1$

end do

---

$\mathbf{x}_1 = \text{Jacobi}(\mathbf{x}_0)$

do  $i = 1, n$

$x_1(i) = b(i)$

    for  $j = 1 : i - 1$

$x_1(i) = x_1(i) - a(i, j) * x_0(j)$

    end do

    do  $j = i + 1, n$

$x_1(i) = x_1(i) - a(i, j) * x_0(j)$

    end do

$x_1(i) = x_1(i) / a(i, i)$

end

# Parallel Jacobi's algorithm

Assume that the matrix  $\mathbf{A}$  is block-row distributed among processes, each process has  $loc\_n = n/p$  rows of  $\mathbf{A}$ .

Assume that  $\mathbf{b}$  is block distributed among processes.

Each process defines

$$n1 = my\_rank * loc\_n + 1; \quad n2 = my\_rank * loc\_n + loc\_n$$

and updates a segment of the vector  $x$

$$loc\_x(1 : loc\_n) = x(n1 : n2)$$

Local Jacobi update:

do  $i = 1, loc\_n$

$$k = n1 + i - 1$$

$$loc\_x(i) = \frac{1}{LOC\_A(i,i)} \left( loc\_b(i) - \sum_{j \neq k} LOC\_A(i,j)x(j) \right)$$

end do

Local error estimate:

$$\text{myerr} = \max\{|loc\_x(i) - x(n1 + i - 1)|, i = 1, 2, \dots, loc\_n\}$$

Global error estimate:

`MPI_ALLREDUCE(myerr, enorm, 1, MPI_REAL, MPI_MAX, comm, ierr)`

Update `x` for the next iteration

`MPI_ALLGATHER(loc_x, loc_n, MPI_REAL, x, loc_n, MPI_REAL, comm,  
ierr)`

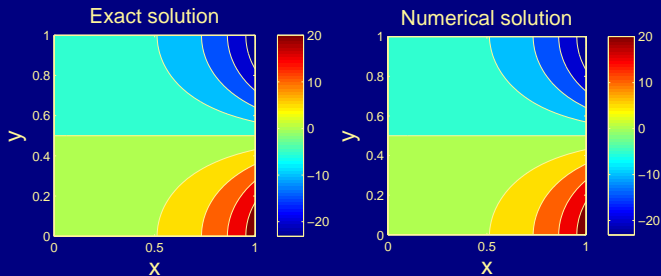
# Case study: 2D Poisson's problem

$$u_{xx} + u_{yy} = 0, \quad 0 \leq x, y \leq 1$$

$$u(0, y) = \cos(\pi y), \quad u(1, y) = e^{\pi} \cos(\pi y)$$

$$u(x, 0) = e^{\pi x}, \quad u(x, 1) = -e^{\pi x}$$

Exact solution:  $u(x, y) = e^{\pi x} \cos(\pi y)$



See [sjacobi.f](http://sjacobi.f), [pjacobi.f](http://pjacobi.f)

# Homework assignment

Consider the boundary value problem:

$$u''(x) = u'(x) + 2u(x) + \cos x, \quad 0 < x < \pi$$

$$u(0) = -0.3, \quad u(\pi) = 0.3$$

The exact solution is  $u(x) = -(\sin x + 3 \cos x)/10$ .

Implement Jacobi's iterative method to solve this problem using a central finite difference discretization of the first and second order derivatives.

PROVIDE:

- serial and parallel code implementation - 20 points (5s + 15p)
- number of iterations required for  $n = 1000$  ( $h = \pi/n$ ), a tolerance  $\epsilon = 10^{-6}$ , and initial guess from linear interpolation of the boundary values.
- Graph of the numerical solution, graph of the error in the numerical solution - 5 points