

IMT2112 - Algoritmos Paralelos en Computación Científica

Preacondicionamiento en paralelo

Elwin van 't Wout

17 de octubre de 2019



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

Facultad de Matemáticas • Escuela de Ingeniería

imc.uc.cl

Clase previa

- Resolver sistemas lineales con métodos iterativos paralelos y preacondicionados

Agenda

- ¿Como mejorar implementar el preacondicionamiento en paralelo?

Precondicionamiento en paralelo

Sección 6.7 del libro de Eijkhout

Métodos para resolver sistemas lineales

- Métodos directos son eficientes para
 - matrices densas
 - tamaños pequeños
 - solución de alta precisión
- Métodos iterativos son eficientes para
 - matrices ralas
 - tamaños grandes
 - flexibilidad en precisión

Métodos del subespacio de Krylov

- Gradientes Conjugados (*CG*)
 - limitado a matrices simétricas positivas definidas (*spd*)
 - algoritmo de recurrencia corta
- Residual mínimo generalizado (*GMRes*)
 - matrices generales
 - algoritmo de recurrencia larga

Eficiencia de métodos iterativos

- El número de iteraciones necesarias depende del espectro de la matriz
 - una medida simple de convergencia es el número de acondicionamiento
- Preacondicionamiento es resolver el sistema $M^{-1}Ax=M^{-1}b$ en lugar de $Ax=b$
 - elegir el preacondicionador M parecida a A y sencilla para resolver
- Paralelización es limitada al álgebra lineal en cada iteración
 - se tiene que paralelizar la solución del preacondicionador

Gradientes Conjugados para $K^{-1}Ax = K^{-1}b$

Compute $r^{(0)} = Ax^{(0)} - b$ for some initial guess $x^{(0)}$

for $i = 1, 2, \dots$

solve $Kz^{(i-1)} = r^{(i-1)}$

$\rho_{i-1} = r^{(i-1)T} z^{(i-1)}$

if $i = 1$

$p^{(1)} = z^{(0)}$

else

$\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$

$p^{(i)} = z^{(i-1)} + \beta_{i-1}p^{(i-1)}$

endif

$q^{(i)} = Ap^{(i)}$

$\delta_i = \rho_{i-1}/p^{(i)T} q^{(i)}$

$x^{(i)} = x^{(i-1)} - \delta_i p^{(i)}$

$r^{(i)} = r^{(i-1)} - \delta_i q^{(i)}$

 check convergence; continue if necessary

end

Precondicionamiento de Jacobi

- Elegir $M=D$ el diagonal de A
 - ‘precondicionamiento de Jacobi’
 - ‘escalamiento diagonal’
- Resolver el precondicionador en paralelo
 - resolver un sistema diagonal es conveniente para parallelizar
- La mejora en convergencia es limitada con este precondicionador simple

Precondicionamiento ILU en paralelo

Sección 6.7 del libro de Eijkhout

Preacondicionamiento ILU

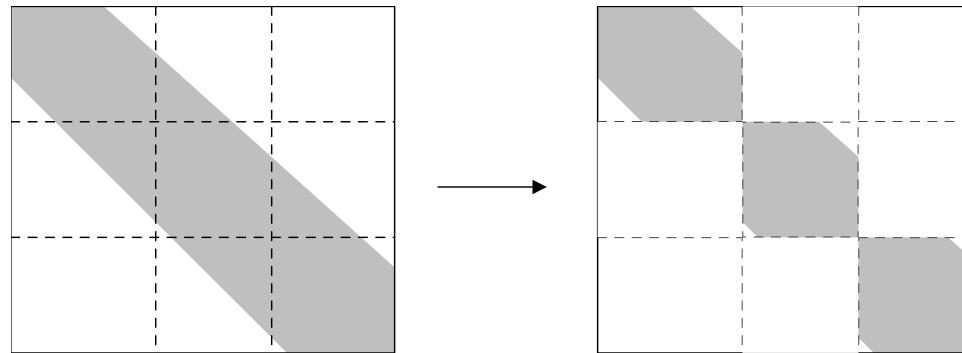
- Recuerden las afirmaciones siguientes
 - la descomposición LU es un preacondicionador perfecto
 - hay *fill-in* para matrices ralas
- La idea de ‘LU incompleta’ (ILU) es:
 - crear una descomposición tipo LU
 - sin usar los elementos de *fill-in*
- Hay versiones distintas de ILU
 - estándar: $L+U$ tiene el mismo *sparsity pattern* que A
 - se puede agregar más diagonales

Preacondicionamiento ILU

- Recuerden que la creación y solución de LU es recursiva
 - hay una iteración dependiente sobre el diagonal
- La creación y solución de ILU sigue el mismo algoritmo, salvo que se omite los elementos de *fill-in*
- La recursión en resolver $Lx=b$ con la matriz particionado por bloques significa que sólo un procesador es activo en un momento
 - un particionamiento cíclico, como en el caso de LU densa, no es eficiente para matrices ralas

Preacondicionamiento ILU por bloques

- Recuerden que el preacondicionador es siempre una aproximación
 - este permite hacer mas aproximaciones con el objetivo de paralelizar el algoritmo
 - siempre cuando las aproximaciones son acotadas
- ILU por bloques: hacer ILU localmente en cada bloque del diagonal



Preacondicionamiento ILU por bloques

- ILU por bloques
 - para matrices con banda pequeña la aproximación tiene un error pequeño
 - todo el algoritmo es independiente por procesador
- ILU por bloques no incluye la interacción entre las secciones del dominio
 - se espera una precisión peor en estos bordes artificiales

Estrategias de numeración para el paralelismo

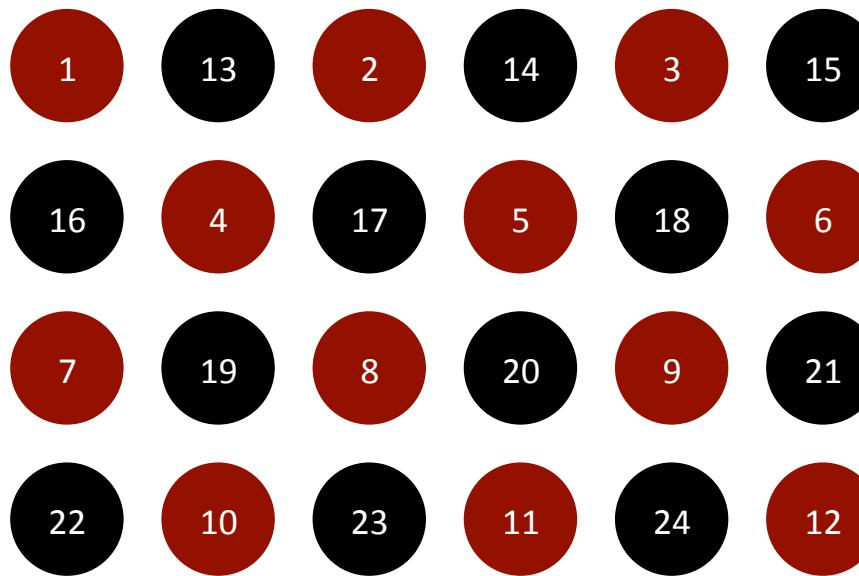
Sección 6.8 del libro de Eijkhout

Preacondicionamiento ILU por bloques

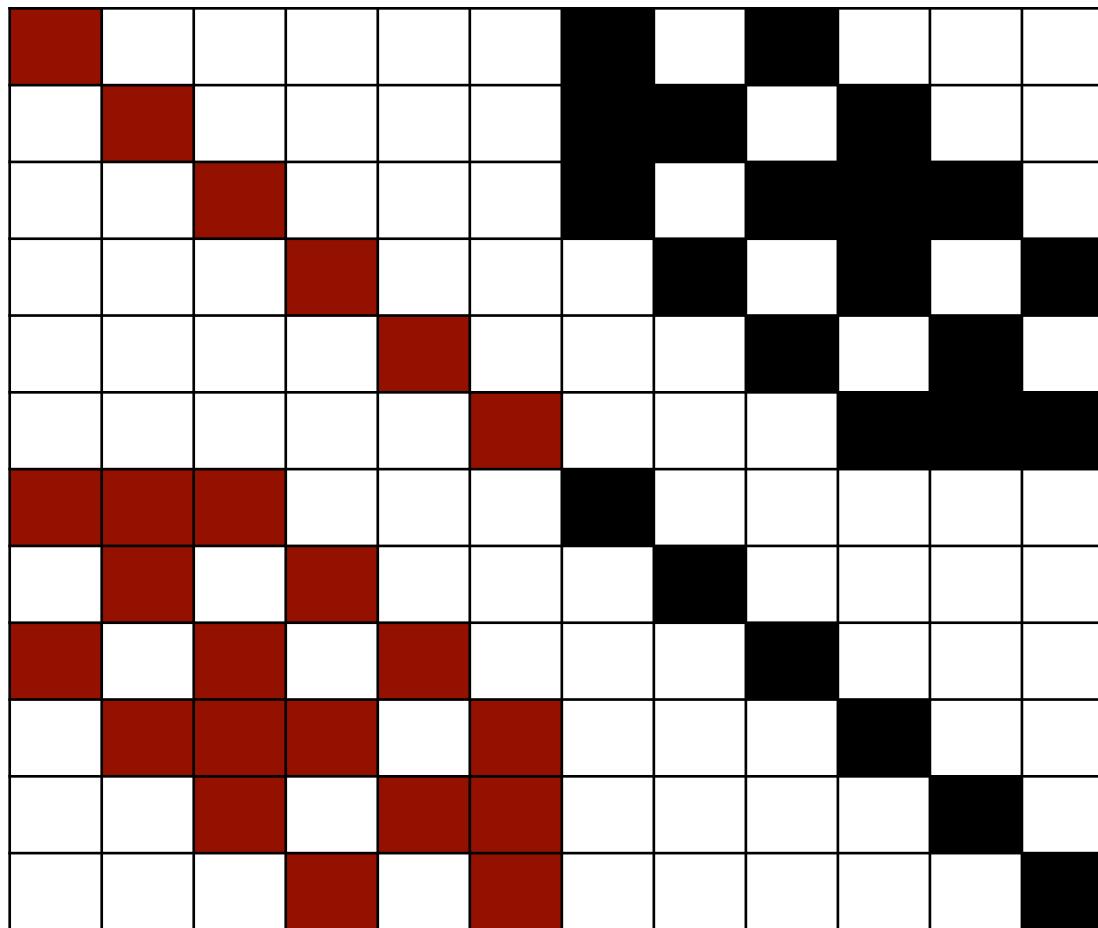
- Hemos visto que:
 - ILU es un preacondicionador eficiente
 - la paralelización requiere una aproximación adicional
- Recuerden que el *sparsity pattern* depende de la numeración de nodos en la malla
- Idea: usar otra estrategia de numeración para crear un ILU paralelo
 - el matvec sigue siendo lo mismo

Estrategia de numeración

- La numeración roja-negra
 - dividir la malla en un tipo de tablero de ajedrez
 - primero numerar los nodos rojos y después numerar los nodos negros
 - la forma (horizontal, vertical, etc.) no es importante



Estrategia de numeración



El *sparsity pattern* de una matriz con numeración roja-negra

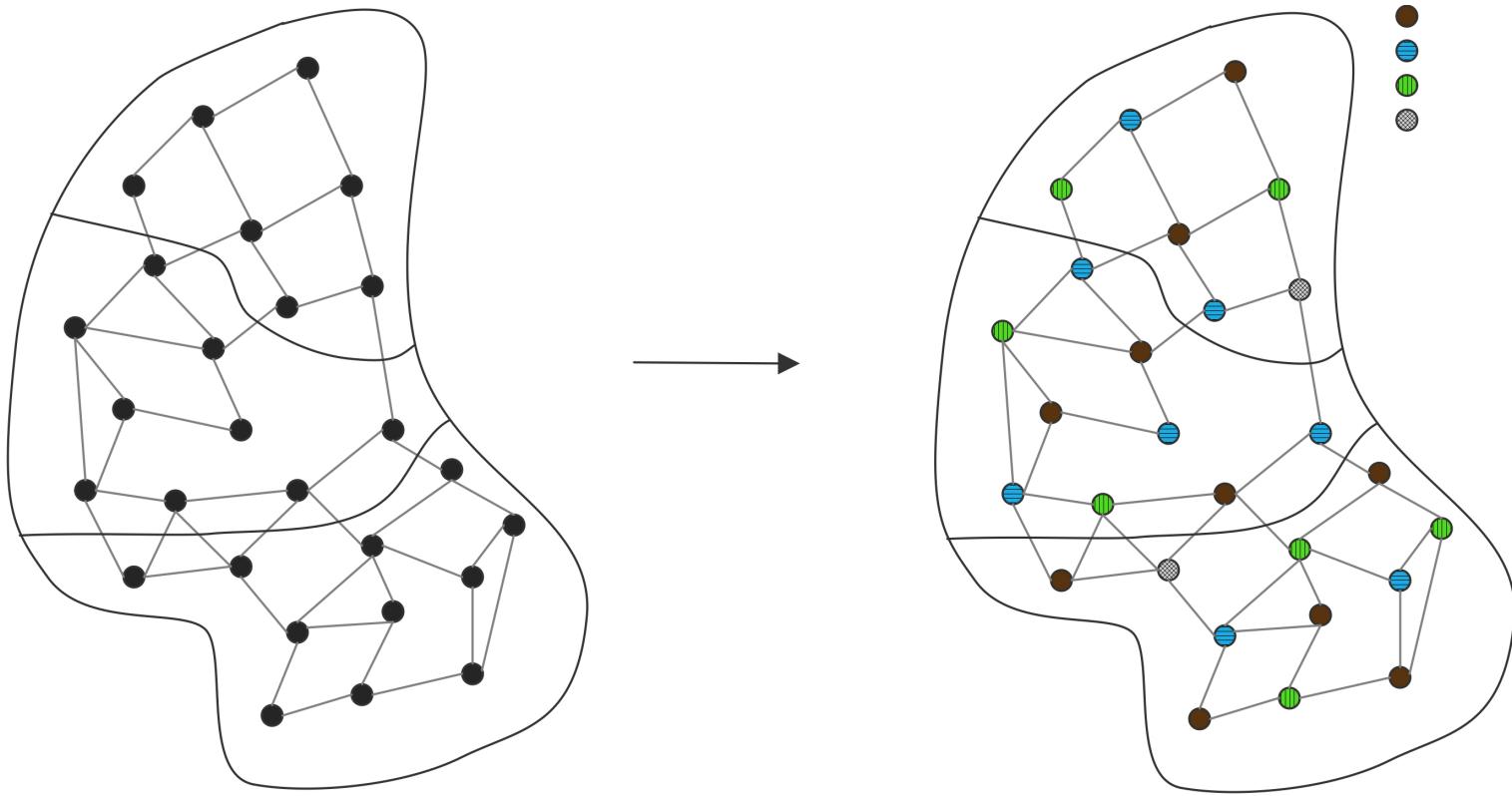
La numeración roja-negra

- Paralelizar $Lx=b$
 - primero, resolver los nodos rojos
 - resolver un bloque diagonal: paralelo y eficiente
 - segundo, resolver los nodos negros
 - una reducción local en los nodos rojos
 - cálculo paralelo de los nodos negros
- Recuerden que se partitiona el dominio, no la matriz
 - cada procesador tiene nodos rojos y negros
 - por lo tanto, hay un buen balanceo de carga

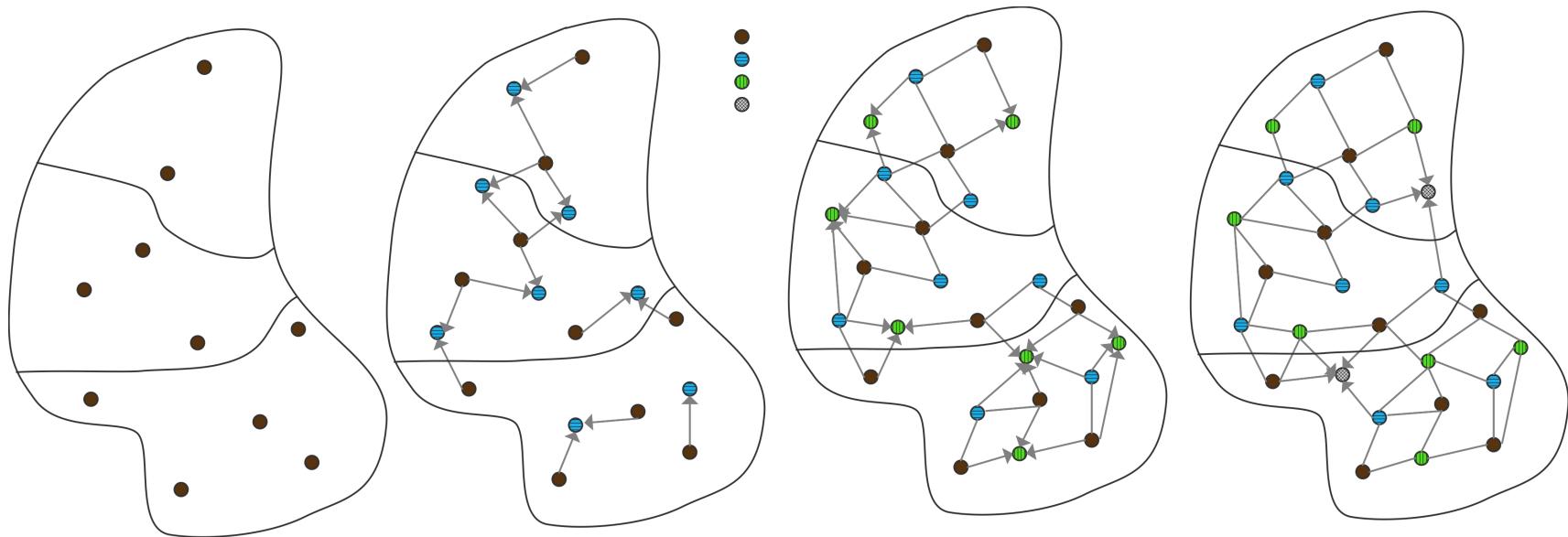
La numeración de colores

- La estrategia general de numeración con colores es crear una matriz que tiene un diagonal de bloques con matrices diagonales
 - se puede usar mas colores que dos
 - la estrategia depende del tipo del *stencil*
 - el centro del stencil tiene color distinto que los demás
- Se sigue particionando el dominio, no la matriz
- La numeración de colores solo se usa para crear el preacondicionador, no para el matvec original

La numeración de colores



La numeración de colores

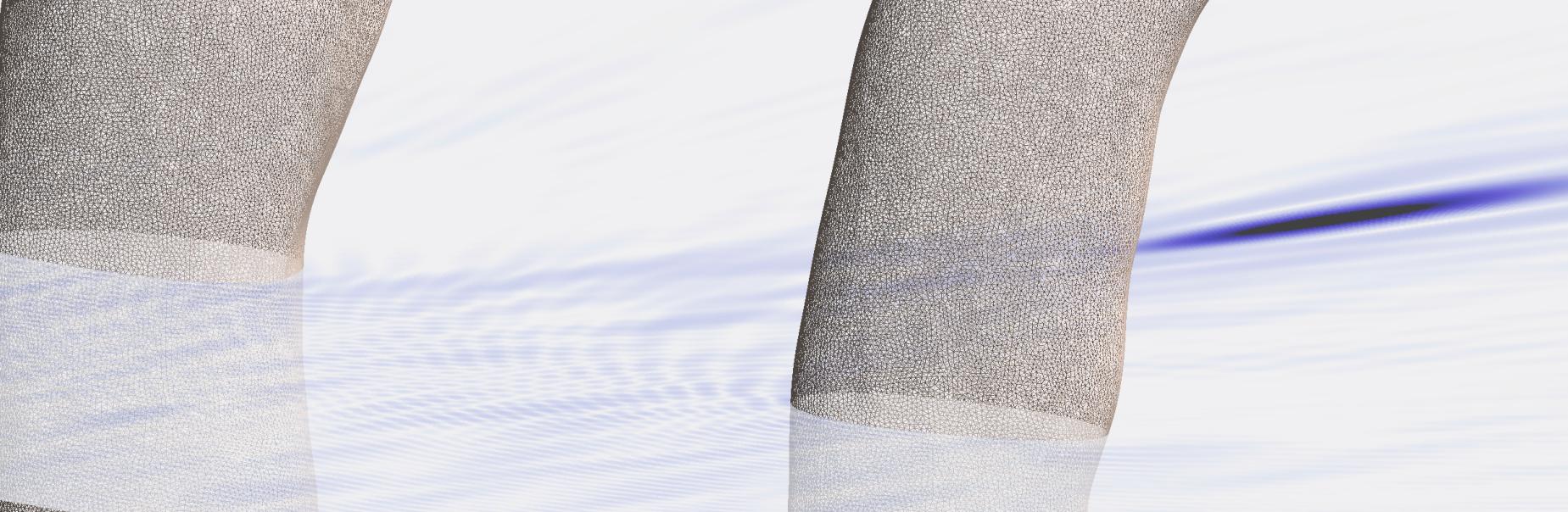


Resumen

- Implementar preacondicionamiento en paralelo
 - preacondicionamiento por bloques diagonales
 - numeración con colores independientes

Clase siguiente

- Computadores heterogéneos



IMT2112 - Algoritmos Paralelos en Computación Científica

Preacondicionamiento en paralelo

Elwin van 't Wout

17 de octubre de 2019



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

Facultad de Matemáticas • Escuela de Ingeniería

imc.uc.cl