Universidade de Aveiro

Computação Paralela — 2019/2020

Mestrado Integrado
em Engenharia Computacional

17 de junho de 2020

A ser entregue (via email) até ao fim do dia 3 de julho

Projeto 2: Paralelização MPI de métodos iterativos de resolução da equação de Poisson

Neste trabalho vai estudar o problema descrito pela equação de Poisson

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

onde

$$f(x,y) = 2 - x^2 - 10y + 50xy,$$

num domínio bidimensional quadrado com $-1 \le x \le +1$ e $-1 \le x \le +1$.

Para obter a solução numérica, o domínio é discretizado nas duas direções. Durante as aulas, foi escrito um programa paralelizado com MPI que se aproxima recursivamente da solução, usando o método de Jacobi baseado em aproximações de diferenças finitas das segundas derivadas num estêncil de 5 pontos.

Na notação que vamos usar, $V^{(k)}(i,j)$ representa o valor de V, após a iteração k, no ponto de coordenadas $x(i) = -1 + ih_x$, com $i = 0, 1, 2, \dots, n_x - 1$, e $y(j) = -1 + jh_y$, com $j = 0, 1, 2, \dots, n_y - 1$. Assuma $h_x = h_y = h$ e, consequentemente, $n_x = n_y$.

Nestas condições, para pontos que não têm o seu valor de V especificado por uma condição fronteira, o método de Jacobi é expresso por

$$V^{(k+1)}(i,j) = \frac{1}{4}\Big[V^{(k)}(i-1,j) + V^{(k)}(i,j-1) + V^{(k)}(i,j+1) + V^{(k)}(i+1,j) - h^2 f(i,j)\Big].$$

Considere que o método convergiu quando o valor de

$$\frac{\sqrt{\sum_{i,j}\left[V^{(k+1)}(i,j) - V^{(k)}(i,j)\right]^2}}{\sqrt{\sum_{i,j}\left[V^{(k+1)}(i,j)\right]^2}}$$

é inferior a uma certa tolerância definida à partida.

A menos que sejam dadas indicações em contrário no enunciado, use as condições fronteira $V(x = \pm 1, y) = V(x, y = \pm 1) = 0$.

Todos os seus programas devem ser baseados numa decomposição do domínio numa malha retangular de subdomínios, de acordo com o código desenvolvido nas aulas.

Escreva um relatório descrevendo os desafios que encontrou em cada alínea, bem como as soluções que encontrou. Faça os comentários que achar relevantes.

a) Altere o programa feito nas aulas (com `2` $\times$ `numprocs/2` subdomínios), fazendo com que, após a convergência, cada processo escreva num ficheiro binário global, usando a função `MPI_File_write_all`, os valores de V que teve a responsabilidade de calcular. Tenha em atenção que o ficheiro deve incluir ainda os valores de V nas condições fronteira. Use um programa externo para importar o ficheiro e produzir uma representação gráfica de V em função de x e y que deverá ser incluída no relatório.

b) Edite o programa da alínea anterior, e obtenha a solução para o mesmo problema, mas com as condições fronteira $V(x = -1, y) = 1 \cdot y$, $V(x = +1, y) = 5/2 + y/2$, $V(x, y = -1) = 1/2 + 3 \cdot x/2$ e $V(x, y = +1) = 2 + 1 \cdot x$. Inclua a representação gráfica do resultado no relatório. Use estas condições fronteira apenas nesta alínea.

c) Os erros resultantes da discretização são menores quando se usa um estêncil de 9 pontos, de tal forma que para pontos que não pertencem às condição fronteira, o método de Jacobi passa a ser expresso por

$$
\begin{aligned}
V^{(k+1)}(i,j) = \frac{1}{20}\Big[& V^{(k)}(i-1,j-1) + 4V^{(k)}(i-1,j) + V^{(k)}(i-1,j+1) \\
& + 4V^{(k)}(i,j-1) + 4V^{(k)}(i,j+1) \\
& + V^{(k)}(i+1,j-1) + 4V^{(k)}(i+1,j) + V^{(k)}(i+1,j+1)\Big] \\
& - \frac{h^2}{40}\Big[f(i-1,j) + f(i,j-1) + 8f(i,j) + f(i,j+1) + f(i+1,j)\Big].
\end{aligned}
$$

Altere o programa da alínea a), passando a usar este algoritmo. Tenha em atenção que cada processo vai ter que comunicar com mais vizinhos.

d) Nesta alínea vai voltar a usar o estêncil original de 5 pontos. A diferença do método de Gauß–Seidel em relação ao método de Jacobi é que em vez de de se calcular o novo valor de V em cada ponto a partir dos valores de V dos vizinhos na iteração anterior, se passa a usar os seus valores na nova iteração, se eles já estiverem disponíveis. Se o programa não fosse paralelizado, ao substituirmos

```
for(i = 1; i < myrows + 1; i++) {
  for(j = 1; j < mycols + 1; j++) {
    mynew[i][j] = 0.25 * (myold[i-1][j] + myold[i][j-1]
      + myold[i][j+1] + myold[i+1][j] - h * h * myf[i][j]);
```

por

```
for(i = 1; i < myrows + 1; i++) {
  for(j = 1; j < mycols + 1; j++) {
    mynew[i][j] = 0.25 * (mynew[i-1][j] + mynew[i][j-1]
      + mynew[i][j+1] + mynew[i+1][j] - h * h * myf[i][j]);
```

estaríamos a aplicar o método de Gauß–Seidel expresso por

$$\begin{aligned}V^{(k+1)}(i,j) = \frac{1}{4}\Big[&V^{(k+1)}(i-1,j) + V^{(k+1)}(i,j-1)\\ &+ V^{(k)}(i,j+1) + V^{(k)}(i+1,j) - h^2 f(i,j)\Big].\end{aligned}$$

Explique porquê no relatório. Explique também porque é que isso já não aconteceria para todos os pontos num programa paralelizado. A abordagem habitual para resolver este problema é usar um esquema de atualização alternativo, designado habitualmente vermelho–preto (ou par–ímpar). Reescreva o programa da alínea a) passando a aplicar o método de Gauß–Seidel com uma atualização do tipo vermelho–preto. Faça uma pesquisa bibliográfica para estudar os pormenores. Pode começar pelo capítulo 2 do livro de Jianping Zhu, *Solving Partial Differential Equations on Parallel Computers*, World Scientific, 1994.