

UFRGS – PPGMAP  
Computação Paralela(Prof. Pedro Konzen)

Uso de OpenMP em uma função evolutiva  
do p-Laplaciano usando diferenças finitas

Roberto Hoo (23/março/2021)

Este trabalho é uma adaptação da dissertação de mestrado

Computação Paralela com OpenMP  
e aplicações ao estudo de funções p-harmônicas.

em matemática aplicada apresentada em 21 de dezembro de 2018 na  
UFRGS/Campus do Vale feita por mim e cujo orientador foi o  
prof. Paulo Zingano(UFRGS)

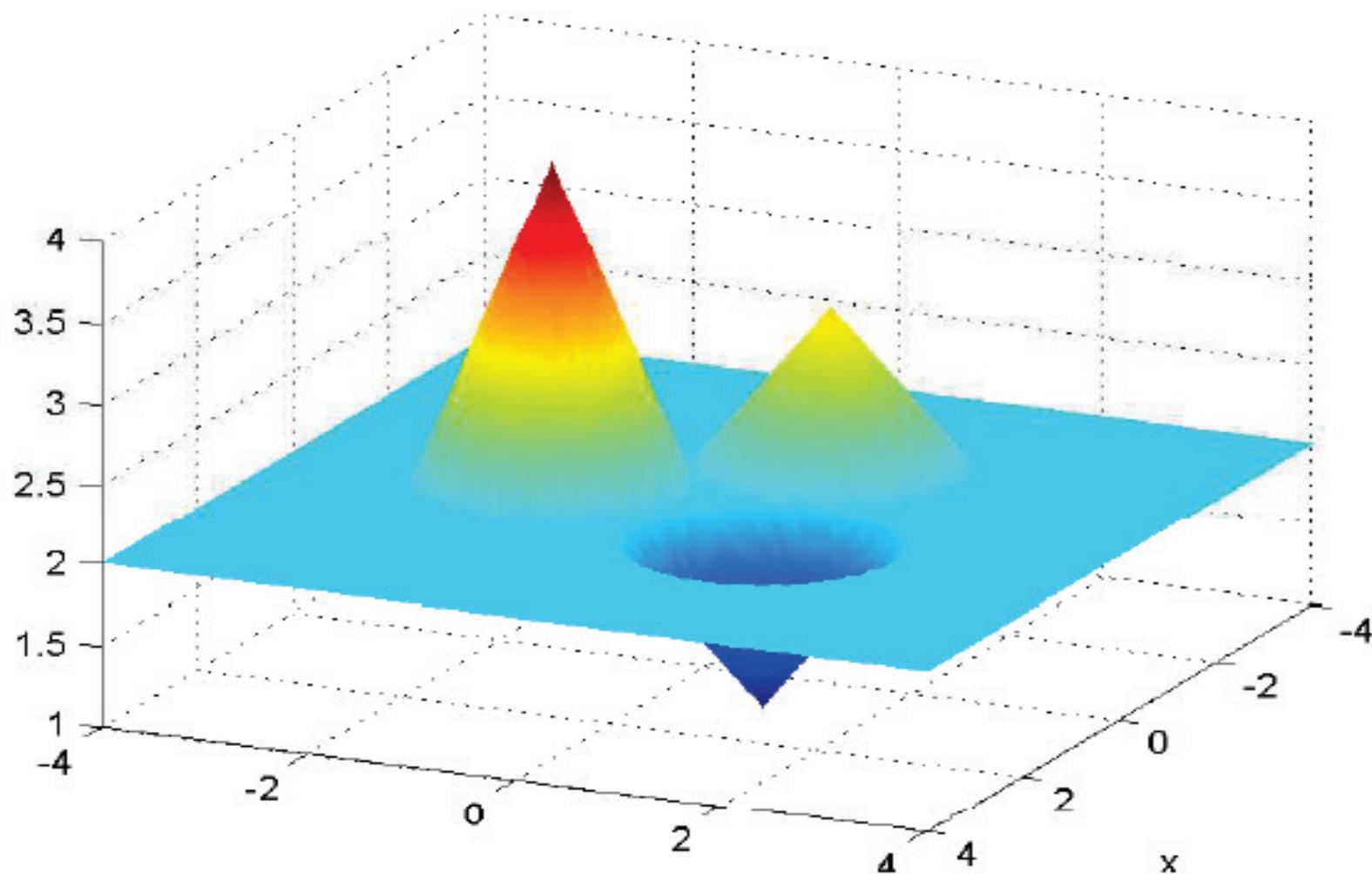
Objeto de estudo:  
O comportamento da função  $u: \mathbb{R}^n \rightarrow \mathbb{R}$  que satisfaz a equação

$$\begin{cases} \Delta_p u \equiv \operatorname{div}[|\nabla u|^{p-2} \nabla u] = 0 & \text{em } \mathbb{R}^n \setminus P \\ & e \\ u(a_j) = b_j, \quad a_j \in P, \quad 1 \leq j \leq m \end{cases} \quad (1)$$

onde  $P = \{a_1, \dots, a_m\}$  são  $m$  pontos de  $\mathbb{R}^n$  e  
 $u(a_j) = b_j, \quad 1 \leq j \leq m$ , são as  $m$  singularidades(bicos, vértices) de  $u$ .

**Quando**  $|r| \rightarrow \infty$ ,  $r = \sqrt{x_1^2 + \dots + x_n^2}$

Por exemplo: um esboço, inicial da função  
 $u = u(x, y)$ ,  $u: \mathbb{R}^2 \rightarrow \mathbb{R}$ ,  $n=2$  com  $m=3$  singularidades



## Explicando a equação (1)

- a) A expressão  $\Delta_p u \equiv \operatorname{div}[|\nabla u|^{p-2} \nabla u]$  (2)  
é chamado de **p-Laplaciano** de  $u = u(x_1, \dots, x_n)$
- b) As funções que satisfazem  $\Delta_p u \equiv \operatorname{div}[|\nabla u|^{p-2} \nabla u] = 0$  (3)  
são chamados funções p-harmônica .

## *Fato 1: Solução fundamental de $\Delta_p u=0$ (3)*

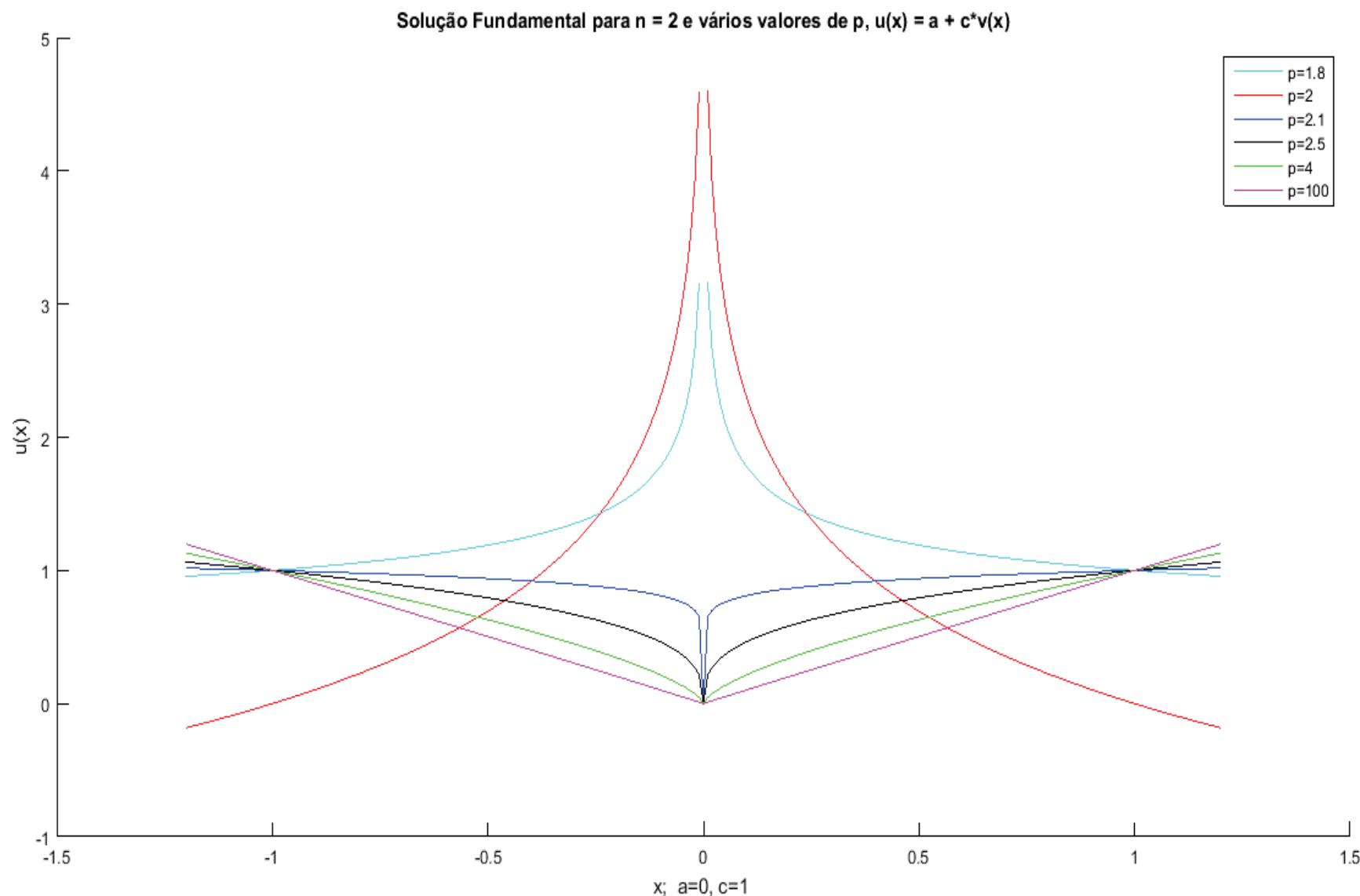
Uma possível solução de (3) é a função

$$v(x) := \begin{cases} |x - x_0|^{p-n/p-1}, & \text{se } p \neq n \\ \text{ou} \\ -\log|x - x_0|, & \text{se } p = n \end{cases}, \quad \text{onde } x \in \mathbb{R}^n \setminus \{x_0\} \quad (4)$$

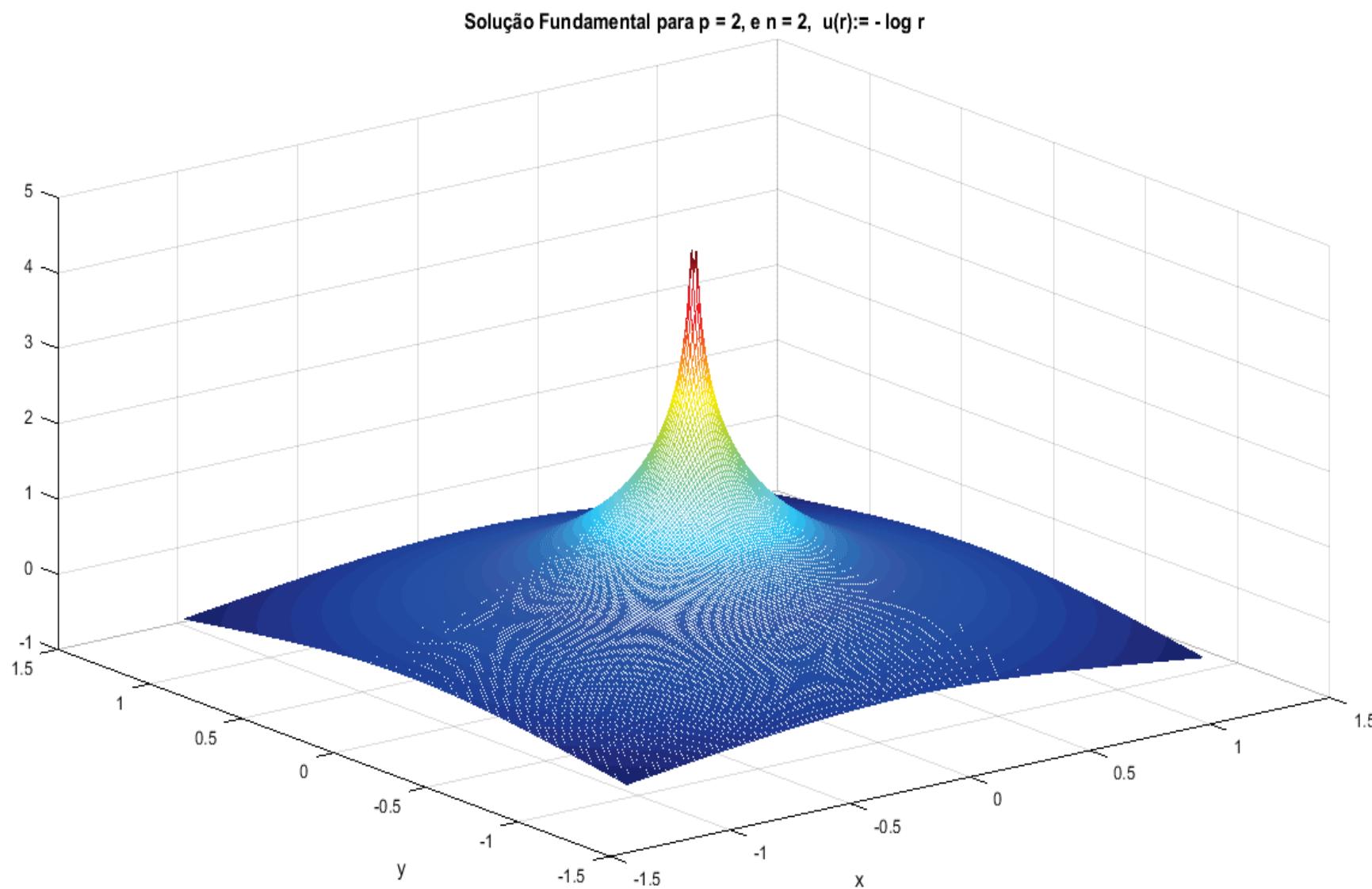
A solução  $v$  possui simetria radial e também é chamado de *função p-harmônica fundamental* ou *solução fundamental de  $\Delta_p u=0$*

Obs: Se  $v(x)$  é p-harmônica,  $\Delta_p v = 0$ , então  $u(x) := a + c * v(x)$  também é, p-harmônica,  $\Delta_p u = 0$ , dados  $a, c \in \mathbb{R}$  quaisquer.

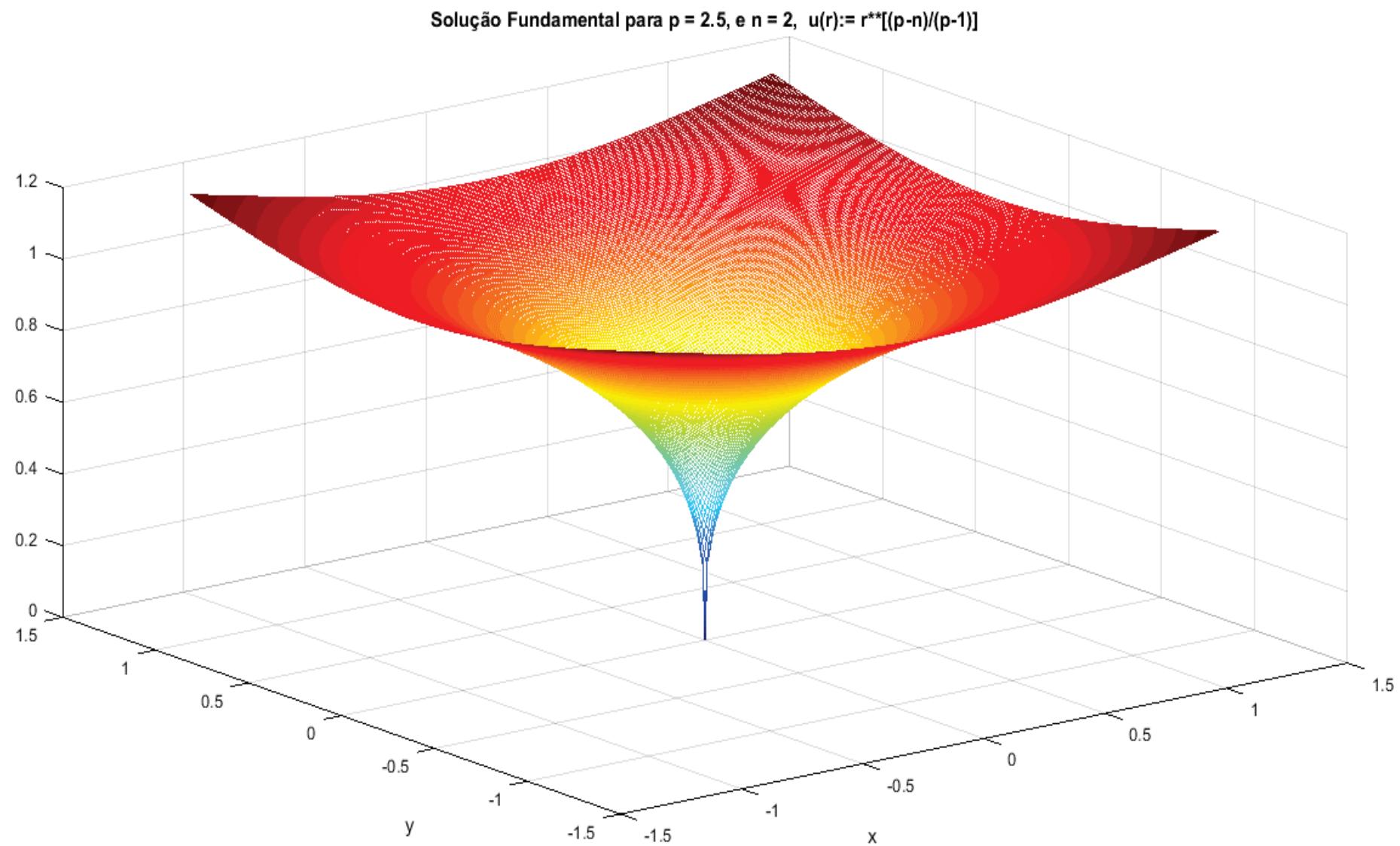
Solução Fundamental para n=2 e vários valores de p,  $u(x) = a + c * v(x)$



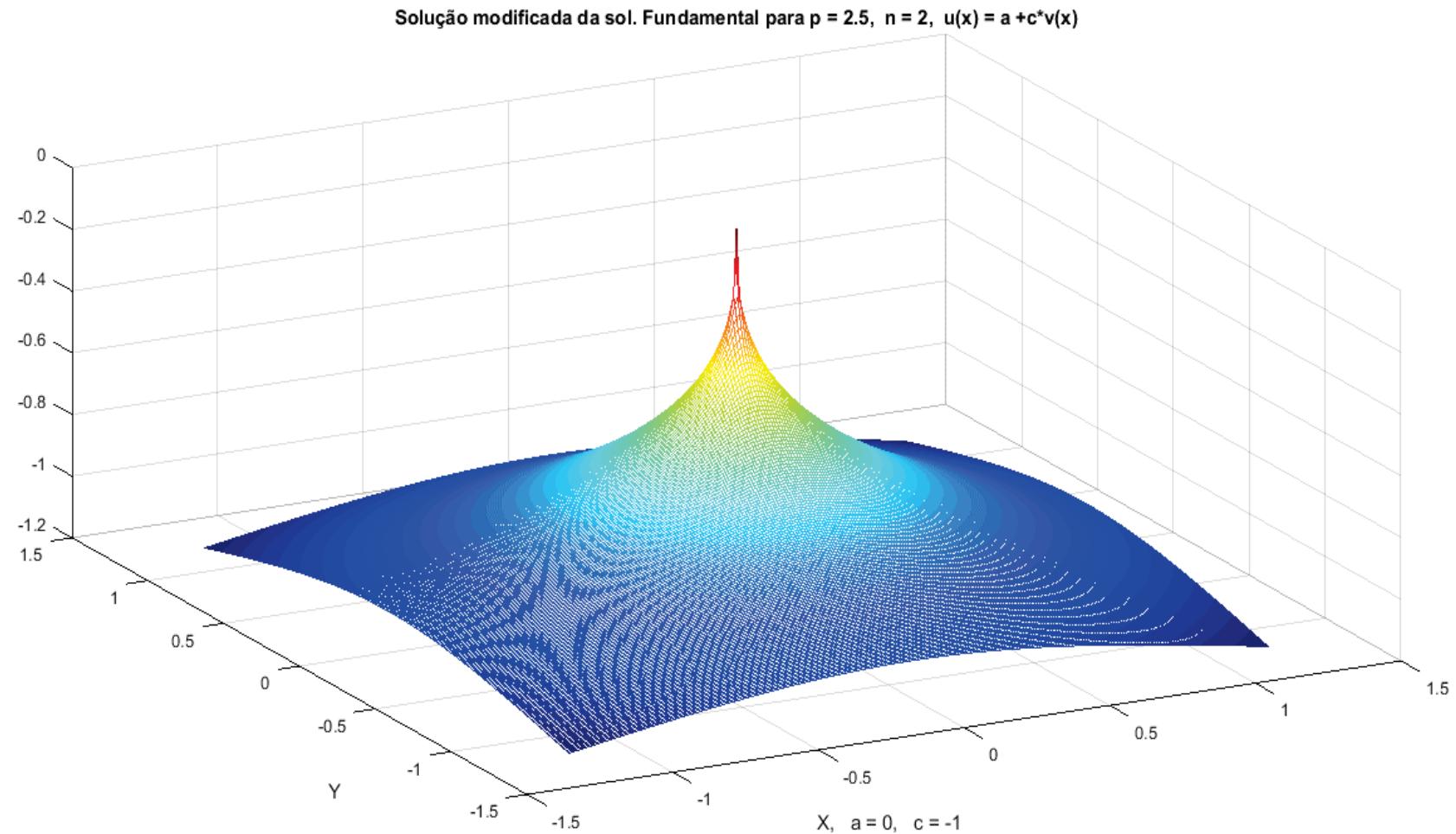
# Solução Fundamental para $p = n = 2$ , $u(r) = -\log(r)$



Solução Fundamental para  $p = 2.5$ ,  $n=2$  ,  $u(r) = r^{(p-n)/(p-1)}$



**Solução Fundamental para  $p = 2.5$ ,  $n=2$  ,    $u(x) = a + c * v(x) = u(r) = -r^{(p-n)/p-1}$ ,  $a = 0, c = -1$**



Fato 2: O artigo feito por: Bonorino, Silva e Zingano: - Liouville's theorem and comparison results for solutions of degenerate elliptic equations in exterior domains – diz que o problema abaixo tem solução.

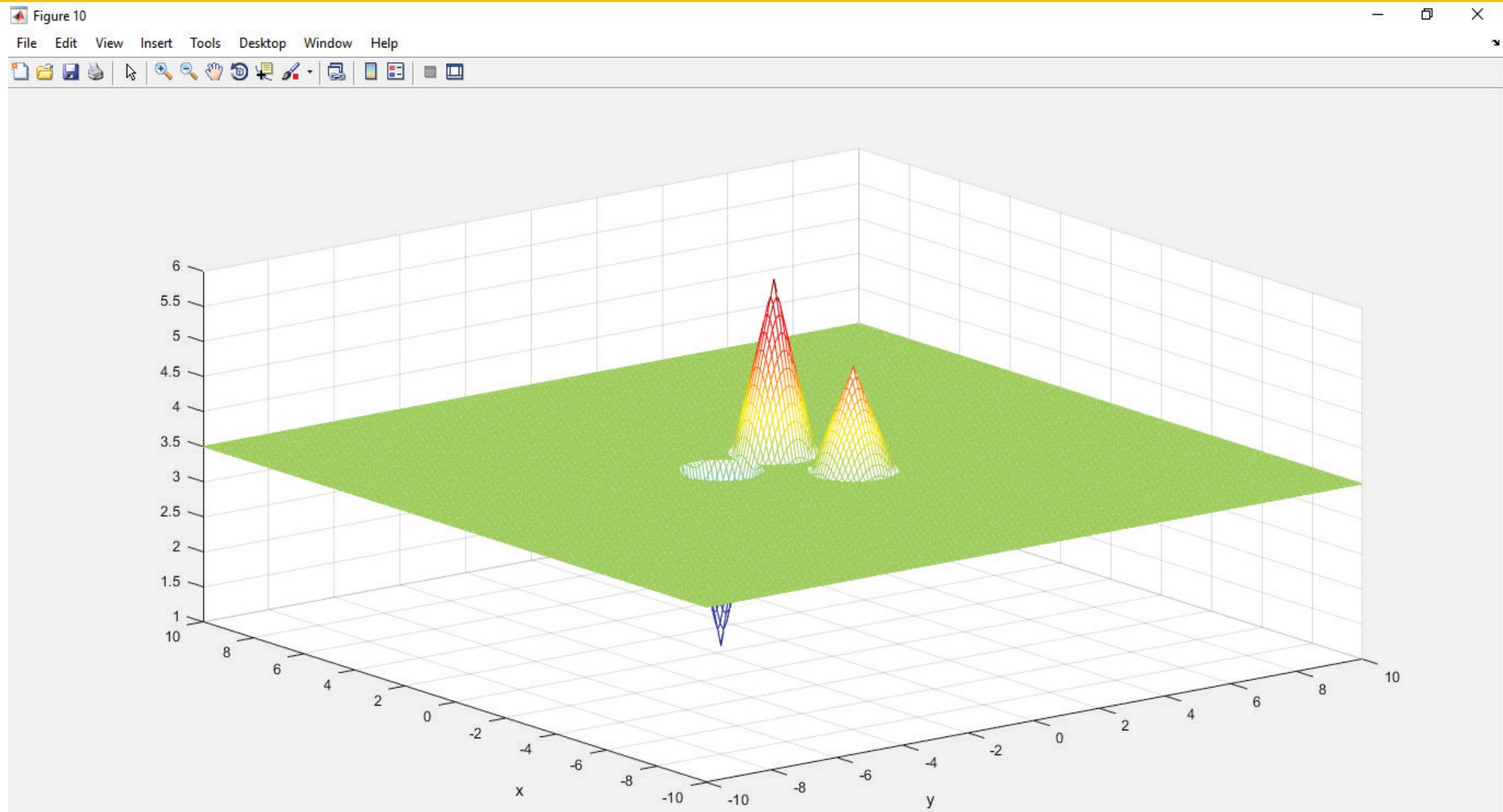
Indeed, for  $p > n$ ,  $P = \{x_1, \dots, x_k\} \subset \mathbb{R}^n$  and  $m_1, \dots, m_k \in \mathbb{R}$ , we show that the problem

$$\begin{cases} \Delta_p u = 0 & \text{in } \mathbb{R}^n \setminus P \\ u(x_i) = m_i & \text{for } i \in \{1, \dots, k\} \end{cases} \quad (20)$$

has a bounded weak solution in  $C(\mathbb{R}^n) \cap C^1(\mathbb{R}^n \setminus P)$ .

**Proposition 5.1** *Problem (20) has a weak solution in  $C(\mathbb{R}^n) \cap C^1(\mathbb{R}^n \setminus P) \cap L^\infty(\mathbb{R}^n)$ .*

# Exemplo de $u(x, y, t = 0)$ usando no Fato 2 e Fato 3



Fato 3: O teorema 2.1(Princípio da Comparação) também adaptado de { Bonorino, Silva, e Zingano}, reproduzido abaixo, diz:

**Teorema 2.1 - Princípio de Comparação** (L. P. Bonorino [2]).

Sejam  $u, v \in C^0(\mathbb{R}^n) \cap L^\infty(\mathbb{R}^n) \cap C^1(\mathbb{R}^n \setminus P)$  soluções da equação (2.6a) em  $\mathbb{R}^n \setminus P$ , onde  $P = \{x_1, \dots, x_m\}$  e  $p \geq n$ . Então, tem-se

$$u(x_j) \leq v(x_j) \quad \forall 1 \leq j \leq m \implies u(x) \leq v(x) \quad \forall x \in \mathbb{R}^n \quad (2.20)$$

ou seja: se  $u \leq v$  em  $P$ , então  $u \leq v$  em todo o  $\mathbb{R}^n$ .

$$\Delta_p u = 0 \quad \text{em } \mathbb{R}^n \setminus P, \quad (2.6a)$$

$$u(x_j) = b_j, \quad 1 \leq j \leq m. \quad (2.6b)$$

Este teorema tem dois corolários, o primeiro corolário diz que a solução do problema (2.6) é única. O segundo corolário diz que  $\forall x \in \mathbb{R}^n$  temos:

$$\min_{1 \leq j \leq m} b_j =: b \leq u(x) \leq B := \max_{1 \leq j \leq m} b_j , \quad u(x_j) = b_j \quad (5)$$

- **Fato 4:** Nestes dois artigos

- [11] M. FRAAS AND Y. PINCHOVER, *Positive Liouville theorems and asymptotic behavior for  $p$ -Laplacian type elliptic equations with a Fuchsian potential*, Confl. Math. 3 (2011), 291-323.
- [12] M. FRAAS AND Y. PINCHOVER, *Isolated singularities of positive solutions of  $p$ -Laplacian type equations in  $\mathbb{R}^d$* , J. Diff. Eqs. 254 (2013), 1097-1119.

Temos este teorema(provado no apêndice A de [11]):

**Theorem 1.12.** *Let  $p \geq d > 1$ , and let  $u$  be a positive solution of the equation  $-\Delta_p(u) = 0$  in a neighborhood of infinity in  $\mathbb{R}^d$ . Then either  $u$  has a removable singularity at  $\infty$  (i.e.,  $u$  admits a finite limit as  $x \rightarrow \infty$ ), or*

$$u(x) \underset{x \rightarrow \infty}{\sim} \begin{cases} |x|^{\alpha^*} & \text{if } p > d, \\ \log|x| & \text{if } p = d. \end{cases} \quad \alpha^* := \frac{p-d}{p-1}$$

O fato 4 nos diz que a longa distância a solução de  $\Delta_p u = 0$  ou é uma constante ou é parecido com a solução fundamental. Observe que o problema do Fato 2 e 3 é semelhante ao problema do Fato 4 (é o Fato 4 mais as singularidades). Mas a solução fundamental, não é limitada, quando  $x$  tende ao infinito e o Fato 3 nos diz que a solução é limitada. Logo:

Juntando os Fatos 1, 2, 3,4 é natural supor, que possivelmente:  $\lim_{|x| \rightarrow \infty} u(x) = b_\infty$   
 Ou seja a longa distância a solução do nosso problema (1), provavelmente, segue o teorema do Fato 4, isto é, tende a uma constante.

$$\begin{cases} \Delta_p u \equiv \operatorname{div}[|\nabla u|^{p-2} \nabla u] = 0 & \text{em } \mathbb{R}^n \setminus P \\ & e \\ u(a_j) = b_j, \quad a_j \in P, \quad 1 \leq j \leq m \end{cases} \quad (1)$$

- Como achar este valor  $\mathbf{b}_\infty$  ?
- Para usar o computador procuramos alguma iteração tal que

$$\mathbf{u}_{k+1} = \mathbf{u}_k + [?]$$

- Vamos supor que  $\{\mathbf{u}_k\}_{k \in \mathbb{R}}$  é uma sequência tal que  $\mathbf{u}_k$  converge para  $\mathbf{u}_\infty$  e  $\operatorname{div}[|\nabla \mathbf{u}_\infty|^{p-2} \nabla \mathbf{u}_\infty] = 0$ , sendo  $\mathbf{u}_k = \mathbf{u}(x, k\Delta t)$  então:

$$\frac{\partial \mathbf{u}_k}{\partial t} \cong \frac{\mathbf{u}_{k+1} - \mathbf{u}_k}{\Delta t} = \frac{\mathbf{u}(x, k\Delta t + \Delta t) - \mathbf{u}(x, k\Delta t)}{\Delta t} \xrightarrow[k \rightarrow \infty]{} \mathbf{0} \quad (6)$$

- E também temos:  $\Delta_p \mathbf{u}_k = \operatorname{div}(|\nabla \mathbf{u}_k|^{p-2} \nabla \mathbf{u}_k) \xrightarrow[k \rightarrow \infty]{} 0 \quad (7)$

- Logo podemos supor  $\frac{\partial \mathbf{u}_k}{\partial t} \approx \operatorname{div}(|\nabla \mathbf{u}_k|^{p-2} \nabla \mathbf{u}_k)$  (8)

Ou seja estamos trocando a equação

$$\begin{cases} \Delta_p u \equiv \operatorname{div}[|\nabla u|^{p-2} \nabla u] = 0 & \text{em } \mathbb{R}^n \setminus P \\ u(a_j) = b_j, \quad a_j \in P, \quad 1 \leq j \leq m \end{cases} \quad (1)$$

Pela equação evolutiva do p-Laplaciano

$$\begin{cases} u_t = \Delta_p u \\ u(x, 0) = u_0(x), \quad x \in \mathbb{R}^n \\ u(x_j, t) = b_j, \quad 1 \leq j \leq m, \quad t \geq 0 \end{cases} \quad (9)$$

- Expandindo (9) e usando  $n=2$  e  $p>n$  (foram estes valores usado na simulação numérica) temos:

$$u_t = \frac{\partial}{\partial x} [(u_x^2 + u_y^2)^{\frac{p-2}{2}} u_x] + \frac{\partial}{\partial y} [(u_x^2 + u_y^2)^{\frac{p-2}{2}} u_y] \quad (10)$$

- O computador usa quantidades discretas, então vamos transformar (10) usando diferenças finitas

$$\frac{u_{i,j,k+1} - u_{i,j,k}}{\Delta t} = \operatorname{div} \left( |\nabla u_{i,j,k}|^{p-2} \nabla u_{i,j,k} \right) \quad (11)$$

$$u_{i,j,k} = u(x_i, y_j, t_k) = u(i * h, j * h, k * \Delta t); \quad i, j, k \in \mathbb{Z} \text{ (e } k \geq 0)$$

Expandindo (11) temos:

$$\begin{aligned}
 \bullet \frac{u_{i,j,k+1} - u_{i,j,k}}{\Delta t} &= \frac{1}{h} \left[ \left( \frac{u_{i+1,j} - u_{i,j}}{h} \right)^2 + \left( \frac{u_{i+1,j+1} + u_{i,j+1} - u_{i+1,j-1} - u_{i,j-1}}{2(2h)} \right)^2 \right]^{\frac{p-2}{2}} \cdot \left( \frac{u_{i+1,j} - u_{i,j}}{h} \right) \\
 &\quad - \frac{1}{h} \left[ \left( \frac{u_{i,j} - u_{i-1,j}}{h} \right)^2 + \left( \frac{u_{i,j+1} + u_{i-1,j+1} - u_{i,j-1} - u_{i-1,j-1}}{2(2h)} \right)^2 \right]^{\frac{p-2}{2}} \cdot \left( \frac{u_{i,j} - u_{i-1,j}}{h} \right) \\
 &\quad + \frac{1}{h} \left[ \left( \frac{u_{i,j+1} - u_{i,j}}{h} \right)^2 + \left( \frac{u_{i+1,j} + u_{i+1,j+1} - u_{i-1,j} - u_{i-1,j+1}}{2(2h)} \right)^2 \right]^{\frac{p-2}{2}} \cdot \left( \frac{u_{i,j+1} - u_{i,j}}{h} \right) \\
 &\quad - \frac{1}{h} \left[ \left( \frac{u_{i,j} - u_{i,j-1}}{h} \right)^2 + \left( \frac{u_{i+1,j-1} + u_{i+1,j} - u_{i-1,j-1} - u_{i-1,j}}{2(2h)} \right)^2 \right]^{\frac{p-2}{2}} \cdot \left( \frac{u_{i,j} - u_{i,j-1}}{h} \right)
 \end{aligned}$$

Ou...

onde

$$v_{i,j} = u_{i,j,k+1}$$

e

$$u_{i,j}; u_{i+1,j}$$

$$u_{i-1,j}; u_{i,j-1}$$

$$u_{i+1,j+1};$$

etc... são  
calculados no  
tempo

$$t_k = k \cdot \Delta t$$

$$u_t = \operatorname{div}(|\nabla u|^{p-2} \nabla u) = \frac{\partial}{\partial x} \left[ (u_x^2 + u_y^2)^{\frac{p-2}{2}} u_x \right] + \frac{\partial}{\partial y} \left[ (u_x^2 + u_y^2)^{\frac{p-2}{2}} u_y \right]$$

é transformada por diferenças finitas em

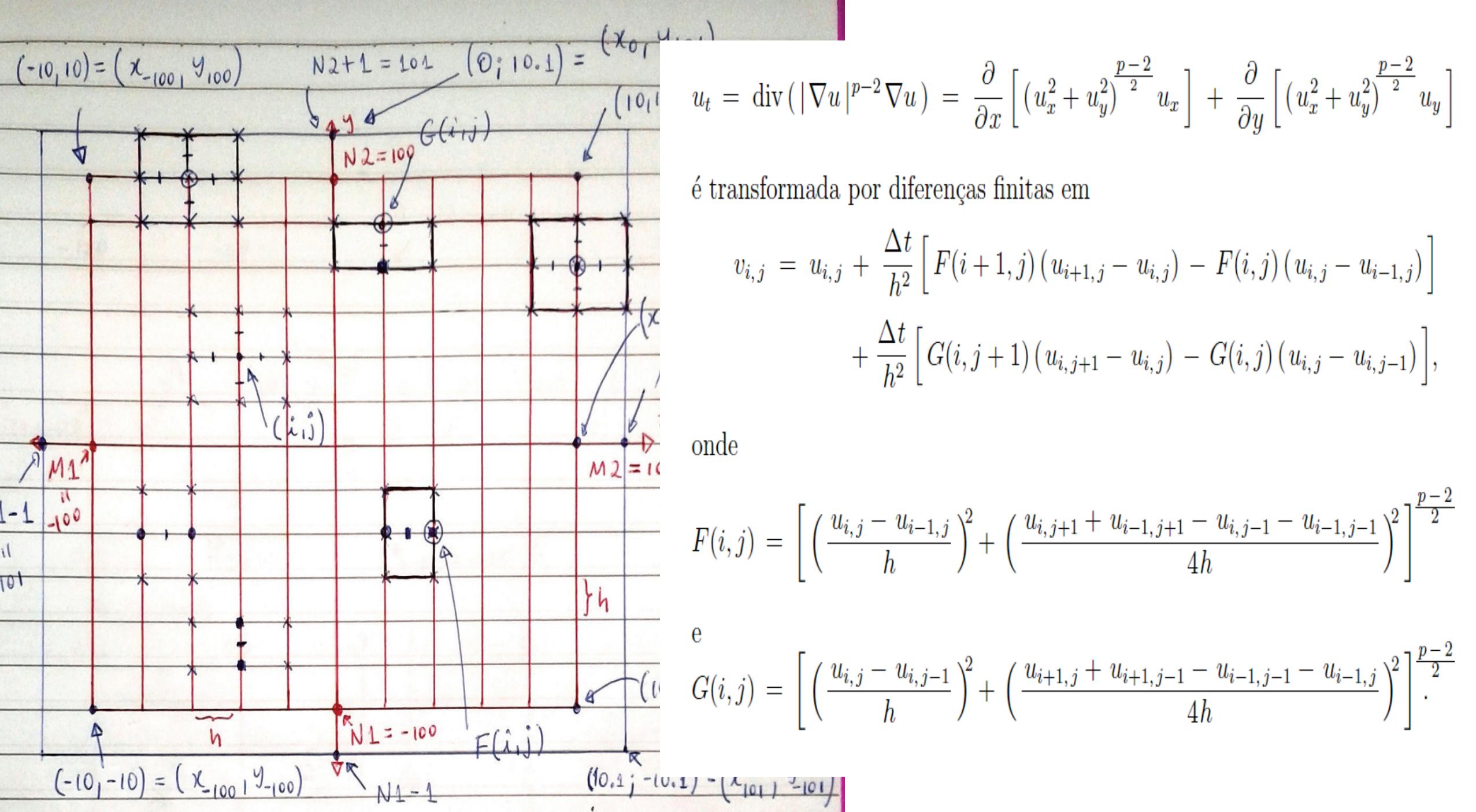
$$\begin{aligned} v_{i,j} &= u_{i,j} + \frac{\Delta t}{h^2} \left[ F(i+1,j)(u_{i+1,j} - u_{i,j}) - F(i,j)(u_{i,j} - u_{i-1,j}) \right] \\ &\quad + \frac{\Delta t}{h^2} \left[ G(i,j+1)(u_{i,j+1} - u_{i,j}) - G(i,j)(u_{i,j} - u_{i,j-1}) \right], \end{aligned}$$

onde

$$F(i,j) = \left[ \left( \frac{u_{i,j} - u_{i-1,j}}{h} \right)^2 + \left( \frac{u_{i,j+1} + u_{i-1,j+1} - u_{i,j-1} - u_{i-1,j-1}}{4h} \right)^2 \right]^{\frac{p-2}{2}}$$

e

$$G(i,j) = \left[ \left( \frac{u_{i,j} - u_{i,j-1}}{h} \right)^2 + \left( \frac{u_{i+1,j} + u_{i+1,j-1} - u_{i-1,j-1} - u_{i-1,j}}{4h} \right)^2 \right]^{\frac{p-2}{2}}.$$

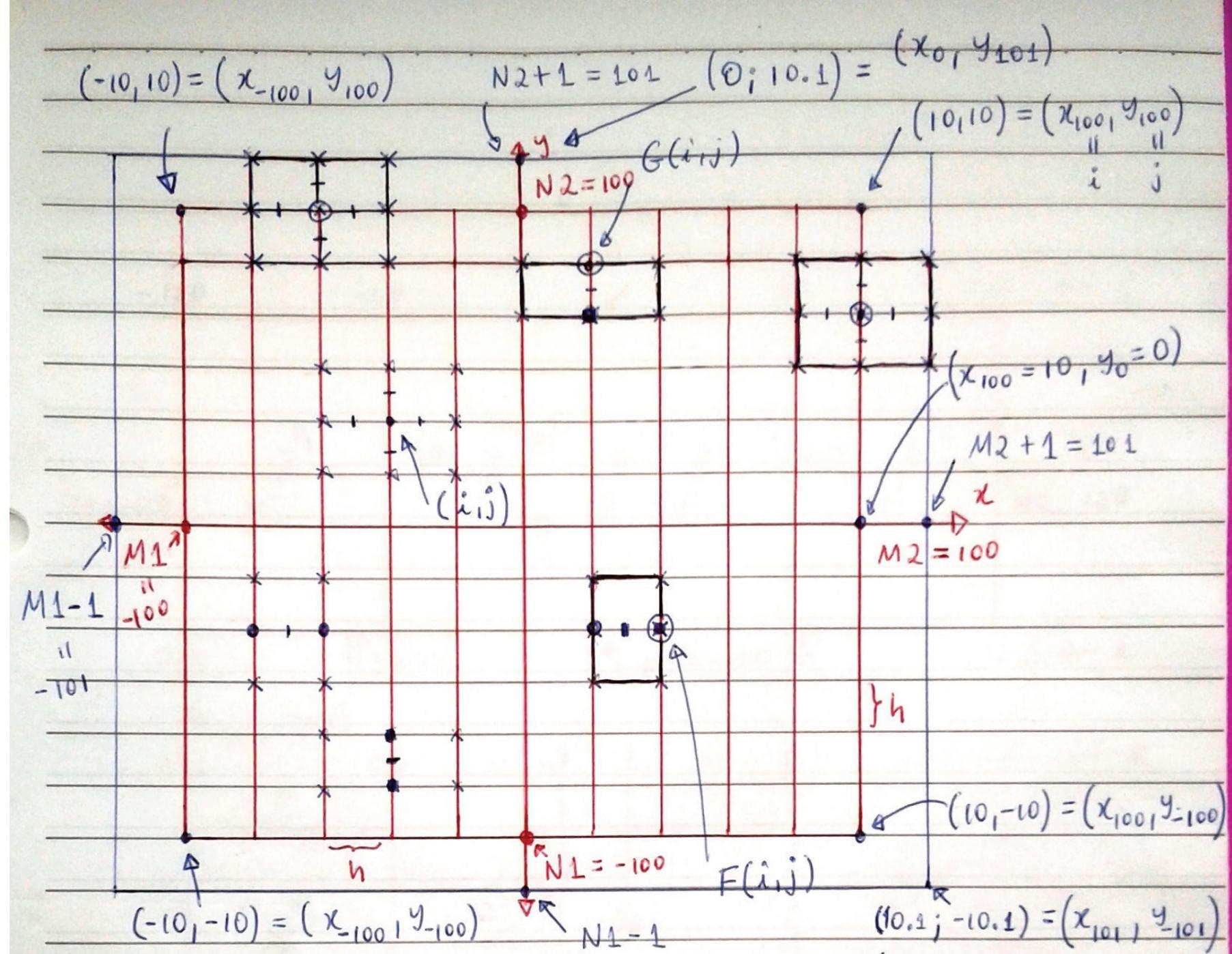


Alguns valores usados nas simulações:

- $M_1 = -100$
- $M_2 = 100$
- $N_1 = -100$
- $N_2 = 100$
- $h = 0.1$
- $\Delta t = 0.0001$

Símbolos usados:

- $x_i = i \cdot h,$
- $y_j = j \cdot h,$
- $t_k = k \cdot \Delta t,$
- $i, j, k \in \mathbb{Z} \text{ (e } k \geq 0)$



## Condição inicial, vamos supor 3 singularidades

Para facilitar, definimos o valor inicial  $u(x, 0) = u_0$ , um plano de altura  $b_0 = 2$  e 3 cones com vértices:

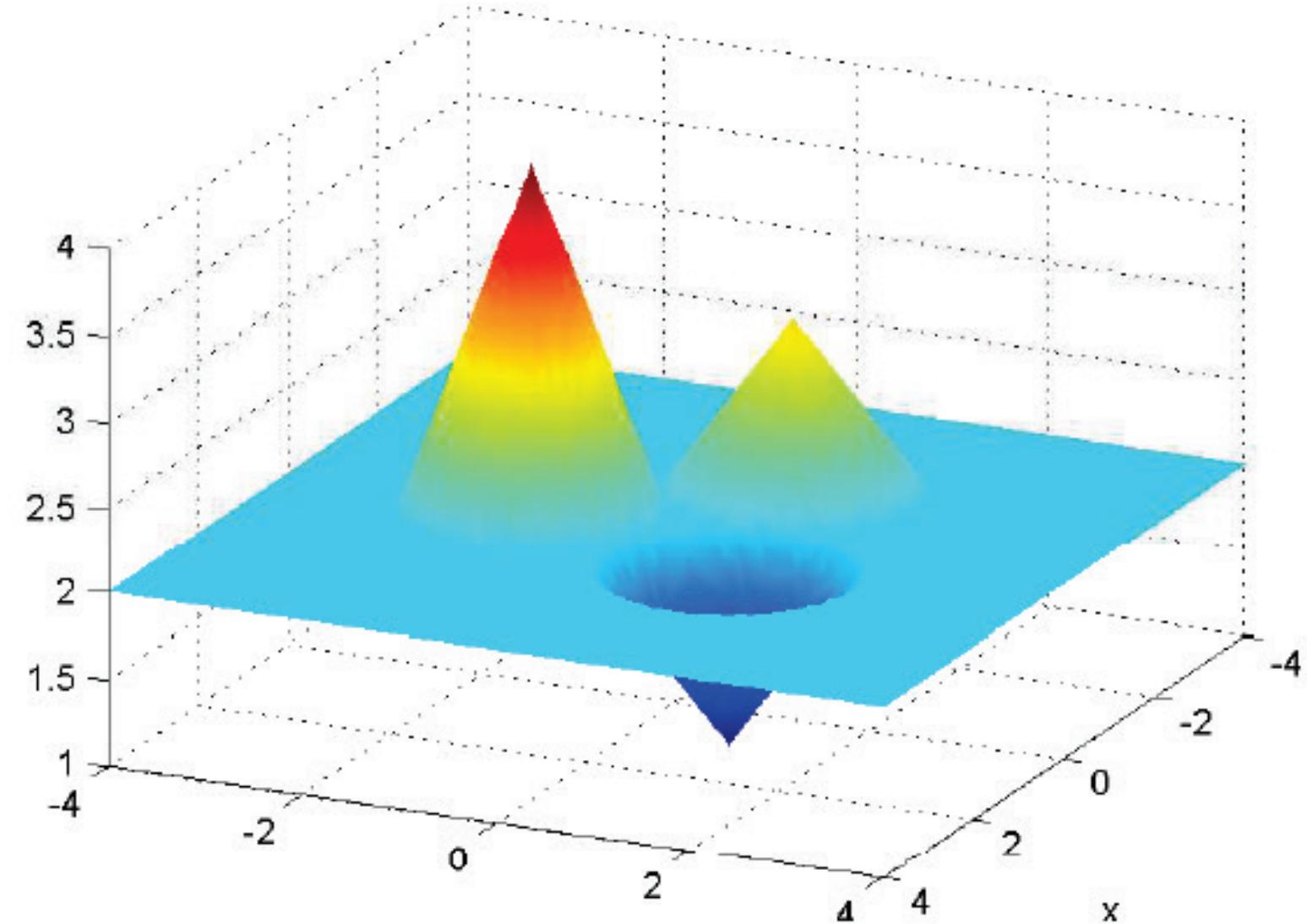
$$a_1 = (1, 1), b_1 = 1;$$

$$a_2 = (-1.5, 0.5), b_2 = 3$$

$$a_3 = (0, -1.5), b_3 = 4$$

$$\text{todos com raio } r_0 = \frac{1}{2} \min\{d(a_i, a_j) : 1 \leq i \leq j \leq m\}$$

A condição inicial em todas as simulações seguiu o mesmo padrão: um plano de altura  $b_0$  e 3 ou 4 cones



# Parte principal do código

u24h\_serial4P.f90 (Serial 4Sing) - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

u24h\_omp10.f90 x u24h\_serial4P.f90 x

```
678      ! computation of F values on the grid: !
679      ! *****
680      ! F(i,j) for M1 <= i <= M2 + 1 and N1 <= j <= N2:
681      F(M1:M2+1,N1:N2) = ( (u(M1:M2+1,N1:N2) - u(M1-1:M2,N1:N2))**2 + &
682      ( (u(M1-1:M2,N1+1:N2+1)+u(M1:M2+1,N1+1:N2+1))-(u(M1-1:M2,N1-1:N2-1)+u(M1:M2+1,N1-1:N2-1)) )**2 /16 )**q / h_to_pm2;
683      !
684      ! computation of G values on the grid: !
685      ! *****
686      ! G(i,j) for M1 <= i <= M2 and N1 <= j <= N2 + 1:
687      G(M1:M2,N1:N2+1) = ( (u(M1:M2,N1:N2+1) - u(M1:M2,N1-1:N2))**2 + &
688      ( (u(M1+1:M2+1,N1-1:N2)+u(M1+1:M2+1,N1:N2+1))-(u(M1-1:M2-1,N1-1:N2)+u(M1-1:M2-1,N1:N2+1)) )**2 /16 )**q / h_to_pm2;
689      !
690      ! computation of v = [ new u values at the new time level ]:
691      ! *****
692      ! v(i,j) for M1 <= i <= M2 and N1 <= j <= N2:
693      v(M1:M2,N1:N2) = u(M1:M2,N1:N2) + &
694      cfl*( F(M1+1:M2+1,N1:N2)*(u(M1+1:M2+1,N1:N2)-u(M1:M2,N1:N2)) - &
695      F(M1:M2,N1:N2)*(u(M1:M2,N1:N2)-u(M1-1:M2-1,N1:N2)) ) + &
696      cfl*( G(M1:M2,N1+1:N2+1)*(u(M1:M2,N1+1:N2+1)-u(M1:M2,N1:N2)) - &
697      G(M1:M2,N1:N2)*(u(M1:M2,N1:N2)-u(M1:M2,N1-1:N2-1)) );
698      !
699      ! extending v to the extra grid points:
700      v(M1-1,N1:N2) = v(M1,N1:N2);
701      v(M2+1,N1:N2) = v(M2,N1:N2);
702      v(M1-1:M2+1,N1-1) = v(M1-1:M2+1,N1);
703      v(M1-1:M2+1,N2+1) = v(M1-1:M2+1,N2);
704      !
705      ! computation of new u values completed!
706      ! *****
707      u = v;      ! <-- updating u (at the new time level)
708      ! *****
709      ! correcting u values at the singular points:
710      u(i1,j1) = b1;
711      u(i2,j2) = b2;
712      u(i3,j3) = b3;
713      u(i4,j4) = b4;
```

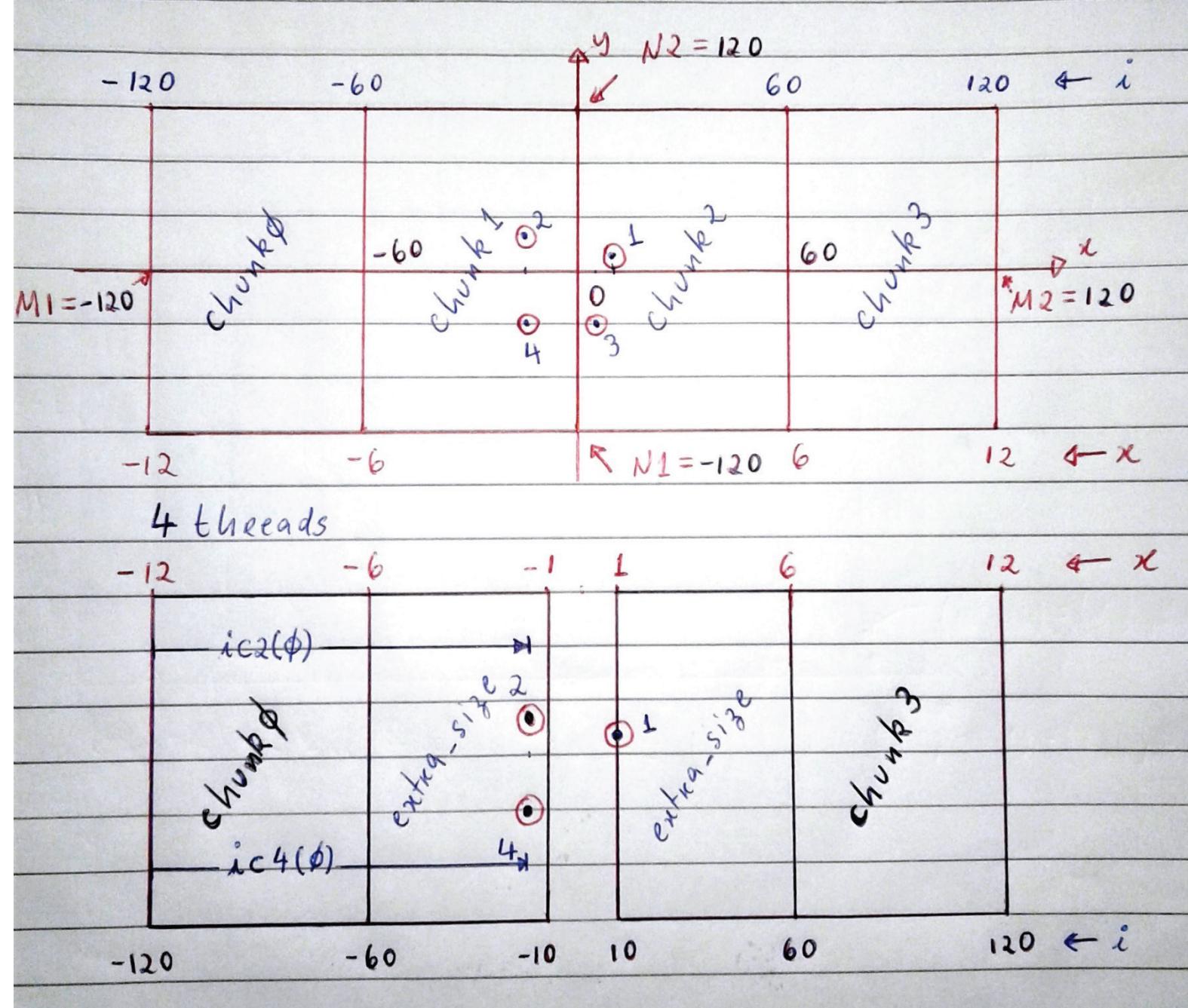
F:\Universidade\2020 02\Computação Paralela\Trabalho 23Mar\Serial 4Sing\u24h\_ser... Fortran Windows (CR+LF) WINDOWS-1252 Line 702, Col 49, Pos 22445 Insert Read/Write default

# Na programação com OpenMP foi usado 2 métodos diferentes

- 1) Digamos que usamos, por exemplo 4 threads. No primeiro método dividimos a malha em 4 pedaços e em cada pedaço adicionamos um pedaço extra de tamanho 50. Cada thread atualiza o seu pedaço 50 vezes e depois unimos todas as partes e atualizamos  $u$ . Método mostrado nos dois slides a seguir.
- 2) No segundo método, dividimos a malha em 4 pedaço e cada thread computa no seu pedaço e no final atualizamos  $u$ .

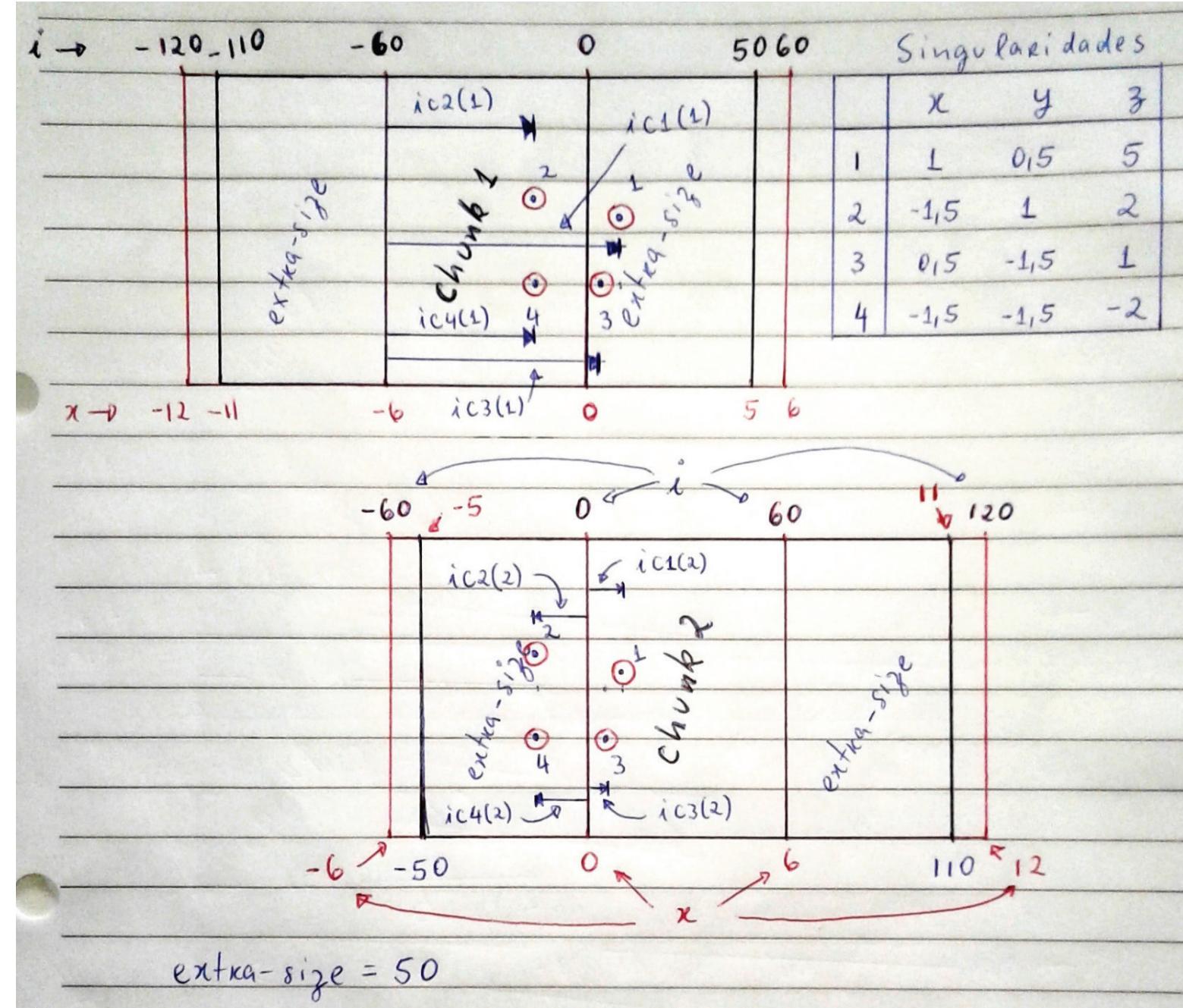
# Parte Principal de u24hOpenMP.f90

- Neste exemplo usamos 4 threads
- $M1 = N1 = -120$
- $M2 = N2 = 120$
- Cada pedaço(cada chunk) tem dimensão  $60 \times 240$
- Temos 4 chunks(4 pedaços)
- Foi adicionado um pedaço-extra de  $50 \times 240$  no lado direito do chunk0 e no lado esquerdo do chunk3.



# Parte Principal de u24hOpenMP.f90

- Nos chunks internos foi adicionado este pedaço-extra nos dois lados
- Cada thread atualizou seu respectivo chunk 50 vezes. Após as 50 atualizações os 4 chunks foram unidos atualizando o quadrado original.
- Repete-se este ciclo: dividir o quadrado em 4 pedaços, cada thread atualiza 50 vezes o seu pedaço, junta os 4 pedaços e atualiza o quadrado original. Até o tempo t atingir o tempo final tf



# Computação feita no Cenapad-SP e Okeanos(Grécia)

## 5.3 Resultado 3 (gerado no CENAPAD-SP e OKEANOS)

Parâmetros do problema:  $p = 2.1$

S1: ( 1.0, 0.5),  $b_1 = 6$ ; S2: (-1.5, 1.0),  $b_2 = 2$ ;

S3: ( 0.5,-1.5),  $b_3 = 2$ ; S4: (-1.5,-1.5),  $b_4 = 2$

Região computacional:  $x_{\min} = -15$ ,  $x_{\max} = 15$ ;  $y_{\min} = -15$ ,  $y_{\max} = 15$

Amostragens de campo distante:  $\text{refv1} = 7.5$ ,  $\text{refv2} = 10$ ,  $\text{refv3} = 12.5$

Malha numérica:  $M_1 = -150$ ,  $M_2 = 150$ ;  $N_1 = -150$ ,  $N_2 = 150$

$h = 0.1$ ,  $dt = 0.0001$ ,  $cfl = 0.01$

Valor inicial para  $b$ : 2;

Valor médio de  $b$ : 3

Valor mínimo de  $u$  em todas as iterações:  $\text{min}_u = 2$

Valor máximo de  $u$  em todas as iterações:  $\text{max}_u = 6$

Os valores de  $u$  nas regiões:  $\pm 7.5$ ,  $\pm 10$ ,  $\pm 12.5$   
numa malha  $[-15,15] \times [-15,15]$

Run	tF	refv1	refv2	refv3	WT (segundos)
0	0	2.00000	2.00000	2.00000	0.115625E+01
8	300	2.62796	2.56971	2.53492	0.728490E+05
11	600	2.86632	2.83862	2.82192	0.719760E+05
15	1000	2.99868	2.98878	2.98273	0.714344E+05
25	2000	3.06728	3.06663	3.06619	0.713252E+05
35	3000	3.07257	3.07263	3.07262	0.100020E+05
45	4000	3.07298	3.07309	3.07311	0.997048E+04
55	5000	3.07301	3.07312	3.07315	0.101663E+05
65	6000	3.07301	3.07313	3.07315	0.983238E+04
75	7000	3.07301	3.07313	3.07315	0.986088E+04
85	8000	3.07301	3.07313	3.07315	0.100014E+05
87	8200	3.07301	3.07313	3.07315	0.100467E+05
88	8300	3.07301	3.07313	3.07315	0.101835E+05
89	8400	3.07301	3.07313	3.07315	0.972933E+04
102	9700	3.07301	3.07313	3.07315	0.998039E+04
$\infty$	$\infty$	3.07301	3.07313	3.07315	

$$\| \Delta u \|_{\max} = \max \{ |v_{i,j} - u_{i,j}| : M1 \leq i \leq M2, N1 \leq j \leq N2 \}$$

$$\Delta \text{massa} = \sum_{i=M1}^{M2} \sum_{j=N1}^{N2} (v_{i,j} - u_{i,j}) h^2$$

$$\text{media\_du/dt} = \frac{\Delta \text{massa}}{h^2(M2 - M1 + 1)(N2 - N1 + 1) \text{dt\_dump}}$$

$$\text{max\_du/dt} = \frac{\| \Delta u \|_{\max}}{\text{dt\_dump}}$$

Run	$\Delta \text{massa}$	$\  \Delta u \ _{\max}$	$\text{media\_du/dt}$	$\text{max\_du/dt}$
0	0.251875E-01	0.252946E+00	0.278004E-02	0.252946E+02
8	0.113083E-01	0.168211E-04	0.124814E-02	0.168211E-02
11	0.528586E-02	0.722756E-05	0.583421E-03	0.722756E-03
15	0.190935E-02	0.258568E-05	0.210742E-03	0.258568E-03
25	0.147455E-03	0.201462E-06	0.162752E-04	0.201462E-04
35	0.113441E-04	0.155173E-07	0.125209E-05	0.155173E-05
45	0.872489E-06	0.119357E-08	0.963001E-07	0.119357E-06
55	0.671032E-07	0.917932E-10	0.740645E-08	0.917932E-08
65	0.516143E-08	0.706102E-11	0.569687E-09	0.706102E-09
75	0.396286E-09	0.532907E-12	0.437396E-10	0.532907E-10
85	0.353585E-10	0.444089E-13	0.390266E-11	0.444089E-11
87	0.212756E-10	0.444089E-13	0.234827E-11	0.444089E-11
88	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
89	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
102	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
$\infty$	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

### 5.3.1 Gráficos do Resultado 3

Os gráficos são:

- a) o tempo de cada ciclo,
- b) o grafico de  $u$ ,
- c) os valores dos campos distantes,
- d) min e máx de  $u$ ,
- e) o gráfico de  $\Delta \text{massa}$  e
- f) o gráfico de  $\|\Delta u\|_{\max}$ .

Visto que os valores de  $\Delta \text{massa}$  e  $\|\Delta u\|_{\max}$  são muito pequenos, para facilitar a visualização desses dois gráficos, utilizamos uma escala logaritmica de base 10.

# O tempo de cada 100 atualizações e os valores de $u$ nas regiões: $\pm 7.5$ , $\pm 10$ , $\pm 12.5$

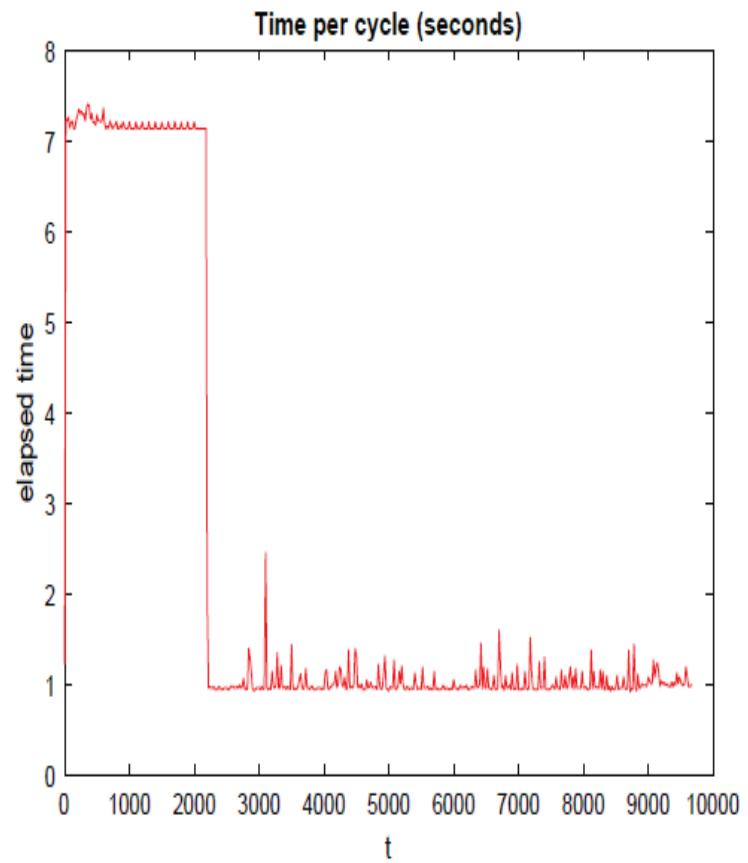


Fig. 5.3a: Cada 1 ciclo =  $0.01t = 0.01tF = 100$  atualizações de  $u$ , levou cerca de 7s no 1º computador ( $t$  de 0 a 2200) e 1s no 2º computador( $t$  de 2300 a 9700) usando malha numérica de tamanho  $301 \times 301$  e  $p = 2.1$ .

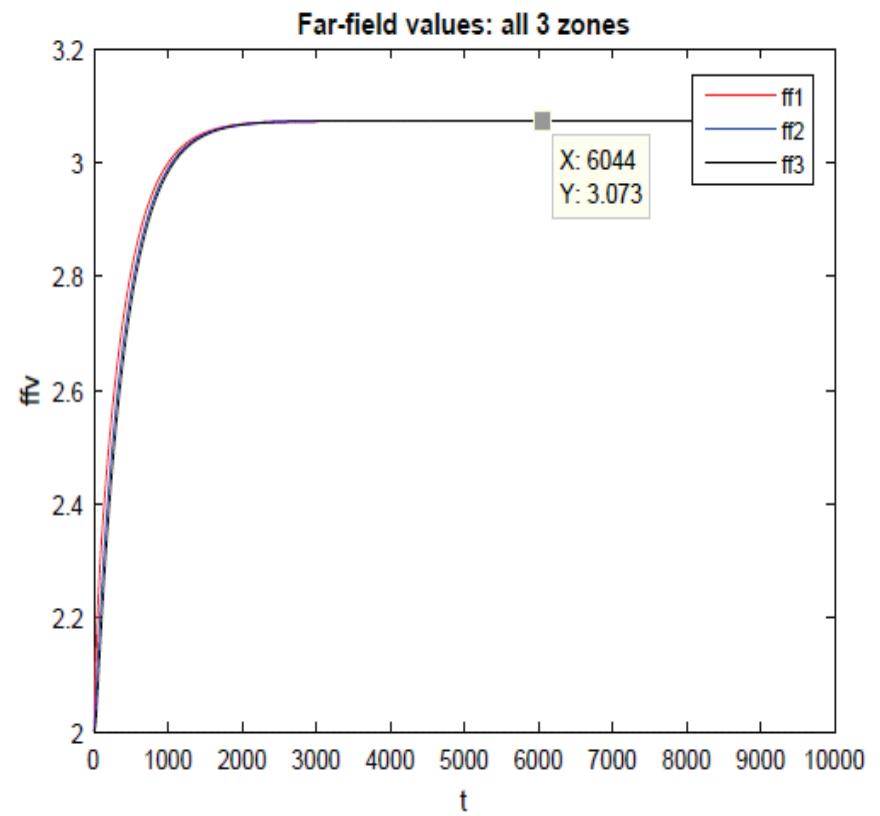
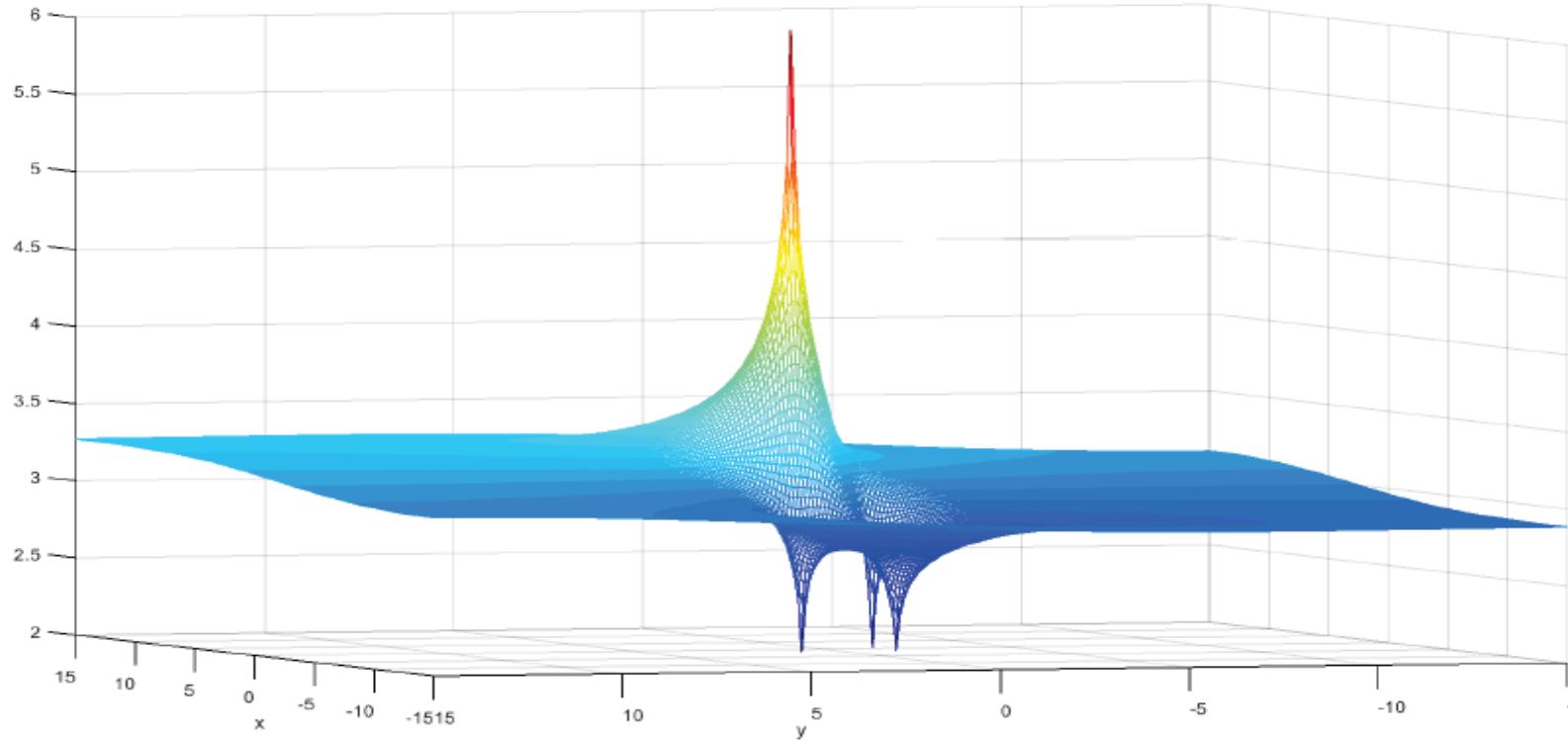
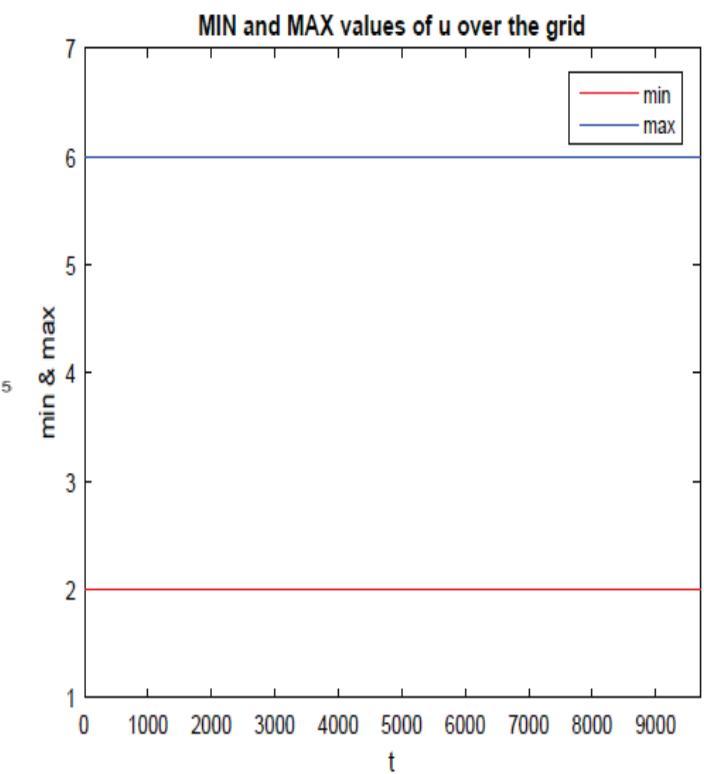


Fig. 5.3c: Os valores da média de  $u$  nas 3 regiões refv1, refv2 e refv3. Observe que a partir de  $t = 3000$  o valor de  $u$  se estabiliza nestas 3 regiões. ( $u$  praticamente não varia a partir de  $t = 3000$ ).

# O gráfico de $u$ e os valores máximos e mínimos de $u$ nas 97.000.000 atualizações



**Fig. 5.3b:** Gráfico de  $u$  para  $tF = 9700$ . Este  $u = u_k$  do problema (3) é o mesmo  $u$  do problema inicial (1) pois para  $tF = 9700$ ,  $\max_{\text{grid}} |du/dt| = \text{máximo } |du/dt| \text{ em toda malha} = 0$



**Fig. 5.3d:** Os valores mínimo e máximo de  $u$  em todas as execuções do programa.

$$\Delta \text{massa} = \sum_{i=M1}^{M2} \sum_{j=N1}^{N2} (v_{i,j} - u_{i,j}) h^2$$

$$\|\Delta u\|_{\max} = \max \left\{ |v_{i,j} - u_{i,j}| : M1 \leq i \leq M2, N1 \leq j \leq N2 \right\}$$

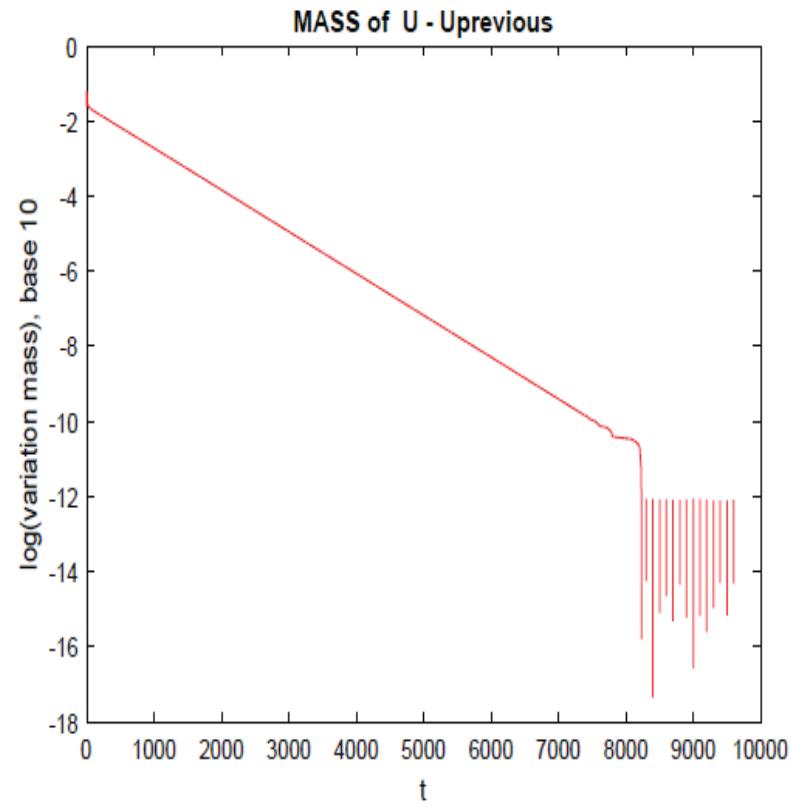


Fig. 5.3e: Visto que os valores de  $\Delta \text{massa}$  são da ordem de  $10^{-1}$  a  $10^{-10}$ , plotamos então o  $\log_{10}(\Delta \text{massa})$  de  $u_k$ . Observe que quando Run = 85,  $tF = 8000$ ,  $\Delta \text{massa} = 0.353585E - 10$  e  $\log_{10}(0.353585E - 10) = -10.45$ , o que concorda com o gráfico acima. Para  $tF > 8200$  o logaritmo vai para  $-\infty$  pois  $\Delta \text{massa} = 0$ .

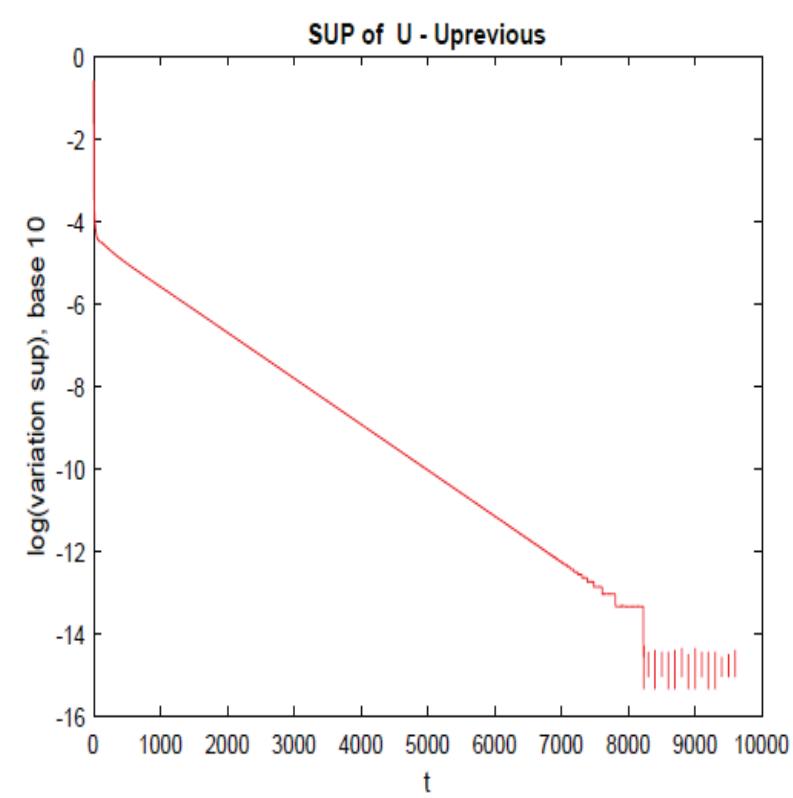


Fig. 5.3f: Os valores de  $\|\Delta u\|_{\max}$  são da ordem de  $10^{-4}$  a  $10^{-13}$ , plotamos então o  $\log_{10}(\|\Delta u\|_{\max})$  de  $u_k$ . Observe que quando Run = 85,  $tF = 8000$ ,  $\|\Delta u\|_{\max} = 0.444089E - 13$  e  $\log_{10}(0.444089E - 13) = -13.35$ , o que concorda com o gráfico acima. Para  $tF > 8200$  o logaritmo vai para  $-\infty$  pois  $\|\Delta u\|_{\max} = 0$ .

u24h\_omp10.f90 [8TExtraSize0] - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

initialize.f90 X u24h\_omp10.f90 X

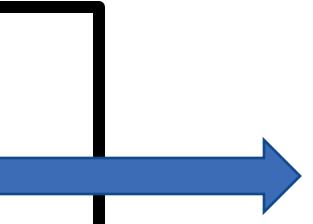
```
771 DO WHILE ( t < tF_check )
772     sum_cputime = 0D0;
773     DO external_iteration_count = 1, no_external_iterations
774         CALL CPU_TIME(time_start)
775         !$OMP PARALLEL DEFAULT(None), SHARED(M1,M2,N1,N2,extra_size,main_chunk_size,Nthreads, &
776             q,u,cfl,h_to_pm2,time_start,b1,b2,b3,b4,j1,j2,j3,j4, &
777             omp_iterations,time_finish,sum_cputime, Qt_Atualizacoes_u) , &
778             PRIVATE(tid,ic0,icF,uc,vc,Fc,Gc)
779         tid = OMP_GET_THREAD_NUM();
780         i0 = M1 + tid*main_chunk_size;
781         iK = i0 + main_chunk_size;
782         if ( tid > 0 .AND. tid < Nthreads-1 ) then
783             uc(-extra_size:main_chunk_size+extra_size, N1-1:N2+1)&
784                 = u(i0-extra_size:iK+extra_size, N1-1:N2+1);
785             ic0 = -extra_size + 1; ! <--- local indexing in chunk tid
786             icF = main_chunk_size + extra_size - 1; ! goes from ic0-1 to icF+1
787             ! **** computation of Fc values on the local grid: !
788             ! **** Gc values on the local grid: !
789             ! **** Fc(i,j) for ic0 <= i <= icF + 1 and N1 <= j <= N2:
790             Fc(ic0:icF+1,N1:N2) = ( (uc(ic0:icF+1,N1:N2) - uc(ic0-1:icF,N1:N2))**2 + &
791                 ( (uc(ic0-1:icF,N1+1:N2+1)+uc(ic0:icF+1,N1+1:N2+1)) - &
792                     (uc(ic0-1:icF,N1-1:N2-1)+uc(ic0:icF+1,N1-1:N2-1)) )**2 /16 )**q / h_to_pm2;
793             ! **** computation of Gc values on the local grid: !
794             ! **** Gc(i,j) for ic0 <= i <= icF and N1 <= j <= N2 + 1:
795             Gc(ic0:icF,N1:N2+1) = ( (uc(ic0:icF,N1:N2+1) - uc(ic0:icF,N1-1:N2))**2 + &
796                 ( (uc(ic0+1:icF+1,N1-1:N2)+uc(ic0+1:icF+1,N1:N2+1)) - &
797                     (uc(ic0-1:icF-1,N1-1:N2)+uc(ic0-1:icF-1,N1:N2+1)) )**2 /16 )**q / h_to_pm2;
798             ! **** computation of vc = [ new uc values at the new time level ]:
799             ! **** vc(i,j) for ic0 <= i <= icF and N1 <= j <= N2:
800             vc(ic0:icF,N1:N2) = uc(ic0:icF,N1:N2) + &
801                 cfl*( Fc(ic0+1:icF+1,N1:N2)*(uc(ic0+1:icF+1,N1:N2)-uc(ic0:icF,N1:N2)) - &
802                     Fc(ic0:icF,N1:N2)*(uc(ic0:icF,N1:N2)-uc(ic0-1:icF-1,N1:N2)) ) + &
803                     cfl*( Gc(ic0:icF,N1+1:N2+1)*(uc(ic0:icF,N1+1:N2+1)-uc(ic0:icF,N1:N2)) - &
804                         Gc(ic0:icF,N1:N2)*(uc(ic0:icF,N1:N2)-uc(ic0:icF,N1-1:N2-1)) );
805             ! **** extending vc to the extra grid points:
806             vc(ic0:icF,N1-1) = vc(ic0:icF,N1);
807             vc(ic0:icF,N2+1) = vc(ic0:icF,N2);
808             .
809             .
810             .
811             .
812             .
813             .
```

F:\Universidade\2020 02\Computação Paralela\Trabalho 23Mar\OpenMP Sem Bloco E... Fortran Windows (CR+LF) WINDOWS-1252 Line 810, Col 67, Pos 27082 Insert Read/Write default

**Comparativo: As colunas 3 a 7 estão em segundos.  
Na 2<sup>a</sup> coluna cada 1 TF = 10 mil atualizações de  $u$**

Linha	T0 a TF	Serial(s)	4Thread sem Bloco Extra	8Thread com Bloco Extra	8Thread sem Bloco Extra	16Thread sem Bloco Extra
1	0 a 0.1	3	1	1	1	2
2	0.2 a 0.5	7	3	4	2	4
3	0.5 a 1	11	6	6	3	5
4	1 a 2	20	10	12	5	8
5	2 a 4	32	15	25	9	13
6	4 a 8	53	22	33	15	22
7	10 a 20	123	51	71	33	51
8	20 a 50	366	156	215	96	156
9	50 a 100	608	241	353	172	200
10	100 a 200	1202	469	725	366	454
11	200 a 300	1202	435	746	372	433
12	300 a 400	1210	434	750	379	447

## Conclusão na dissertação:



1) Transformar o problema original, estático:  $\Delta_p u = 0$ , no p-Laplaciano evolutivo:  $u_t = \Delta_p u$ , e resolver o p-Laplaciano evolutivo por meio de diferenças finitas, “deu certo”, pois todas as 10 simulações convergiram para o problema original, isto é, com estas duas mudanças foi possível usar o computador para descobrir o comportamento da função procurada.

2) Os experimentos(resultados) 1 e 2 apresentam mesma configuração, apenas alturas iniciais diferentes, em ambos, os experimentos convergiram para o mesmo resultado:

$$b_\infty := \lim_{|x| \rightarrow \infty} u(x) = \lim_{k \rightarrow \infty} \left[ \lim_{|x| \rightarrow \infty} u_k(x) \right] \quad (5.7)$$

basta olhar na última linha das tabelas na página 77 e 79. Ou seja, o valor inicial do plano, a configuração inicial, não altera o resultado final, não altera o valor final da convergência  $b_\infty$ .

3) Por outro lado os experimentos 4 e 5 possuem a mesma localização das 4 singularidades e o mesmo

$$b_m := \frac{b_1 + b_2 + b_3 + b_4}{4} = 2.75 \quad (5.8)$$

Mas olhando na última linha das tabelas na página 86 e 88, percebemos claramente, que ambos convergem para resultados diferentes de  $b_\infty$ . O que mostra que a **Observação 2.1** está correta, ou seja, o valor de  $b_\infty$  não é  $b_m$  (a média aritmética das alturas das singularidades).

4) Já o experimento 7, mostra que provavelmente o **Teorema 2.2** está correto, isto é,  $b_\infty$  é a média das alturas das singularidades quando temos somente duas singularidades.

5) Todos os 10 experimentos, levaram várias semanas ou meses para chegar ao resultado final. Isto mostra claramente a necessidade de usar outros computadores em paralelo, para acelerar a velocidade do processamento, ou seja nos próximos trabalhos, devemos usar MPI, ou MP com MPI, ou outra biblioteca de processamento paralelo que use vários computadores simultaneamente.

Maiores detalhes em:

<https://github.com/Roberto-Hoo/Tese-RobertoHoo-PZingano-UFRGS-Matematica-2018>

# Conclusão do trabalho de Computação Paralela

- 1) Inicialmente na dissertação foi usado apenas o código serial e o código com OpenMP com blocos extras de tamanho 50(primeiro método). Foi usado os computadores do Cenapad(Campinas-Brasil), Global-Okeanos(Grécia) e Grid5000(França)
- 2) Tentei várias vezes implementar o segundo método(sem bloco extra) mas sempre era mais lento que o primeiro método. Agora neste trabalho de Computação paralela, estudando melhor as diretivas de OpenMP: - **critical, shared, private, master, for, etc..-** consegui implementar uma versão do segundo método(sem os blocos extras) cujos testes acima se mostraram mais rápidos que com blocos extras.
- 3) O entendimento detalhado e profundo das várias diretivas de OpenMP, leva tempo e prática.
- 4) Na último parágrafo da dissertação foi dito que a computação daqueles resultados levaram vários meses, logo é imperativo o uso de MPI ou MPI + OpenMP em projetos similares como este no futuro.

# Referências:

- 1) <https://phkonzen.github.io/notas/MatematicaNumericaParalela/main.html>
- 2) <https://phkonzen.github.io/notas/>
- 3) <https://github.com/Roberto-Hoo/Trabalho-1-Computacao-paralela-23Mar2021>
- 4) <https://github.com/Roberto-Hoo/Tese-RobertoHoo-PZingano-UFRGS-Matematica-2018>