7
Resultados

Nesta seção mostraremos alguns dos resultados obtidos para três problemas diferentes.

Para cada problema será preciso conhecer o valor da pressão p nas bordas (condição de contorno) e a função $\lambda(x, y)$.

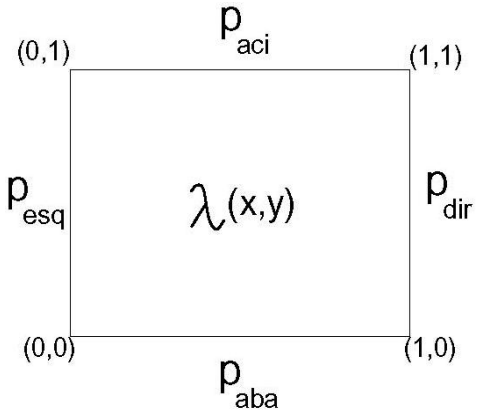

Figura 7.1: Domínio quadrado de testes.

O número de processadores será indicado por P, o número de camadas por C, o número de iterações por $NIter$, o número de iterações da camada grossa dividido pela camada fina por M, a razão linear (em cada eixo) entre as camadas (fina\grossa) por R e teremos $N = mn$.

7.1
Análise de Escalabilidade

Ao limitar o número de iterações na camada mais fina alcançamos escalabilidade pois, mantendo fixo o valor de R a freqüência não resolvida pelo problema grosso permanece a mesma porque ela está relacionada com a razão $R = \frac{H}{h}$ (ver [8]). Assim, para aumentar a resolução do problema resolvendo-o com a mesma duração, basta aumentar na mesma proporção o número de processadores e, sendo igual o tamanho do problema em cada processador, o novo problema será resolvido no mesmo tempo.

Nesta seção mostraremos através de alguns exemplos rodados em um cluster de PCs que o algoritmo usado no problema serial é adequado para que o problema paralelo possa ser escalável.

Três modelos de problemas com soluções conhecidas serão usados para auferir os resultados:

$Prob1 : p = 20 - 40x$ e $\lambda = 1$,
$Prob2 : p = x^3y^4 + x^2 \sin(xy)\cos(y)$ e $\lambda = 1$,
$Prob3 : p = x^3y^4 + x^2 \sin(xy)\cos(y)$ e $\lambda = e^{-x-y}$.

Nas próximas páginas mostraremos gráficos que nos ajudarão a interpretar os resultados.

$Prob1$: $P = 1, C = 1$:		$P = 1, C = 2, R = 4$:		
N	NIter	N	NIter	M
16	41	16	33	4
32	89	32	46	16
64	178	64	47	46
128	347	128	47	98
256	667	256	47	195
512	1275	512	47	378

Tabela 7.1: Iterações para $Prob1$.

Vejamos o gráfico:

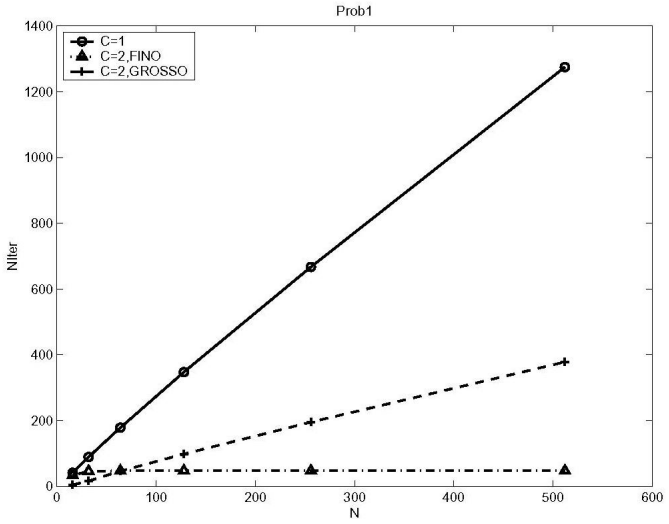

Figura 7.2: Iterações problema 1.

$Prob2:$ $P=1, C=1:$		$P=1, C=2, R=4:$		
N	NIter	N	NIter	M
16	25	16	25	7
32	55	32	39	14
64	116	64	48	17
128	222	128	46	28
256	229	256	40	53
512	603	512	33	94

Tabela 7.2: Iterações para $Prob2$.

Vejamos o gráfico:

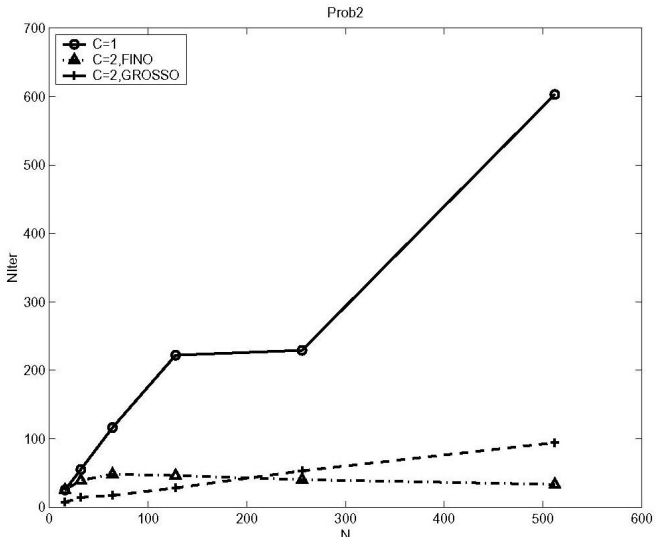

Figura 7.3: Iterações problema 2.

$Prob3$:				
$P = 1, C = 1$:		$P = 1, C = 2, R = 4$:		
N	NIter	N	NIter	M
16	58	16	43	9
32	125	32	51	22
64	255	64	50	42
128	511	128	48	69
256	984	256	42	112
512	1879	512	35	181

Tabela 7.3: Iterações para $Prob3$.

Vejamos o gráfico:

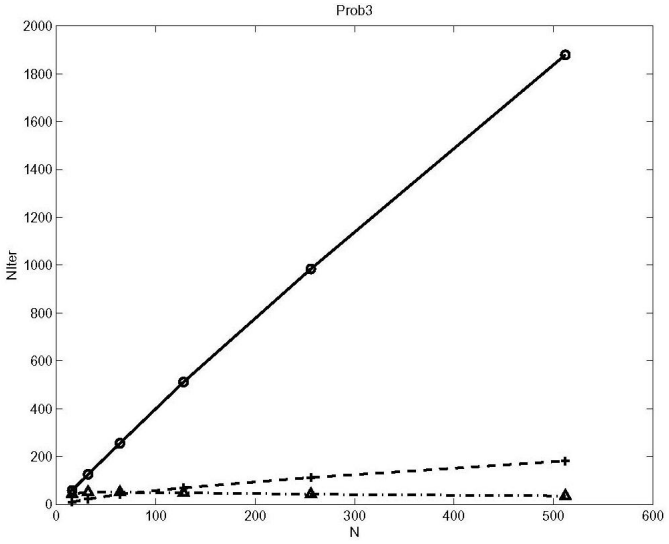

Figura 7.4: Iterações problema 3.

Como podemos obserar nos valores mostrados acima, o número de iterações permanece praticamente fixo a partir de um determinado tamanho que depende do tipo de problema a ser resolvido, desde que a razão R seja mantida fixa. Isto permitirá que o algoritmo paralelo alcance uma boa escalabilidade. Somente o número de iterações do problema grosso cresce à medida que a malha é aumentada; isso permite que, crescendo na mesma proporção o número de processadores, o problema seja resolvido no mesmo tempo.

7.2
Análise do Tempo

Esta seção tem como objetivo mostrar alguns resultados que confirmam o quanto o uso do precondicionador pode melhorar o tempo de processamento do Gradiente Conjugado. Em três exemplos veremos que a porcentagem de tempo usado em relação ao problema sem precondicionadores decai rápido conforme cresce o tamanho da malha.Para cada exemplo, foi usada a camada grossa com a razão R que mais rápido resolve o problema proposto, onde R indica a razão entre o tamanho do problema fino e o problema grosso e $Tempo$ indica a porcentagem do tempo gasto em relação ao problema resolvido sem o uso de uma camada grossa. Abaixo vemos os valores encontrados e os respectivos gráficos:

$Prob1:$		
$P = 1, C = 2:$		
N	$Tempo$	R
16	0.95	4
32	0.87	4
64	0.47	8
128	0.28	8
256	0.30	8
512	0.20	16

Tabela 7.4: Tempo para $Prob1$.

Vejamos o gráfico:

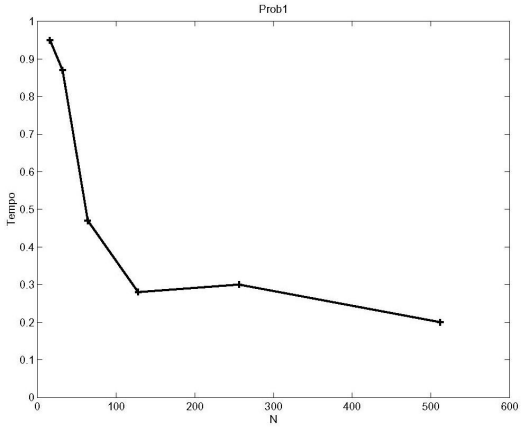

Figura 7.5: Tempo problema 1.

Neste gráfico vemos que o algoritmo a partir de $N = 128$ cresce para depois voltar a diminuir a partir de 256.

$Prob2$:
$P = 1, C = 2$:

N	$Tempo$	R
16	0.99	4
32	0.97	4
64	0.38	8
128	0.43	8
256	0.56	16
512	0.35	16

Tabela 7.5: Tempo para $Prob2$.

Vejamos o gráfico:

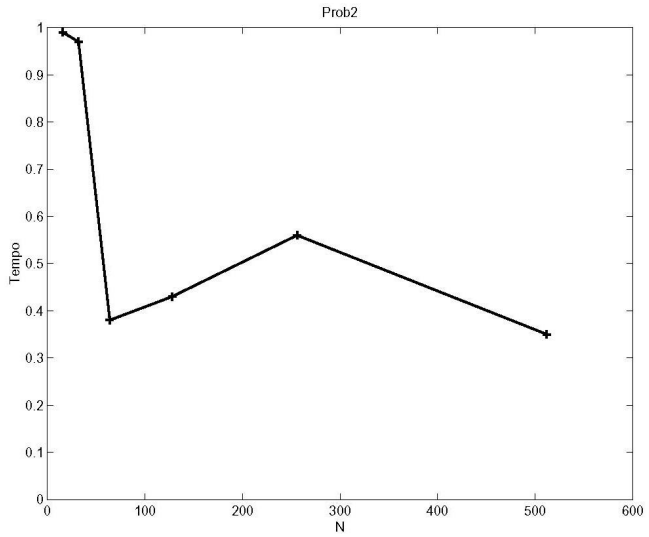

Figura 7.6: Tempo problema 2.

Neste gráfico vemos que o algoritmo a partir de $N = 64$ cresce para depois voltar a diminuir a partir de 256.

$Prob3:$		
$P = 1, C = 2:$		
M	$Tempo$	R
16	0.96	4
32	0.85	4
64	0.42	8
128	0.20	8
256	0.24	8
512	0.11	16

Tabela 7.6: Tempo para $Prob3$.

Vejamos o gráfico:

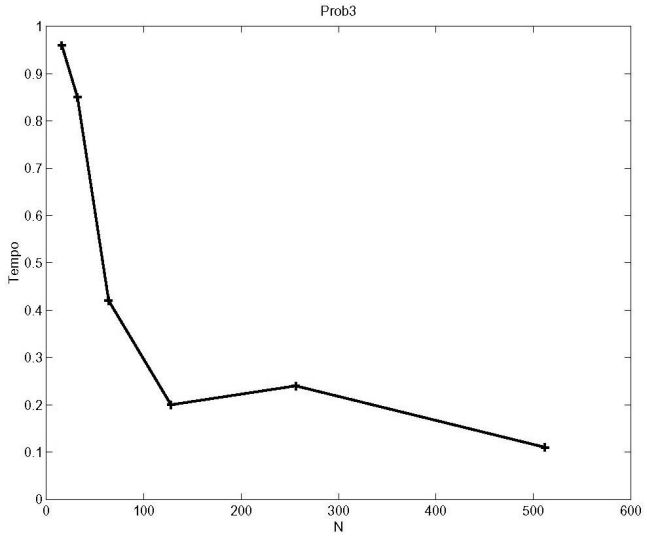

Figura 7.7: Tempo problema3.

Nos três gráficos o crescimento a partir de $N = 64$ e/ou 128 deve-se ao fato de que o problema já não cabe inteiramente na memória cache, diminuindo o ritmo de crescimento até $N = 256$ quando volta a diminuir devido a uma menor influência de memórias rápidas (praticamente todas as variáveis deverão passar pela memória mais lenta).

7.3 Análise do Processamento Paralelo

Nesta seção veremos o quanto escalável está o algoritmo através de alguns exemplos executados em um cluster de 16 nós iguais. O *Tempo* é dado em minutos.

$Prob1, C = 2:$

P	N	R	Tempo
1	512	16	15
4	512	16	7
1	1024	8	310
4	1024	8	50
16	1024	8	26
4	2048	8	482
16	2048	8	103

;

$Prob2, C = 2:$

P	N	R	Tempo
1	512	16	12
4	512	16	5
1	1024	8	235
4	1024	8	24
16	1024	8	8
4	2048	8	265
16	2048	8	34

;

$Prob3, C = 2:$

P	N	R	Tempo
1	512	16	32
4	512	16	18
1	1024	8	336
4	1024	8	63
16	1024	8	32
4	2048	8	477
16	2048	8	98

.

Tabela 7.7: Processamento Paralelo.

Nas próximas páginas veremos os gráficos:

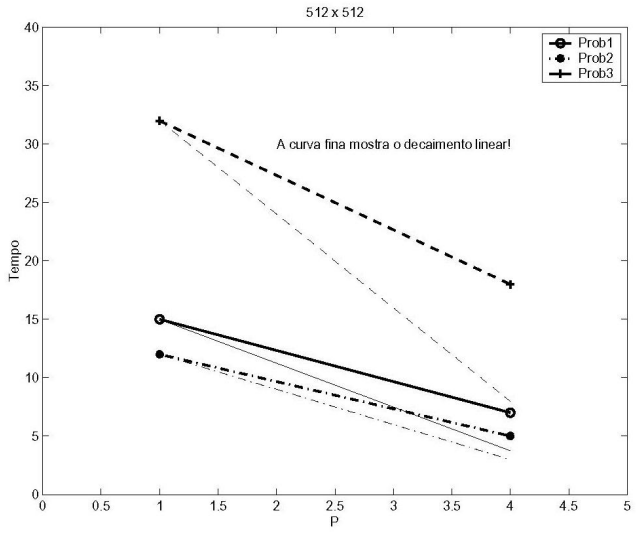

Figura 7.8: Paralelo malha 512×512.

Neste gráfico vemos que o número de processadores diminui quase linearmente o tempo de processamento para o tamanho considerado: 512 x 512.

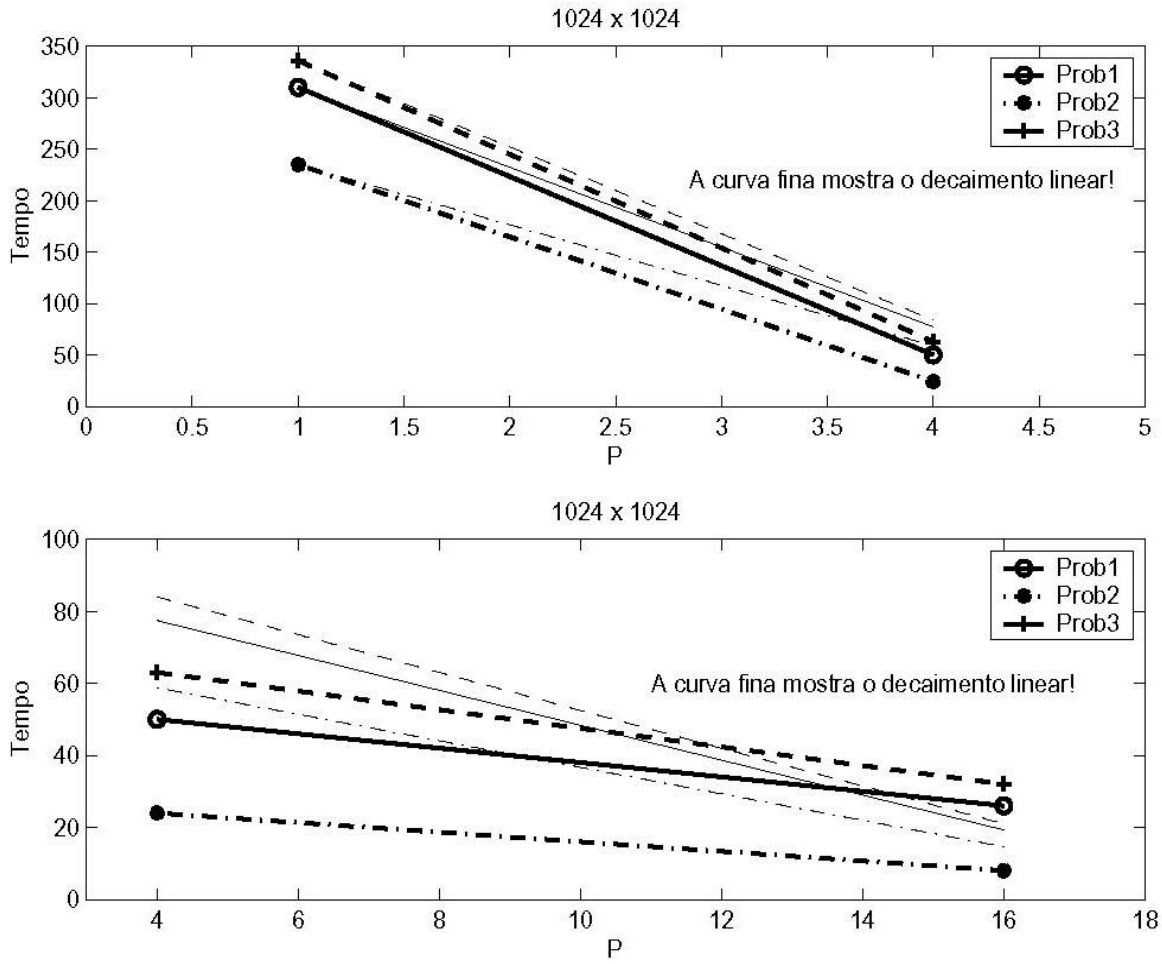

Figura 7.9: Paralelo malha 1024 × 1024.

Neste gráfico vemos que o número de processadores diminui quase linearmente o tempo de processamento para o tamanho considerado: 1024 x 1024.

No primeiro caso (aumento de 1 para 4) o ganho é maior que o linear devido ao melhor aproveitamento da memória. No segundo caso (aumento de 4 para 16) o ganho piora em relação ao caso linear pois o tempo de comunicação começa a custar muito devido à quantidade grande de processadores.

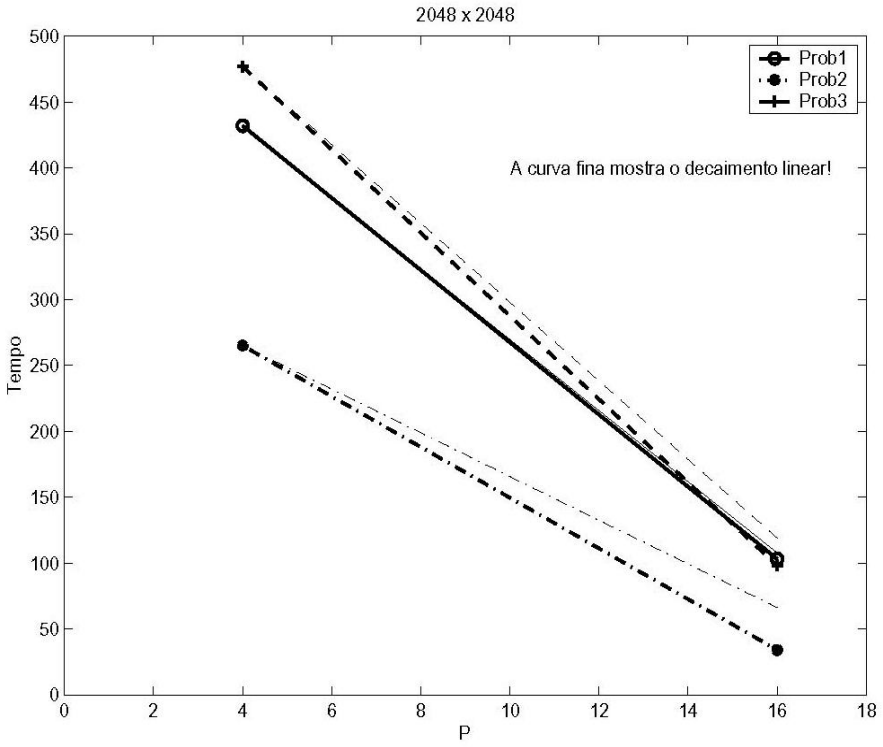

Figura 7.10: Paralelo malha 2048 × 2048.

Neste gráfico o aumento de 4 para 16 processadores não impede o ganho superlinear pois sendo uma malha bastante grande o custo de comunicação para 16 processdores fica relativamente pequeno.

Destes dados podemos avaliar que a escalabilidade é melhor observada nos problemas maiores pois o tempo de comunicação torna-se pouco em relação ao total. O tempo de processamento nestes casos é então inversa e linearmente proporcional ao número de processadores.

7.4
Ordem de Precisão

Nesta seção vamos analisar qual a ordem de convergência alcançada pelo algoritmo. Serão usados $Prob2$ e $Prob3$ já definidos. Para gerar as equações que nos mostra a relação entre a norma do erro e h, consideraremos $h = h_x = h_y$ e $L_x = L_y = 1$.

$Prob2, C = 1$:

N	Tempo
4	12
8	5
16	235
32	24
64	8
128	265
256	34

,

$Prob3, C = 1$:

N	Tempo
4	12
8	5
16	235
32	24
64	8
128	265
256	34

Tabela 7.8: Ordem de Precisão.

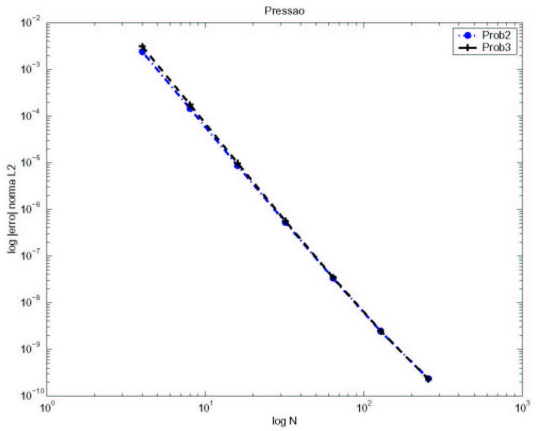

Figura 7.11: Precisão pressão.

Equações: $Prob2 : |erro|_{L^2} = (\frac{0.4714}{h})^{3.9138}$, $Prob3 : |erro|_{L^2} = (\frac{0.659}{h})^{3.9822}$.

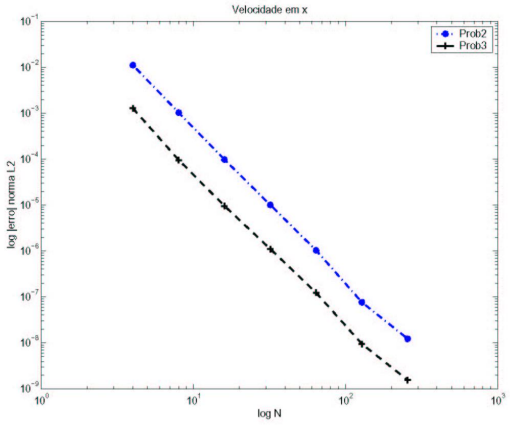

Figura 7.12: Precisão velocidade em x.

Equações: $Prob2 : |erro|_{L^2} = (\frac{1.057}{h})^{3.3337}$, $Prob3 : |erro|_{L^2} = (\frac{0.0981}{h})^{3.2809}$.

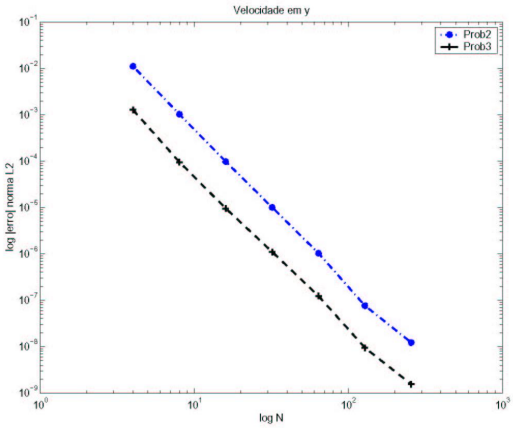

Figura 7.13: Precisão velocidade em y.

Equações: $Prob2 : |erro|_{L^2} = (\frac{1.057}{h})^{3.3337}$, $Prob3 : |erro|_{L^2} = (\frac{0.0981}{h})^{3.2809}$.

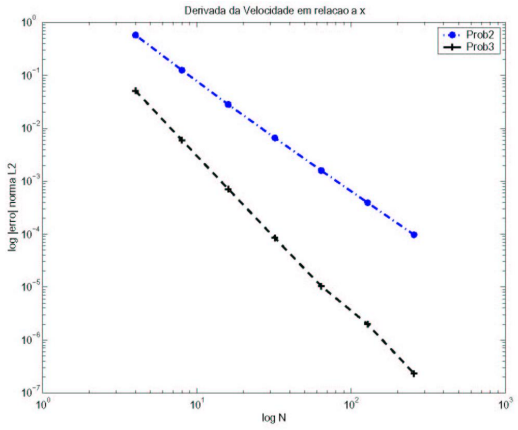

Figura 7.14: Precisão derivada da velocidade em x.

Equações: $Prob2 : |erro|_{L^2} = (\frac{9.5956}{h})^{2.0847}$, $Prob3 : |erro|_{L^2} = (\frac{2.6472}{h})^{2.9439}$.

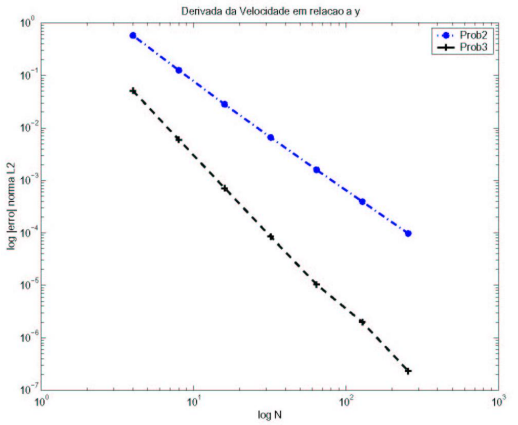

Figura 7.15: Precisão derivada da velocidade em y.

Equações: $Prob2 : |erro|_{L^2} = (\frac{9.5956}{h})^{2.0847}$, $Prob3 : |erro|_{L^2} = (\frac{2.6472}{h})^{2.9439}$.

Embora ainda não seja possível demonstrar a ordem de convergência do algoritmo devido às várias aproximações feitas para diminuir os custos da execução doprograma, obtivemos para os casos analisados entre $O(h^3)$ e $O(h^4)$ para as derivadas das velocidades x, y, o que permite encontrar a saturação s com boa precisão, exigência para as equações relacionadas à Engenharia de Petróleo.