

CALCOLO SCIENTIFICO

Project 3

Student
Pranav KASELA
Roll no. 866261

Student
Roberto CAVASSI
Roll no. 868504

June 2, 2018

Contents

Esercizio 1	1
Exercise 2	7
Exact Solution	7
Upwind Finite Difference Scheme	7
MATLAB Implementation	8
Scharfetter-Gummel Scheme	13
MATLAB Implementation	13
MATLAB Codes	18
Exercise 1 Code	18
$\Delta t_{critico}$ Code	20
$P_{critico}$ Code	20
Exercise 2 Code	21
bern.m	23

Esercizio 1

In questo esercizio viene chiesto di analizzare la seguente equazione di diffusione:

$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} - D \frac{\partial^2 u}{\partial x^2} = 0 \quad x \in (0,1), t \geq 0$$

Con $\beta, D > 0$, costanti. Ed il relativo problema di Cauchy avente condizioni al bordo di Dirichlet in $x=0,1$ e condizioni iniziali tali che la soluzione esatta sia:

$$u(x, t) = e^{-4\pi Dt} \sin(2\pi(x - \beta t))$$

Utilizzando il metodo di Eulero esplicito per discretizzare la parte temporale e il metodo delle differenze finite centrate per la parte spaziale, otteniamo la seguente approssimazione discreta:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = -\beta \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} + D \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}$$

Supponiamo ora che la soluzione sia combinazione lineare di funzioni della forma $A_k^n e^{ik\pi j \Delta x}$, con k che rappresenta il numero d'onda. Per determinare il valore A_k sostituendo semplicemente la soluzione cercata all'interno della precedente uguaglianza, otteniamo quindi:

$$\begin{aligned} \frac{A_k^{n+1} e^{ik\pi j \Delta x} - A_k^n e^{ik\pi j \Delta x}}{\Delta t} \\ = -\beta \frac{A_k^n e^{ik\pi(j+1)\Delta x} - A_k^n e^{ik\pi(j-1)\Delta x}}{2\Delta x} + D \frac{A_k^n e^{ik\pi(j+1)\Delta x} - 2A_k^n e^{ik\pi j \Delta x} + A_k^n e^{ik\pi(j-1)\Delta x}}{\Delta x^2} \end{aligned}$$

Raccogliendo e semplificando $A_k^n e^{ik\pi j \Delta x}$, che è sempre diverso da 0 (a meno di $A_k = 0$), si ottiene:

$$\begin{aligned} \frac{A_k - 1}{\Delta t} &= -\beta \frac{e^{ik\pi\Delta x} - e^{-ik\pi\Delta x}}{2\Delta x} + D \frac{e^{ik\pi\Delta x} - 2 + e^{-ik\pi\Delta x}}{\Delta x^2} \\ A_k &= 1 - \frac{\beta\Delta t}{2\Delta x} 2i \operatorname{sen}(k\pi\Delta x) + \frac{D\Delta t}{\Delta x^2} (2 \cos(k\pi\Delta x) - 2) \end{aligned}$$

Utilizzando i gruppi adimensionali $C = \frac{\beta\Delta t}{\Delta x}$, $Pe = \frac{\beta\Delta x}{2D}$, si ottiene:

$$\begin{aligned} A_k &= 1 + \frac{C}{Pe} (\cos(k\pi\Delta x) - 1) - iC \operatorname{sen}(k\pi\Delta x) \\ A_k &= 1 - \frac{2C}{Pe} \operatorname{sen}^2\left(\frac{k\pi\Delta x}{2}\right) - iC \operatorname{sen}(k\pi\Delta x) \end{aligned}$$

A_k quindi è un numero complesso, il cui valore varia al variare dei coefficienti C , Pe e Δx .

Affinché questo schema sia stabile è necessario che $|A_k| \leq 1$ altrimenti, avanzando nel tempo, il sistema divergerà all'infinito, dato che corrisponde a prendere potenze sempre maggiori di A_k .

Determino quindi per quali condizioni tra C e Pe ottengo la stabilità.

Innanzitutto, dato che A_k è un numero complesso, determino quando il modulo della sua parte immaginaria e di quella reale sono minori di 1.

$$|Im(A_k)| = |-C \operatorname{sen}(k\pi\Delta x)|$$

Dato che k appartiene a Z , ottengo quindi la disequazione:

$$|Im(A_k)| \leq |C|$$

Per definizione C è positivo, quindi la prima condizione è $C \leq 1$.

Per la seconda analizzo la parte reale.

$$|Re(A_k)| = \left| 1 - \frac{2C}{Pe} \sin^2\left(\frac{k\pi\Delta x}{2}\right) \right|$$

Otteniamo due disequazioni, la prima è:

$$1 - \frac{2C}{Pe} \sin^2\left(\frac{k\pi\Delta x}{2}\right) \leq 1$$

$$-\frac{2C}{Pe} \sin^2\left(\frac{k\pi\Delta x}{2}\right) \leq 0$$

Che è sempre vero, dato che sia C che Pe sono coefficienti positivi. La seconda è

$$\frac{2C}{Pe} \sin^2\left(\frac{k\pi\Delta x}{2}\right) - 1 \leq 1$$

$$\frac{2C}{Pe} \sin^2\left(\frac{k\pi\Delta x}{2}\right) \leq 2$$

$$\frac{C}{Pe} \sin^2\left(\frac{k\pi\Delta x}{2}\right) \leq 1$$

Dato che k può assumere qualsiasi valore intero si ottiene:

$$\frac{C}{Pe} \leq 1$$

$$C \leq Pe$$

La seconda condizione è quindi: $C \leq Pe$.

Analizzo ora la norma:

$$|A_k| = \left(1 - \frac{2C}{Pe} \sin^2\left(\frac{k\pi\Delta x}{2}\right) \right)^2 + (C \sin(k\pi\Delta x))^2 \leq 1$$

Sviluppando i seni al primo ordine si ottiene:

$$\left(1 - \frac{2C}{Pe} \left(\frac{k\pi\Delta x}{2} \right)^2 \right)^2 + (Ck\pi\Delta x + o(\Delta x))^2 \leq 1$$

$$1 + \frac{4C^2}{Pe^2} \left(\frac{k\pi\Delta x}{2} \right)^4 - \frac{4C}{Pe} \left(\frac{k\pi\Delta x}{2} \right)^2 + C^2(k\pi\Delta x)^2 + o(\Delta x^2) \leq 1$$

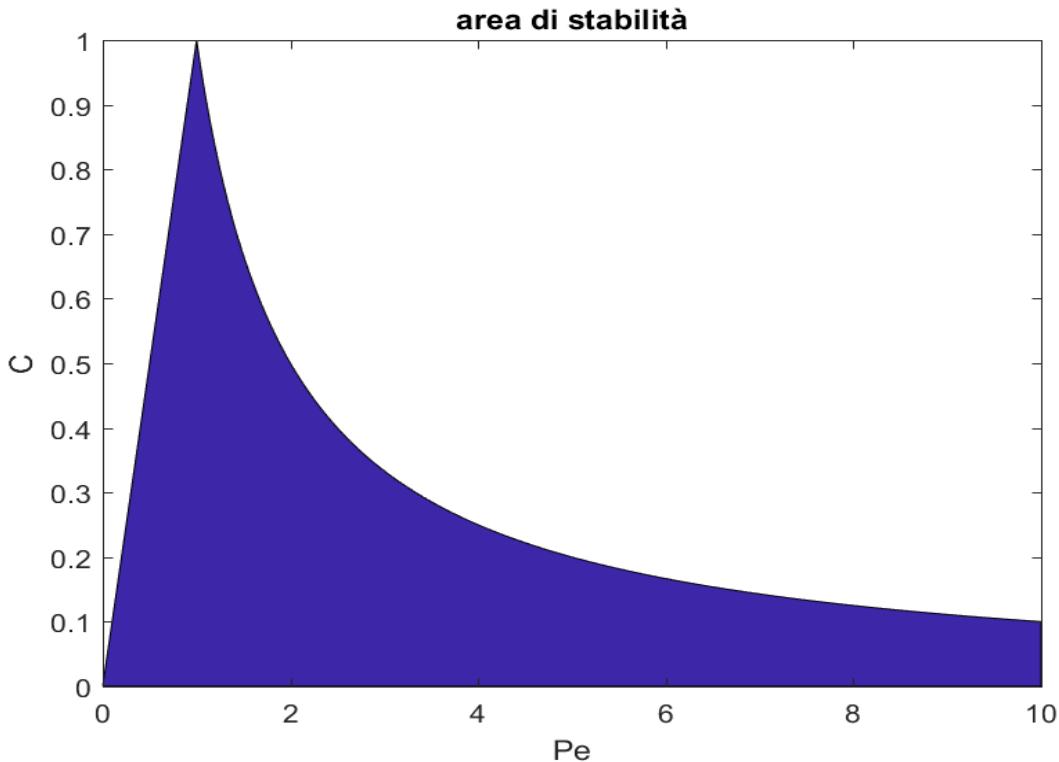
$$\left(C^2 - \frac{C}{Pe} \right) (k\pi\Delta x)^2 \leq 0$$

Dato che C e $(k\pi\Delta x)^2$ sono positivi, si ottiene:

$$C - \frac{1}{Pe} \leq 0$$

$$C \leq \frac{1}{Pe}$$

Ottengo quindi una regione di piano (Pe, C) in cui il metodo è stabile. In particolare la regione è $C \leq Pe$ per $Pe \leq 1$, e $C \leq 1/Pe$ per $Pe \geq 1$.



A questo punto ho implementato il problema di Cauchy, fissando $D=1$. Ho utilizzato lo schema mostrato precedentemente, ovvero:

$$u_j^{n+1} = u_{j-1}^n \left(\frac{\Delta t}{\Delta x^2} + \frac{\beta \Delta t}{2 \Delta x} \right) + u_j^n \left(1 - \frac{2 \Delta t}{\Delta x^2} \right) + u_{j+1}^n \left(\frac{\Delta t}{\Delta x^2} - \frac{\beta \Delta t}{2 \Delta x} \right)$$

Per ricavare i dati iniziali e le condizioni di Dirichlet al bordo utilizzo la soluzione esatta, valutata nei nodi interessati.

Per analizzare il comportamento della soluzione numerica al variare di $\beta, \Delta x, \Delta t$ definisco la funzione 'dt_critico.m'. Questa funzione prende in entrata β e Δx , e calcola il Δt massimo tale per cui il punto (Pe, C) associato appartenga alla regione sopra definita, ovvero il metodo sia stabile. Inoltre sappiamo che, dato che C è proporzionale a Δt , per tutti i valori maggiori di quello trovato dalla funzione il metodo è instabile, mentre per valori minori continua ad essere stabile.

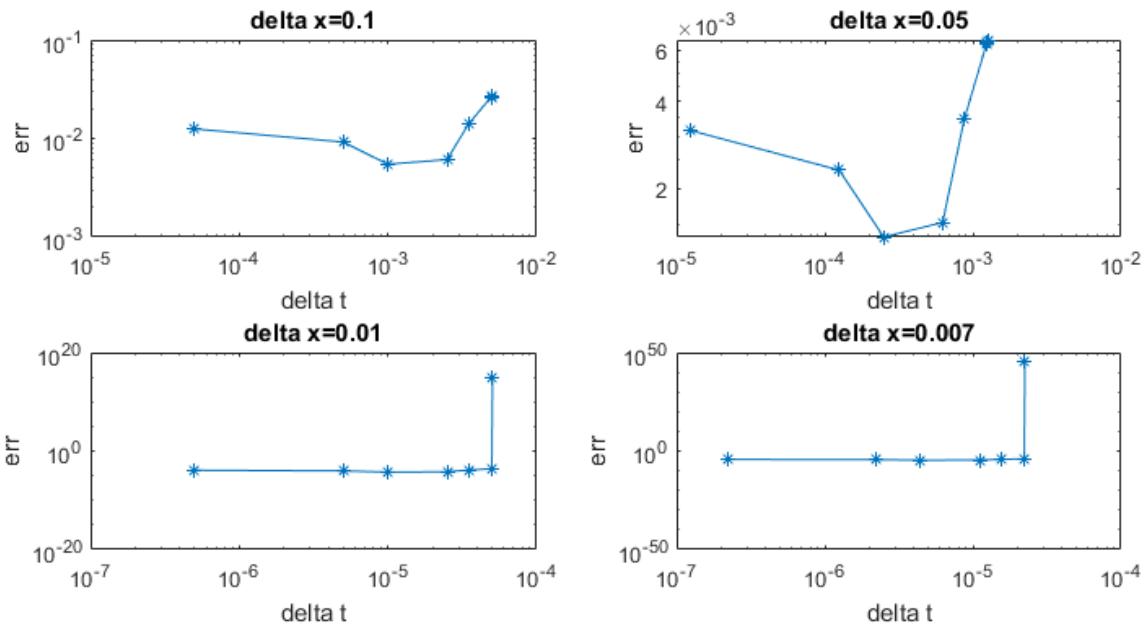
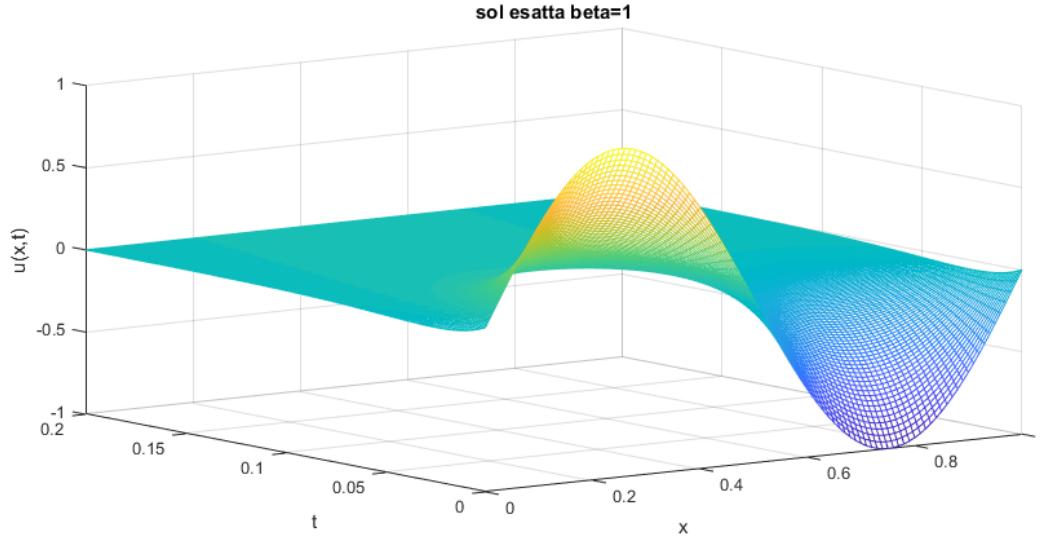
Per calcolarlo la funzione innanzitutto calcola il Pe associato ai dati che riceve, dopodiché calcola C massimo e infine ricava Δt in base alla definizione di C .

Infine, per verificare le condizioni di stabilità trovate precedentemente procedo in questo modo: fisso β , dopodiché calcolo il Δt critico per alcuni Δx , ed analizzo l'errore tra la soluzione esatta e quella numerica per vari multipli dell'intervallo temporale: uno maggiore di poco a quello limite, il valore limite, uno di poco inferiore e infine un centesimo del valore limite.

Nel momento in cui ho implementato il problema pongo il tempo finale a 0,15 per avere più libertà nella scelta dei passi. Rispetto alla verifica della stabilità questa scelta permette di avere degli errori non troppo elevati e al calcolatore di fare tutti i conti necessari.

Il primo β che analizzo è 1, ovvero il caso in cui non prevale né la natura diffusiva del problema, né quella convettiva.

La soluzione esatta è:



Gli errori analizzati sono:

Per la precisione l'errore assume questi valori:

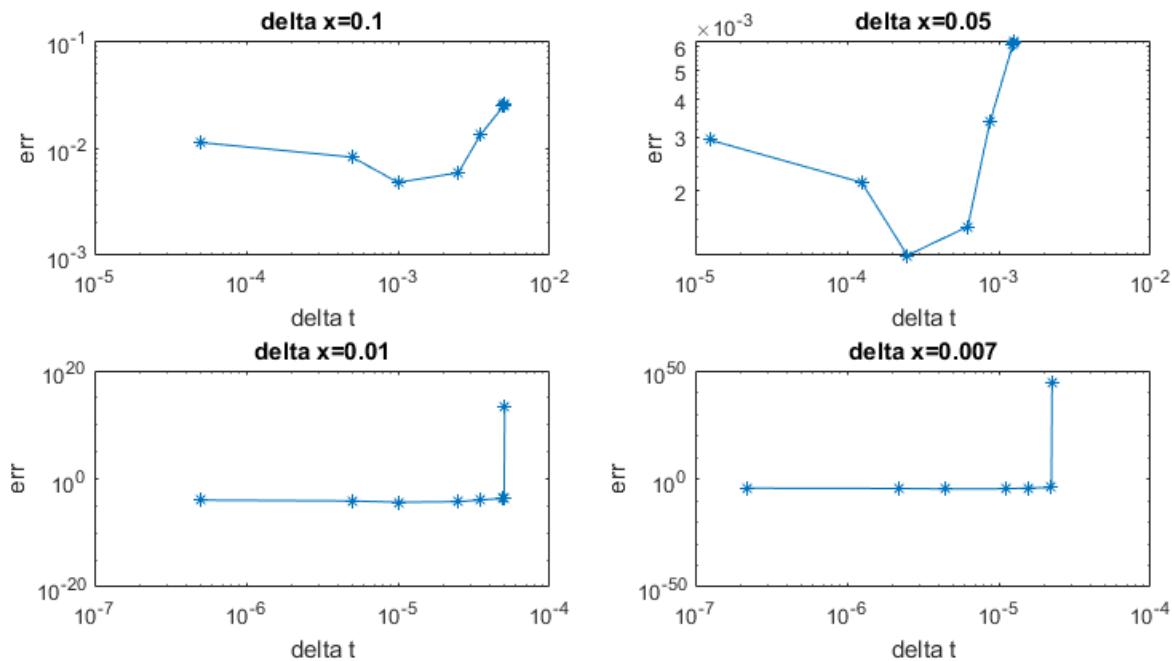
$\Delta x \setminus \Delta t$	$1,01 * \Delta t crit$	$\Delta t crit$	$0,99 * \Delta t crit$	$0,1 * \Delta t crit$	$0,01 * \Delta t crit$
0,1	0,0272	0,0267	0,0263	0,0092	0,0125
0,02	0,0024	0,0010	9,9990e-4	3,7042e-4	5,0854e-4
0,01	5,4200e+14	2,5343e-4	2,4959e-4	9,2563e-05	1,2713e-4
0,007	4,0688e+45	1,1261e-4	1,1091e-4	4,1152e-05	5,6523e-05

Per analizzare meglio l'andamento dell'errore sul grafico sono presenti più valori di Δt rispetto a quelli analizzati nella tabella: sono presenti anche 0,7, 0,5 e 0,2 * Δt critico.

Osservando come varia l'intervallo temporale usato nei vari cicli: nel primo caso, l'intervallo temporale più piccolo è dell'ordine di 10^{-5} , mentre nell'ultimo caso ha ordine di 10^{-7} .

Riguardo agli errori, si osserva che nel primo intervallo spaziale, il più grande, l'errore cresce in maniera lineare intorno al Δt critico, mentre in tutti gli altri casi vi è una profonda differenza tra l'errore prima e dopo il tempo critico. Il dislivello aumenta enormemente al diminuire di Pe, essendo quest'ultimo proporzionale a Δx . Possiamo quindi intuire che il comportamento dell'errore vari al variare del valore di Pe.

Ora analizzo come varia l'errore ponendo $\beta = 0,05$. In questo caso il fenomeno diffusivo prevale sul trasporto.



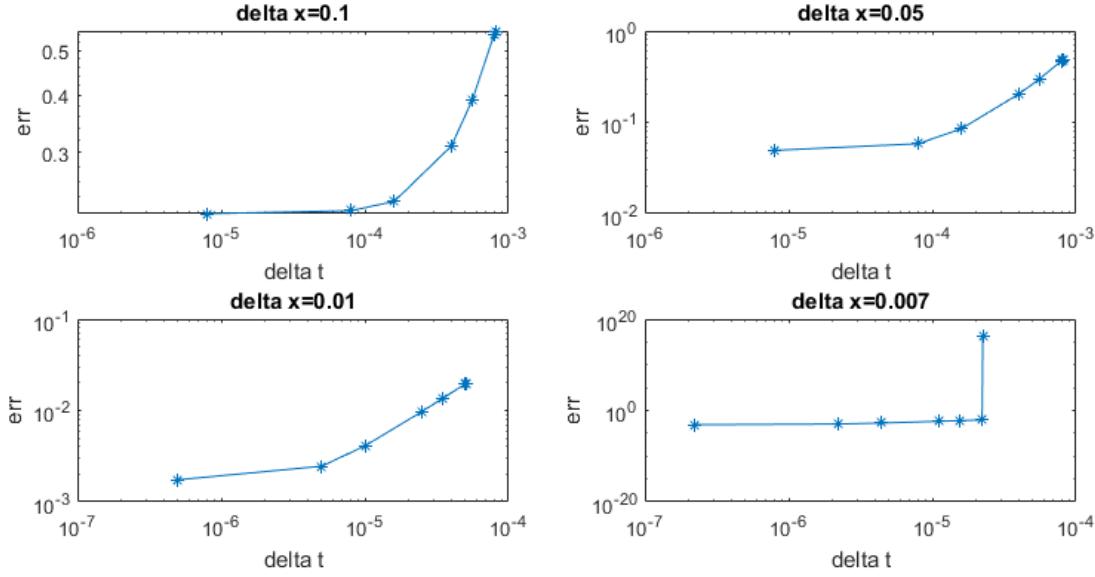
I valori che assume l'errore, sono:

$\Delta x \setminus \Delta t$	$1,01 * \Delta t crit$	$\Delta t crit$	$0,99 * \Delta t crit$	$0,1 * \Delta t crit$	$0,01 * \Delta t crit$
0,1	0,0253	0,0250	0,0245	0,0082	0,0113
0,02	9,8614e-4	9,7154e-4	9,5693e-4	3,3930e-4	4,6996e-4
0,01	2,6970e14	2,4275e-4	2,3911e-4	8,4926e-05	1,1767e-4
0,007	2,0242e+44	1,0784e-4	1,0622e-4	3,7737e-05	5,2289e-05

Anche in questo caso si nota come il valore del Δt critico varia al variare di Δx , si passa infatti da un passo temporale massimo dell'ordine di 10^{-2} , ad un minimo di 10^{-7} .

Riguardo all'errore, si osserva che il comportamento è estremamente simile a quello del caso precedente, l'unica differenza rilevante è che per $\Delta x = 0,02$ l'errore si comporta come per $\Delta x = 0,1$. La causa di ciò può essere la variazione di β , che porta ad una variazione dei valori di Pe.

Infine analizzo come varia l'errore nel caso in cui $\beta = 50$. In questo caso il fenomeno del trasporto è più



intenso di quello diffusivo.

$\Delta x \setminus \Delta t$	$1,01 * \Delta t_{crit}$	Δt_{crit}	$0,99 * \Delta t_{crit}$	$0,1 * \Delta t_{crit}$	$0,01 * \Delta t_{crit}$
0,1	0,5509	0,5426	0,5391	0,2244	0,2211
0,02	0,0835	0,0826	0,0817	0,0099	0,0070
0,01	0,0197	0,0195	0,0194	2,4545e-3	1,74303e-3
0,007	1,2939e+16	8,6032e-3	8,5167e-3	1,0892e-3	7,7421e-4

In questo caso Δt assume valori minori rispetto ai casi precedenti, a causa del fatto che C è proporzionale a β quindi, affinché assuma valori piccoli, è necessario che il passo temporale sia estremamente breve.

Rispetto all'errore, si nota che solo nell'ultimo caso si ha una forte variazione dell'errore intorno a Δt critico. Posso sempre ipotizzare che ciò sia dovuto dalla variazione di β rispetto al caso precedente.

Si nota inoltre come il passo sulle x influenzi la precisione della stima, infatti nonostante negli ultimi due casi il passo temporale più piccolo abbia simile ordine di grandezza, l'errore nel terzo caso è dell'ordine di 10^{-3} , mentre nell'ultimo caso è dell'ordine di 10^{-4} .

A questo punto si possono fare delle osservazioni globali sulla stabilità.

Osservando i due distinti comportamenti che assume l'errore per i Δt maggiori del Δt critico notiamo che in alcuni casi la non stabilità teorica non è evidente, cioè l'errore è simile ai casi in cui la soluzione è stabile. Tuttavia ciò non contraddice i risultati teorici: la motivazione di questo comportamento sta nell'intervallo temporale scelto. Probabilmente prendendo un intervallo temporale maggiore potremmo verificare anche numericamente l'instabilità.

Inoltre, immagino che la scelta d Δx causi l'aumento dell'errore per i Δt più piccoli. Cioè che la convergenza di questo metodo sia data dall'ordine di grandezza di entrambi.

Exercise 2

Exact Solution

We are considering the following problem :

$$\begin{cases} \beta\varphi'(x) - \varphi''(x) = 0 & x \in (0, 1), \beta > 0 \\ \varphi(0) = 1, \quad \varphi(1) = 0 \end{cases} \quad (\star)$$

It's a advection-diffusion equation with $\mu = 1$.

We start by computing the exact solution of the problem (\star) , for which we use the characteristic polynomial for second order linear homogeneous ODE with constant coefficients.

$$-\lambda^2 + \beta\lambda = 0 \Rightarrow \lambda = 0, \lambda = \beta$$

So we have

$$c_1 + c_2 e^{\beta x}$$

Using the BCs we find

$$\begin{cases} c_1 + c_2 = 1 \\ c_1 + c_2 e^\beta = 0 \end{cases} \Rightarrow \begin{cases} c_1 = 1 - c_2 \\ 1 - c_2 + c_2 e^\beta = 0 \end{cases} \Rightarrow \begin{cases} c_1 = 1 - c_2 \\ c_2(e^\beta - 1) = -1 \end{cases} \Rightarrow \begin{cases} c_1 = \frac{e^\beta}{e^\beta - 1} \\ c_2 = -\frac{1}{e^\beta - 1} \end{cases}$$

so the exact solution of (\star) is

$$\varphi(x) = \frac{e^\beta - e^{\beta x}}{e^\beta - 1}$$

Upwind Finite Difference Scheme

Now we discretize the problem, we note that if we were to discretize using central difference we will have to impose some strong restriction on the Peclet number ($Pe = \frac{\beta \Delta x}{2\mu}$) for the stability of the method, so we use another approach using the upwind finite difference scheme losing the one order of truncation accuracy (from $O(\Delta x^2)$ to $O(\Delta x)$) but avoiding the strong restriction :

$$\varphi''(x_i) = \frac{\varphi_{i+1} - 2\varphi_i + \varphi_{i-1}}{\Delta x^2},$$

$$\varphi'(x_i) = \frac{\varphi_i - \varphi_{i-1}}{\Delta x}$$

where $\varphi_j = \varphi(x_j)$ and Δx is the space between the nodes. We can rewrite the upwind scheme as :

$$\frac{\varphi_i - \varphi_{i-1}}{\Delta x} = \frac{\varphi_{i+1} - \varphi_{i-1}}{2\Delta x} - \frac{\Delta x}{2} \frac{\varphi_{i+1} - 2\varphi_i + \varphi_{i-1}}{\Delta x^2}$$

so our discretized problem is

$$\beta \frac{\varphi_{i+1} - \varphi_{i-1}}{2\Delta x} - \left(1 + \frac{\Delta x \beta}{2}\right) \frac{\varphi_{i+1} - 2\varphi_i + \varphi_{i-1}}{\Delta x^2} = 0$$

It is as if we had discretized the problem $\beta \varphi'(x) - \left(1 + \frac{\Delta x \beta}{2}\right) \varphi''(x) = 0$ thus changed the viscosity μ from 1 to $1 + \frac{\Delta x \beta}{2}$, basically introducing artificially more viscosity, calling $Pe = \frac{\Delta x \beta}{2} (\mu = 1)$, and finding Pe^* of the new problem we have that $Pe^* = \frac{Pe}{1 + Pe} < 1$ which is the condition for the stability of the method we may rewrite the discretization as :

$$\beta \frac{\varphi_{i+1} - \varphi_{i-1}}{2\Delta x} - \left(1 + Pe\right) \frac{\varphi_{i+1} - 2\varphi_i + \varphi_{i-1}}{\Delta x^2} = 0$$

MATLAB Implementation

Let's start with some preliminary observations : since this is an advection-diffusion equation with $\mu = 1$ we will have a boundary layer of order $o(1/\beta)$ so if $\beta \gg 1$ we expect a very small boundary layer and if $\beta \ll 1$ we expect $\varphi(x) \approx -x + c$ (it can be verified through a taylor expansion of $e^{\beta x}$) so practically a non negligible boundary layer.

We fix 200 nodes on x ($n_x = 200$) so $\Delta x = 1/n_x = 0.005$, we choose so many nodes because we know that the upwind scheme has a low order of accuracy so we try to reduce error increasing n_x . Now we will confirm what said above with MATLAB, we start with $\beta = 0.01$. In this case the Peclet number is $Pe = 2.5 \cdot 10^{-5}$ and the $Pe^* = 2.49 \cdot 10^{-5}$ which can be calculated by the formula given before. With such a small β , thus the Pe , the problem can be considered as a diffusion dominated problem so as expected the solution is approximable with the line $\varphi(x) = -x + 1$ the c is 1 due the the BCs(See Figure 1). In this case the error is $3.12 \cdot 10^{-8}$. For solving it in MATLAB we used the matrix A, which is tridiagonal matrix, that we can deduce from the discretization above :

$$A = \begin{pmatrix} 2 \frac{\mu_*}{\Delta x^2} & -\frac{\mu_*}{\Delta x} + \frac{\beta}{2\Delta x} & 0 & 0 & \dots & 0 \\ -\frac{\mu_*}{\Delta x} - \frac{\beta}{2\Delta x} & 2 \frac{\mu_*}{\Delta x^2} & -\frac{\mu_*}{\Delta x} + \frac{\beta}{2\Delta x} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & -\frac{\mu_*}{\Delta x} - \frac{\beta}{2\Delta x} & 2 \frac{\mu_*}{\Delta x^2} \end{pmatrix}$$

where $\mu_* = \mu(1 + Pe) = (1 + Pe)$ since $\mu = 1$.

To solve it we use simply the MATLAB incorporated method $\varphi = A \setminus \mathbf{b}$, where \mathbf{b} is the null vector since the is no external force. We put the terms related to φ_1 and φ_{n_x+1} in the known term \mathbf{b} since we have their value from BCs. Finally we add the terms φ_1 and φ_{n_x+1} to the φ we just calculated obtaining the solution.

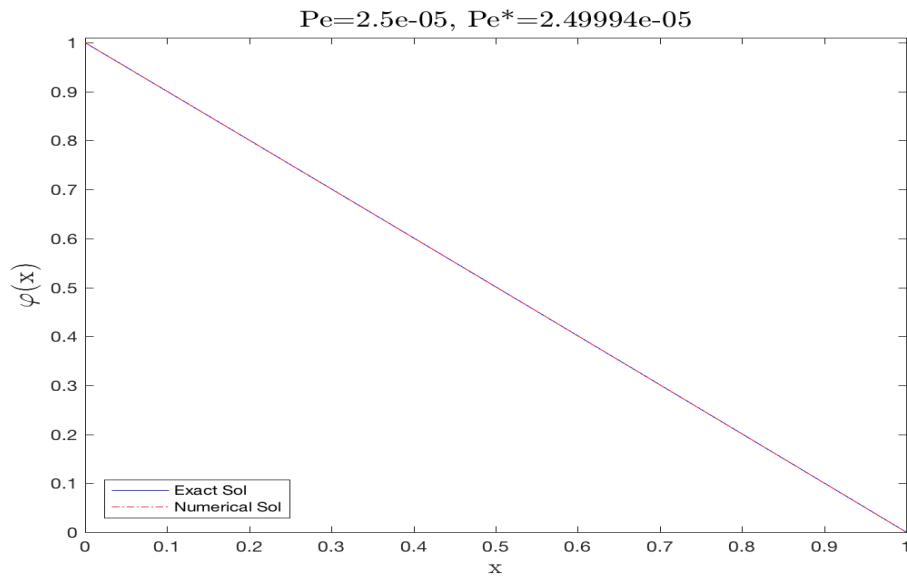


Figure 1: plot with $\beta = 0.01$

Now we choose $\beta = 10$ and observe that the solution start to have a boundary layer of a size between $6/\beta$ and $5/\beta$, in this case the error is $9 \cdot 10^{-3}$

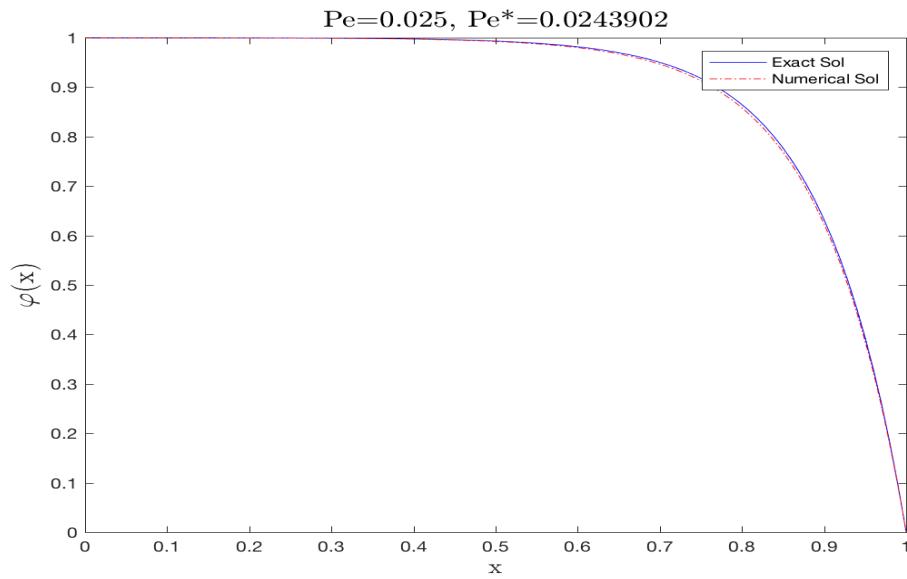


Figure 2: plot with $\beta = 10$

Now we see what happens for an extremely high β , so we choose $\beta = 100$, we see the boundary layer's size is still between $6/\beta$ and $5/\beta$ and we have an error of 0.077.

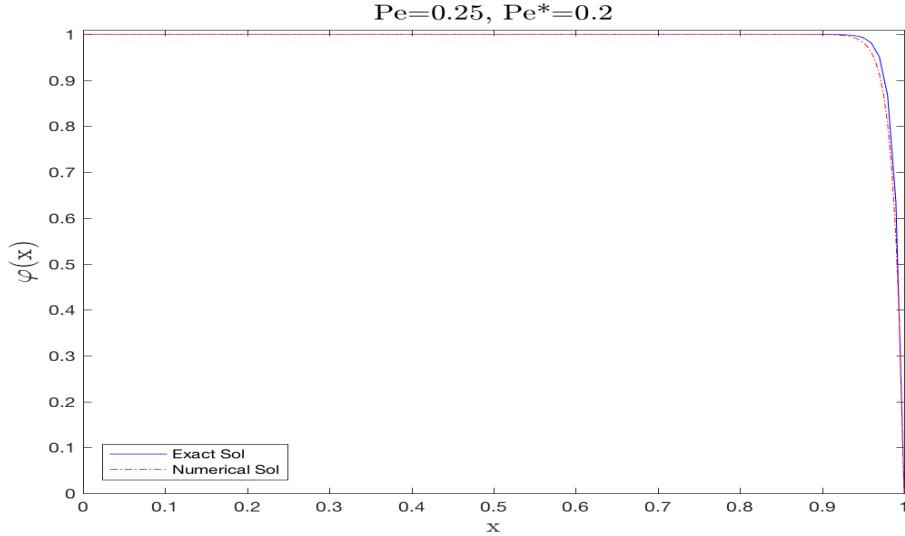


Figure 3: plot with $\beta = 100$

Now let's check the accuracy order, we fix $\beta = 0.1$ and 100 two extreme values of β and see the error for some Δx and do a log-plot. For $\beta = 0.01$ we see that the order of convergence is $O(\Delta x)$ as expected from the Upwind scheme (See Figure 4).

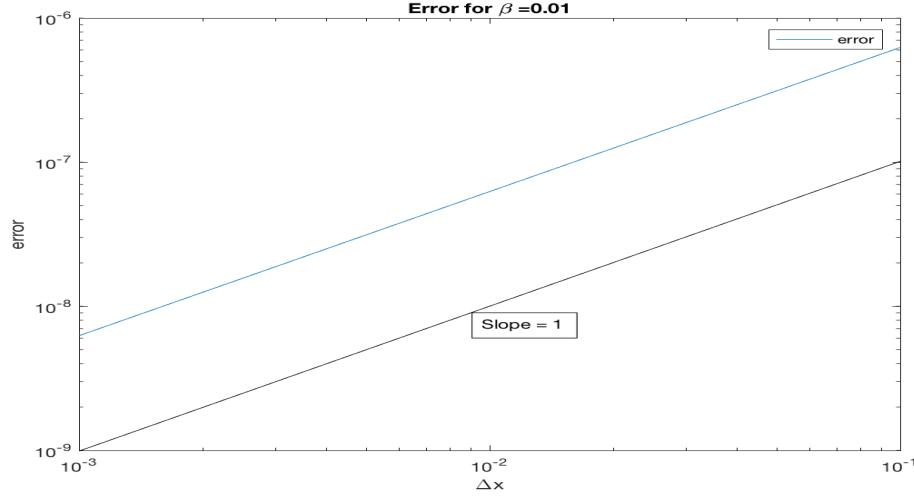


Figure 4: Error log-plot with $\beta = 0.01$

From Figure 5 we see that for $\beta = 100$ we have a strange behaviour for higher Δx , where for higher we mean the bigger value of Δx , for smaller Δx (from $\Delta x > 0.02$) we see the expected convergence order $O(\Delta x)$, what happens for the higher value of Δx is that due to the very small boundary layer (Figure 3), if the Δx is high enough it doesn't "see" the boundary layer so the points near it doesn't create the error that they should be creating so the error calculated is low, and near $\beta = 0.02$ the problem starts seeing the boundary layer but the Δx is not high enough to approximate decently the function near it so the error is higher in that particular β .

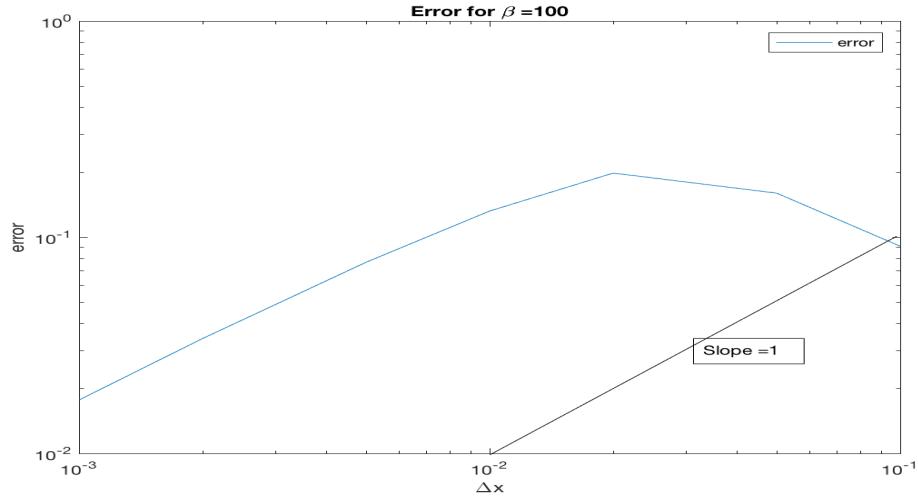


Figure 5: Error log-plot with $\beta = 100$

We write down the errors in a table

Δx	0.1	0.05	0.02	0.01	$5 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
Error($\beta = 0.01$)	$6.2 \cdot 10^{-7}$	$3.1 \cdot 10^{-7}$	$1.2 \cdot 10^{-7}$	$6.2 \cdot 10^{-8}$	$3.1 \cdot 10^{-8}$	$1.3 \cdot 10^{-8}$	$6.3 \cdot 10^{-9}$
Error($\beta = 100$)	$9.1 \cdot 10^{-2}$	$1.6 \cdot 10^{-1}$	$2.0 \cdot 10^{-1}$	$1.3 \cdot 10^{-1}$	$7.7 \cdot 10^{-2}$	$3.4 \cdot 10^{-2}$	$1.8 \cdot 10^{-2}$

We observe that for the same Δx the errors for $\beta = 0.01$ and 100 are the completely of a different order, due to the boundary layer, so now we see what happens for different β fixing $\Delta x = 0.002$, we expect that the error increases as β increases.

We put the errors in a table:

β	0.001	0.005	0.01	0.1	1	5	10
Error($\Delta x = 0.002$)	$1.5 \cdot 10^{-11}$	$3.9 \cdot 10^{-10}$	$1.6 \cdot 10^{-9}$	$1.6 \cdot 10^{-7}$	$1.5 \cdot 10^{-5}$	$2.2 \cdot 10^{-4}$	$4.6 \cdot 10^{-4}$
Error($\Delta x = 0.05$)	$3.1 \cdot 10^{-9}$	$7.8 \cdot 10^{-8}$	$3.1 \cdot 10^{-7}$	$3.1 \cdot 10^{-5}$	$2.9 \cdot 10^{-3}$	$3.9 \cdot 10^{-2}$	$7.6 \cdot 10^{-2}$

β	50	100	200	300	500	700
Error($\Delta x = 0.002$)	$2.3 \cdot 10^{-3}$	$4.6 \cdot 10^{-4}$	$9.0 \cdot 10^{-3}$	$1.3 \cdot 10^{-2}$	$2.2 \cdot 10^{-2}$	$3.0 \cdot 10^{-2}$
Error($\Delta x = 0.05$)	$2.0 \cdot 10^{-1}$	$1.6 \cdot 10^{-1}$	$9.1 \cdot 10^{-2}$	$6.2 \cdot 10^{-2}$	$3.8 \cdot 10^{-2}$	$2.8 \cdot 10^{-2}$

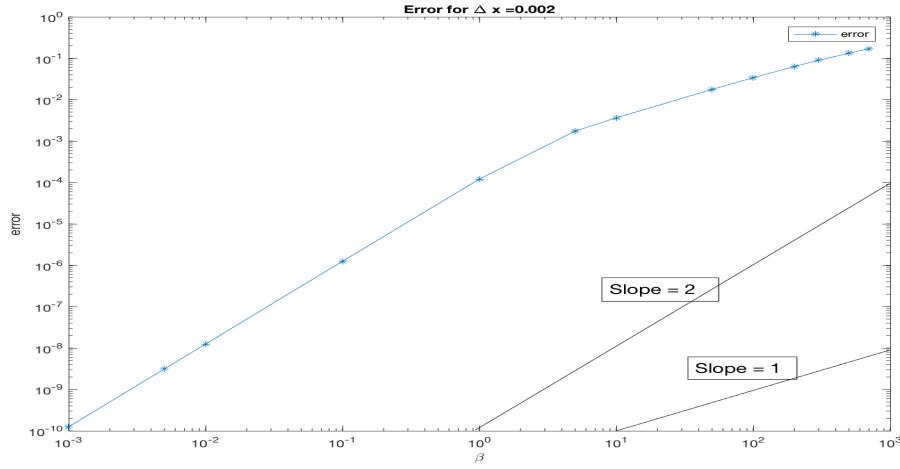


Figure 6: Error log-plot with $\Delta x = 0.002$

In this plot (Figure 6) we can see that until $\beta = 1$ the slope is 2 so the order is $O(\beta^2)$ but for $\beta > 1$ we see that the slope is 1 so the order of $O(\beta)$, so basically as we discussed above when the boundary layer starts to appear we see that the accuracy order in β changes it's order. If we were to take even a smaller Δx we notice the same thing, so we will not be reporting smaller Δx , but it is interesting to see for bigger Δx , for $\beta \leq 1$ we have the same behaviour as above, and for the same reason we mentioned before when discussing Figure 5 we see this fall of error for $\beta \geq 100$, in-fact, the $\Delta x = 0.05$ is too big to see the boundary layer. But we can see that the approximation with $\Delta x = 0.05$ has a really big error since it near 10^{-1} (it can be seen in previous table of errors) so the error is near the 10% of the range of the problem!

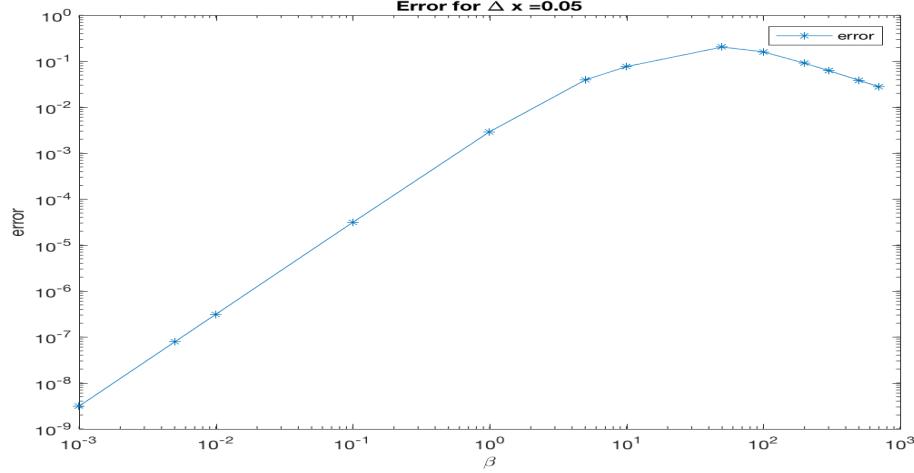


Figure 7: Error log-plot with $\Delta x = 0.05$

Scharfetter-Gummel Scheme

We might generalize the upwind method introducing the artificial viscosity $\mu_h = (1 + \Phi(Pe))$ so the upwind method is a particular case when $\Phi(Pe) = Pe$, the function must satisfy the following conditions :

$$1) \lim_{t \rightarrow 0^+} \Phi(t) = 0 \quad 2) Pe^* = (1 + \Phi(Pe)) < 1 \quad (\text{in our case } \mu = 1)$$

The Scharfetter-Gummel scheme(SG) is with $\Phi(Pe) = Pe - 1 + B(2 \cdot Pe)$ where $B(t) = \frac{t}{e^t - 1}$ so $Pe^* = Pe + B(2 \cdot Pe)$ this is an interesting case to study because of the fact that the artificial viscosity is added in an intelligent way when $Pe \ll 1$ so we can use finite difference scheme, since it's stable, the artificial viscosity of SG scheme is very near to the original viscosity and the method has a truncation error of order $\approx O(\Delta x^2)$ and when $Pe \gg 1$ it adds the necessary viscosity for the stability of the scheme reducing the accuracy to $O(\Delta x)$. This method can be very useful if the β was variable in that case the scheme could decide to have more accuracy if the local Pe is small or to lose a little bit of accuracy in order to have the stability of the method.

MATLAB Implementation

We proceed with the same way of Upwind, so we have our tridiagonal matrix :

$$A = \begin{pmatrix} 2\frac{\mu_*}{\Delta x^2} & -\frac{\mu_*}{\Delta x} + \frac{\beta}{2\Delta x} & 0 & 0 & \dots & 0 \\ -\frac{\mu_*}{\Delta x} - \frac{\beta}{2\Delta x} & 2\frac{\mu_*}{\Delta x^2} & -\frac{\mu_*}{\Delta x} + \frac{\beta}{2\Delta x} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & -\frac{\mu_*}{\Delta x} - \frac{\beta}{2\Delta x} & 2\frac{\mu_*}{\Delta x^2} \end{pmatrix}$$

but with $\mu_* = Pe + B(2Pe)$ with B the Bernoulli function $B(t) = \frac{t}{e^t - 1}$, to calculate the value of the Bernoulli function we will use the 'exp' routine provided by MATLAB, later we will use the bern.m routine to calculate the same. The bern routine should be more accurate for small values of Pe since it uses the Taylor expansion to calculate the value thus reducing the truncation error. Let's start with the 'exp' routine we chose $\Delta x = 0.05$:

We choose $\beta = 0.1, 10$ and 25 , so we see new β we didn't see with the Upwind scheme. For $\beta = 0.1$ in Figure 8 we see that the approximation is really good infact the error is only of $1.7 \cdot 10^{-15}$ which is near the machine precision, we will return on this discussion later on when we discuss the error for this scheme.

For $\beta = 10$ (Figure 9) the error is $4.4 \cdot 10^{-16}$ and for $\beta = 25$ (Figure 10) it's $8.9 \cdot 10^{-16}$, as we can see the errors are really low even for a high Δx which is not expected. Only with 20 nodes we have such a low error.

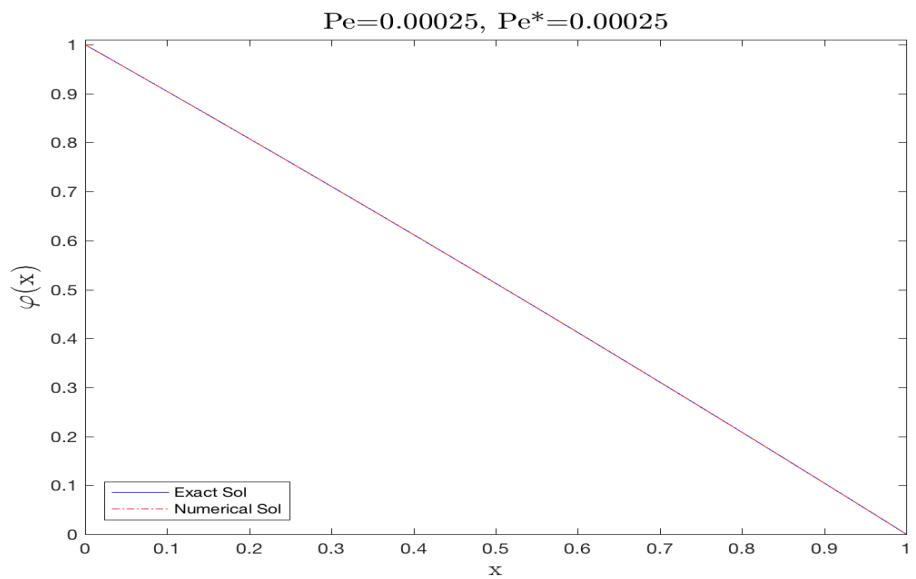


Figure 8: plot with $\beta = 0.1$

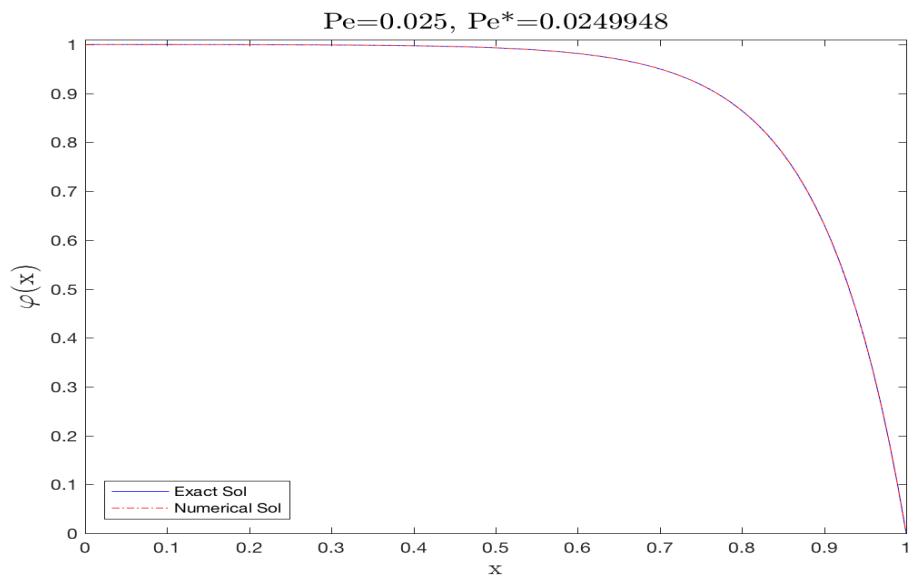


Figure 9: plot with $\beta = 10$

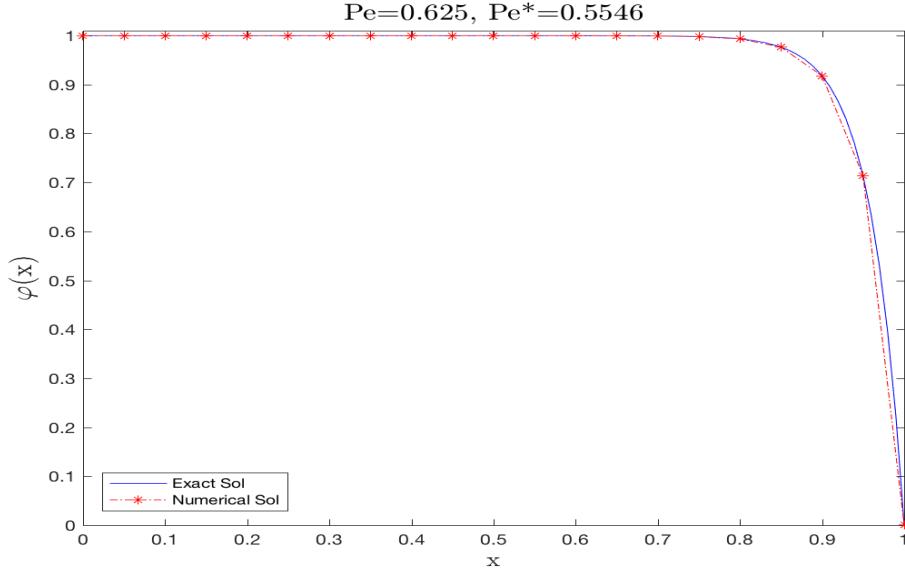


Figure 10: plot with $\beta = 25$

Now let's study the accuracy :
We start by writing down the error table

Δx	0.1	0.05	0.02	0.01	$5 \cdot 10^{-3}$	$2 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
Error($\beta = 0.1$)	$1.22 \cdot 10^{-15}$	$1.7 \cdot 10^{-15}$	$1.8 \cdot 10^{-15}$	$4.7 \cdot 10^{-15}$	$2.5 \cdot 10^{-14}$	$5.4 \cdot 10^{-14}$	$2.0 \cdot 10^{-13}$
Error($\beta = 10$)	$1.0 \cdot 10^{-15}$	$4.4 \cdot 10^{-16}$	$2.7 \cdot 10^{-15}$	$1.7 \cdot 10^{-15}$	$1.5 \cdot 10^{-14}$	$6.7 \cdot 10^{-14}$	$1.2 \cdot 10^{-13}$
Error($\beta = 25$)	$7.8 \cdot 10^{-16}$	$8.9 \cdot 10^{-16}$	$1.1 \cdot 10^{-15}$	$1.3 \cdot 10^{-15}$	$5.9 \cdot 10^{-15}$	$1.7 \cdot 10^{-14}$	$2.1 \cdot 10^{-14}$

As we can see and mentioned before the error is too low (negligible) and as can be seen in Figure 11 does not follow the expected error slope in the log plot, now we explain the possible reason. We take the SG scheme

$$\beta \frac{\varphi_{i+1} - \varphi_{i-1}}{2\Delta x} - \mu_* \frac{\varphi_{i+1} - 2\varphi_i + \varphi_{i-1}}{\Delta x^2} = 0$$

where μ_* is the artificial viscosity mentioned before. We try to find a solution of this equation by putting $C^i = \varphi_i$ and dividing by C^{i-1} finding

$$\left(\frac{\mu_*}{\Delta x^2} - \frac{\beta}{2\Delta x} \right) C^2 - \frac{\mu_*}{\Delta x^2} C + \left(\frac{\mu_*}{\Delta x^2} + \frac{\beta}{2\Delta x} \right) = 0$$

we solve this quadratic equation finding the two roots :

$$C = 1 \text{ and } C = \left(\frac{\mu_*}{\Delta x^2} + \frac{\beta}{2\Delta x} \right) / \left(\frac{\mu_*}{\Delta x^2} - \frac{\beta}{2\Delta x} \right) = \alpha$$

Remembering that $\mu_* = \frac{\beta\Delta x}{2} + \frac{\beta\Delta x}{e^{\beta\Delta x} - 1} = \beta h \left(\frac{1}{2} + \frac{1}{e^{\beta\Delta x} - 1} \right)$ so we rewrite α :

$$\alpha = \left(\frac{\frac{\beta}{\Delta x} \left(\frac{1}{2} + \frac{1}{e^{\beta\Delta x} - 1} \right) + \frac{\beta}{2\Delta x}}{\frac{\beta}{\Delta x} \left(\frac{1}{2} + \frac{1}{e^{\beta\Delta x} - 1} \right) - \frac{\beta}{2\Delta x}} \right) = \frac{1 + \frac{1}{e^{\beta\Delta x} - 1}}{\frac{1}{e^{\beta\Delta x} - 1}} = e^{\beta\Delta x}$$

so we found that $\alpha = e^{\beta\Delta x}$ is one of the roots the other being 1. So the solution is $A + B\alpha^n = A + Be^{\beta\Delta xn}$ call $\Delta xn = x_n$ and impose the BCs for x_0 and $x_{n_x} = 1$ where n_x are the number of nodes, We find the condition $A + B = 1$ and $A + Be^\beta = 0$ which are the same of the ODE (*) we are studying so $A = \frac{e^\beta}{e^\beta - 1}$ and $B = -\frac{1}{e^\beta - 1}$ so the solution is :

$$S(x_n) = \frac{e^\beta - e^{\beta x_n}}{e^\beta - 1}$$

which is the same of our ODE so this scheme for our ODE (*) calculates the exact result thus the error is low and is due to the machine error and not the numerical error of our scheme which can seen easily because:

$$S(x_n) - \varphi(x_n) = 0 \quad \forall x_n \in \text{nodes}$$

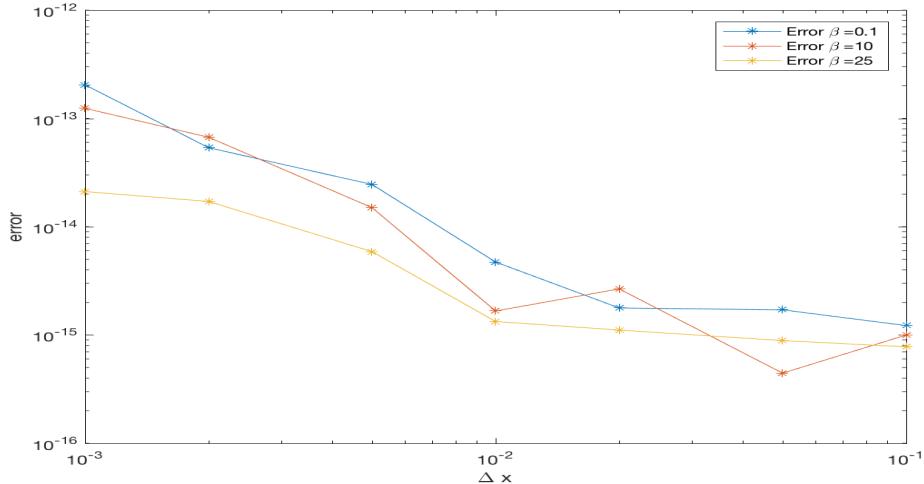


Figure 11: Error log-plot with $\beta = 0.1, 10$ and 25

We see what happens if we use bern.m routine, as we discussed before we expect bern to perform better on lower values of Pe , and from the discussion above we see that the error is only of machine precision so if we were to use bern routine it will help reduce that error, even though it very small. We will not plot the approximated solutions with bern because the difference is very small and cannot be noted through plot, so the plots are basically the same as in Figure 8, 9 and 10. What we will see is fixing a $\Delta x = 0.005$ and choosing different β what is the difference in bern routine and exp routine of MATLAB.

We put in a table the error for different β :

β	0.001	0.005	0.01	0.1	1	5
Error (exp)	$1.45 \cdot 10^{-13}$	$3.82 \cdot 10^{-14}$	$2.16 \cdot 10^{-14}$	$2.45 \cdot 10^{-14}$	$1.51 \cdot 10^{-14}$	$1.0 \cdot 10^{-14}$
Error (bern)	$1.46 \cdot 10^{-3}$	$3.65 \cdot 10^{-14}$	$2.02 \cdot 10^{-14}$	$1.36 \cdot 10^{-14}$	$1.46 \cdot 10^{-14}$	$1.0 \cdot 10^{-14}$

β	10	50	100	200	300	500
Error (exp)	$1.5 \cdot 10^{-14}$	$2.2 \cdot 10^{-14}$	$4.3 \cdot 10^{-14}$	$4.4 \cdot 10^{-16}$	$3.3 \cdot 10^{-2}$	$1.0 \cdot 10^{-15}$
Error (bern)	$1.5 \cdot 10^{-14}$	$2.2 \cdot 10^{-14}$	$4.3 \cdot 10^{-14}$	$4.4 \cdot 10^{-16}$	$3.3 \cdot 10^{-2}$	$1.0 \cdot 10^{-15}$

We observe that after $\beta = 5$ the exp and bern have the same result, in-fact, $2Pe = \beta\Delta x = 2.5 \cdot 10^{-2} > 10^{-2} = x_{lim}$ and the bern routine uses the exp routine for $x > x_{lim}$, we can see that bern is better for smaller number of Pe , in our case the error was already negligible so it doesn't create a great difference using one or another, but if we were in a different situation where the error wasn't negligible anymore the bern routine can provide a lot of help in reducing errors.

Finally we plot the error for both routines :

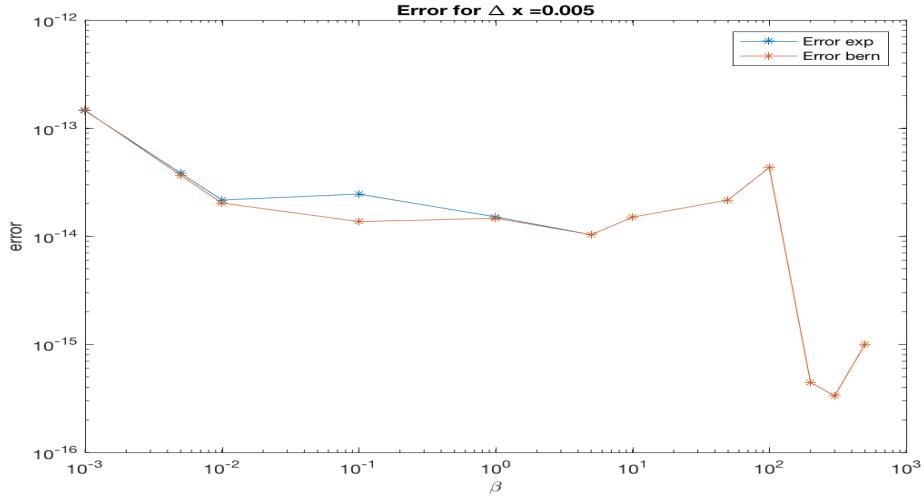


Figure 12: Error log-plot with $\Delta x = 0.005$

MATLAB Codes

Exercise 1 Code

```
1 clear all, close all
2 D=1;
3 %beta=0.05;
4 %beta=1;
5 beta=50;
6
7 %la soluzione esatta e':
8 u=@(x,t) exp(-4*pi^2*D*t).*sin(2*pi*(x-beta*t));
9
10 %determino tempo finale e quanti passi avere nel segmento (0,1)
11 tf=0.15;
12 dex=[10 20 100 150];
13
14 %ora utilizzo la funzione Pe_crit.m per mostrare l'area di convergenza
15 %al
16 %variare di Pe, C
17 % aux=0:0.01:10;
18 % for i= 1: numel(aux)
19 %     Pe_cr(i)=Pe_crit(aux(i));
20 % end
21 % area(aux,Pe_cr),
22 % xlabel('Pe'), ylabel('C'),title('area di stabilita '''),
23 % hold off;
24
25
26 figure
27 for l=1:numel(dex)
28     dx=1/dex(l);
29     x=0:dx:1;
30     dtcrit=dt_critico(dx,beta,D);
31
```

```

32 %determino i dt da analizzare in base al valore critico
33 deltat=[1.01 1 0.99 0.7 0.5 0.2 0.1 0.01]*dtcrit;
34 for k=1: numel(deltat)
35     dt=deltat(k);
36     t=0:dt:tf;
37     Nt=numel(t);
38     Nx=numel(x);
39     v=zeros(Nx,Nt);
40     %metto le condizioni al bordo
41     v(:,1)=u(x,0)';
42     v(1,:)=u(0,t)';
43     v(end,:)=u(1,t)';
44
45     for j=2:Nt
46         for i=2:Nx-1
47             v(i,j)=v(i-1,j-1)*(dt/(dx^2)+beta*dt/(2*dx))+v(i,j-1)
48                 *(1-2*dt/(dx^2))+v(i+1,j-1)*(dt/(dx^2)-beta*dt/(2*dx))
49             );
50         end
51     end
52
53 %questa porzione di codice serve a confrontare graficamente
54 %soluzione numerica e
55 %soluzione esatta
56 %
57 %figure
58 % subplot(1,2,1)
59 % [X,T]=meshgrid(x,t);
60 % mesh(x,t,v'), hold on
61 % xlabel('x'), ylabel('t'); hold off;
62 % subplot(1,2,2)
63 % mesh(X,T,u(X,T))
64 %
65 %figure
66 % mesh(x,t,v'), hold on
67 % mesh(X,T,u(X,T))
68
69
70
71 %calcolo ora l'errore tra sol esatta e approssimazione
72 A=u(X,T)';
73 B=A-v;
74 err(1,k)=max(max(abs(B)));
75
76
77 %costruisco ora il grafico che confronta i vari dt con l'errore
78 str_title=strcat('delta x=',sprintf('%1.0g',dx));
79 subplot(2,ceil(numel(dex)/2),1)
80 loglog(deltat,err(1,:),'*-')

```

```

75     title(str_title)
76     xlabel('delta t'), ylabel('err')
77
78 end

```

$\Delta t_{critico}$ Code

```

1 function dt=dt_critico(dx,beta,D)
2 %questa funzione serve a calcolare il dt massimo per mantenere la
3 %stabilita'
4
5 %Prende in entrata dx e beta, calcola il Pe associato, quindi il C
6 %massimo
7 %e infine ricava dt
8
9 Pe=beta*dx/(2*D);
10 if (Pe<0)
11     dt = 0;
12 end
13 if (Pe<1)
14     dt=Pe*dx/beta;
15 else
16     dt=dx/(Pe*beta);
17 end
18 return

```

$P_e_{critico}$ Code

```

1 function C = Pe_crit(Pe)
2
3 %questa funzione calcola il C massimo per mantenere la stabilita' dato
4 %il Pe in entrata
5
6 if (Pe<=0)
7     C=0;
8 end
9 if (Pe<=1)
10    C=Pe;
11 else
12    C=1./Pe;
13 end

```

Exercise 2 Code

```
1 clear all, close all
2
3
4 x0=0; xL=1;
5 x=linspace(x0,xL);
6
7 mu=1; %coeff di diffusione
8
9 %n=[10 20 50 100 200 500 1000];
10 %bv=[0.1 10 25];
11
12 bv=[0.001 0.005 0.01 0.1 1 5 10 50 100 200 300 500];
13 n=[200];
14 u0=1; uL=0;
15 Pe=zeros(1,numel(bv));
16 Pes=zeros(1,numel(bv));
17 set(0,'DefaultTextInterpreter','latex');
18 for i=1:numel(bv)
19
20     for k=1:numel(n)
21         clear uh A
22         h=abs(xL-x0)/n(k);
23         hv(k)=h;
24         nnodes=n(k)+1;
25         b=bv(i);
26         Pe(i)=h*b/(2*mu);
27         uex=@(x)(exp(b/mu)-exp(b*x/mu))/(exp(b/mu)-1);
28
29         %muh=mu;           %differenze finite centrate
30
31         % TECNICA UPWIND
32         %muh=mu*(1+Pe(i));    %viscosita artificiale
33
34         %TECNICA SG
35         muh=mu*(Pe(i)+bern(Pe(i)*2)); %viscosita artificiale
36         %muh=mu*(Pe(i)+2*Pe(i)/(exp(2*Pe(i))-1));
37
38         Pes(i)=h*b/(2*muh);
39
40         %subplot(2,2,i)
41         %plot(x,uex(x),'b-'), hold on
42         f=zeros(nnodes-2,1); %nodi interni
43         A=zeros(nnodes-2);
44
```

```

45      a1=-muh/h^2+b/(2*h); %colonna j+1
46      b1=2*muh/h^2; %colonna j
47      c1=-muh/h^2-b/(2*h); %colonna j-1
48
49      for j=2:nodes-3
50          A(j,j+1)=a1;
51          A(j,j)=b1;
52          A(j,j-1)=c1;
53      end
54
55      A(1,1)=b1; A(1,2)=a1;
56      A(end,end)=b1; A(end,end-1)=c1;
57
58      %contributo delle BCs sul termine noto
59      f(1)=f(1)-c1*u0;
60      f(end)=f(end)-a1*uL;
61
62      uh=A\f;
63      uh=[u0;uh;uL];
64
65      xh=[x0:h:xL];
66      % plot(xh,uh,'r*-.')
67      % str_title=strcat('Pe=',sprintf('%g',Pe(i)),', Pe*',sprintf('%
68      % g',Pes(i)));
69      % title(str_title,'fontsize',16)
70      % legend('Exact Sol','Numerical Sol','location','SouthWest')
71      % xlabel('x','fontsize',14)
72      % ylabel('$\varphi$(x)','fontsize',16)
73      % hold off
74      err(i,k)=max(abs(uh'-uex(xh)));
75      axis([0 1 0 1.01])
76  end
77 end
78 %print -dpng Ex2SG_b_25
79 set(0,'DefaultTextInterpreter','tex');
80 loglog(bv,err(:,1),'*-' ),hold on
81 % loglog(hv,err(2,:),'*-' )
82 % loglog(hv,err(3,:),'*-' )
83 xlabel('\beta')
84 ylabel('error')
85 % str_title1=strcat('Error \beta = ',sprintf('%g',bv(1)));
86 % str_title2=strcat('Error \beta = ',sprintf('%g',bv(2)));
87 % str_title3=strcat('Error \beta = ',sprintf('%g',bv(3)));
88 % legend(str_title1,str_title2,str_title3)
89 str_titlelog=strcat('Error for \Delta x = ',sprintf('%g',hv(1)));
90 title(str_titlelog)

```

```
90 %print -dpng Error_SG_n_x_50
```

bern.m

```
1 %
2 % [ bp , bm ] = bern (x)
3 %
4 % This function computes the inverse of the Bernoulli function
5 %
6 % bp = B(x) = x / (exp(x)-1)
7 % bm = B(-x) = x + B(x)
8 %
9 function [ bp , bn ] = bern (x)
10
11 xlim = 1e-2;
12 ax = abs (x);
13
14
15 % Calcola la funz. di Bernoulli per x=0
16
17 if (ax == 0)
18     bp=1.;
19     bn=1.;
20     return
21 end;
22
23
24 % Calcola la funz. di Bernoulli per valori
25 % asintotici dell'argomento
26
27 if (ax > 80) ,
28     if (x >0),
29         bp=0.;
30         bn=x;
31         return
32     else
33         bp=-x;
34         bn=0.;
35         return
36     end;
37 end;
38
39
40 % Calcola la funz. di Bernoulli per valori
41 % intermedi dell'argomento
42
```

```

43  if (ax > xlim) ,
44    bp=x./(exp(x)-1);
45    bn=x+bp;
46    return
47  else
48
49
50  % Calcola la funz. di Bernoulli per valori
51  % piccoli dell 'argomento mediante sviluppo
52  % di Taylor troncato dell 'esponenziale
53
54  ii=1;
55  fp=1.;
56  fn=1.;
57  df=1.;
58  segno=1.;
59  while (abs(df) > eps) ,
60    ii=ii+1;
61    segno=-segno;
62    df=df*x/ii;
63    fp=fp+df;
64    fn=fn+segno*df;
65    bp=1./fp;
66    bn=1./fn;
67  end;
68  return
69 end

```