NUMERO: 1813A - ANNO: 2015

A P P U N T I

STUDENTE: Santoro Stefano

MATERIA: Introduction to computational methods for energy applications - prof. Savoldi, Zanino

INTRODUCTION TO COMPUTATIONAL METHODS FOR ENERGY APPLICATIONS

prof.ssa Saboedi Laura 30/9/2014

→ Verificare nome del corso e codice (corso partito quest'anno)

• **Lectures** (Saboedi/Zanino)
Tue 11:30 - 13
Wed 10 - 13

• **Lab sessions** (Saboedi/Bonifetto/Carli)
Tue 14:30 - 17:30 team 1
Thu 14:30 - 17:30 team 2
Fri 14:30 - 17:30 team 3

INTRODUCTION TO THE USE OF MATHLAB

Lez. 1

- Matlab provides functions that operate on:
  - integer, real and complex numbers
  - vectors and matrices
  - structures

MATLAB DOWNLOAD (fino al 1/5)
https://www.areait.polito.it/servizi/default.asp?id_progetto_servizio=195
and follow instructions

- Is an interactive environment, variables are created when they are used (not to declare them). Names can be reused for different types.

→ Matrix dimensions are set dinamically, operations on matrices are applied to all elements of a matrix at once.

⟹ In MATHLAB documents have to be saved in format .m. It's better to write down the scripts.

PRENDI NOTE A4

Lez. 2 (Bonifetto) 1/10/'14

Changing Matrix Elements

- The referenced element can also be changed

>> results (3,4) = 10 or >> results (3,4) = results (3,4) * 100

Accessing Matrix Rows → you can access multiple values from a matrix using ":"

~ To access all columns of a row

>> Variable_name (:, row number)

~ To access all rows of a column

>> Variable_name (:, column number)

~ To change all values in a row or column to zero use

>> results (:,3) = 0 >> results (:,5) = results (:,3) + results (:,4)

→ To overwrite a row or column:

>> results (3,:) = [10, 1, 1, 1] → nella riga 3 piazza i valori

>> results (:,3) = [1, 1, 1, 1, 1, 1, 1] → nella colonna 3 piazza i valori

~> Ovviamente il vettore di valori da inserire deve avere le dimensioni della matrice in esame.

(3, 1:3) ~> in the third row overwrite values from the first to the third column.

(5:2:end, 3) → a partire dalla riga 5, ogni 2, fino alla fine, fai l'operazione nella colonna 3.

Start → dall'inizio fino a x

- Non consecutive rows or columns: vector of indexes using [].

>> Variable_name([index 1, index 2, ...], :)

PRENDI NOTE A4

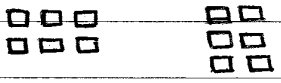
MATHEMATICAL OPERATORS

- Add: +
- Subtract: -
- Divide: ./
- Multiply: .*
- Power: .^

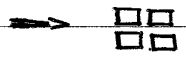
To specify order use only round brackets
→ If you don't put "." is a scalar product and doesn't function with vectors.

~> per prodotto vettoriale DEVO usare ".*"!

A(2,3) .* B(3,2) ovviamente valgono le regole del prodotto matriciale (colonne della prima = righe seconda)



$$\begin{pmatrix}1&2&3\\4&5&6\end{pmatrix}\cdot\begin{pmatrix}1&2\\3&4\\5&6\end{pmatrix}=\begin{pmatrix}22&19\\49&64\end{pmatrix}$$



- Operations on a matrix
  - Matrix Multiply: *
  - Matrix Division: /

- det(A) calculates the determinant of a matrix A
- inv(A) calculates the inverse of a matrix A

>> help inv	→ ti spiega come usare inv

LOGICAL OPERATORS

- Greater than >
- Less than <
- Greater than or equal >=
- less than or equal <=
- Equal ==
- Not equal ~=

aa =

1 2 3
4 5 6

Logical indexing → dà 1 se è vero e 0 se è falso iii = aa>4

0 0 0
0 1 1

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 7 di 86

Create a function

```
function = my_funct(aa,bb)
% This function computes the products of two inputs
% Inputs: aa, bb
prod = aa*bb
end
```

→ FARE ATTENZIONE ALL'ORDINE (aa, bb) PERCHÉ ROTTE VOLTE CAMBIA TUTTO!

→ scrivendo "help my_funct" uscirà tutto ciò che è stato scritto dopo il simbolo "%". Verificare che il nome sia già occupato.

```
>> ff = 10
>> gg = 7
```

my_funct(ff, gg) ans = 70

Plotting = create graphs

- Can be used in different ways, for one data series or for a matrix

→ other functions:
- * hold on
- * hold off
- * title
- * legend
- * axis
- * xlabel
- * ylabel

↳ helpful to create plots

Some useful commands

- pause: pauses the execution of the script
- pause(<time in seconds>)
- "..." + newline: the line continues to the following one

Loops

```
for (<condition>)
    <actions>
end
```

```
while (<condition>)
    <actions>
end
```

→ DO NOT USE SINGLE CHARACTER VARIABLES.

PRENDI NOTE A4

A - A Subtraction

A * A Scalar product (= A^2)

A .* A product element for element

• Non puoi assegnare una variabile che inizia con un numero (Sì A3, no 3A) –
→ Salvando .dat devo aggiungere "-ascii"

cd Lab1 → Salvia nella cartella Lab1

A .* B = il primo di A per il primo di B

A .^ B = il primo di A con esponente il primo di B

C = [A' B' A' B'] → crea una matrice avente come colonne A - B - A - B
→ To select an element: C (raw, column)

D = [A; B; A; B] → crea una matrice avente come righe A - B - A - B

q = W(w,2) → stai selezionando nella matrice W gli elementi nella seconda colonna, selezionando le righe della matrice 1, 3, 5 e 7 (coordinate [1,2], [3,2], [5,2], [7,2])

w = [1 3 5 7]

eye (dimension) → matrice quadrata con 1 sulla diagonale e 0 intorno
ones (dimension) → matrice quadrata con tutti 1
zeros (raw, column)

diag (ones (3,1), 0)

1 0 0
0 1 0
0 0 1

diag (ones (3,1) * 3, 0)

3 0 0
0 3 0
0 0 3

diag (ones (3,1) * 3, 1)

0 3 0 0
0 0 3 0
0 0 0 3

diag (ones (3,1) * 3, -1)

0 0 0 0
3 0 0 0
0 3 0 0
0 0 3 0

PRENDI NOTE A4

- Scaricare GHOSTVIEW (programma per vedere immagini postscritte)

figure (2);
set (gca, 'fontsize', words);
set (0, 'defaultlinelinewidth', line);
orient landscape ~> ruota il foglio dell' immagine

H1 = plot (experimental (i,1), experimental (i,2) * 1e5, 'b');
dd = 100;
H1d = plot (experimental (i:dd:end,1), experimental (i:dd:end,2) * 1e5, 'bs',
'markersize', 15) ~ valore lunghezza linea ← blue
↳ lunghezza linea
set (H1d, 'markerfacecolor', 'y');
↳ colora i quadratini del punto di valore del grafico di giallo.

['r--' significa rosso e tratteggiato]

mnm = get (H1d, 'markersize'); ~> salva in mnm la spessore del tratto
H3 = plot ((-10 -9), (-10 -10), 'bs-', 'markersize', mnm);
↳ coordinate punti da stampare

xlim ((0 1000));
ylim ((1.1e5 1.1e7));
↳ limiti per gli assi → il grafico apparirà solo nello spazio compreso tra questi valori

xlabel ('Time (s)');
ylabel ('Pressure (Pa)');
title ('Coil inlet manifold');

set (gca, 'xtick', (0:100:1000));
set (gca, 'xticklabel', ('0', ' ', '200', ' ', '400', ' ', '600', ' ', '800', ' ', '1000'));
legend ('pippo', 'pluto', 'topolino');

(1D) transient

Heat equation — conduction

$$\frac{\partial u(x,t)}{\partial t} - \mu \frac{\partial^2 u(x,t)}{\partial x^2} = f(x,t),\ x \in (a,b),\ t>0 \quad (9.4)$$

$$\frac{\partial u(x,t)}{\partial t} - \mu \Delta u(x,t) = f(x,t),\ x \in \Omega,\ t>0 \quad (9.5)$$

(2D) h.c.p.

NB $f(x,t)$ is a given function.

→ In steady state problems time is not relevant ($\frac{\partial}{\partial t}=0$, costant).

In (9.4) we have to do an assumption: μ must be a costant. μ is the diffusivity, and is $\mu = K/\rho c_p$, so we assume that temperature influence on conductivity can be ignored.

$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ → To solve equations we need a method.

$$\rho c_p \frac{\partial T}{\partial t} = -\frac{\partial}{\partial x}\left(K \frac{\partial T}{\partial x}\right)$$ one dimensional heat conduction — non-linear equation

- If we impose the costant value of K we can put it out, and obtain in this way a linear equation, deleting temperature dependence.

↝ We have to make a lot of semplifications in order to solve our problems.

* Linearisation = make a non-linear equation into a linear one.

(piccole oscillazioni del pendolo - $\sin\theta \approx \theta$)

so the problem become mathematically tractable.

↝ Parabolic problems correspond to iperbolic equations, of various order in base of the case.

Wave equation $$\frac{\partial^2 u(x,t)}{\partial t^2} - c \frac{\partial^2 u(x,t)}{\partial x^2} = 0$$

PRENDI NOTE A4

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 15 di 86

$$u''(x)\Big|_{x_i} + f(x)\Big|_{x_i} = 0 \quad \text{[generic equation]}$$

→ To solve it we have a set of equations:

$$\begin{cases} u(a) = \alpha \\ u''(x)\Big|_{x_i} + f(x)\Big|_{x_i} = 0 \qquad i = 0, 1, \dots, (n-1) \\ u(b) = \beta \end{cases}$$

- Transform the derivity into a finite differency.

Derivative:

$$\lim_{h \to 0} \frac{1}{h}\left(f(x_0+h) - f(x_0)\right) = f'(x_0)$$

↝ We have to approximate this formulate, deleting the limit.
In this way, we have a new shape of derivative:

$$(\delta_+ f)(x_0) = \frac{f(h+x_0) - f(x_0)}{h}$$

prendo valore a DX di x_0

→ Algebric formule (not ODE now)

this is not a tangent of the function, it's only a link but not the same

FORWARD DIFFERENCE

I have to use Taylor expansion to be sure of the accuracy of our calculation.

$$f(x_0+h) = f(x_0) + h f'(x_0) + \frac{h^2}{2} f''(x_0)$$

$$\delta_+ f = \frac{\cancel{f(x_0)} + h f'(x_0) + \frac{h^2}{2} f''(\xi) - \cancel{f(x_0)}}{h} = f'(x) + \underbrace{\frac{h}{2} f''(\xi)}_{\text{error}} \qquad x_0 < \xi < x_0 + h$$

non importa dove sia

- The error depends of h, decreasing h (more precision) error will decrease, as we want and expect.

$$(\delta_- f)(x_0) = \frac{f(x_0) - f(x_0 - h)}{h}$$

prendo valore a SX di x_0

BACKWARD DIFFERENCE

L'ERRORE VA A ZERO INSIEME A h, MOLTO IMPORTANTE QUESTA PROPORZIONALITÀ!

PRENDI NOTE A4

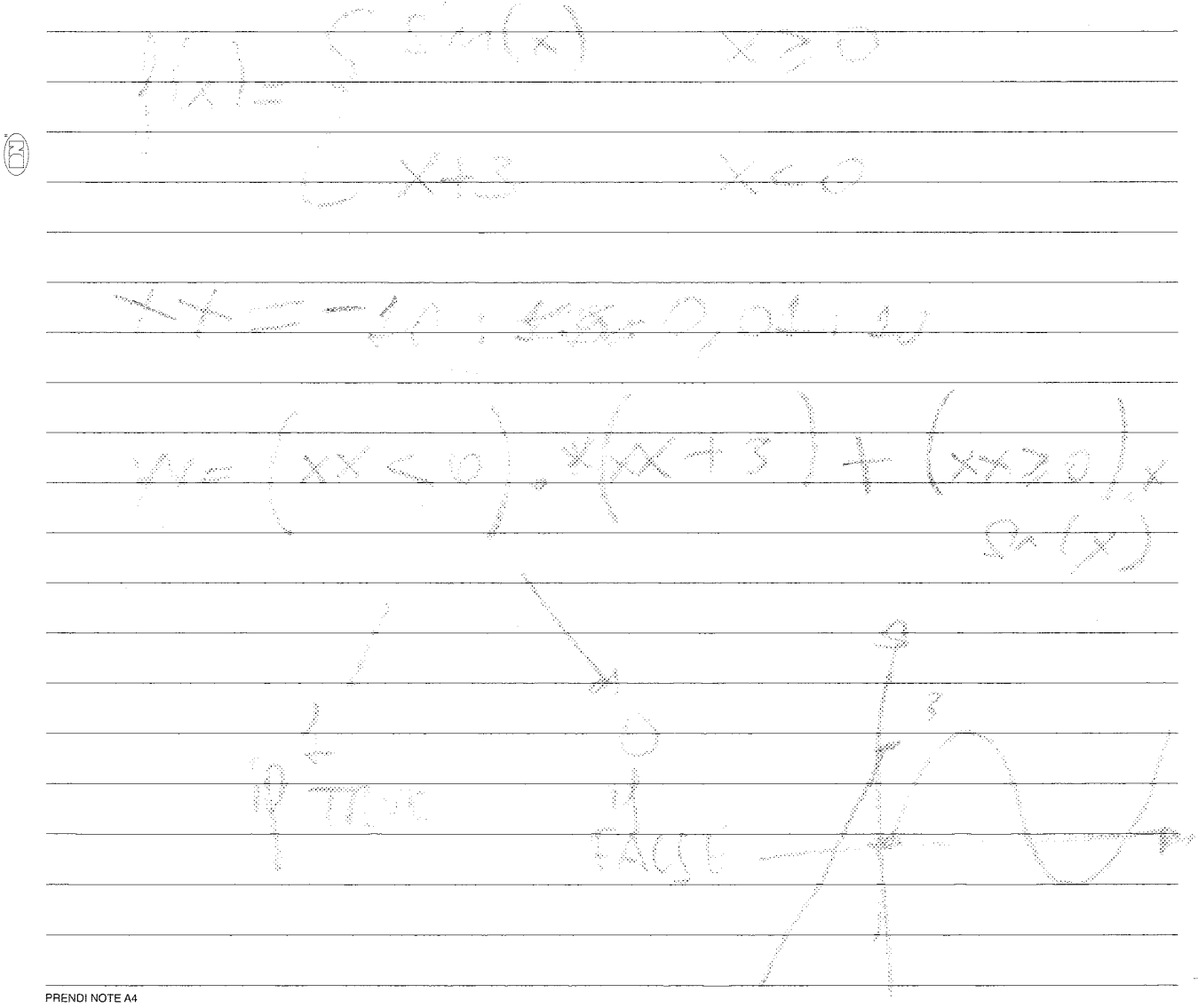
© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 17 di 86

```
>> any([9 7 6 0]>5,2)
   ans = 1;
```

```
>> issorted([1 2 3 4])   (è ordinato)
   ans = 1;
>> issorted([1 2 6 4])
   ans = 0;
```

```
>> sort([1 4 6 3])
   ans =
      [1 3 4 6];
>> sort([1 4 6 3], 'descend')
   ans =
      [6 4 3 1];
```

```
>> trie(A)   prende matrice triangolare superiore nella matrice A
>> tril(A)   inf
>> norm([3 4],2)   indice radice
   ans = 5   (lunghezza vettore)
   (fa il teorema di Pitagora con le dimensioni del vettore)
```



```
clear all
close all
clc
x = [-50:5:50];
ezz = zeros(lenght(x),1);

for i=1:lenght(x)
    exp_app = ES4fun(x(i));
    ezz(i) = abs(exp(x(i)) - exp_app)/exp(x(i));
end

figure(1)
hold on;
grid on;
box on;
semilogy(x,ezz); --> in zero non avremmo perciò nessun valore!
ylabel('Relative error');
xlabel('x');

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

function out = ES4fun(in)
        tol = 1e-14;
        out = 1;
        old = 1;
        error = 1;
        K=1;    out_old = 1;
while error > tol
    old = old * in/K;
    out = out + old;
    K = K+1;
    error = abs(out - (out_old))/abs(out_old);
    out_old = out;
end
```

PRENDI NOTE A4

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 21 di 86

In the polynomial approximation polynomial coefficients (unknows) enter linearly, while in the zational interpolation enter not linearly.

LAGRANGIAN POLYNOMIAL INTERPOLATION

For any set of couples (x_i, y_i), $i = 0, \dots, n$ there exists a unique polynomial of degree less than n, Π_n. (Vedi libro)

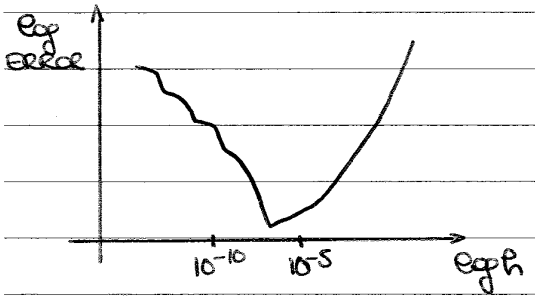
$$\boxed{\Pi_n = \sum_{k=0}^{n} y_k \varphi_k(x)} \quad \text{where} \quad \varphi_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j} \qquad k = 0, \dots, n$$

$$f(x) = e^{sx} \qquad f'(x) = se^{sx}$$

- If we evaluate the error in $x = 0$ between numerical and exact solution we obtain an error, but we have to NORMALIZE the result:

$$\boxed{\text{ERROR} = \frac{|sf(1) - f'(1)|}{|f'(1)|}} \sim \text{definition of error}$$

You have to evaluate error for different h, and plot the results:



* In scala logaritmica qualsiasi polinomio diventa una retta poiché $\log x^{\alpha} = \alpha \log x$, cambia solo la pendenza.

!! If h becomes small and $f(x+h)$ and $f(x-h)$ become closer, the pc can't evaluate the error in relative terms.

[CANCELATION PHENOMENON]

PRENDI NOTE A4

If $h < \varepsilon_M$ the error will increase (previous post) because the machine can't evaluate that situation.

SMALLEST POSSIBLE NUMBER $x_{min} = \beta^{t-1}$

LARGEST POSSIBLE NUMBER $x_{max} = \beta$

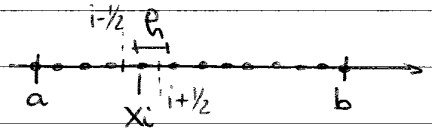
If you go below the lowest you are in underflow, you obtain or zero or an arbitrary value without warnings. Really dangerous, causes errors.
If you go below the largest you are in overflow. Not really dangerous, the execution stops.

1.2.2 Some mathematic operations are invalid cause of the floating point numbers set of precision and max and min.

↳ ASSOCIATIVE and DISTRIBUTIVE properties are NOT valid in floating logic!

Lez. 6 (Zanino) [monte Vinco] 22/10/'14

$$\begin{cases} u''(x) + f(x) = 0 & a < x < b \\ u(a) = \alpha \quad u(b) = \beta \end{cases}$$



discretization in a certain number of nodes between a and b

$$u'\Big|_{x_i} \cong \frac{u_{i+1} - u_{i-1}}{2h}$$ approximation of first order derivative

$$u''\Big|_{x_i} \cong \frac{u'|_{i+½} - u'|_{i-½}}{h} \cong \frac{\frac{(u_{i+1} - u_i)}{h} - \frac{(u_i - u_{i-1})}{h}}{h} \cong \boxed{\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}}$$

- For a given accuracy, I try to minimize the number of nodes.

‖ APPROXIMATION OF SECOND ORDER DERIVATIVE IN FINITE DIFFERENCES

PRENDI NOTE A4

- The second question is: where are the non-zero elements?
- How I should solve a problem like this, a large tridiagonal linear system of equations?

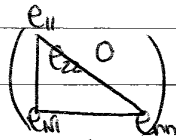
5 To obtain the solution $\boxed{x_i = \frac{\det(A_i)}{\det(A)}}$ $\quad FLOP(n) = n!$

where A_j is the matrix obtained by replacing the i-th column with the vector b. By the Cramer rule each system needs a different time, and it is called "computational cost"

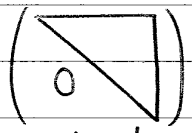
$$\text{CPU time} = \frac{\#\text{ of FLOPS needed}}{(\text{FLOPS})}$$

(FLOPS: flops of your pc)

- In order to obtain solutions by sostitution we have to reconduce to a triangular matrix, that allows us to use this method:



$Lx = b$ FORWARD SOSTITUTION STRATEGY



$Ux = b$ BACKWARD SOSTITUTION STRATEGY

$$Lx = b \qquad x_i = b_i / e_{11} \qquad (x_i) = \frac{\left[b_i - \sum_{k=1}^{i-1} e_{ik} x_k\right]}{e_{ii}}$$

(when I arrive to x_i I mean that others have just been solved)

$$\boxed{x_i = \frac{\left[b_i - \sum_{k=1}^{i-1} e_{ik} x_k\right]}{e_{ii}}} \qquad i = 2, \dots, n \qquad FLOP(n) = n^2 \rightarrow \text{convenient!}$$

- For really simple matrix this method has a computational cost ridiculous confronted by Cramer method.

(matrix with a simple structure (lower triangular, upper triangular and so on))

PRENDI NOTE A4

- You have to eliminate all the elements situed in a position different from the triangular one, that it's the shape to obtain.
  → Each step comport the elimination of the elements present in one column, under the main diagonal.

- Once obtained the upper or lower triangular matrix we can solve it by the backward or forward algorithm.

for $K = 1, \dots, n-1$
  for $i = K+1, \dots, n$
    $e_{ik} = \frac{a_{ik}^{(K)}}{a_{kk}^{(K)}}$ — one for every element of the matrix
    for $j = K+1, \dots, n$
      $a_{ij}^{(K+1)} = a_{ij}^{(K)} - e_{ik} a_{kj}^{(K)}$
      $b_i^{(K+1)} = b_i^{(K)} - e_{ik} b_k^{(K)}$

(K) = obtained in the K-Step of the elimination of Gauss.

$a_{ik}^{(K+1)} = a_{ik}^{(K)} - \frac{a_{ik}^{(K)}}{a_{kk}^{(K)}} \cdot a_{kk}^{(K)} = 0$ IN QUESTO MODO ELIMINO TUTTI GLI ELEMENTI DELLA COLONNA K IN ORDINE

Lez. 7 (Zanino) 28/10/'14

- The matrix upper triangular is called "U factor". The matrix L is made by the Gauss method. The pure factorization leads us to:

$$Ax = b \rightarrow (LU)x = b$$

L obtained by multipleyers $\{e_{ij}\}$

We arrive to $Ux = b_{new}$. It is the classical method used to solve this kind of problems.

eg $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} x = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$ Dobbiamo ottenere una matrice triangolare

for $K = 1, \dots, 1$ ~ dobbiam togliere solo un elemento
  for $i = 1, \dots, 1$
    $e_{ik} = e_{21} = \frac{a_{21}}{a_{11}} = \frac{3}{1} = 3$

PRENDI NOTE A4

- Considering Vandermonde matrix:

$$\boxed{A = a_{ij}} \text{ with } a_{ij} = x_i^{n-1} \quad i,j = 1,\dots,n$$

It's a full matrix, not a sparse matrix. Time required to solve a full linear system of dimension n depends of n, passing from 10^{-4} sec to out of order. The number of operations required are in a cubic scale.

↝ It may not work if matrix A is singular or if any of the principal submatrices is singular. The method of Gauss, in fact, would divide by zero, and it's not acceptable.

- Method of elimination of Gauss with PIVOTING is the right way to proceed. You have to find the largest element of the column and you have to change that row switching with the row of the diagonal. In this way, you won't have singular matrices during the method of Gauss.

eg $\begin{pmatrix} 1 & 1 & 3 \\ 2 & 0 & -4 \\ 3 & 3 & -5 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 1 & 3 \\ 3 & 3 & -5 \\ 2 & 0 & -4 \end{pmatrix}$

the PIVOT is equal to zero, you have to search in that column the largest element and switch it with zero.

DON'T NEED PIVOTING! ↓

- There are some special classes of matrices. In particular:

1) strictly diagonally dominant matrices → they have the property of not being singular, they don't need pivoting. It's strictly dominant by row if:

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \quad i = 1,\dots,n$$

by column if: $|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ji}| \quad i = 1,\dots,n$

- Calling P the permutation matrix we can obtain:

$$PA = LU$$

$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

- At the beginning the P matrix is a MATRICE IDENTITÀ (tutti 1), and every step in which you switch a row you have to switch the corrispondent rows in the P matrix (remain all "ones" on the diagonal).

for $K = 1, \dots, n-1$

find $\bar{r}$ such that $|a_{\bar{r}K}^{(K)}| = \max_{r=K,\dots,n} |a_{rK}^{(K)}|$

exchange row K with row $\bar{r}$ in both A and P

for $i = K+1, \dots, n$

$$e_{iK} = \frac{a_{iK}^{(K)}}{a_{KK}^{(K)}}$$

for $j = K+1, \dots, n$

$$a_{ij}^{(K+1)} = a_{ij}^{(K)} - e_{iK} a_{Kj}^{(K)}$$

- To have very large multipleyers is a problem, because they will increase error, its propagation will be larger than other cases.

→ In some cases the factorization generates two matrices inaccurate, and you can realize it if you make the residual matrix $A - LU$, that should be the null matrix but sometimes it isn't.

HOW ACCURATE IS THE SOLUTION?

- Pivoting allows us to keep errors under control, but in some cases this is not true. Some examples:

Hilbert matrix $a_{ij} = 1/(i+j-1) \quad i,j = 1,\dots,n$

$A_n x_n = b_n$ (chosen to have $x_n = (1,\dots,1)^T$)

→ relative error $E_n = \|x_n - \hat{x}_n\| / \|x_n\|$

If $n > 10$ we will have $E_n > 13$, that is larger than 1000%!

Residual matrix $R_n = L_n U_n - P_n A_n$ after factorization

PRENDI NOTE A4

• We obtain: $\frac{\|\delta x\|}{\|x\|} \leq K(A) \frac{\|\delta b\|}{\|b\|} \rightarrow \leq \varepsilon_m$ ($K(A) \sim 10^p$)

$K_1(A) = \|A\|_1 \cdot \|A^{-1}\|_1$, being $\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$ (for a sparse matrix)

• $K(A_n)$ is a rapidly increasing function of n. One has $K(A) > 15000$, while if $n > 13$ MATLAB warns that matrix is close to singular.

→ Warning tell you not that the result is wrong, but that the result could be wrong. If $K(A_n)$ is small we can be sure that error is small provided that the residual is small, this might not be true when $K(A_n)$ is large.

⇝ Well-posed problem { ∃ solution; unicity of solution; solution dependence on data with continuity ? in ogni punto

• Generally, we can't transform an ill-conditioned problem in a well-posed problem.

TRIDIAGONAL SYSTEM

$$\begin{pmatrix} 1 & 0 & 0 & 0 & & \\ 1 & -2 & 1 & 0 & & 0 \\ 0 & 1 & -2 & 1 & & \\ & & \ddots & & & \\ 0 & & & & \ddots & \\ & & & & 1 & -2 & 1 \\ & & & & 0 & 0 & 1 \end{pmatrix}$$

obtained by discretization

→ We want to use method of elimination of Gauss. We want to obtain a LU factorization.

• What is the computational cost of REG and of LU factorization? Depends of n, because FLOP(n) ~ n. Se use un numero di nodi maggiore di 3 ottengo un polinomio con alta accuratezza, ma con un costo computazionale αn^3.

⇝ THOMAS ALGORITHM

PRENDI NOTE A4

$$Px^{(k)} = \underbrace{(P-A)x^{(k-1)} + b}_{b'}$$

b' is known, because $x^{(k-1)}$ is already calculate and you can insert it -

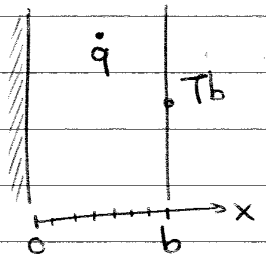
→ The convergence of the series depends on the nature of the matrix and on the choice of P that I make - Now I have to solve one problem but one iteration - The solution of $Px^{(k)} = b'$ costs such a lot of $Px = b$ in term of computational cost - Everything depends on the choice of P:

- $P = A$ ⇝ but computational cost remains the same
- $P =$ diagonal of A ⇝ Jacobi method

$$Dx^{(k+1)} = b - (A-D)\,x^{(k)} \qquad k \geq 0$$

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{\substack{j=1 \\ i \neq 1}}^{n} a_{ij} x_j^{(k)}\right) \qquad i = 1, \dots, n$$

LAIB 4



$$\frac{\partial^2 T}{\partial x^2} = -\frac{\dot{q}}{k}$$

$$\frac{\partial T}{\partial x} = -\frac{\dot{q}}{k}x + C_1$$

$$T(x) = -\frac{q}{2k}x^2 + C_1 x + C_2$$

$$\left.\frac{\partial T}{\partial x}\right|_0 = C_1 = 0$$

$$T(b) = -\frac{\dot{q}}{2k}b^2 + C_2 \rightarrow \quad C_2 = T_b + \frac{\dot{q}}{2k}b^2$$

Dirichlet on side B

Neumann on side A

Discretizzazione del dominio = numero pari di nodi, dispari di intervalli

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2}$$

PRENDI NOTE A4

$I = 50\,A$ T_0 → T_0 ↕ D

$L = 1\,m$ — L

$\rho = 1{,}75e-8$

$K = 350\ w/mK$ $\quad T_0 = 300\,K$

$D = 4mm = 4e-3\,m$

$$q'''_V = RI^2 = \rho \frac{L}{\frac{D^2\pi}{4}} I^2 \qquad \pi r^2$$

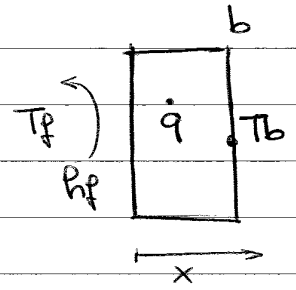
$$\frac{\delta^2 T}{\delta x^2} = -\frac{\dot{q}'''_V}{K} \qquad \frac{\delta T}{\delta x} = -\frac{\dot{q}'''_V}{K}x + C \qquad T(x) = -\frac{\dot{q}'''_V}{2K}x^2 + Cx + C_2$$

$$T(0) = T_0 = C_2$$

$$T(L) = -\frac{\dot{q}'''_V}{2K}L^2 + CL + T_0 \rightsquigarrow C = \frac{\dot{q}'''_V L}{2K}$$

$$T(x) = -\frac{\dot{q}'''_V}{2K}x^2 + \frac{\dot{q}'''_V L}{2K}x + T_0$$

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & & \\ & & \cdots & & \\ & & 1 & -2 & 1 \\ 0 & \cdots & & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} T_1 \\ \cdots \\ T_{end} \end{pmatrix} = \begin{pmatrix} -\dot{q}'''_V + T_0 \\ \cdots \\ -\dot{q}'''_V + T_0 \end{pmatrix}$$



$$T(x) = -\frac{\dot{q}}{2K}x^2 + Cx + C_2 \qquad \frac{\delta T}{\delta x} = -\frac{\dot{q}}{K}x + C$$

$$T_b = -\frac{\dot{q}}{2K}b^2 + Cb + C_2 \rightarrow C_2 = T_b + \frac{\dot{q}}{2K}b^2 - Cb$$

$$-K\left.\frac{\delta T}{\delta x}\right|_0 = h_f(T_0 - T_f) = -KC \rightarrow C = -\frac{h_f}{K}(T_0 - T_f)$$

$$T(x) = -\frac{\dot{q}}{2K}x^2 - \frac{h_f}{K}(T_0 - T_f)x + T_b + \frac{\dot{q}}{2K}b^2 + \frac{h_f}{K}(T_0 - T_f)b$$

$$T_0 = b\frac{h_f}{K}T_0 - \frac{h_f}{K}T_f b + T_b + \frac{\dot{q}}{2K}b^2 \rightarrow T_0 = \left(T_b + \frac{\dot{q}}{2K}b^2 - \frac{h_f T_f b}{K}\right) \Big/ \left(1 - \frac{bh_f}{K}\right)$$

PRENDI NOTE A4

CARACTERISTIC LENGTH OF THE GAUSSIAN $\sim \sqrt{Kt}$

CARACTERISTIC TIME $\sim x^2/K$

- Time variation is infinitesimal, but the temperature variation is really significant in that time interval.

~> Another solution obtained by the fondamental one is:

$$\begin{cases} u_t = K u_{xx} \\ u(x,0) = f(x) \end{cases}$$

$$u(x,t) = \int S(x-y,t) f(y)\, dy$$ It's a function of two arguments

(convolution integral)

$f(y)$ is the initial condition, and we have to multiply it. It's called "CONVOLUTIONAL". It evaluates the influence, the effect, that the condition imposed in y will have on the function S.

- IN QUESTO MODO PASSIAMO DALL' AVERE UN DOMINIO INFINITO ALL' AVERE UN DOMINIO FINITO, E PER QUESTO UTILE NUMERICAMENTE.

$$\begin{cases} u_t = K u_{xx} \\ u(x,0) = f(x) \\ a < x < b \end{cases}$$

To solve this problem we can use the method of lines.

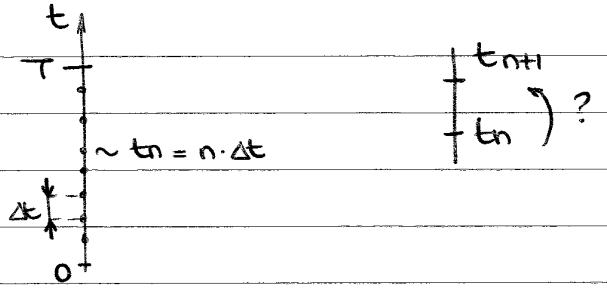
METHOD OF LINES PDE (x,t) 9.2.6

1) Discretization of the spacial coordinate, my space domain, in an uniform way with a finite number of nodes.
2) Approximation of u_{xx} with the finite differences method.

—> So the only derivative remained is the time one, become an ODE problem.

PDE (x,t) —(1,2)—> Set of ODE (t) —(3,4)—> Solve $Au = b$ at each time step

PRENDI NOTE A4



LAB 5

for $i = 1, \dots, \text{Size}(L)$ ⇝ ciclo per righe

  Known = 0

  for $j = 1, \dots, i-1$ ⇝ ciclo per colonne (così non conta gli zero alla fine di ogni riga

    $\text{Known} = \text{Known} + e_{i,j} \cdot x_j$

  end

  $x_i = \dfrac{[b_i - \text{Known}]}{e_{i,i}}$

end

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & -1 & 1 \end{pmatrix} \qquad Lx = b$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 3 \end{pmatrix}$$

PRENDI NOTE A4

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 43 di 86

Lez.

18/11/'14

1-D TRANSIENT CONDUCTION PROBLEM

- The Cauchy problem: we confine to first order differential equations as an equation of order $p>1$ can always be reduced to a system of p equations of order 1.

$$\begin{cases} \ddot{y} - 3\dot{y} + 5y = 0 \\ y(0) = 0 \\ \dot{y}(0) = 3 \end{cases} \qquad y(t)? \qquad \boxed{p=2}$$

$\dot{y} = z(t)$ → you can write this equation as:

$$\begin{cases} \dot{z} - 3z + 5y = 0 \\ \dot{y} - z = 0 \\ y(0) = 0 \\ z(0) = 0 \end{cases}$$

2 equations with $\boxed{p=1}$

- Concentrating on a system with just one equation:

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases} \qquad \forall t \in I$$

where I is an interval $\in \mathbb{R} \to \mathbb{R}$

y_0 is the initial data

Assume that the function is

- continuous with respect to both its arguments
- Lipschitz-continuous with respect to its second argument, there exists a positive constant L (named Lipschitz constant) such that

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2| \qquad \forall t \in I \ \forall y_1, y_2 \in \mathbb{R}$$

Then the solution $y = y(t)$ of the Cauchy problem exists, is unique and belongs to $C^1(I)$.

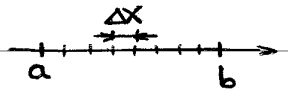
PRENDI NOTE A4

→ In terms of stability, we have advantages in using forward Euler, because of the explicity of the solution. There is another difference between the two methods:

$$\boxed{\frac{du}{dt} = \frac{K}{\rho c_p}\left(\frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}\right)}$$

One solution is easier than the other, because applying different methods we realize that one of them is more complicated in order to obtain the same algorithm.

BE $$\frac{u_i^{n+1} - u_i^n}{h} = \frac{K}{\rho c_p}\left(\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2}\right)$$



$$A u^{n+1} = b \qquad u^{n+1} = \begin{pmatrix} u_0^{n+1} \\ u_1^{n+1} \\ \cdots \\ u_n^{n+1} \end{pmatrix}$$

- Usando BE ottengo una soluzione funzione dell'istante n+1, così posso andare avanti perché u_n è noto (precedente)
  - ↳ Usassi il FE la soluzione sarebbe in funzione di n, ma a questo punto n+1 non lo conoscerei, sarebbe il futuro.

LAB 7

$$Px^{K+1} = (P-A)x^K + b \qquad K \geq 0$$

Jacobi

$$P = \text{diag}(\text{diag}(A))$$

Gauss-Siedel

$$P = \text{tril}(A)$$

$$B = P^{-1}(P-A)$$

$$g = P^{-1}b$$

PRENDI NOTE A4

$$x^3 = \begin{pmatrix} 0 & -72 & -144 \\ -108 & 0 & -216 \\ -48 & -240 & 0 \end{pmatrix}\begin{pmatrix} -157464 \\ -224748 \\ -203184 \end{pmatrix} + \begin{pmatrix} 648 \\ 756 \\ 720 \end{pmatrix} = \begin{pmatrix} 45440352 \\ 60893856 \\ 61497792 \end{pmatrix} + \begin{pmatrix} 648 \\ 756 \\ 720 \end{pmatrix} = \begin{pmatrix} 45441000 \\ 60894612 \\ 61498512 \end{pmatrix}$$

$$B = \begin{pmatrix} 1/6 & 0 & 0 \\ 0 & 1/8 & 0 \\ 0 & 0 & 1/9 \end{pmatrix}\begin{pmatrix} 0 & -1 & -2 \\ -2 & 0 & -4 \\ -1 & -5 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -1/6 & -1/3 \\ -1/4 & 0 & -1/2 \\ -1/9 & -5/9 & 0 \end{pmatrix} \qquad g = \begin{pmatrix} 1/6 & 0 & 0 \\ 0 & 1/8 & 0 \\ 0 & 0 & 1/9 \end{pmatrix}\begin{pmatrix} 9 \\ 14 \\ 15 \end{pmatrix} = \begin{pmatrix} 3/2 \\ 7/4 \\ 5/3 \end{pmatrix}$$

$$x^1 = \begin{pmatrix} 0 & -1/6 & -1/3 \\ -1/4 & 0 & -1/2 \\ -1/9 & -5/9 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 3/2 \\ 7/4 \\ 5/3 \end{pmatrix} = \begin{pmatrix} 3/2 \\ 7/4 \\ 5/3 \end{pmatrix}$$

$$x^4 = \begin{pmatrix} 0 & -1/6 & -1/3 \\ -1/4 & 0 & -1/2 \\ -1/9 & -5/9 & 0 \end{pmatrix}\begin{pmatrix} 3/2 \\ 7/4 \\ 5/3 \end{pmatrix} + \begin{pmatrix} 3/2 \\ 7/4 \\ 5/3 \end{pmatrix} = \begin{pmatrix} 0,6527 \\ 0,5416 \\ 0,527 \end{pmatrix}$$

$$x^3 = \begin{pmatrix} 1,2338 \\ 1,3229 \\ 1,2932 \end{pmatrix}$$

$Ax = b$

$r^k = b - Ax^k$ ← residual vector B = iteration matrix

Lez.

25/11/'14

CONVERGENCE ANALYSIS

- A numerical method is convergent if

$$\boxed{\forall n = 0, \dots, N_h} \quad |y_n - u_n| \le C(h) \qquad C(h) \text{ infinitesimal for } h \to 0$$

Backward and forward Eulero are useful only if the method is convergent – y_n is the exact solution, while u_n is the computed solution

→ We want to analyse the error for forward Eulero method, and we will divide it in two parts:

$$e_n = y_n - u_n = (y_n - u_n^*) - (u_n^* - u_n)$$

PRENDI NOTE A4

The same formula can be rewritten expanding f as a function of y getting:

$$u_n^* - u_n = \left(1 + h\frac{\partial f}{\partial y}(t_{n-1}, y_n)\right) e_{n-1}$$

Discussing the inequality

$$0 < h < 2 / \max_{t\in[t_0,T]} \left|\frac{\partial f}{\partial y}(t, y(t))\right|$$

$\left|1 + h\frac{\partial f}{\partial y}\right| < 1$ CONDIZIONE NECESSARIA AFFINCHÉ LA SOLUZIONE SIA STABILE

$\left|h\frac{\partial f}{\partial y}\right| < 2$

$$|u_n^* - u_n| = e_{n-1} + hL(y_{n-1} - u_{n-1}) \le |e_{n-1}|(1 + hL)\alpha q$$

$\alpha < 0{,}5$

THE CRANK-NICOLSON METHOD

- $u_{n+1} = u_n + h f_n$ (BACKWARD)
- $u_{n+1} = u_n + h f_{n+1}$ (FORWARD)

$$\boxed{u_{n+1} = u_n + \frac{h}{2}(f_n + f_{n+1})}$$ → it's second order accurate in time, different from the other methods

sparisce h perché non stai più analizzando due nodi diversi separati da h allo stesso istante, ma lo stesso nodo in due istanti diversi.

STABILITY ON UNBOUNDED INTERVAL

- We refer to a model problem, a generical Cauchy problem:

$$\begin{cases} y'(t) = \lambda y(t) & t \in (0,\infty) \\ y(0) = 1 \end{cases}$$ unbounded, no limits

PRENDI NOTE A4

Lez.

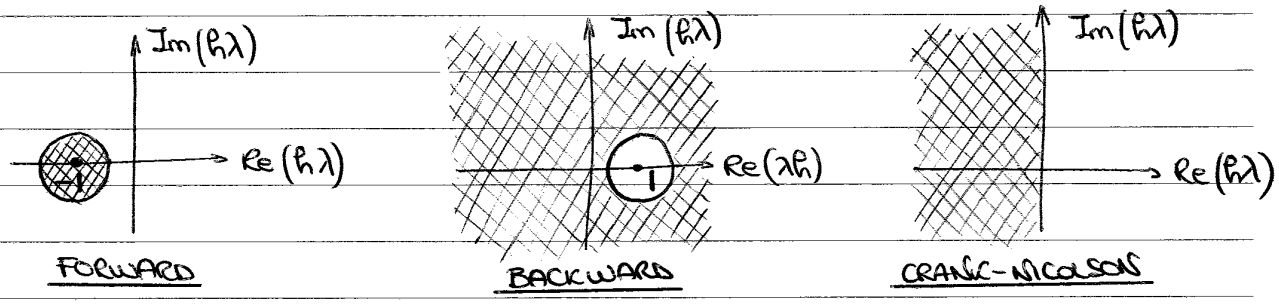
26/11/'14

- Consistency property is trivial evaluated, while stability property is a key property. Absolute stability is a property of the scheme that we want to be a property of the problem.

→ λ is linked to Jacobian

THE REGION OF ABSOLUTE STABILITY

- If λ is a complex number with negative real part:



Are those portions of the Im-Re plain in which the method is absolutely stable.

⇝ $\forall (h\lambda) \in$ blue region gaurantees stability

Forward method: $|1+h\lambda| < 1$ corresponds to a circle in which we can take internal points

Backward method: is A-stable method, because ALL the points with negative real part make the method stable

Crank-Nicolson: is also an A-stable method

Which are the limitations?

→ Crank-Nicolson is a method in which error goes to zero like h^2

→ Forward and Backward euler have errors that go to zero like h

⇝ Computational methods use methods with higher order speed of error going to zero.

PRENDI NOTE A4

$$K_1 = f_n$$

$$K_2 = f\left(t_n + \frac{h}{2}, u_n + \frac{h}{2}K_1\right)$$

$$K_3 = f\left(t_n + \frac{h}{2}, u_n + \frac{h}{2}K_2\right)$$

$$K_4 = f\left(t_n + 1, u_n + hK_3\right)$$

coefficienti di h int				coefficienti di h in y
0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	1/3	1/3	1/6

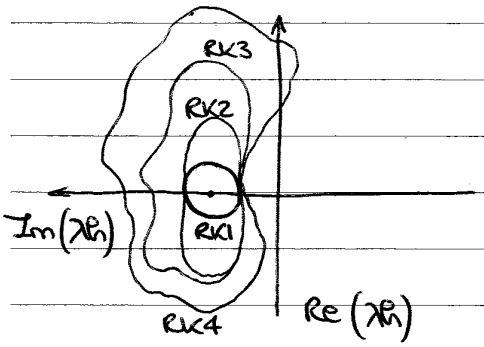
→ It's a 5 time program, compared more iterations respect RE and FE, but gives us results with more accuracy.

$b = (1/6 \;\; 1/3 \;\; 1/3 \;\; 1/6)$ sono i valori ottenuti moltiplicando i coefficienti delle K per $\frac{1}{6}$ (nella RK in analisi)

→ $\frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4) = \frac{1}{6}(1+2+2+1) =$

$= 1/6 + 1/3 + 1/3 + 1/6 = b_1 + b_2 + b_3 + b$

ABSOLUTE STABILITY REGION



Internal points contained in the circle

MULTI-STEP METHODS

- Are methods that let us to solve a problem with many ODE equations and include the single-step method:

$$u_{n+1} = \sum_{j=0}^{p} a_j u_{n-j} + h \sum_{j=0}^{p} b_j f_{n-j} + h b_{n-1} f_{n+1} \qquad n = p, p+1, \ldots$$

PRENDI NOTE A4

• In AM methods where are 1 and 2? AM=1 is Backward Euler and AM=2 Crank-Nicolson, and those methods are A-Stable!

→ AM absolute stability region is larger than the correspondent AB absolute stability region (a parità di n).

STIFF PROBLEMS = initial value problem caracterized by separate time SKILLS.

$$\begin{cases} y'(t) = \lambda\left(y(t) - g(t)\right) + g'(t) & t>0 \\ y(0) = y_0 \end{cases}$$

where g is a regular function and $\lambda<0$ has a very large absolute value, whose solution

$$y(t) = \left(y_0 - g(0)\right)e^{\lambda t} + g(t) \quad t>0$$

is the sum of two components, also called TRANSIENT and PERSISTENT solution. In particular, we set $g(t)=t$, $\lambda=-100$ and $y_0=1$ and solve the problem in the interval $(0,100)$ using the explicit Euler method. Respecting the stability condition solution is linear, while violating it the solution "explodes" losing its clearness.

• In transient problems we obtain a set of ODE, one for each internal node of our space discretization. So we need a set of initial conditions, one for each equation. In these equations the role of λ is played by the eigen value of the Jacobi matrix.

θ METHOD $$\boxed{u_{n+1} = u_n + h\left(\theta F(t_{n+1}, u_{n+1}) + (1-\theta)F(t_n, u_n)\right)} \quad n\geq 0$$

Let us to see in a single formula the 3 previous methods (BE, FE, CN) for a set of differential equations.

→ If $\theta=0$ is FORWARD EULER → If $\theta=1/2$ is CRANK-NICOLSON
→ If $\theta=1$ is BACKWARD EULER

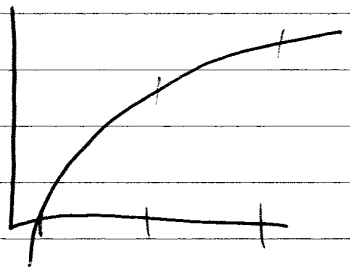
PRENDI NOTE A4

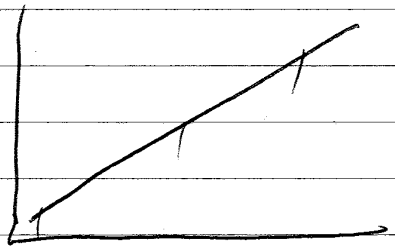
LAIB 8

>> spy compare un cane >> penny compare un penny

(1)

$$u_t = K u_{xx} \qquad \frac{du}{dt} = K \frac{d^2u}{dx^2} \qquad S(x,t,K) = \frac{e^{-\frac{x^2}{4Kt}}}{\sqrt{4\pi K t}}$$





Use loglog Scale. Varia la scala degli assi, non il dominio.

(2) Traslo il centro della fundSol per avere la soluzione nei punti del dominio scelti come nodi. Otteniamo 4 fundamental solutions in 4 different nodes.

Lez.

2/12/'14

9.2.6 FINITE DIFFERENCE APPROXIMATION OF THE ONE-DIMENSIONAL HEAT EQUATION

$$\begin{cases} \dfrac{u^{K+1} - u^K}{\Delta t} = -\dfrac{\mu}{h^2} A\left(\Theta u^{K+1} + (1-\Theta)u^K\right) + \Theta f^{K+1} + (1-\Theta) f^K \\ u^0 \text{ given} \qquad K = 0, 1, \dots \end{cases}$$

$I u^{K+1} = g(u^K, f^K)$ → so we obtain a set of equation that form a diagonal matrix

$$\boxed{A u^{K+1} = b}$$

(A: diagonal)

BACKWARD EULER

$$\boxed{\frac{u_i^{K+1} - u_i^K}{\Delta t} = \frac{\mu}{h^2}\left(u_{i-1}^{K+1} - 2u_i^{K+1} + u_{i+1}^{K+1}\right) + f_i^{K+1}}$$

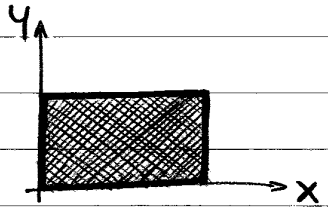
$i = 1, \dots, N$ where N = internal nodes

PRENDI NOTE A4

FINITE DIFFERENCE

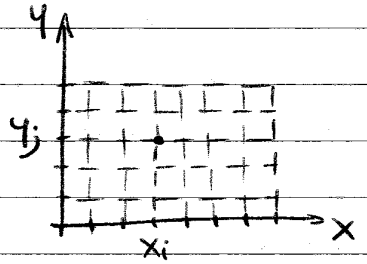
- It makes us able to find the solution only if the shape of the domain is simple (RECTANGULAR or union of RECTANGULARS)

→ For complex shapes of domain we can introduce FINITE ELEMENTS method.



$$0 = K(\nabla^2 T) + q$$

$$\nabla^2 T = \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)$$



$$T(x,y) \rightarrow T(x_i, y_j) \approx T_{ij}$$

$$\begin{cases} 0 = K\nabla^2 T + q \\ \text{Boundary conditions} \end{cases}$$

- I can obtain in this way the algebric solution by solving node by node the problem sostituating each time T_{ij} in its corrispondent node in the grid.

$$\nabla^2 T\Big|_{i,j} = \frac{T_{i-1,j} - 2T_{ij} + T_{i+1,j}}{h_x^2} + \frac{T_{i,j-1} - 2T_{ij} + T_{i,j+1}}{h_y^2}$$

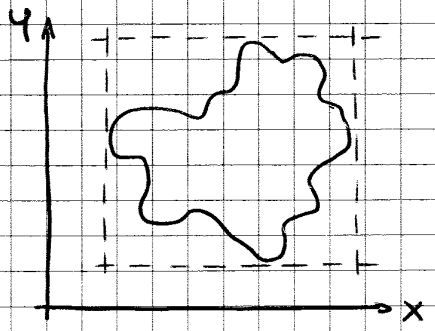
$$\begin{cases} 0 = K\left(\dfrac{T_{i-1,j} - 2T_{ij} + T_{i+1,j}}{h_x^2} + \dfrac{T_{i,j-1} - 2T_{ij} + T_{i,j+1}}{h_y^2}\right) + q(x_i, y_j) \\ \forall i,j \text{ corrisponding to internal nodes} \end{cases}$$

$$\boxed{AT = b} \qquad A = \text{coefficient matrix}$$

- Where are the non-zero elements? It depends.

NEUMANN / ROBIN BOUNDARY CONDITION

$\boxed{\frac{dT}{dx} = 0}$ ⇝ cases against Dirichlet condition, that specify the value of the temperature



In this case finite difference won't give an identification of the domain and of a series of rectangles -

We are in problems that use a symmetric and positive operator (second order derivative)

[C. Johnson, "Numerical solutions of PDEs by the finite elements method" (Cambridge UP, 1987)

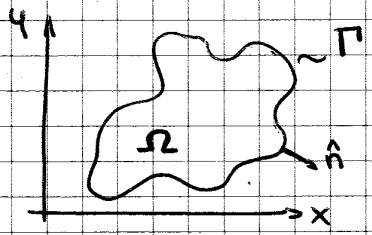
$$\begin{cases} -\Delta u = f \text{ in } \Omega \\ u = 0 \text{ on } \Gamma \end{cases}$$

Ω = domain

Γ = boundary of domain

↳ simple, Dirichlet boundary condition

$-v\Delta u = fv$



$(x,y) \in \Omega \subset \mathbb{R}^2$

$u(x,y)$?

$v(x,y)$

$$\int_\Omega -v\Delta u = \int_\Omega fv$$

$\nabla \cdot (v\nabla u) = \nabla v \cdot \nabla u + v\Delta u$

$-v\Delta u = \nabla v \nabla u - \nabla \cdot (v \cdot \nabla u)$

$$\int_\Omega \nabla v \nabla u - \int_\Omega \nabla \cdot (v\nabla u) = \int_\Omega fv$$

$$\int_\Gamma v\, Du \cdot n \rightsquigarrow \frac{\partial u}{\partial n}$$

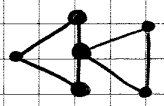
If $v(x,y)$ arbitrary → $v|_\Gamma = 0$

WEAK FORMULATION OF ORIGINAL PDE

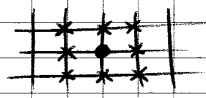
⇝ $\boxed{\int_\Omega \nabla v \nabla u = \int_\Omega fv}$ $\forall v \in V$ (find $u \in V$)

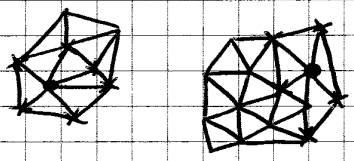
- In order to create the mesh I have to follow some rules:

  * triangles have to be equilateral (meno isotropo) –
    └ so the created error will be minimum
  * mesh has not to leave some spaces void, domain must be full –
  * triangles have not to be overposed in the domain –
  * Vertices of triangles mustn't create "linear triangles"
    └ un vertice non può mai essere sul lato di un altro triangolo, non deve crearsi questa situazione:



→ The number of neighbours in this kind of discretization is not fixed while in the linear discretization is fixed –

STRUCTURAL MESH 

UNSTRUCTURAL MESH 

- How do these meshes be stored in the computer?

$$\#\text{ nodes}\begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \dots & \dots \\ x_{29} & y_{29} \end{pmatrix} \qquad \#\text{ triangles}\begin{pmatrix} & & \\ & & \\ 19 & 3 & 18 \\ & & \end{pmatrix} \text{ row } (S)$$

↳ in ogni riga di questa matrice metto i vertici del triangolo
↳ sto guardando il triangolo S

$$T_h = \bigcup_{K \in T_h} = K_1 \cup K_2 \cup \dots \cup K^n$$

$$h = \max_{K \in T_h} \text{diam}(K) \qquad \text{diam}(K) = \text{longest side of } K$$

↳ il tratto più lungo della discretizzazione del dominio, in Sedoni –

$$V_h = \{ v : v \in \mathcal{C}^0(\Omega)$$
$$v|_K \text{ is linear for } \forall K \in T_h$$
$$v|_\Gamma = 0 \}$$

POLYGONAL APPROXIMATION OF THE BOUNDARY

$$a(u,v) = (f,v) \qquad \boxed{u(x,y) = \sum \xi_i \varphi_i(x,y)}$$

$$a\left(\sum \xi_i \varphi_i, \varphi_j\right) = (f,\varphi_i) \quad j = 1,\dots,n \text{ of internal nodes}$$

$$\sum \xi_i\, a(\varphi_i,\varphi_j) = (f,\varphi_i)$$

$$\boxed{A\xi = b} \qquad A_{ij} = a(\varphi_i,\varphi_j) \quad b_i = (f,\varphi_i)$$

$$\int_\Omega \nabla\varphi_i \nabla\varphi_j = a(\varphi_i,\varphi_j) \qquad \int_\Omega f\varphi_i = (f,\varphi_i)$$

$\xi_i = u_h(x_i,y_i) \approx u(x_i,y_i)$ ~ approximation of the temperature in the node (i,j) in the domain

- A is going to be sparse, and if I use test (function) in un nodo posso ottenere in un colpo solo tutti i valori non nulli contenuti, cioè quelli dei triangoli aventi come vertice quel nodo
  - ↳ limited number of unknows respect to the number of triangles

⇝ I obtain the number of neighbourds for each node, that is given by the number of non-zero elements.

THERE IS NOT A STRUCTURED SPARSITY PATTERN, is not regular!

- We can't use direct methods (full matrix), but we have to use iterative methods (sparse matrix).

How accurate I expect my solution to be? In function of the approximation chosen, and the error will be represented by $u_h - u$, in function of h:

$$\boxed{u_h - u \to 0} \quad \text{as } h^2$$

The process to build the matrix A and the vector b is divided in two steps, in order to be able to assemble them.

1) Build the elemental matrix Ael

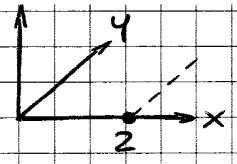
2) Assemble A

$$\int_\Omega \approx \int_{(Uk)} = \sum_{i=1}^{n} \int_{K_i}$$

T_e

↳ assemble

② $\partial/\partial y$



Non passano piani lungo l'asse y che contengono 2, quindi $\frac{\partial \varphi_2}{\partial y} = 0$

	N	NW	W	S	SE	E	P	
1° triangolo	(-1/2)	0	0	0	0	(-1/2)	(1)	
2° triangolo	-1/2	0	0	0	0	0	1/2	
	0	0	-1/2	0	0	0	1/2	
	0	0	-1/2	-1/2	0	0	1	
	0	0	0	-1/2	0	0	1/2	
	0	0	0	0	0	-1/2	1/2	
	-1	0	-1	-1	0	-1	4	→ SOMMA!

Attraverso A ci saranno 7 elementi non nulli, uno in posizione P e gli altri in N, NW, W, S, SE, E.

Scelgo un triangolo e così proseguo in senso antiorario scendendo riga per riga.

LAIB 9

① BISECTION METHOD → Non linear problems

- Only one root in a chosen $[a,b]$
- They are continuous in $[a,b]$

→ It never loses its solution; if it is not found, the interval is split in two subintervals and the process is repeated

To check which interval contains zero we have to check sign:

$$x_m = \frac{a+b}{2}$$

$f(x_m) \cdot f(a) < 0$? — YES, New subinterval $[a' : b'] = [a : x_m]$
— NO, New subinterval $[a' : b'] = [x_m : b]$

... and so on!

Fuiché: ○ $f(x_m) = 0$

○ nº iterations $It_{max} > \log_2\left(\frac{b-a}{\varepsilon}\right) - 1$ ε = tolerance desired

P_1 → Linear interpolation in 2-D (plane)

$a_0 + a_1 x + a_2 y$ where a_0, a_1, a_2 correspond to values linked to the vertices of the triangle

P_2 $a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 y^2 + a_5 xy$

→ For secondary interpolation we need 6 coefficients, because we have 6 degrees of freedom.

- The basis of P_2 is made by three functions which assume the value 1 at the vertices and 0 every other where. We need now another set of functions assuming the value 1 in the center of every side of the triangle.

→ We imagine finite element solution in P_1 context like a diamond surface, but passing in P_2 context how can we imagine that?

- We are not solving the original PDE, but the weak formulation of the original PDE. So the flux continuity between different interfaces is not guaranteed, and we are not able to conserve the local error, solution is good but we can't go after a certain precision.

L solution can't be exact, but is also not wrong

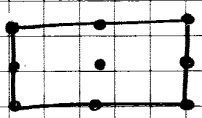
New triangularization form of the domain (P_2)

BUT triangles is not the best shape we can use to discretize the domain, we can naturally use a quadrilateral discretization that is called Q

Q_1 

Flow in a pipe, for example

$a_0 + a_1 x + a_2 y + a_3 xy$

Q_2 

Not so linear polinomial, it's more complex of the Q_1 discretization.

$0 = \nabla^2 u + q$ becomes $0 = K_x u_{xx} + K_y u_{yy} + q$

```
if (condition)
else if (condition)
    else
```

Boolean conditions:

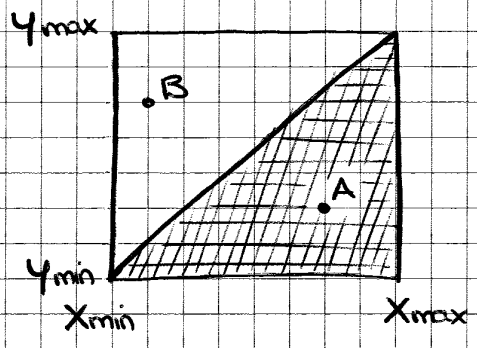
AND	&&	(MatLab &)
OR	||	(MatLab |)

- Causal logic, what does it imply? That unknows have to be on the left side and what is known has to be on the right side.

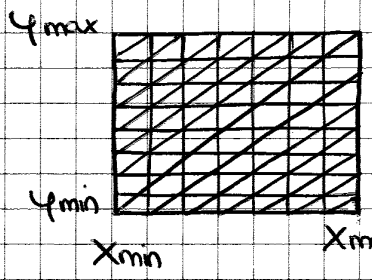
→ CAUSAL LANGUAGE capisce qual è l'incognita, non deve per forza venire scritta in forma esplicita, lo capisce dal contesto.

```
y = 2*x + 5;
```
└ entrambe possono essere incognite

```
int region1 = meshname(xin, yin).region;
int region2 = meshname(xin, yin).region;
```




```
fespace Vh(meshname, P1);
Vh Kappa;
```
└ assegno le variabili che variano il loro valore nel corso dello spostamento nel mesh.




```
Kappa = K1*(region==region1) + K2*(region==region2);
```
└ Boolean instruction

```
plot(Kappa, wait=1, value=1, fill=1);
```

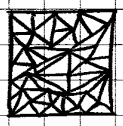
FreeFem++ script file extension: .edp

‖ PLEASE SAVE THE SCRIPTS BEFORE RUNNING THEM!

Ex1_YourSurname.properextension

→ For plot or figures we have to use .eps format extension.

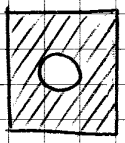
```
border cd (t=0,1){ x = xmax + (xmin-xmax)·t ; y = ymax;};
border da (t=0,1){ x = xmin ; y = ymax + (ymin-ymax)·t; };

mesh pippo = buildmesh (ab(nodes) + bc(nodes) + cd(nodes) + da(nodes));
plot (pippo);
```



```
border eee (t=0,2*pi){ x = (xc) + raduis·cos(t); y = (yc) + raduis·sin(t);};
```

c = centro — posta come nota = 1.5 — = 0,25 — nodo e uguale a 3

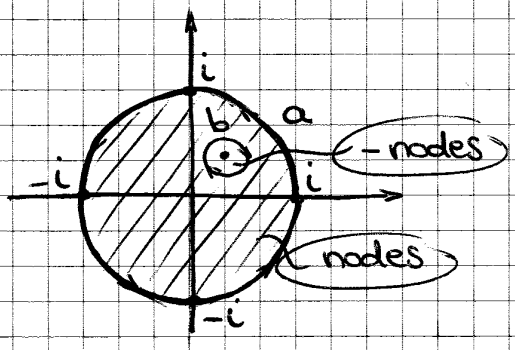
```
mesh pippo = buildmesh(ab(nodes) + bc(nodes) + cd(nodes) + da(nodes) +
                                                  + eee (-nodes));
plot (pippo);
```



Serve per togliere la grigia

- Posso anche aggiungere nella definizione del border una label [label = name;]; e richiamare quel border tramite label e non name.

DEFINE FE SPACE




```
fespace Vh (Th, P1);
Vh uh, vh;

fespace minniemouse (pippo, P1);
minniemouse myu, myv, myef;
myef = 100·sin(pi·x·(1-x))·sin(pi·(y-2.)·(4-y)); plot (myef, fill=1, value=
```

```
problem poisto(myu,myv) = int2d(pippo)(dx(myu)·dx(myv) +
                                + dy(myu)·dy(myv)) - int2d(pippo)·
                                ·(myef·myv) + on(ab, bc, bcd, da, myu=0)
```

Scrivi il problema originale

integrale in 2D

LAIB 11  18/12/14

SOLVING 1D HEAT EQUATION BY FINITE DIFFERENCE

(1)

$$\rho c_p \frac{\partial T}{\partial t} = K \frac{\partial^2 T}{\partial x^2} + q'''$$

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} + \frac{q'''}{\rho c_p} \rightarrow \frac{\partial T}{\partial x} = f(T, x, t)$$

$L = 10\ cm = 0{,}1\ m$

$(=\alpha)\ K = 0{,}1\ cm^2/s = 0{,}1 \cdot 10^{-4}\ m^2/s$

$q''' = 0\ W/m^3$

$T(x,0) = T_0 + \sin(\pi x/L)$

$T_0 = 300\ K$

$\Delta t = 0{,}1\ s$

$\Delta x = 0{,}1\ cm$

→ USARE SOLO UNITÀ DEL SI!!

No (cm²), Sì (m²)

BOUNDARY CONDITION

$$\frac{\delta T_i}{\delta t} = \alpha \frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2} + \frac{q'''}{\rho c_p} + \frac{h(A)}{(\rho V)}(T_f - T_i)$$

A: AREA DI CONTATTO TRA SOLIDO E FLUIDO

V: VOLUME DELL'ELEMENTO i (SOLIDO)

↳ per far tornare le dimensioni

- $(T_f - T_i)$ o $(T_i - T_f)$?

↳ se $(T_f - T_i) > 0$ sto scaldando il solido, $(\delta T/\delta t)$ positiva

→ Bisogna usare $\boxed{(T_f - T_i)}$

FORWARD EULER

$$\begin{pmatrix} T_1 \\ T_2 \\ \dots \\ T_{end} \end{pmatrix}^{n+1} = A \begin{pmatrix} T_1 \\ T_2 \\ \dots \\ T_{end} \end{pmatrix}^{n} + b = \begin{pmatrix} BC & BC & 0 \\ a & 1-2a & a \\ 0 & \diagdown & \diagdown \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ \dots \\ T_{end} \end{pmatrix}^{n} + \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_{end} \end{pmatrix}$$

unconditionally stable

$$T^{n+1} = AT^n + b \rightarrow T^n = T^{n+1}$$

↳ non ho più il back slash

- Con (BE) posso usare il Δt che voglio, con (FE) no!

↳ conditionally stable

$\left(\frac{\alpha \Delta t}{\Delta x^2} \leq \frac{1}{2}\right)$ → condizione da rispettare

ALL'ESAME VERIFICARE SEMPRE LA CONVERGENZA!

(1) Parto da S1 e da S2 in poi devono essere quelli consecutivi

(2) Vado in senso antiorario, quindi devo parametrizzarli in quel senso, altrimenti basta mettere $(-r)$ al posto di (r), così costruisce i triangoli nell'altro verso.

→ Per parametrizzare i lati ragionare come una retta tra 2 punti.

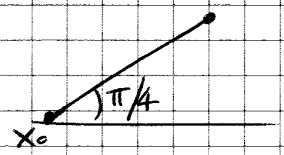
EX3 $D_{in} = 15\,cm$
$D_{out} = 60\,cm$
$S = 5\,cm$

border aa $(t = 0, 2\pi)\{x = r\cos t;\ y = r\sin t\}$;
(tutto il giro)

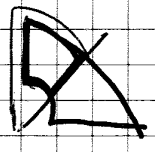
border S1 $\frac{D_{in}}{2}$ $\frac{D_{out}}{2} - S$

EX4 È simmetrico ogni 45°, basta uno specchio!

border aa $(t=0, pi/4)\{x = 2\cos t;\ y = r\sin t;\}$
border bb $(t = 0, pi/4)\{x = \frac{S}{2} + r_{in}\cos t;\ y = r_{in} - r_{in}\sin t;\}$
~~border cc $(t = r_{in}, r_{out})\{x = r_{in} + t;\ y =$~~
border ee $(t = r_{in}, r_{out}-S)\{x = \frac{S}{2};\ y = r_{in} + t;\}$
border ff $(t=0, pi/4)\{x = (r_{out}-S)\cos t;\ y = (r_{out}-S)\sin t;\}$
border gg $(t=0, pi/4)\{x = (r_{in}\cos t);\ y = r_{in}\sin t;\}$
border cc $(t=0, (r_{out}-S-r_{in}))\{x = r_{in}\cos(pi/4) + t\cos(pi/4);\ y = r_{in}\sin(pi/4) + t\sin(pi/4);\}$



$x = x_0 + t\cos\pi/4$
$y = y_0 + t\sin\pi/4$



(BE) → nessun problema con il timestep, è **A-STABLE**

(FE) → ho dei vincoli da rispettare, se Jacobiana è costante (lineare) no problem, se problema non lineare Jacobiana non è costante e quindi ad ogni timestep devo controllare di essere all'interno della regione di stabilità del metodo_

$$h < \left(\frac{\Delta x}{2}\right)^2 = \frac{2}{|\partial f/\partial y|}$$

STABILITY CONDITIONS

- Ellitic equations → no preferential direction
- Iperbolic equations → preferential direction

$$\boxed{v \cdot \left(-\nabla^2 u\right) = v \cdot f}$$

u = temperature
f = heat source

f però è solo q''', perché la parte funzione del tempo $\rho c_p \frac{dT}{dt}$ va trattata in un altro modo_

$$\int \nabla u \nabla v + \int v \frac{du}{d\Omega} = \int f v$$

$$\oint_{\partial\Omega} v(\nabla u \cdot \hat{n}) = \oint_{\partial\Omega} v \frac{\partial u}{\partial n}$$

e va a zero non perché la derivata è nulla (quindi adiabaticità), ma perché la v è per ipotesi nulla sul bordo del dominio_

- The solution of $Ax=b$ is really sensible to small perturbation (if it is ill-conditioned)_ The machine precision is a finite number, so each data in entrata is influenced by the machine precision ε_

$K(A) \cdot \varepsilon$ → Così l'incertezza che potremmo avere aumenta, non possiamo garantire una buona accuratezza_

$K(A)$: valore minimo da avere

(1D) Transient Heat Conduction

$$u_t = -K u_{xx} \qquad \boxed{\Delta t \leq \frac{(\Delta x)^2}{2K}}$$

notare che essendo valido $\lambda = \frac{K}{(\Delta x)^2}$ → $\Delta t \leq \frac{2}{\lambda}$

(diverso coefficiente, ma valido)

→ L'errore non va a zero come x o x^2 perché sotto certi valori viene dominato dai Δt, e viceversa! Ci vorrebbero diverse soluzioni per diversi Δx e Δt_

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 83 di 86

2D - TRANSIENT

$$\int_{Th} \rho c_p \frac{dT}{dt} v \, dx\,dy + \int_{Th} K \nabla u \nabla v \, d\Omega - \int_{Th} v f \, dx\,dy = 0$$

- Using the method of lines:

$$\int_{th} \rho c_p \frac{\delta u}{\delta t} v \, dx\,dy \approx \int_{Th} \rho c_p \frac{u}{\Delta t} v \, dx\,dy - \int_{Th} \rho c_p \frac{u_{old}}{\Delta t} v \, dx\,dt$$

int2d(Th)(2ho * cp * u/dt) - int2d(Th)(2ho * cp * uold/dt)

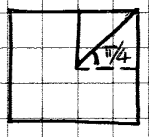
USE BACKWARD EULER

int2d(Th)(ρcpu/dt) - int2d(Th)(ρcp uold/dt) + int2d(K(dx(u)dx(v) + dy(u)dy(v)))
- int2d(Th)(fv) + on(...)

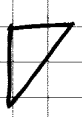
→ DEVO USARE CICLI **FOR** (se so numero iterazioni) o **WHILE** (se ho tolleranza) e devo AGGIORNARE LE VARIABILI (uold = u) ALLA FINE DELL'ITERAZIONE

(1) NB il Δt dell'essere dell'ordine di $\frac{L^2}{\alpha}$ per avere convergenza

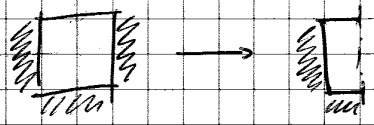
→ $\Delta t \cong \frac{L^2/\alpha}{10}$



posso dividerlo in 8 parti simmetriche:



ma avendo condizioni al contorno devo avere metà dominio!



Told = Tin; → variabile di Th

mesh ...

DEFINE
fe space ...

DEFINE
problem $T1^{(T,v)}$ = int2d( )(Tcp/dt + αρcp(dx(T)dx(v) + dy(T)dy(v))) - int2d( )
(cp ρ Told/dt) + on(, T = Tc);