INTEGRAZIONE NUMERICA

- Obiettivo: calcolare valore di ***integrale definito*** di una funzione f

- Integrale viene calcolato mediante ***insieme discreto di valori noti*** di f nell'intervallo dato

Uso di tecniche numeriche:

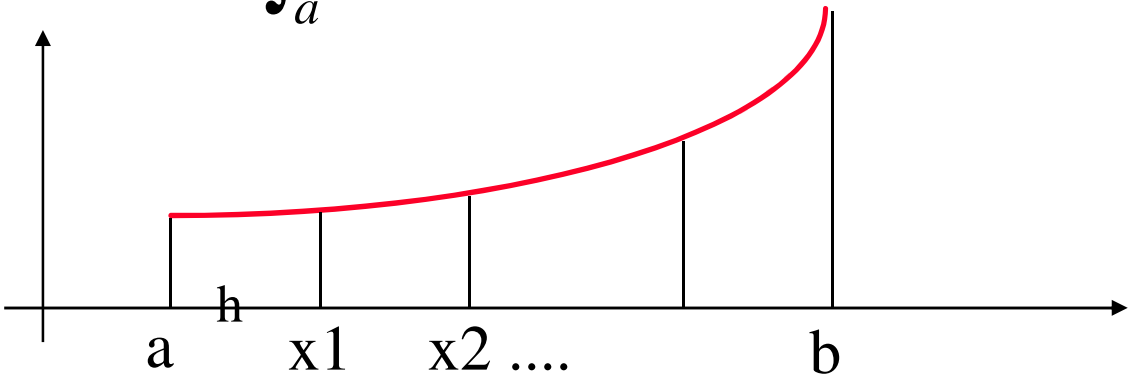
  - se integrale della funzione è ***difficilmente calcolabile***
  - se funzione è ***nota solo per punti***

- Il problema dell'integrazione viene spesso risolto ***rappresentando la funzione mediante una formula di interpolazione*** e poi integrando questa

INTEGRAZIONE NUMERICA

- Data la funzione f(X) integrabile nell'intervallo [a,b] calcolare:

$$I = \int_a^b f(x)dx$$



- Si suddivide l'intervallo (a,b) in ***n sub-intervalli di ampiezza h*** (h=(b-a)/n)
- Si ottiene la successione di valori:

$X_0 = a, X_1, X_2, ..., X_n=b$

con $X_i = X_{i-1} + h$ per i=1,2,...,n

INTEGRAZIONE NUMERICA

- In ogni intervallo i-simo (X_{i-1},X_i) si ***approssima f(X) con una funzione $F_i(X)$***:

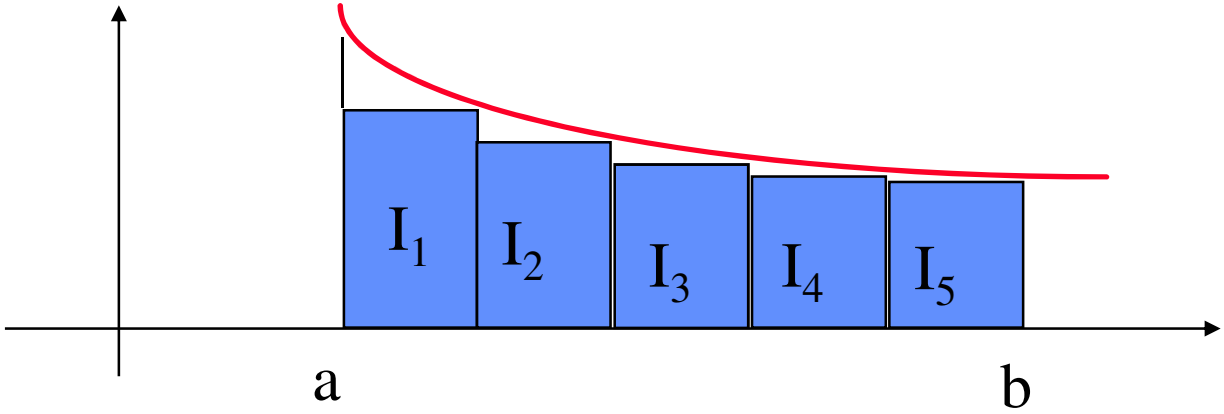
$$I \approx \int_a^{X_1} F_1(X)dX + \int_{X_1}^{X_2} F_1(X)dX + \cdots$$

$$+ \int_{X_{n-1}}^{b} F_n(x)dX = \sum_{i=1}^{n} \int_{X_{i-1}}^{X_i} F_i(X)dX$$

- ***Funzioni approssimanti F(X) differenti producono metodi approssimanti differenti***

METODO DEI RETTANGOLI

- $F_i(X) = f(X_i)$ i=1,2,...,n



Si approssima ogni areola i-sima sottesa dalla funzione f con ***l'area del rettangolo di base h ed altezza $f(X_i)$***: in ogni sub-intervallo, ***funzioni approssimanti sono delle costanti***

$$I_i = f(X_i).h \qquad i=1,2,...n$$

$$I \approx \sum_{i=1}^{n} f(X_i) \cdot h = h \cdot \sum_{i=1}^{n} f(X_i)$$

INTEGRAZIONE CON METODO DEI RETTANGOLI

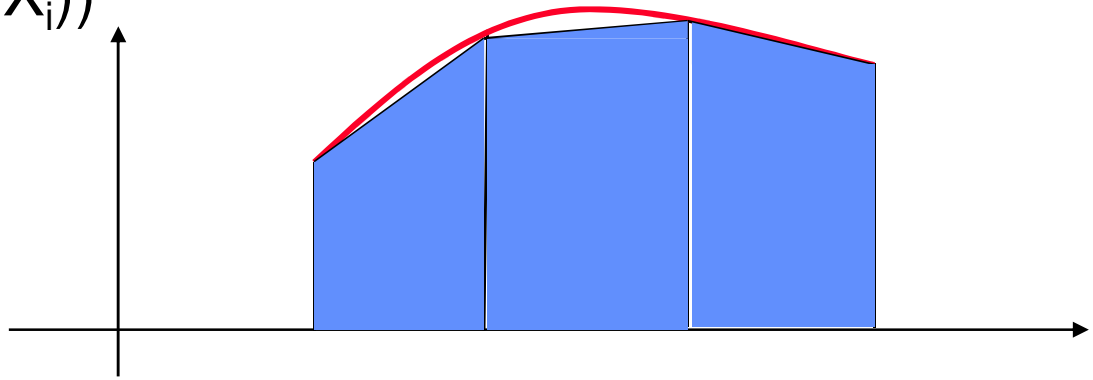
```
float func(float y) {
   return y*y-sin(y); }

float rettangoli (float a, float b, int n) {
/*func funzione integranda; a,b estremi dell'intervallo,
   n numero dei punti */

   float x,sum,h;int j;
   h=(b-a)/n;
   x=a;
   for (sum=0.0,j=0;j<n;j++,x+=h) sum = sum+h*func(x);
   return sum;
}
```

METODO DEI TRAPEZI (Bezout)

- $F_i(X)$ = retta passante per i punti di coordinate $(X_{i-1}, f(X_{i-1}))$, $(X_i, f(X_i))$



Si approssima l'i-sima areola sottesa dalla funzione f con ***trapezio ottenuto collegando i due punti estremi con una retta***

$F_i(X) = f(X_{i-1}) + ((f(X_i)-f(X_{i-1}))/h)*(X-X_{i-1})$

$I_i = (h/2)* (f(X_{i-1}) + f(X_i))$ i = 1,2,...n

$$I \approx \frac{h}{2}\left(f(X_0)+f(X_1)\right)+\frac{h}{2}\left(f(X_1)+f(X_2)\right)+\ldots+\frac{h}{2}\left(f(X_{n-1})+f(X_n)\right)=$$

$$=h\left(\frac{f(X_0)+f(X_n)}{2}+\sum_{i=1}^{n-1}f(X_i)\right)$$

METODO DEI TRAPEZI

```
float func(y) {
   return y*y+7*sin(y); }

float trapezi (float a, float b, int n) {
/* func funzione integranda; a,b estremi intervallo,
   n  numero punti */

   float x,sum, s ,h;
   int j;
   h=(b-a)/n;
   s=0.5*(func(a)+func(b));
   x=a+h;
   for (sum=0.0,j=1; j<n; j++,x+=h)
      sum += func(x);
   s=h*(s+sum);
   return s;
}
```

METODO DI SIMPSON

- Si consideri ***n pari***

  Per ogni i pari: (i=0,2,4,.....,n-2) si consideri la coppia di intervalli:

  $$[X_i, X_{i+1}], [X_{i+1}, X_{i+2}]$$

- si approssima funzione in ogni intervallino con ***parabola passante per i punti*** di coordinate

  $(X_i, f(X_i)), (X_{i+1}, f(X_{i+1})), (X_{i+2}, f(X_{i+2}))$

  cioè $F_i(X) = a_i X^2 + b_i X + c_i$

  a_i, b_i, e c_i si ottengono imponendo

  $F_i(X) = f(X_k)$ per $k = i, i+1, i+2$

- ***Sistema lineare di 3 equazioni in 3 incognite***

METODO DI SIMPSON

$$I_i = \int_{X_i}^{X_{i+2}} (a_i X^2 + b_i X + c_i) dX =$$

$$= \left[\frac{a_i}{3} \cdot X^3 + \frac{b_i}{2} X^2 + c_i X \right]_{X_i}^{X_{i+2}} =$$

$$= c_i (X_{i+2} - X_i) + \frac{b_i}{2} (X_{i+2}^2 - X_i^2) + \frac{a_i}{3} (X_{i+2}^3 + X_i^3)$$

h lunghezza intervallo $[x_{i+1}, x_i]$

Sostituendo i valori di a_i, b_i e c_i ottenuti risolvendo il sistema, si ottiene:

$$I_i = \frac{h}{3} \cdot (f(X_i) + 4 \cdot f(X_{i+1}) + f(X_{i+2}))$$

$$\Rightarrow I = \sum_{i=0}^{n-2} I_i = \frac{h}{3} \cdot (f(X_0) + 4 f(X_1) + f(X_2) +$$

$$+ f(X_2) + 4 \cdot f(X_3) + f(X_4) + \ldots)$$

METODO DI SIMPSON

Generalizzando, si ottiene ***formula di integrazione di Simpson:***

$$I = \frac{h}{3} \cdot \sum_{i=0}^{n} K_i \cdot f(X_i)$$

dove

$$K_i = \begin{cases} 1 & se \quad i = 0, i = n \\ 2 & se \quad i \quad pari \\ 4 & se \quad i \quad dispari \end{cases}$$

METODO DI SIMPSON

```
float func(y) {
   return y+y*y; }

float simpson (float a, float b, int n) {
/* func funzione integranda; a,b estremi intervallo, n
   numero punti */

   float sum2, sum4 ,h, s; int i;
   h=(b-a)/n;
   sum2=0;
   sum4=func(a+h);
   for (i=2;i<n-1;i=i+2) {
      sum2=sum2+func(a+i*h);
      sum4=sum4+func(a+(i+1)*h);
    }
   s=(func(a)+func(b)+4*sum4+2*sum2)*h/3;
   return s;
}
```

INTEGRAZIONE NUMERICA: Esempio di Utilizzo

```
float rettangoli(float a, float b, int n);
float trapezi(float a, float b, int n);
float simpson(float a, float b, int n);

float func(float y) {
   return y*y;}
/* float func(float z) {
   return z*z*z; } */

int main(void) {
   printf("integr. numerica x**2 \n\n");
   printf("%f\n", rettangoli(1, 3, 1000));
   printf("%f\n", trapezi(1, 3, 1000));
   printf("%f\n", simpson(1, 3, 1000));
   }
```

EQUAZIONI DIFFERENZIALI ORDINARIE

- Si consideri l'equazione differenziale di ordine p:

  (1) $F(x,y,y',\ldots,y^{(p)}) = 0$

  con x appartenente ad [a,b] e dove con $y^{(k)}$ si indica

$$y^{(k)} = \frac{dy^k}{dx^k}$$

- Risolvere l'equazione (1) significa ***trovare una funzione y(x) definita e differenziabile p volte in [a,b]***:

  (2) $F(X,y(X),y'(X),\ldots,y^{(p)}(X)) = 0$

- In generale esistono diverse soluzioni di (1): necessario fornire ***vincoli*** (*condizioni al contorno*)

EQUAZIONI DIFFERENZIALI

- Vincoli specificano alcune proprietà (valori funzione e di alcune sue derivate in determinati punti):
  - se p è l'ordine dell'equazione differenziale, sono necessarie ***p condizioni al contorno***

Ad esempio: condizioni iniziali

$$y(X_0) = c_0$$
$$y'(X_0) = c_1$$
$$\ldots\ldots\ldots$$
$$y^{(p-1)}(X_0) = c_{p-1}$$

dove $c_0,..c_{p-1}$ rappresentano costanti date e $a<=X_0<=b$

- Il problema di determinare y(X) in modo da soddisfare (2) e condizioni iniziali viene detto ***problema ai valori iniziali***
- Se i punti in cui vengono dati i valori sono distinti, allora si parla di ***problema ai valori al contorno***

EQUAZIONI DIFFERENZIALI

- Se nell'equazione differenziale di ordine p è possibile esplicitare ***derivata p-esima***:

$$y^{(p)}(X) = f(X, y(X), y'(X), \ldots, y^{(p-1)}(X))$$

- L'equazione di ordine p può essere rappresentata da un ***sistema di p equazioni differenziali del primo ordine***, introducendo (p-1) variabili ausiliarie

Infatti, ad esempio, data l'equazione differenziale del secondo ordine:

$y''(x)=f(x,y,y')$ con $y(X_0)=c_0$ e $y'(X_0)=c_1$

si può porre $y'(x)=z(x)$, ottenendo

$y'(x)=z(x)$

$z'(x)=f(x,y,z)$ con $y(X_0)=c_0$ e $z(X_0)=c_1$

- Perciò si può limitare la trattazione ai ***metodi di risoluzione di (sistemi di) equazioni differenziali del primo ordine***

SOLUZIONE DI EQUAZIONI DIFFERENZIALI
Approccio generale

- Per approssimare la soluzione si utilizzano metodi basati sul ***principio della discretizzazione*:**
  - si rappresenta l'intervallo continuo [a,b] come un insieme discreto di punti (X_0, X_1, X_2,..X_n)
- **Si *approssima* la soluzione y(X) nell'insieme discreto di punti (X_0, X_1,...,X_n):**
  - **in ogni punto X_i la soluzione esatta $y(X_i)$ viene approssimata da un valore Y_i**
- Errore locale dovuto all'approssimazione:

$$\varepsilon_i = |y(X_i) - Y_i|$$

- Ogni metodo fornisce come risultato un insieme discreto di valori Y_0, Y_1, .. Y_n

EQUAZIONI DIFFERENZIALI PRIMO ORDINE
Approccio generale

Consideriamo quindi equazioni del tipo ***y'(x)=f(x,y)***

con $y(X_0)=Y_0$

- si parte dalla coppia di valori noti (X_0,Y_0) per calcolare il valore approssimato di y(x) nel punto X_1 immediatamente vicino a X_0
- una volta noto X_1, con lo stesso procedimento si ricava X_2, e così via fino a X_n

❑ ***Metodi Numerici***

- **a un passo:** Y_i si ricava conoscendo soltanto Y_{i-1}
- **a più passi (o multistep):** Y_i si ricava conoscendo valore di più Y_{i-k} (*con k variabile da metodo a metodo*): è quindi necessario tenere traccia di valori calcolati in precedenza

❑ Vedremo solo ***metodi a un passo*** in quanto più efficienti

❑ Ci baseremo ***sull'uso diretto o indiretto dello sviluppo in serie di Taylor*** della funzione y(X)

METODO DI EULERO

- ***Metodo più semplice, ma meno preciso***

Data **$y'(x)=f(x,y)$** con **$y(X_0)=Y_0$**

Ricordando definizione di derivata di una funzione y(x):

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$$

Supponendo Δx sufficientemente piccolo:

$$\frac{dy}{dx} \approx \frac{\Delta y}{\Delta x}$$

allora l'equazione differenziale può essere scritta come:

$$\Delta y = f(x,y)\Delta x$$

METODO DI EULERO

- si fissa arbitrariamente ***valore Δx sufficientemente piccolo***

- partendo dal punto noto (X_0, Y_0) si ricavano X_1, Y_1 applicando le formule:

$$X_1 = X_0 + \Delta x$$

$$Y_1 - Y_0 = f(X_0, Y_0) * \Delta x$$

$$\Rightarrow \quad Y_1 = Y_0 + f(X_0, Y_0) * \Delta x$$

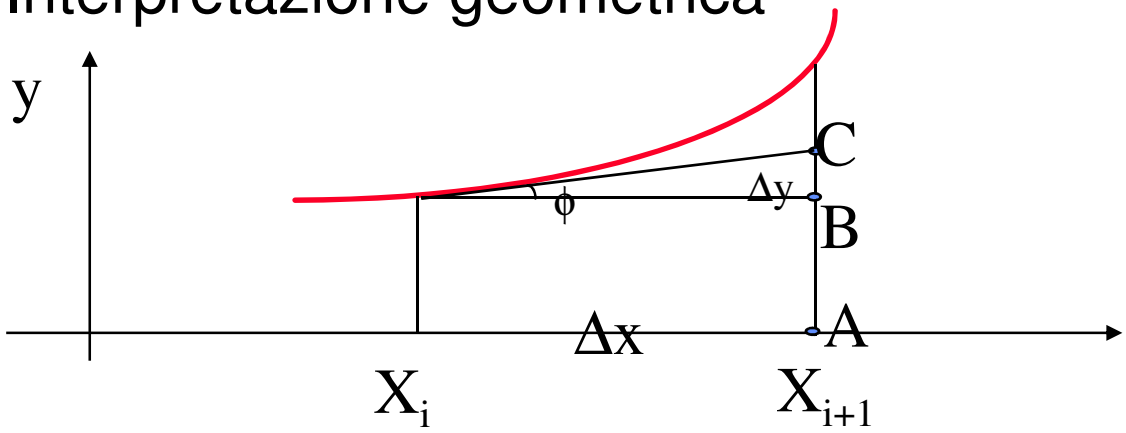
- generalizzando, al passo i-simo:

$$\mathbf{X_i = X_{i-1} + \Delta x}$$

$$\mathbf{Y_i = Y_{i-1} + f(X_{i-1}, Y_{i-1}) * \Delta x}$$

METODO DI EULERO

- Interpretazione geometrica



$AC = AB + BC = y_i + tg\phi\ \Delta X$

$tg\phi = (dY/dX)$ per $X = X_i$, $= f(X_i, Y_i)$

$Y_{i+1} = Y_i + f(X_i, Y_i)\ \Delta X$

- Stessa formula di Eulero si potrebbe ricavare ***arrestando sviluppo in serie di Taylor*** di y(X) nell'intorno del punto Xi al ***termine di primo grado***

$Y(X_i+\Delta X_i) = Y(X_i) + Y'(X_i)\ \Delta X_i + \ldots$

METODO DI EULERO

```
void Eulero (float a, float b, int n, float Y0, float *Y) {
/* a,b estremi intervallo; n numero punti,
   Y0=Y(a), Y vettore soluzioni*/

   float h, h2, X; int i;

   h=(b-a)/n;
   X=a;
   Y[0]=Y0;
   for (i=1; i<=n; i++) {
      Y[i]=Y[i-1]+ h * func(
                              X,
                              Y[i-1] + func(X,Y[i-1])*h
      );
       X=X+h;
   }
   return;
}
```

METODO DI EULERO: Utilizzo

- Si risolva equazione differenziale: $y'(x)=x^2$

```
#define N 200
#include <math.h>
#include <stdio.h>
void Eulero (float a, float b, int n, float Y0, float *Y);

float func(float x, float y) {return x*x;}

int main(void) {
float Y[N],A,B,YA,d,x; int i,n=10;
printf("Estremi intervallo (A,B) ?");
scanf("%f%f", &A,&B);
printf("Condizione al contorno: Y(A)? ");
scanf("%f", &YA);
Eulero(A,B,n,YA,Y);
printf("risultati: \n");
d=(A+B)/n;
for (x=A, i=0; i<n; i++, x+=d)
   printf("X[%d]=%f\tY[%d]=%f\n",i,x,i,Y[i]); }
```

METODI DI RUNGE-KUTTA

- Si applicano alla ***soluzione numerica*** di equazioni differenziali e ***non richiedono l'uso di derivate***

Data $Y' = f(X,Y)$ con $Y(X_0)=Y_0$

- Sostituiscono uno sviluppo in serie di Taylor della forma

$$(*)\ Y_{n+1} = Y_n + hY'_n + Y''_n h^2/2 + Y'''_n h^3/3 + \ldots$$

con ***un'espressione approssimata*** in cui Y_{n+1} viene calcolato come:

$$(**)\ \mathbf{Y_{n+1} = Y_n + \delta}$$

dove ***h*** (vedi seguito) rappresenta ***l'incremento della variabile X*** e δ ***è una stima dell'incremento di Y*** nell'intervallo $[X_n, X_{n+1}]$

METODI DI RUNGE-KUTTA

- δ viene calcolato come media pesata degli incrementi di Y calcolati in vari punti dell'intervallo $[X_n, X_{n+1}]$

$$\delta = h\,(\,a_0 f(X_n,Y_n) + a_1 f(X_n + c_1 h, Y_n + b_1 h) + a_2 f(X_n + c_2 h, Y_n + b_2 h) + \ldots a_p f(X_n + c_p h, Y_n + b_p h))$$

Quindi
$$Y_{n+1} = Y_n + h\,(\,a_0 f(X_n,Y_n) + a_1 f(X_n + c_1 h, Y_n + b_1 h) + a_2 f(X_n + c_2 h, Y_n + b_2 h) + \ldots\ldots\ldots + a_p f(X_n + c_p h, Y_n + b_p h))$$

- Scriviamo poi (**) nella forma:

$$\mathbf{Y_{n+1} = Y_n + a_0 k_0 + a_1 k_1 + \ldots a_p k_p}$$

dove $k_0 = h\, f(X_n, Y_n)$; … $k_2 = h * f(X_n + c_2 h, Y_n + q_{2,0} k_0 + q_{2,1} k_1)$

..... $k_p = h * f(X_n + c_p h, Y_n + q_{p,0} k_0 + \ldots q_{p,p-1} k_{p-1})$

  - a_i pesi attribuiti ai vari incrementi conteggiati nella media
  - k_i vari incrementi di Y conteggiati nella media

METODI DI RUNGE-KUTTA

- Numero dei valori (k_i) utilizzati nella media indica ***l'ordine del metodo di soluzione***

- Restano da ***determinare i coefficienti*** a_i oppure $c_i, q_{i,j}$

  - Vediamo il caso **p=3** (*metodo standard di Runge-Kutta*: ordine 4); ci limitiamo a fornire le formule finali:

$$Y_{n+1} = Y_n + (k_0 + 2k_1 + 2k_2 + k_3)/6$$
$$X_{n+1} = X_n + h$$

dove $k_0 = h\, f(X_n, Y_n)$

$k_1 = h\, f(X_n + h/2,\ Y_n + k_0/2)$

$k_2 = h\, f(X_n + h/2,\ Y_n + k_1/2)$

$k_3 = h\, f(X_n + h,\ Y_n + k_2)$

METODI DI RUNGE-KUTTA

Alcuni casi particolari

- ***Metodo di Heun***

$$Y_{n+1} = Y_n + (k_0+k_1)h/2$$
$$X_{n+1} = X_n + h$$

dove $k_0 = f(X_n, Y_n)$ $\qquad$ $k_1 = f(X_n + h, Y_n + hk_0)$

- ***Metodo di Eulero Modificato*** (del punto medio)

$$Y_{n+1} = Y_n + hk_1$$
$$X_{n+1} = X_n + h$$

dove $k_0 = f(X_n, Y_n)$ $\qquad$ $k_1 = f(X_n+h/2,\ Y_n+k_0h/2)$

METODO DI EULERO MODIFICATO

```
void Eulerm (float a, float b, int n, float Y0, float *Y) {
   float h, h2, X;
   int i;

   h=(b-a)/n;
   h2=h/2;
   X=a;
   Y[0]=Y0;

   for (i=1;i<=n;i++) {
      Y[i]=Y[i-1]+ h*func( X+h2, Y[i-1]+func(X,Y[i-1])*h2 );
      X=X+h;
   }
return;
}
```

METODO DI EULERO MODIFICATO: Utilizzo

- Si risolva equazione differenziale: $y'(x)=5x^3$

```
#define N 200
#include <math.h>
#include <stdio.h>
void Eulerm (float a, float b, int n, float Y0, float *Y);
float func(float x, float y) {return 5*x*x*x;}

int main(void) {
   float Y[N],A,B,YA,d,x; int i,n=10;
   printf("Estremi intervallo (A,B) ?");
   scanf("%f%f", &A,&B);
   printf("Condizione al contorno: Y(A)? ");
   scanf("%f", &YA);
   Eulerm(A,B,n,YA,Y);
   printf("risultati: \n");
   d=(A+B)/n;
   for (x=A, i=0; i<n; i++, x+=d)
        printf("X[%d]=%f\tY[%d]=%f\n", i, x, i, Y[i]); }
```

CONFRONTI

- Metodo di Eulero modificato è in genere più preciso di quello di Eulero

Esercizio

Trovare la soluzione per l'equazione differenziale:

$$\begin{cases} dY/dX = 0.5(X + Y) - 1 \\ y(0) = 1 \end{cases}$$

che ha come soluzione analitica $y = e^{X/2} - X$

utilizzando ***sia il metodo di Eulero che quello di Eulero modificato***

ALTRO ESEMPIO

- Si risolva numericamente l'equazione differenziale:

  $y' = x + y$

  $x0 = 0$ e $y0=1$

Si assuma $\Delta X = 0.05$ e si applichi il metodo di Eulero:

$$y_{i+1} = y_i + f(x_i,y_i)\,\Delta x_i$$

$X1 = 0.05$; $Y1 = 1+(0+1)0.05 = 1.05$

$X2 = 0.1$; $Y2 = 1.05+(0.05 + 1.05)0.05 = 1.105$

$X3 = 0.15$; $Y3 = 1.105+(0.1+1.105)0.05 = 1.16525$

$X4 = 0.2$; $Y4 = 1.16525+(0.15+1.16525)0.05 = 1.23101$

- La soluzione analitica è

  $y(x) = 2e^X - X - 1$ $\quad (dy/dx = 2e^X - 1)$

  da cui $X1=0.05$; $y(X1)=1.05254$; $X2=0.1$; $y(X2)=1.11034$

  $X3=0.15$; $y(X3)=1.17367$; $X4=0.2$; $y(X4)=1.24280$

ALTRO ESEMPIO

- Applicando il metodo di ***Runge-Kutta a quattro passi*** e assumendo h=0.1 per il *primo intervallo di integrazione* si ha:

k0 = 0.1(1) = 0.1 k1 = 0.1(0.05 + 1.05) = 0.11

k2 = 0.1(0.05+1.055) = 0.1105 k3 = 0.1(0.1+1.1105) = 0.12105

da cui X1= 0.1; Y1= 1+1/6(0.1+0.22+0.221+0.12105)=1.11034

  - Per il *secondo intervallo di integrazione* si ha:

k0 = 0.1(0.1+1.11034) = 0.12103

k1 = 0.1(0.1+0.05+1.11034+0.06051) = 0.13208

k2 = 0.1(0.1+0.05+1.11034+0.06604) = 0.13263

k3 = 0.1(0.1+0.1+1.11034+0.13263) = 0.14429

da cui X2 = 0.2;

Y2 =1.1103 + (0.12103+0.26416+0.26526+0.14429) = 1.24280 .....

ALTRO ESEMPIO

Pertanto gli errori sono

- Per il metodo di Eulero:

  EE(0.05) = 0.00254

  EE(0.1) = 0.00534

  EE(0.15) = 0.00842

  EE(0.2) = 0.01179

- Per Runge-Kutta

  ERK (0.1) = 0

  ERK (0.2) = 0

SISTEMI EQUAZIONI DIFFERENZIALI

Sia dato il sistema di ***k equazioni differenziali del primo ordine*** nelle variabili dipendenti $y_1, y_2, \ldots y_k$

$$\begin{cases} dy_1/dx = f_1(x, y_1, y_2, \ldots y_k) \\ dy_2/dx = f_2(x, y_1, y_2, \ldots y_k) \\ \ldots\ldots\ldots\ldots \\ dy_k/dx = f_n(x, y_1, y_2, \ldots y_k) \end{cases}$$

con condizioni iniziali nel punto x_0

$$\begin{cases} y_1(x_0) = a_1 \\ y_2(x_0) = a_2 \\ \ldots\ldots\ldots \\ y_k(x_0) = a_k \end{cases}$$

SISTEMI EQUAZIONI DIFFERENZIALI: APPLICAZIONE METODI ITERATIVI

- Metodo di Eulero:

$y_{j,i+1} = y_{j,i} + f_j(x_i, y_{1,i}, y_{2,i} \ldots . y_{k,i}) * \Delta x$

$xi+1 = xi + \Delta xi$

$j=1,2,...k$

$i=0,1,...,n-1$

- Metodo di Runge-Kutta

$y_{j,i+1} = y_{j,i} + (k_{j,0} + 2k_{j,1} + 2k_{j,2} + k_{j,3})/6$

$x_{i+1} = x_i + \Delta x$

$j=1,2,...k$

dove

$k_{j,0} = hf_j(X_i, Y_{1,i}, Y_{2,i}, \ldots . Y_{k,i})$

$k_{j,1} = hf_j(X_{i+h}/2,\ Y_{1,i} + k_{1,0}/2, Y_{2,i} + k_{2,0}/2, \ldots Y_{k,i} + k_{k,0}/2)$

$k_{j,2} = hf_j(X_{i+h}/2, Y_{1,i} + k_{1,1}/2, Y_{2,i} + k_{2,1}/2, \ldots Y_{k,i} + k_{k,1}/2)$

$k_{j,3} = hf_j(X_{i+h}, Y_{1,i} + k_{1,2}, Y_{2,i} + k_{2,2}, \ldots Y_{k,i} + k_{k,2})$