Richiami sul metodo di Jacobi

Dobbiamo risolvere il sistema $A\mathbf{x} = \mathbf{b}$.

Lo schema di Jacobi è:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + D^{-1}\mathbf{r}_k$$

D = la matrice diagonale contenente gli elementi diagonali di A

$\mathbf{r}_k$ = il residuo al passo k = $\mathbf{b} - A\mathbf{x}_k$.

Introducendo la matrice di iterazione di Jacobi:

$$E_J = I - D^{-1}A = -D^{-1}(L+U)$$

(L e U sono le matrici triangolare bassa e triangolare alta di A, con elementi della diagonale principale nulli, tali che $L + D + U = A$) lo schema di Jacobi si può anche scrivere come

$$\mathbf{x}_{k+1} = -D^{-1}(L+U)\mathbf{x}_k + D^{-1}\mathbf{b}$$

o, equivalentemente,

$$\mathbf{x}_{k+1} = D^{-1}[-(L+U)\mathbf{x}_k + \mathbf{b}]$$

In forma estesa, si ha:

per $i = 1, n$

$$x_{k+1}^{(i)} = \overset{D^{-1}}{\underset{\Uparrow}{\frac{1}{a_{ii}}}} \left[b_i - \overset{(L+U)\mathbf{x}_k}{\overset{\Uparrow}{\sum_{j=1, j\neq i}^{n}}} a_{ij} x_k^{(j)} \right]$$

o, equivalentemente, per $i = 1, n$

$$x_{k+1}^{(i)} = \overset{D^{-1}}{\underset{\Uparrow}{\frac{1}{a_{ii}}}} \left[b_i - \overset{L\mathbf{x}_k}{\overset{\Uparrow}{\sum_{j=1}^{i-1}}} a_{ij} x_k^{(j)} - \overset{U\mathbf{x}_k}{\overset{\Uparrow}{\sum_{j=i+1}^{n}}} a_{ij} x_k^{(j)} \right]$$

In FORTRAN

Se vogliamo tradurre quest'ultimo algoritmo in un programma FORTRAN, dobbiamo calcolare due sommatorie (quelle relative a $L\mathbf{x}_k$ e a $U\mathbf{x}_k$) prima di aggiornare la componente i-sima del vettore che approssima la soluzione del sistema lineare.

Chiamiamo con `xold` il vettore $\mathbf{x}_k$ e con `xnew` il vettore $\mathbf{x}_{k+1}$.

Sia `n` la dimensione del sistema da risolvere, `A` la matrice, `b` il vettore termine noto.

L'implementazione del metodo di Jacobi in FORTRAN può dunque essere scritta come:

```
.....       do i=1,n
.....            som1=0.d0
.....            som2=0.d0
.....            do j=1,i-1
.....                 som1=som1+A(i,j)*xold(j)
.....            end do
.....            do j=i+1,n
.....                 som2=som2+A(i,j)*xold(j)
.....            end do
.....            xnew(i) = (b(i)-som1-som2)/A(i,i)
.....       end do
```

Stima della velocità asintotica di convergenza

Utilizzando la norma euclidea dello scarto al passo k e al passo $k+1$, possiamo calcolare una stima del fattore di convergenza del metodo (vale a dire il modulo dell'autovalore dominante della matrice di iterazione di Jacobi).

Dalla teoria, sappiamo che, per $k \to \infty$, l'errore

$$\mathbf{e}_{k+1} = E\mathbf{e}_k \approx \lambda_1^{k+1} c_1 \mathbf{v}_1 \Rightarrow \lim_{k\to\infty} \frac{\|\mathbf{e}_{k+1}\|}{\|\mathbf{e}_k\|} = |\lambda_1|.$$

Per $k \to \infty$ lo scarto rappresenta una buona approssimazione per l'errore: introducendo lo scarto `sc(i) = xnew(i) - xold(i)`, per `i=1,n` e calcolando la norma euclidea dello scarto a due iterazioni successive, possiamo stimare il fattore di convergenza $M = |\lambda_1|$ come rapporto di queste due norme.

Analogamente (considerando la definizione di velocità asintotica di convergenza $R = -\log_{10}(|\lambda_1|)$), possiamo stimare R applicando $-\log_{10}$ a tale rapporto.

In FORTRAN, dando alle variabili `scarto` e `scarto_old` il significato della norma euclidea dello scarto al passo $k+1$ (nuovo) e al passo k (old), possiamo scrivere, dopo aver valutato `xnew` :

```
do i=1,n
   sc(i)=xnew(i) - xold(i)
end do
scarto=euc(n,sc)  %euc: function su norma euclidea
asint=scarto/scarto_old
R=-log10(asint)
```

Si va avanti nel calcolo di `xnew`, il vettore che approssima la soluzione del sistema lineare, fino a quando la norma euclidea dello scarto non è minore di una prefissata tolleranza `toll` (per esempio `toll = 1.d-10`). Un altro controllo viene fatto anche sul numero massimo di iterazioni (`itmax`) (il procedimento è del tutto analogo a quanto si è già visto nei programmi FORTRAN sul metodo del punto fisso o di Newton-Raphson).

Stampa dei risultati parziali

Durante ciascuna iterazione, vengono calcoltati `xnew`, `scarto`, `asint`, `R`. Ad ogni iterazione ci facciamo stampare `scarto, asint, R`.

A tal fine possiamo pensare di stampare i valori secondo un certo formato, per esempio con un formato numerico con 8 cifre decimali dopo la virgola, o con un formato scientifico con 8 cifre nella mantissa...

Vediamo un esempio:

```
......write(16,100) iter, scarto, asint, R
.100..format(i6, e20.8, 2f14.8)
```

Nella istruzione `write` compaiono due numeri, 16 e 100: 16 dice su quale file vanno scritti i risultati 100 invia alla ”label” 100 dove viene specificato il formato da usare.
Nella riga successiva, è scritto 100 a partire dalla seconda colonna, e poi, a partire dalla settima colonna c'e scritto `format` e, tra parentesi, sono date indicazioni sul formato da seguire nella stampa delle corrispondenti variabili che sono sull'istruzionde di `write`:

```
......write(16,100) iter, scarto, asint, R
.100..format(i6, e20.8, 2f14.8)
```

`i6` è associato alla prima variabile, cioè `iter`;
`e20.8` è associato alla seconda variabile, `scarto`;
`2f14.8` è associato alla terza e quarta variabile dell'elenco (il 2 davanti alla `f` indica 2 variabili con lo stesso formato numerico).

- `i6`: sono assegnati 6 caratteri per la variabile intera `iter`. Quindi il valore di `iter` sarà scritto utilizzando 6 caratteri. Se il numero da scrivere avesse più di 6 caratteri (per esempio `iter` = 1000001) la stampa sarebbe con degli ******.
- `e20.8`: il valore della variabile reale `scarto` è stampato secondo un formato esponenziale (perciò c'è la `e`) con 20 caratteri, di cui 8 per la mantissa. Perchè 20 caratteri? C'è da tenere presente che un carattere va al segno + o -, un carattere allo 0, un carattere per il . e poi 4 o 5 caratteri vanno all'esponente (per esempio E-01 o E-001, a seconda del compilatore): quindi 7 o 8 caratteri sono riservati a questo scopo. Il numero totale di caratteri (in questo caso 20), deve dunque essere maggiore o uguale della somma di questi 7/8 caratteri più le cifre da destinare alla mantissa, altrimenti sono stampati degli **********.

In questo caso dobbiamo almeno scrivere `e16.8` per essere sicuri di non avere **********. Se mettiamo un numero più grande (20) è perchè così vengono lasciati più spazi vuoti a quella variabile).

- `2f14.8`: potremmo scrivere anche `f14.8, f14.8` per dire che le due variabili `asint` e `R` hanno lo stesso formato. In forma compatta, scriviamo `2f14.8` perche' sono 2 variabili con lo stesso formato. In questo caso il formato è fisso, per stampare numeri reali usando virgola fissa. Abbiamo 14 caratteri in virgola fissa e 8 caratteri sono per le cifre decimali. Se il valore da stampare è troppo grande o troppo piccolo non lo si riesce a stampare.

Cenni sui vari tipi di formato

`iN`	stampa di numeri interi usando `N` caratteri
`eM.N`	stampa di numeri reali in formato esponenziale usando `M` caratteri, di cui `N` per la mantissa
`fM.N`	stampa di numeri reali in formato fisso `M` caratteri sono in virgola fissa `N` caratteri sono per le cifre decimali
`aN`	formato alfanumerico per stringhe di `N` caratteri
`Nx`	stampa `N` caratteri bianchi (vuoti)
`/`	serve per andare a capo nella stampa

Esempi

```
      write(16,99) 'iterazione', iter
99    format(1x,a10,3x,i4)

      write(16,100) iter,scarto,asint,r
100   format(i6,e20.8,2f14.8)

      write(16,101) iter, scarto, asint, R
101   format(1x, i4, 2x, e16.8, 2x, 2f14.8)
```

in output ⇓

```
iterazione      7

  21       0.53629231E-02     0.80901699     0.09204236

 21     0.53629231E-02       0.80901699     0.09204236
```