Laboratorio di Calcolo Numerico
Lezione 3

1 Disegnare un cerchio

Il seguente comando disegna un cerchio.

```
>> t=0:.001:2*pi;
>> plot(cos(t),sin(t))
```

Suggerimento: Il comando `t = 0:.001:2*pi` può essere sostituito da un comando `linspace`. Provate ad utilizzarlo nei prossimi esercizi.

Esercizio 1. Scrivere una funzione `circle(z,r)` che, dato un complesso `z` e un reale $\texttt{r} \geq 0$, disegni il cerchio di centro `(real(z),imag(z))` e raggio `r`. Testare con `circle(1+2*i,2)`.

Il disegno del cerchio può essere un poco più elaborato scegliendo un colore per il bordo e per la parte interna. A tal proposito si può utilizzare la funzione `patch` (per vedere come funziona digitare `help patch`).

2 Cerchi di Gerschgorin

La seguente funzione disegna gli autovalori e i cerchi di Gerschgorin di una matrice quadrata `A`.

```
function disegna_gersh(A)
n=size(A, 1); %estrae il numero di righe di A

close all %elimina i disegni preesistenti
hold on %fa si' che ogni disegno non cancelli il precedente
axis('equal') %forza la stessa scala su x e y

autovalori=eig(A);
plot(real(autovalori),imag(autovalori),'*');
% '*': disegna solo i punti, non collegandoli con linee
%'help plot' per altre stringhe magiche
for k=1:n
```

```
        center=A(k,k);
        radius=0; %accumulatore
        for j=1:n
        if(j~=k)
                radius=radius+abs(A(k,j));
        end
end
circle(center,radius);
end
end
```

Esercizio 2. Modificare la funzioni `disegna_gersh` e `circle` in modo che disegnino cerchi di un colore assegnato dall'utente e si testino su alcune matrici, ad esempio: `rand(5)`, `randn(5)`, `hilb(5)`.

Nel caso si sia utilizzato un colore per la parte interna dei cerchi, si può utilizzare il comando `alpha(x)` per regolarne l'intensità; ad esempio si può dare il comando:

```
alpha(.15)
```

quando tutti i cerchi sono stati aggiunti alla figura.

Esercizio 3. Si provi a riscrivere `disegna_gersh` con un unico ciclo for. Infatti, poiché MATLAB è un linguaggio *interpretato*, eseguire ogni singola istruzione ha un "costo" non trascurabile (il computer deve leggere la riga, interpretarla e trasformarla in codice macchina). Per questo se si riesce a riscrivere le funzioni utilizzando delle operazioni sui vettori invece che dei cicli `for`, il programma gira molto più velocemente. Per esempio, è molto più veloce

```
s=sum(abs(v));
```

(una istruzione da interpretare) rispetto a

```
s=0;
for k=1:length(v)
s=s+abs(v(k));
end
```

($O(n)$ istruzioni da interpretare, dove n è la lunghezza del vettore `v`). Negli anni recenti, MATLAB ha incorporato un compilatore JIT che ha permesso di accelerare i for più semplici come quello riportato sopra. Tuttavia, questa osservazione sarà importante nel caso di for annidati per il calcolo delle fattorizzazioni matriciali, che vedremo nelle prossime lezioni.

Esercizio 4. Si scriva una funzione `disegna_gersh2` che data una matrice A disegni in una figura: i suoi autovalori, i cerchi di gershgorin della matrice A **in blu** ed **in giallo** quelli associati alla matrice A^T; in questo modo si dovrebbe poter visualizzare l'insieme

$$\left(\bigcup_{i=1}^{n} F_i(A)\right) \cap \left(\bigcup_{i=1}^{n} F_i(A^T)\right)$$

come l'intersezione delle due regioni colorate.