Università di Pisa

Facoltà di Ingegneria

Esercizi svolti per il corso di "Elementi di Informatica Applicata"

(DRAFT)

Nicola Forgione

Pisa, 2011

Prefazione

Questo testo raccoglie alcuni degli esercizi mostrati in classe come esempi applicativi o risolti in sede d'esame durante il corso di "Elementi di Informatica Applicata" della Facoltà di Ingegneria, tenuto dal sottoscritto a partire dal 2005. Tali esercizi rappresentano tipici problemi ingegneristici e sono stati risolti principalmente mediante l'uso dell'ambiente di programmazione MATLAB. Per la risoluzione di alcuni esercizi del primo capitolo sono stati utilizzati anche i linguaggi FORTRAN 90 e C++ al fine di mostrarne le differenti modalità di programmazione rispetto al MATLAB.

La prima parte del testo tratta la risoluzione di problemi relativamente semplici in modo da prendere confidenza con i principali comandi MATLAB riguardanti gli arrays, le istruzioni cicliche e quelle condizionali. In pratica, in questa parte del libro si pongono le basi per l'uso del MATLAB. Vengono presentati, tra l'altro, esempi di risoluzione di sistemi di equazioni lineari e di equazioni non lineari con l'uso del metodo di bisezione, delle sostituzioni successive, di Newton-Raphson, ecc..

Nella seconda parte vengono risolti problemi più complessi mediante l'uso dei file di funzione. Vengono utilizzati comandi di lettura e scrittura di dati su file e di creazione di grafici bi e tri-dimensionali. In tale parte viene posta particolare attenzione ai metodi numerici di regressione, di interpolazione e di integrazione.

La terza parte del volume è dedicata alla risoluzione di problemi ingegneristici coinvolgenti la risoluzione numerica di sistemi di equazioni differenziali ordinarie (Ordinary Differential Equations) ai valori iniziali (Initial Value Problems) ed al contorno (Boundary Value Problems). Laddove possibile, la soluzione numerica viene confrontata con la soluzione analitica esatta o approssimata.

Nell'ultima parte del testo vengono riportati alcuni esempi di applicazioni MATLAB utili in ambito ingegneristico, ma che non rientrano nel programma del corso di "Elementi di Informatica Applicata".

1. VETTORI, MATRICI, CICLI E ISTRUZIONI CONDIZIONALI

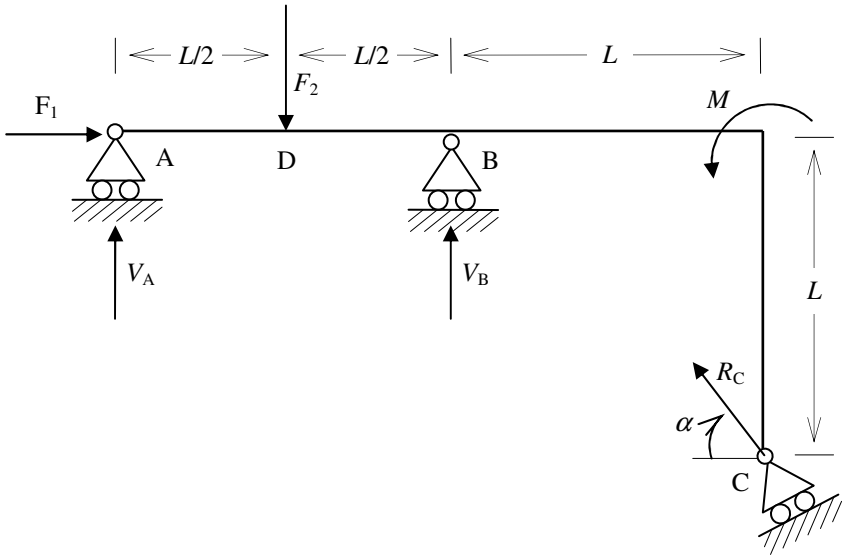
1.1 Risoluzione di un sistema di equazioni Ax=b con A e b noti (Scienza delle costruzioni)

La struttura mostrata in figura è sollecitata da una forza concentrata F_1 orizzontale ed applicata in A, da una forza concentrata F_2 verticale ed applicata in D e da una coppia M applicata in corrispondenza della giunzione tra il tratto verticale e quello orizzontale del portale zoppo.

Si conoscono: F_2=3000 N, M=2000 Nm e L=2 m.

Determinare le reazioni vincolari V_A, V_B e R_C al variare dell'angolo α tra zero e 90° e al variare dell'intensità della forza $F_1 \geq 0$.

Realizzare un programma MATLAB, uno in FORTRAN ed uno in C++ che, assegnate le grandezze note, legga da tastiera l'angolo α ed il valore della forza F_1 e, nel caso di problema staticamente determinato, restituisca a monitor i valori delle reazioni vincolari.



Soluzione

Assunti come versi positivi delle reazioni vincolari incognite quelli mostrati in figura, le suddette reazioni vincolari possono essere calcolate, se il sistema risulta staticamente determinato, scrivendo le equazioni di equilibrio alla traslazione lungo l'asse verticale, alla traslazione lungo l'asse orizzontale ed alla rotazione intorno ad un punto (ad es. intorno al punto C).

$$\begin{cases} V_A + V_B + R_C \sin\alpha = F_2 \\ R_C \cos\alpha = F_1 \\ V_A\, 2L + V_B\, L = -F_1\, L + F_2 \dfrac{3}{2} L + M \end{cases}$$

Ne risulta un sistema di tre equazioni algebriche lineari nelle tre incognite V_A, V_B e R_C. Indicando con A la matrice incompleta del sistema e con A^* la matrice completa, si possono avere i seguenti casi:

1. per $0 \leq \alpha < 90°$ sia il rango della matrice incompleta che quello della matrice completa è uguale a 3 qualsiasi sia il valore della forza F_1, per cui il problema risulta staticamente determinato;
2. per $\alpha = 90°$ il rango della matrice incompleta è pari a 2, mentre quello della matrice completa è uguale a 2 se la forza F_1 è nulla (problema staticamente indeterminato) oppure è uguale a 3 se la forza F_1 è non nulla (problema staticamente impossibile).

Nel seguito sono riportati i file MATLAB, C++ e FORTRAN 90 messi a punto per la risoluzione numerica del problema proposto. Come si può notare il file MATLAB risulta essere più breve di quelli C++ e FORTRAN in quanto in MATLAB esistono delle funzioni intrinseche che nei compilatori FORTRAN e C++ non sono presenti nella loro versione base, seppure possono essere aggiunte tramite apposite librerie.

```
% Problema di Scienza delle Costruzioni (in MATLAB)
clc, clear, close all

% Dati di input
F2 = 3000;  % Carico concentrato [N]
L = 2;      % Lunghezza [m]
M = 2000;   % Coppia in [N*m]
alfa = input('Inserire l''angolo alfa in gradi: \n');
alfa = alfa * pi / 180;
F1 = input('Inserire la forza F1 in N: \n');

% Assegnazione della matrice dei coefficienti A
% e del vettore dei termini noti b
A = [1, 1, sin(alfa); 0 0 cos(alfa); 2, 1, 0];
b = [F2; F1; (-F1+1.5*F2+M/L)];
disp(' ');

if rank(A)==rank([A b])
   if rank(A)==3
      disp('Esiste una soluzione unica che vale:');
      x=A\b;
      fprintf('\nVa = %10.2f N\n',x(1))
      fprintf('Vb = %10.2f N\n',x(2))
      fprintf('Rc = %10.2f N\n',x(3))
   else
      disp('Problema staticamente indeterminato');
   end
else
   disp('Problema staticamente impossibile');
end
```

```
------------------------------------------------------------


// Problema di Scienza delle Costruzioni (in C++)
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>

const int MAX_DIM = 10;
int skip_col[MAX_DIM];

int pivot(int dim, int j, double matr[MAX_DIM][MAX_DIM], double tn[MAX_DIM]);
int gauss(double coef_mat[MAX_DIM][MAX_DIM], double term_noti[MAX_DIM],double
sol[MAX_DIM], int dim);
double DET(double A[][MAX_DIM],int row,int SIZE);

int main(void)
{
   double A[MAX_DIM][MAX_DIM],b[MAX_DIM],x[MAX_DIM];
   double alfa,F1,d;
   // Dati di input
   double pi=3.1415926535898;  // pi greco
   double F2=3000;             // Carico concentrato [N]
   double L=2;                 // Lunghezza [m]
   double M=2000;              // Coppia in [N*m]
   int dim,success;
   printf("Inserire l''angolo alfa in gradi: \n");
   scanf("%lf",&alfa);
   alfa=alfa*pi/180;
   printf("Inserire la forza F1 in N: \n");
   scanf("%lf",&F1);

   /* Assegnazione della matrice dei coefficienti A
   e del vettore dei termini noti b */
   A[0][0]=1;A[0][1]=1;A[0][2]=sin(alfa);
   A[1][0]=0;A[1][1]=0;A[1][2]=cos(alfa);
   A[2][0]=2;A[2][1]=1;A[2][2]=0;
   b[0]=F2;b[1]=F1;b[2]=-F1+1.5*F2+M/L;

   dim = 3;
   d = DET(A,dim,dim); // Determinante della matrice dei coefficienti A
   //   printf("Detrminante(A): %0.14lf \n",d);

   if (( d > -1e-13)&&(d < 1.e-13 ))
   {
      printf("Il sistema e' indeterminato o impossibile\n\n");
   }
   else
   {
      success=gauss(A,b,x,dim);// Soluzione del sistema con Gauss
      printf("Esiste una soluzione unica che vale: \n");
      printf("Va=  %lf\n",x[0]);
      printf("Vb=  %lf\n",x[1]);
      printf("Rc=  %lf\n\n",x[2]);
   }
   system("PAUSE");
   return EXIT_SUCCESS;
}

int gauss( double coef_mat[MAX_DIM][MAX_DIM], double term_noti[MAX_DIM],
double sol[MAX_DIM], int dim)
{
   int i, j, k, // indici di scansione della matrice
```

```
   check = 0;
   double piv, alfa, temp;
   for( j=0; j<dim-1; j++)
   {
      // scegli pivot;
      check = pivot(dim, j, coef_mat, term_noti);
      if( check == 0 )
         piv=coef_mat[j][j];
      else
        return(-1);
      // combinazione lineare
      for( i=j+1;i<dim; i++)
      {
         // calcolo del coefficente della combinazione lineare
         alfa = coef_mat[i][j]/piv;
         // sottrazione
         for( k=j; k<dim; k++)
         coef_mat[i][k]=coef_mat[i][k]-alfa*coef_mat[j][k];
         // termini noti
         term_noti[i]=term_noti[i]-alfa*term_noti[j];
      } // for su righe i da j+1 a n
   }    //for sui primi n-1 elemnti della diagonale a[j][j]
   // Check su determinante
   if( coef_mat[dim-1][dim-1] == 0)
   {
      return(1);
   }
   // Soluzione
   sol[dim]=term_noti[dim]/coef_mat[dim][dim];
   for(i=dim-1; i>=0; i--)
   {
      temp = term_noti[i];
      for( k=i+1; k<dim; k++)
         temp=temp-coef_mat[i][k]*sol[k];
      sol[i] = temp/coef_mat[i][i];
   }
   return(0);
}// gauss()


// Funzione di pivoting parziale;
// cerca il pivot per la colonna j-esima
// e se diverso da a[j][j] esegue lo scambio
int pivot( int dim, int j, double matr[MAX_DIM][MAX_DIM], double tn[MAX_DIM])
{
   int k,r;
   double temp;
   char cont;
   // ricerca del primo elemento non nullo
   k=j;
   while( k<dim && matr[k][j]==0 ) ++k;
   if( k<dim ) // esiste k tc matr[k,j]==0
      printf("\n");
   else
   {
      printf("\n");
      return(-1);
   }
   if( k!=j )
   {
      // scambio righe
      for( r=j; r<dim; r++ )
      {
         temp = matr[j][r];
```

```
         matr[j][r] = matr[k][r];
         matr[k][r] = temp;
      }
      temp = tn[j];
      tn[j] = tn[k];
      tn[k] = temp;
   }
   return(0);
} // pivot()

// Funzione ricorsiva per il calcolo del
// determinante di una matrice
double DET(double A[][MAX_DIM],int row,int SIZE)
{
   int col,i,test;
   double det=0;
   int minor_col=0;
   if(row>1)
   for(col=0;col<SIZE;col++)
   {
      if(col==0&&row==SIZE)
      {
         for(i=0;i<100;i++) skip_col[i]=100;
      }
      test=0;
      for(i=SIZE-1;i>1;i--)
      if(col==skip_col[i])test++;
      if(test!=0)continue;
      skip_col[row-1]=col;
      det+=pow(-1,(row-1)+minor_col)*A[row-1][col]*DET(A,row-1,SIZE);
      minor_col++;
      skip_col[row-1]=100;
   }
   if(row==1)
   {
      for(col=0;col<SIZE;col++)
      {
         test=0;
         for(i=SIZE-1;i>=1;i--)
         if(col==skip_col[i])test++;
         if(test==0)break;
       }
       return(A[row-1][col]);
   }
   return(det);
}
```

```
PROGRAM portalez
  ! Problema di Scienza delle Costruzioni (in FORTRAN 90)

  ! Sezione dichiarativa
  IMPLICIT NONE
  REAL(KIND(0.d0)), PARAMETER      :: pi=3.141593d00
  INTEGER, PARAMETER               :: n=3
  REAL(KIND(0.d0))                 :: F1,F2,M,L,alfa,d,det
  REAL(KIND(0.d0)), DIMENSION(3,3) :: A
  REAL(KIND(0.d0)), DIMENSION(3)   :: x,b

  ! Dati di input
  F2=3000         ! Carico concentrato [N]
```

```
  L=2             ! Lunghezza [m]
  M=2000          ! Coppia in [N*m]
  WRITE(*,*) 'Inserire l''angolo alfa in gradi:'
  READ(*,*) alfa
  alfa = alfa * pi / 180
  WRITE(*,*) 'Inserire la forza F1 in N:'
  READ(*,*) F1

  ! Calcolo dei dati di output
  A = &
RESHAPE((/1.d00,1.d00,sin(alfa),0.d00,0.d00,cos(alfa),2.d00,1.d00,0.d00/),(/3,3/))
  A = TRANSPOSE(A)                  !Matrice incompleta
  b = (/F2, F1, (-F1+1.5*F2+M/L)/)  !Vettore dei termini noti
  d = det(A,n)
  IF (dabs(d)<1.d-13) THEN
     WRITE(*,*) "Soluzione indeterminata o impossibile"
  ELSE
     CALL lupt(A,b,n)
     WRITE(*,*)"Soluzione:"
     WRITE(*,*)"Va = ",b(1)
     WRITE(*,*)"Vb = ",b(2)
     WRITE(*,*)"Rc = ",b(3)
  END IF

  PAUSE
END PROGRAM portalez


RECURSIVE FUNCTION det(A,n) RESULT(risultato)
! Funzione che calcola il determinante della matrice A mediante la regola di
Laplace,
! sviluppando il determinante lungo la prima riga

  IMPLICIT NONE
  INTEGER :: n,i
  REAL(KIND(0.d0)), DIMENSION(n,n)     :: A
  REAL(KIND(0.d0)), DIMENSION(n-1,n-1) :: B
  REAL(KIND(0.d0)) :: s,risultato

  IF (n==1) THEN
     s=A(1,1)
  ELSE
     s=0.d0
     DO i=1,n
        B(1:n-1,1:i-1)=A(2:n,1:i-1)
        B(1:n-1,i:n-1)=A(2:n,i+1:n)
        s=s+(-1)**(i+1)*A(1,i)*det(B,n-1)
     END DO
  END IF
  risultato=s

END FUNCTION det

SUBROUTINE lupt(A,b,n)
  ! Calcola la fattorizzazione LU di A con la strategia del pivot totale
  IMPLICIT NONE
  INTEGER  :: n,i,k,j
  INTEGER, DIMENSION(2)            :: piv
  REAL(KIND(0.d0)), DIMENSION(n,n) :: A
  REAL(KIND(0.d0)), DIMENSION(n)   :: b
  REAL(KIND(0.d0)), DIMENSION(n)   :: y
  REAL(KIND(0.d0))                 :: xpiv
  INTEGER, DIMENSION(n) :: r,c ! Vettori per lo scambio delle righe e colonne
  DO i=1,n
```

```
      r(i)=i
      c(i)=i
   END DO
   elimina: DO k=1,n-1
      ! Cerca l'indice di pivot
      ! piv=minval(MAXLOC(ABS(a(r(k:n),c(k:n)))))+k-1
      piv(1)=k; piv(2)=k
      xpiv=ABS(A(r(k),c(k)))
      DO i=k,n
         DO j=k,n
            IF(DABS(A(r(i),c(j)))>xpiv)THEN
               piv(1)=i
               piv(2)=j
               xpiv=DABS(A(r(i),c(j)))
            END IF
         END DO
      END DO
      ! Scambia righe e colonne
      IF(piv(1)/=k)THEN
         i=r(piv(1))
         r(piv(1))=r(k)
         r(k)=i
      END IF
      IF(piv(2)/=k)THEN
         i=c(piv(2))
         c(piv(2))=c(k)
         c(k)=i
      END IF
      ! Aggiorna la matrice L
      A(r(k+1:n),c(k))=A(r(k+1:n),c(k))/A(r(k),c(k))
      ! Aggiorna la matrice U
      DO i=k+1,n
         A(r(i),c(k+1:n))=a(r(i),c(k+1:n))-A(r(i),c(k))*A(r(k),c(k+1:n))
      END DO
   END DO elimina
   ! risoluzione del sistema
   y(1)=b(r(1))
   lower: DO i=2,n
      y(i)=b(r(i))
      DO j=1,i-1
         y(i)=y(i)-A(r(i),c(j))*y(j)
      END DO
   END DO lower
   b(c(n))=y(n)/a(r(n),c(n))

   upper: DO i=n-1,1,-1
      b(c(i))=y(i)
      DO j=i+1,n
         b(c(i))=b(c(i))-A(r(i),c(j))*b(c(j))
      END DO
      b(c(i))=b(c(i))/A(r(i),c(i))
   END DO upper
END SUBROUTINE lupt
```

1.2 Risoluzione di una equazione non lineare (Eq. di Bernoulli generalizzata)

Una tubazione lunga 10 km, avente un diametro interno di 300 mm ed una rugosità superficiale di 0.03 mm, convoglia acqua (ν = 1.13e-6 m^2/s) da un bacino posto a 1100 m s.l.m. ad un piccolo lago artificiale posto a 700 m s.l.m. Considerando trascurabili le perdite di carico localizzate, si determini la portata volumetrica e la velocità media dentro la tubazione utilizzando i seguenti metodi:

1. metodo di bisezione;
2. metodo della falsa posizione;
3. metodo delle approssimazioni successive;
4. metodo delle secanti.

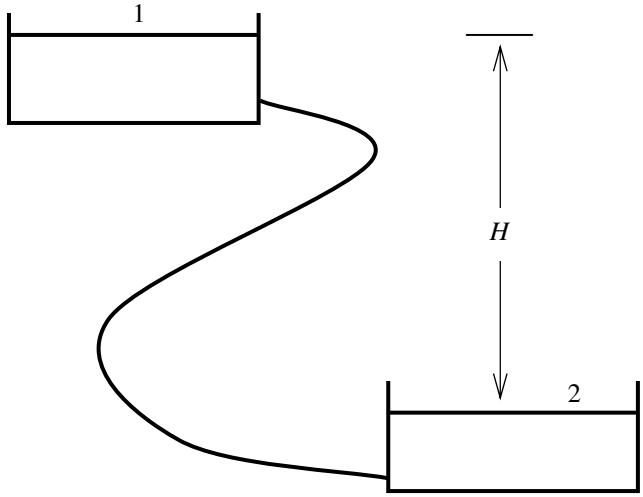
Realizzare dei programmi in MATLAB, in C++ ed in FORTRAN che, assegnate le grandezze note, forniscano a monitor i valori della velocità e della portata volumetrica.

Soluzione

L'applicazione dell'equazione di Bernoulli generalizzata al problema in esame fornisce:

$$\frac{p_2}{\gamma}+\frac{w_2^2}{2g}+z_2=\frac{p_1}{\gamma}+\frac{w_1^2}{2g}+z_1+h'-h_A$$

$$h_A=z_1-z_2 \quad \Rightarrow \mathrm{f_{att}}\frac{L}{D}\frac{w^2}{2g}=H$$



Quindi la funzione da azzerare è:

$$f(w)=\mathrm{f_{att}}\frac{L}{D}\frac{w^2}{2g}-H$$

dove il fattore di attrito, f_{att}, può essere valutato, ad es., mediante la formula di Swamee & Jain:

$$\mathrm{f_{att}}=\frac{0.25}{\left[\log_{10}\left(\frac{\varepsilon}{3.7D}+\frac{5.74}{\mathrm{Re}^{0.9}}\right)\right]^2}$$

Nel seguito sono riportati i file MATLAB, C++ e FORTRAN 90 messi a punto per la risoluzione numerica del problema proposto. Come si può notare i file MATLAB risultano essere più brevi dei corrispondenti file FORTRAN e C++ in quanto in MATLAB esistono delle funzioni intrinseche che nel FORTRAN e nel C++ base non sono presenti.

```
% Problema risolto con il metodo di "bisezione" (in MATLAB)
clc, clear, close all

% Assegnazione dei dati di input
D = 0.3;         % diametro interno del tubo [m]
e = 0.03e-3;     % rugosità del tubo [m]
H = 400.0;       % dislivello tra i due bacini di raccolta [m]
L = 10000.0;     % lunghezza della tubazione [m]
ni = 1.13e-6;    % viscosità cinematica dell'acqua [m2/s]

% Approssimazioni iniziali
wa = 1.0;
Rey = wa*D/ni;
fatt = 0.25/(log10(e/D/3.7+5.74/Rey^0.9))^2; % (Swamee & Jain)
fa = fatt*L/D*wa*wa/2/9.81-H;

wb = 5.0;
Rey = wb*D/ni;
fatt = 0.25/(log10(e/D/3.7+5.74/Rey^0.9))^2;
fb = fatt*L/D*wb*wb/2/9.81-H;

if fa*fb>0
    disp('Le due approssimazioni iniziali non separano lo zero');
    return
end

% Ciclo del procedimento iterativo
for i = 1:100
   wx = (wa + wb) / 2;
   Rey = wx*D/ni;
   fatt = 0.25/(log10(e/D/3.7+5.74/Rey^0.9))^2;
   fx = fatt*L/D*wx*wx/2/9.81-H;
   if fx==0
      wb=wx;
      wa=wx;
   elseif fa*fx>0
      wa=wx;
      fa=fx;
   else
      wb=wx;
      fb=fx;
   end
   error = abs(wb-wa)/wx; % errore relativo
   if error<1e-6, break, end
end
i                % numero di iterazioni necessarie
wx               % velocità media [m/s]
Q=wx*pi*D*D/4    % portata volumetrica [m3/s]
```

```
------------------------------------------------------------
% Problema risolto con il metodo della "falsa posizione" (in MATLAB)
clc, clear, close all

% Assegnazione dei dati di input
D = 0.3;         % diametro interno del tubo [m]
e = 0.03e-3;     % rugosità del tubo [m]
H = 400.0;       % dislivello tra i due bacini di raccolta [m]
L = 10000.0;     % lunghezza della tubazione [m]
ni = 1.13e-6;    % viscosità cinematica dell'acqua [m2/s]

% Approssimazioni iniziali
wa = 1.0;
Rey = wa*D/ni;
fatt = 0.25/(log10(e/D/3.7+5.74/Rey^0.9))^2; % (Swamee & Jain)
fa = fatt*L/D*wa*wa/2/9.81-H;

wb = 5.0;
Rey = wb*D/ni;
fatt = 0.25/(log10(e/D/3.7+5.74/Rey^0.9))^2;
fb = fatt*L/D*wb*wb/2/9.81-H;

if fa*fb>0
    disp('Le due approssimazioni iniziali non separano lo zero');
    return
end

wold = 0.0;
% Ciclo del procedimento iterativo
for i = 1:100
   wx = wa-fa*(wb-wa)/(fb-fa);
   Rey = wx*D/ni;
   fatt = 0.25/(log10(e/D/3.7+5.74/Rey^0.9))^2;
   fx = fatt*L/D*wx*wx/2/9.81-H;
   if fx==0
      wb=wx;
      wa=wx;
   elseif fa*fx>0
      wa=wx;
      fa=fx;
   else
      wb=wx;
      fb=fx;
   end
   error = abs(wx-wold)/wx; % errore relativo approssimato
   if error<1e-6, break, end
   wold = wx;
end
i                % numero di iterazioni necessarie
wx               % velocità media [m/s]
Q=wx*pi*D*D/4    % portata volumetrica [m3/s]
```

```
------------------------------------------------------------

% Problema risolto con il metodo delle "approssimazioni successive" (in MATLAB)
clc, clear, close all

% Assegnazione dei dati di input
D = 0.3;        % diametro interno del tubo [m]
e = 0.03e-3;    % rugosità del tubo [m]
H = 400.0;      % dislivello tra i due bacini di raccolta [m]
L = 10000.0;    % lunghezza della tubazione [m]
ni = 1.13e-6;   % viscosità cinematica dell'acqua [m2/s]

wold = 1.0;     % velocità di primo tentativo

for i = 1:100
   Rey = wold*D/ni;
   fatt = 0.25/(log10(e/D/3.7+5.74/Rey^0.9))^2; % (Swamee & Jain)
   wnew = sqrt(H * 2 * 9.81 * D / fatt / L);
   error = abs(wnew - wold) / wnew; % errore relativo approssimato
   if error<1e-6, break, end
   wold = wnew;
end
i                % numero di iterazioni necessarie
wnew             % velocità media [m/s]
Q=wnew*pi*D*D/4  % portata volumetrica [m3/s]

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

% Problema risolto con il metodo delle "secanti" (in MATLAB)
clc, clear, close all

% Assegnazione dei dati di input
D = 0.3;        % diametro interno del tubo [m]
e = 0.03e-3;    % rugosità del tubo [m]
H = 400.0;      % dislivello tra i due bacini di raccolta [m]
L = 10000.0;    % lunghezza della tubazione [m]
ni = 1.13e-6;   % viscosità cinematica dell'acqua [m2/s]

% Approssimazioni iniziali
wa = 1.0;
Rey = wa*D/ni;
fatt = 0.25/(log10(e/D/3.7+5.74/Rey^0.9))^2; % (Swamee & Jain)
fa = fatt*L/D*wa*wa/2/9.81-H;

wb = 5.0;
Rey = wb*D/ni;
fatt = 0.25/(log10(e/D/3.7+5.74/Rey^0.9))^2;
fb = fatt*L/D*wb*wb/2/9.81-H;

for i = 1:100
    K = (fb-fa)/(wb-wa);
    w = wa - fa / K;
    error = abs(wa-w)/w; % errore relativo approssimato
    if error < 1e-6
       break
    end
    Rey = w*D/ni;
    fatt = 0.25/(log10(e/D/3.7+5.74/Rey^0.9))^2;
    f = fatt*L/D*w*w/2/9.81-H;
    fb = fa;
    wb = wa;
    fa = f;
```

```
    wa = w;
end
i                % numero di iterazioni necessarie
w                % velocità media [m/s]
Q=w*pi*D*D/4     % portata volumetrica [m3/s]
```

```
PROGRAM Bisezione
   ! Problema risolto con il metodo di "bisezione" (in FORTRAN 90)

   ! Dichiarazioni delle variabili
   IMPLICIT NONE
   REAL(KIND(0.d0)), PARAMETER      :: pi=3.1415926535898d00
   INTEGER                          :: i
   REAL(KIND(0.d0))                 :: D,e,H,L,ni
   REAL(KIND(0.d0))                 :: Rey,fatt,wa,wb,wx,fa,fb,fx,error,Q

   ! Assegnazione dei dati di input
   D = 0.3d00         ! diametro interno del tubo [m]
   e = 0.03d-03       ! rugosità del tubo [m]
   H = 400.d00        ! dislivello tra i due bacini di raccolta [m]
   L = 10000.d00      ! lunghezza della tubazione [m]
   ni = 1.13d-06      ! viscosità cinematica dell'acqua [m2/s]

   ! Approssimazioni iniziali
   wa = 1.d00
   Rey = wa*D/ni
   fatt = 0.25d00/(LOG10(e/D/3.7d00+5.74d00/Rey**0.9d00))**2.d00
   fa = fatt*L/D*wa*wa/2.d00/9.81d00-H

   wb = 5.d00
   Rey = wb*D/ni
   fatt = 0.25d00/(LOG10(e/D/3.7d00+5.74d00/Rey**0.9d00))**2.d00
   fb = fatt*L/D*wb*wb/2.d00/9.81d00-H

   IF (fa*fb>0) THEN
      WRITE (*,*) "Le due approssimazioni iniziali non separano lo zero"
      STOP
   END IF

   ! Ciclo del procedimento iterativo
   DO i=1,100
      wx = (wa+wb)/2.d00
      Rey = wx*D/ni
      fatt = 0.25d00/(LOG10(e/D/3.7d00+5.74d00/Rey**0.9d00))**2.d00
      fx = fatt*L/D*wx*wx/2.d00/9.81d00-H
      IF (fx==0.d00) THEN
         wa = wx
         wb = wx;
      ELSEIF (fa*fx>0.d00) THEN
         wa = wx
         fa = fx
      ELSE
         wb = wx
         fb = fx
      END IF
      error = DABS(wb-wa)/wx       ! Errore relativo
      IF (error<1.d-06) THEN
        GOTO 10
        END IF
   END DO

10 Q = wx*pi*D**2.d00/4.d00        ! Portata volumetrica
```

```
   WRITE(*,*) 'Numero di iterazioni necessarie: ',i
   WRITE(*,*) 'Velocità media acqua [m/s]:      ',wx
   WRITE(*,*) 'Portata volumetrica [m3/s]:      ',Q
PAUSE
END PROGRAM Bisezione

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

PROGRAM Regulafalsi
   ! Problema risolto con il metodo di "regula falsi" (in FORTRAN 90)

   ! Dichiarazioni delle variabili
   IMPLICIT NONE
   REAL(KIND(0.d00)), PARAMETER    :: pi=3.1415926535898d00
   INTEGER                            :: i
   REAL(KIND(0.d00))               :: D,e,H,L,ni
   REAL(KIND(0.d00))               :: Rey,fatt,wa,wb,wx,fa,fb,fx,error,Q,wold

   ! Assegnazione dei dati di input
   D = 0.3d00       ! diametro interno del tubo [m]
   e = 0.03d-03     ! rugosità del tubo [m]
   H = 400.d00      ! dislivello tra i due bacini di raccolta [m]
   L = 10000.d00    ! lunghezza della tubazione [m]
   ni = 1.13d-06    ! viscosità cinematica dell'acqua [m2/s]

   ! Approssimazioni iniziali
   wa = 1.d00
   Rey = wa*D/ni
   fatt = 0.25d00/(LOG10(e/D/3.7d00+5.74d00/Rey**0.9d00))**2.d00
   fa = fatt*L/D*wa*wa/2.d00/9.81d00-H

   wb = 5.d00
   Rey = wb*D/ni
   fatt = 0.25d00/(LOG10(e/D/3.7d00+5.74d00/Rey**0.9d00))**2.d00
   fb = fatt*L/D*wb*wb/2.d00/9.81d00-H

   IF (fa*fb>0) THEN
      WRITE (*,*) "Le due approssimazioni iniziali non separano lo zero"
      STOP
   END IF

   wold = 0.d00
   ! Ciclo del procedimento iterativo
   DO i=1,100
      wx = wa-fa*(wb-wa)/(fb-fa)
      Rey = wx*D/ni
      fatt = 0.25d00/(LOG10(e/D/3.7d00+5.74d00/Rey**0.9d00))**2.d00
      fx = fatt*L/D*wx*wx/2.d00/9.81d00-H
      IF (fx==0.d00) THEN
         wa = wx
         wb = wx
      ELSEIF (fa*fx>0.d00) THEN
         wa = wx
         fa = fx
      ELSE
         wb = wx
         fb = fx
      END IF
      error = DABS(wold-wx)/wx  ! Errore relativo
      IF (error<1.d-06) THEN
        GOTO 10
      END IF
      wold = wx
   END DO
```

```
10 Q = wx*pi*D**2.d00/4.d00     ! Portata volumetrica
   WRITE(*,*) 'Numero di iterazioni necessarie: ',i
   WRITE(*,*) 'Velocità media acqua [m/s]:      ',wx
   WRITE(*,*) 'Portata volumetrica [m3/s]:      ',Q

   PAUSE
END PROGRAM Regulafalsi
------------------------------------------------------------

PROGRAM Approximation
! Problema risolto con il metodo delle "approssimazioni successive"(in FORTRAN 90)

   ! Dichiarazioni delle variabili
   IMPLICIT NONE
   REAL(KIND(0.d00)), PARAMETER     :: pi=3.1415926535898d00
   INTEGER                          :: i
   REAL(KIND(0.d00))                :: D,e,H,L,ni
   REAL(KIND(0.d00))                :: Rey,fatt,wold,wnew,error,Q

   ! Assegnazione dei dati di input
   D = 0.3d00         ! diametro interno del tubo [m]
   e = 0.03d-03       ! rugosità del tubo [m]
   H = 400.d00        ! dislivello tra i due bacini di raccolta [m]
   L = 10000.d00      ! lunghezza della tubazione [m]
   ni = 1.13d-06      ! viscosità cinematica dell'acqua [m2/s]

   ! Approssimazioni iniziali
   WRITE(*,*) 'Inserisci il valore di primo tentativo per la velocità'
   READ(*,*) wold

   ! Ciclo del procedimento iterativo
   DO i = 1,100
      Rey = wold*D/ni;
      fatt=0.25d00/(lOG10(e/D/3.7d00+5.74d00/Rey**0.9d00))**2.d00;
      wnew = DSQRT(H * 2.d00 * 9.81d00 * D / fatt / L)
      error = DABS(wnew - wold) / wnew    ! errore relativo
      IF (error < 1.d-06) EXIT
      wold = wnew
   END DO

   Q = wnew*pi*D**2.d00/4.d00
   WRITE(*,*) 'Numero di iterazioni necessarie: ',i
   WRITE(*,*) 'Velocità media acqua [m/s]:      ',wnew
   WRITE(*,*) 'Portata volumetrica [m3/s]:      ',Q

   PAUSE

END PROGRAM Approximation


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

PROGRAM Secanti
   ! Problema risolto con il metodo di "regula falsi" (in FORTRAN 90)

   ! Dichiarazioni delle variabili
   IMPLICIT NONE
   REAL(KIND(0.d00)), PARAMETER    :: pi=3.1415926535898d00
   INTEGER                            :: i
   REAL(KIND(0.d00))                  :: D,e,H,L,ni
   REAL(KIND(0.d00))                  :: Rey,fatt,wa,wb,wx,fa,fb,fx,error,Q

   ! Assegnazione dei dati di input
```

```
   D = 0.3d00        ! diametro interno del tubo [m]
   e = 0.03d-03      ! rugosità del tubo [m]
   H = 400.d00       ! dislivello tra i due bacini di raccolta [m]
   L = 10000.d00     ! lunghezza della tubazione [m]
   ni = 1.13d-06     ! viscosità cinematica dell'acqua [m2/s]

   ! Approssimazioni iniziali
   wa = 1.d00
   Rey = wa*D/ni
   fatt = 0.25d00/(LOG10(e/D/3.7d00+5.74d00/Rey**0.9d00))**2.d00
   fa = fatt*L/D*wa*wa/2.d00/9.81d00-H

   wb = 5.d00
   Rey = wb*D/ni
   fatt = 0.25d00/(LOG10(e/D/3.7d00+5.74d00/Rey**0.9d00))**2.d00
   fb = fatt*L/D*wb*wb/2.d00/9.81d00-H

   ! Ciclo del procedimento iterativo
   DO i=1,100
      wx = wa-fa*(wb-wa)/(fb-fa)
      error = DABS(wa-wx)/wx  ! Errore relativo
      IF (error<1.d-06) EXIT
      Rey = wx*D/ni
      fatt = 0.25d00/(LOG10(e/D/3.7d00+5.74d00/Rey**0.9d00))**2.d00
      fx = fatt*L/D*wx*wx/2.d00/9.81d00-H
      fb=fa
      wb=wa
      fa=fx
      wa=wx
   END DO

   Q = wx*pi*D**2.d00/4.d00     ! Portata volumetrica
   WRITE(*,*) 'Numero di iterazioni necessarie: ',i
   WRITE(*,*) 'Velocità media acqua [m/s]:      ',wx
   WRITE(*,*) 'Portata volumetrica [m3/s]:      ',Q

   PAUSE
END PROGRAM Secanti
-------------------------------------------------------------
```

```
------------------------------------------------------------

// Problema risolto con il metodo di "bisezione" (in C++)

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>

// Dichiarazioni costanti
const double pi=3.1415926535898;

int main( )
{
    // Dichiarazioni delle variabili
    double D,e,H,L,ni;
    double Rey,fatt,wa,wb,wx,fa,fb,fx,error,Q;
    int i;

    // Assegnazione dei dati di input
    D = 0.3;          // diametro interno del tubo [m]
    e = 0.03e-3;      // rugosità del tubo [m]
    H = 400.0;        // dislivello tra i due bacini di raccolta [m]
    L = 10000.0;      // lunghezza della tubazione [m]
    ni = 1.13e-6;     // viscosità cinematica dell'acqua [m2/s]

    // Approssimazioni iniziali
    wa = 1.0;
    Rey = wa*D/ni;
    fatt = 0.25/pow(log10(e/D/3.7+5.74/pow(Rey,0.9)),2);
    fa = fatt*L/D*wa*wa/2/9.81-H;

    wb = 5.0;
    Rey = wb*D/ni;
    fatt = 0.25/pow(log10(e/D/3.7+5.74/pow(Rey,0.9)),2);
    fb = fatt*L/D*wb*wb/2/9.81-H;

    if (fa*fb>0.0)
    {
       printf("Le due approssimazioni iniziali non separano lo zero \n");
       return 0;
    }

    // Ciclo del procedimento iterativo
    for(i=1;i<=100;i++)
    {
       wx = (wa+wb)/2;
       Rey = wx*D/ni;
       fatt = 0.25/pow(log10(e/D/3.7+5.74/pow(Rey,0.9)),2);
       fx = fatt*L/D*wx*wx/2/9.81-H;
       if (fx==0.0)
       {
          wa=wx; wb=wx;
       }
       else if (fa*fx>0.0 )
       {
          wa=wx; fa=fx;
       }
       else
       {
          wb=wx; fb=fx;
       }
       error = fabs(wb-wa)/wx; // Errore relativo
       if (error<1.e-6) break;
```

```
    }
    Q = wx*pi*pow(D,2)/4.0; // Portata volumetrica
    printf("Numero di iterazioni necessarie per la convergenza: %d \n",i);
    printf("Velocità media del fluido nella tubazione: %f m/s \n",wx);
    printf("Portata volumetrica: %f m3/s \n",Q);
    system("PAUSE");
    return 0;
}
```

```
// Problema risolto con il metodo della "falsa posizione" (in C++)

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>

// Dichiarazioni costanti
const double pi=3.1415926535898;

int main( )
{
    // Dichiarazioni delle variabili
    double D,e,H,L,ni;
    double Rey,fatt,wa,wb,wx,fa,fb,fx,error,Q,wold;
    int i;

    // Assegnazione dei dati di input
    D = 0.3;          // diametro interno del tubo [m]
    e = 0.03e-3;      // rugosità del tubo [m]
    H = 400.0;        // dislivello tra i due bacini di raccolta [m]
    L = 10000.0;      // lunghezza della tubazione [m]
    ni = 1.13e-6;     // viscosità cinematica dell'acqua [m2/s]

    // Approssimazioni iniziali
    wa = 1.0;
    Rey = wa*D/ni;
    fatt = 0.25/pow(log10(e/D/3.7+5.74/pow(Rey,0.9)),2);
    fa = fatt*L/D*wa*wa/2/9.81-H;

    wb = 5.0;
    Rey = wb*D/ni;
    fatt = 0.25/pow(log10(e/D/3.7+5.74/pow(Rey,0.9)),2);
    fb = fatt*L/D*wb*wb/2/9.81-H;

    if (fa*fb>0.0)
    {
       printf("Le due approssimazioni iniziali non separano lo zero \n");
       return 0;
    }

    wold = 0.0;
    // Ciclo del procedimento iterativo
    for(i=1;i<=100;i++)
    {
       wx = wa-fa*(wb-wa)/(fb-fa);
       Rey = wx*D/ni;
       fatt = 0.25/pow(log10(e/D/3.7+5.74/pow(Rey,0.9)),2);
       fx = fatt*L/D*wx*wx/2/9.81-H;
       if (fx==0.0)
       {
          wa = wx;
          wb = wx;
```

```
        }
        else if (fa*fx>0.0 )
        {
           wa = wx;
           fa = fx;
        }
        else
        {
           wb=wx;
           fb=fx;
        }
        error = fabs(wx-wold)/wx;  // Errore relativo approssimato
        if (error<1.e-6) break;
          wold = wx;
    }

    Q = wx*pi*pow(D,2)/4.0; // Portata volumetrica
    printf("Numero di iterazioni necessarie per la convergenza: %d \n",i);
    printf("Velocità media del fluido nella tubazione: %f m/s \n",wx);
    printf("Portata volumetrica: %f m3/s \n",Q);
    system("PAUSE");
    return 0;
}
-----------------------------------------------------------------
```

1.3 Problema di irraggiamento e convezione combinati

Una termocoppia è situata sull'asse di un condotto metallico in cui scorre aria a T_a = 300 °C, per misurarne la temperatura. Il rivestimento della termocoppia ha una emissività ε = 0.9. Le pareti della tubazione si trovano alla temperatura T_p = 600 °C. Il coefficiente di convezione tra l'aria e la termocoppia vale h = 200 W/(m^2 K). Determinare la temperatura misurata dalla termocoppia utilizzando il MATLAB.

Soluzione

Dato che la termocoppia è a regime, il suo scambio termico globale deve essere nullo. In particolare, essa riceve calore per irraggiamento dalle pareti calde della tubazione e lo cede per convezione all'aria circostante. Detta quindi T la temperatura della termocoppia, si ha:

$$q''_{irr}+q''_{conv}=0$$

ed essendo la termocoppia stessa assimilabile ad un corpo molto piccolo contenuto dentro uno molto grande

$$\varepsilon\sigma\left(T^4-T_p^4\right)+h\left(T-T_a\right)=0$$

Si può risolvere l'equazione di quarto grado precedente sfruttando la funzione intrinseca `roots` del MATLAB oppure si può ottenere la soluzione implementando nel MATLAB un metodo numerico iterativo, come ad esempio quello delle sostituzioni successive. In alternativa si potrebbe usare il comando `fzero` del MATLAB.

```
% Calcolo della temperatura superficiale di una termocoppia
% con il metodo delle approssimazioni successive
clc, clear, close all

% Assegnazione dei dati di input
Ta = 300 + 273.15; % temperatura dell'aria [K]
Tp = 600 + 273.15; % temperatura della parete del tubo [K]
emi = 0.9;         % emissività della termocoppia [m]
h = 200.0;         % coefficiente di scambio termico convettivo [W/(m2 K)]
sigma = 5.67e-8;   % costante di Stefan-Boltzmann [W/(m2 K4)]
Told = Ta;
error = 1;

while error>1e-6
   Tnew = Ta + emi * sigma / h * ( Tp^4 - Told^4 );
   error = abs(Tnew - Told) / Tnew;
   Told = Tnew;
end
T=Tnew-273.15
```

```
function irraggia
% Calcolo della temperatura superficiale di una termocoppia
% con il metodo che fa uso del comando "fzero"

clc, clear, close all

global Ta Tp emi alfa sigma

% Assegnazione dei dati di input
Ta = 300 + 273.15; % temperatura dell'aria [K]
Tp = 600 + 273.15; % temperatura della parete del tubo [K]
emi = 0.9;         % emissività della termocoppia [m]
alfa = 200.0;      % coefficiente di scambio termico convettivo [W/(m2 K)]
sigma = 5.67e-8;   % costante di Stefan-Boltzmann [W/(m2 K4)]

T = fzero(@funirr,Ta);

T = T - 273.15     % temperatura misurata dalla termocoppia [°C]
end

function y = funirr(x);
% Sottofunzione richiamata nel comando fzero
global Ta Tp emi alfa sigma
y = alfa * (x - Ta) + emi * sigma * ( x^4 - Tp^4 );
end
```

1.4 Grafico della potenza di decadimento di un LWR

Si realizzi un programma che fornisca il grafico dell'andamento temporale della potenza termica di decadimento di un reattore nucleare LWR, calcolata sulla base dell'ANSI/ANS-5.1 nell'ipotesi conservativa di un periodo infinito di funzionamento prima dello scram. Per i calcoli si faccia riferimento alle unità 2, 3, 4 e 5 di Fukushima (BWR-4, MARK I) la cui potenza termica nominale era pari a 2380 MWth.

```
% Programma per graficare l'andamento temporale della potenza
% di decadimento per LWRs basato sull'ANSI/ANS-5.1 e valido
% nell'ipotesi conservativa di un periodo infinito di funzionamento
% prima dello scram.
% N.B.: Le unità 2, 3, 4 e 5 di Fukushima (BWR-4, MARK I) avevano
% una full thermal power di 2380 MWth, mentre l'unità 1 (BWR-3, MARK I)
% aveva una full thermal power di 1380 MWth.

clear               % Elimina vecchie variabili dalla memoria del PC
clc                 % Ripulisce la command window
close all           % Chiude tutte le finestre dei grafici rimaste aperte

% Dati di input
c = 0;              % Variabile per il ciclo while
p0 = input('Potenza iniziale in MW: \n');
while c==0
    td = input('Tempo trascorso in giorni (< di 46 gg): \n');
    t = td * 24 * 60 *60;
    if (t<4e6), c = 1; end,
end

% Inizio struttura if
if (t > 0)&(t <= 10)              % Primo intervallo (0<t<= 10)
    x1 = [1:t];
```

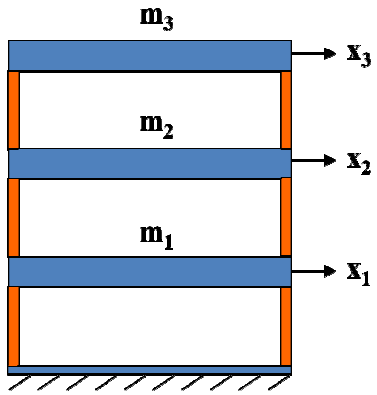
```
    pf1 = 0.0603 * (x1).^(-0.0639);
    x=x1; y=pf1
elseif (t > 10)&(t <= 150)          % Secondo intervallo (10<t<=150)
    x1 = [1:10];
    x2 = [11:t];
    pf1 = 0.0603 * (x1).^(-0.0639);
    pf2 = 0.0766 * (x2).^(-0.1810);
    x=[x1,x2]; y=[pf1,pf2];
else                                % Terzo intervallo (150<t<=4e6)
    x1 = [1:10];
    x2 = [11:150];
    x3 = [151:t];
    pf1 = 0.0603 * (x1).^(-0.0639);
    pf2 = 0.0766 * (x2).^(-0.1810);
    pf3 = 0.1300 * (x3).^(-0.2830);
    x=[x1,x2,x3]; y=[pf1,pf2,pf3];
end

semilogx(x,y*100),xlabel('Tempo [s]'),ylabel('Potenza [%]'),grid,
y(length(x))*p0
figure    % comando che consente di avere una seconda figura
semilogx(x,y*p0),xlabel('Tempo [s]'),ylabel('Potenza [MW]'),grid,
```

1.5 Calcolo delle frequenze naturali di un edificio

Si consideri l'edificio a tre piani riportato in figura avente 4 pilastri che collegano ogni solaio al piano precedente. Ogni pilastro può essere considerato con le due estremità incastrate ai solai per cui la costate elastica o rigidezza è data da:

$$k = \frac{12\,E\,I}{L^3}$$

Supponendo che le rigidezze dei pilastri di ogni singolo piano valgono $k_1=60\cdot10^6$, $k_2=50\cdot10^6$ e $k_3=40\cdot10^6$ N/m rispettivamente per il piano terra, il primo piano ed il secondo e che le masse dei tre solai sono $m_1=10\cdot10^3$ $m_2=9\cdot10^3$ e $m_3=8\cdot10^3$ kg si calcoli le frequenze naturali della struttura.




```
% Calcolo delle frequenze naturali di un edificio a due piani
clc, clear, close all

% Assegnazione dei dati di input
k1 = 60e6; % rigidezza [N/m]
k2 = 50e6; % rigidezza [N/m]
k3 = 40e6; % rigidezza [N/m]

m1 = 10e3; % massa [kg]
m2 = 9e3;  % massa [kg]
m3 = 8e3;  % massa [kg]

M(1,1) = m1;
M(2,2) = m2;
M(3,3) = m3;

K(1,1) = k1 + k2;
K(1,2) = -k2;
K(2,1) = -k2;
K(2,2) = k2 + k3;
K(2,3) = -k3;
K(3,2) = -k3;
K(3,3) = k3;

lambda = eig(K,M);
omega = lambda .^(0.5);
freq = omega / 2 / pi
```

2. USO DEI FILE DI FUNZIONE

2.1 Conversione della temperatura

Si realizzi un file di funzione che assegnati in ingresso il valore di temperatura in gradi centigradi ed un numero intero, *k*, scelto tra 1 2 o 3 restituisca in uscita la temperatura convertita in gradi Kelvin se *k*=1, in gradi Fahrenheit se *k*=2 e in gradi Rankine se *k*=3.

$$T_K = T_C + 273.15$$

$$T_F = \frac{9}{5} T_C + 32$$

$$T_R = \frac{9}{5} \left(T_C + 273.15 \right)$$

```
function t=tempconv(tc,k);

%--------------------------------------------------------------------%
% Funzione per la conversione della temperatura espressa
% in gradi Celsius in quella espressa in gradi Kelvin,
% Fahrenheit e Rankine

if k==1
   t = tc + 273.15; % gradi Kelvin
elseif k==2
   t = 9.0 / 5.0 * tc + 32.0; % gradi Fahrenheit
elseif k==3
   t = 9.0 / 5.0 * ( tc + 273.15 ); % gradi Rankine
else
   error('Errore');
end
```

2.2 Fattore di attrito

Si realizzi un file di funzione che assegnati in ingresso il valore del numero di Reynolds e della rugosità relativa restituisca in uscita il fattore di attrito calcolato mediante la formula di Churchill.

```
function f=darcy(Rey,epsr)
%--------------------------------------------------------------------%
% Funzione per il calcolo del fattore di attrito
% mediante la formula di Churchill
% Input  - Rey  = numero di Reynolds (scalare)
%        - epsr = rugosità relativa (scalare)
% Output - f    = fattore di attrito (scalare)
%--------------------------------------------------------------------%

A = ( - 2.457 * log( (7 / Rey)^0.9 + 0.27 * epsr ) )^16;
B = ( 37530 / Rey )^16;
f = 8 * ( ( 8 / Rey )^12 + (A + B)^(-1.5) )^(1 / 12);

end

%--------------------------------------------------------------------%
% Calcolo del fattore di attrito
% e realizzazione del diagramma di Moody
```

```
clc, clear, close all

% Assegnazione dei dati di input
Rey=logspace(3,8,1000);
epsr=[0,0.00005,0.0001,0.0002,0.0005,0.001,0.002,0.005,0.01,0.02,0.03];

for j=epsr
   for i=1:1000
      f(i) = darcy(Rey(i),j);
   end
   loglog(Rey,f)
   axis([1e3, 1e8, 0.01, 0.1])
   xlabel('Reynolds number'),ylabel('Friction factor')
   title(sprintf('Relative roughness = %10.5f',j))
   hold on
   pause
end
```

2.3 Potenza di decadimento di un LWR

Si realizzi una function MATLAB che ricevendo come dato in ingresso il tempo in secondi restituisca come dato in uscita la potenza termica di decadimento di un reattore nucleare LWR, calcolata sulla base delle formule dell'ANSI/ANS-5.1 nell'ipotesi conservativa di un periodo infinito di funzionamento prima dello scram:

$$\frac{\dot{Q}}{\dot{Q}_0} = a\,t^{-b}$$

dove

$a = 0.0603$ e $b = 0.0639$ per $t < 10$ s

$a = 0.0766$ e $b = 0.1810$ per $10 <= t < 150$ s

$a = 0.1300$ e $b = 0.2830$ per $150 <= t < 4.e6$ s

Si realizzi un programma MATLAB che utilizzi la suddetta function per graficare l'andamento temporale della potenza termica di decadimento normalizzata al valore nominale.

```
function y=decayheat(t)
 % Funzione per il calcolo della potenza termica di decadimento per un LWR
% basato sull'ANSI/ANS-5.1 e valido nell'ipotesi conservativa
% di un periodo infinito di funzionamento prima dello scram.

if (t>=0)&(t<10)
    y = 0.0603 * t ^ (-0.0639);
elseif (t>=10)&(t<150)
    y = 0.0766 * t ^ ( -0.1810);
elseif (t>=150)&(t<=4e6)
    y = 0.1300 * t ^ ( -0.2830);
else
    error('Errore: tempo troppo lungo (max. 4.e6 secondi)');
end
end
```

```
function graficodecayheat2
```

```
% Programma per graficare l'andamento temporale della potenza
% di decadimento per LWRs basato sull'ANSI/ANS-5.1 e valido
% nell'ipotesi conservativa di un periodo infinito di funzionamento
% prima dello scram.
% N.B.: Le unità 2, 3, 4 e 5 di Fukushima (BWR-4, MARK I) avevano
% una full thermal power di 2380 MWth, mentre l'unità 1 (BWR-3, MARK I)
% aveva una full thermal power di 1380 MWth.

clear               % Elimina vecchie variabili dalla memoria del PC
clc                 % Ripulisce la command window
close all           % Chiude tutte le finestre dei grafici rimaste aperte

% Dati di input
c = 0;              % Variabile per il ciclo while
p0 = input('Potenza iniziale in MW: \n');
while c==0
    td = input('Tempo trascorso in giorni (< di 46 gg): \n');
    t = td * 24 * 60 *60;
    if (t<4e6), c = 1; end,
end

x = logspace(0,log10(t),round(t/2));
y(1:round(t/2))=0.0;

for i=1:round(t/2)
    y(i) = decayheat(x(i));
end
semilogx(x,y*100,'o'),xlabel('Tempo [s]'),ylabel('Potenza [%]'),grid,
y(length(x))*p0
figure    % comando che consente di avere una seconda figura
semilogx(x,y*p0),xlabel('Tempo [s]'),ylabel('Potenza [MW]'),grid,

end
```

2.4 Risoluzione di un sistema di equazioni non lineari mediante il metodo di Newton-Raphson

Si risolva il sistema di equazioni non lineari riportato nel seguito:

$$\begin{cases} x_1 + 2x_2 - 2 = 0 \\ x_1^2 + 2x_2^2 - 4 = 0 \end{cases}$$

```
function Newton
% Parte principale del programma per la risoluzione
% di un sistema di equazioni non lineari
% x1 + 2*x2 - 2 = 0
% x1^2 + 4*x2^2 - 4 = 0

clc, clear, close all
format long
x0 = [1; 0];           % vettore soluzione di primo tentativo
[r, niter] = NR1(x0, eps, 100)
end

%--------------------------------------------------------------------%
function [r, niter] = NR1(x0, tol, maxiter)
% Metodo di Newton-Raphson per sistemi di equazioni non lineari

% r = zero del sistema di equazioni non lineari f(x) = 0
% x0 = approssimazione iniziale dello zero r
% Il calcolo viene interrotto sia se la norma di f
```

```
% all'approsimazione corrente risulta essere minore di tol
% sia se l'errore relativo tra due soluzioni successive
% risulta essere minore di tol, o se viene raggiunto
% il numero massimo di iterazioni maxiter.

Jc = rcond(J0(x0));     % reciproco del numero di condizionamento
                        % Se J è mal condizionata rcond è vicino ad eps
if Jc < 1e-10
   error('Tenta con una diversa approssimazione iniziale x0');
end

xold = x0;
for k=1:maxiter
   xnew = xold - J0(xold)\fun0(xold) % risolvo il sistema lineare
   if (norm(fun0(xnew))<tol)|(norm(xold-xnew,'inf')/norm(xnew,'inf')<tol)
      break
   end
   xold = xnew;
end

niter = k;     % numero di iterazioni effettivamente eseguite
r = xnew;      % zero del sistema di equazioni non lineari
end

%---------------------------------------------------------------------%
function f = fun0(x)
% Vettore delle funzioni valutate in x
f = zeros(2,1);                % vettore con due righe ed una colonna
f(1) = x(1) + 2*x(2) - 2;      % primo elemento (funzione) del vettore
f(2) = x(1)^2 + 4*x(2)^2 - 4; % secondo elemento (funzione) del vettore
end

%---------------------------------------------------------------------%
function J = J0(x)
% Matrice iacobiana di f valutata in x
J = [1 2;2*x(1) 8*x(2)];       % matrice jacobiana
end
```

2.5 Temperatura di saturazione dell'acqua in funzione della pressione

Si realizzi un file di funzione che assegnato in ingresso il valore della pressione in Pascal restituisca per l'acqua pura il valore della temperatura di saturazione in Kelvin, sfruttando l'equazione fornita dall'IAPWS-IF97.

```
function x=tsat(p)
%-------------------------------------------------------------------------%
%  Saturation temperature: tsat [K]; p[Pa]
%  Equation of IAPWS-IF97
%
%  Range of validity: 611.213=<p<=22.064e6 Pa
%-------------------------------------------------------------------------%
%
if (p>22.064e6)|(p<611.213)
    error('pressure data out of range')
end
%
n1= 0.11670521452767E+04; n2=-0.72421316703206E+06;
n3=-0.17073846940092E+02; n4=0.12020824702470E+05;
n5=-0.32325550322333E+07; n6=0.14915108613530E+02;
n7=-0.48232657361591E+04; n8=0.40511340542057E+06;
```

```
n9=-0.23855557567849E+00; n10=0.65017534844798E+03;
%
p=p*1e-6;p=p.^0.25;p2=p.*p;
E=p2+n3*p+n6;F=n1*p2+n4*p+n7;
G=n2*p2+n5*p+n8;D=2*G./(-F-(F.*F-4*E.*G).^0.5);
x=n10+D-((n10+D).^2-4*(n9+n10*D)).^0.5; x=0.5*x;
%
```

2.6 Pressione di saturazione dell'acqua in funzione della temperatura

Si realizzi un file di funzione che assegnato in ingresso il valore della temperatura in Kelvin restituisca per l'acqua pura la pressione di saturazione in Pascal, sfruttando l'equazione fornita dall'IAPWS-IF97.

```
function x=psat(t)
%  Saturation pressure: psat [Pa]; T[K]
%  Equation of IAPWS-IF97
%
%  Range of validity: 273.15=<T<=647.096 K
%-------------------------------------------------------------------------%
%
if (t>647.096)|(t<273.15)
    error('temperature data out of range')
end
%
n1= 0.11670521452767E+04; n2=-0.72421316703206E+06;
n3=-0.17073846940092E+02; n4=0.12020824702470E+05;
n5=-0.32325550322333E+07; n6=0.14915108613530E+02;
n7=-0.48232657361591E+04; n8=0.40511340542057E+06;
n9=-0.23855557567849E+00; n10=0.65017534844798E+03;
%
t=t+n9./(t-n10);
t2=t.*t;
A=t2+n1.*t+n2;B=n3.*t2+n4.*t+n5;C=n6.*t2+n7.*t+n8;
x=2*C./(-B+sqrt(B.*B-4*A.*C)); x=x.*x.*x.*x.*1e6;
%
```

2.7 Proprietà termodinamiche dell'acqua in fase liquida

Si realizzi un file di funzione che dati il valore della pressione in Pascal e la temperatura in Kelvin restituisca le proprietà termodinamiche, con unità di misura del Sistema Internazionale (SI), dell'acqua in fase liquida calcolate mediante le equazioni dell'IAPWS-IF97.

```
function H2Ol=liquid(p,t)
%-------------------------------------------------------------------------%
%  Liquid properties in SI
%  Equations of IAPWS-IF97
%
%  Range of validity: 273.15=<T<=623.15 K; psat(T)=<p<=100e6 Pa
%-------------------------------------------------------------------------%
I(1)=0;  J(1)=-2;  n(1)=0.14632971213167;
I(2)=0;  J(2)=-1;  n(2)=-0.84548187169114;
I(3)=0;  J(3)=0;   n(3)=-0.37563603672040e1;
I(4)=0;  J(4)=1;   n(4)= 0.33855169168385e1;
I(5)=0;  J(5)=2;   n(5)=-0.95791963387872;
I(6)=0;  J(6)=3;   n(6)=0.15772038513228;
I(7)=0;  J(7)=4;   n(7)=-0.16616417199501e-1;
I(8)=0;  J(8)=5;   n(8)=0.81214629983568e-3;
I(9)=1;  J(9)=-9;  n(9)=0.28319080123804e-3;
```

```
I(10)=1; J(10)=-7; n(10)=-0.60706301565874e-3;
I(11)=1; J(11)=-1; n(11)=-0.18990068218419e-1;
I(12)=1; J(12)=0;  n(12)=-0.32529748770505e-1;
I(13)=1; J(13)=1;  n(13)=-0.21841717175414e-1;
I(14)=1; J(14)=3;  n(14)=-0.52838357969930e-4;
I(15)=2; J(15)=-3; n(15)=-0.47184321073267e-3;
I(16)=2; J(16)=0;  n(16)=-0.30001780793026e-3;
I(17)=2; J(17)=1;  n(17)=0.47661393906987e-4;
I(18)=2; J(18)=3;  n(18)=-0.44141845330846e-5;
I(19)=2; J(19)=17; n(19)=-0.72694996297594e-15;
I(20)=3; J(20)=-4; n(20)=-0.31679644845054e-4;
I(21)=3; J(21)=0;  n(21)=-0.28270797985312e-5;
I(22)=3; J(22)=6;  n(22)=-0.85205128120103e-9;
I(23)=4; J(23)=-5; n(23)=-0.22425281908000e-5;
I(24)=4; J(24)=-2; n(24)=-0.65171222895601e-6;
I(25)=4; J(25)=10; n(25)=-0.14341729937924e-12;
I(26)=5; J(26)=-8; n(26)=-0.40516996860117e-6;
I(27)=8; J(27)=-11;n(27)=-0.12734301741641e-8;
I(28)=8; J(28)=-6; n(28)=-0.17424871230634e-9;
I(29)=21;J(29)=-29;n(29)=-0.68762131295531e-18;
I(30)=23;J(30)=-31;n(30)=0.14478307828521e-19;
I(31)=29;J(31)=-38;n(31)=0.26335781662795e-22;
I(32)=30;J(32)=-39;n(32)=-0.11947622640071e-22;
I(33)=31;J(33)=-40;n(33)=0.18228094581404e-23;
I(34)=32;J(34)=-41;n(34)=-0.93537087292458e-25;
%
p=p*1e-6;Rw=0.461526;
tau=1386/t;pi=p/16.53;
g=0;gp=0;gt=0;gtt=0;gpp=0;gpt=0;
for i=1:34
    tauj=(tau-1.222)^J(i);
    pii=n(i)*(7.1-pi)^I(i);
    g=g+pii*tauj;
    gp=gp-pii/(7.1-pi)*I(i)*tauj;
    gt=gt+pii*J(i)*tauj/(tau-1.222);
    gtt=gtt+pii*J(i)*(J(i)-1)*tauj/(tau-1.222)/(tau-1.222);
    gpp=gpp+I(i)*(I(i)-1)*pii/(7.1-pi)/(7.1-pi)*tauj;
    gpt=gpt-I(i)*J(i)*pii/(7.1-pi)*tauj/(tau-1.222);
end
vl=Rw*t/p*pi*gp*1e-3;
hl=Rw*t*tau*gt*1e3;
ul=hl-p*vl*1e6;
sl=(tau*gt-g)*Rw*1e3;
cpl=-Rw*tau*tau*gtt*1e3;
cvl=cpl+Rw*(gp-tau*gpt)^2/gpp*1e3;
betal=(gp-tau*gpt)*Rw*1e-3/vl/16.35;
kappal=t*vl*betal*betal/(cpl-cvl);
viscl=mi(t,vl);
thconl=ck(t,vl);
%  Delta Lambda correction for the conductivity
delta=1/vl/317.763;tau=647.226/t;psi=viscl/55.071e-6;
dpidtau=647.226*16.53*(gpt*1386-gp*t)/22.115/t/t/gpp;
ddeltadpi=-22.115*gpp/317.763/Rw/t/gp/gp*1e3;
add2=0.0013848/psi*(tau*delta)^(-2)*dpidtau*dpidtau*(delta*ddeltadpi)^0.4678;
add2=add2*sqrt(delta)*exp(-18.66*(1/tau-1)^2-(delta-1)^4);
thconl=thconl+add2*0.4945;
%
H2Ol.p=p*1.e6;H2Ol.t=t;
H2Ol.v=vl;H2Ol.h=hl;H2Ol.u=ul;H2Ol.s=sl;H2Ol.cp=cpl;H2Ol.cv=cvl;
H2Ol.visc=viscl;H2Ol.cond=thconl;H2Ol.beta=betal;H2Ol.kappa=kappal;
%
%-------------------------------------------------------------------------%
function x=mi(t,v)
%-------------------------------------------------------------------------%
```

```
%  Dynamic viscosity: mi [Pa s)]; t [K]; v [m3/kg]
%  Equation of IAPWS-IF97
%
%  Range of validity: 273.15<=T<=423.15 K  and p<=500 MPa
%                     423.15<=T<=873.15 K  and p<=350 MPa
%                     873.15<=T<=1173.15 K and p<=300 MPa
%--------------------------------------------------------------------%
tst=647.226;rhost=317.763;
a1=0.978197;a2=0.579829;a3=-0.202354;
I(1)=0;  J(1)=0;  n(1)=0.5132047;
I(2)=0;  J(2)=1;  n(2)=0.3205656;
I(3)=0;  J(3)=4;  n(3)=-0.7782567;
I(4)=0;  J(4)=5;  n(4)=0.1885447;
I(5)=1;  J(5)=0;  n(5)=0.2151778;
I(6)=1;  J(6)=1;  n(6)=0.7317883;
I(7)=1;  J(7)=2;  n(7)=0.1241044E+01;
I(8)=1;  J(8)=3;  n(8)=0.1476783E+01;
I(9)=2;  J(9)=0;  n(9)=-0.2818107;
I(10)=2; J(10)=1; n(10)=-0.1070786E+01;
I(11)=2; J(11)=2; n(11)=-0.1263184E+01;
I(12)=3; J(12)=0; n(12)=0.1778064;
I(13)=3; J(13)=1; n(13)=0.4605040;
I(14)=3; J(14)=2; n(14)=0.2340379;
I(15)=3; J(15)=3; n(15)=-0.4924179;
I(16)=4; J(16)=0; n(16)=-0.4176610E-01;
I(17)=4; J(17)=3; n(17)=0.1600435;
I(18)=5; J(18)=1; n(18)=-0.1578386E-01;
I(19)=6; J(19)=3; n(19)=-0.3629481E-02;
%
ratt=t/tst; ratrho=1/v/rhost; rratt=1.0/ratt;
xt=rratt-1; xr=ratrho-1;
den0=((a3*rratt+a2)*rratt+a1)*rratt+1;
recden=1.0/den0;
dyvfu0=sqrt(ratt)*recden;
%
x=0.0;
for i=1:19
    x=x+n(i)*xr^I(i)*xt^J(i);
end
x=x*ratrho; x=dyvfu0*exp(x)*55.071e-6;
%
%--------------------------------------------------------------------%
function x=ck(t,v)
%--------------------------------------------------------------------%
%  Thermal conductivity: ck [W/(m K)]; t [K]; v [m3/kg]
%  Equation of IAPWS-IF97
%
%  Range of validity: 273.15<=T<=398.15 K  and p<=400 MPa
%                     398.15<=T<=523.15 K  and p<=200 MPa
%                     523.15<=T<=673.15 K  and p<=150 MPa
%                     673.15<=T<=1073.15 K and p<=100 MPa
%  Note: The conductivity formulation has been implemented without
%        the Delta Lambda term
%--------------------------------------------------------------------%
tst=647.226;rhost=317.763;
a1=6.978267;a2=2.599096;a3=-0.998254;
b00=1.3293046;  b10=1.7018363;  b20=5.2246158;  b30=8.7127675;  b40=-1.8525999;
b01=-0.40452437;b11=-2.2156845; b21=-10.124111; b31=-9.5000611; b41=0.9340469;
b02=0.2440949;  b12=1.6511057;  b22=4.9874687;  b32=4.3786606;  b42=0.0;
b03=0.018660751;b13=-0.76736002;b23=-0.27297694;b33=-0.91783782;b43=0.0;
b04=-0.12961068;b14=0.37283344; b24=-0.43083393;b34=0.0;        b44=0.0;
b05=0.044809953;b15=-0.1120316; b25=0.13333849; b35=0.0;        b45=0.0;
%
ratt=t/tst; ratrho=1/v/rhost; rratt=1.0/ratt;
```

```
xt=rratt-1; xr=ratrho-1;
den0=((a3*rratt+a2)*rratt+a1)*rratt+1;
thcfu0=sqrt(ratt)/den0;
%
e0=(((b40*xt+b30)*xt+b20)*xt+b10)*xt+b00;
e1=(((b41*xt+b31)*xt+b21)*xt+b11)*xt+b01;
e2=(((b42*xt+b32)*xt+b22)*xt+b12)*xt+b02;
e3=(((b43*xt+b33)*xt+b23)*xt+b13)*xt+b03;
e4=(((b44*xt+b34)*xt+b24)*xt+b14)*xt+b04;
e5=(((b45*xt+b35)*xt+b25)*xt+b15)*xt+b05;
x=((((e5*xr+e4)*xr+e3)*xr+e2)*xr+e1)*xr+e0;
x=x*ratrho; x=thcfu0*exp(x)*0.4945;
%
```

2.8 Proprietà termodinamiche dell'acqua in fase vapore surriscaldato

Si realizzi un file di funzione che dati il valore della pressione in Pascal e della temperatura in Kelvin restituisca le proprietà termodinamiche, con unità di misura del Sistema Internazionale (SI), dell'acqua in fase vapore calcolate mediante le equazioni dell'IAPWS-IF97.

```
%------------------------------------------------------------------------%
function H2Ov=vapour(p,t)
%------------------------------------------------------------------------%
%  Vapour properties in SI
%  Equations of IAPWS-IF97
%
%  Range of validity: 273.15=<T<=623.15 K; 611.213=<p<=psat(T) Pa
%                     623.15=<T<=1073.15 K; 611.213=<p<=16.52916e6 Pa
%------------------------------------------------------------------------%
J0(1)=0;   n0(1)=-0.96927686500217e+1;
J0(2)=1;   n0(2)=0.10086655968018E+2;
J0(3)=-5;  n0(3)=-0.56087911283020E-2;
J0(4)=-4;  n0(4)=0.71452738081455E-1;
J0(5)=-3;  n0(5)=-0.40710498223928;
J0(6)=-2;  n0(6)=0.14240819171444E+1;
J0(7)=-1;  n0(7)=-0.43839511319450E+1;
J0(8)=2;   n0(8)=-0.28408632460772;
J0(9)=3;   n0(9)=0.21268463753307E-1;
%
I(1)=1;   J(1)=0;    n(1)=-0.17731742473213E-2;
I(2)=1;   J(2)=1;    n(2)=-0.17834862292358E-1;
I(3)=1;   J(3)=2;    n(3)=-0.45996013696365E-1;
I(4)=1;   J(4)=3;    n(4)=-0.57581259083432E-1;
I(5)=1;   J(5)=6;    n(5)=-0.50325278727930E-1;
I(6)=2;   J(6)=1;    n(6)=-0.33032641670203E-4;
I(7)=2;   J(7)=2;    n(7)=-0.18948987516315E-3;
I(8)=2;   J(8)=4;    n(8)=-0.39392777243355E-2;
I(9)=2;   J(9)=7;    n(9)=-0.43797295650573E-1;
I(10)=2;  J(10)=36; n(10)=-0.26674547914087E-4;
I(11)=3;  J(11)=0;  n(11)=0.20481737692309E-7;
I(12)=3;  J(12)=1;  n(12)=0.43870667284435E-6;
I(13)=3;  J(13)=3;  n(13)=-0.32277677238570E-4;
I(14)=3;  J(14)=6;  n(14)=-0.15033924542148E-2;
I(15)=3;  J(15)=35; n(15)=-0.40668253562649E-1;
I(16)=4;  J(16)=1;  n(16)=-0.78847309559367E-9;
I(17)=4;  J(17)=2;  n(17)=0.12790717852285E-7;
I(18)=4;  J(18)=3;  n(18)=0.48225372718507E-6;
I(19)=5;  J(19)=7;  n(19)=0.22922076337661E-5;
I(20)=6;  J(20)=3;  n(20)=-0.16714766451061E-10;
I(21)=6;  J(21)=16; n(21)=-0.21171472321355E-2;
I(22)=6;  J(22)=35; n(22)=-0.23895741934104E+2;
I(23)=7;  J(23)=0;  n(23)=-0.59059564324270E-17;
I(24)=7;  J(24)=11; n(24)=-0.12621808899101E-5;
I(25)=7;  J(25)=25; n(25)=-0.38946842435739E-1;
I(26)=8;  J(26)=8;  n(26)=0.11256211360459E-10;
I(27)=8;  J(27)=36; n(27)=-0.82311340897998E+1;
I(28)=9;  J(28)=13; n(28)=0.19809712802088E-7;
I(29)=10; J(29)=4;  n(29)=0.10406965210174E-18;
I(30)=10; J(30)=10; n(30)=-0.10234747095929E-12;
I(31)=10; J(31)=14; n(31)=-0.10018179379511E-8;
I(32)=16; J(32)=29; n(32)=-0.80882908646985E-10;
I(33)=16; J(33)=50; n(33)=0.10693031879409;
I(34)=18; J(34)=57; n(34)=-0.33662250574171;
I(35)=20; J(35)=20; n(35)=0.89185845355421E-24;
I(36)=20; J(36)=35; n(36)=0.30629316876232E-12;
I(37)=20; J(37)=48; n(37)=-0.42002467698208E-5;
```

```
I(38)=21; J(38)=21; n(38)=-0.59056029685639E-25;
I(39)=22; J(39)=53; n(39)=0.37826947613457E-5;
I(40)=23; J(40)=39; n(40)=-0.12768608934681E-14;
I(41)=24; J(41)=26; n(41)=0.73087610595061E-28;
I(42)=24; J(42)=40; n(42)=0.55414715350778E-16;
I(43)=24; J(43)=58; n(43)=-0.94369707241210E-6;
%
p=p*1e-6;Rw=0.461526;
tau=540/t;pi=p;
g0=log(pi);
g=0;gp=0;gpp=0;gpt=0;gt0=0;gt=0;gtt0=0;gtt=0;
for i=1:9
    g0=g0+n0(i)*tau^J0(i);
    gp0=1/pi;
    gt0=gt0+n0(i)*J0(i)*tau^(J0(i)-1);
    gtt0=gtt0+n0(i)*J0(i)*(J0(i)-1)*tau^(J0(i)-2);
    gpp0=-1/pi/pi;
end
for i=1:43
    tauj=(tau-0.5)^J(i);
    pii=pi^I(i);
    g=g+n(i)*pii*tauj;
    gp=gp+n(i)*I(i)*pii*tauj/pi;
    gt=gt+n(i)*pii*J(i)*tauj/(tau-0.5);
    gtt=gtt+n(i)*pii*J(i)*(J(i)-1)*tauj/(tau-0.5)/(tau-0.5);
    gpp=gpp+n(i)*I(i)*(I(i)-1)*pii*tauj/pi/pi;
    gpt=gpt+n(i)*I(i)*pii/pi*J(i)*tauj/(tau-0.5);
end
vv=Rw*t/p*(1+pi*gp)*1e-3;
hv=Rw*t*tau*(gt0+gt)*1e3;
uv=hv-p*vv*1e6;
sv=(tau*(gt+gt0)-(g+g0))*Rw*1e3;
cpv=-Rw*tau*tau*(gtt0+gtt)*1e3;
cvv=cpv-Rw*(1+pi*gp-tau*pi*gpt)^2/(1-pi*pi*gpp)*1e3;
betav=(1/pi+gp-tau*gpt)*Rw*1e-3/vv;
kappav=t*vv*betav*betav/(cpv-cvv);
viscv=mi(t,vv);
thconv=ck(t,vv);
%  Delta Lambda correction for the conductivity
delta=1/vv/317.763;tau=647.226/t;psi=viscv/55.071e-6;
dpidtau=647.226*(gpt*540-(gp+gp0)*t)/22.115/t/t/(gpp+gpp0);
ddeltadpi=-22.115*(gpp+gpp0)/317.763/Rw/t/(gp+gp0)^2*1e3;
add2=0.0013848/psi*(tau*delta)^(-2)*dpidtau*dpidtau*(delta*ddeltadpi)^0.4678;
add2=add2*sqrt(delta)*exp(-18.66*(1/tau-1)^2-(delta-1)^4);
thconv=thconv+add2*0.4945;
%
H2Ov.p=p*1e6;H2Ov.t=t;
H2Ov.v=vv;H2Ov.h=hv;H2Ov.u=uv;H2Ov.s=sv;H2Ov.cp=cpv;H2Ov.cv=cvv;
H2Ov.visc=viscv;H2Ov.cond=thconv;H2Ov.beta=betav;H2Ov.kappa=kappav;
%
%------------------------------------------------------------------------%
function x=mi(t,v)
%------------------------------------------------------------------------%
%  Dynamic viscosity: mi [Pa s)]; t [K]; v [m3/kg]
%  Equation of IAPWS-IF97
%
%  Range of validity: 273.15<=T<=423.15 K  and p<=500 MPa
%                     423.15<=T<=873.15 K  and p<=350 MPa
%                     873.15<=T<=1173.15 K and p<=300 MPa
%------------------------------------------------------------------------%
tst=647.226;rhost=317.763;
a1=0.978197;a2=0.579829;a3=-0.202354;
I(1)=0;  J(1)=0;  n(1)=0.5132047;
I(2)=0;  J(2)=1;  n(2)=0.3205656;
```

```
I(3)=0;  J(3)=4;  n(3)=-0.7782567;
I(4)=0;  J(4)=5;  n(4)=0.1885447;
I(5)=1;  J(5)=0;  n(5)=0.2151778;
I(6)=1;  J(6)=1;  n(6)=0.7317883;
I(7)=1;  J(7)=2;  n(7)=0.1241044E+01;
I(8)=1;  J(8)=3;  n(8)=0.1476783E+01;
I(9)=2;  J(9)=0;  n(9)=-0.2818107;
I(10)=2; J(10)=1; n(10)=-0.1070786E+01;
I(11)=2; J(11)=2; n(11)=-0.1263184E+01;
I(12)=3; J(12)=0; n(12)=0.1778064;
I(13)=3; J(13)=1; n(13)=0.4605040;
I(14)=3; J(14)=2; n(14)=0.2340379;
I(15)=3; J(15)=3; n(15)=-0.4924179;
I(16)=4; J(16)=0; n(16)=-0.4176610E-01;
I(17)=4; J(17)=3; n(17)=0.1600435;
I(18)=5; J(18)=1; n(18)=-0.1578386E-01;
I(19)=6; J(19)=3; n(19)=-0.3629481E-02;
%
ratt=t/tst; ratrho=1/v/rhost; rratt=1.0/ratt;
xt=rratt-1; xr=ratrho-1;
den0=((a3*rratt+a2)*rratt+a1)*rratt+1;
recden=1.0/den0;
dyvfu0=sqrt(ratt)*recden;
%
x=0.0;
for i=1:19
    x=x+n(i)*xr^I(i)*xt^J(i);
end
x=x*ratrho; x=dyvfu0*exp(x)*55.071e-6;
%
%------------------------------------------------------------------------%
function x=ck(t,v)
%------------------------------------------------------------------------%
%  Thermal conductivity: ck [W/(m K)]; t [K]; v [m3/kg]
%  Equation of IAPWS-IF97
%
%  Range of validity: 273.15<=T<=398.15 K  and p<=400 MPa
%                     398.15<=T<=523.15 K  and p<=200 MPa
%                     523.15<=T<=673.15 K  and p<=150 MPa
%                     673.15<=T<=1073.15 K and p<=100 MPa
%  Note: The conductivity formulation has been implemented without
%        the Delta Lambda term
%------------------------------------------------------------------------%
tst=647.226;rhost=317.763;
a1=6.978267;a2=2.599096;a3=-0.998254;
b00=1.3293046;  b10=1.7018363;  b20=5.2246158;  b30=8.7127675;  b40=-1.8525999;
b01=-0.40452437;b11=-2.2156845; b21=-10.124111; b31=-9.5000611; b41=0.9340469;
b02=0.2440949;  b12=1.6511057;  b22=4.9874687;  b32=4.3786606;  b42=0.0;
b03=0.018660751;b13=-0.76736002;b23=-0.27297694;b33=-0.91783782;b43=0.0;
b04=-0.12961068;b14=0.37283344; b24=-0.43083393;b34=0.0;        b44=0.0;
b05=0.044809953;b15=-0.1120316; b25=0.13333849; b35=0.0;        b45=0.0;
%
ratt=t/tst; ratrho=1/v/rhost; rratt=1.0/ratt;
xt=rratt-1; xr=ratrho-1;
den0=((a3*rratt+a2)*rratt+a1)*rratt+1;
thcfu0=sqrt(ratt)/den0;
%
e0=(((b40*xt+b30)*xt+b20)*xt+b10)*xt+b00;
e1=(((b41*xt+b31)*xt+b21)*xt+b11)*xt+b01;
e2=(((b42*xt+b32)*xt+b22)*xt+b12)*xt+b02;
e3=(((b43*xt+b33)*xt+b23)*xt+b13)*xt+b03;
e4=(((b44*xt+b34)*xt+b24)*xt+b14)*xt+b04;
e5=(((b45*xt+b35)*xt+b25)*xt+b15)*xt+b05;
%
```

```
x=((((e5*xr+e4)*xr+e3)*xr+e2)*xr+e1)*xr+e0;
x=x*ratrho; x=thcfu0*exp(x)*0.4945;
%
```

2.9 Risoluzione di un sistema di equazioni lineari con un metodo iterativo

Si risolva il sistema di equazioni riportato nel seguito con il metodo di Jacobi, con quello di Gauss-Siodel e con il metodo SOR, utilizzando un opportuno coefficiente di sovrarilassamento.

$$\begin{cases} 2x_1 - 2x_2 = 1 \\ -2x_1 + 3x_2 - x_3 = 1 \\ x_1 - x_2 + 4x_4 = 1 \end{cases}$$

```
%-------------------------------------------------------------------------%
function x = jacobi(A,b,xold)
%-------------------------------------------------------------------------%
% Iterative solution of the linear system Ax = b by the Jacobi method
%
% Input data:
%    A    coefficient matrix for linear system (must be a square matrix)
%    b    right-hand side vector for linear system (coloumn vector)
%    xold vector containing initial guess for the solution (row vector)
%
% Output data:
%    x    approximate solution
%-------------------------------------------------------------------------%

n = length(b); [r c] = size(A);
if ((c ~= n)|(r ~= c))
   error ('Matrix A dimensions and/or vector b dimension not compatible')
end

xnew = zeros(1,n);

for iter = 1:10000

    xnew(1) = ( -sum( A(1,2:n) .* xold(2:n) ) + b(1) ) / A(1,1);
    for i = 2:n-1
        xnew(i) = ( -sum( A(i,1:i-1) .* xold(1:i-1) ) -sum( A(i,i+1:n) .*
xold(i+1:n) ) + b(i) ) / A(i,i);
    end
    xnew(n) = ( -sum( A(n,1:n-1) .* xold(1:n-1) ) + b(n) ) / A(n,n);

    err = norm(xnew - xold) / norm(xnew);
    if (err < 1e-8)
       x = xnew;
       iter_jacobi = iter
       return
    end
    xold = xnew;

end

error('Maximum number of 10000 iterations exceeded');

end
```

```
%-------------------------------------------------------------------------%
function x = gauss_sidel(A,b,xold)
%-------------------------------------------------------------------------%
% Iterative solution of the linear system Ax = b by the Gauss-Sidel method
%
% Input data:
%    A    coefficient matrix for linear system (must be a square matrix)
%    b    right-hand side vector for linear system (coloumn vector)
%    xold vector containing initial guess for the solution (row vector)
%
% Output data:
%    x    approximate solution
%-------------------------------------------------------------------------%

n = length(b); [r c] = size(A);
if ((c ~= n)|(r ~= c))
   error ('Matrix A dimensions and/or vector b dimension not compatible')
end

xnew = zeros (1,n);

for iter = 1:10000

    xnew(1) = ( -sum( A(1,2:n) .* xold(2:n) ) + b(1)  ) / A(1,1);
    for i = 2:n-1
        xnew(i) = ( -sum( A(i,1:i-1) .* xnew(1:i-1) ) -sum( A(i,i+1:n) .*
xold(i+1:n) ) + b(i) ) / A(i,i);
    end
    xnew(n) = ( -sum( A(n,1:n-1) .* xnew(1:n-1) ) + b(n) ) / A(n,n);

    err = norm(xnew - xold) / norm(xnew);
    if (err < 1e-8)
       x = xnew;
       iter_GS = iter
       return
    end
    xold = xnew;

end

error('Maximum number of 10000 iterations exceeded');

end

%-------------------------------------------------------------------------%
function x = SOR(A,b,xold,omega)
%-------------------------------------------------------------------------%
% Iterative solution of the linear system Ax = b by the SOR method
%
% Input data:
%    A    coefficient matrix for linear system (must be a square matrix)
%    b    right-hand side vector for linear system (coloumn vector)
%    xold vector containing initial guess for the solution (row vector)
%
% Output data:
%    x    approximate solution
%-------------------------------------------------------------------------%

n = length(b); [r c] = size(A);
if ((c ~= n)|(r ~= c))
   error ('Matrix A dimensions and/or vector b dimension not compatible')
end

xnew = zeros(1,n);
```

```
for iter = 1:10000

    xnew(1) = ( -sum( A(1,2:n) .* xold(2:n) ) + b(1)  ) / A(1,1);
    for i = 2:n-1
        xnew(i) = ( -sum( A(i,1:i-1) .* xnew(1:i-1) ) -sum( A(i,i+1:n) .*
xold(i+1:n) ) + b(i) ) / A(i,i);
    end
    xnew(n) = ( -sum( A(n,1:n-1) .* xnew(1:n-1) ) + b(n) ) / A(n,n);
    xnew = xold + omega * (xnew - xold); % over-relaxation

    err = norm(xnew - xold) / norm(xnew);
    if (err < 1e-8)
       x = xnew;
       iter_SOR = iter
       return
    end
    xold = xnew;

end

error('Maximum number of 10000 iterations exceeded');

end
```

3. LETTURA/SCRITTURA DATI, INTERPOLAZIONE, REGRESSIONE E INTEGRAZIONE

3.1 Taratura di un misuratore di portata

Per la taratura di un misuratore di portata di acqua viene riempito ripetutamente un recipiente graduato fino alla tacca corrispondente a 10 litri ed ogni volta viene registrato il tempo necessario per il riempimento mediante un cronometro che approssima il centesimo di secondo. La taratura viene effettuata per quattro differenti portate; per ogni valore della portata, l'operazione di raccolta dell'acqua a 20 °C viene ripetuta 5 volte consecutivamente e gli intervalli di tempo (in secondi) e la corrispondente tensione (in volt) vengono immagazzinati dal sistema di acquisizione dati (SAD) in un file di nome dati.txt in forma colonnare come segue:

```
100.19    4.08
100.25    4.13
99.71     4.03
100.14    4.10
99.86     4.05
150.12    3.58
150.03    3.56
149.91    3.51
149.83    3.49
150.12    3.59
199.75    3.00
200.09    3.05
200.01    3.03
199.98    3.01
199.89    3.04
249.75    2.48
250.19    2.52
250.05    2.49
249.97    2.47
250.06    2.51
```

Realizzare un file MATLAB che legga i dati dal file dati.txt e determini:

- le quattro portate medie (litri/min), relative ai quattro differenti livelli di portata, ed i quattro corrispondenti valori medi di tensione;
- l'equazione della curva di calibrazione parabolica che lega la portata (in litri/min) alla tensione acquisita (volt).

Riportare su di uno stesso grafico sia i quattro punti sperimentali ottenuti mediando i dati acquisiti dal SAD sia l'andamento della portata in funzione della tensione ottenuto dalla curva di calibrazione.

```
% Taratura di un misuratore di portata
clc
clear
close all

% lettura dei dati dal file dati.txt
fid = fopen('dati.txt');
A = fscanf(fid,'%f',[2,inf])';
fclose(fid);

Y1 = mean(A(1:5,:));
Y2 = mean(A(6:10,:));
Y3 = mean(A(11:15,:));
Y4 = mean(A(16:20,:));

Y = [Y1;Y2;Y3;Y4];      % matrice delle portate e delle tensioni medie

y = 10./Y(:,1)*60;      % vettore portate medie in litri/min

x = Y(:,2);             % vettore tensioni medie in volt

% curva di calibrazione parabolica
p = polyfit(x,y,2)      % coefficienti del polinomio di grado 2

t = linspace(min(x),max(x),1000);   % vettore delle ascisse curva di calib.
pp = p(1)*t.*t+p(2)*t+p(3);         % vettore delle ordinate curva di calib.

% grafico dei dati sperimentali e della curva di calibrazione parabolica
plot(x,y,'or',t,pp),xlabel('tensione [V]'),ylabel('Portata [l/min]'),grid,
```

3.2 Consumo di carburante di un'auto nuova

Per verificare il consumo di una nuova automobile, la stessa viene provata su di un circuito automobilistico. Vengono analizzate 12 differenti velocità medie.

I dati della velocità media e del corrispondente carburante consumato nel percorrere 100 km sono riportati nella tabella seguente.

Velocità media [km/h]	*Consumo totale [l]*
10.0	10.2
20.2	9.5
29.5	9.0
40.5	8.5
50.1	8.0
59.9	7.3
70.4	6.8
80.2	6.4
90.5	6.3
110.5	6.2
129.4	7.1
149.1	8.3

Realizzare un file MATLAB che legga i dati precedenti dal file dati.txt e che visualizzi un grafico che mostri l'andamento dei chilometri percorsi per litro di carburante in funzione della velocità media.
Si determini l'equazione del polinomio di terzo grado che approssima i dati di cui sopra, fornendo il valore dei coefficienti del polinomio stesso.
Riportare su di un nuovo grafico sia i punti ottenuti dalle misure che quelli dell'andamento polinomiale.
Si valuti, inoltre, il consumo di carburante in corrispondenza di una velocità media di 100 km/h percorsi sempre sulla stessa pista in cui sono state effettuate le altre prove sperimentali.
Determinare, infine, il coefficiente di correlazione *R* dei dati sperimentali relativo alla funzione polinomiale di regressione determinata precedentemente.

```
% consumo di carburante auto nuova
clc
clear
close all
format long e     % formato esponenziale con 16 cifre significative

% lettura dei dati dal file dati.txt
fid = fopen('dati.txt');
A = fscanf(fid,'%f %f',[2,inf])';
fclose(fid);

wm = A(:,1);            % vettore delle velocità

c = 100 ./ A(:,2);      % vettore dei consumi

% determinazione del polinomio di regressione
p = polyfit(wm,c,3)

% creazione dei vettori ascissa ed ordinata della curva di regressione
xp = linspace(min(wm),max(wm),1000);
yp = polyval(p,xp);
polyval(p,100)          % consumo stimato per la velocità di 100 km/h

% grafico dei dati iniziali e della relativa curva di regressione
plot(wm,c,'o',xp,yp);
xlabel('velocità media [km/h]'),ylabel('consumo specifico [km/l]'), grid,

% calcolo del coefficiente di correlazione
S2 = sum((c - mean(c)) .^ 2)
E = sum((c - polyval(p,wm)) .^ 2)
R = sqrt((S2 - E) / S2)
```

3.3 Superamento di un dislivello durante una corsa podistica

Una corsa podistica effettuata su una distanza totale di 10 km richiede il superamento da parte dei partecipanti di una collina con un dislivello in salita tra il punto di partenza e la quota massima di 900 m ed in discesa tra la quota massima e l'arrivo alla quota di 300 m. I dati della quota rispetto al livello del mare in funzione della distanza percorsa dai podisti dal punto di partenza sono riportati in tabella.

Distanza percorsa [km]	*Quota [m]*
0	100
1	300
3	500
5	700
6	900
8	700
9	450
10	300

Realizzare un file MATLAB che legga i dati precedenti dal file dati.txt e si realizzi un grafico che mostri l'andamento della quota in funzione della distanza percorsa.

Supponendo che il consumo medio di energia da parte dei podisti per superare 100 m di dislivello in salita sia mediamente di 50 kcal e in discesa sia la metà, si realizzi un grafico dell'andamento del consumo energetico da parte di un podista in funzione della distanza percorsa. Scrivere, infine, i dati della distanza percorsa e del corrispondente consumo energetico, in forma colonnare, nel file risultati.txt.

```
% Superamento di un dislivello durante una corsa podistica
clc
clear
close all

%lettura dati dal file dati.txt
fid = fopen('dati.txt');
D = fscanf(fid,'%f %f',[2,inf])';
fclose(fid);

d = D(:,1);q=D(:,2);
Cal = 50;

plot(d,q); xlabel('distanza [km]'); ylabel('quota [m]');

c(1:length(d)) = 0

for i=2:length(d)
   if (q(i)-q(i-1))>=0
       c(i) = c(i-1)+(q(i)-q(i-1))/100*Cal;
```

```
    else
        c(i) = c(i-1)-(q(i)-q(i-1))/100*Cal/2
    end
end

figure
plot(d,c), xlabel('distanza [km]'), ylabel('consumo [kcal]'),

% scrittura dati nel file risultati.txt
fid=fopen('risultati.txt','a');
for i=1:length(d)
    fprintf(fid,'%f %f \n',d(i),c(i));
end
fclose(fid);
```

3.4 Viscosità di un nuovo fluido refrigerante

Per determinare l'andamento della viscosità dinamica di un nuovo fluido refrigerante al variare della sua temperatura viene utilizzato un viscosimetro a rotazione. I dati sperimentali ottenuti per la sola fase liquida sono i seguenti:

T [°C]	μ [Pa s]
20.01	5.9e-04
31.10	5.2E-04
40.05	4.6E-04
49.92	4.1E-04
60.13	3.8E-04
70.02	3.6E-04
79.98	3.4E-04
90.02	3.3E-04

Realizzare un file MATLAB che legga i dati precedenti dal file dati.txt e determini l'equazione della curva di regressione polinomiale di terzo grado.

Riportare su di uno stesso grafico sia gli otto punti dei dati sperimentali sia l'andamento del curva polinomiale di best fit.

Determinare, infine, il coefficiente di correlazione R dei dati sperimentali relativo alla funzione polinomiale di regressione determinata precedentemente.

```
% Viscosità di un nuovo fluido refrigerante
clear
clc
close all

% tabella di input
T = [20.01 31.01 40.05 49.92 60.13 70.02 79.98 90.02]';
mu = [5.9e-4 5.2e-4 4.6e-4 4.1e-4 3.8e-4 3.6e-4 3.4e-4 3.3e-4]';

fid = fopen('dati.txt','w');
fprintf(fid,' T  \t\t mu \n');
for i = 1:8
    fprintf(fid,' %f \t %f \n',T(i),mu(i));
end
fclose(fid);

y = polyfit(T,mu,3);
Tn=linspace(min(T),max(T),100);
mun = polyval(y,Tn);

fid = fopen('dati3.txt','r');
tline=fgetl(fid);
A = fscanf(fid,'%f %f',[2 inf])';
fclose(fid);

plot(A(:,1),A(:,2),'.',Tn,mun,'r')
title('fluido refrigerante')
xlabel('temperatura T [*C]')
ylabel('viscosita` mu [Pa*s]')
legend('dati sper.','best fit')

muy = polyval(y,T);
fm = mean(mu);
S2 = sum((mu-fm).^2);
E = sum((muy-mu).^2);
R = sqrt((S2-E)/S2)
```

3.5 Volume specifico di un gas reale

Per calcolare il volume specifico di un gas reale, in sostituzione dell'equazione per gas ideali, può essere utilizzata l'equazione di Van der Waals:

$$\left(p+\frac{a}{v^2}\right)(v-b)=\mathbb{R}T$$

dove il termine b è una correzione per il volume delle molecole ed il termine a/v^2 è una correzione per l'attrazione delle molecole. Il termine $\mathbb{R}$ è la costante dei gas perfetti (0.082054 litri atm/(mole K)), T è la temperatura assoluta e v è il volume specifico del gas (espressa in litri/mole). I valori di a e di b dipendono dal tipo di gas. Per l'anidride carbonica a = 3.59 litri^2 atm / mole^2 e b = 0.0427 litri/mole. Scrivere un programma che fornisca il volume specifico v (in litri/mole) dell'anidride carbonica (CO_2), ricavato con l'equazione di Van der Waals, in corrispondenza di T = 300 K e per differenti valori della pressione: 1, 2, 3, 4, 5, 6, 7 atm. A tal fine, come valore di primo tentativo per il calcolo del volume specifico si utilizzi quello ottenuto dall'equazione dei gas perfetti.

Stampare su di un file di output i dati di pressione nella prima colonna in bar (1 atm = 1.01325 bar) e quelli del corrispondente volume specifico nella seconda in m^3/kg, sapendo che il peso molecolare dell'anidride carbonica vale $44.01*10^{-3}$ kg/mole.

Riportare i punti precedentemente trovati su di un grafico in cui sulle ascisse c'è la pressione (in bar) e sulle ordinate il volume specifico (in m^3/kg).

```
function realgas
% Calcolo del volume specifico di un gas reale (cloro)
clc
close all

global R a b T pp

R = 0.082054;     % Costante universale dei gas perfetti [litri atm / (mole K)]
a = 3.59;         % Costante [litri^2 atm / mole^2]
b = 0.0427;       % Modulo di elasticità normale [litri/mole]
T = 300;          % Temperatura [K]
PM = 44.01e-3;    % Peso molecolare [kg/mole]

for i=1:7
    pp=i;
    p(i)=i;
    v1 = R * T / p(i);
    v(i) = fzero(@myfun,v1);
end

p = p * 1.01325;
v = v / PM * 1e-3;

% Scrittura dei dati nel file output.txt
fid=fopen('output.txt','a');
for i=1:7
    fprintf(fid,'%f %f \n',p(i),v(i));
end
fclose(fid);

plot(p,v,'o');
xlabel('pressione [bar]');
ylabel('volume specifico [m3/kg]');

% ------------------------------------------------------------
function fx = myfun(x)
global R a b T pp
fx = (pp + a / x / x)*(x - b) - R * T;
```

3.6 Misura dello spessore di un film d'acqua

Un sensore che rileva lo spessore di un film di liquido in caduta su di una piastra verticale ha fornito i dati dell'andamento dello spessore in funzione del tempo riportati nel file *film.txt* (la prima colonna rappresenta il tempo in secondi, mentre la seconda rappresenta il corrispondente spessore in millimetri).

Scrivere all'interno di un file di output, chiamato *output.txt*, le seguenti grandezze derivate:

- lo spessore medio, minimo e massimo;
- la deviazione standard del segnale da valutare con la formula:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\delta - \mu)^2}$$

dove: N = numeri di misurazioni;

δ = spessore del film;

μ = spessore medio del film.

Realizzare, infine, un grafico dell'andamento dello spessore in funzione del tempo.

```
% Misura dello spessore di un film d'acqua
clc
clear
close all

fid = fopen('film.txt');
A = fscanf(fid,'%f %f',[2,inf])';
fclose(fid);

t = A(:,1);
d = A(:,2);

media = mean(d);
minimo = min(d);
massimo = max(d);
sigma = sqrt(mean((d-media).^2));

fid = fopen('output.txt','w');
fprintf(fid,'media = %f mm \n',media);
fprintf(fid,'minimo = %f mm \n',minimo);
fprintf(fid,'massimo = %f mm \n',massimo);
fprintf(fid,'sigma = %f mm \n',sigma);
fclose(fid);

plot(t,d);
```

3.7 Taratura di una termocoppia

Per la taratura di una termocoppia viene utilizzato un termometro campione con una precisione di 0.01 °C ed un fornetto termostatato. La taratura viene effettuata in un intervallo che va da circa 20 °C a circa 100 °C, con incrementi della temperatura del fornetto di circa 10 °C. I valori della temperatura misurata mediante il termometro campione e quella corrispondente misurata mediante la termocoppia sono riportati, rispettivamente, nella prima e nella seconda colonna del file dati.txt come segue:

20.01 20.23
29.90 30.07
40.05 40.10
50.02 50.00
60.03 59.95
70.02 69.95
79.98 79.85
90.02 89.90
99.90 99.76

Realizzare un file MATLAB che legga i dati dal file dati.txt e determini:

- l'errore (differenza tra la temperatura fornita dal termometro campione e la temperatura fornita dalla termocoppia) commesso dalla termocoppia in corrispondenza ad ognuno dei 9 livelli di temperatura;
- l'equazione della curva di correzione parabolica che fornisce il valore da sommare alla temperatura fornita dalla termocoppia per ridurre l'incertezza associata alla misura.

Riportare su di uno stesso grafico sia i 9 punti di errore ricavati dai dati sperimentali (in funzione della temperatura del termometro campione) sia l'andamento del curva polinomiale di best fit.

```
% Taratura di una termocoppia
clc
clear

% lettura dei dati dal file dati.txt
fid = fopen('dati.txt');
A = fscanf(fid,'%f',[2,inf])'
fclose(fid);

x = A(:,1);
y = A(:,1) - A(:,2);

% curva di calibrazione parabolica
p = polyfit(x,y,2);
t = linspace(min(x),max(x),1000);
pp = p(1) * t .* t + p(2) * t + p(3);
plot(x,y,'or',t,pp), xlabel('Errore [°C]'), ylabel('Temperatura [°C]'), grid
```

3.8 Acqua scaricata da un recipiente

Si conosce l'andamento della portata di acqua che fuoriesce da un contenitore in funzione del tempo. Il valore di portata viene acquisito ad intervalli di tempo di 1 s mediante un misuratore il cui segnale viene inviato ad un PC. I dati (in kg/s) ottenuti in sequenza sono quelli riportati in tabella.

0
1.5
2.0
3.0
3.5
4.0
4.1
4.15
4.16
4.05
3.7
2.9
1.3
1.0
0.3
0

Realizzare un file MATLAB che legga i dati dal file dati.txt e si determini la massa di acqua fuoriuscita dal contenitore e la portata media. Disegnare infine un grafico che mostri l'andamento della portata in funzione del tempo.

Si determini la funzione polinomiale di quarto grado, con il metodo di regressione ai minimi quadrati, e si riporti il suo andamento sullo stesso grafico in cui sono mostrati i punti sperimentali.

```
% Acqua scaricata da un recipiente
clc
clear

% lettura dati
fid=fopen('dati.txt');
W=fscanf(fid,'%f %f',[1,inf])';
fclose(fid);

t=[0:length(W)-1]';
s1 = 0;
for i=2:length(W)
   s1 = s1 + 0.5 * ( W(i-1) + W(i) ) * ( t(i) - t(i-1) );
end
fprintf('massa di acqua = %f kg\n',s1);
fprintf('portata media  = %f kg/s\n',mean(W));

p = polyfit(t,W,4);
tp=linspace(min(t),max(t),1000);
Wp = polyval(p,tp);
plot(t,W,'o',tp,Wp);
```

3.9 Problema sulla regressione

Determinare i parametri a e b tali che la funzione $f(x) = a\,e^{bx}$ approssimi ai minimi quadrati i seguenti 6 punti:

x_i	1.1	2.8	4.3	5.4	6.8	8.0
y_i	7.5	16.1	38.9	67.0	146.6	266.5

Realizzare un file MATLAB che legga i dati della suddetta tabella dal file dati.txt e che determini i coefficienti a e b della funzione di best fit di tipo esponenziale.

Disegnare un grafico x-y che riporti sia la posizione dei punti della tabella che l'andamento della curva di best fit.

Suggerimento: una semplificazione potrebbe essere quella di trasformare la funzione esponenziale in una di tipo lineare applicando il logaritmo a primo ed a secondo membro.

```
% Problema sulla regressione
clc
clear

% lettura dei dati dal file esterno
fid = fopen('dati.txt');
A = fscanf(fid,'%f %f',[2,inf])';
fclose(fid);

x = A(:,1); y = A(:,2);
ylog = log(y);

% determinazione della funzione esponenziale
p = polyfit(x,ylog,1)
a = exp(p(2))
b =p(1)

% creazione dei vettori ascissa ed ordinata della curva di regressione
xp = linspace(min(x),max(x),1000);
yp = a * exp(b*xp);

% grafico dei dati iniziali e della relativa curva di regressione
plot(x,y,'o',xp,yp);

% calcolo del coefficiente di correlazione
S2 = sum((y - mean(y)).^2)
E = sum((y - a.*exp(b.*x)).^2)
R = sqrt((S2-E)/S2)
```

3.10 Misuratore di pressione differenziale

Un misuratore di pressione differenziale posto a cavallo di un orifizio calibrato viene utilizzato per la misura indiretta della portata di acqua che attraversa un circuito. Il costruttore ha fornito lo strumento con la seguente legge che lega il Δp (espresso in Pa) alla portata massica (espressa in kg/s):

$$W = 4.5 * \sqrt{\Delta p}$$

Prima del suo uso lo strumento è stato sottoposto a calibrazione, mediante il metodo della "pesata", ottenendo i seguenti risultati (immagazzinati nel file dati00.txt):

Δp [Pa]	W [kg/s]
9	14
51	33
109	48
121	51
143	55
150	56

Supponendo trascurabili gli errori commessi nel calcolo della portata massica con il metodo della pesata, si determini la nuova curva di calibrazione con legge del tipo:

$$W = Cost * \sqrt{\Delta p}$$

Realizzare un grafico di W in funzione di Δp su cui si riportino sia i dati sperimentali, sia l'andamento derivante dalla nuova curva di calibrazione.

Riportare su un secondo grafico l'andamento dell'errore relativo commesso dalla legge fornita dal costruttore rispetto alla nuova curva di calibrazione in funzione della portata.

```
% Misuratore di pressione differenziale
Clear, clc, close all

format short

% tabella dati input
Dp = [9 51 109 121 143 150]; % pressione differenziale [Pa]
W = [14 33 48 51 55 56];     % portata massica [kg/s]
A = [Dp ; W];
dp = sqrt(Dp);

fid = fopen('DpW.txt','w');
fprintf(fid,'Dp\t\t\t  W \n');
fprintf(fid,'%-f\t  %-f\t \n',A);
fclose(fid);

wc = 4.5*dp;

c = sum(dp.*W)/sum(Dp)
ws = c*dp;

subplot(2,1,1)
```

```
plot(dp,W,'o',dp,wc,'g',dp,ws,'b')
legend('dati','costuttore')
xlabel('sqrt(Dp)'),ylabel('W')

err = (ws-wc)./ws;

subplot(2,1,2)
plot(W,err,'o')
xlabel('W'),ylabel('errore')
```

3.11 Integrazione numerica di una funzione

Si integri la funzione $y = \sin(\pi x)$ sull'intervallo [0, 1] con la formula dei trapezi e con la regola 1/3 di Simpson. Si confronti il risultato numerico ottenuto con i due suddetti metodi con la soluzione esatta e con la soluzione numerica ottenuta utilizzando la funzione intrinseca `quad` di MATLAB.

Svolgimento

```
function quadratura
% calcolo dell'integrale della funzione sin(pi*x) sull'intervallo [0,1]

clc, clear, close all, format long
% Dati di input
a = 0; b = 1;
n = input('Inserire il numero di intervalli (pari):  \n');
% Dati di output
St = trapezi(@fun,a,b,n)
Ss = simprl(@fun,a,b,n)
Sq = quad(@fun,a,b)
ErrorSt = abs(St-2/pi)/2*pi*100
ErrorSs = abs(Ss-2/pi)/2*pi*100
ErrorSq = abs(Sq-2/pi)/2*pi*100
%-------------------------------------------------------------
function S=trapezi(f,a,b,n)
%Input    - f è la funzione integranda
%         - a e b sono i limiti inferiore e superiore di integrazione
%         - n è il numero di sottointervalli
%Output   - S è la somma ottenuta con la regola dei trapezi
dx = (b - a) / n;
s1 = 0;
for i=1:(n-1)
   x = a + dx * i;              % x(i) con i dispari
   s1 = s1 + f(x);              % somma degli f(x(i))
end
S = dx * (f(a) + 2 * s1 + f(b)) / 2;
%-------------------------------------------------------------
function S=simprl(f,a,b,n)
%Input    - f è la funzione integranda
%         - a e b sono i limiti inferiore e superiore di integrazione
%         - n è il numero di sottointervalli
%Output   - S è la somma ottenuta con la regola di Simpson
dx = (b - a) / n;
s1=0; s2=0;
for i=1:n/2
   x = a + dx * (2 * i - 1); % x(i) con i dispari
```

```
   s1 = s1 + f(x);            % somma degli f(x(i)) con i dispari
end
for i=1:(n/2-1)
   x = a + dx * 2 * i;        % x(i) con i pari
   s2 = s2 + f(x);            % somma degli f(x(i)) con i pari
end
S = dx * (f(a) + 4 * s1 + 2 * s2 + f(b)) / 3;
%-------------------------------------------------------------
function y=fun(x);
y = sin(pi*x);
```

Un modo alternativo per risolvere con il MATLAB il problema assegnato è il seguente:

```
function quadraturan
% calcolo dell'integrale della funzione sin(pi*x) sull'intervallo [0,1]

clc, clear, close all, format long
% Dati di input
a = 0; b = 1;
n = input('Inserire il numero di intervalli (pari):  \n');
% Dati di output
dx = (b - a) / n;  x = [a:dx:b];  y = fun(x);
% integrazione numerica con la regola dei trapezi
St = dx * (y(1) + 2 * sum(y(2:n)) + y(n+1)) / 2;
% integrazione numerica con la regola di Simpson
Ss = dx * (y(1) + 4 * sum(y(2:2:n)) + 2 * sum(y(3:2:n-1)) + y(n+1)) / 3;
% integrazione numerica con il comando quad di MATLAB
Sq = quad(@fun,a,b)
% calcolo degli errori associati
ErrorSt = abs(St-2/pi)/2*pi*100
ErrorSs = abs(Ss-2/pi)/2*pi*100
ErrorSq = abs(Sq-2/pi)/2*pi*100

end
%-------------------------------------------------------------
function y=fun(x)
y = sin(pi*x);
end
```

4. EQUAZIONI DIFFERENZIALI ALLE DERIVATE ORDINARIE (ODE)

4.1 Risoluzione di un'equazione differenziale ordinaria del primo ordine (IVP)

Si risolva numericamente l'equazione differenziale dy/dx=f(x,y) dove: f(x,y)=-2*x^3+12*x^2-20*x+8.5 sull'intervallo [0, 4] e con la condizione iniziale y(x=0) = 1. Si risolva il suddetto problema con il metodo di Eulero, con il metodo di Eulero modificato e con il metodo di Runge-Kutta del quarto ordine. Si confronti nei tre casi la soluzione numerica con quella analitica.

```
function eulero
%-------------------------------------------------------------------%
% Esempio di applicazione del metodo di Eulero esplicito
% per integrare numericamente l'eq. differenziale dy/dx=f(x,y)
% dove: f(x,y)=-2*x^3+12*x^2-20*x+8.5 sull'intervallo [0, 4]
% e con la condizione iniziale y(x=0) = 1.
%-------------------------------------------------------------------%

clc, clear, close all

% Intervallo di integrazione
a = 0;
b = 4;

% Condizioni iniziali
x(1) = a;
y(1) = 1;

h = input('Inserisci il passo \n');

% Soluzione numerica (approssimata)
for i=1:round((b-a)/h)
    x(i+1) = x(i) + h;
    y(i+1) = y(i) + f(x(i),y(i)) * h;
end

% Soluzione analitica (esatta)
xv = linspace(a,b,100);
yv = -0.5 * xv.^4 + 4 * xv.^3 - 10 * xv.^2 + 8.5 * xv + 1;

% Confronto grafico delle due soluzioni
plot(x,y,'ob',xv,yv,'r',x,y); xlabel('x'); ylabel('y');
legend('soluzione numerica','soluzione analitica');

end
%-----------------------------------------------------------%
function dydx = f(x,y)
dydx = -2 * x^3 + 12 * x^2 - 20 * x + 8.5;
end
```

```
function eulerom
%-------------------------------------------------------------------%
% Esempio di applicazione del metodo di Eulero modificato
% per integrare numericamente l'eq. differenziale dy/dx=f(x,y)
% dove: f(x,y)=-2*x^3+12*x^2-20*x+8.5 sull'intervallo [0, 4]
% e con la condizione iniziale y(x=0) = 1.
```

```
%-------------------------------------------------------------------------%

clc, clear, close all

% Intervallo di integrazione
a = 0;
b = 4;

% Condizioni iniaziali
x(1) = a;
y(1) = 1;
h=input('Inserisci il passo \n');

% Soluzione numerica (approssimata)
for i=1:round((b-a)/h)
    x(i+1) = x(i) + h;
    y(i+1) = y(i) + f(x(i),y(i)) * h;
    y(i+1) = y(i) + 0.5 * ( f(x(i),y(i)) + f(x(i+1),y(i+1)) ) * h;
end

% Soluzione analitica (esatta)
xv = linspace(a,b,100);
yv = -0.5 * xv.^4 + 4 * xv.^3 - 10 * xv.^2 + 8.5 * xv + 1;

% Confronto grafico delle due soluzioni
plot(x,y,'ob',xv,yv,'r',x,y); xlabel('x'); ylabel('y');
legend('soluzione numerica','soluzione analitica');

end
%-----------------------------------------------------------%
function dydx = f(x,y)
dydx = -2 * x^3 + 12 * x^2 - 20 * x + 8.5;
end
```

```
function RK4
%-------------------------------------------------------------------------%
% Esempio di applicazione del metodo di Runge-Kutta del quarto ordine
% per integrare numericamente l'eq. differenziale dy/dx=f(x,y)
% dove: f(x,y)=-2*x^3+12*x^2-20*x+8.5 sull'intervallo [0, 4]
% e con la condizione iniziale y(x=0) = 1
%-------------------------------------------------------------------------%

clc, clear, close all

% Intervallo di integrazione
a = 0;
b = 4;

% Condizioni iniaziali
x(1) = a;
y(1) = 1;
h=input('Inserisci il passo \n');

% Soluzione numerica (approssimata)
for i=1:round((b-a)/h)
    x(i+1) = x(i) + h;
    k1 = f(x(i),y(i));
    k2 = f(x(i)+0.5*h,y(i)+0.5*k1*h);
    k3 = f(x(i)+0.5*h,y(i)+0.5*k2*h);
    k4 = f(x(i)+h,y(i)+h);
```

```
    y(i+1) = y(i) + ( k1 + 2*k2 + 2*k3 + k4 ) /6 * h;
end

% Soluzione analitica (esatta)
xv = linspace(a,b,100);
yv = -0.5 * xv.^4 + 4 * xv.^3 - 10 * xv.^2 + 8.5 * xv + 1;

% Confronto grafico delle due soluzioni
plot(x,y,'ob',xv,yv,'r',x,y); xlabel('x'); ylabel('y');
legend('soluzione numerica','soluzione analitica');

end
%-------------------------------------------------------%
function dydx = f(x,y)
dydx = -2 * x^3 + 12 * x^2 - 20 * x + 8.5;
end
```

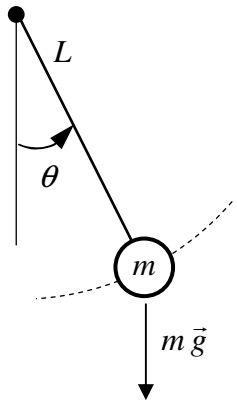
4.2 Moto del pendolo (IVP)

Si analizzi il moto del pendolo nel caso di grandi oscillazioni e si confronti i risultati ottenuti con quelli relativi al caso dell'approssimazione per piccole oscillazioni. Si assuma che la massa venga lasciata cadere da un angolo iniziale di $\theta_0 = \pi/4$ con velocità di rotazione nulla.

Soluzione

Si consideri il pendolo mostrato in figura costituito da una massa concentrata *m* collegata all'estremità di un'asta di massa trascurabile e di lunghezza L. L'equazione del moto della massa m è ottenibile dall'equazione differenziale non lineare:

$$m\vec{a} = \vec{F} \;\Rightarrow\; \ddot{\theta} = -\frac{g}{L}\sin\theta$$



Se si applica l'approssimazione delle piccole oscillazioni, cioè se si pone $\sin\theta \cong \theta$ allora la precedente equazione differenziale diventa lineare ed omogenea e la sua soluzione particolare:

$$\theta = \theta_0 \cos\sqrt{\frac{g}{L}}\,t$$

Il moto del corpo risulterebbe di tipo armonico con periodo $P = 2\pi/\sqrt{\frac{g}{L}}$ ed ampiezza θ_0.

Se non si applica la semplificazione delle piccole oscillazioni, ponendo $y_1 = \dot{\theta}$ e $y_2 = \ddot{\theta}$, l'equazione differenziale del secondo ordine precedente può essere scritta come un sistema di due equazioni differenziali del primo ordine

$$\begin{cases} \dfrac{dy_1}{dt} = y_2 \\ \dfrac{dy_2}{dt} = -\dfrac{g}{L}\sin y_1 \end{cases}$$

A cui vanno aggiunte le due condizioni iniziali: $y_1(t=0)=\pi/4$ e $y_2(t=0)=0$.

```
function pendolo
clc
clear
close all

global g L
g=9.81; L=1;

tspan = [0, 4*pi]; % intervallo di integrazione
y0 = [pi/4, 0.0];  % condizioni iniziali
[t,y] = ode23(@pend,tspan,y0); % soluzione numerica esatta

t1 = linspace(tspan(1),tspan(2),1000);
y1 = y0(1) .* cos((g/L)^0.5.*t1); % soluzione analitica approssimata
plot(t,y(:,1),'o',t1,y1);
%-------------------------------------------------------------
function dydt=pend(t,y)
global g L
dydt=[y(2);-(g/L)*sin(y(1))];
```

4.3 Moto del pendolo con attrito viscoso (IVP)

Si analizzi il moto del pendolo del problema precedente sulla cui massa agisca anche una forza per attrito viscoso e cioè una forza del tipo:

$$\vec{F}_v = -C\,\vec{w}$$

Si supponga che la massa concentrata *m* all'estremità del pendolo sia pari ad 1 kg e che la costante di attrito viscoso *C* sia uguale 0.5 kg/s.

Soluzione

L'equazione del moto della massa *m* è in questo caso data da:

$$m\vec{a} = \vec{F} \;\Rightarrow\; \ddot{\theta} = -\frac{C}{mL^2}\dot{\theta} - \frac{g}{L}\sin\theta$$

Come si nota, al contrario del problema 4.2, l'equazione del moto viene a dipendere dalla massa *m* del corpo.

```
function odependolo
% Modello di pendolo non lineare con attrito viscoso

clc, clear, close all

global m g L C

% Variabili di input
m=1; g=9.81; L=1;
C = 0.5;     % costante di attrito viscoso viscoso (kg/s)
```

```
h = 0.01;    % incremento della variabile indipendente (dt)
T = 10;      % periodo del transitorio da studiare
t=[0:h:T];   % assegnazione del vettore dei tempi
Y = zeros(length(t),2);
Y(1,1) = pi; % angolo iniziale in [rad]
Y(1,2) = 10; % velocità di rotazione iniziale [rad/s]

% Sistema di eq. diff. risolto con il metodo di Runge-Kutta a 4 passi
for i=1:length(t)-1
   k1=pend(t(i),Y(i,:));              % f(x,y)
   k2=pend(t(i)+h/2,Y(i,:)+h.*k1'/2); % f(x+h/2;y+h*k1/2)
   k3=pend(t(i)+h/2,Y(i,:)+h.*k2'/2); % f(x+h/2;y+h*k2/2)
   k4=pend(t(i)+h,Y(i,:)+h.*k3');     % f(x+h;y+h*k3)
   Y(i+1,:)=Y(i,:)+h*(k1'+2*k2'+2*k3'+k4')/6;  % nuovo valore della funzione

   x=L*sin(Y(i,1)); y=-L*cos(Y(i,1));
   plot(x,y,'o',[0,x],[0,y],'-k');
   grid on
   title(sprintf('Tempo [s]: %9.3f',t(i)));
   axis([-1.5, 1.5, -1.5, 1.5]);
   pause(0.0001);
end

[tt,YY]=ode45(@pend,[0,T],[Y(1,1),Y(1,2)]);

figure
plot(t,Y(:,1),'b',tt,YY(:,1),'r')
xlabel('t [s]'),ylabel('teta [rad]'),grid on
legend('soluzione RK4','soluzione ODE45')

end
%-------------------------------------------------------------
function ydot=pend(t,y)
global m g L C
ydot=[y(2);-C/m/L/L*y(2)-(g/L)*sin(y(1))];
end
```

4.3 Vibrazioni (IVP)

Si analizzi con il MATLAB il movimento di una massa *m* attaccata ad una molla in presenza di un fattore di smorzamento.

```
function vibrazioni
% Moto di una massa puntiforme soggetta all'azione di un amolla
% e di uno smorzatore viscoso

clc, clear, close all

global k m c

% Input parameters
k = 1; m = 1; c = 0.5; % c=coefficiente di smorzamento viscoso

omegan = sqrt(k/m); % pulsazione naturale
ccr = 2 * m * omegan;   % smorzamento critico
zeta = c / ccr      % fattore di smorzamento

x0 = 1;  % posizione iniziale della massa m
v0 = 1;  % velocità iniziale della massa m
[t,y] = ode45(@der,[0 20],[x0 v0]);

% soluzione analitica
tt = linspace(0,20,10000);
omegas = omegan * sqrt(1 - zeta * zeta);

if zeta<1      % smorzamento sottocritico
   xx = exp(-
zeta*omegan*tt).*(x0*cos(omegas*tt)+((v0+zeta*omegan*x0)/omegas)*sin(omegas*tt));
elseif zeta==1 % smorzamento critico
   xx = (x0+(v0+omegan*x0)*tt).*exp(-omegan*tt);
else           % smorzamento sovracritico
   a1 = omegan*(zeta+sqrt(zeta*zeta-1));
   a2 = omegan*(-zeta+sqrt(zeta*zeta-1));
   b1 = (x0*a1+v0)/2/omegan/sqrt(zeta*zeta-1);
   b2 = (x0*a2-v0)/2/omegan/sqrt(zeta*zeta-1);
   xx = b1*exp(a2*tt)+b2*exp(-a1*tt);
end

plot(t,y(:,1),'ro',tt,xx,'b');
xlabel('t'); ylabel('x');
legend('soluzione numerica','soluzione analitica');

end
%-------------------------------------------------------%
function dydt = der(t,y)

global k m c
dydt(1) = y(2);                          % derivata della posizione x
dydt(2) = - k / m * y(1) - c / m * y(2); % derivata della velocità v
dydt = [dydt(1),dydt(2)];

end
```

4.4 Velocità di caduta del paracadutista (IVP)

Un paracadutista di massa 70 kg si lancia da un aereo. Calcolare la velocità di caduta prima dell'apertura del paracadute sapendo che la resistenza aerodinamica C è uguale a circa 12.5 kg/s.

Dalla seconda legge di Newton si ha:

$$F = m\frac{dw}{dt}$$

Per un corpo in caduta libera in prossimità della superficie terrestre, la forza risultante è composta da due forze contrapposte: la forza peso diretta verso il basso, F_G,e la forza resistente dell'aria diretta verso l'alto, F_V:

$$F_{g,z} + F_{V,z} = m\frac{dw}{dt} \Rightarrow m\,g + F_{V,z} = m\frac{dw}{dt}$$

Della resistenza dell'aria si può tener conto considerando la forza proporzionale alla velocità:

$$F_{V,z} = -C\,w$$

dove C è una costante di proporzionalità chiamata **resistenza aerodinamica**. All'aumentare della velocità di caduta aumenta quindi anche la forza diretta verso l'alto dovuta alla resistenza dell'aria.

Si ottiene la seguente equazione differenziale del primo ordine non omogenea:

$$\frac{dw}{dt} = -\frac{C}{m}w + g$$

Se il paracadutista è inizialmente immobile (w=0 per t=0), la soluzione analitica è la seguente:

$$w(t) = \frac{g\,m}{C}\left[1 - e^{-(C/m)t}\right]$$

Si noti che la velocità aumenta con il tempo e tende alla velocità limite $w_\infty = g\,m/C$ che in questo caso vale: 54.94 m/s. Nel seguito è riportato il file di funzione MATLAB mediante il quale è stato risolto numericamente il suddetto problema sia utilizzando la funzione intrinseca ODE45, sia utilizzando il metodo di eulero esplicito.

```
function gravity
% funzione che risolve il problema del paracadutista

clc, clear, close all

tspan = [0 30]  % Intervallo di integrazione
y0 = 0;         % condizione iniziale
[t,y] = ode45(@der,tspan,y0);       % soluzione numerica funzione ODE

t1=linspace(0,30,1000);
y1=9.81*70/12.5*(1-exp(-12.5/70.*t1)); % soluzione analitica esatta

[t2,y2] = eulero(@der,tspan,y0);      % soluzione numerica Eulero esplicito

plot(t1,y1,'r',t,y,'ob',t2,y2,'og'),xlabel('t [s]'),ylabel('w [m/s]');
title('Andamento della velocità di un paracadutista in caduta libera');
```

```
legend('soluzione analitica','ODE45','Eulero');
grid

end
%--------------------------------------%
function dydt = der(t,y)

C=12.5;
m=70;
g=9.81;
dydt=-C/m*y+g;

end
%--------------------------------------%
function [x,y] = eulero(f,tspan,y0)
% Soluzione numerica con Eulero esplicito

h=input('Inserisci il passo \n');
a = tspan(1);
b = tspan(2);
y(1) = y0;
x(1) = a;
for i=1:round((b-a)/h)
    x(i+1) = x(i) + h;
    y(i+1) = y(i) + f(x(i),y(i)) * h;
end

end
```

4.5 Picco di avvelenamento da xeno (IVP)

A seguito dello spegnimento istantaneo (a t=0) di un reattore nucleare le leggi di variazione delle concentrazioni dello Xe^{135} e dello I^{135}, il primo dei quali è detto "veleno" perché cattura molti neutroni, sono descritte dal seguente sistema di equazioni differenziali

$$\begin{cases} \dfrac{dX}{dt} = -\lambda_X X + \lambda_I I \\ \dfrac{dI}{dt} = -\lambda_I I \end{cases}$$

a cui sono imposte le seguenti condizioni iniziali:

$$X(t=0) = X_0 = \frac{(\gamma_X + \gamma_I)\bar{\Sigma}_f \phi_0}{\lambda_X + \sigma_X \phi_0}$$

$$I(t=0) = I_0 = \frac{\gamma_I \bar{\Sigma}_f \phi_0}{\lambda_I}$$

I simboli utilizzati assumono il seguente significato e/o valore:

X = numero di atomi di Xe^{135} presenti per unità di volume al tempo t;

I = numero di atomi di I^{135} presenti per unità di volume al tempo t;

$\lambda_X = 2.09 \cdot 10^{-5}$ s^{-1} = costante di decadimento dello Xe^{135};

$\lambda_I = 2.87 \cdot 10^{-5}$ s^{-1} = costante di decadimento dello I^{135};

$\gamma_X = 0.003$ = tasso di produzione dello Xe^{135} come prodotto diretto della fissione;

$\gamma_I = 0.061$ = tasso di produzione dello I^{135} come prodotto diretto della fissione;

$\sigma_X = 3 \cdot 10^{-18}$ cm^2 = sezione d'urto microscopica di assorbimento dei neutroni termici da parte dello Xe^{135};

$\phi_0 = 5 \cdot 10^{13}$ neutroni/(cm^2 s) = flusso medio dei neutroni termici prima dello spegnimento del reattore;

$\Sigma_f = 0.1154$ cm^{-1} = sezione d'urto macroscopica media del combustibile.

- Risolvere il precedente sistema di equazioni differenziali nell'intervallo di tempo [0; 100 ore] facendo uso della funzione `ode45`.
- Riportare su di un grafico (figura 1) l'andamento della concentrazione di Xe^{135} in funzione del tempo, così come determinato dalla funzione `ode45` confrontandola con quella determinata mediante la seguente soluzione analitica:

$$X(t) = X_0 e^{-\lambda_X t} + \frac{\lambda_I I_0}{\lambda_I - \lambda_X}\left(e^{-\lambda_X t} - e^{-\lambda_I t}\right)$$

- Riportare su di un grafico (figura2) l'andamento della concentrazione di I^{135} in funzione del tempo, così come determinato dalla funzione ode45 confrontandola con quella determinata mediante la seguente soluzione analitica:

$$I(t) = I_0 e^{-\lambda_I t}$$

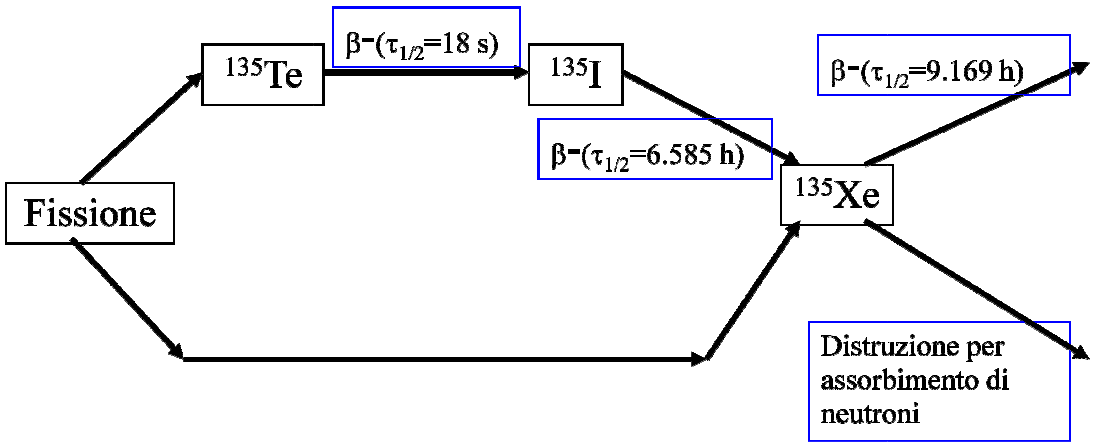
- (*FACOLTATIVO*) Determinare in modo approssimato il tempo in corrispondenza del quale la concentrazione di Xe^{135} raggiunge il suo valore massimo.

Svolgimento

Lo Xe^{135} è un prodotto di fissione la cui concentrazione in condizioni di funzionamento stazionario raggiunge un valore asintotico dipendente dal valore del flusso neutronico. Ciò è dovuto al fatto che lo Xe^{135} viene con continuità prodotto e distrutto. La produzione avviene attraverso le fissioni ed il decadimento beta del suo precursore (lo I^{13}), mentre la distruzione avviene attraverso il suo decadimento radioattivo e l'assorbimento di neutroni per formare l'isotopo Xe^{136}. Lo Xe^{135} ha quindi due modi per essere prodotto e due per essere distrutto. Lo I^{135}ha invece un modo per essere prodotto ed uno per essere distrutto (vedi figura).

Nel caso generale di transitorio senza azzeramento del flusso neutronico le due equazioni differenziali che regolano il fenomeno sono le seguenti:

$$\begin{cases} \dfrac{dX}{dt} = -\lambda_X X - \sigma_X X \phi + \lambda_I I + \gamma_X \Sigma_f \phi \\ \dfrac{dI}{dt} = -\lambda_I I + \gamma_I \Sigma_f \phi \end{cases}$$




```
function Xenon
% funzione per calcolare il picco di avvelenamento da Xeno

clc, clear, close all

global lambdaX lambdaI

lambdaX = 2.09e-5; lambdaI = 2.87e-5;
gammaX = 0.003; gammaI = 0.061;
sigmaX = 3e-18;
fi0=5e13;
Sigmaf=0.1154;

% Condizioni iniziali
X0 = (gammaX + gammaI) / (lambdaX + sigmaX * fi0) * Sigmaf * fi0
I0 = gammaI / lambdaI * Sigmaf * fi0

tspan = [0, 360000]; % intervallo di integrazione (fino a 100 ore)
y0 = [X0, I0];       % condizioni iniziali
[t,y] = ode45(@fun,tspan,y0); % soluzione numerica


rho=sigmaX*y(:,1)/Sigmaf/2.44;
% soluzione analitica approssimata
t1 = linspace(tspan(1),tspan(2),1000);
y1 = X0*exp(-lambdaX*t1)+lambdaI*I0/(lambdaI-lambdaX)*(exp(-lambdaX*t1)-exp(-
lambdaI*t1));
y2 = I0*exp(-lambdaI*t1);
plot(t/3600,y(:,1),'o',t1/3600,y1); grid,
xlabel('Time after shutdown [hr]'),ylabel('Xenon concentration [nucleous/cm3]')
legend('numerical solution', 'analytical solution');
figure
plot(t/3600,rho*100,'o');
xlabel('Time after shutdown [hr]'),ylabel('Negative reactivity [%]'),
grid

end
%--------------------------------------%
function dydt=fun(t,y)

global lambdaX lambdaI
dydt=[-lambdaX*y(1)+lambdaI*y(2);-lambdaI*y(2)];

end
```

4.6 “Lavaggio” con anidride carbonica di un locale contenente aria (IVP)

Si abbia un locale avente un volume libero V di 1000 m^3 contenente inizialmente aria a pressione atmosferica e a temperatura di 293 K. La stanza è dotata di una bocca di ingresso e di una di uscita. All'istante iniziale (t = 0 s) inizia l'immissione nel locale di anidride carbonica che entra con una portata costante pari a 3 kg/s ed una temperatura di 200 K. Si ipotizzi che la CO_2 una volta entrata si misceli uniformemente con l'aria, sia da un punto di vista termico che di concentrazione, e che dalla bocca di uscita esca una portata di gas miscelato tale che all'interno del locale la pressione si mantenga costantemente uguale a quella atmosferica.

La soluzione del problema fisico in esame si riconduce alla soluzione del seguente sistema di due equazioni differenziali:

$$\begin{cases} \dfrac{dT}{dt} = \dfrac{W_{CO_2,i}}{M_{tot}} \dfrac{c_{p,CO_2}\left(T_i - T\right)}{\left[\omega\, c_{p,CO_2} + \left(1-\omega\right) c_{p,air}\right]} \\ \dfrac{d\omega}{dt} = \dfrac{W_{CO_2,i}}{M_{tot}}\left(1-\omega\right) \end{cases}$$

in cui T è il valore della temperatura della miscela all'istante temporale t, $\omega \equiv M_{CO_2} / M_{tot}$ è la frazione massica di CO_2 nella miscela gassosa aria-CO_2 presente nel locale di volume V all'istante t, $W_{CO2,i}$ è la portata di CO_2 in entrata al locale, T_i è la temperatura della CO_2 in entrata nel locale, mentre la massa totale M_{tot} di miscela gassosa contenuta nel locale di volume V è da calcolarsi mediante la seguente relazione:

$$M_{tot} = \frac{1}{\left[\left(1-\omega\right) R_{air} + \omega\, R_{CO_2}\right]} \frac{p_{atm} V}{T}$$

Le altre grandezze termodinamiche che compaiono nelle precedenti formule assumono il seguente valore:

R_{air} = 287.04 J/(kg K); R_{CO2} = 188.92 J/(kg K);

$c_{p,CO2}$ = 800 J/(kg K); $c_{p,air}$ = 1010 J/(kg K); p_{atm} = 101325 Pa

Si risolva numericamente il sistema di equazioni differenziali precedente per l'intervallo temporale [0, 100 s] applicando le opportune condizioni iniziali. Si realizzi, infine, un diagramma multiplo composto da 4 grafici. Il primo grafico che mostri l'andamento temporale della temperatura della miscela gassosa all'interno del volume V, il secondo l'andamento temporale della variabile ω, il terzo l'andamento temporale della massa di CO_2 contenuta nel volume V ed il quarto l'andamento temporale della massa di aria sempre nel volume V.

```
function CO2
% "Lavaggio" con anidride carbonica di un locale contenente aria

clc, clear, close all

global Rair Rco2 Win Tin V cpco2 cpair patm
```

```
Rair = 8314.472 / 28.966 ;
Rco2 = 8314.472 / 44.01 ;
patm = 101325;
V = 1000;      % Volume della stanza

Win = 3.0;     % Portata massica di CO2 in ingresso
Tin = 200.00; % Temperatura della CO2 in ingresso
cpco2 = 800;  % Calore specifico medio della CO2 nella stanza
cpair = 1010; % Calore specifico medio dell'aria nella stanza

% Condizioni iniziali
y0 = [293.15,0.0]; % condizioni iniziali (temperatura ambiente e tutt'aria)

tspan = [0,100];              % intervallo di integrazione (fino a 100 s)
[t,y] = ode45(@fun,tspan,y0); % soluzione numerica esatta

subplot(2,2,1);
plot(t,y(:,1));
xlabel('Tempo [s]'),ylabel('Temperatura [°C]'), grid

subplot(2,2,2);
plot(t,y(:,2));
xlabel('Tempo [s]'),ylabel('Omega [-]'), grid

Mtotale = patm * V ./ y(:,1) ./ ((1.-y(:,2)) .* Rair + y(:,2) * Rco2)

subplot(2,2,3);
plot(t,Mtotale.*y(:,2));
xlabel('Tempo [s]'),ylabel('Massa CO2 [kg]'), grid

subplot(2,2,4);
plot(t,Mtotale.*(1-y(:,2)));
xlabel('Tempo [s]'),ylabel('Massa aria [kg]'), grid

end
%-------------------------------------------------------------
function dydt=fun(t,y)

global Rair Rco2 Win Tin V cpco2 cpair patm
Mtot=1/((1-y(2))*Rair+y(2)*Rco2)*patm*V/y(1);
dydt=[Win*cpco2*(Tin-y(1))/(Mtot*(y(2)*cpco2+(1-y(2))*cpair));Win/Mtot*(1-y(2))];

end
```

4.7 Equazione di Van der Pol (IVP)

Si consideri la seguente equazione differenziale:

$$\frac{d^2y}{dt^2} - m\left(1-y^2\right)\frac{dy}{dt} + y = 0$$

dove *m* è un numero reale positivo che rappresenta lo smorzamento del sistema. Considerato *m* pari a 1 si risolva numericamente l'equazione differenziale precedente nell'intervallo di tempo [0, 30 s] che soddisfi alle condizioni iniziali:

$$\begin{cases} y(0) = 5 \\ \dfrac{dy}{dt}(0) = 0 \end{cases}$$

Si realizzi un diagramma multiplo composto da 2 grafici. Il primo grafico che mostri l'andamento di *y* in funzione di *t* ed il secondo in cui sia riportato l'andamento di *dy/dt* in funzione di *y*.
Si valuti, infine, l'istante temporale in cui la funzione *y*(*t*) esibisce il suo primo minimo e si trovi il valore assunto dalla funzione *y* in corrispondenza di tale istante.

```
function Vander
% Risolve l'quazione differenziale di Van der Pol

clc, clear, close all

tspan = [0, 30];
y0 = [5; 0];
[t,y] = ode45(@myode, tspan, y0);

[ymin,i] = min(y(:,1))
t(i)

% Plot of the solution
subplot(1,2,1);
plot(t,y(:,1)),xlabel('t'),ylabel('solution y'),grid
title('van der Pol Equation, m = 1')
%
subplot(1,2,2);
plot(y(:,1),y(:,2)),xlabel('y'),ylabel('dy/dt'),grid
title('Van der Pol equation, m = 1')

end
%-------------------------------------------------------------
function ydot=myode(t,y)

m=1;
ydot=[y(2);m*(1-y(1)^2)*y(2)-y(1)];

end
```

4.8 Riscaldamento di un bagno di olio (IVP)

In un bagno di olio è presente una sorgente di calore la cui potenza è funzione del tempo. Il bagno è miscelato in modo che la temperatura dell'olio si può considerare uniforme. Il bilancio di energia del bagno fornisce l'equazione differenziale:

$$M\,c_p\,\frac{dT}{dt} = -A\,h\left(T - T_a\right) + \dot{Q}\left(t\right)$$

dove:

M = massa di olio = 10 kg;

c_p = calore specifico dell'olio = 2000 J/(kg K);

A = area di scambio termico con l'ambiente esterno del recipiente che contiene l'olio = 2 m^2;

H = conduttanza relativa allo scambio termico dell'olio con l'esterno = 20 W/(m^2 K);

T_a = temperatura dell'aria esterna = 20 °C;

$\dot{Q}\left(t\right) = C \sin \omega t$, dove C = 10000 W e ω = 0.005 s^{-1}.

Si risolva la suddetta equazione differenziale sull'intervallo di tempo [0, 200 s] utilizzando la funzione `ode45`, con l'ipotesi che la temperatura iniziale dell'olio sia uguale alla temperatura esterna.

Si riporti su di un grafico l'andamento temporale della temperatura dell'olio.

Si calcoli infine il tempo necessario affinché la temperatura dell'olio raggiunga 50 °C.

```
function oil
% Riscaldamento di un bagno di olio
clc
clear
close all

global M cp A H Ta C w

M = 10;       % Massa di olio
cp = 2000;    % Calore specifico dell'olio
A = 2;        % Area di scambio
H = 20;       % Conduttanza termica
Ta = 20;      % Temperatura aria
C = 10000;    % Costante
w = 0.005;    % Costante

% Condizioni iniziali
tspan = [0, 200]; % intervallo di integrazione
y0 = 20;          % condizione iniziale
[t,y] = ode45(@fun,tspan,y0); % soluzione numerica

plot(t,y(:,1),'o');
xlabel('Time [s]'),ylabel('temperature [°C]')

%-------------------------------------------------------------
function dydt=fun(t,y)
global M cp A H Ta C w
dydt = ( - A * H * ( y - Ta ) + C * sin(w*t)) / M / cp;
```

4.9 Variazione del flusso neutronico in caso di inserzione positiva di reattività (IVP)

Nel caso di un solo gruppo di neutroni ritardati, avente una costante di decadimento $\lambda = 0.08\ \text{s}^{-1}$ ed una frazione di neutroni ritardati di $\beta = 0.0065$, il modello di cinetica del reattore è descritto dal sistema di equazioni differenziali

$$\begin{cases} \dfrac{dn}{dt} = k\left(\rho - \beta\right)\dfrac{n}{l_p} + \lambda C \\ \dfrac{dC}{dt} = k\dfrac{\beta n}{l_p} - \lambda C \end{cases}$$

a cui sono imposte le seguenti condizioni iniziali:

$$n\left(t=0\right) = n_0 = 2.272 \cdot 10^{14}\ \text{neutroni/m}^3$$

$$C\left(t=0\right) = C_0 = \frac{\beta}{\lambda l_p} n_0$$

I simboli utilizzati in precedenza assumono il seguente significato e/o valore:

n = densità neutronica (numero di neutroni per unità di volume, al tempo t);

C = concentrazione dei precursori dei neutroni ritardati, al tempo t;

$l_p = 1 \bullet 10^{-4}$ s = vita media dei neutroni pronti;

k = fattore di moltiplicazione effettivo;

$\rho = (k - 1) / k$ = reattività.

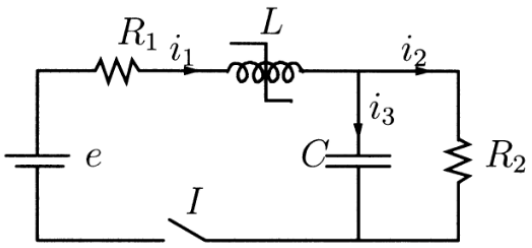
- Risolvere il precedente sistema di equazioni differenziali nell'intervallo di tempo [0; 1 s] nel caso che il fattore di moltiplicazione effettivo nell'istante iniziale si porti dal valore unitario a $k = 1.001$, facendo uso della funzione `ode45`.
- Riportare su di un grafico l'andamento della densità neutronica adimensionalizzata (n/n_0) in funzione del tempo, così come determinato dalla funzione `ode45` confrontandola con quella determinata analiticamente mediante la seguente formula approssimata:

$$n\left(t\right)/n_0 = \left[\frac{\beta}{\beta-\rho} e^{\lambda \rho t/(\beta-\rho)} - \frac{\rho}{\beta-\rho} e^{-(\beta-\rho)t/l_p}\right]$$

```
function cineticar
% Variazione del flusso neutronico in caso di inserzione positiva di reattività
clc
clear
close all

global k rho beta lp lambda

lambda = 0.08;  % costante di decadimento
beta = 0.0065;  % frazione di neutroni ritardati
lp = 1e-4;      % vita media dei neutroni pronti
k=1.0065;       % fattore di moltiplicazione effettiva
rho=(k-1)/k     % reattività

% Condizioni iniziali
n0 = 2.272e14;                  % densità neutronica iniziale
C0 = beta / lambda / lp * n0;   % concentrazione iniziale dei precursori

tspan = [0, 1]; % intervallo di integrazione (fino ad 1 s)
y0 = [n0, C0]; % condizioni iniziali
[t,y] = ode45(@fun,tspan,y0); % soluzione numerica

% soluzione analitica
t1 = linspace(tspan(1),tspan(2),1000);
y1 = beta/(beta-rho)*exp(lambda*rho*t1/(beta-rho))-rho/(beta-rho)*exp(-(beta-
rho)*t1/lp);
plot(t,y(:,1)/n0,t1,y1,'o');
xlabel('Time [s]'),ylabel('n(t)/n0')

%-------------------------------------------------------------
function dydt=fun(t,y)
global k rho beta lp lambda
dydt=[k*(rho-beta)/lp*y(1)+lambda*y(2);k*beta*y(1)/lp-lambda*y(2)];
```

4.10 Circuito elettrico (IVP)

Si consideri il circuito elettrico riportato in figura e si supponga che l'induttanza L sia costante e valga 0.1 Henry, che $R_1 = R_2 = R = 10$ Ohm che il condensatore abbia capacità C = 1e-3 Farad e che la f.e.m. sia $e = 5$ V.



Il circuito precedentemente aperto e scarico viene chiuso all'istante che considereremo istante iniziale. L'andamento temporale della differenza di potenziale ai capi del condensatore può essere ottenuto risolvendo la seguente equazione differenziale del secondo ordine:

$$LC\frac{d^2V}{dt^2}+\left(\frac{L}{R}+RC\right)\frac{dV}{dt}+2V=e$$

Si risolva con il MATLAB la suddetta equazione differenziale nell'intervallo [0, 0.1 s] e si realizzi un diagramma multiplo nel quale si riporti i grafici degli andamenti temporali di V e di $\frac{dV}{dt}$. Si calcoli i valori massimi di V e di $\frac{dV}{dt}$ e gli istanti temporali in cui si hanno tali valori massimi; scrivere i suddetti valori nel file output.txt.

```
function circuitoel
clc
clear
close all

global L R C e

L = 0.1;  % Induttanza [Henry]
R = 10.0; % Resistenza [Ohm]
C = 1E-3; % Capacita' [Farad]
e = 5.0;  % F.e.m. [Volt]

tspan = [0, 0.1]; % intervallo di integrazione (fino a 1000 s)
y0 = [0, 0]; % condizioni iniziali
[t,y] = ode45(@fun,tspan,y0); % soluzione numerica esatta

%Visualizzazione risultati
subplot(1,2,1)
plot(t,y(:,1));
xlabel('Tempo [s]'),ylabel('Tensione [V]'),grid
subplot(1,2,2)
plot(t,y(:,2));
xlabel('Tempo [s]'),ylabel('dV/dt [V/s]'),grid

[a,i] = max(y(:,1))
tmax_V = t(i)

[a,i] = max(y(:,2))
tmax_dVdt = t(i)

end

%-------------------------------------------------------------
function dydt=fun(t,y)
global L R C e
dydt=[y(2);(e - 2 * y(1) - (L/R+R*C) * y(2)) / L /C];
end
```

4.11 Problema sulle equazioni differenziali ordinarie di tipo IVP

Si risolva numericamente, mediante l'uso del MATLAB, il seguente sistema di equazioni differenziali:

$$\begin{cases} \dfrac{d\,y_1}{d\,t} = -y_2 - \dfrac{y_1\,y_3}{\sqrt{y_1^2 + y_2^2}} \\ \dfrac{d\,y_2}{d\,t} = y_1 - \dfrac{y_2\,y_3}{\sqrt{y_1^2 + y_2^2}} \\ \dfrac{d\,y_3}{d\,t} = \dfrac{y_1}{\sqrt{y_1^2 + y_2^2}} \end{cases}$$

nell'intervallo temporale [0, 10 s] con le seguenti condizioni iniziali:

$$y_1(0) = 2; \quad y_2(0) = 0; \quad y_3(0) = 0$$

Si realizzi un diagramma tridimensionale a linea utilizzando il comando "plot3" avente x, y e z come parametri di input. Si disegni, inoltre su di un'altra figura un diagramma bidimensionale che riporti la proiezione della linea sul piano di equazione $z = 0$.

Determinare, infine, le coordinate del punto P = (y_1, y_2, y_3) in corrispondenza dell'istante t = 10 s.

```
function esame13
% Problema sulle equazioni differenziali ordinarie di tipo IVP
clc
clear
close all
format long

[t,Y]=ode23(@myode,[0,2],[2,0,0]);

plot3(Y(:,1),Y(:,2),Y(:,3)),grid
xlabel('X'),ylabel('Y'),zlabel('Z')

figure;
plot(Y(:,1),Y(:,2)),grid
N = length(Y(:,1));
P = [Y(N,1),Y(N,2),Y(N,3)]
%-------------------------------------------------------
function ydot=myode(t,y)
r = sqrt(y(1)*y(1)+y(2)*y(2));
ydot=[-y(2)-y(1)*y(3)/r;y(1)-y(2)*y(3)/r;y(1)/r];
```

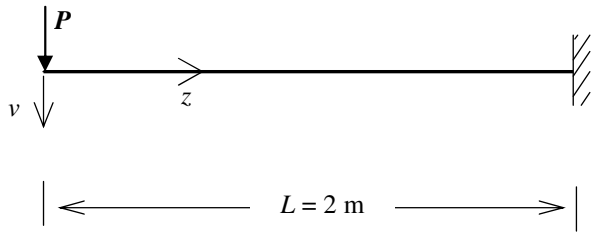
4.12 Trave a mensola caricata all'estremità libera (BVP)

Una trave, avente una luce di 2 m ed una sezione trasversale con momento d'inerzia di 1.0E-4 m^4, è incastrata ad una estremità ed è caricata altra estremità con un carico concentrato verticale di intensità pari a 20000 N (vedi figura).

Si determini l'andamento degli spostamenti trasversali subiti dai punti dell'asse della trave risolvendo numericamente l'equazione differenziale della linea elastica.

$$\frac{d^2v}{dz^2} = -\frac{M(z)}{EJ}$$

dove v rappresenta lo spostamento trasversale dell'asse della trave, mentre dv/dz rappresenta la rotazione della generica sezione. Il modulo di elasticità della trave risulta pari ad $E = 2*10^{11}$ Pa.

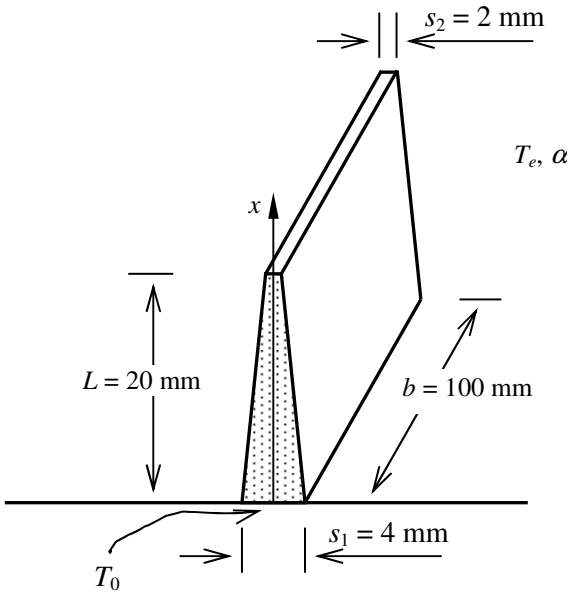



```
function mybvp
% Calcolo della linea elastica di una trave a mensola, di sezione
% costante, soggetta ad un carico concentrato P
clc

global P E J

L = 2.0;       % lunghezza della trave [m]
P = 20000;     % Carico concentrato [N]
E = 2.e11;     % Modulo di elasticità normale [Pa]
J = 1e-4 ;     % Momento di inerzia della sezione [m^4]

solinit = bvpinit(linspace(0,L,10),[0;0]);
sol=bvp4c(@myode,@mybc,solinit);
% sol è un array di strutture

xint = linspace(0,L,100);
% Sxint = deval(sol,xint);

% soluzione analitica
v = (P*xint.^3/6-P*L*L*xint/2+P*L^3/3)/E/J;

plot(sol.x,sol.y(1,:),'or',xint,v);
legend('Soluzione numerica','soluzione analitica');
xlabel('z [m]'),ylabel('v(z) [m]')

% --------------------------------------------------------------
function dydx = myode(x,y)
global P E J
```

```
dydx = [y(2); P*x/E/J];

% ------------------------------------------------------------
function res = mybc(ya,yb)
res = [yb(1); yb(2)];
```

4.13 Aletta di raffreddamento (BVP)

Sulla parete esterna di una piastra riscaldata internamente sono presenti alette di raffreddamento con profilo a forma di trapezio isoscele (vedi figura). Le alette sono realizzate in alluminio avente una conducibilità termica k = 170 W/(m K), mentre il coefficiente di scambio termico convettivo medio tra la superficie dell'aletta e la corrente di aria esterna è $\alpha = 20$ W/m^2.



L'equazione differenziale necessaria a risolvere il suddetto problema è:

$$A(x)\frac{d^2\theta}{dx^2}+\left(\frac{dA}{dx}\right)\left(\frac{d\theta}{dx}\right)-\frac{\alpha P(x)}{k}\theta=0, \quad \theta(x)\equiv T(x)-T_e$$

dove $A(x)$ e $P(x)$ rappresentano, rispettivamente, l'area ed il perimetro della sezione dell'aletta alla generica quota x. Alla precedente equazione differenziale vanno imposte le seguenti condizioni al contorno:

$$\begin{cases}\theta(x=0)=T_0-T_e\\ \left.\dfrac{d\theta}{dx}\right|_{x=L}=0\end{cases}$$

Supponendo che la temperatura alla base dell'aletta, T_0, sia pari a 80 °C e quella dell'aria esterna, T_e, sia uguale a 25 °C, si calcoli l'andamento della temperatura e del flusso termico conduttivo lungo l'asse x dell'aletta. Si realizzi un diagramma multiplo nel quale si riporti sia il grafico dell'andamento della temperatura che quello dell'andamento del flusso termico dentro l'aletta in funzione di x.

Si calcoli la temperatura sulla sommità dell'aletta e l'*efficienza dell'aletta*, η, cioè il rapporto tra la potenza termica effettivamente scambiata dall'intera aletta e quella che verrebbe scambiata se l'intera superficie dell'aletta si trovasse alla temperatura T_0.

A scopo semplificativo, si osservi che $P(x) \cong 2\,b$, mentre:

$$A(x) = b \cdot s(x) = b \cdot \left[s1 - (s1 - s2)\frac{x}{L} \right], \qquad \frac{dA}{dx} = -\frac{b(s1 - s2)}{L}$$

```
function aletta
% Aletta di raffreddamento
clc
clear
close all

global k a L b s1 s2 T0 Te
k=170;     %Conducibilita' Termica [W/(m K)]
a=20;      %Coefficiente di scambio convettivo [W/(m^2 K)]
L=0.02;    %Altezza aletta [m]
b=0.1;     %Lunghezza aletta [m]
s1=0.004; %Spessore base aletta [m]
s2=0.002; %Spessore estremita' aletta [m]
T0=80;     %Temperatura base [C]
Te=25;     %Temperatura esterna [C]

%Risoluzione Equazione Differenziale
solinit=bvpinit(linspace(0,L,100),[80 0]);
sol=bvp4c(@myode,@mybc,solinit);

%Visualizzazione risultati
subplot(1,2,1)
plot(sol.x*1000,sol.y(1,:)+Te,'r')
xlabel('x [mm]')
ylabel('T [C]')
grid
subplot(1,2,2)
plot(sol.x*1000,-sol.y(2,:)*k,'b')
xlabel('x [mm]')
ylabel('flusso termico [W/m^2]')
grid

aa=length(sol.x);
disp('Temperatura alla estremita: ')
T=sol.y(1,aa)+25;
disp(T)

%Calcolo efficienza dell'aletta
Qaletta=-k*sol.y(2,1)*(s1*b);
l=sqrt((s1/2-s2/2)^2+L^2);
Al=l*b;
Al2=s2*L+(s1/2-s2/2)*L;
Q1=a*(T0-Te)*(2*Al+2*Al2);
eta=Qaletta/Q1;
disp('Efficienza aletta')
disp(eta)

%-------------------------------------------------------------------------
function dydx=myode(x,y)
global k a L b s1 s2 T0 Te
dydx=[y(2);1/(s1-(s1-s2)*x/L)*((s1-s2)/L*y(2)+2*a/k*y(1))];
```

```
%--------------------------------------------------------------------------
function res=mybc(ya,yb)
global k a L b s1 s2 T0 Te
res=[ya(1)-(T0-Te);yb(2)];
```

4.14 Moto di un proiettile (BVP)

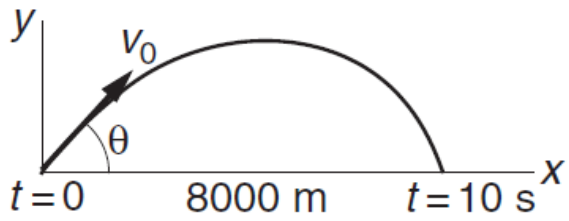
Un proiettile di massa m = 2 kg viene lanciato da un cannone con velocità v_0 ed angolo di inclinazione rispetto al piano orizzontale pari a θ. Considerando che il proiettile si muove sotto l'azione della forza di gravità (g = 9.81 m/s^2) e della forza aerodinamica dell'aria (C = 2.0E-04 kg/m), il moto del proiettile (v. figura) è regolato dalle seguenti due equazioni differenziali:

$$\ddot{x} = -\frac{C}{m} v \dot{x}$$

$$\ddot{y} = -\frac{C}{m} v \dot{y} - g$$

dove:

$$v \equiv \sqrt{\dot{x}^2 + \dot{y}^2}$$



Il proiettile colpisce il bersaglio, posto sullo stesso piano orizzontale del punto di lancio, ad una distanza di 8 km impiegando un tempo di volo di 10 s.

Si risolva il suddetto problema mediante l'uso del MATLAB.

Si realizzi un grafico nel quale si riporti la traiettoria del proiettile, cioè y in funzione di x. Si calcoli la velocità di lancio v_0 e il suo angolo di inclinazione θ. Si valuti infine la massima altezza raggiunta dal proiettile dal piano orizzontale di lancio.

```
function bvpesame
% Esame di Elementi di informatica applicata del 16-05-07

Clc, clear, close all

global m g C
% dati di input
m = 2;
g = 9.81;
C = 3e-04;

solinit = bvpinit(linspace(0,10,100),[0 0 0 0]); % initial guess of the solution
sol = bvp4c(@myode,@mybc,solinit);

plot(sol.y(1,:),sol.y(3,:),'o-')
v0=sqrt(sol.y(2,1)^2+sol.y(4,1)^2)
teta = atan(sol.y(4,1)/sol.y(2,1))*180/pi
[ymax,imax] = max(sol.y(3,:))

% --------------------------------------------------------------------------
function dydx = myode(x,y)
%ODE function for the Example of the BVP
global m g C
v = sqrt(y(2)*y(2)+y(4)*y(4));
```

```
dydx = [  y(2) ; -C/m*v*y(2); y(4); -C/m*v*y(4)-g ];
end
% --------------------------------------------------------------------
function res = mybc(ya,yb)
%Boundary conditions for the Example of the BVP
res = [ya(1) ; yb(1)-6000; ya(3); yb(3)];
end
```

4.15 Diffusione attraverso un film gassoso stagnante (BVP)

Si consideri il "tubo di Stefan" riportato in figura, cioè un canale nel quale si ha la diffusione di una specie gassosa A (vapore) all'interno di una miscela gassosa contenente sia la specie A che una specie "incondensabile" B (aria).

Il liquido A mantenuto a temperatura costante evapora all'interfaccia liquido-gas, mentre che il livello del liquido viene mantenuto sempre nella stessa posizione (stessa quota), che fisseremo come $z = 0$. All'esterno del tubo vi è una corrente gassosa che mantiene sulla sommità del tubo stesso (z = h = 0.2 m) una determinata concentrazione molare X della specie A.

In questo tubo viene a stabilirsi una diffusione di A attraverso B in un regime stazionario governata dalla seguente equazione differenziale:

$$\frac{d}{dz}\left(\frac{1}{1-X_A}\frac{dX_A}{dz}\right)=0$$

1. Si risolva numericamente tale equazione mediante l'uso del MATLAB considerando che la frazione molare della specie A all'interfaccia ($X_A(z=0)$) è pari all'80%, mentre quella nella corrente gassosa esterna ($X_A(z=h)$) è pari al 20%.
2. Si realizzi un grafico che mostri l'andamento di X_A e di X_B in funzione di z (si noti che $X_A+X_B=1$).
3. Partendo dai dati ottenuti dalla risoluzione della precedente equazione differenziale si calcoli poi il rateo di evaporazione all'interfaccia attraverso la formula:

$$\dot{m} = -S\,M_A\frac{c\,D_{AB}}{1-X_A}\frac{dX_A}{dz}$$

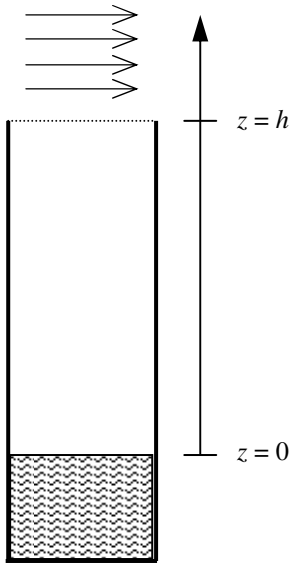
dove:

S = area di interfaccia = 1 m^2

M_A = peso molecolare della specie A = 18 kg/kmole

c = concentrazione della miscela = 0.05 kmole/m^3

D_{AB} = coefficiente di diffusione = 3.5x10^{-5} m^2/s



Si può porre:

$$\frac{d}{dz}\left(\frac{1}{1-X_A}\frac{dX_A}{dz}\right)=0$$

$$y_1 \equiv X_A \qquad y_2 \equiv \frac{1}{1-y_1}\frac{dy_1}{dz}$$

$$\begin{cases} \dfrac{dy_1}{dz} = y_2\left(1-y_1\right) \\ \dfrac{dy_2}{dz} = 0 \end{cases}$$

$$\begin{cases} y_1\left(z=0\right) = 0.8 \\ y_1\left(z=h\right) = 0.2 \end{cases}$$

```
function bvpevaporazione
% Diffusione attraverso un film gassoso stagnante
clc
close all

% dati di input
S = 1;
MA = 18;
c = 0.05;
DAB=3.5e-5;
h=0.2;

options = bvpset('stats','on');
solinit = bvpinit(linspace(0,h,100),[0.5; 0]); % initial guess of the solution
sol1 = bvp4c(@myode,@mybc,solinit,options);

fprintf('\n');

% soluzione analitica
z=linspace(0,h,200);
XA=1-0.2.*exp(log(4).*z./h);
plot(sol1.x,sol1.y(1,:),'o-',sol1.x,1-sol1.y(1,:),'o-r')
% plot(sol1.x,sol1.y(1,:),'o',z,XA,'-o')
xlabel('z [m]')
ylabel('XA [-]')

mp= - S*MA*c*DAB*sol1.y(2,1)

figure
% plot(sol1.x,sol1.y(2,:)./(1-sol1.y(1,:)),'o-')
xlabel('z [m]')
ylabel('m [kg/s]')

% --------------------------------------------------------------------------
function dydx = myode(x,y)
% ODE function for the Example of the BVP
dydx = [  y(2) * ( 1 - y(1) ); 0 ];

% --------------------------------------------------------------------------
function res = mybc(ya,yb)
% Boundary conditions for the Example of the BVP
res = [ya(1)-0.8; yb(1)-0.2];
```

4.16 Diffusione di una sostanza con contemporanea reazione chimica (BVP)

Un problema stazionario di determinazione dell'andamento della concentrazione di una sostanza che subisce diffusione e contemporanea reazione chimica in un dominio sferico è riconducibile alla seguente equazione differenziale:

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{dc}{dr}\right)=\frac{\alpha c}{1+k c}$$

doce $k = 1$ ed $\alpha = 5$.

Si tenga presente che nella suddetta equazione:

- il raggio r è adimensionalizzato rispetto al vero raggio del dominio sferico, cioè $r = 1$ significa che ci trova sulla superficie esterna del dominio sferico;
- la concentrazione c è stata adimensionalizzata in modo tale che $c(r = 1) = 1$.

Le condizioni al contorno per l'equazione differenziale suddetta sono ottenibili da quanto riportato precedentemente detto e considerando il fatto che, grazie alla simmetria sferica del problema, la concentrazione in funzione di r dovrà essere simmetrica rispetto ad $r = 0$.

Si risolva numericamente l'equazione precedente mediante l'uso del MATLAB e si realizzi un grafico con due diagrammi: su quello di destra bisognerà riportare l'andamento della concentrazione c in funzione del raggio r e su quello di sinistra l'andamento del gradiente di concentrazione in funzione del raggio r.

Si fornisca il valore della concentrazione in prossimità del centro del dominio sferico.

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{dc}{dr}\right)=\frac{\alpha c}{1+k c}$$

$$y_1 \equiv c\,; \qquad y_2 \equiv \frac{dy_1}{dr}$$

Il sistema equivalente è

$$\begin{cases}\dfrac{dy_1}{dr} = y_2 \\ \dfrac{dy_2}{dr} = \dfrac{\alpha\, y_1}{1+k\, y_1} - \dfrac{2\, y_2}{r}\end{cases}$$

mentre le condizioni al contorno sono:

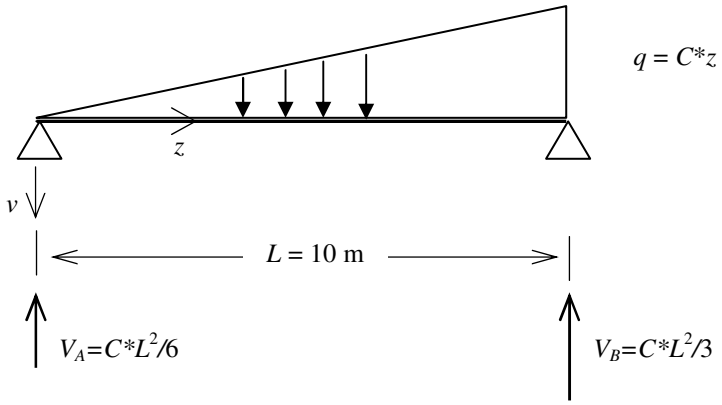
$$\begin{cases} y_1(r=1)=1 \\ y_2(r=0)=0 \end{cases}$$

```
function bvpesame
% Diffusione di una sostanza con contemporanea reazione chimica

clc
close all
```

```
global k alfa
% dati di input
k = 1;
alfa = 5;

options = bvpset('stats','on');
solinit = bvpinit(linspace(0.0001,1,100),[0.9 0.1]); % initial guess of the
solution
sol = bvp4c(@myode,@mybc,solinit,options);

sol.y(1,1)

fprintf('\n');

subplot(1,2,1);
plot(sol.x,sol.y(1,:),'o-')
xlabel('r')
ylabel('c')
subplot(1,2,2);
plot(sol.x,sol.y(2,:),'s-')
xlabel('r')
ylabel('dc/dr')

% --------------------------------------------------------------------------
function dydx = myode(x,y)
%ODE function for the Example of the BVP
global k alfa
dydx = [  y(2) ; alfa*y(1)/(1+k*y(1))-2*y(2)/x ];

% --------------------------------------------------------------------------
function res = mybc(ya,yb)
%Boundary conditions for the Example of the BVP
res = [(yb(1)-1) ; ya(2)];
```

4.17 Trave con carico distribuito in maniera non uniforme (BVP)

Una trave appoggiata alle sue due estremità, avente una luce L = 10 m, è caricata da un carico distribuito con andamento lineare secondo la legge $q = C*z$ dove C = 500 Pa (vedi figura).



L'equazione differenziale della linea elastica necessaria per risolvere il suddetto problema è:

$$\frac{d^2v}{dz^2} = -\frac{M(z)}{EJ}$$

dove v rappresenta lo spostamento trasversale dell'asse della trave, mentre dv/dz rappresenta la rotazione della generica sezione. Il modulo di elasticità della trave risulta pari ad $E = 2*10^{11}$ Pa, mentre il momento d'inerzia della sezione vale $J = 10^{-4}$ m^4.

Note le reazioni vincolari V_A e V_B (vedi figura), si determini l'andamento del momento flettente $M(z)$ e si risolva numericamente l'equazione differenziale precedente applicando le opportune condizioni al contorno.

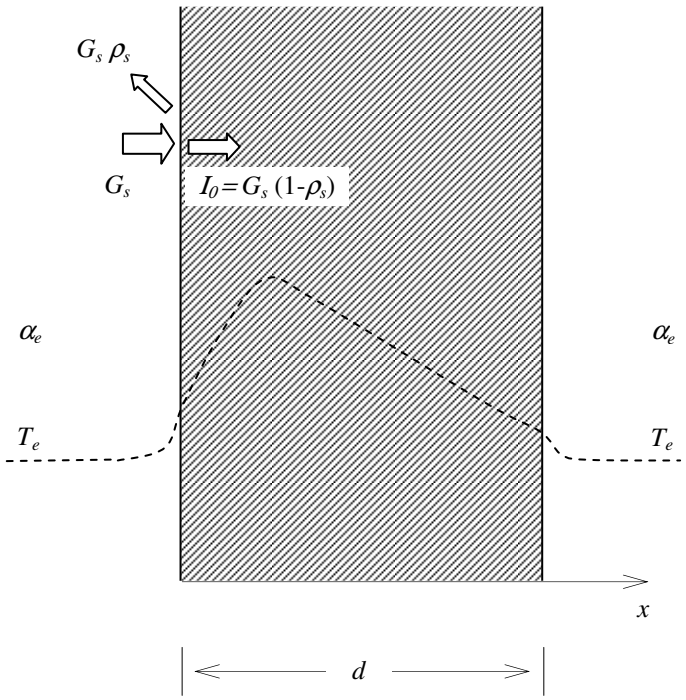
Si valuti il valore della freccia e la posizione z^* in corrispondenza della quale v risulta massima. Si riporti, infine, su di un grafico l'andamento della linea elastica deformata.

```
function trave
% Trave con carico distribuito in maniera non uniforme

clear, clc, close all

global E J L c

% dati di input
L = 10;
E = 2e11;
J = 1e-4;
c = 500;

solinit = bvpinit(linspace(0,10,100),[0;0]);
sol = bvp4c(@myode,@mybc,solinit);

[v i] = max(sol.y(1,:));
v
z = sol.x(i)
```

```
%subplot(1,2,1)
%plot(z,M)

%subplot(1,2,2)
plot(sol.x,sol.y(1,:),'r')

function dydz = myode(z,y)
global E J L c
dydz = [y(2) ; (c*z^3/6-c*L^2/6*z)/E/J];

function res = mybc(ya,yb)
res = [ya(1) ; yb(1)];
```

4.18 Assorbimento della radiazione solare da una lastra di vetro (BVP)

Si consideri una lastra di vetro di spessore d = 5 mm e conducibilità termica λ = 0.5 W/(mK) immersa in aria alla temperatura T_e = 20 °C. La lastra, irraggiata dal sole con un'intensità di irradiazione pari a G_s= 400 W/m^2, ha un coefficiente di estinzione per assorbimento pari a μ = 100 m^{-1} ed un coefficiente di riflessione alla radiazione solare pari a ρ_s = 0.1. Supponendo che lo scambio termico con l'aria avvenga solo per convezione con un coefficiente termico convettivo α_e = 5 W/m^2, si determini l'andamento di temperatura all'interno della lastra di vetro.



Quando una radiazione di intensità I_0 passa attraverso uno strato di vetro di spessore d, l'assorbimento di energia raggiante in uno spessore infinitesimo dx è fornito dalla relazione:

$$dI_x = -\mu\, I_x\, dx$$

dove: I_x è l'intensità di radiazione alla distanza x;

μ è il coefficiente di estinzione per assorbimento o sezione d'urto.

Integrando questa equazione differenziale all'interno dello spessore *x* si ottiene:

$$I_x = I_0\, e^{-\mu x}, \quad \text{dove} \quad I_0 = G_s\,(1-\rho_s)$$

In questo caso, se si trascura il contributo dovuto alle riflessioni multiple, l'energia generata per unità di volume è data da

$$g(x) = -\frac{d\,I_x}{d\,x}$$

Ne discende che:

$$g(x) = -I_0\,\mu\, e^{-\mu x}$$

L'equazione della conduzione all'interno della parete è quella monodimensionale stazionaria con sorgente interna:

$$\frac{d^2T}{dx^2} + \frac{I_0\,\mu\, e^{-\mu x}}{\lambda} = 0$$

La soluzione generale di questa equazione differenziale risulta:

$$T(x) = -\frac{I_0\, e^{-\mu x}}{\mu\,\lambda} + C_1\, x + C_2$$

La soluzione particolare può essere determinata imponendo le condizioni al contorno del terzo tipo:

$$-\lambda \left.\frac{d\,T}{d\,x}\right|_{x=0} = \alpha_e\,[T_e - T(x=0)]$$

$$-\lambda \left.\frac{d\,T}{d\,x}\right|_{x=d} = \alpha_e\,[T(x=d) - T_e]$$

Risolvendo il precedente sistema di 2 equazioni nelle due incognite C_1 e C_2 si ottiene:

$$C_1 = -\frac{\frac{\alpha_e\, I_0}{\mu\,\lambda}\left(1-e^{-\mu d}\right) + I_0\left(1+e^{-\mu d}\right)}{2\lambda + \alpha_e\, d}$$

$$C_2 = T_e + \frac{I_0}{\mu\,\lambda} + \frac{I_0}{\alpha_e} + \frac{\lambda}{\alpha_e} C_1$$

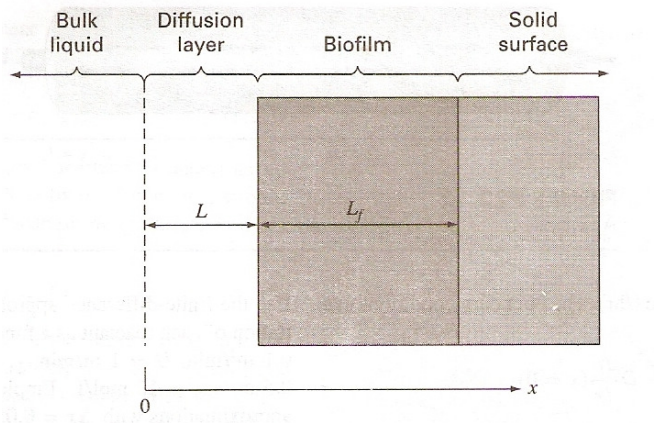
L'andamento qualitativo della temperatura all'interno dello spessore di vetro è quello riportato in figura.

```
function bvpvetro
% Assorbimento della radiazione solare da una lastra di vetro

clc, clear, close all
```

```
global d lambda Te Gs mu rhos alfae I0

% Dati di input
d = 5.e-3;    % spessore del vetro
lambda = 0.5; % conducibilità termica del vetro
Te = 20.0;    % temperatura esterna (aria)
Gs = 400.0;   % intensità di irradiazione
mu = 100.0;   % coefficiente di estinzione per assorbimento
rhos = 0.1;   % coefficiente di riflessione
alfae = 5.0;  % coefficiente di scambio termico convettivo

I0 = Gs * ( 1 - rhos );

% Si fissa inizialmente la temperatura costante a 20 °C
% ed il gradiente di temperatura uguale a zero
solinit = bvpinit(linspace(0,d,10),[20;0]);

solinit.x
solinit.y
% risolutore dell'equazione differenziale
sol = bvp4c(@myode,@mybc,solinit);

sol.x

xint = linspace(0,d,100);
Sxint = deval(sol,xint);

% soluzione analitica
C1 = (-alfae*I0/mu/lambda*(1-exp(-mu*d))-I0*(1+exp(-mu*d)))/...
     (2*lambda+alfae*d);
C2 = Te + I0/mu/lambda + I0/alfae + lambda/alfae*C1;
t=-I0/mu/lambda*exp(-mu*xint)+C1*xint+C2;

plot(xint,Sxint(1,:),'or',xint,t);
legend('Soluzione numerica','soluzione analitica');
xlabel('x [m]'),ylabel('T(x) [°C]')
% --------------------------------------------------------------
function dydx = myode(x,y)

global d lambda Te Gs mu rhos alfae I0

dydx = [  y(2); -I0*mu*exp(-mu*x)/lambda ];
% --------------------------------------------------------------
function res = mybc(ya,yb)
global d lambda Te Gs mu rhos alfae I0
res = [ya(2)+alfae/lambda*(Te-ya(1));yb(2)+alfae/lambda*(yb(1)-Te)];
```

4.19 Sviluppo di un bio-film su una superficie solida (BVP)

Un bio-film, avente spessore pari a L_f = 0.004 cm, ricopre la superficie di un solido come mostrato in figura. Dopo aver attraversato uno strato diffusivo di spessore pari a L = 0.008 cm, un componente chimico A diffonde all'interno del bio-film in cui è soggetto ad una reazione irreversibile del primo ordine che lo converte nel componente B.



Un bilancio di massa in condizioni stazionarie porta a scrivere le seguenti equazioni differenziali per il componente A:

$$D\frac{d^2c_a}{dx^2}=0 \qquad 0 \le x < L$$

$$D_f\frac{d^2c_a}{dx^2}-kc_a=0 \qquad L \le x < L+L_f$$

dove: $D = 0.8$ cm^2/d (Coefficiente di diffusione nello strato diffusivo)

$D = 0.64$ cm^2/d (Coefficiente di diffusione nel bio-film)

$k = 0.1$ d^{-1} (costante di equilibrio del primo ordine per la reazione)

Le condizioni al contorno risultano:

$$c_a = c_{a0} \qquad \text{in } x = 0$$

$$\frac{dc_a}{dx}=0 \qquad \text{in } x = L+L_f$$

essendo c_{a0} la concentrazione iniziale del componente A pari a 100 mol/L.

Si determini con il MATLAB la distribuzione allo stato stazionario del componente A in entrambi gli strati diffusivo e bio-film.

```
function biofilm_bvp

clc
clear all
close all

global D Df k L Lf
D = 0.8; % [cm^2/day]
Df = 0.64; % [cm^2/day]
k = 0.1; % [day^-1]
L = 0.008; % [cm]
Lf = 0.004; % [cm]

Deltax = linspace(0,L+Lf,1.e4);
solinit = bvpinit(Deltax,[0 0]);
sol = bvp4c(@bvp4ode,@bvp4bc,solinit);
plot(sol.x,sol.y(1,:)), grid on
xlabel('layer length [cm]')
ylabel('Concentration of the component Ca [mol/L]')
end
%-------------------------------------------------------
function dcdx = bvp4ode(x,y)
global Df k L Lf
if (x<L) && (x>=0)
    dcdx = [y(2); 0];
else (x<L+Lf) && (x>=L);
    dcdx = [y(2); k*y(1)/Df];
end
end
%-------------------------------------------------------
function res = bvp4bc(ya,yb)
res = [ya(1)-100; yb(2)];
end
```

```
function bvpesempio
%   The BVP  y'' + exp(y) = 0, y(0) = 0 = y(1) is a standard example
%   of a problem with two solutions.  It is easy enough to solve, but
%   some experimentation with the guess may be necessary to get both.

clc
close all

options = bvpset('stats','on');
solinit = bvpinit(linspace(0,1,5),[0.1 0]); % initial guess of the solution
sol1 = bvp4c(@myode,@mybc,solinit,options);

fprintf('\n');

% Change the initial guess to converge to a different solution.
solinit = bvpinit(linspace(0,1,5),[3 0]);
sol2 = bvp4c(@myode,@mybc,solinit,options);

plot(sol1.x,sol1.y(1,:),sol2.x,sol2.y(1,:))
title('Bratu''s equation has two solutions when \lambda = 1.')
xlabel('x')
ylabel('y')

% --------------------------------------------------------------------------
function dydx = myode(x,y)
%ODE function for the Example of the BVP
dydx = [  y(2); -exp(y(1))];

% --------------------------------------------------------------------------
function res = mybc(ya,yb)
%Boundary conditions for the Example of the BVP
res = [ya(1); yb(1)];
```

5. PROBLEMI DI UTILITA' GENERALE

5.1 Conduzione del calore in simmetria piana ed in condizioni transitorie

Si consideri un problema di conduzione del calore attraverso una piastra infinitamente estesa (simmetria piana). L'equazione del calore in assenza di sorgenti interne assume la seguente forma:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$$

Supponiamo che lo spessore della piastra sia pari ad 1 m e che la diffusività termica sia uguale a 0.02 m^2/s. Consideriamo poi le seguenti condizioni iniziali ed al contorno:

$$T(t=0,x) = T_0 = 30\ °C \qquad \forall x \in [0,1]$$

$$\begin{aligned} T(t,x=0) &= T_0 = 30\ °C \\ T(t,x=1) &= T_1 = 100\ °C \end{aligned} \qquad \forall t \in [0,\ 10]$$

```
function conduzione
%
clc, clear, close all

m = 0;
xmesh = linspace(0,1,130);
tspan = linspace(0,10,10);

sol = pdepe(m,@pdefun,@icfun,@bcfun,xmesh,tspan);

plot(xmesh',sol(:,:,1)');

figure % crea una nuova figura senza chiudere la precedente
mesh(xmesh,tspan,sol)
title(sprintf('Numerical solution computed with %d mesh points',length(xmesh)))
xlabel('Distance x [m]')
ylabel('Time t [s]')
zlabel('Temperature T [°C]')

% --------------------------------------------------------------
function [c,f,s] = pdefun(x,t,u,DuDx)
c = 1;
f = 0.02*DuDx;
s = 0;
% --------------------------------------------------------------
function u0 = icfun(x)
u0 = 30;
% --------------------------------------------------------------
function [pl,ql,pr,qr] = bcfun(xl,ul,xr,ur,t)
pl = ul-30;
ql = 0;
pr = ur-100;
qr = 0;
```

5.2 Trasformata rapida di Fourier (Fast Fourier Transform)

Scrivere una funzione MATLAB per calcolare il periodogramma. La sintassi dovrebbe essere:

`[P,f] = Pgram(x,fs)`

dove `P` è il vettore densità spettrale ed `f` il vettore delle frequenze corrispondenti. I dati di input sono il vettore dei dati campionati, `x`, e la frequenza di campionamento, `fs`.

Soluzione

Per la valutazione del periodogramma si passa attraverso il calcolo della Trasformata di Fourier Discreta (DFT) calcolata per mezzo dell'algoritmo della Fast Fourier Transform (FFT).

$$X_k = \sum_{n=0}^{N-1} \left[x_n \; e^{i\frac{2\pi k n}{N}} \right]$$

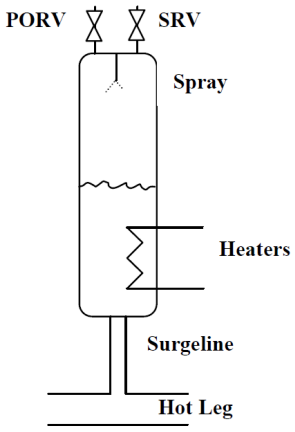
```
function Fourier
% Trasformata rapida di Fourier (FFT)

clc
clear

fs=1000; % frequenza di campionamento
t = 0:1/fs:1.0;
length(t)
x = sin(2*pi*50*t) + sin(2*pi*120*t);
x = x + 2*randn(size(t));
% plot(t,x)

[P,f] = Pgram(x,fs);
plot(f,P),grid

% Hs=spectrum.welch;
% psd(Hs,x,'fs',fs)

[y,j] = max(P);
fs/length(x)

%-------------------------------------------------------------
function [P,f]=Pgram(x,fs)
% Pgram.m - funzione per il calcolo del periodogramma
% Input  - x vettore contenente il segnale campionato ad una frequenza fs
% Output - P vettore densità spettrale approssimata
%          f vettore frequenza

N = length(x);
f = fs / N * (0:round(N/2)-1); % teorema di Nyquist
X = fft(x); % esegue la discrete Fourier transform (DFT) del vettore x
P = X .* conj(X) / N;
P = P(1:round(N/2));
```

5.3 Il pressurizzatore di un impianto nucleare PWR

Facendo uso del modello di equilibrio termodinamico, valutare la pressione finale in un pressurizzatore di un impianto PWR a seguito di un rientro di acqua dalla hot leg causato da un aumento di 1 °C della temperatura media del fluido refrigerante primario.

I dati iniziali sono i seguenti:

- pressione iniziale nel pressurizzatore = 15.5 MPa;
- volume di liquido refrigerante nel primario = 290 m^3;
- volume del pressurizzatore = 60 m^3;
- volume di liquido nel pressurizzatore = 30 m^3;
- temperatura dell'acqua nella hot leg = 320 °C;
- temperatura media del liquido nel primario = 300 °C.



Ripetere il calcolo della pressione finale considerando una temperatura iniziale dell'acqua nella hot leg di 310 °C anziché 320 °C e commentare il differente risultato.

Soluzione

Innanzitutto, calcoliamo la massa, il titolo e l'energia interna iniziale nel pressurizzatore.

$$M_{f,init} = V_{f,init} / v_f\left(p_{init}\right)$$

$$M_{g,init} = V_{g,init} / v_g\left(p_{init}\right)$$

$$M_{init} = M_{f,init} + M_{g,init};\ x_{init} = M_{g,init} / M_{init}$$

$$U_{f,init} = M_{f,init}\, u_f\left(p_{init}\right)$$

$$U_{g,init} = M_{g,init}\, u_g\left(p_{init}\right)$$

$$U_{init} = U_{f,init} + U_{g,init}$$

Per determinare le condizioni finali a seguito di rientro nel pressurizzatore di liquido, per effetto dell'espansione termica del refrigerante primario, si possono utilizzare le equazioni di bilancio di massa e dell'energia a parametri concentrati:

$$\frac{dM}{dt} = W_{l,inlet}$$

$$\frac{dU}{dt} = W_{l,inlet}\, h_{l,inlet}$$

Integrando queste due equazioni tra l'istante iniziale e finale del transitorio si ottiene:

$$M_{final} = M_{init} + \left(\Delta M\right)_{l,inlet}$$

$$U_{final} = U_{init} + (\Delta M)_{l,inlet} h_{l,inlet}$$

Nelle due precedenti equazioni c'è da valutare la massa di liquido che dalla hot leg entra nel pressurizzatore per effetto dell'aumento di 1 °C di temperatura. Essa può essere calcolata sulla base della conoscenza del coefficiente di dilatazione termica volumetrica del liquido a 300 °C (0.00287 $°C^{-1}$):

$$(\Delta M)_{l,inlet} = \Delta V / v_{hl} = V \beta \Delta T / v_{hl}$$

dove v_{hl} è il volume specifico da determinare nelle condizioni di hot-leg.

Il volume specifico e l'energia interna specifica finali sono quindi dati da:

$$v_{final} = V_{PRZ} / M_{final}$$

$$u_{final} = U_{final} / M_{final}$$

Noti i valori finali di volume specifico e di energia interna specifica è possibile scrivere il seguente sistema di due equazioni non lineari nelle due incognite titolo finale e pressione finale:

$$\begin{cases} v_{final} = x_{final} v_g(p_{final}) + (1 - x_{final}) v_f(p_{final}) \\ u_{final} = x_{final} u_g(p_{final}) + (1 - x_{final}) u_f(p_{final}) \end{cases}$$

E' possibile ridurre il sistema ad una sola equazione non lineare esplicitando dalla prima equazione il titolo e sostituendo nella seconda equazione:

$$\frac{v_{final} - v_f(p_{final})}{v_g(p_{final}) + v_f(p_{final})} - \frac{u_{final} - u_f(p_{final})}{u_g(p_{final}) + u_f(p_{final})} = 0$$

Nel seguito si riporta la soluzione MATLAB del problema in cui si è fatto uso delle funzioni relative alle proprietà dell'acqua (`liquid` e `vapour` rispettivamente dell'esercizio 2.7 e 2.8).

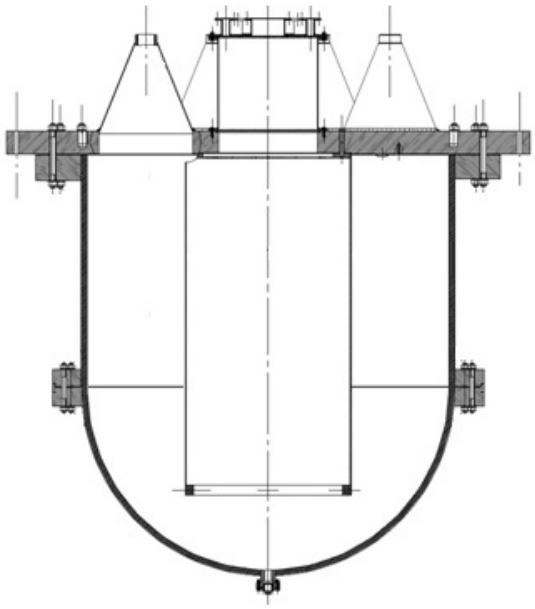
Si lascia al lettore il calcolo della pressione nel caso di temperatura dell'acqua nella hot leg di 310 °C anziché 320 °C.

```
function pressurizer
%
clc, clear, close all
global v_final u_final

% input data
p_init = 15.5e6;        % Initial pressure [Pa]
V = 290.0;              % Coolant volume [m^3]
Vprz = 60.0;            % Pressurizer volume [m^3]
Vprz_l_init = 30.0;     % Pressurizer liquid volume [m^3]
Thl_init = 320+273.15; % Hot leg temperature [K]
Tp_init = 300+273.15;  % Primary liquid average temperature [K]
DT = 1;                 % Prymary average temperature increase

% Initial Mass and Energy in the pressurizer
H2Ol = liquid(p_init,tsat(p_init));
Mf_init = Vprz_l_init / H2Ol.v;
Uf_init = Mf_init * H2Ol.u;
```

```
H2Ov = vapour(p_init,tsat(p_init));
Mg_init = (Vprz - Vprz_l_init) / H2Ov.v;
Ug_init = Mg_init * H2Ov.u;

M_init = Mf_init + Mg_init;
x_init = Mg_init / M_init;
U_init = Uf_init + Ug_init;

v_init = Vprz / M_init;
u_init = U_init / M_init;

% Thermal expansion coefficient
H2Ol_hl = liquid(p_init,Thl_init);
H2Ol_pr = liquid(p_init,Tp_init);

DV = V * H2Ol_pr.beta * DT;
DM = DV / H2Ol_hl.v;
h_hl = H2Ol_hl.h;

% Final Mass and Energy in the pressurizer
M_final = M_init + DM;
U_final = U_init + DM * h_hl;

v_final = Vprz / M_final;
u_final = U_final / M_final;
p_final = fzero(@fun,[p_init-0.1e6 p_init+0.1e6])

end
%-------------------------------------------------------------
function y=fun(p)

global v_final u_final

H2Ol = liquid(p,tsat(p));
vf = H2Ol.v; uf = H2Ol.u;
H2Ov = vapour(p,tsat(p));
vg = H2Ov.v; ug = H2Ov.u;
y = (v_final - vf) / (vg - vf) - (u_final - uf) / (ug - uf);

end
```

5.4 Recipiente in pressione

Si debba progettare, secondo le norme ASME, il recipiente in pressione il cui schema è mostrato in figura. Si tratta di un recipiente in acciaio inox AISI 304 per quanto riguarda il fasciame cilindrico e la semisfera e in acciaio Fe430 per quanto riguarda le tre flange. Il vessel ha un volume interno di circa 1.5 m^3 e contiene acqua alla pressione differenziale di 0.5 bar ed alla temperatura massima di 60 °C.



Il recipiente verrà sostenuto tramite la piastra superiore sulla quale sono presenti i coni di ingresso dell'acqua.

Soluzione

5.4.1Condizioni operative e carichi ammissibili

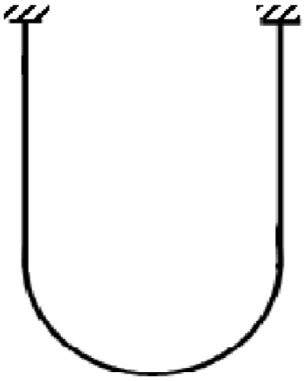
Si assume come condizione di progetto una pressione massima dell'acqua nel vessel pari a:

p_{max} = 0.05 MPa

$$\sigma_{p,0.2}^{AISI\,304} = 241\,MPa \quad \Rightarrow \quad \sigma_{amm} = \frac{\sigma_{p,0.2}^{AISI\,304}}{\underbrace{1.6}_{\substack{fattore\ di\ riduzione \\ rispetto\ a\ \sigma_p}}} \times \underbrace{0.7}_{\substack{fattore\ di\ riduzione \\ per\ le\ saldature}} \approx 105\,\text{MPa}$$

5.4.2 Verifica del mantello alle tensioni membranali

Si considera che lo sforzo di pressione e quello dovuto al peso proprio e al peso del volume liquido contenuto si sovrappongano. Si ha:



$V_{fluido} = 1.5 \text{ m}^3$

$P_{fluido} = \rho g V_{fluido} = 998.2 \times 9.81 \times 1.5 \approx 14689 \text{ N}$

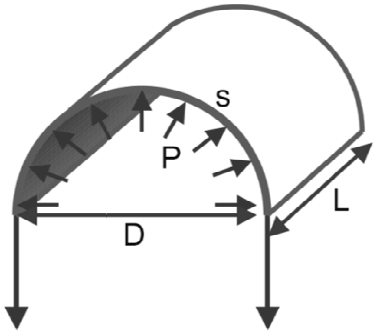
$s = 6 \text{ mm}$ (Spessore del mantello)

$P_{struttura} \approx 4315 \text{ N}$ $\rho_{acciaio} = 7860 \text{ kg/m}^3$

Schema di massima della struttura

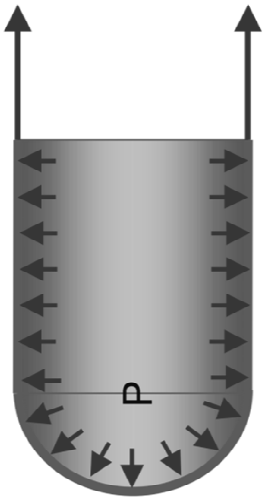
Per l'equilibrio circonferenziale si ha:

$$\sigma_\theta = \frac{pR_m}{s} = \frac{0.05MPa \times 629mm}{6mm} = 5.24 \text{ MPa}$$



Schema per il calcolo delle sollecitazioni circonferenziali

Per l'equilibrio assiale risulta:



$$\sigma_z = \frac{P_{fluido} + P_{struttura} + p\pi R_m^2}{s\pi D_m} =$$

$$= \frac{14689 + 4315 + 50000 \cdot \pi \cdot (0.632)^2}{0.006 \cdot \pi \cdot 1.264} = 3.43 \text{MPa}$$

Schema per il calcolo delle sollecitazioni assiali

La tensione equivalente σ_e risulta:

1) Tresca

$$\boxed{\sigma_e = \sigma_\theta = 5.24 \text{ MPa} << \sigma_{amm}}$$

2) Von Mises

$$\boxed{\sigma_e = \frac{1}{\sqrt{2}}\sqrt{(\sigma_\theta - \sigma_z)^2 + (\sigma_\theta)^2 + (\sigma_z)^2} = 4.61 \text{ MPa} << \sigma_{amm}}$$

5.4.3 Verifiche relative alle guarnizioni ed ai bulloni (ASME Section VIII Division 1, Appendix 2)

Tale verifica è stata realizzata seguendo le indicazioni riportate sulla raccolta ASME VIII [1-2]. In particolare, in questo paragrafo e nel successivo vengono indicate le procedure di dimensionamento delle guarnizioni, dei bulloni e delle flange. Alla fine del prossimo paragrafo successivo in Tabella 1 e 2 vengono riassunti i dati numerici relativi, rispettivamente, al dimensionamento della flangia superiore e della flangia inferiore.

Per chiarire la procedura utilizzata, nel seguito verrà riportato l'elenco delle grandezze utilizzate.

b = larghezza utile di assetto guarnizione in [m]:

$$b = \begin{cases} b_0 & \text{per} \quad b_0 \le 6.3 \cdot 10^{-3} \text{ m} \\ 2.52 \cdot 10^{-3} \sqrt{b_0} & \text{per} \quad b_0 > 6.3 \cdot 10^{-3} \text{ m} \end{cases}$$

b_0 = larghezza convenzionale di assetto guarnizione in [m] che per la nostra applicazione vale:

$$b_0 = N / 2$$

N = larghezza della parte della guarnizione in [m] che fa effettivamente tenuta;

G = diametro della circonferenza in [m] che passa per il punto di applicazione della reazione della guarnizione;

y = carico unitario di assetto guarnizione in [Pa] (è il carico unitario minimo iniziale con cui occorre serrare la guarnizione a temperatura ambiente e senza pressione interna al recipiente della flangia al fine di garantire l'assetto della guarnizione sulla flangia);

m = coefficiente di tenuta (è un fattore che moltiplicato per la pressione interna al recipiente fornisce il carico unitario minimo che deve sempre essere presente sulla guarnizione durante le condizioni di esercizio al fine di garantire la tenuta).

I valori di y e m sono stati scelti dalla Tabella 2-5.1 presente nella ASME Section VIII Division 1 appendix 2, relativamente ad una guarnizione in elastomero senza fibre di amianto.

In base a quanto riportato nelle ASME si ha:

- La forza minima di serraggio dei bulloni al fine di garantire la tenuta nelle condizioni di esercizio vale:

$$W_{m1} = P_{fluido} + P_{struttura} + \frac{\pi}{4} G^2 \ p + \pi \ G \ 2 \ b(m \ p)$$

- La forza minima di serraggio dei bulloni che in fase di montaggio (serraggio) deve garantire l'adattamento plastico della guarnizione alle irregolarità della sede (assetto guarnizione) vale:

$$W_{m2} = P_{struttura} + \pi\, G\, b\, y$$

Tramite la W_{m1} e la W_{m2} si calcola il valore minimo dell'area dei bulloni richiesto, rispettivamente, per il caso operativo e per l'assetto della guarnizione:

$A_{m1} = W_{m1} / \sigma_{b,amm}$

$A_{m2} = W_{m2} / \sigma_{ba,amm}$

L'area effettiva totale dei bulloni deve essere maggiore del massimo tra le due aree valutate precedentemente:

A_m = area totale minima dei bulloni calcolata sul diametro del nocciolo = $max\{A_{m1}, A_{m2}\}$

Determinata l'area minima dei bulloni e conoscendo il diametro del bullone scelto si può quindi calcolare il numero minimo (dipendente dal diametro del bullone scelto) di bulloni necessario per la giunzione. Indicando con A_b l'area effettiva totale dei bulloni effettivamente presenti sulla flangia dovrà quindi risultare $A_b \geq A_m$.

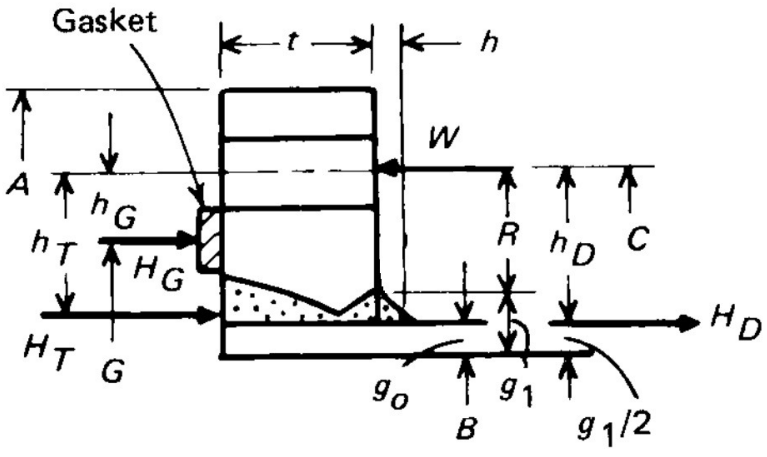
Per quanto riguardo il carico dei bulloni per il dimensionamento della flangia si è scelto di utilizzare i carichi minimi necessari per la tenuta e l'assetto, senza considerare un eventuale errato sovratensionamento dei bulloni. In pratica, i bulloni andranno stretti con chiave dinamometrica fornendo la coppia minima necessaria per assicurare la tenuta in condizioni operative. In tali condizioni il carico massimo sui bulloni e quindi dei bulloni sulla flangia vale:

$W = max\{W_{m1}, W_{m2}\}$

5.4.4 Verifiche relative allo spessore della flangia (ASME Section VIII Division 1, Appendix 2)

Il materiale con il quale verranno progettate le flange è acciaio al carbonio Fe430 per il quale si ha:

$$\sigma_{p,0.2}^{S235JR} = 260 \text{ MPa} \quad \Rightarrow \quad \sigma_{amm} = \frac{\sigma_{p,0.2}^{S235JR}}{\underbrace{1.6}_{\substack{fattore\ di\ riduzione \\ rispetto\ a\ \sigma_p}}} \approx 162.5 \text{ MPa}$$



Con riferimento alla figura precedente viene imposto l'equilibrio dei momenti rispetto all'asse del foro dei bulloni. Le forze H, H_D, H_T *e* H_G, che rappresentano rispettivamente la forza idrostatica, la forza idrostatica nella zona interna alla flangia, la differenza tra la forza idrostatica totale e la forza idrostatica interna alla flangia, il carico della guarnizione, sono date da:

$$H = P_{fluido} + P_{struttura} + \frac{\pi}{4} G^2 p$$

$$H_D = P_{fluido} + P_{struttura} + \frac{\pi}{4} B^2 p$$

$$H_T = H - H_D$$

$$H_G = W - H$$

La norma ASME VIII Division 1 Appendix 2 non tiene conto dei carichi esterni applicati oltre alla pressione. Nel calcolo effettuato si è invece tenuto conto anche del peso della struttura e del fluido.

Siano inoltre:

$\sigma_{fa,amm}$ = tensione ammissibile del materiale della flangia a temperatura ambiente;

$\sigma_{ba,amm}$ = tensione ammissibile del materiale dei bulloni a temperatura ambiente;

$\sigma_{f,amm}$ = tensione ammissibile del materiale della flangia a temperatura di esercizio;

$\sigma_{b,amm}$ = tensione ammissibile del materiale dei bulloni a temperatura di esercizio.

Data la relativamente bassa temperatura dell'acqua all'interno del vessel in condizioni di esercizio, nel seguito sarà considerata soltanto la tensione ammissibile a temperatura ambiente sia per il materiale della flangia che per il materiale del bullone.

Il braccio (h_D, h_T, h_G) delle rispettive forze si calcola attraverso le seguenti relazioni:

$$h_D = R + 0.5 \cdot g_1$$

$$h_T = 0.5 \cdot (R + g_1 + h_g)$$

$$h_G = 0.5 \cdot (C - G)$$

dove R rappresenta la distanza radiale tra la circonferenza passante per gli assi dei bulloni e l'intersezione tra l'eventuale collarino e l'interno della flangia, calcolato attraverso la seguente relazione:

$$R = \frac{C - B}{2} - g_1$$

Il momento M_0 da applicare alla flangia per il caso operativo (M_{op}) e per il caso gasket seating (M_{GS}) si calcola rispettivamente come:

$$M_{op} = H_T \cdot h_T \cdot H_D \cdot h_D \cdot H_G \cdot h_G$$

$$M_{GS} = W \cdot h_G$$

dove W è dato da:

$$W = \max\left(W_{m1}, W_{m2}\right)$$

Lo spessore minimo richiesto per la flangia risulta quindi il massimo spessore tra:

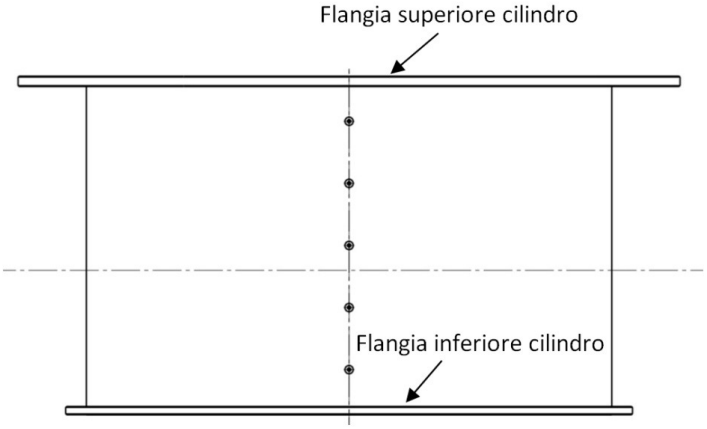
$$t_{\min,\mathrm{op}} = \left[\frac{M_{op}Y}{\sigma_{f,amm}\cdot B}\right]^{\frac{1}{2}}$$

$$t_{\min,\mathrm{GS}} = \left[\frac{M_{GS}Y}{\sigma_{b,amm}\cdot B}\right]^{\frac{1}{2}}$$

dove il coefficiente adimensionale Y è calcolato attraverso la formula:

$$Y = \frac{1}{\frac{A}{B}-1}\cdot\left[0.66845 + 5.71690\cdot\frac{\left(\frac{A}{B}\right)^2\log_{10}\left(\frac{A}{B}\right)}{\left(\frac{A}{B}\right)^2-1}\right]$$

Nelle Tabelle 1 e 2 vengono riassunti i risultati relativi al dimensionamento delle due flange della virola cilindrica (superiore e inferiore), mentre la flangia della semisfera da applicare sotto il cilindro è uguale alla flangia inferiore del cilindro con la quale si accoppia.



Flange nella parte cilindrica

Condizioni di progetto			**Guarnizione**	**Flangia**
Pressione di progetto		0.05 MPa	N = 0.05 m	A = 1.6 m
Temp. di progetto		$T_{ambiente}$	b = 0.0126 m	B = 1.258 m
Materiale della flangia		S235JR	G =1.37 m	C = 1.48 m
Materiale dei bulloni		S235JR	$y = 1.4\cdot10^6$ Pa	
Riduzione di corrosione		No corrosione	m = 1	
Tensioni ammissibili	Flangia	Condizioni di progetto $S_f = 162.5\cdot10^6$ Pa	**Carichi e calcolo dei bulloni**	

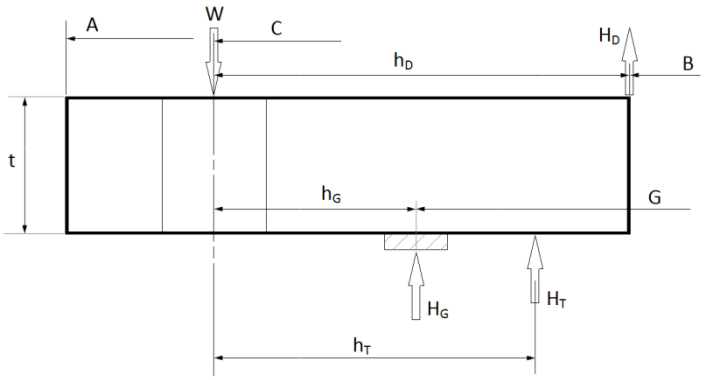
<table>
<tr><td></td><td>Bulloni</td><td>Condizioni di progetto
$S_b = 162.5 \cdot 10^6$ Pa</td><td>W_{m1} =98729 N</td><td>$A_m = 6.07 \cdot 10^{-4}$ m</td></tr>
<tr><td colspan="2"></td><td></td><td>W_{m2} = 75922 N</td><td>$A_b = 6.1 \cdot 10^{-3}$ m</td></tr>
<tr><td colspan="2">Diametro bullone</td><td>Numero bulloni 24</td><td>H = 93306 N</td><td></td></tr>
<tr><td colspan="2">0.016 m</td><td>Numero minimo bulloni 4</td><td colspan="2"></td></tr>
<tr><td colspan="2">Condizione</td><td colspan="3">Forza x braccio</td></tr>
<tr><td colspan="2" rowspan="5">Operativa</td><td>H_D = 81747 N</td><td colspan="2">h_D = 0.1045 m</td></tr>
<tr><td>H_G = 5423 N</td><td colspan="2">h_G = 0.0550 m</td></tr>
<tr><td>H_T =11559 N</td><td colspan="2">h_T = 0.0830 m</td></tr>
<tr><td></td><td>Operating</td><td>M_{op} = 9800.2 N·m</td></tr>
<tr><td></td><td>Seating</td><td>M_{GS} = 4175.7 N·m</td></tr>
<tr><td colspan="3" rowspan="3"> 
</td><td colspan="2">Costanti dovute alla forma</td><td>$K = A/B$= 1.2719
Y = 8.211</td></tr>
<tr><td>t min</td><td>$t = \sqrt{\frac{M_{op} Y}{\sigma_{f\,a,amm} B}}$</td><td>0.0198 m</td></tr>
</table>

Tabella 1: Dimensionamento flangia superiore

<table>
<tr><th colspan="3">Condizioni di progetto</th><th colspan="2">Guarnizione</th><th>Flangia</th></tr>
<tr><td colspan="2">Pressione di progetto</td><td>0.05 MPa</td><td>N = 0.01 m</td><td colspan="2">A = 1.37 m</td></tr>
<tr><td colspan="2">Temp. di progetto</td><td>$T_{ambiente}$</td><td>b = 0.005 m</td><td colspan="2">B = 1.258 m</td></tr>
<tr><td colspan="2">Materiale della flangia</td><td>S235JR</td><td>G =1.295 m</td><td colspan="2">C = 1.33 m</td></tr>
<tr><td colspan="2">Materiale dei bulloni</td><td>S235JR</td><td>$y = 1 \cdot 10^6$ Pa</td><td colspan="2"></td></tr>
<tr><td colspan="2">Riduzione di corrosione</td><td>No corrosione</td><td>m = 1</td><td colspan="2"></td></tr>
<tr><td rowspan="2">Tensioni ammissibili</td><td>Flangia</td><td>Condizioni di progetto
$S_f = 162.5 \cdot 10^6$</td><td colspan="3">Carichi e calcolo dei bulloni</td></tr>
<tr><td>Bulloni</td><td>Condizioni di progetto
$S_b = 162.5 \cdot 10^6$</td><td>W_{m1} =87491 N</td><td colspan="2">$A_m = 5.4 \cdot 10^{-4}$ m</td></tr>
<tr><td colspan="2"></td><td></td><td>W_{m2} = 28479 N</td><td colspan="2">$A_b = 3.2 \cdot 10^{-3}$ m</td></tr>
<tr><td colspan="2">Diametro bullone</td><td>Numero bulloni 24</td><td>H = 85457 N</td><td colspan="2"></td></tr>
<tr><td colspan="2">0.013 m</td><td>Numero minimo bulloni 4</td><td colspan="3"></td></tr>
</table>

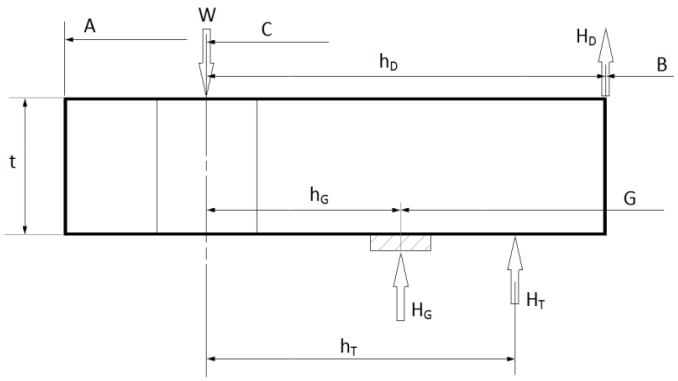
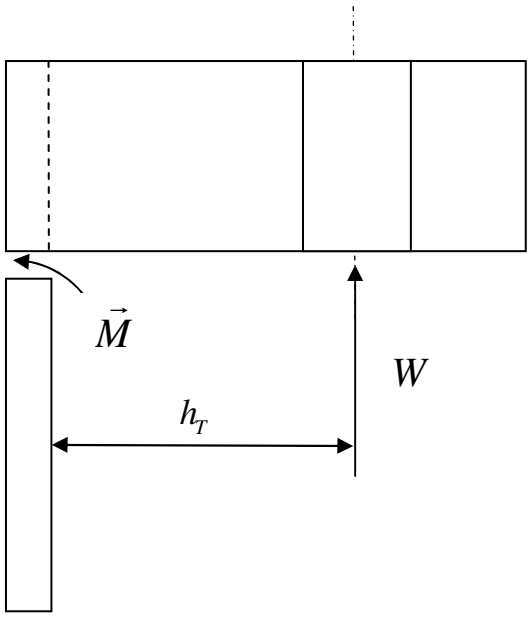
<table>
<tr><th>Condizione</th><th colspan="3">Forza x braccio</th></tr>
<tr><td rowspan="5">Operativa</td><td>H_D = 81747 N</td><td colspan="2">h_D = 0.0295 m</td></tr>
<tr><td>H_G = 2034.2 N</td><td colspan="2">h_G = 0.0175 m</td></tr>
<tr><td>H_T = 3709.5 N</td><td colspan="2">h_T = 0.0268 m</td></tr>
<tr><td></td><td>Operating</td><td>M_{op} = 2546.4 N·m</td></tr>
<tr><td></td><td>Seating</td><td>M_{GS} = 498.3744 N·m</td></tr>
<tr><td rowspan="2"> 
</td><td colspan="2">Costanti dovute alla forma</td><td>$K = A/B$ = 1.089</td></tr>
<tr><td colspan="2">Y = 22.6748</td></tr>
<tr><td colspan="2">t</td><td>$t = \sqrt{\frac{M_{op} Y}{\sigma_{fa,amm} B}}$</td><td>0.0168 m</td></tr>
</table>

Tabella 2: Dimensionamento flangia inferiore

5.4.5 Verifica del fasciame cilindrico in corrispondenza della flangia superiore

Il materiale scelto per la progettazione del fasciame cilindrico e del fondo semisferico è, come indicato nel paragrafo 1, acciaio AISI 304. Con riferimento alla figura riportata nel seguito si è verificata la sezione del fasciame cilindrico alla base del collegamento con la flangia superiore, in modo da garantirne l'integrità strutturale sotto l'applicazione del momento trasmesso al fasciame a causa del collegamento con la flangia stessa.



Schema per il dimensionamento della sezione di collegamento del fasciame cilindrico con la flangia

Nella figura precedente W rappresenta il valore massimo tra W_{m1} e W_{m2}.

La tensione massima sul fasciame sottoposto a momento flettente M è data da:

$$\sigma_{\max} = \frac{6M}{Lh^2} \leq \sigma_{amm}$$

dove $L = \pi B$, mentre h è lo spessore del fasciame (6 mm).

In definitiva si ottiene:

$$\sigma_{\max} = \frac{6M}{Lh^2} = 4.86 \text{ MPa} \leq \sigma_{amm} = 105 \text{ MPa}$$