07/01/2014

Informatica per la Finanza
Informatica per le Scienze Economiche
Informatica e Programmazione per il Turismo

Problemi di Code

- Le code di attesa sono un tipo di processo stocastico che ha applicazioni pratiche in moltissimi campi:
  - Reti di computer
  - Sistemi operativi
  - Sistemi per la gestione delle code dei clienti in servizi pubblici/privati
  - …

Problemi di Code

- E' stata sviluppata una notevole teoria che porta a dei risultati analitici sui parametri fondamentali di valutazione come:
  - Tempo medio atteso in coda
  - Lunghezza media della coda
- Le code di attesa sono fra l'altro caratterizzate:
  - Distribuzione numero degli arrivi in coda
  - Numero dei servitori
    - Con code indipendenti per ogni servitore
    - Con una coda unica
  - Distribuzione del tempo di servizio del servitore
  - Disciplina della coda (LIFO, FIFO, priorità, …)

Problemi di Code

- Non volendo ricorre ai risultati analitici, la simulazione può essere un utile strumento
- Nel seguito si mostrano alcuni esempi di simulazione

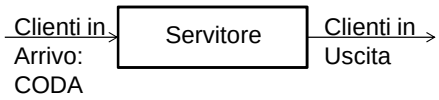
Q4 5/09/2012

In una coda arrivano:
- con una probabilità del 5% 15 clienti al minuto,
- con una probabilità del 10% 5 clienti al minuto,
- negli altri casi arrivano 10 clienti al minuto.

Inizialmente la coda è vuota.
La coda ha un unico servitore che:
- nel 90% dei casi serve in un minuto 12 clienti,
- nel 5% del casi serve in un minuto clienti 8 clienti,
- nel restante 5% serve in un minuto clienti 10 clienti.

Scrivere la funzione Matlab **`quesito6()`** che simula l'andamento di questa coda simulando gli arrivi ed i clienti serviti minuto per minuto. La funzione
ha in ingresso:
- **`T`** durata della simulazione espressa in minuti.

Ha in uscita:
- **`Coda`** vettore 1x**T** con il numero di clienti in coda minuto per minuto (i valori sono, ovviamente, solo positivi).



Una coda contiene sempre un numero di clienti POSITIVO o NULLO

Q4 5/09/2012

- La simulazione procede considerando che esistono due distribuzioni indipendenti:
  - numero clienti arrivati in coda;
  - numero di clienti che possono essere serviti;
- Si deve ricordare:
  - Il servitore prende i clienti dalla coda
  - La coda può avere:
    - numero di clienti inferiore a quelli che possono essere serviti:
      - » Si servono solo i clienti in coda e all'istante successivo la coda è vuota;
      - » i clienti che si potevano ancora servire NON si cumulano nell'istante di tempo successivo.
    - numero di clienti superiore a quelli che possono essere serviti:
      - » i clienti non serviti restano in coda e si sommano a quelli che arrivano nell'istante di tempo successivo.

Q4 5/09/2012

Esercizio: generalizzare per distribuzioni qualsiasi (usare due parametri)

```
function [ coda ] = codaSemplice( T )
  distrArr =[5/100, 15/100, 100/100; 15, 5, 10];
  distrServ=[90/100, 95/100, 100/100; 12, 8, 10];

  inCoda = 0;
  coda = zeros(1,T);

  vcArr=rand(1,T);
  vcServ=rand(1,T);

  for i=1:T   %simulazione
    %prima riempio la coda poi servo
    inCoda = inCoda + valore(distrArr,vcArr(i));
    servibili = valore(distrServ,vcServ(i));

    serviti = realmenteServiti(inCoda, servibili);

    inCoda=inCoda-serviti; %clienti restanti in coda
    coda(i)= inCoda;
  end

end
```

Le distribuzioni sono in una matrice
- prima riga cumulata
- seconda riga valori corrispondenti

La funzione *valore()* preleva i valori dalle distribuzioni

Ricava il numero di clienti effettivamente serviti e che escono dalla coda

Q4 5/09/2012

```
function qta=valore(matValori, val)
  qta0 = matValori(2,find(matValori(1,:) > val,1));

  for i=1:size(matValori,2)
      if matValori(1,i) > val
          qta1=matValori(2,i);
          break;
      end
  end

  if qta0 ~= qta1
      error('impossibile!')
  else
      qta=qta0;
  end
end

function num=realmenteServiti(cd, sr)
  if cd > sr
      num = sr;
  else
      num = cd;
  end
end
```

Modo funzionale

Modo iterativo

obbligatorio

I risultati debbono essere uguali!

Q4 28/01/2013

In una coda arrivano:

- con una probabilità del 5% 25 clienti al minuto,
- con una probabilità del 10% 20 clienti al minuto,
- negli altri casi arrivano 30 clienti al minuto.

Inizialmente la coda è vuota.

La coda ha **due** servitori che lavorano contemporaneamente e indipendentemente ma con lo stesso comportamento così descritto:

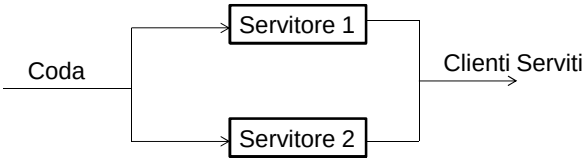
- nel 90% dei casi ogni servitore serve in un minuto 15 clienti,
- nel 5% del casi ogni servitore serve in un minuto 10 clienti,
- nel restante 5% ogni servitore serve in un minuto 5 clienti.

Scrivere la funzione Matlab **`quesito5()`** che simula l'andamento di questa coda simulando gli arrivi ed i clienti serviti minuto per minuto. La funzione ha in ingresso:

- **`T`** durata della simulazione espressa in minuti.

Ha in uscita:

- **`Coda`** vettore 1x**`T`** con il numero di clienti in coda minuto per minuto (i valori sono, ovviamente, solo positivi).



Q4 28/01/2013

- Per la risoluzione valgono le considerazioni precedenti
- I due servitori lavorano indipendentemente
  - I numeri casuali vanno estratti per entrambi
  - Si deve calcolare il numero di clienti complessivamente servibili

Q4 28/01/2013

```
function [ coda ] = coda2Servitori( T )
  distrArr =[5/100, 15/100, 100/100; 25, 20, 30];
  distrServ=[90/100, 95/100, 100/100; 15, 10, 5];

  inCoda = 0;
  coda = zeros(1,T);

  vcArr=rand(1,T);
  vcServ1=rand(1,T);
  vcServ2=rand(1,T);

  for i=1:T   %simulazione
    %prima riempio la coda poi servo
    inCoda = inCoda + valore(distrArr,vcArr(i));
    servibili1 = valore(distrServ,vcServ1(i));
    servibili2 = valore(distrServ,vcServ2(i));

    serviti = realmenteServiti(inCoda, servibili1+servibili2);

    inCoda=inCoda-serviti; %clienti restanti in coda
    coda(i)= inCoda;
  end

end
```

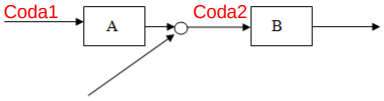
Il codice è simile a quello dell'esempio precedente

Q4 20/02/2013

Scrivere la funzione **quesito4()** che ha:

- in ingresso un numero intero **N** positivo,
- in uscita due vettori **coda1** e **coda2**.

La funzione simula l'andamento di un sistema di code che opera in questo modo:



I clienti arrivano nella coda al servitore A con un tasso di 20 clienti/ora; il servitore A li serve secondo questa distribuzione casuale:

- nel 30% dei casi serve 15 clienti/ora
- nel 60% dei casi serve 25 clienti/ora
- nel 10% dei casi serve 20 clienti/ora

i clienti che escono dalla prima coda si sommano ad altri clienti che arrivano secondo questa distribuzione:

- nel 50% dei casi arrivano 5 clienti/ora
- nel 40% dei casi arrivano 10 clienti/ora
- nel 10% dei casi non arrivano clienti

Il servitore B serve secondo questa distribuzione:

- nel 80% dei casi 35 clienti/ora
- nel 10% dei casi 20 clienti/ora
- nel 10% dei casi 30 clienti/ora

Se **N** indica il numero delle ore di simulazione scrivere nelle variabili **coda1** e **coda2** il numero di clienti in coda ora per ora rispettivamente al servitore A ed al servitore B, ricordando che:

- inizialmente le code sono vuote,
- il numero di clienti in coda non è mai negativo (se lo diventa riportare il valore a 0),
- i clienti non serviti in un'ora si cumulano con quelli dell'ora successiva.

Q4 20/02/2013

– In questo caso occorre tener presente che:
- Il tasso di arrivo nella prima coda è fisso
- i clienti serviti dalla primo servitore entrano nella seconda coda nell'istante di tempo successivo;
- esiste un flusso indipendente di ingressi al secondo servitore;
- nel primo istante di tempo nella seconda coda entrano solo i clienti del flusso B;

Q4 20/02/2013

```
function [ coda1, coda2 ] = codeInCascata( N )
  distArr1=20;
  distrArr2 =[50/100, 90/100, 100/100; 5, 10, 0];

  distrServA=[30/100, 90/100, 100/100; 15, 25, 20];
  distrServB=[80/100, 90/100, 100/100; 35, 20, 30];

  inCoda1 = 0;   inCoda2 = 0;
  uscitaCoda1=0; %uscita coda1 tempo precedente
  coda1 = zeros(1,N); coda2 = zeros(1,N);

  vcArr2=rand(1,N);
  vcServA=rand(1,N);
  vcServB=rand(1,N);

  for i=1:N   %simulazione
    %prima riempio la coda poi servo
    inCoda1 = inCoda1 + distArr1;
    servibili1 = valore(distrServA,vcServA(i));

    inCoda2 = inCoda2 + uscitaCoda1 + valore(distrArr2,vcArr2(i));
    servibili2 = valore(distrServB,vcServB(i));

    uscitaCoda1 = realmenteServiti(inCoda1, servibili1);
    inCoda1=inCoda1-uscitaCoda1; %clienti restanti in coda
    coda1(i)= inCoda1;

    uscitaCoda2 = realmenteServiti(inCoda2, servibili2);
    inCoda2=inCoda2-uscitaCoda2; %clienti restanti in coda
    coda2(i)= inCoda2;
  end

end
```

Ancora una volta il codice è simile

Nella seconda coda si sommano quelli in uscita dalla prima al tempo precedente dati da *uscitaCoda1*

Q1 20/02/2013

```
'
'Scrivere la funzione fQuesito1() che
'  - ha in ingresso i parametri v1 e v2 di tipo vettore di Double dei quali non si
'    conosce a priori il numero di elementi né il valore iniziale dell'indice del vettore
' - la funzione restituisce il numero di elementi di v1
'   che sono contenuti in v2
'
'Scrivere la sub sQuesito1() che:
'  - definisce due vettori dinamici X ed Y
'  - carica in X i valori numerici della colonna A di Foglio1 (la colonna non contiene solo valori
'    numerici) il riempimento termina quando si incontra una cella vuota o quando si sono riempiti
'    20 elementi di X. Il vettore viene ridimensionato al numero effettivo di valori inseriti.
' - legge due valori A e B con una InputBox finché A>0 e A<B
' - dimensiona il vettore Y con indici da A a B
' - riempie il vettore Y con i valori letti dal file dati.txt (il file contiene solo valori
'   numerici). Il riempimento termina quando è finito il file o il vettore è riempito (il vettore
'   non va più ridimensionato)
' - scrive in C1 il valore restituito da fQuesito1() richiamata con parametri X ed Y
'
```


Q1 20/02/2013

```
Sub sQuesito1()
  Dim X() As Double, Y() As Double
  Dim rg As Integer, j As Integer
  Dim A As Double, B As Double

  ReDim X(20)
  rg = 1: j = 1
  Do While Not IsEmpty(Cells(rg, 1)) And j <= 20
    If IsNumeric(Cells(rg, 1)) Then
      X(j) = Cells(rg, 1)
      j = j + 1
    End If
    rg = rg + 1
  Loop
  ReDim Preserve X(j - 1)

  Do
     A = InputBox("dammi un numero")
     B = InputBox("dammi un numero")
  Loop While Not (A > 0 And A < B)

  ReDim Y(A To B)

  Open ThisWorkbook.Path & "\dati.txt" For Input As 1
  j = A
  Do While Not EOF(1) And j <= B
    Input #1, Y(j)
    j = j + 1
  Loop
  Close(1)
  Range("C1") = fQuesito1(X, Y)
End Sub
```

Tipo di ciclo più adatto

Notare la condizione

Q1 20/02/2013

```
Function fQuesito1(X() As Double, Y() As Double)
  Dim i As Integer, j As Integer

  fQuesito1 = 0
  For i = LBound(X) To UBound(X)
    For j = LBound(Y) To UBound(Y)
      If X(i) = Y(j) Then
        fQuesito1 = fQuesito1 + 1
      End If
    Next
  Next
End Function
```

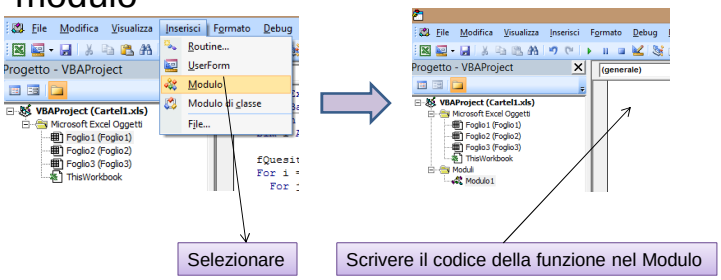
Usare **UBound()** ed **LBound()** rende il codice indipendente dalle dimensioni dei vettori

Q2 14/09/2009 (Elementi)

- Si desidera definire nell'ambiente Excel la nuova funzione **`calcolaDispari()`**. Essa ha in ingresso un intervallo di celle e calcola quanti valori dell'intervallo sono numeri interi dispari.
  - Si ricorda che
    - l'operatore MOD va applicato solo sui numeri interi
      - » quindi bisogna selezionare solo le celle che contengono numeri interi (usare le funzioni **IsNumeric()** e **CInt()** in modo opportuno)
    - non esiste una funzione che verifica se una cella contiene un valore di tipo Integer.

Q2 14/09/2009 (Elementi)

– Per aggiungere una funzione all'ambiente Excel del foglio di lavoro essa va scritta in un modulo



Q2 14/09/2009 (Elementi)

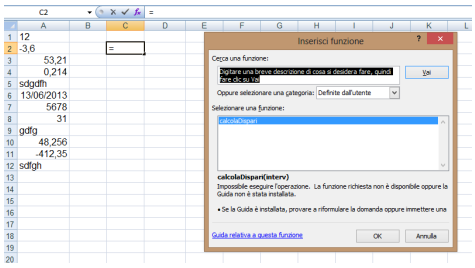
```
Option Explicit
Function calcolaDispari(interv As Range)
  Dim q As Variant

  For Each q In interv
    If IsNumeric(q) Then
      If q = CInt(q) Then 'numero intero
        If q Mod 2 = 1 Then
          calcolaDispari = calcolaDispari + 1
        End If
      End If
    End If
  Next
End Function
```

Intervallo di valori

Il tipo della variabile per scorrere l'intervallo deve essere **Variant** (le celle contengono di tutto)

Q2 14/09/2009 (Elementi)

– Selezionando le *funzioni Definite* dall'utente troviamo la funzione appena scritta

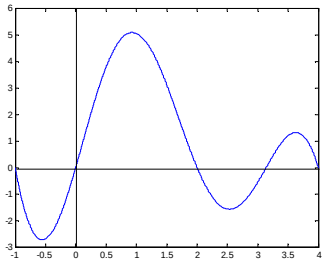


Q4 04/04/2013

Sia data la funzione $f(x) = (x^3 - 5 \cdot x^2 + 2 \cdot x + 8) \cdot \sin x$. Scrivere la funzione Matlab **`quesito4()`** che:

- disegna la funzione con 10000 punti equispaziati fra -1 e 4;
- calcola il valore dell'integrale con il metodo del valor medio della funzione fra -1 e 4 e lo restituisce attraverso il parametro **`area1`**;
- calcola il valore dell'integrale con il metodo accetta/rifiuta della funzione fra -1 e 4 e lo restituisce attraverso il parametro **`area2`**.

– Dal grafico della funzione si comprende come procedere



Nel metodo accetta/rifiuta bisogna considerare che la funzione ha aree positive e negative che vanno sommate algebricamente

Il metodo del valor medio non ha questi problemi

Q4 04/04/2013

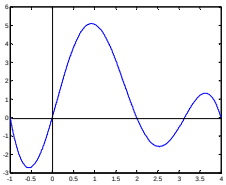
– per il metodo accetta/rifiuta:
- Può essere utile individuare gli zeri
  - quelli del polinomio:

```
sort(roots([1 -5 2 8]))
  ans =  -1.0000  2.0000   4.0000
```

  - quelli di sin(x):

    x=Kπ: 0, π
- Dedurre dal grafico i massimi e i minimi

4 zone: da -1 a 0 area negativa
da 0 a 2 area positiva
da 2 a π area negativa
da π a 4 area positiva



Q4 04/04/2013

```
function [areaVM, areaAR]=quesito4(punti)
  x=linspace(-1,4,10^4);
  plot(x,f(x));

  %metodo del valor medio
  xR=rand(1,punti)*5 -1;
  y=f(xR);
  areaVM=mean(y)*(1- -4);

  %metodo accetta rifiuta
  xAr=rand(1,punti)*5 -1;
  yCal=f(xAr);
  yAr=rand(1,punti)*(6 - -3) -3;

  pos = (xAr >0 & xAr<=2) | (xAr > pi & xAr<4);
  neg = (xAr <=0) | (xAr >2 & xAr<=pi);
  percPos = sum(pos & (yCal >= yAr) & (yAr >= 0)) / sum(pos);
  percNeg = sum(neg & (yCal <= yAr) & (yAr <= 0)) / sum(neg);
  areaRetPos = (2 - 0) * 9 + (4 - pi) * 9;
  areaRetNeg = (0 - -1) * 9 + (pi - 2) * 9;
  areaAR1 = areaRetPos * percPos - areaRetNeg * percNeg;

areaAR = areaAR1;
end

function y=f(x)
  yp = polyval([1 -5 2 8],x);
  ys = sin(x);
  y=yp.*ys;
end
```

Valori generati casualmente

```
 neg=0; pos=0;
  areaNeg=0; areaPos=0;
  for i=1:length(xAr)
      if (-1<=xAr(i) && xAr(i)<=0) || (2<=xAr(i) && xAr(i)<=pi)
          if yAr(i)<=0 && yAr(i) > yCal(i)
              areaNeg=areaNeg+1;
          end
          neg=neg+1;
      else
          if yAr(i) >=0 && yAr(i) <= yCal(i)
              areaPos=areaPos+1;
          end
          pos=pos+1;
      end
  end
  areaNeg = ((0 - -1)+(pi -2))*9*areaNeg/neg;
  areaPos = ((2 - 0)+(4-pi))*9*areaPos/pos;
  areaAR2 = areaPos - areaNeg;
```

Modo alternativo per il calcolo

Q4 04/04/2013

Richiamando la funzione precedente 10 volte e mediando sia ha:

```
q=[];
for i=1:10
    [x y]=quesito4(10^5);
    q=[q;[x y]];
end
mean(q)
```

```
ans = 4.0263    4.0433
```

Usando un metodo di integrazione numerica classico:

```
integral(@(x)(polyval([1 -5 2 8],x).*sin(x)),-1,4)
        ans = 4.0399
```