

THE TI-NSPIRE PROGRAMS

JAMES KEESLING

The purpose of this document is to list and document the programs that will be used in this class. For each program there is a screen shot containing an example and a listing of the TN-Nspire CX CAS program. The student is responsible to enter each program and be familiar with its use.

1. SOLVING $f(x) = 0$

In this section are the Newton-Raphson method and the Bisection method.

```
newtonraphson(x2-2,1,10)
```

1.00000000000
1.50000000000
1.41666666667
1.41421568627
1.41421356237
1.41421356237
1.41421356237
1.41421356237
1.41421356237
1.41421356237
1.41421356237
1.41421356237

Done

```
Define newtonraphson(f,a,n)=Prgm
  newMat(n+1,1)→soln
  d/dx(f)→df
  x←f/df→g
  approx(a)→soln[1,1]
  For i,1,n
    g|x=soln[i,1]→soln[i+1,1]
  EndFor
  Disp soln
EndPrgm
```

FIGURE 1. The Newton-Raphson method applied to $x^2 - 2 = 0$

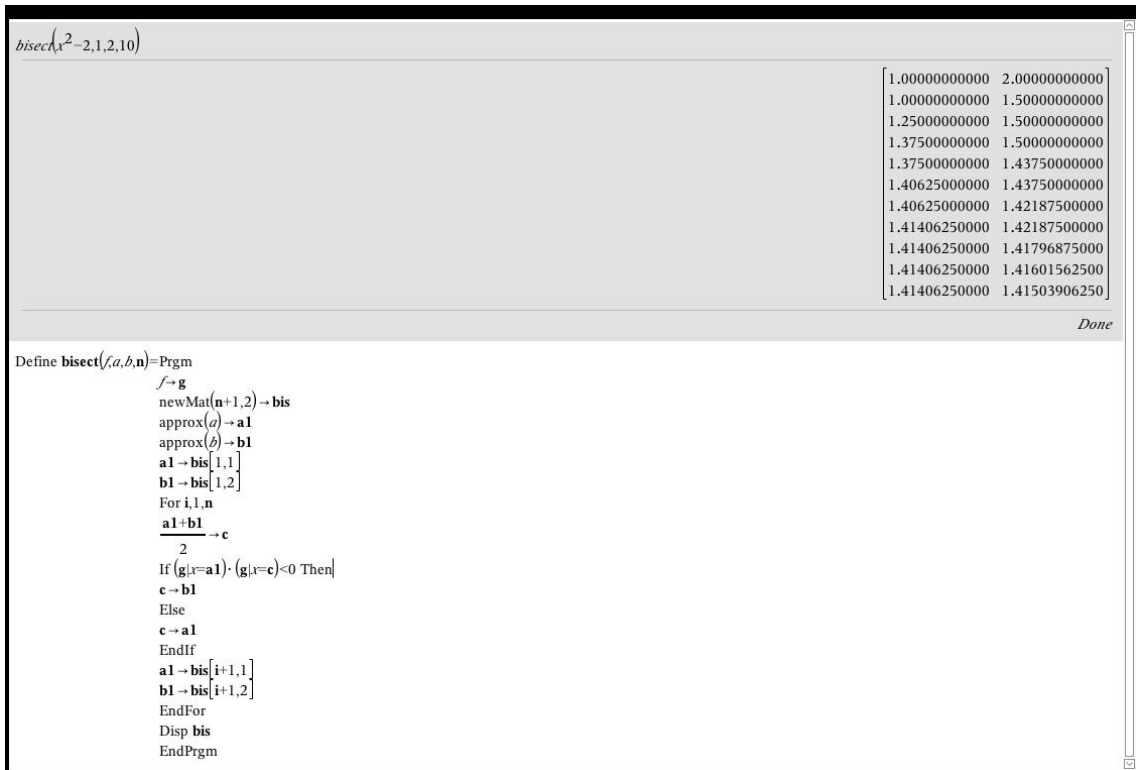


FIGURE 2. The Bisection method applied to $x^2 - 2 = 0$

2. LAGRANGE POLYNOMIALS

In this section are the programs for the VanderMonde matrix and the Lagrange polynomial through a set of points $\{(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)\}$. Note that the polynomial program calls for the points in the form $[x_0, x_1, \dots, x_n]$ and $[y_0, y_1, \dots, y_n]$.

```

vandermonde([1 2 3])

```

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}$$

Done

```

Define vandermonde(a)=Prgm
  dim(a)[2]-1 → n
  newMat(n+1,n+1) → vander
  For i,1,n+1
  For j,1,n+1
  If a[1,i]=0 and j=1 Then
  1 → vander[i,j]
  Else
  a[1,i]j-1 → vander[i,j]
  EndIf
  EndFor
  EndFor
  Disp vander
  EndPrgm

```

FIGURE 3. Program for the VanderMonde matrix for the points $\{1, 2, 3\}$

```

lagrange([0 1 3],[1 0 2])

```

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 3 & 9 \end{bmatrix}$$

$$\frac{2 \cdot x^2}{3} - \frac{5 \cdot x}{3} + 1$$

Done

```

Define lagrange(a,b)=Prgm
  DelVar x
  dim(a)[2]-1 → n
  vandermonde(a)
  newMat(n+1,1) → coef
  vander-1 · bT → coef
  0 → p
  For i,0,n
  p+coef[i+1,1] · xi → p
  EndFor
  Disp p
  EndPrgm

```

FIGURE 4. The Lagrange polynomial through the points $\{(0, 1), (1, 0), (3, 2)\}$

```

newtonpolynomial([0 1 3],[1 0 2])

```

$$\begin{bmatrix} 1 & -1 & \frac{2}{3} \\ 0 & 1 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

$$\frac{(x-1) \cdot (2 \cdot x-3)}{3}$$

Done

```

Define newtonpolynomial(a,b)=Prgrm
  dim(a)[2]-1 → n
  newMat(n+1,n+1) → coef
  For i,1,n+1
    a[1,i] → coef[i,1]
  EndFor
  For j,1,n
    For i,1,n-j+1
      coef[i+1,j]-coef[i,j]
      a[1,i+j]-a[1,i] → coef[i,j+1]
    EndFor
  EndFor
  0 → p
  For j,1,n+1
    coef[1,j] → temp
    For i,1,j-1
      temp·(x-a[1,i]) → temp
    EndFor
    p+temp → p
  EndFor
  Disp coef
  Disp p
EndPrgrm

```

FIGURE 5. The Newton polynomial determined by the points $\{(0, 1), (1, 0), (3, 2)\}$

3. NUMERICAL INTEGRATION

In this section are programs to compute the normalized Newton-Cotes coefficients and to estimate an integral using Newton-Cotes integration. We also have programs for Romberg integration and Gaussian integration. These last programs are greatly superior to Newton-Cotes. For Gaussian quadrature we theoretically need the Legendre polynomials. We give a program that will list the first $n + 1$ of these polynomials from degree 0 to degree n .

```

newtoncotes(5)

```

1	0	0	0	0	0
1	1	1	1	1	1
1	2	4	8	16	32
1	3	9	27	81	243
1	4	16	64	256	1024
1	5	25	125	625	3125
19					
288					
25					
96					
25					
144					
25					
144					
25					
96					
19					
288					

Done

```

Define newtoncotes(n)=Prgm
  newMat(n+1,n+1)→vander
  newMat(1,n+1)→temp
  For i,0,n
    i→temp[1,i+1]
  EndFor
  vandermonde(temp)
  For i,0,n
     $\frac{n^{i+1}}{i+1}$ →temp[1,i+1]
  EndFor
  (vander)-1·temp→coef
   $\frac{1}{n}$ ·coef→coef
  n
  Disp coef
EndPrgm

```

FIGURE 6. Program for the Newton-Cotes coefficients

```

romberg(sin(x),5,0,π)

```

0.000000000000	2.09439510239	1.99857073182	2.00000554998	1.99999999459	2.000000000000
1.57079632679	2.00455975498	1.99998313095	2.0000001629	2.000000000000	0
1.89611889794	2.00026916995	1.9999975245	2.00000000006	0	0
1.97423160195	2.00001659105	1.99999999619	0	0	0
1.99357034377	2.00000103337	0	0	0	0
1.99839336097	0	0	0	0	0

Done

```

Define romberg(/(n,a,b)=Prgm
  f→g
  approx(a)→a1
  approx(b)→b1
  newMat(n+1,n+1)→romb
  For i,0,n
     $\frac{b1-a1}{2 \cdot 2^i} \left( (g(x=a1) + (g(x=b1) + 2 \cdot \sum_{j=1}^{2^i-1} (g(x=a1 + \frac{j \cdot (b1-a1)}{2^i}))) \right)$ →romb[i+1,1]
  EndFor
  For j,1,n
    For i,1,n-j+1
       $\frac{4^j \cdot romb[i+1,j] - romb[i,j]}{4^j - 1}$ →romb[i,j+1]
    EndFor
  EndFor
  Disp romb
EndPrgm

```

FIGURE 7. Program for Romberg Integration

```

gaussianquad(f(x),0,pi)
2.00000000000
Done

Define gaussianquad(f(x),a,b)=Prgm
  approx(a)→a1
  approx(b)→b1
  [0.96028985649754 -0.79666647741363 -0.52553240991633 -0.18343464249565 0.18343464249565 0.52553240991633 0.79666647741363 0.96028985649754]→points
  [0.10122853629037 0.22238103445337 0.31370664587789 0.36268378337836 0.36268378337836 0.31370664587789 0.22238103445337 0.10122853629037]→weights
  0→p
  For i,1,8
  For j,1,8
  p1← $\frac{b1-a1}{2}$ ·weights[i,j]+ $\frac{b1-a1}{2}$ ·points[i,j]
  Disp p
  EndFor
EndPrgm

```

FIGURE 8. Program for Gaussian quadrature

```

legendre(5)

$$\begin{bmatrix} 1 \\ x \\ \frac{3 \cdot x^2 - 1}{2} \\ x \cdot \frac{(5 \cdot x^2 - 3)}{2} \\ \frac{35 \cdot x^4 - 30 \cdot x^2 + 3}{8} \\ x \cdot \frac{(63 \cdot x^4 - 70 \cdot x^2 + 15)}{8} \end{bmatrix}$$

Done

Define legendre(n)=Prgm
  newMat(n+1,1)→poly
  For i,1,n+1
   $\frac{1}{2^{i-1} \cdot (i-1)!} \cdot \frac{d^{i-1}}{dx^{i-1}} \left( (x^2-1)^{i-1} \right)$ →poly[i,1]
  EndFor
  Disp poly
EndPrgm

```

FIGURE 9. Program for the Legendre polynomials up to degree n

gaussiancoef(3)

$$\frac{1-x^2}{3x^2-1} \cdot \frac{x^2-3}{2x}$$

1.00000000000	-0.774596669241	0.600000000000
1	0.00000000000	0.00000000000
1.00000000000	0.774596669241	0.600000000000
[-0.774596669241 0.00000000000 0.774596669241]		
[0.555555555556 0.88888888889 0.555555555556]		

Done

```

Dfne gaussiancoef(n)=Prgm
  legendf(n)
  newMat(1,n)→points
  newMat(1,n)→temp
  newMat(1,n)→coef
  For i,1,n
    If mod(i,2)=0 Then
      0→temp[1,i]
    Else
       $\frac{2}{i}$ →temp[1,i]
    EndIf
  EndFor
  zeros(poly[n+1,1],x)→zer
  For i,1,n
    approx(zer[i])→points[1,i]
  EndFor
  vandermonde(points)
  (vander)-1·temp→coef
  coef→coef
  Disp points
  Disp coef
EndPrgm

```

FIGURE 10. Program for n points and n coefficients for Gaussian quadrature

4. NUMERICAL DIFFERENTIATION

In this section we give the programs needed for numerical differentiation of a function $f(x)$. There are two of these programs. The first program determines the coefficients to be used in estimating $f^{(k)}(a)$ using the $n+1$ points $\{a - m_0 \cdot h, a - m_1 \cdot h, \dots, a - m_n \cdot h\}$. In the programs $b = [m_0, m_1, \dots, m_n]$

```

numdiff(sin(x),1,[-1 2 3],1)

```

$$\begin{bmatrix} 1 & -1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}$$

$$\begin{bmatrix} -5 \\ \frac{12 \cdot h}{2} \\ \frac{3 \cdot h}{-1} \\ \frac{4 \cdot h}{12 \cdot h} \end{bmatrix}$$

$$p = \frac{-\sin(3 \cdot h+1)}{4 \cdot h} + \frac{2 \cdot \sin(2 \cdot h+1)}{3 \cdot h} + \frac{5 \cdot \sin(h-1)}{12 \cdot h}$$

```

Define numdiff(f,a,b,k)=Prgm
  dim(b)[2]-1 → m
  numerdif(b,k)
  0 → p
  For i,0,m
  p+coef[i+1,1]:f(x=a+h[1,i+1]:h → p
  EndFor
  Disp "p=",p
  EndPrgm

```

FIGURE 11. Program determining the coefficients to be used in estimating the k th-derivative at a using the points $a - h \cdot b$ with $b = [m_0, m_1, \dots, m_n]$

```

numerdif([-1 2 3],1)

```

$$\begin{bmatrix} 1 & -1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}$$

$$\begin{bmatrix} -5 \\ \frac{12 \cdot h}{2} \\ \frac{3 \cdot h}{-1} \\ \frac{4 \cdot h}{12 \cdot h} \end{bmatrix}$$

```

Define numerdif(a,k)=Prgm
  dim(a)[2]-1 → n
  newMat(n+1,n+1) → vander
  newMat(n+1,1) → coef
  newMat(n+1,1) → temp
  vandermonde(a)
  k! → temp[k+1,1]
  k! → temp[k+1,1]
  (vander)-1 · temp → coef
  Disp coef
  EndPrgm

```

FIGURE 12. Program giving the formula to estimate $f^{(k)}(a)$ using the points $a - h \cdot b$ with $b = [m_0, m_1, \dots, m_n]$

5. DIFFERENTIAL EQUATIONS

In this section we give some programs useful for solving ordinary differential equations. We give a theoretical solution based on Picard iteration and numerical methods based some method of integration. We also give a program for the Taylor method.

picard(x,0,1,5)

$$1$$

$$\frac{x^2}{2} + 1$$

$$\frac{x^4}{8} + \frac{x^2}{2} + 1$$

$$\frac{x^6}{48} + \frac{x^4}{8} + \frac{x^2}{2} + 1$$

$$\frac{x^8}{384} + \frac{x^6}{48} + \frac{x^4}{8} + \frac{x^2}{2} + 1$$

$$\frac{x^{10}}{3840} + \frac{x^8}{384} + \frac{x^6}{48} + \frac{x^4}{8} + \frac{x^2}{2} + 1$$

Done

```

Define picard(a,b,n)=Prgm
  newMat(n+1,1)→pic
  b→pic[1,1]
  For i,1,n
    b+∫abf(x=pic[i,1]) and f=z dz→pic[i+1,1]
  EndFor
  Disp pic
EndPrgm

```

FIGURE 13. Program giving the Picard iteration method

```

taylorMethod(x,0,1,10,1/10)

r r x r^2 x+x r(2+3) x (A+6 r^2+3) x r(A+10 r^2+15) x (r^6+15 r^4+45 r^2+15) x r(r^6+21 r^4+105 r^2+105) x (r^8+28 r^6+
r^10
840 384 48 8 2
+ + + + + 1
0.00000000000 1.00000000000
).100000000000 1.00501252086
).200000000000 1.02020134003
).300000000000 1.04602785991
).400000000000 1.08328706767
).500000000000 1.13314845307
).600000000000 1.19721736312
).700000000000 1.27762131320
).800000000000 1.37712776434
).900000000000 1.49930250006
1.000000000000 1.64872127070

```

FIGURE 14. Example using the Taylor method

```

Define taylorMethod(c,c,d,t,n) = Prgm
newMat(1,t+1) -> coef
newMat(n+1,2) -> soln
approx(c) -> a
approx(d) -> h
x -> coef[1,1]
a -> soln[1,1]
b -> soln[1,2]
f -> coef[1,2]
For i,2,t
coef[i,1] = f(x) -> g
g(x) = f(x) -> g
d/dx (g) -> g
g(d/dx (g)) -> g
g(d/dx (g)) -> g
g(d/dx (g)) -> g
g(d/dx (g)) -> g
EndFor
For i,1,n
b -> temp
For j,1,t
temp = (coef[i,j+1] * x - b) / (x - a) - temp
EndFor
temp -> b
a = d -> a
a -> soln[i+1,1]
b -> soln[i+1,2]
EndFor
x -> p
DelVar z
For j,1,t
p = (coef[1,j+1] - p) / (x - a)
EndFor
p/c and a = d -> p
p/c -> p
taylor(p,t,c) -> p
Disp coef
Disp p
Disp soln
EndPrgm

```

FIGURE 15. Program for the Taylor method

```
eulermethod(x,0,1,1/10,10)
```

0.00000000000	1.00000000000
0.10000000000	1.00000000000
0.20000000000	1.01000000000
0.30000000000	1.03020000000
0.40000000000	1.06110600000
0.50000000000	1.10355024000
0.60000000000	1.15872775200
0.70000000000	1.22825141712
0.80000000000	1.31422901632
0.90000000000	1.41936733762
1.00000000000	1.54711039801

Done

```
Define eulermethod(f,a,b,h,n)=Prgm
  newMat(n+1,2)→soln
  approx(a)→a
  approx(b)→b
  approx(h)→h
  a→soln[1,1]
  b→soln[1,2]
  For i,1,n
    b+h: f=f-a and x=b→b
    a+h→a
    a→soln[i+1,1]
    b→soln[i+1,2]
  EndFor
  Disp soln
EndPrgm
```

FIGURE 16. Program giving the Euler method

```
modifiedeuler(x,0,1,1/10,10)
```

0.00000000000	1.00000000000
0.10000000000	1.00500000000
0.20000000000	1.02015037500
0.30000000000	1.04590917197
0.40000000000	1.08306509530
0.50000000000	1.1327778318
0.60000000000	1.19663813070
0.70000000000	1.27675305355
0.80000000000	1.37586100934
0.90000000000	1.49748712256
1.00000000000	1.64615015665

Done

```
Define modifiedeuler(f,a,b,h,n)=Prgm
  approx(a)→a
  approx(b)→b
  approx(h)→h
  newMat(n+1,2)→soln
  a→soln[1,1]
  b→soln[1,2]
  For i,1,n
    f_x=b and t=a→temp
    b+h: f=f-a*(h/2) and x=b+(h/2): temp→b
    a+h→a
    a→soln[i+1,1]
    b→soln[i+1,2]
  EndFor
  Disp soln
EndPrgm
```

FIGURE 17. Program giving the modified Euler method

```

heun( $f, x, 0, 1, \frac{1}{10}, 10$ )

```

0.0000000000	1.0000000000
0.1000000000	1.0050000000
0.2000000000	1.0201755000
0.3000000000	1.04598594015
0.4000000000	1.08322303962
0.5000000000	1.13305129944
0.6000000000	1.19706869786
0.7000000000	1.27739200749
0.8000000000	1.37677310567
0.9000000000	1.49875520283
1.0000000000	1.64788134551

Done

```

Define heun( $f, a, b, h, n$ )=Prgm
  approx( $a$ )→ $a$ 
  approx( $b$ )→ $b$ 
  approx( $b$ )→ $h$ 
  newMat( $n+1, 2$ )→ $soln$ 
   $a$ → $soln[1,1]$ 
   $b$ → $soln[1,2]$ 
  For  $i, 1, n$ 
     $f$ → $b$  and  $f$ → $a$ → $temp$ 
     $b$ → $\frac{h}{2}$ → $(temp, f$ → $a+h$  and  $x=b+h$ → $temp$ )→ $b$ 
     $a+h$ → $a$ 
     $a$ → $soln[i+1,1]$ 
     $b$ → $soln[i+1,2]$ 
  EndFor
  Disp  $soln$ 
EndPrgm

```

FIGURE 18. Program giving the Heun method

```

rungekutta( $f, x, 0, 1, \frac{1}{10}, 10$ )

```

0.0000000000	1.0000000000
0.1000000000	1.00501252083
0.2000000000	1.02020133976
0.3000000000	1.04602785889
0.4000000000	1.08328706482
0.5000000000	1.13314844612
0.6000000000	1.19721734741
0.7000000000	1.27762127947
0.8000000000	1.37712769490
0.9000000000	1.49930236245
1.0000000000	1.64872100705

Done

```

Define rungekutta( $f, a, b, h, n$ )=Prgm
  approx( $a$ )→ $a$ 
  approx( $b$ )→ $b$ 
  approx( $b$ )→ $h$ 
  newMat( $n+1, 2$ )→ $soln$ 
   $a$ → $soln[1,1]$ 
   $b$ → $soln[1,2]$ 
  For  $i, 1, n$ 
     $f$ → $b$  and  $f$ → $a$ → $k1$ 
     $f$ → $a$ → $\frac{h}{2}$  and  $x=b$ → $\frac{h}{2}$ → $k1$ → $k2$ 
     $f$ → $a$ → $\frac{h}{2}$  and  $x=b$ → $\frac{h}{2}$ → $k2$ → $k3$ 
     $f$ → $a+h$  and  $x=b$ → $k3$ → $k4$ 
     $b$ → $\frac{h}{6}$ → $(k1+2$ → $k2+2$ → $k3+k4)$ → $b$ 
     $a+h$ → $a$ 
     $a$ → $soln[i+1,1]$ 
     $b$ → $soln[i+1,2]$ 
  EndFor
  Disp  $soln$ 
EndPrgm

```

FIGURE 19. Program giving the Runge-Kutta method

6. STOCHASTIC SIMULATION

In this section we do not show copies of the programs. They can now be downloaded. So, there is no need for them to be given to be copied into your calculator. However, we do give examples of how the data is to be entered when the programs are run and what the typical output will look like and how it should be interpreted.



FIGURE 20. Program giving an example of the Bowling program. The output is a sequence of scores for a bowler whose probability of a strike, spare, and open frame are the numbers entered in that order.

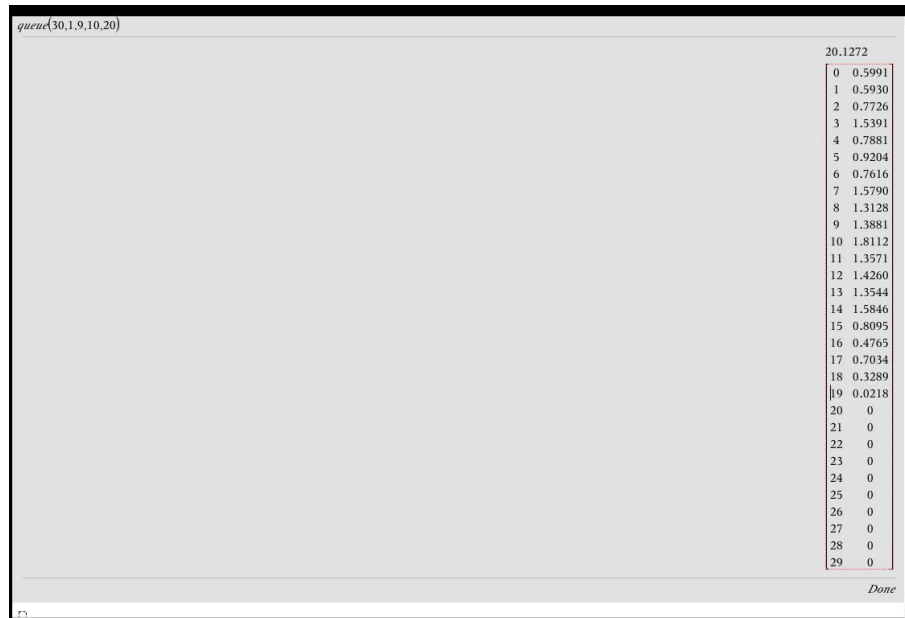


FIGURE 21. Program giving an example of the Queue simulation program. The output gives the time spent in each state given the arrival rate, the service rate, and the number of servers.

The image shows a TI-NSPIRE calculator screen with two windows. The left window, titled 'montecarlo(x^2,2,3,100,20)', displays a list of 20 numerical results. The right window, titled 'montecarlo', shows the program code and a mathematical formula for the Monte Carlo method.

Left Window (Results):

```
montecarlo(x^2,2,3,100,20)
6.38409578628
6.55582420933
6.33720914386
6.32641550558
6.51017338415
6.31826192235
6.28002781073
6.30095033517
6.36871245521
6.53176307021
6.46528902262
6.20875318153
6.23029782653
6.27355442341
6.29106980588
6.30986276208
6.57884629686
6.12190154735
6.67891514886
6.17380987873
Done
```

Right Window (Program Code):

```
montecarlo 3/5
Define montecarlo(a,b,n,k)=
Prgm
newMat(k,1)→s
For i,1,k
  1/n ∑_{j=1}^n ((b-a)·rand()→a)·(b-a)→s[i,1]
EndFor
Disp s
EndPrgm
```

FIGURE 22. This program estimates the integral $\int_a^b dx$ by the Monte Carlo method using n random points in the interval $[a, b]$. It gives k estimates of the integral.