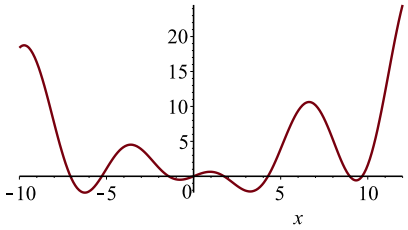
Problem 1

First I initilize the formulas that are general and used for all 3 runs.

$f := x \to 0.1 \cdot x^2 + x \cdot \cos(x)$:

$plot(f(x), x = -10..12)$



$g := (x, T) \to \exp\left(-\frac{f(x)}{T}\right)$:

$gs := (\mathrm{mu}, q) \to \frac{1}{q \cdot \mathrm{sqrt}(2 \cdot \mathrm{Pi})} \cdot \exp\left(-\frac{(x - \mathrm{mu})^2}{2 \cdot q^2}\right)$:

$rr := x \to stats[random, uniform[-10, 12]](1)$:

$printlevel := 0$:

Now I do calculations and plots for each of three cases

Temp=3

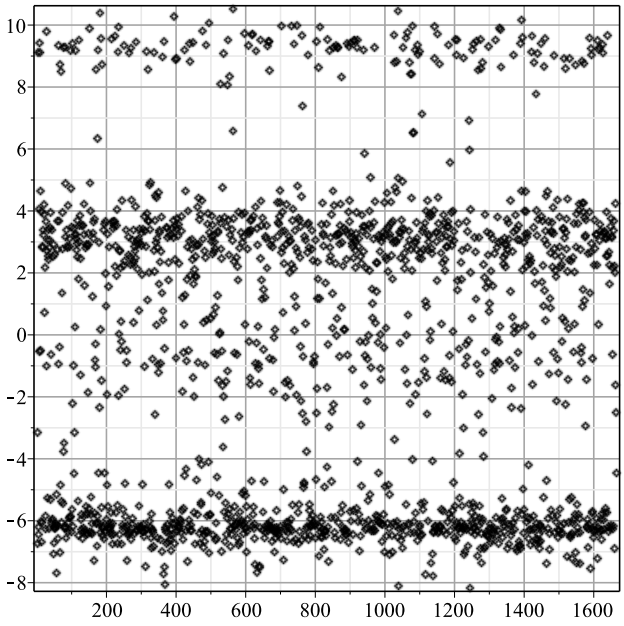
1) The Loop for metropolis monte carlo algoritm where we create the list of the points we need.

```
x0 := evalf(rr( )) :
rlist := [x0] :
 for i from 1 to 20000  do
   xp := evalf(rr( ));
   if evalf(f(xp)) < f(rlist[1])  then
     rlist := [xp, op(rlist)];
     else
       rt := evalf(abs(rr( )));
       if  g(xp, 3) / g(rlist[1], 3) > rt then rlist := [xp, op(rlist)] else rt := "rt"
      end if
     end if
  end do
N := nops(rlist)
```

1665 **(1)**

2)create a sequence of the algoritm points so we can plot and see the how are the points distributed. Helpes to decide the boundaries for 4 lists (groups or minima that we are looking for). Also create a plot of the histagram to see the statistical distibution. We will fit gaussian on that histogram.

```
aa := [seq([i, rlist[i]], i = N..1, -1)] :
with(plots) :
pointplot(aa, connect = false, thickness = 0, axes = boxed, gridlines = true)
```




```
with(Statistics) :
P := Histogram(rlist, axes = boxed, view = [ -10 .. 12 , default], labels = [typeset("x"),
     typeset("count") ], labeldirections = ["horizontal", "vertical"], colour = red , gridlines = true,
     bincount = 50, frequencyscale = relative)
```

PLOT(...) **(2)**

3)A loop, that separates out rlist into the 4 lists according to the 4 minima that we are lookig at and expecting. Boundaries determined by looking at the point plot. We also found probabilies of each minima by deviding the number of point of each separate list by the total number of points. This should give us a good estimate and a good starting point for gaussian fitting.

```
list1 := [ ] :
 list2 := [ ] :
 list3 := [ ] :
 list4 := [ ] :
for i from 1 to N do
   if rlist[i] ≤ -4 then
     list1 := [rlist[i], op(list1) ];
   elif 1.5 ≥ rlist[i] > -4 then
     list2 := [rlist[i], op(list2) ];
   elif 7 ≥ rlist[i] > 1.5 then
     list3 := [rlist[i], op(list3) ];
   else
```

```
    list4 := [rlist[i], op(list4)];
  end if
 end do
N1 := nops(list1):
N2 := nops(list2):
N3 := nops(list3):
N4 := nops(list4):
```

$$Prob1 := evalf\left(\frac{N1}{N}\right)$$

$$0.3885885886 \quad \textbf{(3)}$$

$$Prob2 := evalf\left(\frac{N2}{N}\right)$$

$$0.1549549550 \quad \textbf{(4)}$$

$$Prob3 := evalf\left(\frac{N3}{N}\right)$$

$$0.3669669670 \quad \textbf{(5)}$$

$$Prob4 := evalf\left(\frac{N4}{N}\right)$$

$$0.08948948949 \quad \textbf{(6)}$$

4) We use the statistics to find the mean and standart deviation of the 4 lists and use these parameters to create the gaussian curve for these points. Coefficients k1 through k4 are the multiplication of the gaussian and the sum off these coefficients should give us 1 becuase we did out histogram as relative frequency. We create all four plots for the regions that it supposed to cover and display it together with the histogram

```
with(Statistics):
SD1 := StandardDeviation(list1):
M1 := Mean(list1):
k1 := 0.39:
gs1 := k1·gs(M1, SD1):
SD2 := StandardDeviation(list2):
M2 := Mean(list2):
k2 := 0.15:
gs2 := k2·gs(M2, SD2):
SD3 := StandardDeviation(list3):
M3 := Mean(list3):
k3 := 0.38:
gs3 := k3·gs(M3, SD3):
SD4 := StandardDeviation(list4):
M4 := Mean(list4):
k4 := 0.08:
gs4 := k4·gs(M4, SD4):
k1 + k2 + k3 + k4
```

$$1.00 \quad \textbf{(7)}$$

```
GS1 := plot(gs1, x = -10 .. -4, color = blue)
```

$$PLOT(\ldots)$$

```
GS2 := plot(gs2, x = -4 ..1.5, color = blue)
```

$$PLOT(\ldots) \quad \textbf{(9)}$$

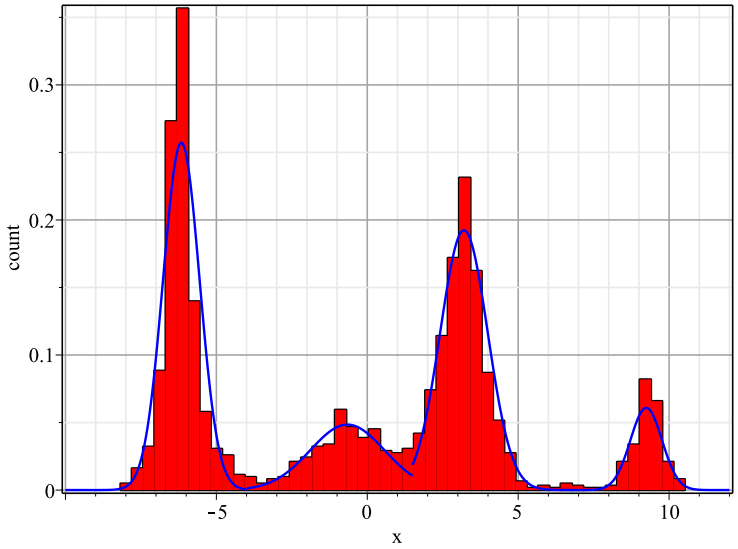
$GS3 := plot(gs3, x = 1.5..7, color = blue)$

$PLOT(\ldots)$ **(10)**

$GS4 := plot(gs4, x = 7..12, color = blue)$

$PLOT(\ldots)$ **(11)**

$display(P, GS1, GS2, GS3, GS4)$



We observe how it fits the histogram nicely. The area under the gaussian curve should be equa to the area of the bins. It is hard to estimate exact and we see there are little disagreemens between the probabilities we found by just points ratio and the gaussian coefficients. Also 2 and 3 gaussians overlap which also is part of the error, becuase they both take part of the same area (even though it is a small part).
Also I want to point out how there we can see how 2 peaks dominate and other 2 have a very small probability.

Now we repeat the same process but for other to cases (smaller temperature of 1, and 0.3) .

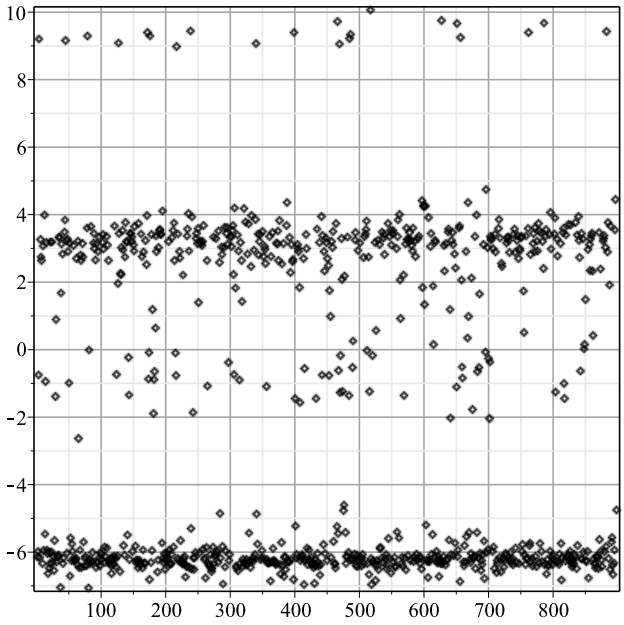
Temp=1

```
x0 := evalf(rr( )) :
rlist := [x0] :
 for i from 1 to 25000 do
   xp := evalf(rr( ));
   if evalf(f(xp)) < f(rlist[1]) then
     rlist := [xp, op(rlist)];
     else
       rt := evalf(abs(rr( )));
```

```
      if g(xp, 1)/g(rlist[1], 1) > rt then rlist := [xp, op(rlist)] else rt := "rt"
      end if
    end if
  end do
N := nops(rlist)
```

898 **(12)**

```
aa := [seq([i, rlist[i]], i = N..1, -1)]:
pointplot(aa, connect = false, thickness = 0, axes = boxed, gridlines = true)
```




```
P := Histogram(rlist, axes = boxed, view = [-10 .. 12, default], labels = [typeset("x"),
     typeset("count")], labeldirections = ["horizontal", "vertical"], colour = red, gridlines = true,
     bincount = 50, frequencyscale = relative)
```

PLOT(...) **(13)**

```
list1 := []:
list2 := []:
list3 := []:
list4 := []:
for i from 1 to N do
```

```
    if rlist[i] ≤ -4 then
      list1 := [rlist[i], op(list1)];
    elif 1.5 ≥ rlist[i] > -4 then
      list2 := [rlist[i], op(list2)];
    elif 7 ≥ rlist[i] > 1.5 then
      list3 := [rlist[i], op(list3)];
    else
      list4 := [rlist[i], op(list4)];
    end if
  end do
N1 := nops(list1):
N2 := nops(list2):
N3 := nops(list3):
N4 := nops(list4):
Prob1 := evalf(N1/N)
```

$$0.5155902004 \quad \textbf{(14)}$$

```
Prob2 := evalf(N2/N)
```

$$0.08017817372 \quad \textbf{(15)}$$

```
Prob3 := evalf(N3/N)
```

$$0.3808463252 \quad \textbf{(16)}$$

```
Prob4 := evalf(N4/N)
```

$$0.02338530067 \quad \textbf{(17)}$$

```
SD1 := StandardDeviation(list1):
M1 := Mean(list1):
k1 := 0.53:
gs1 := k1·gs(M1, SD1):
SD2 := StandardDeviation(list2):
M2 := Mean(list2):
k2 := 0.07:
gs2 := k2·gs(M2, SD2):
SD3 := StandardDeviation(list3):
M3 := Mean(list3):
k3 := 0.375:
gs3 := k3·gs(M3, SD3):
SD4 := StandardDeviation(list4):
M4 := Mean(list4):
k4 := 0.025:
gs4 := k4·gs(M4, SD4):
k1 + k2 + k3 + k4
```

$$1.000 \quad \textbf{(18)}$$

```
GS1 := plot(gs1, x = -10..-4, color = blue)
```

$$PLOT(\ldots) \quad \textbf{(19)}$$

```
GS2 := plot(gs2, x = -4..1.5, color = blue)
```

$$PLOT(\ldots) \quad \textbf{(20)}$$

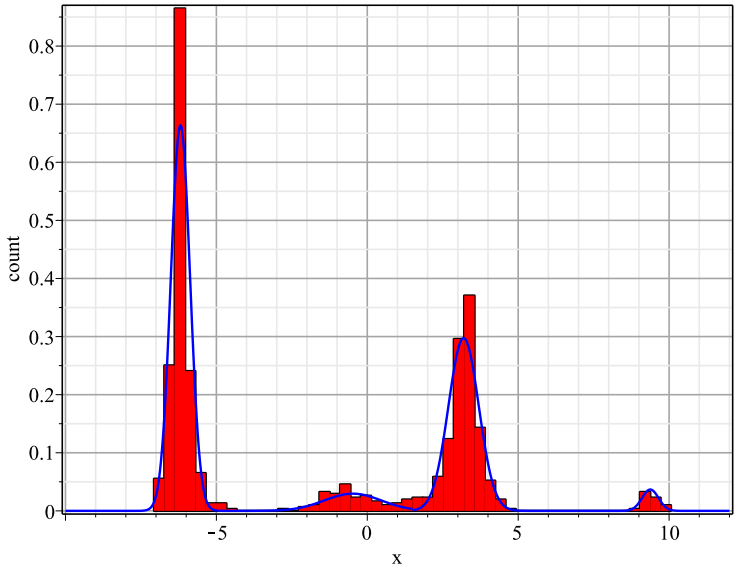
$GS3 := plot(gs3, x = 1.5..7, color = blue)$

$PLOT(\ldots)$ **(21)**

$GS4 := plot(gs4, x = 7..12, color = blue)$

$PLOT(\ldots)$ **(22)**

$display(P, GS1, GS2, GS3, GS4)$



The distribution for temperature 1 shows that as we decrease the temperature systems is tend to settle more into the first minima. Second and fourth minima have super small probability.

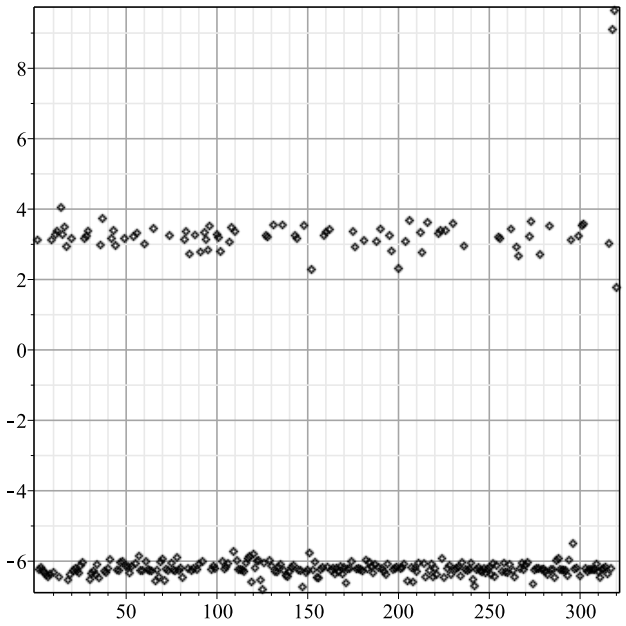
Temp=0.3

```
x0 := evalf(rr( )) :
rlist := [x0] :
 for i from 1 to 25000 do
   xp := evalf(rr( ));
   if evalf(f(xp)) < f(rlist[1]) then
     rlist := [xp, op(rlist)];
     else
       rt := evalf(abs(rr( )));
       if g(xp, 0.3)/g(rlist[1], 0.3) > rt then rlist := [xp, op(rlist)] else rt := "rt"
      end if
     end if
  end do
```

$N := nops(rlist)$

320 **(23)**

$aa := [seq([i, rlist[i]], i = N..1, -1)]:$
$pointplot(aa, connect = false, thickness = 0, axes = boxed, gridlines = true)$



$P := Histogram(\text{rlist}, \text{axes} = \text{boxed}, view = [-10 .. 12, \text{default}], \text{labels} = [typeset(\text{"x"}), typeset(\text{"count"})], labeldirections = [\text{"horizontal"}, \text{"vertical"}], \text{colour} = \text{red}, \text{gridlines} = \text{true}, \text{bincount} = 50, \text{frequencyscale} = \text{relative})$

$PLOT(...)$ **(24)**

$list1 := [\,]:$
$list2 := [\,]:$
$list3 := [\,]:$
$list4 := [\,]:$
for i **from** 1 **to** N **do**
  if $rlist[i] \le -4$ **then**
    $list1 := [rlist[i], op(list1)];$
  elif $1.5 \ge rlist[i] > -4$ **then**
    $list2 := [rlist[i], op(list2)];$
  elif $7 \ge rlist[i] > 1.5$ **then**
    $list3 := [rlist[i], op(list3)];$
  else

```
    list4 := [rlist[i], op(list4)];
  end if
 end do
N1 := nops(list1) :
N2 := nops(list2) :
N3 := nops(list3) :
N4 := nops(list4) :
```

$$Prob1 := evalf\left(\frac{N1}{N}\right)$$

$$0.7375000000 \quad (25)$$

$$Prob2 := evalf\left(\frac{N2}{N}\right)$$

$$0. \quad (26)$$

$$Prob3 := evalf\left(\frac{N3}{N}\right)$$

$$0.2562500000 \quad (27)$$

$$Prob4 := evalf\left(\frac{N4}{N}\right)$$

$$0.006250000000 \quad (28)$$

```
SD1 := StandardDeviation(list1) :
M1 := Mean(list1) :
k1 := 0.735 :
gs1 := k1· gs(M1, SD1) :
SD2 := StandardDeviation(list2) :
Error, (in Statistics:-StandardDeviation) data sample must
contain at least two elements of non-zero weight
M2 := Mean(list2) :
k2 := 0.00001 :
gs2 := k2· gs(M2, SD2) :
SD3 := StandardDeviation(list3) :
M3 := Mean(list3) :
k3 := 0.25499 :
gs3 := k3· gs(M3, SD3) :
SD4 := StandardDeviation(list4) :
M4 := Mean(list4) :
k4 := 0.01 :
gs4 := k4 ·gs(M4, SD4) :
k1 + k2 + k3 + k4
```

$$1.00000 \quad (29)$$

$$GS1 := plot(gs1, x = -10 .. -4, color = blue)$$

$$PLOT(...) \quad (30)$$

$$GS2 := plot(gs2, x = -4 .. 1.5, color = blue)$$

$$PLOT(...) \quad (31)$$

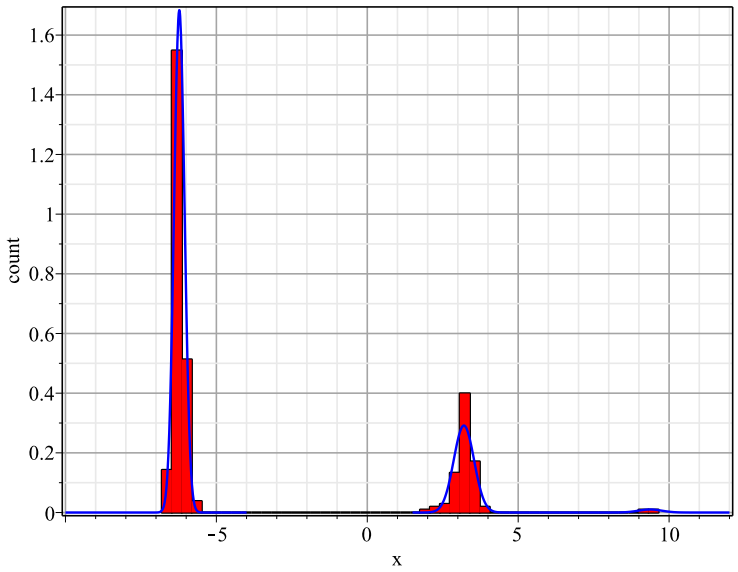
$$GS3 := plot(gs3, x = 1.5 .. 7, color = blue)$$

$$PLOT(...) \quad (32)$$

$$GS4 := plot(gs4, x = 7 .. 12, color = blue)$$

$$PLOT(...) \quad (33)$$

```
display(P, GS1, GS2, GS3, GS4)
```



For a very small temperature we observe how the system basically settles in the first minima. If we keep decreasing the temp we would be all in first minima. There are pretty much 0 probbabiliyt for 2 and 4 minimas and small probabiliyt for 3rd minima.
Note - most of the time there are not enough points in the 2nd and 4th lists to do statistical analysis so it gives error saying that needs at least 2 points. But that's fine just leaeve it as error and we know that is probability 0 and so we do not see it on plot. I gave it probability 0.00001 just so there is a tiny chance just in case at some random run there is a point.

Problem 2

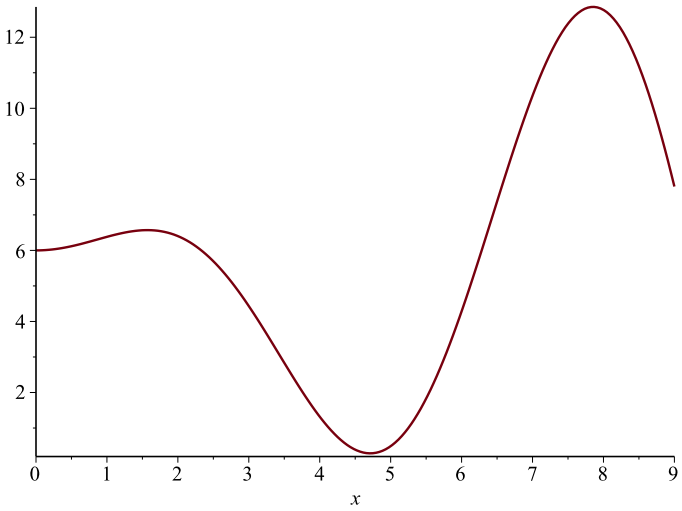
1) We plot the function to see what it lookes like and what are the bounds of the "grid" where we have to generate the points.

restart :

$f := x \to (5 + \cos(x) + x \cdot \sin(x))$

$$x \to 5 + \cos(x) + x \sin(x) \quad \textbf{(34)}$$

$plot(f(x), x = 0..9)$



2)We know that we want integral from 0 to 9. On the plot we see that y values will be from 0 to 13. So we need 2 different random numbers generators to create X and Y values.

$ExactValue := evalf(\ int(f(x), x = 0..9)\ );$

54.02440933 **(35)**

$with(stats):$
$with(plots):$
$with(Statistics):$
$with(StringTools):$
$X := x \rightarrow stats[random, uniform[0, 9]](1):$
$X();$

3.561469744 **(36)**

$Y := y \rightarrow stats[random, uniform[0, 13]](1):$
$Y();$

2.510817613 **(37)**

3) We create the loop where we see if the randomly generated point with coords X,Y is below or above the function. Then we find the estimated area under the curve by deviding the number of points under the curve over the total points. This ratio is scaled in between 0 and 1 so to find out integral (area) we have to multiply it by the total area of the "grid" (grid area = maxX*maxY = 9*13=117).
Then we run this process for 20 times and count good and bad cases, where good is when estimate is within 0.5 of the exact value.

```
CountGood := 0:
CountBad := 0:
for k from 1 to 20 do
  n := 30000:
  Zn := 0:
  for i from 1 to n do:
```

```
    x := X( ) :
    y := Y( ) :
    if y ≤ f(x) then;
      Zn := Zn + 1 :
    end if;
  end do:
  Estimate := 117· evalf( Zn / n );
  if abs(Estimate − ExactValue) ≤ 0.5 then
    CountGood := CountGood + 1;
  else
    CountBad := CountBad + 1;
  end if:
  x :='x':
end do:
print( CountGood, CountBad, evalf( CountGood / (CountBad + CountGood) ) )
```

$$19,\ 1,\ \frac{19}{20} \tag{38}$$

We find that with using 30000 points we get 0.95 probability (could be something close to it on different runs) that our result is within 0.5 from the exact result. On out print statement first number is count of good estimates , second number is count of bad estimates and third number is the probabiliyt of a good estimate.