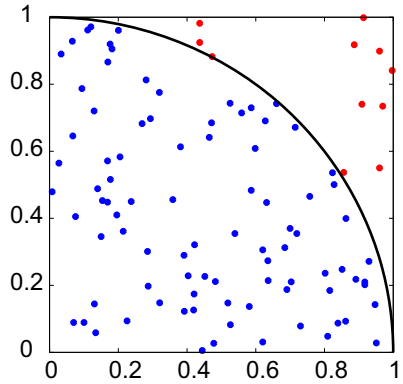
1 予備知識

乱数を用いたシミュレーション手法を一般に Monte Carlo 法と呼び、特に、乱数を用いた数値積分法が有名です。

具体例として、定積分 $\int_0^1 \sqrt{1-x^2}\,dx$ を考えてみましょう。定積分は面積を求める計算ですから、この定積分の場合は、曲線 $y=\sqrt{1-x^2}$ と x 軸とで挟まれた領域、すなわち、四分円の面積が求める値です。そこで、この四分円を含む 1 辺の長さが 1 の正方形を考え、正方形の中に目を瞑って胡麻（あるいは小石）を投げ入れます。今、n 個の胡麻を投げ入れ、その内の k 個が四分円に入ったとします。胡麻が四分円に入る確率は、面積に比例しますから、

$$\frac{k}{n} \fallingdotseq \frac{\text{四分円の面積}}{\text{正方形の面積}} = \text{四分円の面積} = \int_0^1 \sqrt{1-x^2}\,dx$$

成り立ち、従って、不定積分の近似値を求めることが出来ます。



2 円周率の近似値

コンピュータを用いて Monte Carlo 法を実行する場合は、胡麻を投げ入れる操作を、（疑似）乱数によって実現します。定積分 $\int_0^1 \sqrt{1-x^2}\,dx$ の場合は、例えば、0 以上 1 未満の乱数を n 組 (r_x, r_y) 発生させ、不等式 $r_y < \sqrt{1-r_x{}^2}$ を満たすものの個数 k を数えます。このとき、

$$\frac{k}{n} \fallingdotseq \int_0^1 \sqrt{1-x^2}\,dx = \frac{\pi}{4}$$

が成り立ちますから、十分大きな n に対してこの実験を行えば、$\frac{4k}{n}$ が π のよい近似値を与えることが期待されます。

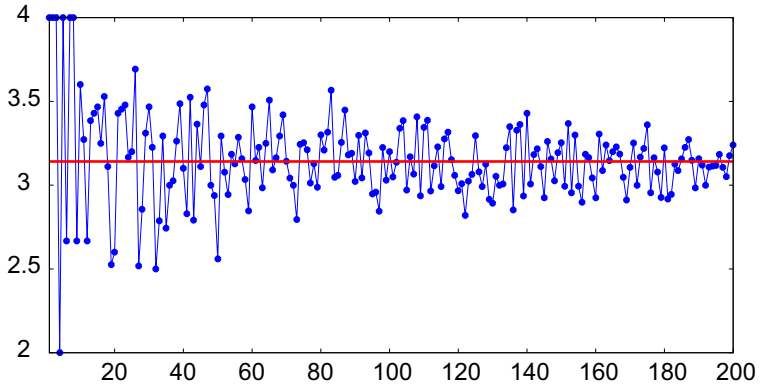
実際に、Maxima を使って円周率 π の近似値を求めてみましょう。なお、下記で定義している関数 `monte(n)` では、不等式 $r_y < \sqrt{1-r_x{}^2}$ の代わりに、不等式 $r_x{}^2 + r_y{}^2 < 1$ を利用しています。

```
(%i1) monte(n) := block([k: 0],
    for i: 1 thru n do if random(1.0)^2 + random(1.0)^2 < 1 then k: k + 1,
    float(4 * k / n)
);
(%o1) monte(n) := block([k : 0], for i thru n
                  2              2                                  4 k
     do if random (1.0) + random (1.0) < 1 then k : k + 1, float(---))
                                                                     n
(%i2) monte(10);
(%o2)                                4.0
(%i3) monte(100);
(%o3)                                3.0
(%i4) monte(1000);
(%o4)                               3.068
(%i5) monte(10000);
(%o5)                              3.1256
```

引数を 10、100、1000、10000 と大きくするにつれて、円周率 π の近似値に近づいていくことが分かります。

近似の様子をより直感的に把握するため、関数 monte(n) のグラフを書いてみたいと思います。

```
(%i6) load("draw")$
(%i7) draw2d(point_type = 7, points_joined = true,
    color = blue, points(makelist(monte(n), n, 1, 200)),
    color = red,  explicit(%pi, n, 1, 200)
);
(%o7)                       [gr2d(points, explicit)]
```



このグラフから分かるように、Monte Carlo 法による数値積分は非常に効率の悪い方法ですが、処理が単純なことから、おおよその値を把握する際には有効な手段です。

③ 分布図

1 ページの分布図は次の関数で書いています。

– monte_graph.mac –

```
1  monte_graph(n) := block([r: [], I: [], O: []],
2      for i: 1 thru n do (
3          r: [random(1.0), random(1.0)],
4          if r[1]^2 + r[2]^2 < 1 then I: cons(r, I) else O:
               cons(r, O)
5      ),
6      draw2d(point_type = 7, point_size = 1, line_width = 2,
7          user_preamble = ["set size ratio 1"],
8          color = blue,   points(I),
9          color = red,    points(O),
10         color = black,  explicit(sqrt(1 - x^2), x, 0, 1)
11     )
12 );
```

四分円に入る点をリスト L に蓄積し、入らない点をリスト O に蓄積し（4 行目）、それらをそれぞれ、青い点と赤い点でプロットしています（8、9 行目）。

7 行目は、グラフを正方形に出力するための設定です。

なお、四分円のグラフは陽関数 $y = \sqrt{1-x^2}$ によって書いていますが（10 行目）、媒介変数表示を利用する方法

```
parametric(cos(t), sin(t), t, 0, %pi/4)
```

の方が適切かもしれません。

作成日；平成 20 年 9 月 15 日～ 9 月 21 日
ソフトウェア：Maxima 5.16.0cvs & CMU Common Lisp Snapshot 2008-09 (19E)