モンテカルロ法で πを求める。

円周率πは、多くの人を引き付ける定数です。円の直径と円周との関係を示すこのπを求めるために、多くの数学者がいろんな方法を考え出してきました。ここでは、乱数を使ったモンテカルロ法で円周率πを求めてみることにしましょう。

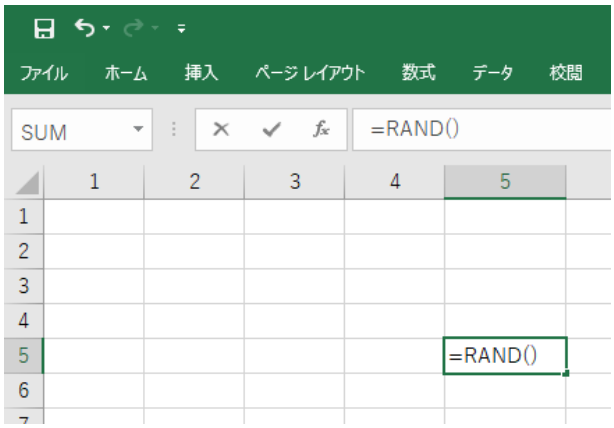
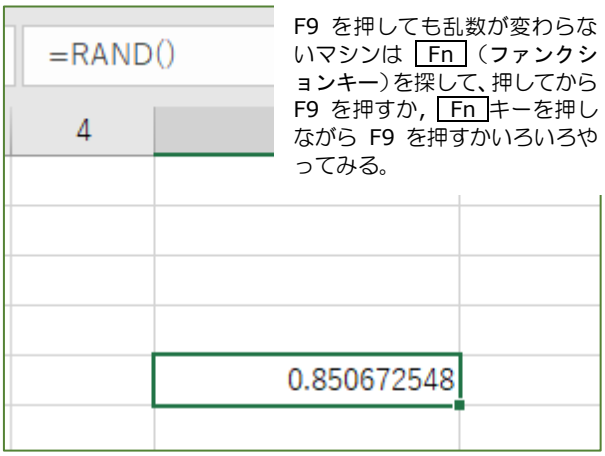


図 4-1



F9 を押しても乱数が変わらないマシンは Fn（ファンクションキー）を探して、押してから F9 を押すか，Fn キーを押しながら F9 を押すかいろいろやってみる。

図 4-2

乱数を発生させる

エクセルの中で乱数を発生させてみましょう。乱数は英語で Random number といいますので、表計算のセルの中に乱数を発生させるときは半角で

=RAND()

と書いてエンターキーを押すと乱数が発生します。でも「間違った？」と思うかもしれません。エクセルで発生する乱数は

0<RAND()<1

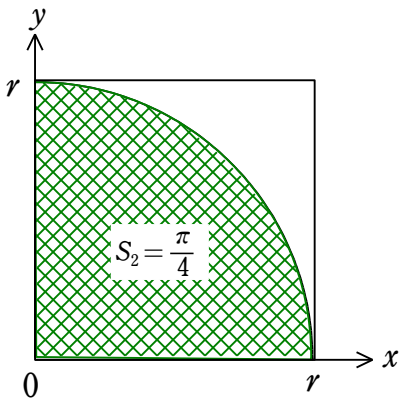
で小数点以下 9 桁の乱数です.
変な乱数と思うかもしれませんが、これがプログラムを書くとき効果を発揮するのです。

再計算するときは、「**数式**」タブを開いて一番右にある「**再計算実行**」を押します。面倒くさいなあと思う人は、「F9」キーを押せば再計算し、違う乱数が出てきます（はよ言わんかい！）。「F9」を押して何回かサイコロを振ってみましょう。まず、同じ数字になることはありません。

※**プログラムコード**で書くときは

RND()

と A がなくなります。なんで？



グラフ１

理論

グラフ1のように一辺 r の正方形の面積 S_1 は

$$S_1=r^2$$

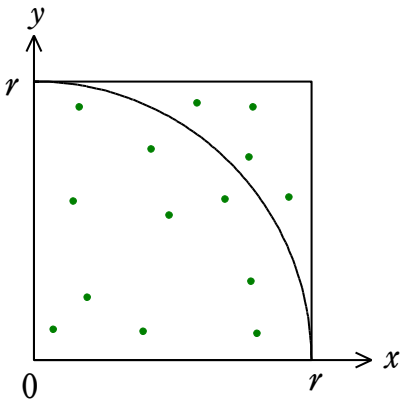
半径 r の円の $\frac{1}{4}$ の円弧を考える

円の中心 O と円弧が作る扇型の面積 S_2 は

$$S_2=\frac{1}{4}\pi r^2$$

よって

$$\frac{S_2}{S_1}=\frac{\pi}{4} \quad \cdots \quad ①$$



グラフ２

x , y を$0<x<1$, $0<y<1$ の乱数として

乱数 (x,y) の組を N_1 個発生させたとき

扇型の内側に N_2 が入ったとすると

確率とπ

モンテカルロ法を使って円周率πの値を求める方法の理論的な説明をします。まず左の**グラフ１**を見てください。半径ｒの四分の一の円と、一辺がｒの正方形が書かれています。

このときそれぞれの面積を求めて面積の比をとると①式のように四分のπとなって、ｎで表すことができます。

それでは、この**グラフ１**の正方形内に、**グラフ２**のように N_1 個の乱数による点を発生させたらどうなるでしょう。乱数による点は、数が多くなればなるほど正方形内に均一にばらまかれる状態になっていくことが予想されますね。

すると、四分の一の扇型の中に入っている乱数による点の数 N_2 の割合は、扇型の面積 S_2 に比例していそうですね。

つまり正方形全体にある乱数による点の数 N_1 のうち、扇型の中に入った点の数 N_2 の割合は、それぞれの面積に比例していると考えるわけです。

すると②式にように、なんと点の数の比で円周率πが表せることになります。（ほんとかよー！）

N_1 が十分に大きいときは

扇型の内側に乱数の組が入る確率 p は

面積に比例すると考えて

$$p=\frac{N_2}{N_1}=\frac{S_2}{S_1}=\frac{\pi}{4}$$

すると

$$\pi=\frac{4N_2}{N_1} \quad \cdots \quad ②$$

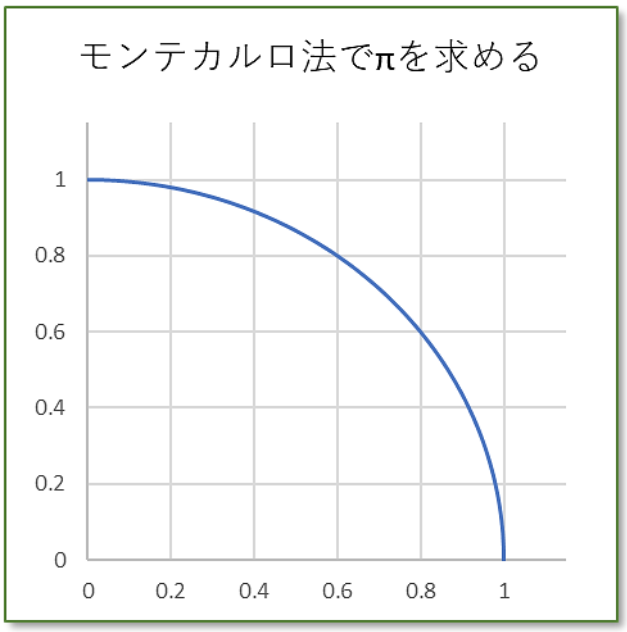


図 4-3

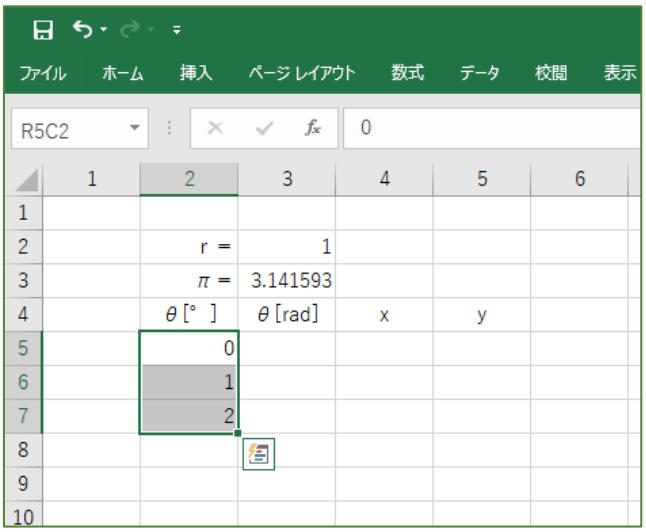


図 4-4

実験ラボの作成

初めに新しいシートに書いてみた＝RAND()　を消去してください。

そのシートに**図 4-3** のような、サイコロを振るための舞台をまず作ることにしましょう。

このエクセルブックの名前を「**モンテカルロ法によるπの導出**」として、「**EXCEL マクロ有効ブック**」で保存しておきます。

図 4-4 のように、円の半径 r を 1 として、円周率πの値を適当に書いておきます。ここではπを 3.1415926 とかおいていますが、これは、円の曲線を書くだけのもので、実際の乱数の境界条件とは関係がありませんので安心してください。

円周を形成する x，y のデータを 0 度から 90 度まで 1 度刻みで作ります。まず**図 4-5** でセル (5,3) に左隣の角度をラジアンに変換する計算式を書きます。

ラジアンにはどうやって直すのだったか思い出してください。

実はエクセルにはラジアンで書く命令もあるのですが、このテキストは、もっともシンプルな所から最も複雑なことができるというポリシーで書かれていますので、興味のある人はネットで「**エクセル VBA　ラジアン**」として検索すればすぐ出てくると思います。

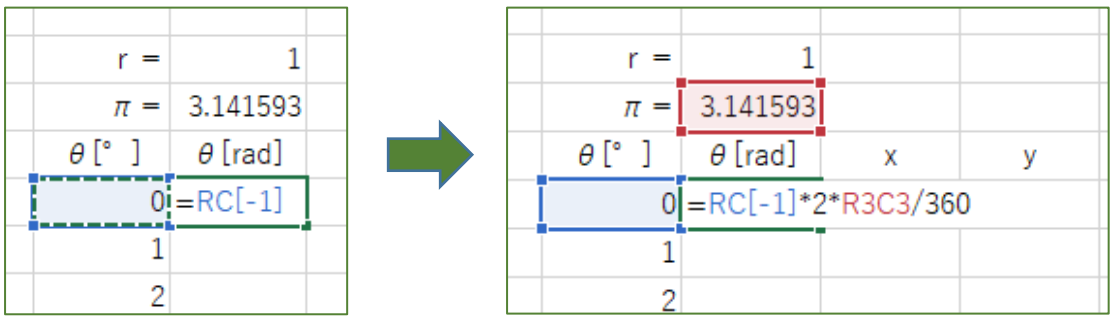


図 4-5

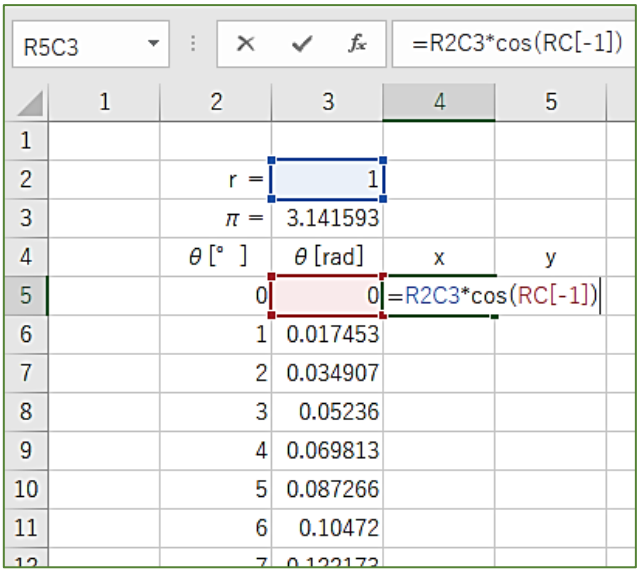


図 4-6

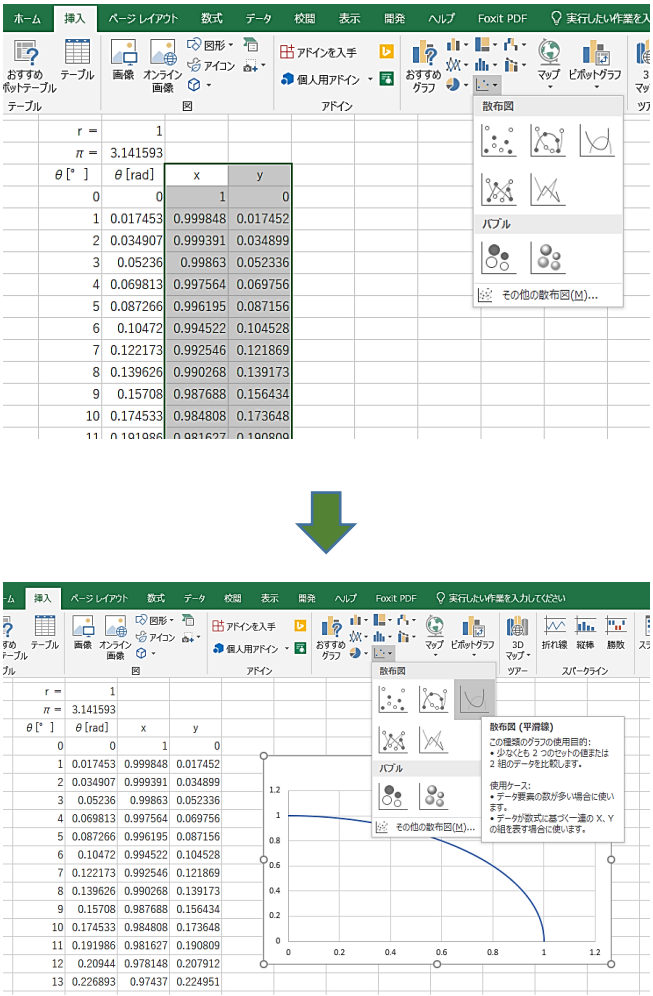


図 4-7

あとは 90°までフィルハンドルして完成させましょう。

次に円の曲線を形成する x，y ですが、

x=rcosθ

y=rsinθ

として**図 4-6** のように出だすことができますね。最初の θ が 0 の行で x，y の式を書いたら θ=90°まで**フィルハンドル**してください。次に、**図 4-7** のように x と y の組をドラッグし、「**挿入**」タブの「**散布図**」からラインだけのものを選んで**図 4-7** の下のようなグラフを作ります。

ここですでに気付いているとは思いますが、セルの座標が縦横数字になっていますね。

これは、これからプログラムを書くためにしている基本設定です。

「ああ、プログラムコードを書いていくんだな」と気が付いて設定しましたか。

忘れていた人は、Excel の「**ファイル**」の一番下にある「**オプション**」を選択して、「**数式**」の中の「**数式の処理**」の

☐ R1C1 参照形式を使用する

にチェックを入れておきましょう。

さて、次に**図 4-8** にあるようなグラフまで持っていきましょう。座標は最大 1.05 までにしてあります。すると 1 以上の数値がグラフに入らず、シンプルになります。

また、縦横が正方形に近くなるよう調整しましょう。

図 4-8 に近くなればまず「**実験ラボ**」の第 1 段階の完成です。

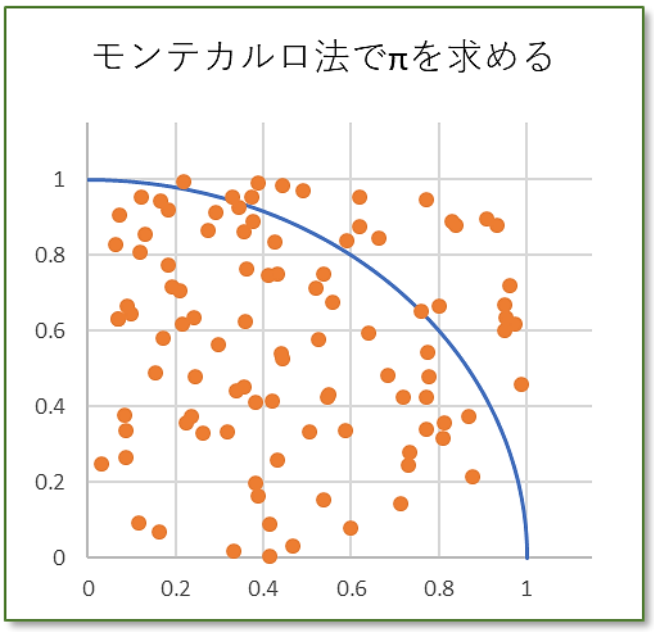


図 4-8

乱数をプロット

次に図 4-8 のように乱数を 100 個発生させて、グラフにプロットされるようにしましょう。これは、この計算に大きなミスがないことが分かるチェック機能にもなっています。

できれば、サイコロを振るたびに乱数の点が増えていくようにするともっと面白そうです。

まず、図 4-9 のようにこのグラフの右側に、乱数の一覧を書く表の各データの名前を書きます。また、サイコロを振った数を n として 1 から 100 までを縦にフィルハンドルを使って書いておきます。

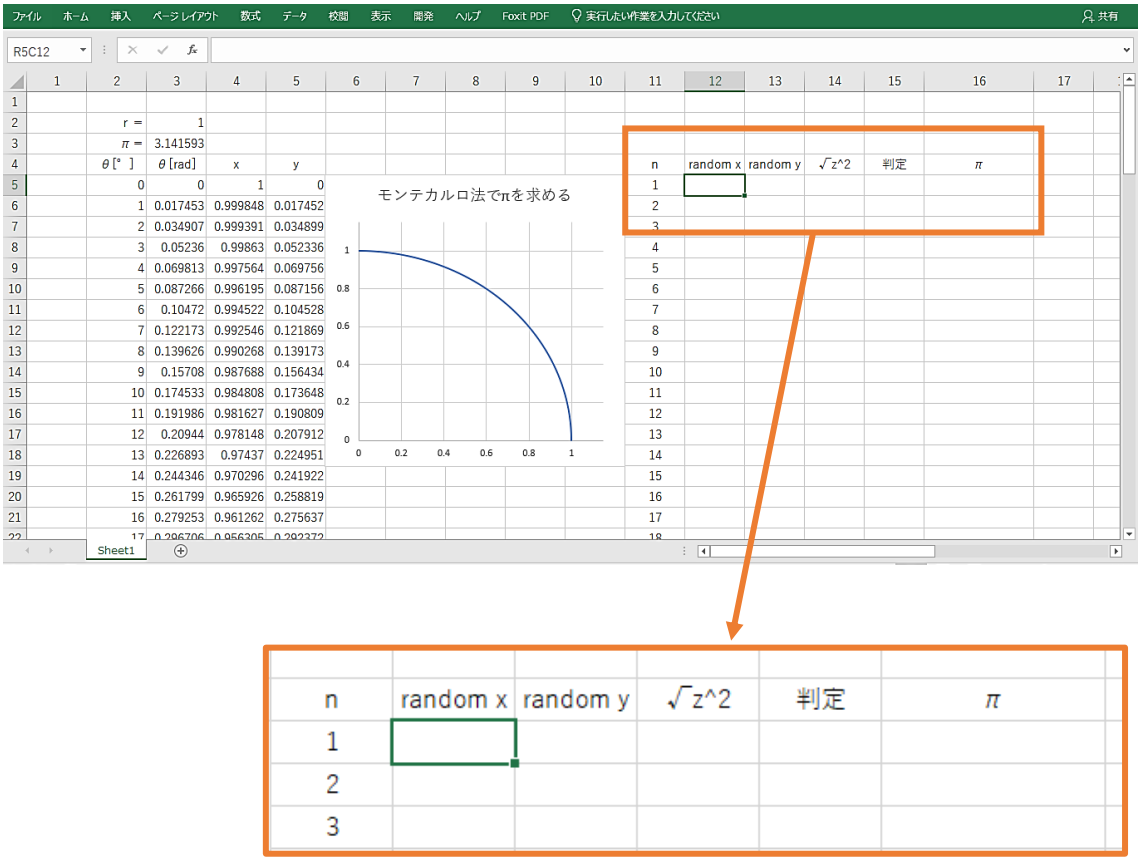


図 4-9

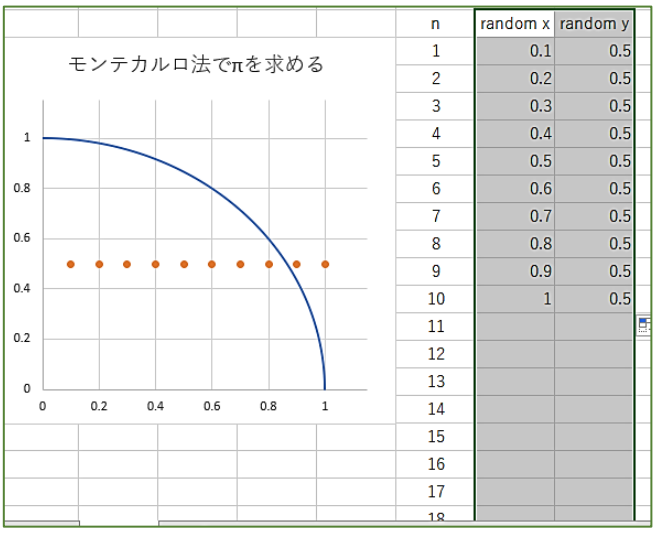


図 4-10-a

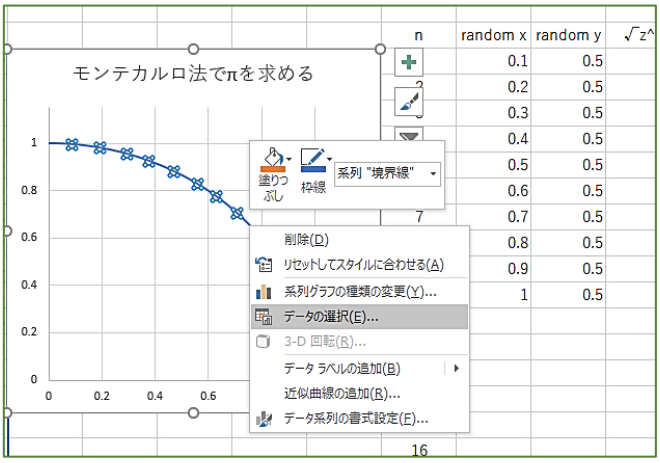


図 4-10-b

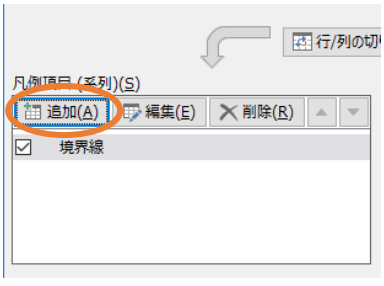


図 4-10- c

乱数が点で出てこない場合は、グラフのデータを右クリックして「**系列グラフの種類の変更**」で乱数を点で表わすように変更します。

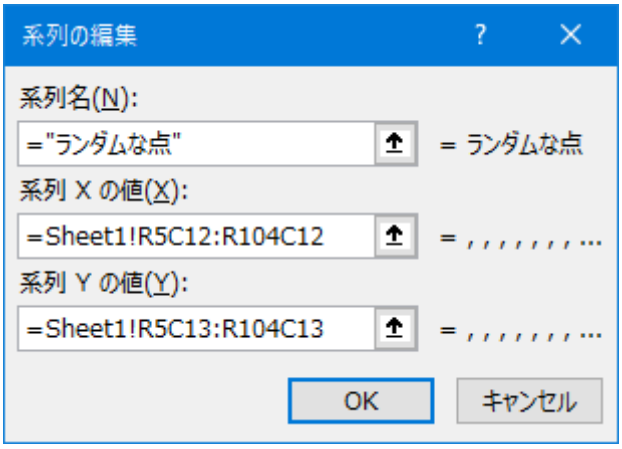


図 4-10-d

乱数をグラフに

図 4-10-a のように random x と random y の欄に乱数が発生したら、グラフに点として出てくるようにします。今回、実際の乱数は、プログラムコードで発生させるようにします。

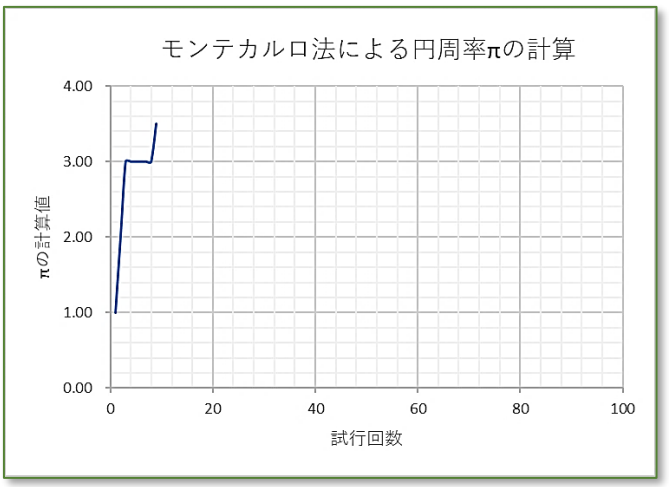
まず図 4-10-b のように random x と random y の欄にダミー（偽物）となるデータを適当に 10 個ほど入れておきます。

次にグラフの四分の一円のラインを左クリックしてアクティブにし、右クリックして、図 10—c のように「**データの選択**」をいうメニューを出し「**追加**」を選びます。

そして、「データの選択」を左クリックして「**系列の編集**」の中に乱数の n=1 から 100 までのセットが入るようにします。random x と random y の欄をそれぞれ「**系列 X の値**」と「**系列 Y の値**」として

をそれぞれ押し n=100 までドラッグします。図 4-10-d のようにしてダミーの点が図 4-10-a のようにグラフに現れたら「OK」です。

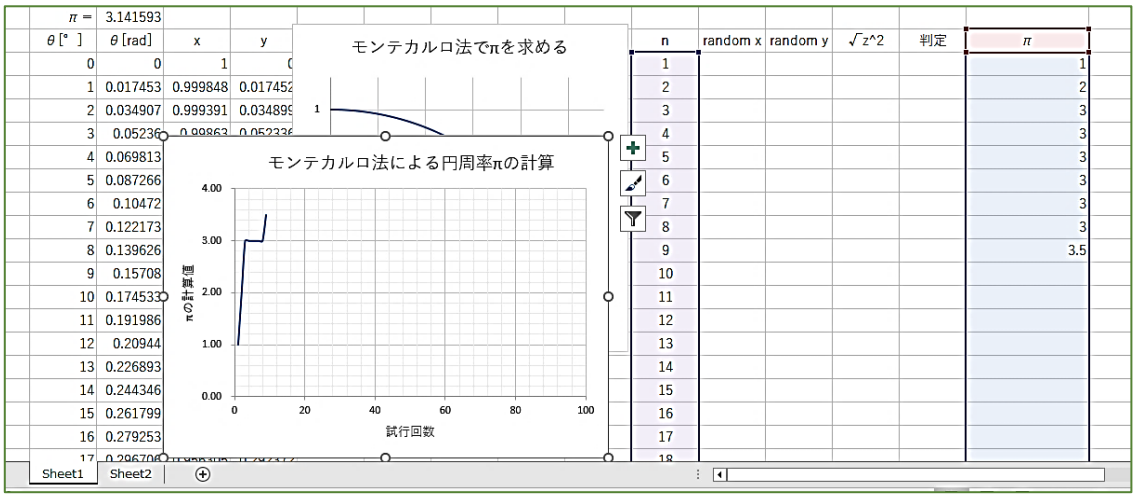
これで、この欄に乱数が表れてもグラフでチェックすることができますね。



πのグラフ作成

もう一つ**図 4-11** に示したグラフを作っておきます。これは、乱数を発生させたとき計算されるπの値がサイコロを振った回数（試行回数 n ）でどのように変化するかを視覚的にわかるようにするグラフです。

これは自分の力で作れますね。縦軸、横軸に軸の量の名前を忘れないようにしてください。「**散布図**」のプロットを結んだ線だけのものがわかりやすいと思います。



これで実験ラボの部屋の完成

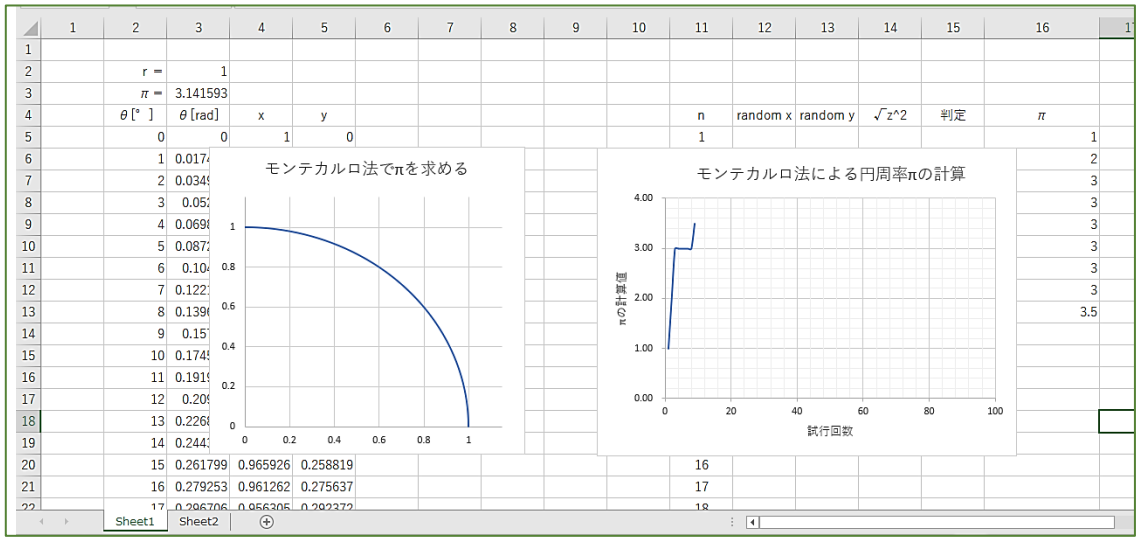


図 4-11

Dim n As Integer
これは今から書くプログラムコードで使う n という変数は正の整数(Integer)しかとりませんという断り書きです。

```
Dim n As Integer

For n = 1 To 100
    [          ]
Next n

End Sub
```

※プログラムコードで乱数を発生させるときは
RND()
と書きます。表の計算式と違って A がなくなります。理由はマイクロソフト社にいらっしゃるエクセルの設計者にお聞きください。

プログラムを書く

それでは**プログラムコード**を書いていきましょう。「**開発**」タブから Visual Basic をクリックしてコードを書く白い紙を出します。

コードは、誰でも最初からごちゃごちゃと書けるわけではありません。一つ一つ動くかどうか確認しながら書いていきます。

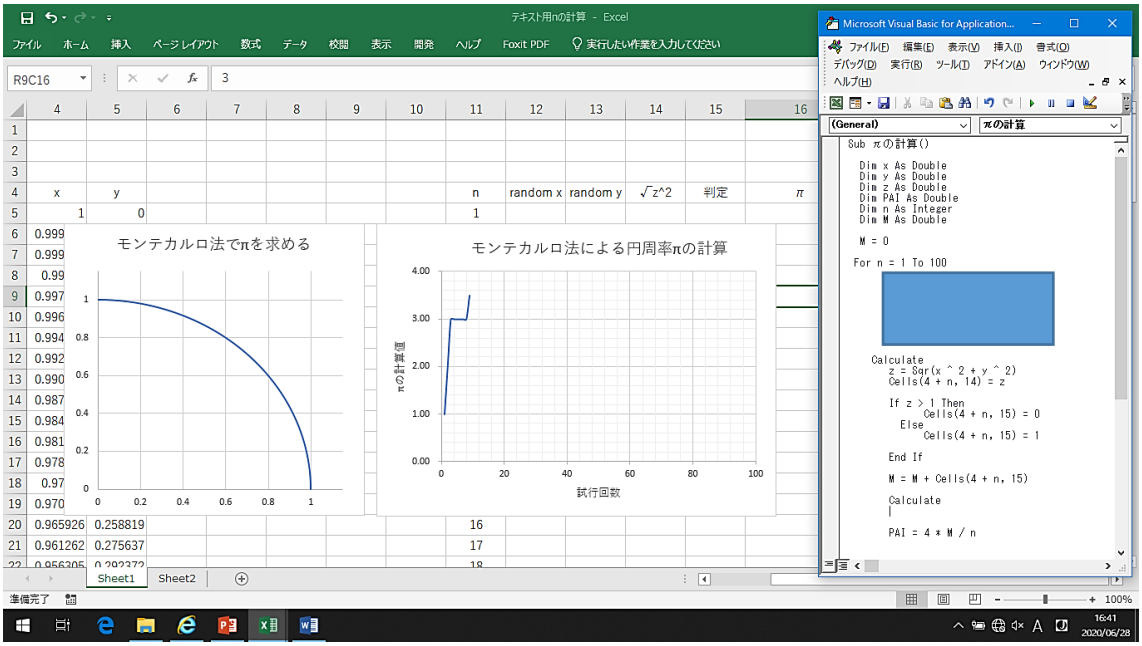
テキストのコードを全部写して、動いた、終わりというようなタイプの人はプログラミングは無理です。ちょっとしたプログラミングも人の工夫が入っており、「人の思考」に興味がある人でないと大成しません。人工知能を生みだした科学者たちは、脳の思考方法を何とかコンピュータに反映させようと 70 年以上努力をしてきたのです。そしてディープラーニングを生み出します。「**ブロック崩し**」で少し自信がついた人は、まず四分の一円のグラフに、乱数が出てくるように自分で考えてコードを書いてみましょう。ただし RND()に気を付けて！

わからない人は一緒にやっていきますから安心してください。

方法①

一つの方法として、n をうまく使って、セル(5,12)→セル(5, 13)→セル(6, 12)→・・・→セル(104,13)まで次々と乱数を入れてやる方法があります。

成功すると、四分の一の円を描いたグラフに、乱数の点が 100 個出てくるはずです。



```
Sub πの計算()
  Dim n As Integer
  For n = 1 To 100
      Cells(4 + n, 12) = Rnd()
      Cells(4 + n, 13) = Rnd()
      DoEvents
  Next n
End Sub
```

次の段階へ行く

アルゴリズム（考えの順番） 1

①解決すべきしい小さな問題を作る
乱数の点のうち円の中に入ったものを数えるには？

②解決すべき解を順番に考えていく
セル(4+n,12）に入っている x 座標の乱数を x、
セル(4+n,13)に入っている y 座標の乱数を y
とおく。円の中に入っているというのは
$X^2 + y^2 < 1^2$
ということだから、このことを使って入っているかいないかを判定できる。

もし $\sqrt{}$（$X^2 + y^2$） < 1 ならその乱数の点は円内に入っている、つまり 1 と判定する。そうでなかったら 0 と判定する。この 1 か 0 を「判定」という列に書く。

これを n=1 から n=100 まで繰りかえす。「判定」という列を全部足すと、100 回中何回入ったかが計算できる。

乱数の点を数える

左のようなプログラムが書けましたか。できたら F8 を使ってうまくプログラムが動くか確かめてみます。n=1,2,3 と n が増えていくと、グラフのランダムな点がプロットされていくように、ここでは

DoEvents

という命令を最後に入れました。これは、一旦、グラフの描画などシート 1 の作業に戻りなさい。という命令です。パソコンの計算速度が速すぎる場合は、DoEvents をもう 1 回書くという裏技もあります。

さて次に、この乱数の点が円の内側に入っていたら 1 と判定し、円の外側にある場合は 0 と判定するプログラムを今書いたコードに付け加えてみましょう。

どんな風に考えていけばいいでしょう。プログラミングが面白いと思う人は、この時、自分で考えてコードを書きながら試行錯誤していくとよいでしょう。このようなプログラムの流れを、「**アルゴリズム**」といいます。いわば「**思考の道のり**」です。この思考の流れが成功するとコンピュータが答えてくれるので、次第にコンピュータが「ちょっと気難しいけど素直ないいヤツ」に思えてきます。これが「**プログラムを考える**」という行為です。

BASIC 講座

セル(4n,12)の値を x に入れる
```
x = Cells(4+n,12).Value
```

$\sqrt{}$（x^2+y^2）を計算して z とする
```
Calculate
z = Sqr(x^2+y^2)
```

Basic 言語ではルート（二乗根）は
Sqr(数または文字式)と書きます。

もし [A] だったら [B] とし
それ以外なら [C] とする

```
If  A  then  B
       Else  C
End if
```

←Sqr スクエアルートという意味

```
Sub π の計算()
 `x,y,z を倍精度の実数として定義
    Dim x As Double
    Dim y As Double
    Dim z As Double
 `n を整数として定義
    Dim n As Integer

 For n = 1 To 100
        Cells(4 + n, 12) = Rnd()
        x = Cells(4 + n, 12).Value
        Cells(4 + n, 13) = Rnd()
        y = Cells(4 + n, 13).Value
        DoEvents
    Calculate
        z = Sqr(x ^ 2 + y ^ 2)
        Cells(4 + n, 14) = z
        If z > 1 Then
                Cells(4 + n, 15) = 0
            Else
                Cells(4 + n, 15) = 1
        End If
 Next n
End Sub
```

アルゴリズム２

①解決すべきしい小さな問題を作る
　途中の n 回目の時に、それまでに円内に入った数を出すにはどうしたらいいのか

⬇

②解決するための解
　それまでの入った数を格納しておく変数 M を作っておき、はじめは M=0 としておく。各回ごとに M にその時の判定の数字を加えて、新しい M としておく。

⬇

　すると n 回試行して M 回円内に入ったことになるので、その時の n の確率としての値が出てくる。

Pai = ４M／n

円内判定

最初に宣言する変数が増えました。このように少しずつプログラムを書きながら、新しい変数をつくったら、それが、実数で**倍精度**（16 桁）なのか、**単精度**（８桁）なのか、**整数**なのか、０と１しかとらない**ブーリアン**なのか書いておく必要があります。わからないときは「**エクセル VBA　宣言文**」とかで検索するとすぐネットが教えてくれます。そうやって言葉の数を増やしていきます。

さて自分の力で IF 文を書くことができましたか。プログラミングとは、IF 文のような制御文が心臓部になっています。**条件判断こそが思考のポイント**なわけです。

円内に入っているということが、その点の x , y 座標の二乗の和のルートが１より小さい、つまり原点からの距離が半径より小さいところにあるというのも大きなアイデアですね。

こうやって少しずつプログラムを作っていきます。そのたびに「F8」を使ってディバッグしてプログラムにミスがないか調べていきます。うまく流れたら、**再生▶　一時停止⏸　リセット■**を使って動かしてみます。

さて次の問題は、サイコロを振ったときに、その時のπの値を計算して、グラフでサイコロを振るたびにπの値がどんなふうに変化していくかが、この実験ラボの目標になっています。

実際、プログラムを作っている人は左の「**アルゴリズム２**」のように考えました。また変数 M が増えますが、これを整数にしなかったのは、最後に数値計算する必要があるからです。それでは、πをサイコロを振るたびに出てくるプログラムを考えてみましょう。

```
Sub πの計算()

  Dim x As Double
  Dim y As Double
  Dim z As Double
  Dim PAI As Double
  Dim n As Integer
  Dim M As Double

  M = 0

 For n = 1 To 100
       Cells(4 + n, 12) = Rnd()
           x = Cells(4 + n, 12).Value
       Cells(4 + n, 13) = Rnd()
           y = Cells(4 + n, 13).Value
       DoEvents
       Calculate
           z = Sqr(x ^ 2 + y ^ 2)
           Cells(4 + n, 14) = z
       If z > 1 Then
                Cells(4 + n, 15) = 0
            Else
                Cells(4 + n, 15) = 1
       End If
           M = M +
       Calculate
           PAI = 4 * M / n
           Cells(4 + n, 16) = PAI
       DoEvents
 Next n
End Sub
```

π を計算する

さあ、君のプログラムはどうなりましたか。きっとつまらないところで詰まったり、失敗したりしている人もいるでしょう。

でも、それがプログラミングの学習では宝物なのです。自分の作りたいイメージを、少しずつ現実のものにしていく面白さを味わってください。

左のプログラムコードが最終形です。新しく変数に M が加わり、はじめ M は 0 になるように設定してありますね。

If 文のあと

M = M +

とわざとブランクにしてあります。考えてみましょう。

プログラムのコードは＝を使ったコードは、数学の＝と違って「M に を加えたものを改めて M とおく。」という意味になります。

問題 4-1

左のコードの空欄を埋めよ。

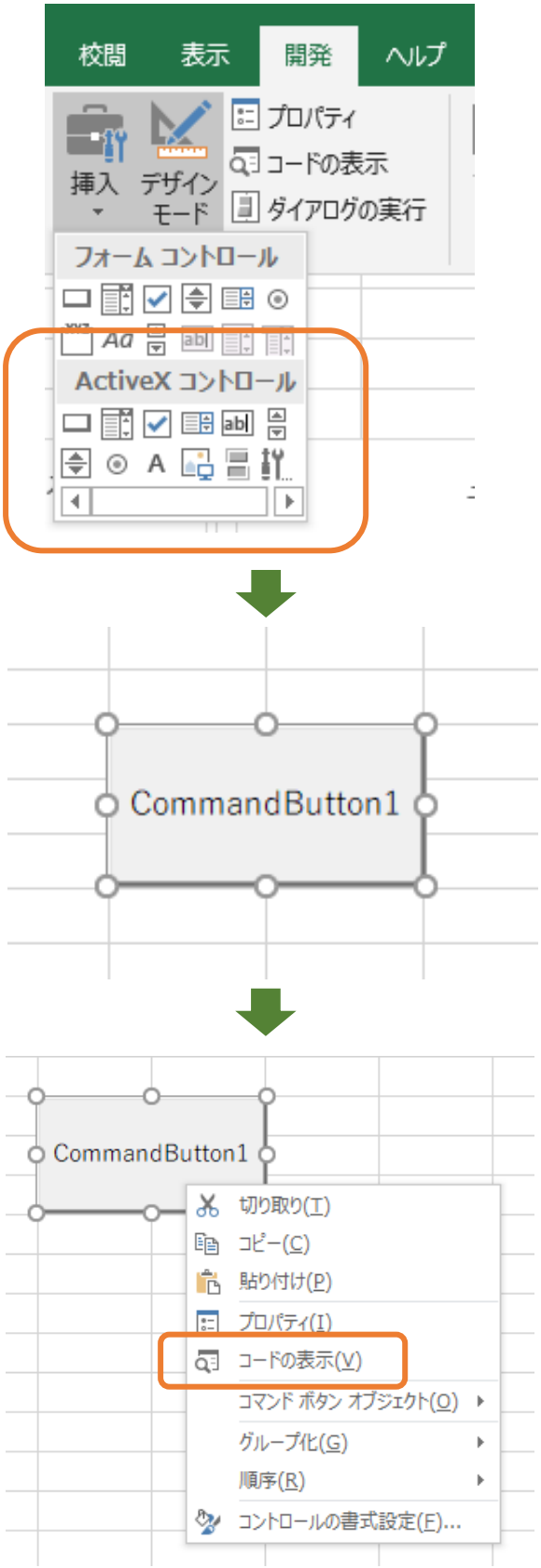


図 4-12

ボタンを設定

これでようやく「**モンテカルロ法でπを求めてみる**」という実験ができるようになりました。

しかし、シート側にスタートするスイッチがないので勝手が悪いですね。スイッチボタンを作りましょう。

図 4-12 の上から順にみていきながら読んでください。

「**開発**」タブを開いて、「**挿入**」をクリックします。すると「**フォームコントロール**」というのと「**ActiveXコントロール**」という名のボタン類のアイコンが出てきます。

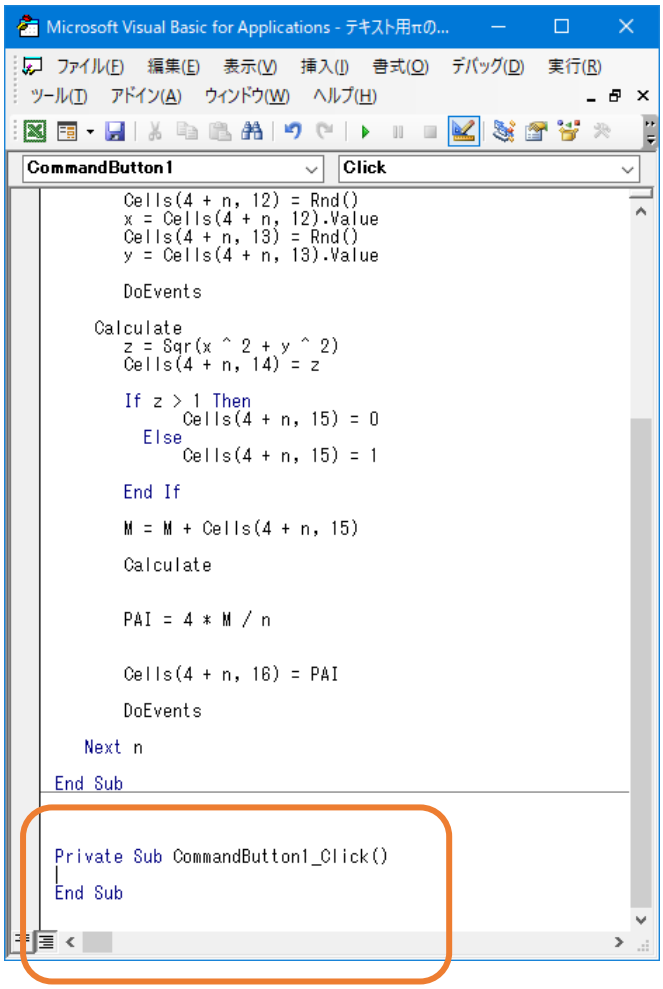
「**ActiveX コントロール**」のボタン類の左上の四角をクリックして、シートの上にドラッグして広げます。

すると「CommandButton1」という名のボタンが現れます。

このボタンを右クリックして「**コードの表示**」をクリックします。

ここで「なんでコントロールボタンのメニューが二種類あるの？」と思ったと思います。「**フォームコントロール**」は「**マクロ用のボタン**」です。マクロというのは、エクセルというのは、プログラミングの知識がない人が使う場合でも、処理の手順を記憶してくれて、その手順通り動いてくれるという「**マクロ**」という機能があるのです。そのマクロを動かすボタンが「**フォームコントロール**」にあるボタン類です。

君たちは、プログラミングの学習として BASIC という言語を学びながら操作をコードを書くことでコントロールしようとしていますので、その場合をこのエクセルは「**ActiveX コントロール**」と呼んでいるのです。



Private Sub CommandButton1_Click()

End Sub

Private Sub CommandButton1_Click()

πの計算

End Sub

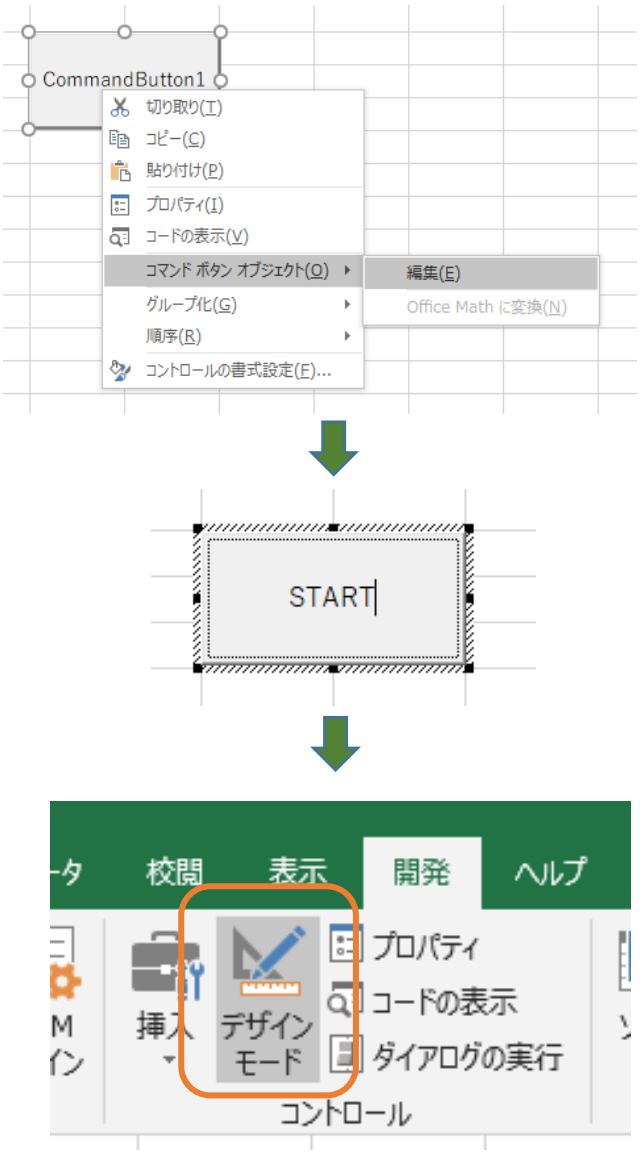
図 4-13

ボタン用コード

すると図 4-13 の上のようなコードのウインドウが出てきます。これはシート１用に書いているコードですね。でも、一番下を見ると「コマンドボタン１クリック」というプライベートサブルーチンがでています。中身は何も書かれてありません。

そこで「スタート」という名前のボタンを押したら「πの計算」というプログラムが走るようにします。そのために四角の中に「πの計算」と書いておきましょう。

これでこのボタンの役割が決定しました。



ボタンの名前

次にボタンの名前を「START」ということにしましょう。ボタンの名前を変えても、このボタンの機能は commandButton1 Click というプログラムですから安心してください。

CommandButton1 を右クリックして出てきたメニューから「**コマンドボタンオブジェクト**」を選びます。その中の「**編集**」を選ぶと、ボタンの周りが、ゲジゲジのようになります。そこで、「START」という名前に書き替えます。

しかし、こうしてボタンを押してもプログラムは走りません。リボンメニューの「**開発**」タブをクリックすると、「**デザインモード**」が濃くなっていて、アクティブになっています。つまり、この時はボタンのデザインや名前、機能をデザインしている状態にあったということです。

そこで「**デザインモード**」をクリックして、デザインは終わったよということを知らせます。

これで「START」ボタンをクリックしてください。走りますか？

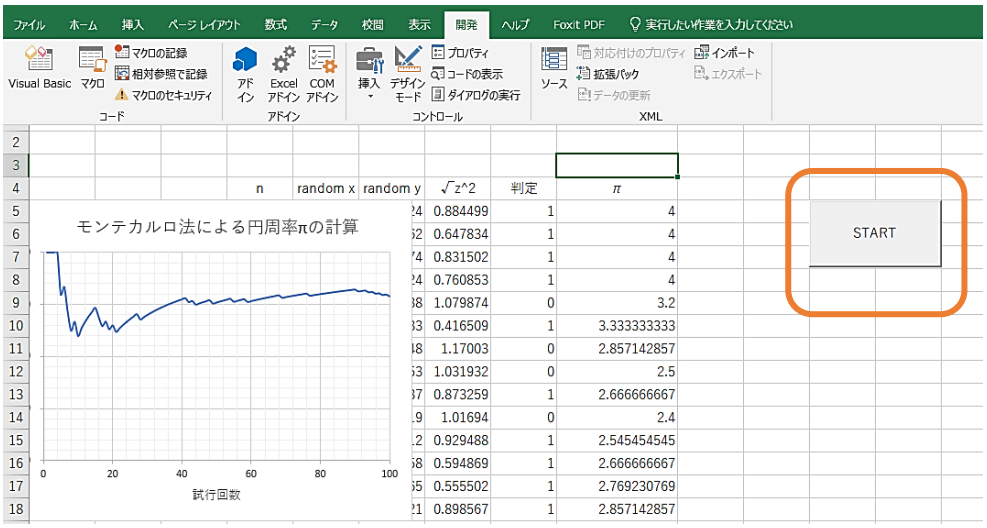
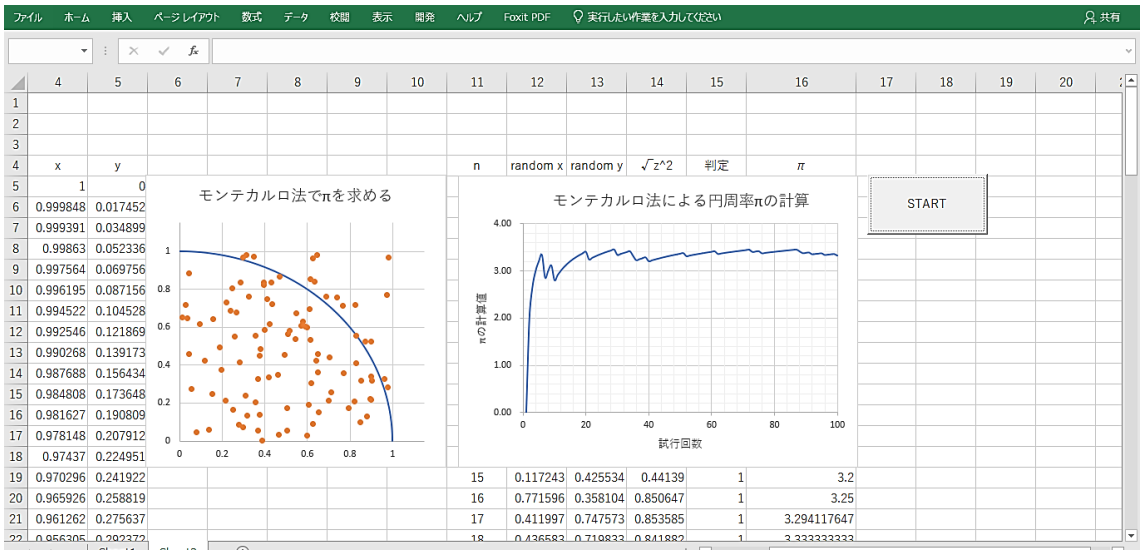
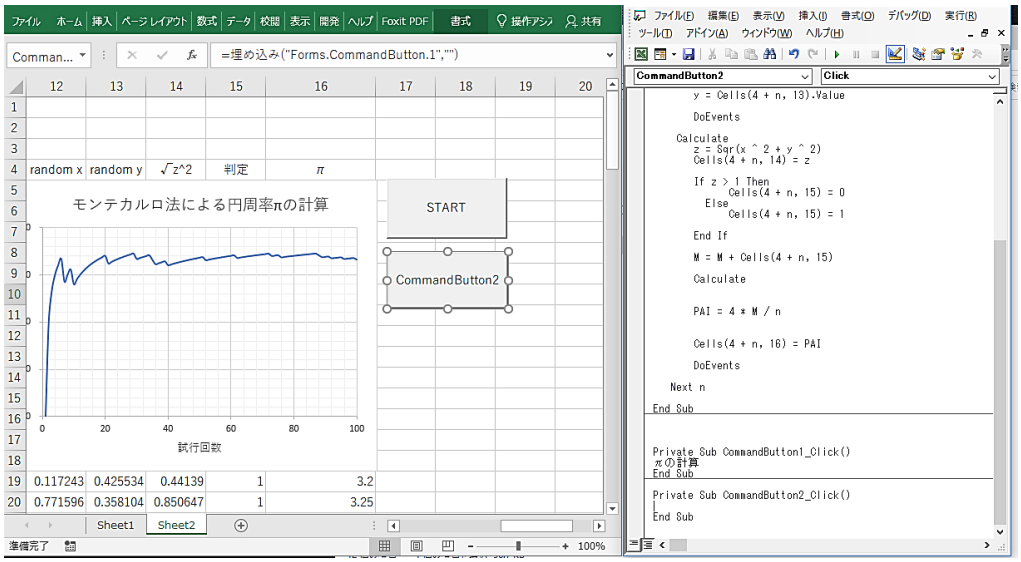


図 4-14





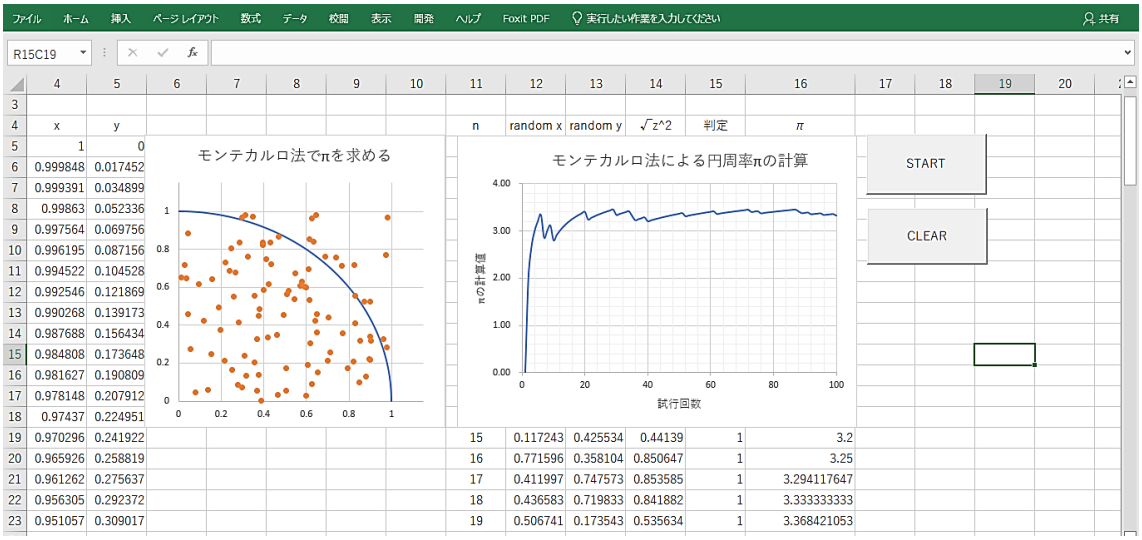

```
Private Sub CommandButton2_Click()
    Range(Cells(5, 12), Cells(104, 16)).Clear
End Sub
```

図 4-15

もう一つのボタン

この「START」ボタンだけだと前の結果が残ったまま、上書きするように実験が始まります。どうもしっくりきません。一回グラフをまっさらにするボタン(Clear ボタン)も作っておくことにします。

セル(5,12)からセル(104,16)までの**レンジ（範囲）**をまっさら（CLEAR）にするという命令が**図 4-15** に書かれています。



	x	y	n	random x	random y	√z^2	判定	π
5	1	0						
6	0.999848	0.017452						
7	0.999391	0.034899						
8	0.99863	0.052336						
9	0.997564	0.069756						
10	0.996195	0.087156						
11	0.994522	0.104528						
12	0.992546	0.121869						
13	0.990268	0.139173						
14	0.987688	0.156434						
15	0.984808	0.173648						
16	0.981627	0.190809						
17	0.978148	0.207912						
18	0.97437	0.224951						
19	0.970296	0.241922	15	0.117243	0.425534	0.44139	1	3.2
20	0.965926	0.258819	16	0.771596	0.358104	0.850647	1	3.25
21	0.961262	0.275637	17	0.411997	0.747573	0.853585	1	3.294117647
22	0.956305	0.292372	18	0.436583	0.719833	0.841882	1	3.333333333
23	0.951057	0.309017	19	0.506741	0.173543	0.535634	1	3.368421053

図 4-16

計算機実験

図 4-16 **が出来上がった計算機実験ラボです。**何回か実験してみましょう。円周率 3.1 に近づいていくことが分かると思います。しかし、もう気付いている人もいると思いますが、このプログラムは言い換えると「**100 で割って 4 倍すると円周率に近くなる整数を探せ**」という問題と同値ということになります。

それでは、1000 回にしたら？1500 回にしたら？面白い問題ですね。