# 数値計算

## 数値計算とは

　プログラミングの用途として「数値計算」という分野があります。コンピューターはもともと「計算機」と呼ばれ、計算をするための道具でした。数値計算は「方程式や微分方程式などを計算によって解く」というもので、独特の世界があります。

　あらゆる物理現象は、それを記述する偏微分方程式があります。その偏微分方程式を計算機で解く（シミュレートする）ことで、実験を行わなくても、実験を行ったのと同等の結果を得ることができます。新型コロナウイルスの感染防止の研究として、スパコン富岳による飛沫のシミュレーションの動画を目にした人も多いと思います。これは数値計算によって得られたものです。

　本章では数値計算の世界を少しだけ覗いてみましょう。

## 方程式の解を求める

### 二分法

　関数*f*(*x*)があり、*x* = 1と *x* = 3の間に解 f(x) = 0 が1個あることが分かっているとします。このとき、*f*(*x*) = 0 となる*x*を見つける方法として、2分法と呼ばれる方法があります。



図8.1　2分法のアルゴリズム

　図8.1は2分法のアルゴリズムです。*a*と*b*の間に解があることが分かっている場合、その中間の位置である*c* = (*a*+*b*) / 2の関数値*f*(*c*)を求めます。

(1) *f*(*a*)と*f*(*c*) が異符号‥‥‥解は*a*と*c*の間にあります。新しい探索区間を*a*－*c*に設定します。言い換えると*b* ← *c*として、*b*の値を更新します。

(2) *f*(*a*)と*f*(*c*) が同符号‥‥‥解は*c*と*b*の間にあります。新しい探索区間を*c*－*b*に設定します。言い換えると*a* ← *c*として、*a*の値を更新します。

*f*(*a*)と*f*(*c*) が同符号か異符号かは以下のように判定します。

　この方法は何ともコンピューターらしいテクニックですね。

　 が起こる可能性は限りなく低いですが、発生した場合について考えてみます。以下の3つの方法があり、どれを採用しても正しい答えが求まります。以下のプログラムでは (c) を採用しています。

(a) を解として計算を終了する

(b) 異符号の場合に含めて計算を続行する

(c) 同符号の場合に含めて計算を続行する

　2分法の計算を1回実行すると、探索区間は に狭まります。*n*回繰り返すと区間の幅は になります。分母がねずみ算式に大きくなるので、高速に収束します。210 = 1024なので、10回の繰り返し計算で探索区間は千分の1程度になります。20回繰り返すと、分母がさらに1000倍されるので、探索区間は百万分の1程度になります。

　「いつ計算を終了するのか？」については、収束判定基準をとするとき、 となったら終了し、解を とします。 の大きさは必要な精度に応じて決定します。

【例題1】

*f*(*x*) = *x*2 + *x*  3の解は *x* = 0 と *x* = 2 の間に1個あることが分かっています。解を求めなさい。ただし、収束判定基準はに設定しなさい。

Sub nibun()

a = 0

b = 2

Debug.Print "--- start ---"

For i = 1 To 100

fa = f1(a)

fb = f1(b)

c = (a + b) / 2

fc = f1(c)

Debug.Print i & " " & a & " : " & fa & " " & b & " : " & fb

If fa \* fc < 0 Then

b = c

Else

a = c

End If

If b - a < 10 ^ -6 Then

Exit For

End If

Next i

c = (a + b) / 2

fc = f1(c)

Debug.Print "ans"

Debug.Print c & " : " & fc

End Sub

Function f1(x)

f1 = x ^ 2 + x - 3

End Function

　関数の部分はFunction文を使用しました。

　「収束するまで繰り返す」という計算をするので、ループは本来ならDo While Trueを用いるべきです。しかし、プログラムのミスにより無限ループに陥ると、Excelがハングアップする可能性が高いので、Forループを用いて、最大100回で計算を終了するようにしています。1回の計算ごとにDebug.Printに途中経過を書き出します。

【課題1】

　 の解は *x* = 1 と *x* = 4 の間に1個あります。2分法を用いて求めなさい。ただし、*x*の精度は10-6以下に設定しなさい。

　2分法とよく似た方法として、はさみうち法と呼ばれる方法があります。方法を図8.2(a) に示します。2分法は区間の中間点を次の点として選びましたが、はさみうち法では、(*a*, *f*(*a*))と(*b*, *f*(*b*)) を結ぶ直線を引き、直線と*x*軸の交点を次の点として選びます。図8.2(a) の場合は、2分法より収束が速いように見えます。2分法より優秀な方法に見えますが、図8.2(b) のケースのように関数が鍋底のような形をしている場合は、収束が遅くなります。2分法はどんな場合でも1回の計算で探索範囲が半分になるので、はさみうち法よりロバスト（頑強）な方法です。筆者はかつて光導波路の研究をしていましたが、光導波路のモードを求める計算において、2分法を用いていました。

　　　　

　　　　　　　(a) 考え方　　　　　　　　　　　　　(b) うまくいかない例

図8.2　はさみうち法

### ニュートン法

　先に学んだ2分法は優れた方法ですが、収束するまでに何回も関数の値を求める必要があります。今回の関数f1(x)のように、関数値が一瞬で求められる場合はいいのですが、現実の問題を解く場合、関数値を1回求めるために、何十分もかかる計算が必要とされることは珍しくありません。

*x*の初期値と解の区間において関数が単調減少（増加）であることが分かっている場合、2分法よりも遙かに少ない反復回数で解を求める方法があります。それがニュートン法です。ニュートン法は2分法に比べると、収束が桁違いに速いです。



図8.3　ニュートン法

　ニュートン法のアルゴリズムを図8.3に示します。*x*0が出発点です。

1. 現在のxにおける接線を求める

2. 接線とx軸の交点を次のxの点とする

3. 前回のxと現在のxの差がε（収束判定基準）より小さいとき終了。そうでないとき1. へ戻る。

　上記の手順の繰り返しにより、x0, x1, x2,…………は解に収束します。

　ニュートン法を一言で言うと「現在のx付近を1次式で近似したときの解の位置へジャンプする」ということを繰り返すことで、xの値を解へ収束させる方法です。

　ニュートン法を使うには現在のxにおける関数値*f*(*x*)に加えて、傾き*f'*(*x*)を知る必要があります。*f'*(*x*) が数式として得られる場合は数式を用います。数式として得られない場合は、Δ*x*として小さな値を設定し、

|  |  |
| --- | --- |
|  |  |

を計算します。

　xの値がx1のときの関数値をy1、傾きを*s*1とします。xの次の値*x*2を求める式を導出します。傾きが*s*1のとき、接線は次式で表されます。

|  |  |
| --- | --- |
|  |  |

*y =* 0となる*x*を求めます。*y*に0を代入し、*x*について解くと、

が得られます。*x*の更新式が得られました。

【例題2】

　を求める方法として、 の解を求めるという方法があります。この様子を図8.4に示します。



図8.4　平方根の求め方

　です。5の平方根をニュートン法を用いて求めなさい。*x*の初期値は5に設定し、収束判定基準は、更新前の*x*と更新後の*x*の差が10-6以下になれば収束したと判定します。

Function f2(x)

f2 = x ^ 2 - 5

End Function

Function f2d(x)

f2d = 2 \* x

End Function

Sub newton()

x = 5 ' 初期値

For i = 1 To 100

y = f2(x) ' 関数値と傾きを求める

s = f2d(x)

Debug.Print i & " " & x & " " & y

x2 = x - y / s ' x の更新式

If Abs(x2 - x) < 10 ^ -6 Then

Exit For

End If

x = x2 ' x の値を更新する

Next i

Debug.Print "ans " & x2

End Sub

　xとx2はどちらが大きいか分からないので、収束したかどうかを判断するために、Abs()という絶対値を取る関数を用いています。

　高校の数学の教科書に*a*の平方根を求める方法として、*x*に適当な正の初期値を入れて、

を反復するという方法が載っています。これは、実はニュートン法です。変形すると

となります。一方で、ニュートン法の反復式

の右辺に

を代入すると、右辺は

となり、数学の教科書の式と一致します。

【課題2】

*a*の3乗根は の解を求めることで得られます。この方法を利用して、10の3乗根を求めなさい。*x*の初期値は10とします。ただし、関数の微分値は を使わずに、

を用いて求めなさい。Δ*x*の値は に設定しなさい。

　ニュートン法は非常に収束が速く、2分法より遥かに少ない反復回数で方程式を解くことが出来ます。しかし、その反面、

　　探索点と解の区間において、関数値が単調減少（増加）でなければならない

という制約があります。例えば図8.5の場合、*x*0, *x*1, *x*2, ……が発散するので、解が求まりません。



図.5　ニュートン法で解が発散する例

## 微分方程式と偏微分方程式

### 1階の微分方程式

　世の中のほとんどの物理現象は微分方程式で表されます。微分方程式とは未知関数を求めるための式です。例えば *y*(*x*) が *x* の関数であるとき、

|  |  |
| --- | --- |
|  | (1) |

は微分方程式です。この方程式を満たす未知関数 *y*(*x*) を見つけることを「微分方程式を解く」と言います。この場合、解析解は

|  |  |
| --- | --- |
|  |  |

となります。*C*は任意定数で、初期条件（*x* = 0のときの*y*の値）より決まります。

　(1) の場合は、微分方程式を解析的に解くことができ、その解は解析関数で表すことができました。しかし、多くの場合、微分方程式を解析的に解くことはできません。すなわち解は解析関数にはなりません。

　解析的に解けない微分方程式に対するアプローチとして「数値的に解く」という方法があります。微分方程式をよく見ると、左辺 は関数 *y*(*x*) の傾きを表しています。*x* = 0のときの*y*の値が分かっていれば、そのときの傾きを用いて微小距離Δ*x*進むことで、 における関数値*y*が分かります。この式の場合、傾きが *ay* なので、*x*がΔ*x*増えたときの*y*の増分値は、Δ*x*・(*ay*) です。これを繰り返すことにより、関数*y*(*x*)の形を求めることができます。この様子を図8.6に示します。この方法は、微分方程式を解く最も簡単な方法であり、オイラー法と呼ばれます。



図8.6　オイラー法

【例題3】

　(1) において*a* = 1とし、*x* = 0のとき*y* = 10とする。*x* = 0から*x* = 3の範囲において、関数 *y*(*x*) を求めなさい。ただし の刻み幅で計算しなさい。

Sub bibun()

a = 1

x = 0

y = 10

dx = 0.1

Cells(1, 1) = x

Cells(1, 2) = y

Cells(1, 3) = 10 \* Exp(-a \* x)

For i = 2 To 1000

dy = -a \* y \* dx

x = x + dx

y = y + dy

Cells(i, 1) = x

Cells(i, 2) = y

Cells(i, 3) = 10 \* Exp(-a \* x)

If x > 3 Then

Exit For

End If

Next i

End Sub

　この微分方程式の解析解はです。比較のため、C列に解析解を書き込んでいます。指数関数 はExp(x) と書きます。

　B列の数値解とC列の解析解を比較するために、グラフを書きなさい。解析解と数値解の差を観察するために、グラフは縦軸は長くしなさい。

　この例では に設定しました。 にすると、解析解と数値解の差が大きくなります。 とすると、小さくなります。確認しなさい。

【課題3】

(a) 以下の微分方程式を*x* = 0 から *x* = 2 の範囲で解くプログラムを作成しなさい。ただし*x* = 0 のとき *y* = 1 です。刻み幅はΔ*x* = 0.05 に設定しなさい。*x*の値をA列、*y*の値をB列に入れなさい。

(b) その結果をグラフで描きなさい。

(c) 解析解を求め、C列に書き込み、解析解もグラフに加えなさい。

### 2階の微分方程式

　上の例題では、独立変数は*x*、従属変数は*y*でした。物理現象を表す多くの微分方程式では、独立変数は時刻*t*となります。これ以降は、物理量は時刻*t*の関数です。



図8.7　バネとおもり

　図8.7を見て下さい。おもりがバネにつながれています。おもりの位置*x*が のとき、バネの力は0になるとします。おもりに働く力を考えます。右方向への力を正とします。バネの力は が正のとき左方向、 が負のとき右方向なので、

|  |  |
| --- | --- |
|  |  |

です。方向を考慮してマイナスがついています。 はバネの強さを表す係数です。おもりの位置 は時刻 の関数です。位置を微分すると速度を表し、速度を微分すると加速度を表します。すなわち位置を2回微分すると加速度になります。加速度は

|  |  |
| --- | --- |
|  |  |

で表されます。ですから、力は

|  |  |
| --- | --- |
|  |  |

です。この力がバネの力と等しいので、

|  |  |
| --- | --- |
|  |  |

という微分方程式が得られます。*m*を右辺へ移項して

|  |  |
| --- | --- |
|  |  |

となります。

　微分方程式を数値的に解くには、左辺を1階微分を表す式（ex. ) にする必要がありますが、上式は2階微分の式になっています。左辺を1階微分を表す式にするために、新たな変数を1個導入します。 の1階微分、すなわち速度を表す変数*v*を導入します。以下の連立微分方程式が得られます。

|  |  |
| --- | --- |
|  | (2) |
|  | (3) |

　上の2つの式の左辺はどちらも1階微分を表す式となっています。

　初期条件は のとき、バネを引っ張って であったと仮定しましょう。 とします。離した瞬間の速度は0ですから、 です。これを解きます。

【例題4】

　(2)(3) の連立の微分方程式を解くプログラムを作成しなさい。A列に時刻 , B列に の値, C列に の値を入れなさい。 の刻み幅は0.05に設定し、 まで解きなさい。グラフを書きなさい。

Sub bane()

x = 1

v = 0

t = 0

dt = 0.05

For i = 1 To 1000

Cells(i, 1) = t

Cells(i, 2) = x

Cells(i, 3) = v

dx = dt \* v

dv = dt \* -0.5 \* x

x = x + dx

v = v + dv

t = t + 0.05

If t > 13 Then

Exit For

End If

Next i

End Sub

【課題4】

(a) 解析解は , です。D列に の解析解、E列に の解析解を入れなさい。グラフを書き、計算で求めた解と比較しなさい。

(b) 縦軸を （B列）, 横軸を （C列）の値にとり、計算して求めた値（BC列）のグラフを描きなさい。

### 偏微分方程式

　偏微分というと難しい響きがしますが、英語ではpartial differentialと書きます。直訳すると「部分微分」であり、難しいものではありません。「偏微分」という言葉に飲まれてしまう人がいますが、恐れないようにしましょう。



図8.8　金属棒

　図8.8のような金属棒を考えます。棒上の各場所による温度を考えます。温度を という文字で表すと、 は時刻 と場所 の関数です。

|  |  |
| --- | --- |
|  |  |

と書けます。 は と の2つの変数の関数なので、「 を微分する」というと、どちらの変数で微分するのか分からなくなってしまいます。そこで「 について微分する」ときは「 で偏微分する」、「 について微分する」ときは「 で偏微分する」と言います。

　ある時刻 における の温度分布が与えられたとします。各場所の温度変化は以下の偏微分方程式で与えられます。

|  |  |
| --- | --- |
|  | (1) |

　左辺は温度*u*の時刻*t*に対する傾きを表します。「傾き」は直線において定義される量であり、*t*が1増加したときの*u*の増加量です。*t*の単位は秒です。変化が緩やかなとき、「傾き」＝「1秒後の*u*の増加量」です。ゆえに、左辺は1秒後にその場所の温度が何度上がっている（下がっている）かを表します。右辺は温度*u*の*x*に関する2階微分に係数*a*をかけたものなので、この方程式は以下のことを表しています[[1]](#footnote-1)。

ある場所の温度の時間変化は、その場所の温度の空間方向の2階微分に比例する

　この方程式が導出される理由を考察します。



図8.9　熱伝導方程式の導出

　熱は温度が高い場所から低い場所へ移動し、移動量は温度差に比例します。 の場所をそれぞれ「エリアA」「エリアB」「エリアC」とし、温度は20度、30度、50度とします。1度につき熱の粒が1個あると仮定します。隣り合うエリア間で移動する熱の量は、温度差（個数差）の1/10と仮定します。

　「エリアB」から「エリアA」へ1秒間に移動する熱の個数は差（10個）の1/10なので、1個です。「エリアC」から「エリアB」へは、差（20個）の1/10なので、2個です。1秒後、「エリアB」は 個の熱の粒があるので、31度となります。この例では*x*の間隔は1でした。*x*の間隔をΔ*x*とすると、各エリアの熱の個数をA, B, Cとするとき、以下のようになります。

　すなわち、全ての場所で、温度の時間微分は、空間の2階微分に比例します。ちょっと強引な説明でしたが、偏微分方程式が表す意味を説明してみました。

　この微分方程式をExcel VBAで計算するときの数式は以下のようになります。

　偏微分方程式を解くには、境界条件といって領域の端における条件が必要です。境界条件は次の2種類があります。どちらかを与えます。

　　(a) 境界における値を与える

　　(b) 境界における 方向の微分値を与える

　ここでは (a) を使います。

　具体的な問題に入りましょう。金属棒の長さを20とし、 ～ までの区間で考えます。サンプル点を 座標上に間隔1でとります。21点における温度を考えます。

　左端の境界は100度、右端の境界は0度とします。初期条件として棒の温度は左端以外全て0度とします。左端から棒を熱することを意味します。 を離散化します。離散点を で表します。 は時刻方向の点の番号、 は 方向の点の番号です。(2)を離散化すると以下の式が得られます。時間の刻み幅を 、 方向の刻み幅を とします。

|  |  |
| --- | --- |
|  | (4) |

　 の形に書くと以下のようになります。

|  |  |
| --- | --- |
|  | (5) |

【例題5】

　 とし、 から までの温度分布を求めるプログラムを作成しなさい。

　以下のようになります。ただし、刻み幅は とし、3秒ごとに値を出力します。現在の温度分布はx(1)～x(21), 次の時刻の温度分布はx2(1)～x2(21)です。

Sub netudendou()

Dim x(30), x2(30)

' セルのクリア

Cells.ClearContents

' 初期値の代入

t = 0

dt = 0.1

dx = 1

a = 1

Row = 0

For i = 1 To 21

x(i) = 0

Next i

x(1) = 100

For Count = 0 To 1000

If Count Mod 30 = 0 Then

Row = Row + 1

For i = 1 To 21

Cells(Row, i) = x(i)

Next i

End If

x2(1) = 100 ' 境界条件

x2(21) = 0

For i = 2 To 20

x2(i) = x(i) + a \* dt / (dx \* dx) \* \_

(x(i - 1) - 2 \* x(i) + x(i + 1))

Next i

For i = 1 To 21

x(i) = x2(i)

Next i

t = t + dt

If t > 90 Then

Exit For

End If

Next Count

End Sub

【課題5】

　「3D等高線」のグラフを描きなさい。できるだけ見やすいように、視点を設定しなさい。

・グラフを回転させる方法

　「グラフツール：書式」→「現在の選択範囲をグラフエリア（あるいはプロットエリア）に設定し、選択対象の書式設定をクリック」→「五角形をクリック」→「3D回転でX方向に回転の角度を変更」　-30度に設定したいときは330度に設定する。

・縦軸、横軸を入れ替える方法

　「グラフツール：デザイン」→「行／列の切り替え」

　最終的な熱平衡状態において、温度分布はどのようになるか？

（補足）

　今回、示した方法は熱伝導方程式を解く最も簡単な方法です。陽解法と呼ばれる方法の一種で、式は簡単なのですが、条件によっては不安定現象が起こるというデメリットがあります。時間の刻み幅をΔt, 空間の刻み幅をΔxとするとき、以下の条件を満たす必要があります。

　この条件を満たすためには、Δ*t*をかなり小さな値にとる必要があります。本書では触れませんが、陰解法を用いると、「1ステップ進むたびに未知数の個数が*x*方向の離散点数（今回は21個）である連立1次方程式を解く必要がある」のですが、Δ*t*を大きくとれるため、結果的に計算速度は速くなる場合が多いです。

【課題6】

　初期条件（ のときの温度分布）を以下のように設定して計算した場合のグラフを描きなさい。

For i = 1 To 21

x(i) = 0

Next i

x(10) = 100

計算条件は以下の通りです。

　・境界条件は両端は常に0度です。

　・時間の刻み幅は に設定します。0.5秒ごとに値をセルに書き出し、 まで計算しなさい。

1. 係数*a*は数式を簡潔にするための本書独自の記号である。一般の教科書では などの記号が用いられる。 [↑](#footnote-ref-1)