数値計算法

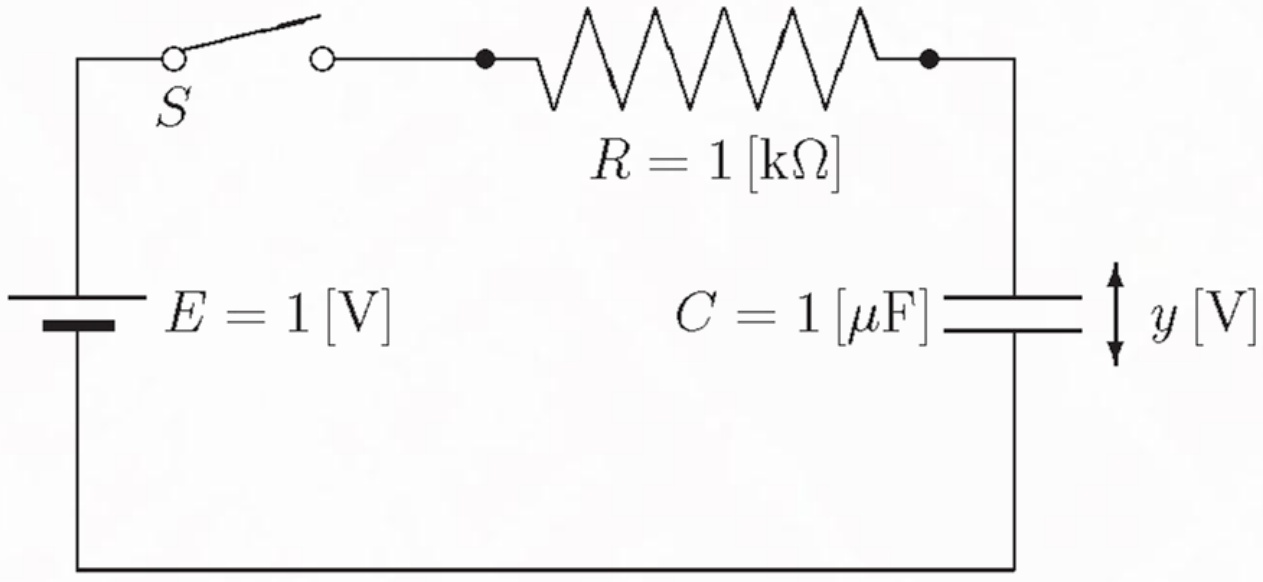
第七回
微分方程式の解法(続き)
― Runge-Kutta法 ―

若狭　智嗣

粒子物理学講座

微分方程式の数値解法

- **例題**
  - 以下のRC回路を考える。始めコンデンサCに電荷は蓄えられていないとする。時刻 $x=0$ でスイッチSを入れる時、時刻 x でのコンデンサCの両端の電圧 y を求めよ。



微分方程式

- 回路を流れる電流 i は、コンデンサに蓄えられた電荷を Q として、

$$i = \frac{dQ}{dt} = \frac{d(CV)}{dt} = C\frac{dV}{dt}$$

- キルヒホッフの法則から

$$E = V + iR = V + CR\frac{dV}{dt}$$

- $t=x,\ V=y,\ C=1\mu F,\ R=1k\Omega,\ E=1V$より

$$\frac{dy}{dx} = 1 - y \qquad \text{初期条件}: y(0) = 0$$

  - 時定数$\tau=RC$=1 ms であるので、時刻 x は ms で表す（電圧 y はV）

復習ー数値計算における微分の考え方ー

- x の関数 y を求める（任意の x の y の値を求める）ことが目標
- 微分の定義

$$\left.\frac{dy}{dx}\right|_{x=x_0} = \lim_{h\to 0}\frac{y(x_n+h)-y(x_n)}{h} \equiv f(x_n, y(x_n))$$

有限の微小量hで離散化

$$= \frac{y(x_n+h)-y(x_n)}{h}$$

- 条件（ $x_1 = x_0 + h$ での y を求める時を考えると）
  - 初期条件から、x_0 での y は分かっている
  - 微分方程式から、x_0 での dy/dx は分かっている
  - 離散化の幅 h は自分で決める（必要な精度に応じて決める）

以上の条件（情報）から、$y(x_1 = x_0 + h)$ を求める

Euler法（Taylor展開の1次までの近似）

- 微分の定義

$$\left.\frac{dy}{dx}\right|_{x=x_0} = \lim_{h\to 0}\frac{y(x_n+h)-y(x_n)}{h} \equiv f(x_n, y(x_n)) = \frac{y(x_n+h)-y(x_n)}{h}$$

- Euler法（一番簡単な微分方程式の数値解法）
  - y の x_n の周りでのTaylor展開

Euler法の誤差の程度

$$y(x_n+h) = y(x_n) + hy'(x_n) + \frac{h^2}{2!}y''(x_n) + \frac{h^3}{3!}y'''(x_n) + \cdots$$

無視

$$y(x_n+h) = y(x_n) + hf(x_n, y(x_n))$$

$$\begin{aligned} y(x_1) &= y(x_0) + hf(x_0, y(x_0)) \\ y(x_2) &= y(x_1) + hf(x_1, y(x_1)) \\ &\vdots \\ y(x_m) &= y(x_{m-1}) + hf(x_{m-1}, y(x_{m-1})) \end{aligned}$$

  - 前の点(x_{m-1}, y_{m-1})の傾き（微分）をたどって、次の点(x_m, y_m)を求める

先週の演習

- サンプルプログラムは
  /home/teacher/z6wt01in/SAMPLE/
  に
  rc_circuit.f diff_eq.f euler.f
  として置いてある

- 演習0：各人コピーし、実行ファイル rc_curcuitを
  % frt -o rc_curcuit rc_circuit.f diff_eq.f euler.f
  により生成し実行せよ
  （コンパイル時に”jwd2008i-i “diff_eq.f”, line 1: この仮引数‘x’は,副プログラム中で使用されていません.”というメッセージが出るが無視して構わない）
- 演習1：厳密解 $y(x)=1-e^{-x}$と比較せよ
- 演習2：刻み幅 STEP を変えて実行し、厳密解との差がどうなるか検討せよ
  （但し、0.0001より小さくしないこと！！！）
- 演習3：計算結果をファイルに出力せよ（リダイレクションを用いよ）
- 演習4：計算結果を gnuplot を用いてグラフにせよ
- 演習5：計算結果と厳密解を gnuplot を用いてグラフにせよ

厳密解との比較の仕方（例）

```
c 時間xでの電圧をyとし、時間がSTEPだけ経過した時の
c 電圧y_nextを微分方程式を解いて求める
      call calc_next_step(x,y,STEP,y_next)
      x     = x + STEP                          ! STEP後の時間
      y     = y_next                            ! STEP後の電圧
      write(*,'(a,F5.2,a,F8.5)')                ! STEP後の時間と
     &   'Time: ',x,' Voltage: ',y               ! 電圧を出力
```

```
c 時間xでの電圧をyとし、時間がSTEPだけ経過した時の
c 電圧y_nextを微分方程式を解いて求める
      call calc_next_step(x,y,STEP,y_next)
      x     = x + STEP                          ! STEP後の時間
      y     = y_next                            ! STEP後の電圧
      write(*,'(a,F5.2,2(a,F8.5))')             ! STEP後の時間と
     &   'Time: ',x,' Voltage: ',y,              ! 電圧、厳密解との差を出力
     &   ' Diff: ',y-(1.0-exp(-x))
```

出力例

```
Time:  0.25 Voltage:  0.25000 Diff:  0.02880
Time:  0.50 Voltage:  0.43750 Diff:  0.04403
Time:  0.75 Voltage:  0.57812 Diff:  0.05049
Time:  1.00 Voltage:  0.68359 Diff:  0.05147
```

結果のSTEP依存性

- **単純な予想**
  - Euler法はTaylor展開の2次（$\propto h^2$）以降を無視
  - Euler法の誤差はh^2に比例するので、hを1/10にすれば、厳密解との差（誤差）は1/100になる
- **計算誤差を考慮した予想**
  - 計算数はh^{-1}に比例し、数が増えると桁落ちなどの計算誤差が累積する。従って厳密解との差はh^2ではなく、hに比例する
- **結果**
  - STEP=0.1　Time: 4.00 Voltage: 0.98522 Diff: 0.00353
  - STEP=0.01　Time: 4.00 Voltage: 0.98205 Diff: 0.00037
  - STEP=0.001　Time: 4.00 Voltage: 0.98165 Diff: 0.00004
  - STEP=0.0001　Time: 4.00 Voltage: 0.98163 Diff: 0.00004

1/10

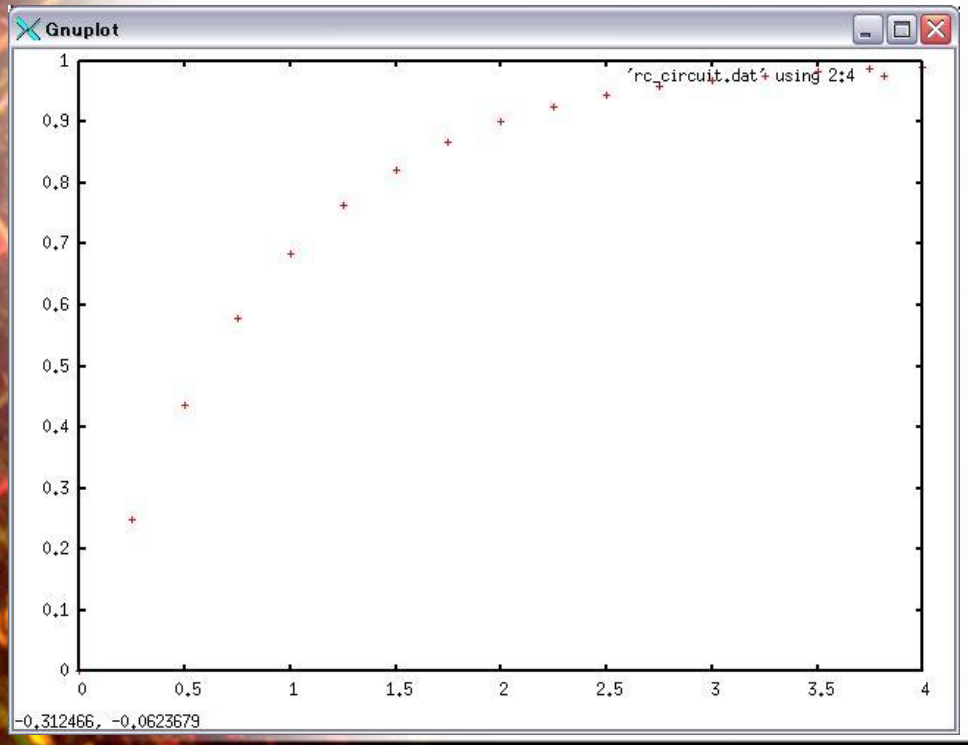
1/10

ここまで

桁落ちの影響が支配的になり精度が向上しない

リダイレクションと結果の視覚化（グラフ化）

- **結果をファイルに出力する（リダイレクション）**
  - % rc_circuit > rc_circuit.dat
- **Gnuplotによるグラフ描画**
  - gnuplot> plot ‘rc_circuit.dat’ using 2:4



2列目をx

4列目をy

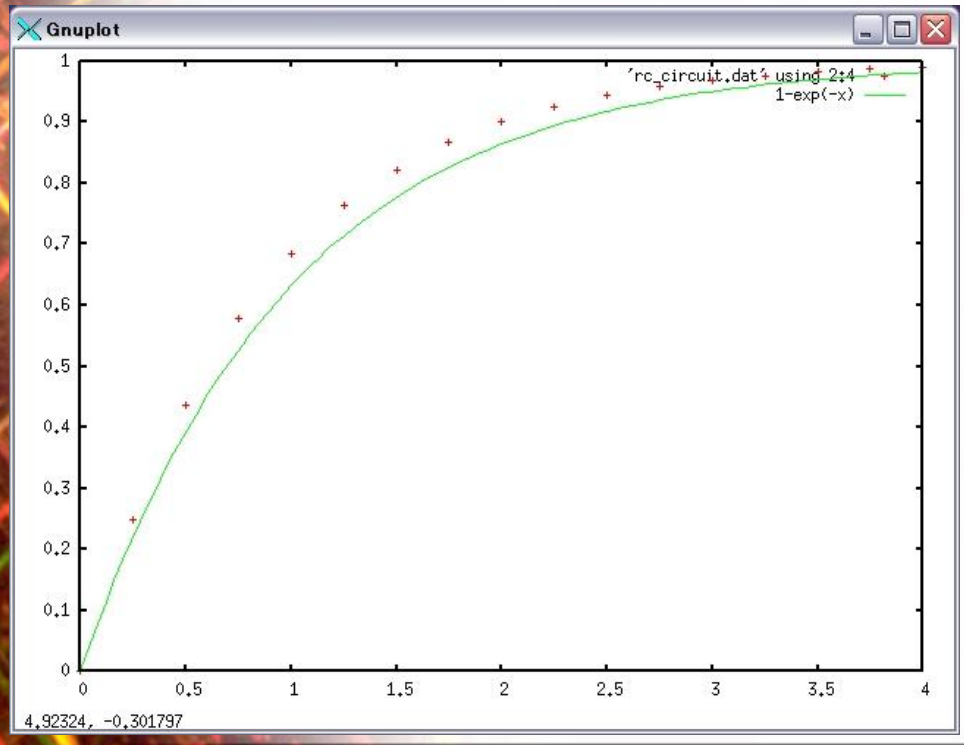
mule@ah

File Edit Options Buffers Tools Help

Time:	0.00	Voltage:	0.00000
Time:	0.25	Voltage:	0.25000
Time:	0.50	Voltage:	0.43750
Time:	0.75	Voltage:	0.57812
Time:	1.00	Voltage:	0.68359
Time:	1.25	Voltage:	0.76270
Time:	1.50	Voltage:	0.82202

Euler法の結果の視覚化と厳密解との比較

- 厳密解：1-exp(-x)
- 結果をファイルに出力する（リダイレクション）
  - % rc_circuit > rc_circuit.dat
- Gnuplotによるグラフ描画
  - gnuplot> plot ‘rc_circuit.dat’ using 2:4,1-exp(-x)



2列目をx

4列目をy

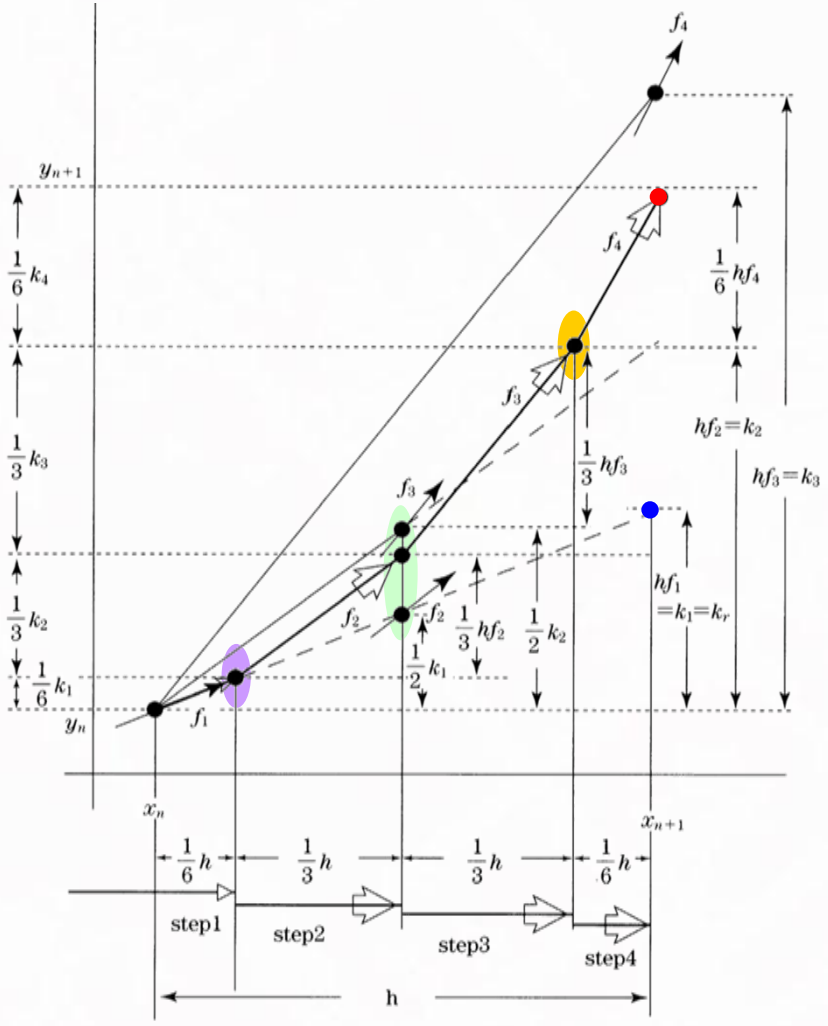
mule@ah

File Edit Options Buffers Tools Help

Time:	0.00	Voltage:	0.00000
Time:	0.25	Voltage:	0.25000
Time:	0.50	Voltage:	0.43750
Time:	0.75	Voltage:	0.57812
Time:	1.00	Voltage:	0.68359
Time:	1.25	Voltage:	0.76270
Time:	1.50	Voltage:	0.82202

Euler法とRunge-Kutta法（高精度の微分方程式の解法）

- **Euler法**
  - x_{n+1}でのyの値(y_{n+1})を計算するのに以下のものを用いる
    - x_nでのyの値(y_n)
    - x_nでのyの微係数
  - 誤差はh(=$x_{n+1}-x_n$)の2乗程度（繰り返しにより1乗）
- **Runge-Kutta法**
  - Euler法に加えて、以下の点での情報を用いる
    - x_nから　h/6 進んだ点
    - x_nから 3h/6 進んだ点
    - x_nから 5h/6 進んだ点
  - 誤差はh(=$x_{n+1}-x_n$)の5乗程度



Runge-Kutta法のサンプルプログラムの説明

(/home/teacher/z6wt01in/SAMPLE/runge_kutta.f)

- k1 : (x　　,y　)での勾配でstepだけ進んだ時のyの増分
- k2 : (x+step/2,y+k1)での勾配でstepだけ進んだ時のyの増分
- k3 : (x+step/2,y+k2)での勾配でstepだけ進んだ時のyの増分
- k4 : (x+step　,y+k3)での勾配でstepだけ進んだ時のyの増分

```
c begin:
      k1 = step * diff_eq(x              , y          )
      k2 = step * diff_eq(x + 0.5*step, y + 0.5*k1)
      k3 = step * diff_eq(x + 0.5*step, y + 0.5*k2)
      k4 = step * diff_eq(x +     step, y +     k3)
      y_next = y + (k1 +2.0*k2 +2.0*k3 +k4)/6.0
      return
      end
```

$$y_{n+1} = y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4$$

Runge-Kutta法の中身を完全に理解しないといけないか？

- **Euler法は微分の定義そのものである（Tayler展開1次の項まで）**
  - 本来0にするべき h を微小量として数値計算で扱う（理解が容易）
- **Runge-Kutta法はTayler展開の4次の項まで消えている**
  - 式は若干複雑
- **基本的には、サンプルで与えたサブルーチンを道具として使えればよい。**
- **留意すべき点** **（正しい道具を正しく使う）**
  - 道具が正しいか（バグが無いか）？
  - 道具を正しく使っているか？
- **正しい道具を正しく使っているか判断するには？**
  - 解析的に解ける問題を扱う（厳密解との比較）
  - hに対する依存性
    - Euler法ならhを1/10にすれば精度（誤差）は1/10
    - Runge-Kutta法ならhを1/10にすれば精度（誤差）は1/10000

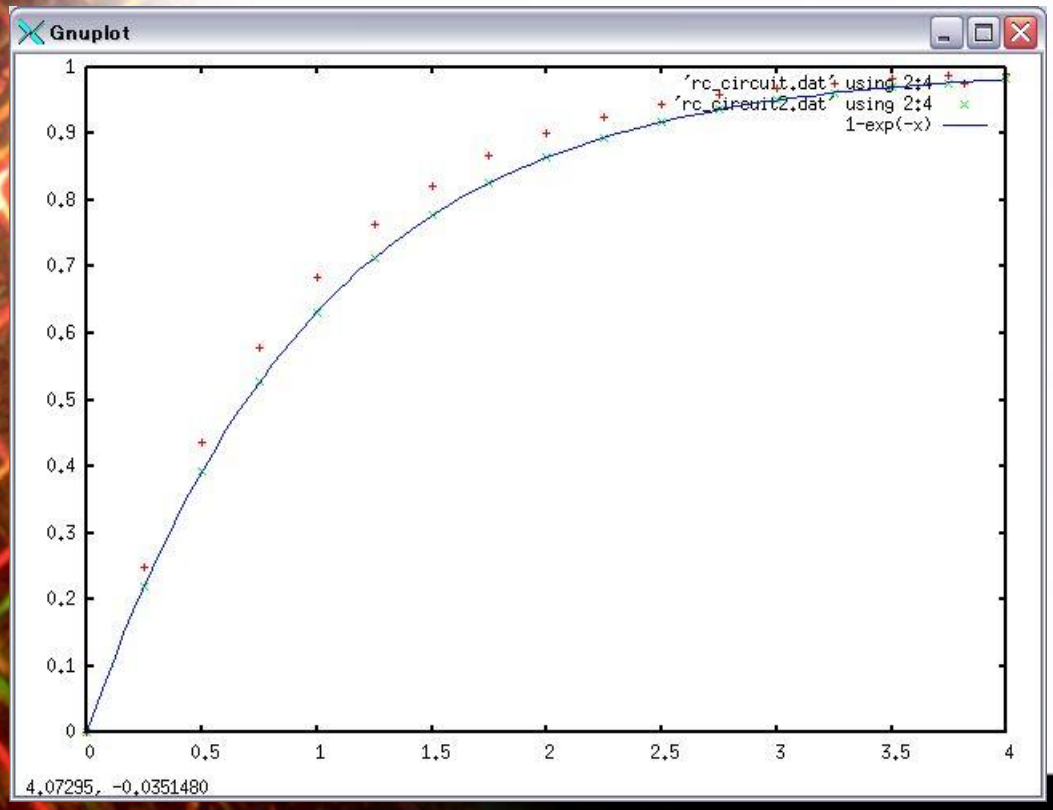
演習

- サンプルプログラムは
  /home/teacher/z6wt01in/SAMPLE/
  に
  rc_circuit.f diff_eq.f runge_kutta.f
  として置いてある
- 演習0：各人コピーし、実行ファイル rc_curcuitを
  % frt -o rc_curcuit2 rc_circuit.f diff_eq.f runge_kutta.f
  により生成し実行せよ
  （コンパイル時に”jwd2008i-i “diff_eq.f”, line 1: この仮引数‘x’は,副プログラム中で使用されていません.”というメッセージが出るが無視して構わない）
- **演習１：厳密解 $y(x)=1-e^{-x}$と比較せよ**
  - gnuplotを用いて視覚化（グラフ化）すること
- **演習２：Euler法の結果と比較せよ**
  - gnuplotを用いて視覚化（グラフ化）すること
  - 厳密解とも比較すること
- **演習４：刻み幅 STEP を 1.0 と 0.1 に変えて実行し、厳密解との差を検討せよ**
  - Runge-Kutta法が正しく機能しているか検討せよ

結果の視覚化（グラフ化）

- 結果をファイルに出力する（リダイレクション）
  - % rc_circuit2 > rc_circuit2.dat
- Gnuplotによるグラフ描画(Euler法の結果＋Runge-Kutta法の解+厳密解)
  - gnuplot> plot ‘rc_circuit.dat’ using 2:4, ’rc_circut2.dat’ using 2:4,1-exp(-x)

プロンプト



2列目をx

4列目をy

mule@ah

File Edit Options Buffers Tools Help

Time :	0.00	Voltage :	0.00000
Time :	0.25	Voltage :	0.22119
Time :	0.50	Voltage :	0.39346
Time :	0.75	Voltage :	0.52762
Time :	1.00	Voltage :	0.63211
Time :	1.25	Voltage :	0.71348
Time :	1.50	Voltage :	0.77686

結果のSTEP依存性

- **単純な予想**
  - Runge-Kutta法の誤差はh^5に比例するので、hを1/10にすれば、厳密解との差（誤差）は1/100000になる
- **計算誤差を考慮した予想**
  - 計算数はh^{-1}に比例し、数が増えると桁落ちなどの計算誤差が累積する。従って厳密解との差はh^5ではなく、h^4(=1/10000)に比例する
- **結果**
  - STEP=1.0　Time: 4.00 Voltage: 0.9802246 Diff: −0.0014598
  - STEP=0.1　Time: 4.00 Voltage: 0.9816843 Diff: −0.0000001
  - STEP=0.01　Time: 4.00 Voltage: 0.9816842 Diff: −0.0000001

1/10000

ここまで

桁落ちの影響が支配的になり精度が向上しない

演習

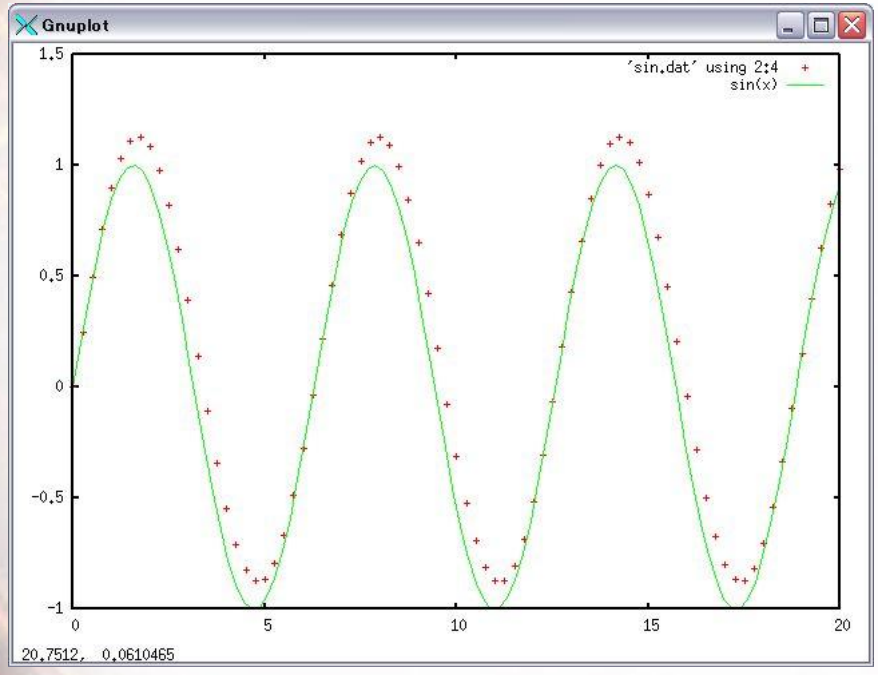
- 以下の微分方程式

$$\frac{dy}{dx} = \cos(x), \quad y(0) = 0$$

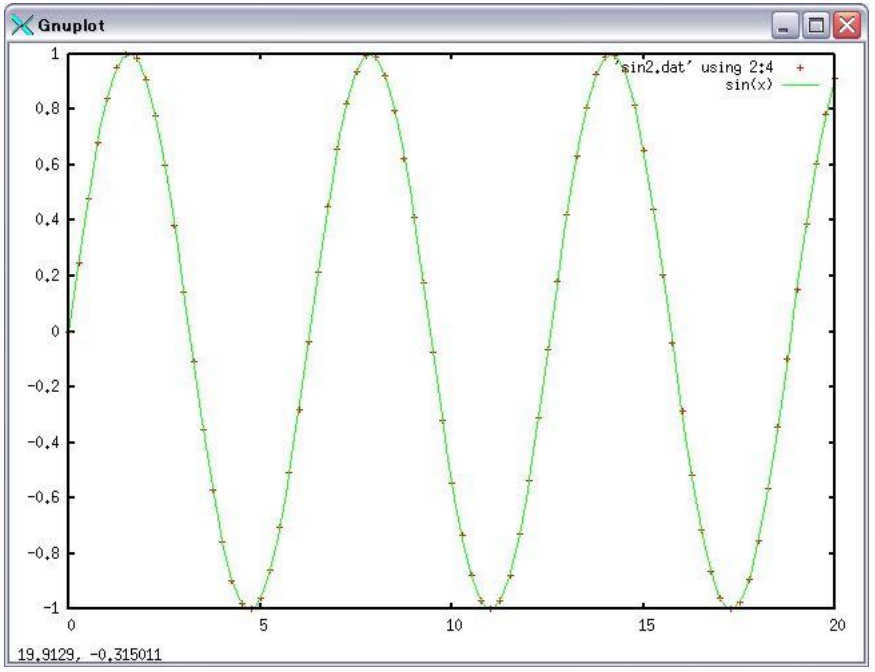
について、xが[0,20]の領域に対して、刻み幅(STEP)を0.25にしてEuler法とRunge-Kutta法で数値計算により解を求めよ。また厳密解との比較をgnuplotを用いて行え。

結果の例

Euler法



Runge-Kutta法



課題（数値微分） 提出期限：来週のこの時間

- 時刻 $t=0$ で、質量 M、速度 0 のロケットが、単位時間当り質量 μ　の燃料をロケットから見て速さ v で噴射している。
  ロケットの速度 $V(t)$ は運動量保存則から、以下の微分方程式

$$\frac{dV(t)}{dt} = \frac{\mu v}{M - \mu t} \quad \rightarrow \quad \frac{dy}{dx} = \frac{1}{1-x} \qquad \left(x = \frac{\mu}{M} t,\ y = \frac{V(t)}{v} \right)$$

  を満たす。
  - 微分方程式を導け
  - この微分方程式を、Euler法により数値微分により解き、厳密解との比較の図をgnuplotで描け。なお、step（刻み幅）は以下の２通り行え
    - ０．１
    - ０．０５
  - 上記2つの刻み幅の場合の厳密解との比較から、Euler法が正しく機能しているか論ぜよ。
  - この微分方程式を、Runge-Kutta法により、刻み幅０．１で数値微分により解き、厳密解との比較の図をgnuplotで描け。また、Euler法の結果と比較せよ。

応用（発展）

微分方程式の境界条件のある場合の解法

境界条件のある場合の微分方程式の数値解法

- **例題**

原点 $(x, y) = (0, 0)$ にある点電荷 q、すなわち電荷分布

$$\rho(x, y) = \begin{cases} q & x = y = 0 \\ 0 & \text{それ以外} \end{cases}$$

がつくり出す静電場のポテンシャルを、境界条件

$$\phi(\pm 1, \pm 1) = 0 \quad \text{復号任意}$$

の下で求めよ。

電磁場の方程式

- 時間に依存しないMaxwellの方程式

$$\nabla \cdot E = \rho(x)$$

$$\nabla \times E = 0$$

- 第2式の渦なしの条件から、電場は静電ポテンシャルを用いて以下のように現される

$$E = -\nabla \phi(x)$$

- Laplace-Poisson方程式

$$\nabla^2 \phi(x) = -\rho(x)$$

  - 2階の微分方程式
  - 境界条件がある

2階の微分方程式の解法

- 1階の微分の場合　→　差分で置き換えるのは悪い近似（Euler法）
- 2階の微分の場合　→　差分で置き換えるのは悪くない近似
- 次の2階の常微分方程式を考える

$$\frac{d^2}{dx^2}\phi(x)+g(x)=0$$

  - 区間[x_0, x_N]が間隔 h で N 個に分割されている（ $h=(x_N-x_0)/N$ ）
- Tayler展開

$$\phi(x_{i\pm1})=\phi(x_i)\pm h\frac{d}{dx}\phi(x_i)+\frac{h^2}{2}\frac{d^2}{dx^2}\phi(x_i)\pm\frac{h^3}{6}\frac{d^3}{dx^3}\phi(x_i)+O(h^4)$$

- 差分式での置き換え

$$\frac{d^2}{dx^2}\phi(x_i)\approx\frac{\phi(x_{i+1})-2\phi(x_i)+\phi(x_{i-1})}{h^2}$$

  - 奇数番目の項が±であるので、h^4 以上の項を無視した事に等しい（Euler法の場合は h^2 以上の項を無視）

差分方程式と境界値問題

- **差分方程式**

$$\phi(x_{i+1}) - 2\phi(x_i) + \phi(x_{i-1}) + h^2 g(x_i) = 0$$

- **解くべき微分方程式**

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & -1 & 2 & -1 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 \end{pmatrix} \begin{pmatrix} \phi(x_0) \\ \phi(x_1) \\ \phi(x_2) \\ \vdots \\ \phi(x_{N-1}) \\ \phi(x_N) \end{pmatrix} = \begin{pmatrix} \phi_0 \\ h^2 g(x_1) \\ h^2 g(x_2) \\ \vdots \\ h^2(x_{N-1}) \\ \phi_N \end{pmatrix}$$

  - *i=0* と *N* に関しては境界条件を適用
  - *i=1* ～ *N−1* は差分方程式から求める

解法

- 行列を用いる方法

$$AY = B \quad \rightarrow \quad Y = A^{-1}B$$

  - 一般に消去法などにより、A^{-1} を求めることは可能
  - 今考えている A は極めて疎な（0の多い）行列
    - 逆行列の計算は無駄が多く、また精度も出ない（計算回数による誤差の蓄積）
- Relaxation法
  - 近似解 $Y^{(k)}$ からより精度の高い解 $Y^{(k+1)}$ を求める

加速パラメータ
・ω=1～2で収束を加速
・問題の性質が不明な時はω=1が無難

$$Y_i^{(k+1)} = Y_i^{(k)} + \frac{\omega}{A_{ii}} \left[b_i - \sum_{j=1}^{i-1} A_{ij} Y_j^{(k+1)} - \sum_{j=i}^{N} A_{ij} Y_j^{(k)} \right]$$

1次元（1変数）の場合のRelaxation法の漸化式

- 1次元（1変数）の場合のRelaxation法の漸化式

$$\phi^{(k+1)}(x_i) = \phi^{(k)}(x_i) + \frac{\omega}{2}\left[h^2 g(x_i) + \phi^{(k+1)}(x_{i-1}) + \phi^{(k)}(x_{i+1}) - 2\phi^{(k)}(x_i)\right]$$

例題（電荷の作る静電ポテンシャル）の場合

- Laplace-Poisson方程式

$$\frac{\partial^2}{\partial x^2}\phi + \frac{\partial^2}{\partial y^2}\phi = -\rho(x,y)$$

- 2階微分を差分で近似

$$\frac{\partial^2}{\partial x^2}\phi \approx \frac{\phi(x_{i+1},y_j) - 2\phi(x_i,y_j) + \phi(x_{i-1},y_j)}{h^2}$$

$$\frac{\partial^2}{\partial y^2}\phi \approx \frac{\phi(x_i,y_{j+1}) - 2\phi(x_i,y_j) + \phi(x_i,y_{j-1})}{h^2}$$

- 差分方程式

$$\phi(x_{i+1},y_j) + \phi(x_{i-1},y_j) + \phi(x_i,y_{j+1}) + \phi(x_i,y_{j-1}) - 4\phi(x_i,y_j) = -h^2\rho(x_i,y_j)$$

- Relaxation法の漸化式

$$\phi^{(k+1)}(x_i,y_j) = \phi^{(k)}(x_i,y_j) + \omega\left[\frac{\phi^{(k)}(x_{i+1},y_j) + \phi^{(k)}(x_{i-1},y_j) + \phi^{(k)}(x_i,y_{j+1}) + \phi^{(k)}(x_i,y_{j-1})}{4} - \phi^{(k)}(x_i,y_j) + \frac{h^2}{4}\rho(x_i,y_j)\right]$$

プログラム(laplace)の説明
(/home/teacher/z6wt01in/SAMPLE/laplace.f)

```
      program laplace
      implicit none
c constants:
      integer N,LOOP
      parameter(N    =  50)              ! 区間の分割数
      parameter(LOOP = 200)              ! Relaxation 法繰り返し数
      real*8  H,W
      parameter(H = 1.0D0/N)             ! 刻み幅 (領域は [-0.5,0.5])
      parameter(W = 1.8D0)               ! 加速パラメータ
      integer LUNOUT,ERR
      parameter(LUNOUT=11,ERR=-1)        ! 出力機番及びエラー識別子
c local:
      real*8  rho(0:N,0:N)               ! 2 次元電荷分布
      real*8  phi(0:N,0:N)               ! 2 次元ポテンシャル
      real*8  corr,cmax                  ! 補正値とその最大値
      integer status                     ! ファイル操作の状態
      integer i,j,k
```

サブルーチン file_open用

プログラム(laplace)の説明(続き)
(/home/teacher/z6wt01in/SAMPLE/laplace.f)

```
c begin:
      do i=0,N
         do j=0,N
            rho(i,j)=0.0D0
            phi(i,j)=0.0D0
         end do
      end do
      rho(N/2,N/2) = 1.0D4 * H**2          ! 原点(N/2,N,2)に点電荷*H^2
c Relaxation法によるポテンシャル計算
      do k=0,LOOP
         cmax=0.0D0
         do i=1,N-1
            do j=1,N-1
               corr = w*(
     &              (phi(i-1,j)+phi(i,j-1)+phi(i+1,j)+phi(i,j+1))/4.0D0
     &              -phi(i,j)+rho(i,j)/4.0D0)
```

電荷分布と電位の初期値を全て0にする

補正項の最大値

Relaxation法の漸化式の補正項

プログラム(laplace)の説明(続き)
(/home/teacher/z6wt01in/SAMPLE/laplace.f)

補正項がそれまでの最大値
より大きければ、最大値を更新

```
            if (abs(corr).gt.cmax) cmax = abs(corr)
            phi(i,j) = phi(i,j) + corr
          end do
        end do
        if (mod(k,10).eq.0) then
          write(*,'(a,I3,a,E10.3)')      ! 収束の度合を確認
     &       'k = ',k,', Correction = ',cmax
        endif
      end do
```

補正を適用

mod(i,j)
iをjで割った余り

Kが10の倍数毎に

プログラム(laplace)の説明(続き)
(/home/teacher/z6wt01in/SAMPLE/laplace.f)

(±0.5、±0.5)の領域を縦横共にN分割(H=1/N)

- 横: $x = -0.5 + iH \quad (i = 1 \ldots N)$
- 縦: $y = -0.5 + jH \quad (j = 1 \ldots N)$

```
c 計算結果をファイルに出力
 1    continue
      call file_open('output',LUNOUT,status)
      if (status.eq.ERR) goto 1
      do j=0,N
         write(LUNOUT,'(2F7.3,E12.3)')
     &        (-0.5D0+i*H,-0.5D0+j*H,phi(i,j),i=0,N)
         write(LUNOUT,*) ! 結果をgnuplotのsplotで読み込む為に必要な空行
      end do
      stop
      end
```

演習

- サンプルプログラムは
  /home/teacher/z6wt01in/SAMPLE/
  に
  laplace.f file_open.f
  として置いてある

- 演習0：各人コピーし、実行ファイル laplace を
  % frt -o laplace laplace.f file_open.f
  により生成し実行せよ

- 演習１：加速パラメータを１から２の間の数に変えて実行し、
  k=200での補正項の大きさを比較せよ
- 演習２：LOOPの数を変えて、最後の補正項の大きさを比較せよ
- 演習３：gnuplot用のファイルをsplotコマンドで表示せよ

加速パラメータ依存性

- 結果
  - W = 1.0D0　　k = 200, Correction = 0.158e-02
  - W = 1.2D0　　k = 200, Correction = 0.150e-02
  - W = 1.4D0　　k = 200, Correction = 0.119e-02
  - W = 1.6D0　　k = 200, Correction = 0.539e-03
  - W = 1.8D0　　k = 200, Correction = 0.156e-04　最速
  - W = 2.0D0　　k = 200, Correction = 0.413e+00

LOOP数依存性

- 結果
  - k = 50, Correction = 0.538e-02
  - k = 100, Correction = 0.767e-03
  - k = 150, Correction = 0.110e-03
  - k = 200, Correction = 0.156e-04
  - k = 250, Correction = 0.223e-05
  - k = 300, Correction = 0.319e-06
  - k = 350, Correction = 0.456e-07
  - k = 400, Correction = 0.651e-08

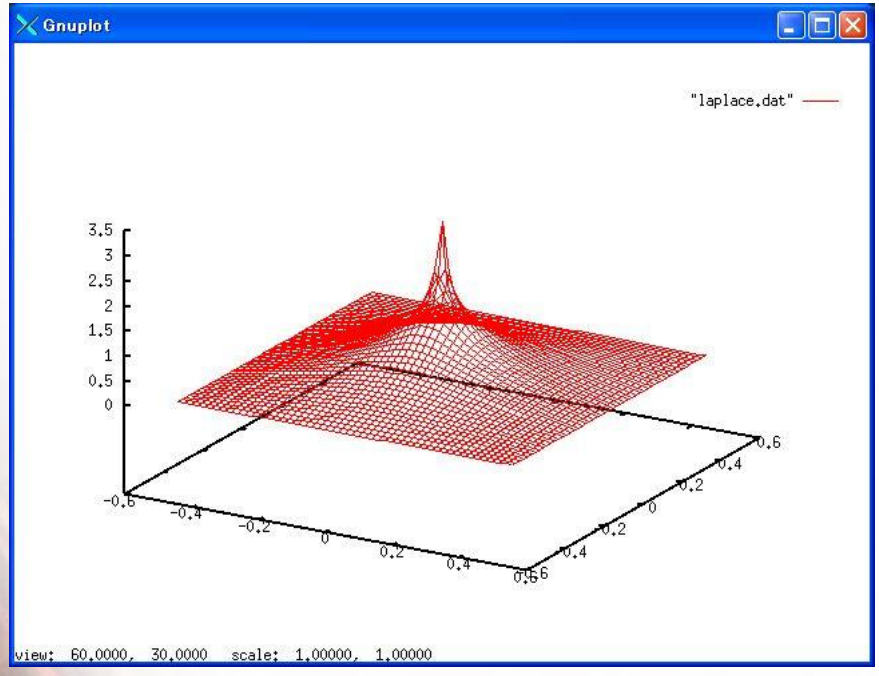
補正はより小さくなる
（正解に近づいている）

求めたい精度に合わせてLOOP数を選択する事！！

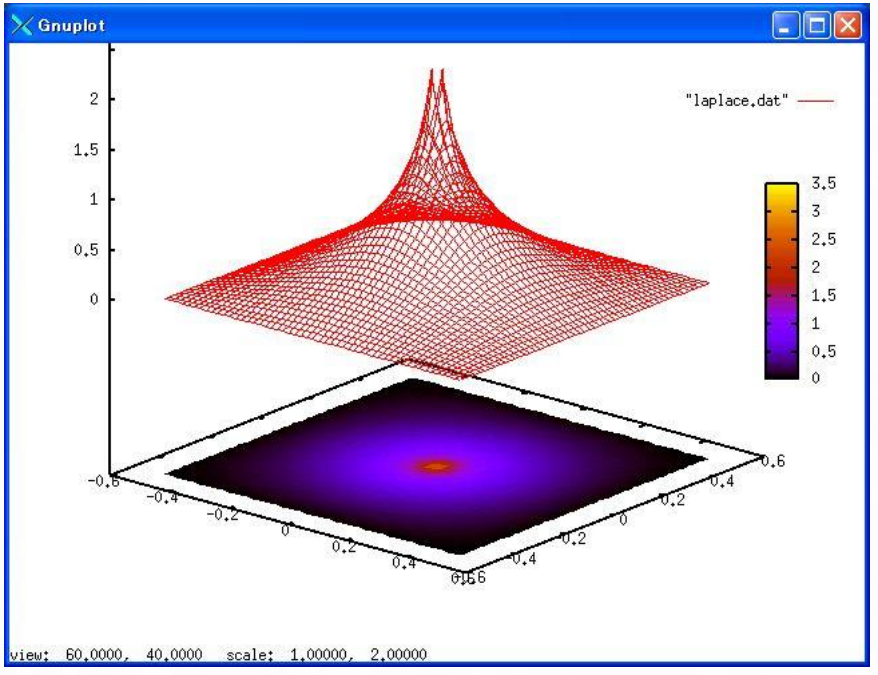
gnuplotによる計算結果の視覚化（グラフ化）

- 簡単な例
  - gnuplot> splot ‘laplace.dat’ with lines

gnuplotのプロンプト



- ちょっと凝った例
  - gnuplot> set view 60,40,1,2
  - gnuplot> set pm3d at b
  - gnuplot> splot ‘laplace.dat’ with lines



演習(応用問題)

- ４箇所

(−0.5, −0.5) に +q の電荷
(+0.5, +0.5) に +q の電荷
(−0.5, +0.5) に -q の電荷
(+0.5, −0.5) に -q の電荷

があるときの静電場ポテンシャルφを境界条件

$\phi(\pm 1, \pm 1) = 0$　復号任意

の下で求めよ。

演習(応用問題)解答例

mule@ah

File Edit Options Buffers Tools Fortran Help

```
      program laplace
      implicit none
c constants:
      integer N,LOOP
      parameter(N    = 100)                ! 区間の分割数
      parameter(LOOP = 200)                ! Relaxation法繰り返し数


      rho(  N/4,  N/4) = 1.0D4 * H**2      ! 点(-0.5,-0.5)に+点電荷*H^2
      rho(3*N/4,3*N/4) = 1.0D4 * H**2      ! 点( 0.5, 0.5)に+点電荷*H^2
      rho(  N/4,3*N/4) =-1.0D4 * H**2      ! 点(-0.5, 0.5)に-点電荷*H^2
      rho(3*N/4,  N/4) =-1.0D4 * H**2      ! 点( 0.5,-0.5)に-点電荷*H^2


      do j=0,N
         write(LUNOUT,'(2F7.3,E12.3)')
     &        (-1.0D0+i*H,-1.0D0+j*H,phi(i,j),i=0,N)
         write(LUNOUT,*) ! 結果をgnuplotのsplotで読み込む為に必要な空行
      end do
```

演習(応用問題)解答例

- gnuplot> set view 60,40,1,2
- gnuplot> set pm3d at b
- gnuplot> splot ‘laplace4.dat’ with lines