モンテカルロ法の例題

RC梁の耐力分布算定

信頼性工学特論

レポート課題例

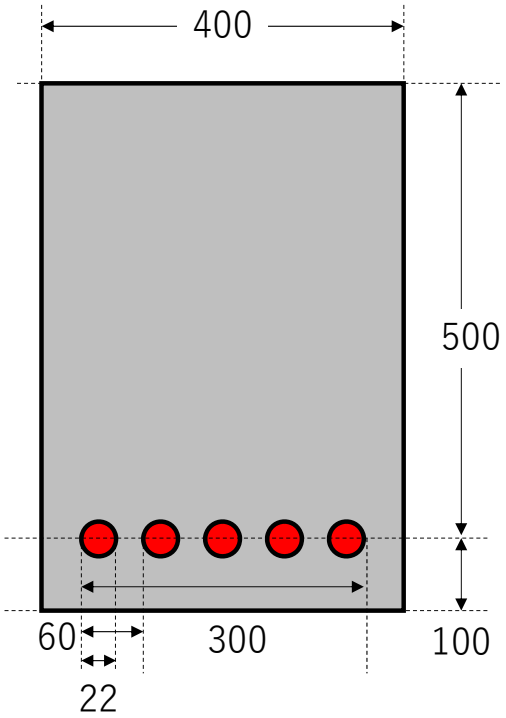
概要

- **純曲げを受けるRC梁**の終局曲げ耐力分布を求める

  - どうやって求めるか？…**モンテカルロ法**を利用
    - **材料パラメータ**（強度等）と**部材パラメータ**（寸法）を乱数とする
    - RC梁の上縁部がコンクリート圧壊歪ε_uに達する時点を**終局曲げ破壊**とする
    - ファイバー法による断面耐力の計算コードを作成する

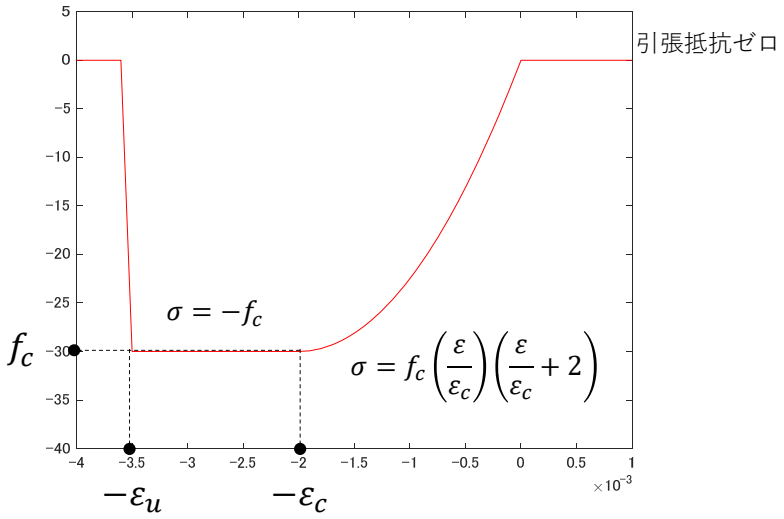
問題設定

- RC梁断面を右のように定める
  - **コンクリート**
  - **主鉄筋**D19×5本（5-D19）
    - 1本の断面積：1.986(cm^2)＝198.6(mm^2)
    - 5-D19の断面積：1435(mm^2)
  - **材料パラメータ全て**と**主鉄筋の断面積**が変動

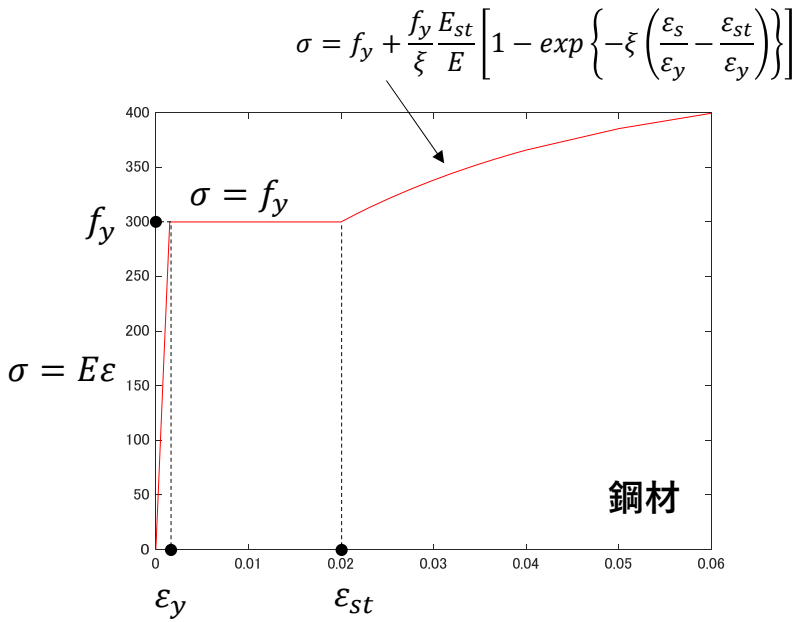


材料パラメータと応力－歪曲線

コンクリート



塑性化歪：ε_c　圧壊歪：ε_u　圧縮強度：f_c



ヤング率：E　降伏強度：f_y　降伏歪：$\varepsilon_y = \frac{E}{f_y}$
硬化歪：ε_{st}　硬化係数：E_{st}　硬化曲率：ξ

コンクリートの応力－歪曲線

strainがベクトルで与えられる場合を想定しIf文を避ける。
指定された範囲で1、それ以外で0を返す変数a、b、 c を用意して利用する

paramにコンクリートの材料パラメータを入れておく

Matlabでは 例えば
a=[1 2 3 4]に対して
a>2は[0 0 1 1]を意味する

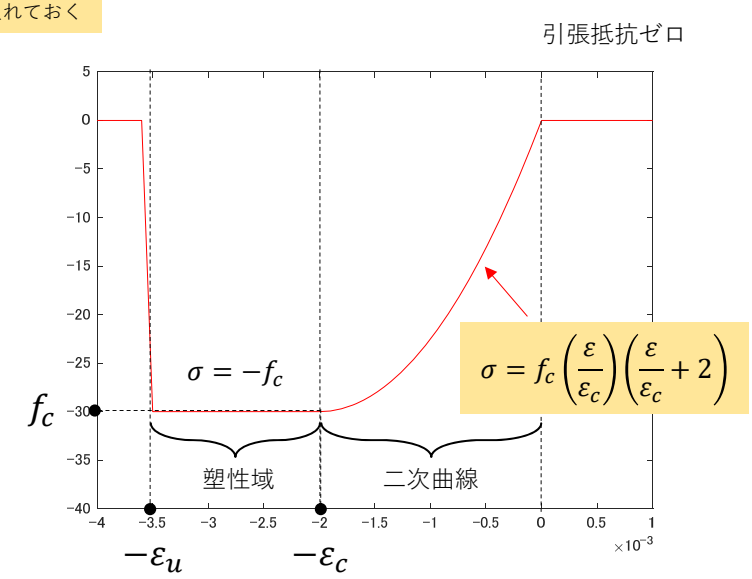
```
function [stress] = ss_concrete(strain, param)
%SS_CONCRETE returns stress corresponding to the given strain
ec = param.ec;
eu = param.eu;

a = strain <= 0; %-- tensile stress is zero

b = strain >= -ec;
s2 = param.fc * strain/ec.*(strain/ec+2);

c = strain >= -eu & strain < -ec;
s3 = -param.fc;

stress = a.*(b.*s2 + c.*s3);

end
```

引張域 $\sigma = 0$

二次曲線

塑性域

Matlabの掛け算は
（*）は行列積
（.*）は要素ごとの掛け算



鋼材の応力－歪曲線

paramに鋼材の材料パラメータを入れておく

```
function stress = ss_steal(strain, param)
%SS_STEAL returns stress corresponding to the given strain
ey = param.fy/param.E;

a=abs(strain)<ey;
s1 = param.E * strain;

b=abs(strain)<param.est & abs(strain)>=ey;
s2 = param.fy*ones(size(strain));
s2 = sign(strain).*s2;

c=abs(strain)>=param.est;
alpha = param.fy / param.xi * param.Est / param.E;
s3 = param.fy + alpha * (1-exp(-param.xi*(abs(strain)/ey-param.est/ey)));
s3 = sign(strain).*s3;

stress = a.*s1 + b.*s2 + c.*s3;

end
```

弾性域 $\sigma = E\varepsilon$

降伏棚 $\sigma = f_y$

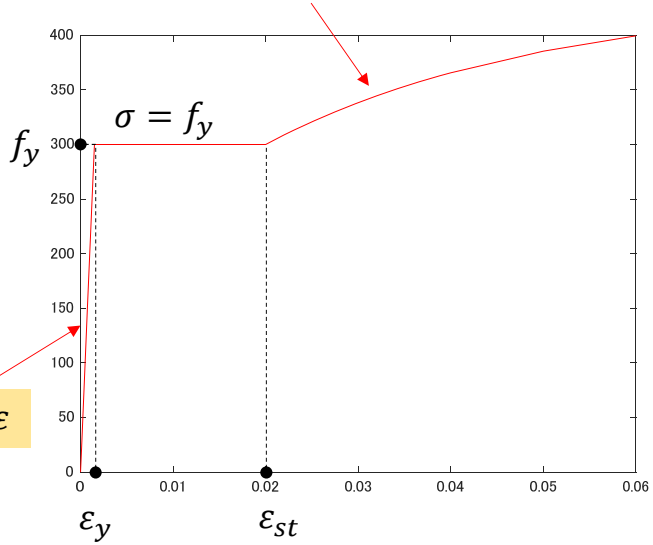
硬化後

strainがベクトルで与えられる場合を想定しIf文を避ける。
指定された範囲で1、それ以外で0を返す変数a、b、 c を用意して利用する

sign関数は正負の符号を返すMatlab標準関数
strain./abs(strain)に相当するが
ゼロが分母に来た時エラーが生じるため
sign関数を使うほうが便利

$$\sigma = f_y + \frac{f_y}{\xi}\frac{E_{st}}{E}\left[1 - exp\left\{-\xi\left(\frac{\varepsilon_s}{\varepsilon_y} - \frac{\varepsilon_{st}}{\varepsilon_y}\right)\right\}\right]$$

$\sigma = E\varepsilon$



検証①｜適切に動作するか確認

実行スクリプト

```
%-- Setting Parameters
fc = 30; %-- Compression Strength
ec= 0.002; %-- Platcization Strain
eu=0.0035; %-- Ultmate Strain
param_concrete = init_concrete(fc, ec, eu);

strain = [-4000:100:1000]/1000000;
stress = ss_concrete(strain ,param_concrete);
figure(3)
plot(strain, stress, 'r-')
ylim([-40 5])
```

正しく応力を返すか
あらゆる歪を与え、
グラフ化して確認する

コンクリートの材料パラメータをparamにまとめる

```
function [param] = init_concrete(fc, ec, eu)
%INIT_CONCRETE  sets the parameters of steel material
param.fc = fc;
param.ec = ec;
param.eu = eu;
end
```

歪を与えると応力を返す関数｜ss_concrete

```
function [stress] = ss_concrete(strain, param)
%SS_CONCRETE returns stress corresponding to the given strain
ec = param.ec;
eu = param.eu;

a = strain <= 0; %-- tensile stress is zero

b = strain >= -ec;
s2 = param.fc * strain/ec.*(strain/ec+2);

c = strain >= -eu & strain < -ec;
s3 = -param.fc;

stress = a.*(b.*s2 + c.*s3);

end
```

検証②｜適切に動作するか確認

実行スクリプト

```
%-- Setting Parameters
E = 200000; %-- Young Modulus 200GPa
fy= 300; %-- Yield Strength
xi=0.050; %-- Hardening Curvature
Est=4500; %-- Hardening Coefficient
est=0.020000; %-- Hardening Strain (20000u)
param_steel = init_steal(E,fy,xi,Est,est);

strain = [0:100:2000 4000:2000:40000 50000:10000:300000]'/1000000;
strain = [flipud(-strain);strain];
stress = ss_steal(strain,param_steel);

figure(1)
plot(strain, stress,'r-');
figure(2)
plot(strain,stress,'r-');
xlim([0 60000]/1000000)
```

正しく応力を返すか
あらゆる歪を与え、
グラフ化して確認する

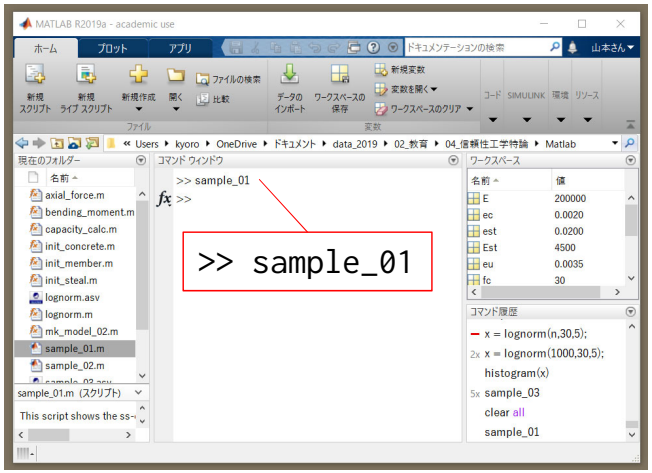
鋼材の材料パラメータをparamにまとめる

```
function [param] = init_steal(E,fy,xi,Est,est)
%INIT_STEAL sets the parameters of steel material
param.E = E;
param.fy = fy;
param.xi = xi;
param.Est = Est;
param.est = est;
end
```

歪を与えると応力を返す関数｜ss_steal

```
function stress = ss_steal(strain, param)
%SS_STEAL returns stress corresponding to the given strain
ey = param.fy/param.E;

a=abs(strain)<ey;
s1 = param.E * strain;

b=abs(strain)<param.est & abs(strain)>=ey;
s2 = param.fy*ones(size(strain));
s2 = sign(strain).*s2;

c=abs(strain)>=param.est;
alpha = param.fy / param.xi * param.Est / param.E;
s3 = param.fy + alpha * (1-exp(-param.xi*(abs(strain)/ey-param.est/ey)));
s3 = sign(strain).*s3;

stress = a.*s1 + b.*s2 + c.*s3;

end
```

検証③｜適切に動作するか確認



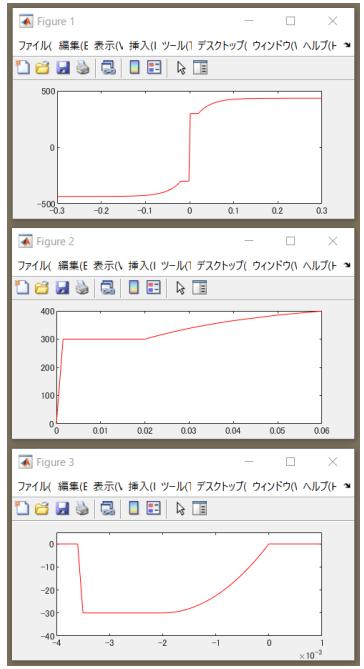
ss_concreteとss_stealが
正しく動作することを確認

sample_01を実行

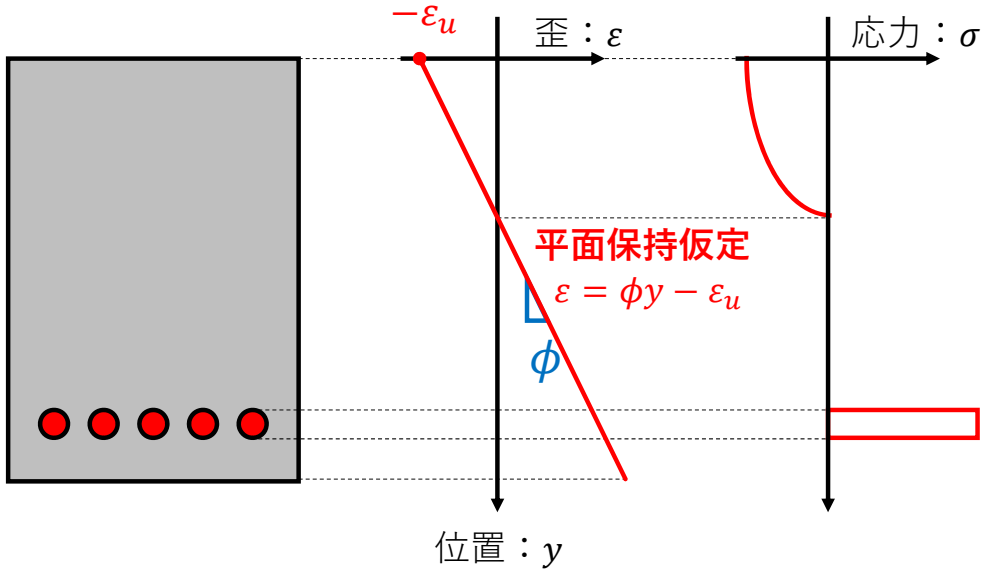
```
%-- Setting Parameters
fc = 30; %-- Compression Strength
ec= 0.002; %-- Platcization Strain
eu=0.0035; %-- Ultmate Strain
param_concrete = init_concrete(fc, ec, eu);

strain = [-4000:100:1000]/1000000;
stress = ss_concrete(strain ,param_concrete);
figure(3)
plot(strain, stress, 'r-')
ylim([-40 5])
```

```
%-- Setting Parameters
E = 200000; %-- Young Modulus 200GPa
fy= 300; %-- Yield Strength
xi=0.050; %-- Hardening Curvature
Est=4500; %-- Hardening Coefficient
est=0.020000; %-- Hardening Strain (20000u)
param_steel = init_steal(E,fy,xi,Est,est);

strain = [0:100:2000 4000:2000:40000 50000:10000:300000]'/1000000;
strain = [flipud(-strain);strain];
stress = ss_steal(strain,param_steel);

figure(1)
plot(strain, stress,'r-');
figure(2)
plot(strain,stress,'r-');
xlim([0 60000]/1000000)
```

実行結果



軸力の計算方法



平面保持仮定が成り立つと仮定
（歪分布は直線）

終局限界状態では$y = 0$で$\varepsilon = -\varepsilon_u$
未知数は**曲率**：ϕのみ

$$\varepsilon = \phi y - \varepsilon_u$$

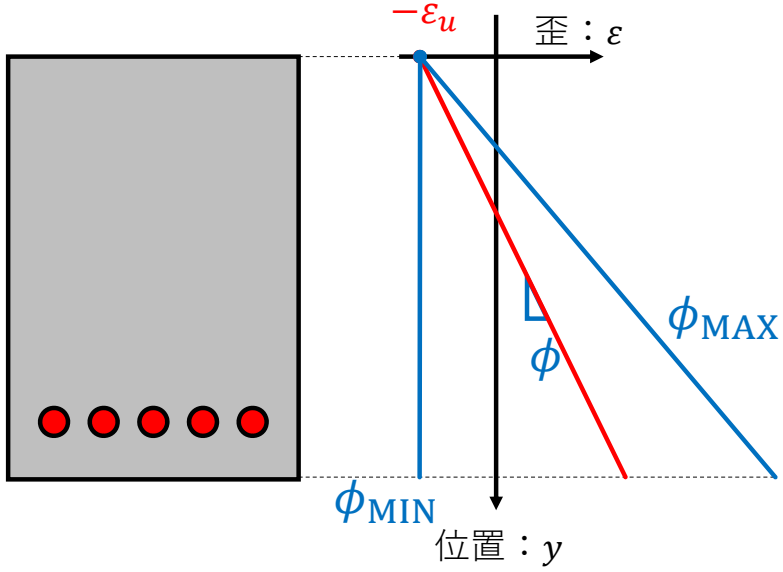
σはεから求められる

軸力Nはゼロなので

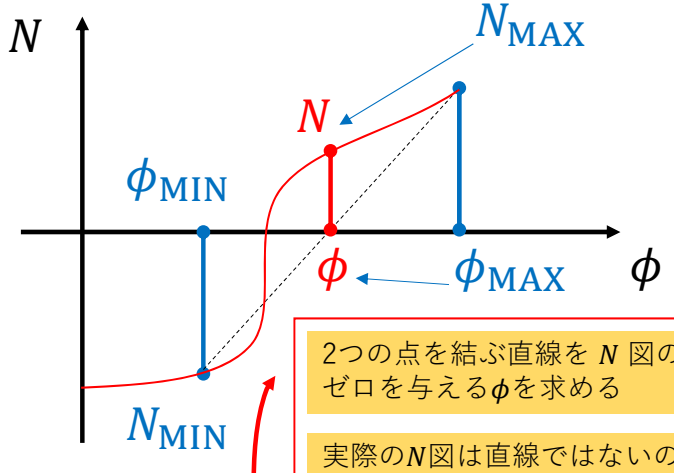
$$N = \int_S \sigma \, \mathrm{d}A = \int_0^H \sigma B \mathrm{d}y + \sigma_s A_s = 0$$

を満たす**曲率**：ϕを求めれば良い

ニュートン法による曲率の計算



ϕ の値を最大値と最小値で適当に決め、
それぞれの場合で軸力Nを求める



2つの点を結ぶ直線を N 図の推定値として、
ゼロを与えるϕを求める

実際のN図は直線ではないので、ここでの
ϕに対するN値はゼロとはならない

Nがゼロより大きいときは最大値を、
ゼロより小さいときは最小値を更新する

くり返すとゼロに収束するはず

軸力の計算

itr_maxは繰り返し上限値
収束しなかった場合、永遠に終わらないことのないように打ち切る回数を決めておく

ニュートン法では、Nはゼロに近づくだけでゼロにはならないので、err_maxでゼロと見なす最大値を定めておく

```
param=mk_model_02();
itr_max = 500;
err_max = 0.001; %-- it is assumed to be zero

%-- fiber
y =  param.member.y;
eu = param.concrete.eu;
hs = param.member.hs;

%-- min/max curvature
phi1 = 0;
phi2 = 0.1;

%-- strain distribution
ec1 = -eu + phi1 * y;
er1 = -eu + phi1 * hs;
ec2 = -eu + phi2 * y;
er2 = -eu + phi2 * hs;

%-- stress distribution
sc1 = ss_concrete(ec1, param.concrete);
sr1 =  ss_steal(er1, param.steel);
sc2 = ss_concrete(ec2, param.concrete);
sr2 =  ss_steal(er2, param.steel);

%-- axial force
N1=axial_force(y, sc1, sr1, param);
N2=axial_force(y, sc2, sr2, param);
```

ecはコンクリートの歪分布（ベクトル）

ε分布をϕ_{MIN}とϕ_{MAX}のそれぞれで求める

erは鉄筋の歪（スカラー）

scはコンクリートの応力分布（ベクトル）

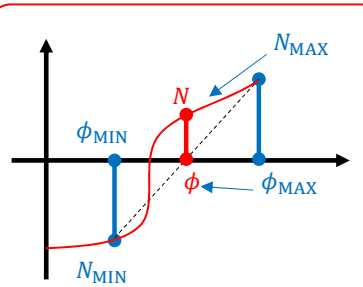
予め作っておいたss_concrete、ss_stealで応力分布を求める

srは鉄筋の応力（スカラー）

軸力を求める

axial_forceという関数を新たに作る

```
%-- Newton method
for kk = 1:itr_max
    %-- calculating next phi
    a=(N2-N1)/(phi2-phi1);
    phi=phi2-N2/a;

    %-- calculating stress distribution
    ec = -eu + phi * y;
    er = -eu + phi * hs;
    sc = ss_concrete(ec, param.concrete);
    sr =  ss_steal(er, param.steel);
    N = axial_force(y, sc, sr, param);

    %-- checking axial force
    if N<=err_max
        break;
    elseif N<0
        phi1 = phi;
        N1 = N;
    elseif N>0
        phi2 = phi;
        N2 = N;
    else
        %-- not used
        break;
    end
end

y0=eu/phi;
M = bending_moment(y,y0,sc, sr, param);
```



ϕ_{MIN}とN_{MIN}、
ϕ_{MAX}とN_{MAX}から
ϕとNを求める

axial_force関数はここでも使う

軸力Nが**ゼロ**（err_max以下）ならbreak（ループを出る）

ϕ_{MIN}かϕ_{MAX}をϕで更新

軸力ゼロのときのϕ（から求められる応力分布）で曲げモーメントを求める

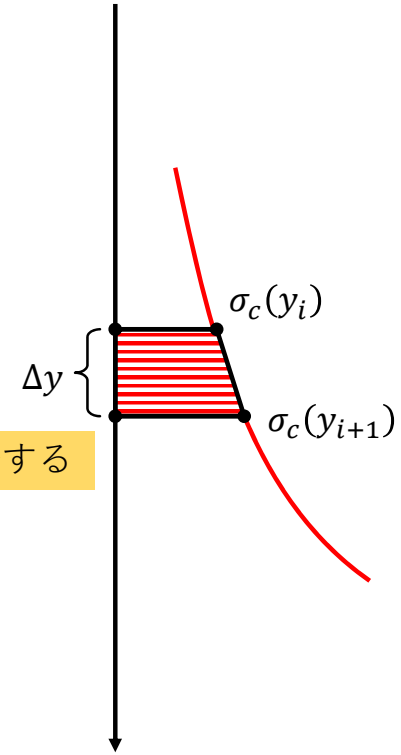
軸力の計算

```
function [N] = axial_force(y, sc, sr, param)
%AXIAL_FORCE returns axial force of the RC beam

Nc = 0;

for ii=1:(length(y)-1)
    dy = y(ii+1)-y(ii);
    bw = param.member.bw;
    Nc =  Nc + (sc(ii)+sc(ii+1)) * dy / 2 * bw;
end

As = param.member.As;
Ns = sr * As;

N = Nc + Ns;

end
```

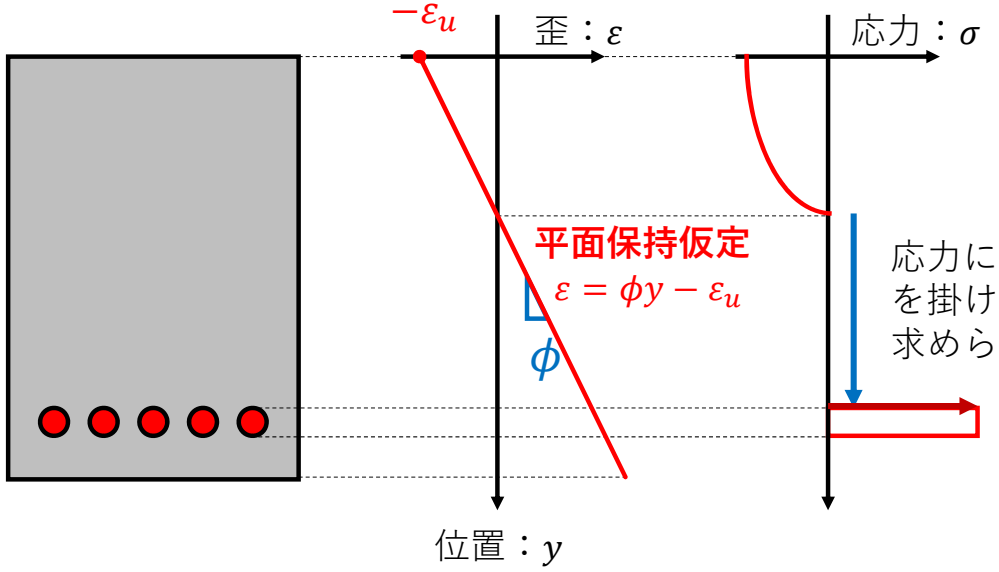
Ncはコンクリートが分担する軸力

応力分布を台形近似しながら積分する

Nsは鉄筋が分担する軸力



曲げモーメントの計算



$$M = \int_S \sigma(y - y_0)\,\mathrm{d}A$$

$$= \int_0^H \sigma(y - y_0)B\mathrm{d}y + \sigma_s(h - y_0)A_s$$

応力に**腕の長さ**（中立軸からの距離）
を掛け算するとその応力のモーメントが
求められる

曲げモーメントの計算

y0は中立軸位置

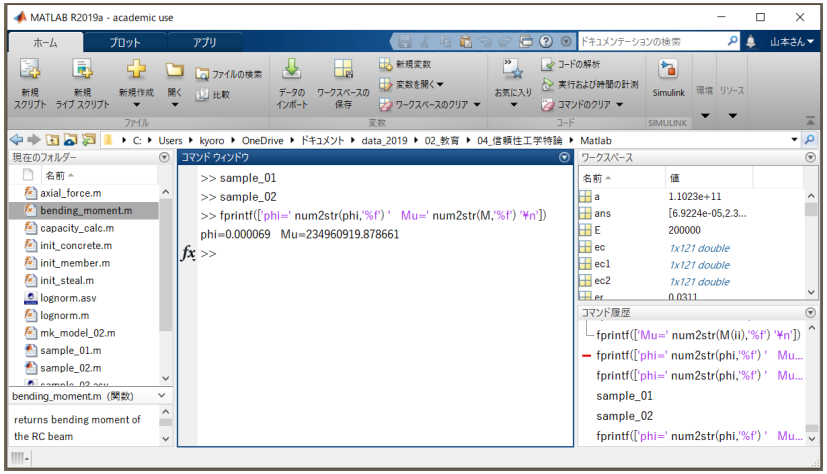
```
function [M] = bending_moment(y, y0, sc, sr, param)
%BENDING_MOMENT returns bending moment of the RC beam

Mc = 0;
for ii=1:(length(y)-1)
    dy = y(ii+1)-y(ii);
    bw = param.member.bw;
    Mc =  Mc + (sc(ii)*(y(ii)-y0)+sc(ii+1)*(y(ii+1)-y0)) * dy / 2 * bw;
end

As = param.member.As;
hs = param.member.hs;
Ms = sr * As * (hs-y0);

M = Mc + Ms;
end
```

応力分布を台形近似しながら積分する

中立軸位置からの腕の長さを掛け算して
応力モーメントを求めてから積分する

検証④｜正しく動作するか確認

sample_02を実行



さらに、
>> fprintf([‘phi=‘ num2str(phi,’%f’) ‘ Mu=‘ num2str(M,’%f’) ‘¥n’])
を実行すれば収束後のphiとMの値が確認できる

```
param=mk_model_02();
itr_max = 500;
err_max = 0.001; %-- it is assumed to be zero

%-- fiber
y =  param.member.y;
eu = param.concrete.eu;
hs = param.member.hs;

%-- min/max curvature
phi1 = 0;
phi2 = 0.1;

%-- strain distribution
ec1 = -eu + phi1 * y;
er1 = -eu + phi1 * hs;
ec2 = -eu + phi2 * y;
er2 = -eu + phi2 * hs;

%-- stress distribution
sc1 = ss_concrete(ec1, param.concrete);
sr1 =  ss_steal(er1, param.steel);
sc2 = ss_concrete(ec2, param.concrete);
sr2 =  ss_steal(er2, param.steel);

%-- axial force
N1=axial_force(y, sc1, sr1, param);
N2=axial_force(y, sc2, sr2, param);
```

```
%-- Newton method
for kk = 1:itr_max
    %-- calculating next phi
    a=(N2-N1)/(phi2-phi1);
    phi=phi2-N2/a;

    %-- calculating stress distribution
    ec = -eu + phi * y;
    er = -eu + phi * hs;
    sc = ss_concrete(ec, param.concrete);
    sr =  ss_steal(er, param.steel);
    N = axial_force(y, sc, sr, param);

    %-- checking axial force
    if N<=err_max
        break;
    elseif N<0
        phi1 = phi;
        N1 = N;
    elseif N>0
        phi2 = phi;
        N2 = N;
    else
        %-- not used
        break;
    end
end

y0=eu/phi;
M = bending_moment(y,y0,sc, sr, param);
```

曲げモーメントの繰り返し計算の準備

```
function [M, phi] = capacity_calc(param)
%CAPACITY_CALC returns capacity of RC cross section from given parameters

itr_max = 500;
err_max = 0.001; %-- it is assumed to be zero

%-- fiber
y =  param.member.y;
eu = param.concrete.eu;
hs = param.member.hs;

%-- min/max curvature
phi1 = 0;
phi2 = 0.1;

%-- strain distribution
ec1 = -eu + phi1 * y;
er1 = -eu + phi1 * hs;
ec2 = -eu + phi2 * y;
er2 = -eu + phi2 * hs;

%-- stress distribution
sc1 = ss_concrete(ec1, param.concrete);
sr1 =  ss_steal(er1, param.steel);
sc2 = ss_concrete(ec2, param.concrete);
sr2 =  ss_steal(er2, param.steel);

%-- axial force
N1=axial_force(y, sc1, sr1, param);
N2=axial_force(y, sc2, sr2, param);
```

sample_02は元々スクリプトだが、paramを入力引数とする**関数**へ変更

```
%-- Newton method
for kk = 1:itr_max
    %-- calculating next phi
    a=(N2-N1)/(phi2-phi1);
    phi=phi2-N2/a;

    %-- calculating stress distribution
    ec = -eu + phi * y;
    er = -eu + phi * hs;
    sc = ss_concrete(ec, param.concrete);
    sr =  ss_steal(er, param.steel);
    N = axial_force(y, sc, sr, param);

    %-- checking axial force
    if N<=err_max
        break;
    elseif N<0
        phi1 = phi;
        N1 = N;
    elseif N>0
        phi2 = phi;
        N2 = N;
    else
        %-- not used
        break;
    end
end

y0=eu/phi;
M = bending_moment(y,y0,sc, sr, param);

end
```

Mを返す

モンテカルロ法の準備

Matlabでは**対数正規分布**を返す関数LognormalDistributionを使用するには
Statistics and Machine Learning Toolboxを購入する必要がある

平均と**分散**を与えると、対数正規分布のパラメータμとσを求めて
対数正規乱数を発生させる関数を自作した

```
function [x] = lognorm(n,mean,sd)
%LOGNORM returns n random values obeying lognorm distribution

mu = log(mean^2/sqrt(mean^2+sd^2));
sigma = sqrt(log(sd^2/mean^2+1));

x = exp(sigma*randn(n,1)+mu);

end
```

randnは標準正規分布乱数を
発生させる標準関数

信頼性工学特論
話題提供#02の資料参照

Xが**対数正規分布**（平均：μ_X、分散：σ_X^2）なら、
Y（$=\ln X$）は**正規分布**（平均：μ_Y、分散σ_Y^2）

$$Y = \ln X$$

$$\mu_Y = \ln \frac{\mu_X^2}{\sqrt{\mu_X^2 + \sigma_X^2}} \qquad \sigma_Y^2 = \ln\left(\frac{\sigma_X^2}{\mu_X^2} + 1\right)$$

曲げモーメント使用

```
n=1000;

%-- material
fc = lognorm(n, 30, 5); %-- Compression Strength
ec = lognorm(n, 0.002000, 0.000200); %-- Platcization Strain
eu = lognorm(n, 0.003500, 0.000200); %-- Ultmate Strain

E = lognorm(n, 200000, 1000); %-- Young Modulus 200GPa
fy= lognorm(n, 300, 10); %-- Yield Strength
xi=lognorm(n, 0.050, 0.010); %-- Hardening Curvature
Est=lognorm(n, 4500, 1000); %-- Hardening Coefficient
est=lognorm(n, 0.020000, 0.003000); %-- Hardening Strain (20000u)

%-- member
H = lognorm(n, 600,1); %-- height
bw=lognorm(n, 400, 3); %-- width [mm]
As=lognorm(n, 1435, 1); %-- Area of Riber
hs=lognorm(n, 500, 1); %-- height of Riber

M=zeros(n,1);
phi=zeros(n,1);
for ii = 1:n
   y=0:H(ii)/120:H(ii); %-- fiber height
   concrete = init_concrete(fc(ii), ec(ii), eu(ii));
   steel = init_steal(E(ii),fy(ii),xi(ii),Est(ii),est(ii));
   member = init_member(y, bw(ii), As(ii), hs(ii));
   param.concrete = concrete;
   param.steel = steel;
   param.member = member;
   [M(ii), phi(ii)] = capacity_calc(param);
   fprintf([num2str(ii,'%0.4d') ': phi=' num2str(phi(ii),'%f')])
   fprintf(['   Mu=' num2str(M(ii),'%f') '¥n'])
end
histogram(M)
```

n個発生させた乱数の組み合わせの内、
ii番目の値の組み合わせ（粒子という）を用いて
終局曲げ耐力を計算

sample_03

材料・部材パラメータの値を乱数として発生させる

たとえばfc（･･･f_c：コンクリート圧縮強度）は
平均：30[N/mm^2]、分散：5[N/mm^2]とした

ニュートン法で**曲げモーメント**を算出

sample_02を関数化した
capacity_calc関数を利用

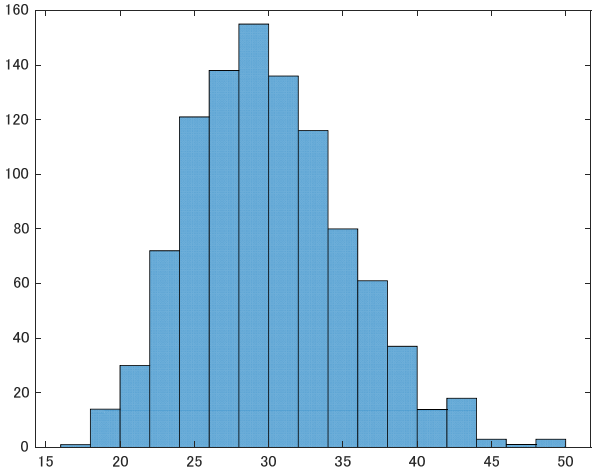
以下を実行する

```
>> sample_03
```

実行結果

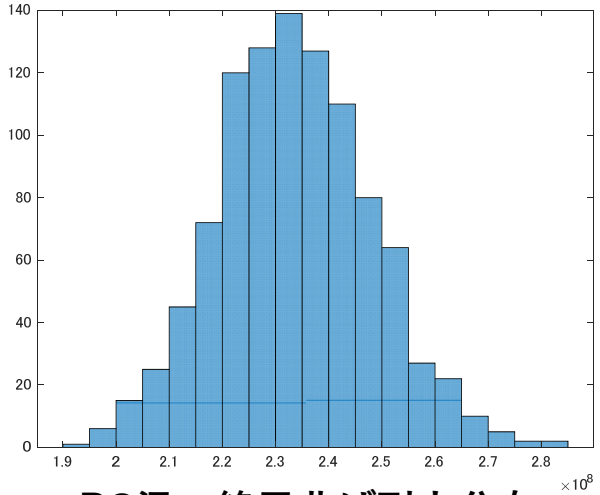
材料・部材パラメータを対数正規乱数とした時の**RC梁の終局曲げ耐力分布**が得られた

```
>>histogram(fc)
```



コンクリートの圧縮強度

```
>>histogram(M)
```



RC梁の終局曲げ耐力分布