最优化第十二次作业

张晋 15091060

2017 年 12 月 4 日

5.9 首先画出 Rosenbrock 函数的图像，本题中 Armijo 线搜索的参数为 $\gamma = 0.5, \rho = 0.01$.

最后分别以梯度下降法和牛顿法迭代，并画出等高线、运动轨迹、迭代值如下:

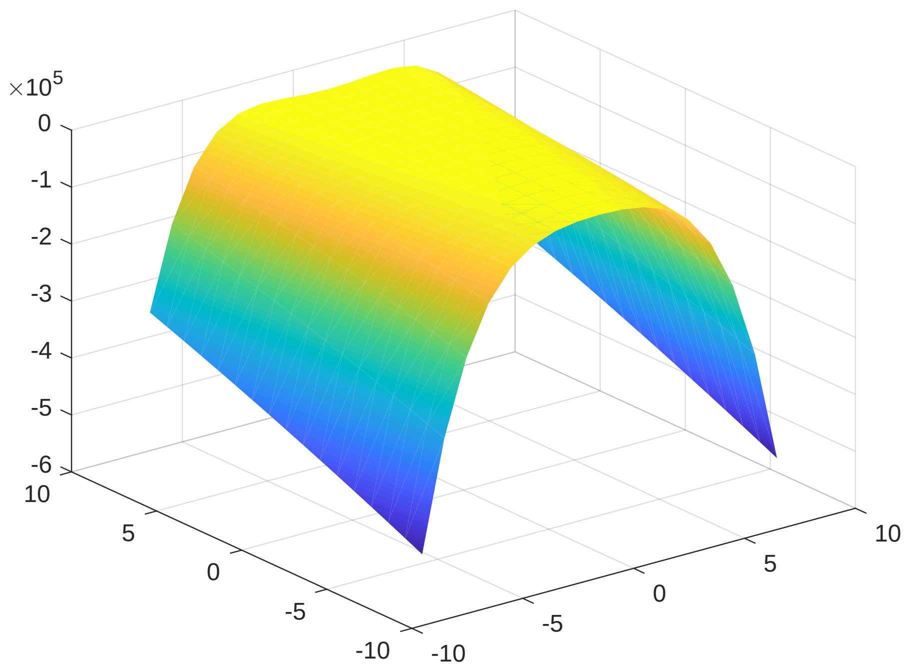

图 1: Rosenbrock 函数图像

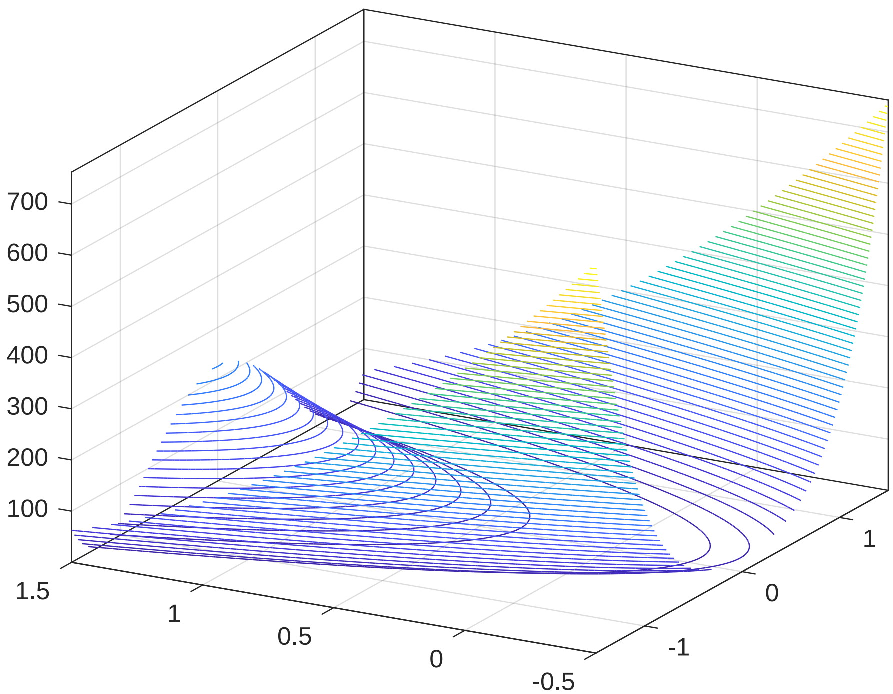

图 2: 在 $(1,1)$ 处附近的三维等高线

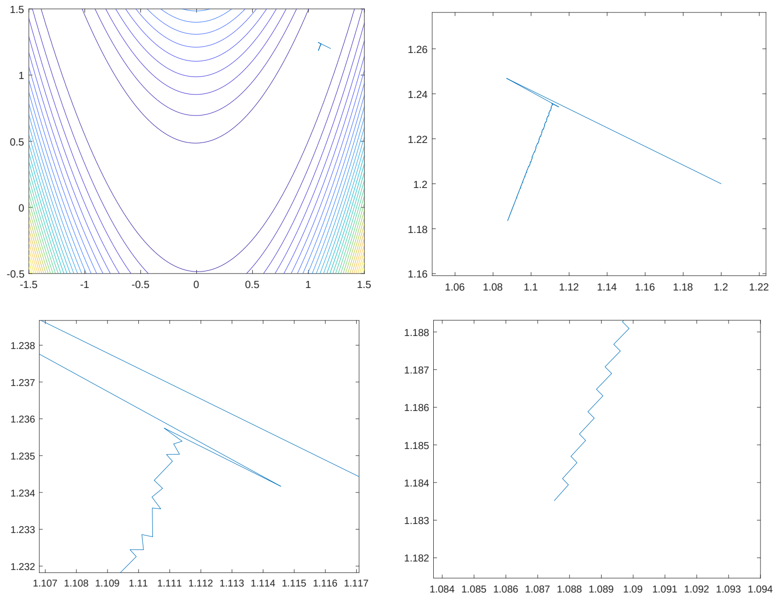

图 3: Steepest-denscent in (1.2,1.2)

Step[190]: x=[1.088846 1.186477] optim_fx=0.007973
Step[191]: x=[1.089052 1.186303] optim_fx=0.007937
Step[192]: x=[1.088577 1.185882] optim_fx=0.007924
Step[193]: x=[1.088779 1.185710] optim_fx=0.007889
Step[194]: x=[1.088310 1.185288] optim_fx=0.007874
Step[195]: x=[1.088507 1.185118] optim_fx=0.007841
Step[196]: x=[1.088044 1.184696] optim_fx=0.007825
Step[197]: x=[1.088236 1.184529] optim_fx=0.007793
Step[198]: x=[1.087780 1.184104] optim_fx=0.007776
Step[199]: x=[1.087965 1.183941] optim_fx=0.007745
Step[200]: x=[1.087517 1.183515] optim_fx=0.007727
最速下降法,共迭代 200 步
结果：
x=[1.087517e+00 1.183515e+00] optim_fx=0.007727

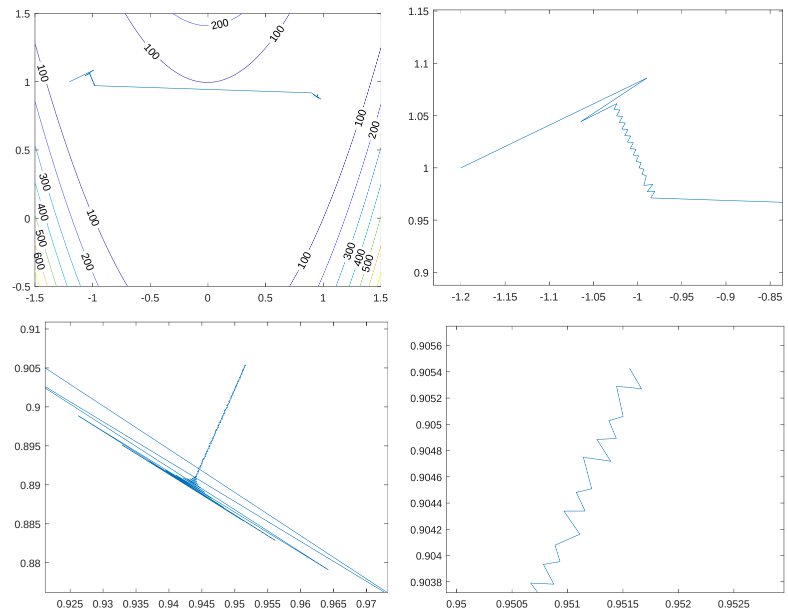

图 4: Steepest-denscent in (-1.2,1)

Step[190]: x=[0.951079 0.904481] optim_fx=0.002394

Step[191]: x=[0.951218 0.904509] optim_fx=0.002389

Step[192]: x=[0.951143 0.904748] optim_fx=0.002388

Step[193]: x=[0.951390 0.904719] optim_fx=0.002381

Step[194]: x=[0.951265 0.904884] optim_fx=0.002375

Step[195]: x=[0.951440 0.904892] optim_fx=0.002370

Step[196]: x=[0.951373 0.905027] optim_fx=0.002365

Step[197]: x=[0.951501 0.905060] optim_fx=0.002361

Step[198]: x=[0.951442 0.905290] optim_fx=0.002358

Step[199]: x=[0.951668 0.905271] optim_fx=0.002352

Step[200]: x=[0.951559 0.905427] optim_fx=0.002347

最速下降法,共迭代 200 步

结果：

x=[9.515588e-01 9.054274e-01] optim_fx=0.002347

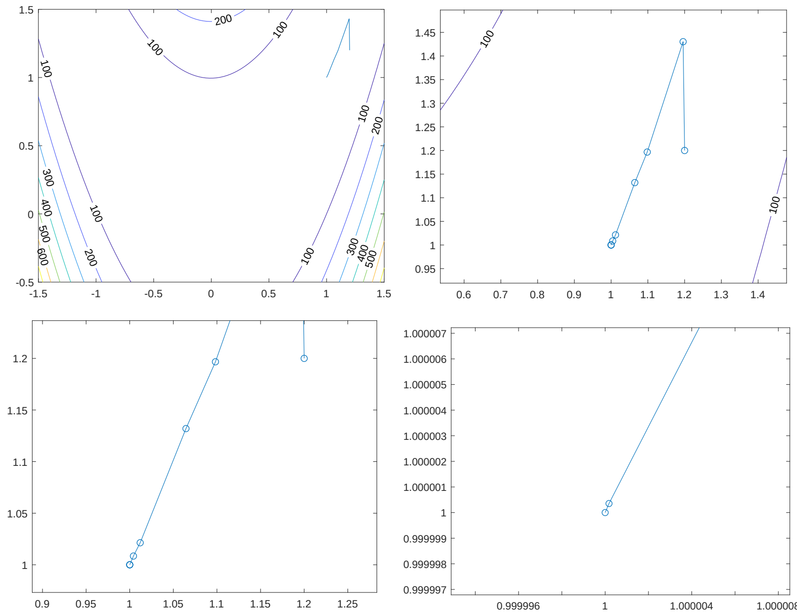

图 5: Newton-Armijo in (1.2,1.2)

Step[1]: x=[1.200000 1.200000] optim_fx=5.800000

Step[2]: x=[1.195918 1.430204] optim_fx=0.038384

Step[3]: x=[1.098284 1.196688] optim_fx=0.018762

Step[4]: x=[1.064488 1.131993] optim_fx=0.004289

Step[5]: x=[1.011992 1.021372] optim_fx=0.000903

Step[6]: x=[1.004261 1.008481] optim_fx=0.000019

Step[7]: x=[1.000050 1.000083] optim_fx=0.000000

Step[8]: x=[1.000000 1.000000] optim_fx=0.000000

Step[9]: x=[1.000000 1.000000] optim_fx=0.000000

牛顿 Armijo 回溯法,,共迭代 9 步

结果：

x=[1.000000e+00 1.000000e+00] optim_fx=0.000000

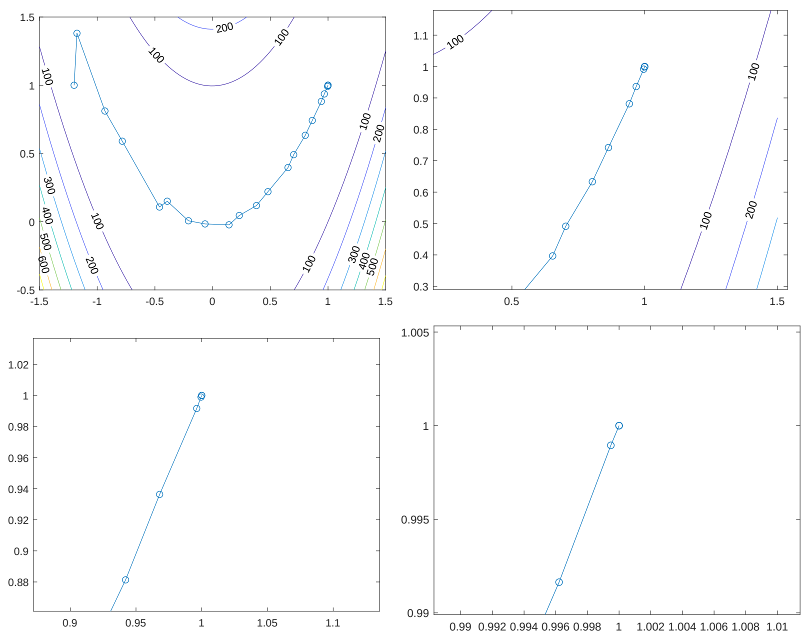

图 6: Newton-Armijo in (-1.2,1)

Step[15]: x=[0.802786 0.633221] optim_fx=0.051535

Step[16]: x=[0.863491 0.741931] optim_fx=0.019993

Step[17]: x=[0.942079 0.881336] optim_fx=0.007169

Step[18]: x=[0.967992 0.936337] optim_fx=0.001070

Step[19]: x=[0.996210 0.991639] optim_fx=0.000078

Step[20]: x=[0.999479 0.998948] optim_fx=0.000000

Step[21]: x=[0.999999 0.999998] optim_fx=0.000000

Step[22]: x=[1.000000 1.000000] optim_fx=0.000000

牛顿 Armijo 回溯法,,共迭代 22 步

结果：

x=[1.000000e+00 1.000000e+00] optim_fx=0.000000

5.19 迭代次数和求解的值如下:

n=5	**n=8**	**n=12**	**n=20**
k=6	k=19	k=35	k=66
x	x	x	x
5.00E+00	5.90E-11	-9.61E+00	-1.10E+01
-1.20E+02	-6.97E-11	8.15E+02	1.05E+03
6.30E+02	-5.16E-10	-1.65E+04	-2.40E+04
-1.12E+03	1.12E-09	1.36E+05	2.20E+05
6.30E+02	3.17E-10	-5.36E+05	-9.65E+05
	-6.55E-10	1.03E+06	1.99E+06
	-6.57E-10	-6.43E+05	-1.25E+06
	4.59E-10	-6.58E+05	-1.34E+06
		8.04E+05	8.83E+05
		6.63E+05	1.69E+06
		-1.24E+06	3.88E+05
		4.66E+05	-1.31E+06
			-1.71E+06
			-5.28E+05
			1.21E+06
			2.00E+06
			9.45E+05
			-1.43E+06
			-2.65E+06
			1.89E+06

Step	$x^{(k)}$
1	$(0, 0, 0, 0)$
2	$(0.9045, 0, -1.8090, -2.0225)$
3	$(-0.4472, -1.8944, -3.3416, -2.7889)$

5.21

$$G = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}$$

$$P = \begin{bmatrix} g \\ G * g \\ G^2 * g \\ G^3 * g \end{bmatrix}^T = \begin{bmatrix} -1 & 0 & 2 & \sqrt{5} \\ -2 & -1 & 4-\sqrt{5} & 2\sqrt{5}-2 \\ -3 & \sqrt{5}-4 & 11-4\sqrt{5} & 5\sqrt{5}-8 \\ -\sqrt{5}-2 & 6\sqrt{5}-16 & 34-14\sqrt{5} & 14\sqrt{5}-27 \end{bmatrix}$$

计算 MatrixRank[P]=2, 故可知其独立向量只有两个

附录代码

```
%绘制Rosenbrock函数图形
x=[-8:1:8];
y=x;
[X,Y]=meshgrid(x,y);
[row,col]=size(X);
for l=1:col
    for h=1:row
        z(h,l)=Rosenbrock([X(h,l),Y(h,l)]);
    end
end
surf(X,Y,z);
shading interp

function result=Rosenbrock(x)
%Rosenbrock 函数
%输入x,给出相应的y值,在x=(1,1) 处有全局极小点0,为得到最大值，返回值取
    相反数
[row,col]=size(x);
if row>1
    error('输入的参数错误');
end
result=100*(x(1,2)-x(1,1)^2)^2+(x(1,1)-1)^2;
result=-result;
end
```

```
%本实验采用最速下降法，步长为精确步长，测试函数为Rosenbrock函数
%%
%用符号表达式定义目标函数
clc;
clear;
syms x1 x2;
X=[x1,x2];
f=100*(X(2)-X(1)^2)^2+(1-X(1))^2;
F=eval(['@(x1,x2)',vectorize(f)]);
fx=diff(f,x1); %求f对x1偏导数
fy=diff(f,x2); %求f对x2偏导数
```

```
fxx=diff(fx,x1); %求二阶偏导数 对x1再对x1
fxy=diff(fx,x2); %求二阶偏导数 对x1再对x2
fyx=diff(fy,x1); %求二阶偏导数 对x2再对x1
fyy=diff(fy,x2); %求二阶偏导数 对x2再对x2
Gradient=[fx;fy];  %计算梯度表达式
Hesse=[fxx,fxy;fyx,fyy];
x=[-1.2,1];      %定义初始点

%%
N=200;    %总迭代次数
e=0.000001;
P=zeros(N,2);  %储存点的轨迹
OPT=zeros(N,2);  %储存最优值下降的轨迹
g=subs(Gradient,[x1 x2],[x(1) x(2)]);
step=1;
P(step,:)=x;
optim_fx=subs(f,[x1 x2],[x(1) x(2)]);
fprintf('Step[%d]: x=[ %f %f ] optim_fx=%f\n',step,x(1),x(2),double
    (optim_fx));
OPT(step,:)=optim_fx;
%%
while (norm(g)>e && step < N)   %当g的2-范数小于特定值时, 或迭代
    次数到达上限时, 停止迭代
    step=step+1;
    %计算目标函数点x(k)处一阶导数值
    g=subs(Gradient,[x1 x2],[x(1) x(2)]);
    %计算目标函数点x(k)处Hesse矩阵
    G=subs(Hesse,[x1 x2],[x(1) x(2)]);
    %计算目标函数点x(k)处搜索方向p
    %p=-G\g';
    p=-g;
    %点x(k)处的搜索步长

    ak=1;
   % ak=Alpha(p,g,G);
    xk=x+ak*double(p');
    %采用Armijo法则计算近似步长ak
    while(F(xk(1),xk(2)) > (F(x(1),x(2))+0.1*double(p'*g)*ak))
        ak=0.5*ak;
```

```
        xk=x+ak*double(p');
    end
    x=x+double(ak*p');
    %输出结果
    optim_fx=subs(f,[x1 x2],[x(1) x(2)]);
    fprintf('Step[%d]: x=[ %f %f ] optim_fx=%f\n',step,x(1),x(2),
        double(optim_fx));
    P(step,:)=x;
    OPT(step,:)=optim_fx;
    g=subs(Gradient,[x1 x2],[x(1) x(2)]);
end
%输出结果
optim_fx=subs(f,[x1 x2],[x(1) x(2)]);
fprintf('\n最速下降法,共迭代 %d 步\n结果: \n x=[ %d %d ] optim_fx
    =%f\n',step,x(1),x(2),double(optim_fx));
P(step+1:N,:)=[];   %删去P中的多余空间\
figure;
plot(P(:,1),P(:,2))
hold on;
xx =linspace(-1.5,1.5);
yy = linspace(-0.5,1.5);
[X,Y] = meshgrid(xx,yy);
Z=F(X,Y);
contour(X,Y,Z,'ShowText','on')
figure;
plot(OPT)
%%
% 最速下降法子程序, 计算精确步长ak
function a=Alpha(p,g,G)
a=-(p'*g)/(g'*G*g);
a=double(a);
end
```

```
%牛顿Armijo回溯法
%%
%用符号表达式定义目标函数
clc;
clear;
```

```
syms x1 x2;
X=[x1,x2];
f=100*(X(2)-X(1)^2)^2+(1-X(1))^2;

F=eval(['@(x1,x2)',vectorize(f)]);
fx=diff(f,x1); %求f对x1偏导数
fy=diff(f,x2); %求f对x2偏导数
fxx=diff(fx,x1); %求二阶偏导数 对x1再对x1
fxy=diff(fx,x2); %求二阶偏导数 对x1再对x2
fyx=diff(fy,x1); %求二阶偏导数 对x2再对x1
fyy=diff(fy,x2); %求二阶偏导数 对x2再对x2
Gradient=[fx;fy];  %计算梯度表达式
Hesse=[fxx,fxy;fyx,fyy];
x=[-1.2,1];      %定义初始点

%%
N=100;    %总迭代次数
e=0.000001;
P=zeros(N,2);  %储存点的轨迹
OPT=zeros(N,2);  %储存最优值下降的轨迹
g=subs(Gradient,[x1 x2],[x(1) x(2)]);
step=1;
P(step,:)=x;
optim_fx=subs(f,[x1 x2],[x(1) x(2)]);
fprintf('Step[%d]: x=[ %f %f ] optim_fx=%f\n',step,x(1),x(2),double(optim_fx));
OPT(step,:)=optim_fx;
%%

while (norm(g)>e && step < N)   %当g的2-范数小于特定值时，或迭代次数到达上限时，停止迭代
   step=step+1;
   %计算目标函数点x(k)处一阶导数值
   g=subs(Gradient,[x1 x2],[x(1) x(2)]);
   %计算目标函数点x(k)处Hesse矩阵
   G=subs(Hesse,[x1 x2],[x(1) x(2)]);
   %计算目标函数点x(k)处搜索方向p
   p=-inv(G)*g;
   %p=-g;
```

```
    %点x(k)处的搜索步长
    %ak=Alpha(p,g,G);
    %ak=1;
    ak=1;
    xk=x+ak*double(p');
    %采用Armijo法则计算近似步长ak
    while(F(xk(1),xk(2)) > (F(x(1),x(2))+0.01*double(p'*g)*ak))
        ak=0.5*ak;
        xk=x+ak*double(p');
    end
    x=x+double(ak*p');
    %输出结果
    optim_fx=subs(f,[x1 x2],[x(1) x(2)]);
    fprintf('Step[%d]: x=[ %f %f ] optim_fx=%f\n',step,x(1),x(2),
        double(optim_fx));
    P(step,:)=x;
    OPT(step,:)=optim_fx;
    g=subs(Gradient,[x1 x2],[x(1) x(2)]);
end
%输出结果
optim_fx=subs(f,[x1 x2],[x(1) x(2)]);
fprintf('\n牛顿Armijo回溯法,,共迭代 %d 步\n结果：\n x=[ %d %d ]
    optim_fx=%f\n',step,x(1),x(2),double(optim_fx));
P(step+1:N,:)=[];   %删去P中的多余空间
figure;
plot(P(:,1),P(:,2),'-o')
hold on;
xx =linspace(-1.5,1.5);
yy = linspace(-0.5,1.5);
[X,Y] = meshgrid(xx,yy);
Z=F(X,Y);
contour(X,Y,Z,'ShowText','on')
figure;
plot(OPT)
%%
% 计算精确步长ak
function a=Alpha(p,g,G)
    a=-(p'*g)/(p'*G*p);
end
```

```
%采用Armijo法则计算近似步长ak
function a=Armijo(x,p,g)
    a=1;
    xk=x+a*double(p);
    while(F(xk(1),xk(2)) > F(x(1),x(2))+0.01*doubel(p'*g)*a)
        a=0.9*a;
        xk=xk+a*double(p);
    end
end
```

```
%Conj_Grad 共轭梯度法
clc;
clear;
N=4;
G=zeros(N,N);
b=[-1,0,2,5^(0.5)]';
x0=zeros(N,1);
for i= 1:N
     G(i,i) = 2;
end

for i= 1:N-1
    G(i+1,i)=-1;
    G(i,i+1)=-1;
end
    x = x0;x'
    g = G*x+b;
    p = -g;
    k = 0;
    while 1
        if norm(g, 2)<1e-6
            break
        end
        k = k + 1;

        d=G*p;
        a=(g'*g)/(p'*d);
```

```

     x = x+a*p;x'
     t=g+a*d;
    beta=(t'*t)/(g'*g);
    g=t;
    p=-g+beta*p;

  end
```