```
begin
    using Symbolics   , Plots   , LaTeXStrings   ,Latexify   ,LinearAlgebra
end
```

使用牛顿法求解$Powell$ 函数的最小值

$$(x_2 - 2x_3)^4 + (x_1 + 10x_2)^2 + 10(x_1 - x_4)^4 + 5(x_3 - x_4)^2$$

```
begin
    @variables x₁,x₂,x₃,x₄,powell(x₁,x₂,x₃,x₄)
    powell=(x₁+10x₂)^2+5(x₃-x₄)^2+(x₂-2x₃)^4+10(x₁-x₄)^4 #for human

end
```

org_f (generic function with 1 method)

```
org_f(u)=(u[1]+10u[2])^2+5(u[3]-u[4])^2+(u[2]-2u[3])^4+10(u[1]-u[4])^4  # for computer
```

xk (generic function with 1 method)

```
begin
    vars=[x₁,x₂,x₃,x₄]
    xk(x)=Pair.(vars,x)|>Dict  #用于生成带入表达式的字典 substitute(ex, Dict([x => z, sin(z) => z^2]))
end
```

grad (generic function with 1 method)

```
grad(x)=substitute(∇f,xk(x))
```

get_val (generic function with 1 method)

```
get_val(x)=substitute(powell,xk(x))
```

准备工作

求解函数梯度和黑塞矩阵

```
md"""
## 准备工作
求解函数梯度和黑塞矩阵
"""
```

∇f =

$$[2x_1 + 20x_2 + 40(x_1 - x_4)^3,\ 20x_1 + 200x_2 + 4(x_2 - 2x_3)^3,\ 10x_3 - 10x_4 - 8(x_2 - 2x_3)^3,\ -10x_3 +$$

```
∇f=Symbolics.gradient(powell,vars)  # 函数梯度
```

hess =

$$\begin{bmatrix} 2 + 120(x_1 - x_4)^2 & 20 & 0 & -120(x_1 - x_4)^2 \\ 20 & 200 + 12(x_2 - 2x_3)^2 & -24(x_2 - 2x_3)^2 & 0 \\ 0 & -24(x_2 - 2x_3)^2 & 10 + 48(x_2 - 2x_3)^2 & -10 \\ -120(x_1 - x_4)^2 & 0 & -10 & 10 + 120(x_1 - x_4)^2 \end{bmatrix}$$

```
hess=Symbolics.hessian(powell,vars)   #黑塞矩阵
```

H (generic function with 1 method)

```
H(x)=xk(x)|>d->substitute(hess,d) #遍历方法
```

1.第一次迭代

```
md"""
##  1.第一次迭代
"""
```

x₀ = [3, -1, 0, 1]

```
x₀=[3,-1,0,1]   # 从这点开始
```

fx1 =

$$\begin{bmatrix} 482 & 20 & 0 & -480 \\ 20 & 212 & -24 & 0 \\ 0 & -24 & 58 & -10 \\ -480 & 0 & -10 & 490 \end{bmatrix}$$

```
fx1=H(x₀)
```

inv_fx1 =

$$\begin{bmatrix} 0.1126228269085413 & -0.008881330309901745 & 0.015400604686318984 & 0.110 \\ -0.008881330309901746 & 0.005650037792894936 & 0.0008408919123204827 & -0.008 \\ 0.015400604686318984 & 0.00084089191232048 & 0.020261715797430083 & 0.0154 \\ 0.1106386999244143 & -0.008682917611489047 & 0.015499811035525333 & 0.1107 \end{bmatrix}$$

```
inv_fx1=inv(fx1)
```

g1 = $[306, -144, -2, -310]$

```
g1=grad(x₀)
```

desc1 =

$[1.4126984126984112, -0.8412698412698405, -0.25396825396825484, 0.7460317460317469]$

```
desc1=inv_fx1*g1
```

x1 =

$[1.5873015873015888, -0.1587301587301595, 0.25396825396825484, 0.25396825396825307]$

```
x1=x₀-desc1
```

$$31.8024691358027$$

```
org_f(x1)  #第一次迭代点的函数值
```

第二次迭代

g2 = $[94.8148148148153, -1.185185185185322, 2.3703703703704146, -94.81481481481534]$

```
g2=substitute(∇f,xk(x1))
```

fx2 =

$$\begin{bmatrix} 215.33333333333408 & 20 & 0 & -213.3333333 \\ 20 & 205.33333333333337 & -10.666666666666748 & 0 \\ 0 & -10.666666666666748 & 31.333333333333496 & -10 \\ -213.33333333333408 & 0 & -10 & 223.3333333 \end{bmatrix}$$

```
fx2=H(x1)
```

inv_fx2 =

$$\begin{bmatrix} 0.13860544217687054 & -0.011479591836734667 & 0.03890306122448949 & 0.1 \\ -0.011479591836734667 & 0.005909863945578229 & -0.0015093537414965674 & -0.0 \\ 0.0389030612244895 & -0.0015093537414965687 & 0.04388180272108 52 & 0.03 \\ 0.13414115646258482 & -0.011033163265306099 & 0.03912627551020378 & 0.1 \end{bmatrix}$$

```
inv_fx2=inv(fx2)
```

desc2 =

$[0.5291005291005337, -0.05291005291005324, 0.0846560846560851, 0.08465608465608199]$

```
desc2=inv_fx2*g2
```

x2 =

$[1.058201058201055, -0.10582010582010626, 0.16931216931216975, 0.16931216931217108]$

```
x2=x1-desc2
```

$$6.281969212010226$$

```
org_f(x2)  #第二次迭代点的函数值
```

3.第三次迭代

g3 =

$[28.093278463648353, -0.3511659807957628, 0.7023319615912138, -28.093278463648353]$

```
g3=substitute(∇f,xk(x2))
```

fx3 =

$$\begin{bmatrix} 96.81481481481376 & 20 & 0 & -94.81481481481 \\ 20 & 202.37037037037038 & -4.740740740740769 & 0 \\ 0 & -4.740740740740769 & 19.481481481481538 & -10 \\ -94.81481481481376 & 0 & -10 & 104.81481481481 \end{bmatrix}$$

```
fx3=H(x2)
```

inv_fx3 =

$$\begin{bmatrix} 0.19706632653061193 & -0.017325680272108807 & 0.09178358843537365 & 0.1870 \\ -0.01732568027210881 & 0.006494472789115642 & -0.0067974064625849805 & -0.016 \\ 0.09178358843537365 & -0.006797406462584981 & 0.09704772534013552 & 0.0922 \\ 0.18702168367346897 & -0.01632121598639451 & 0.09228582057823079 & 0.187 \end{bmatrix}$$

```
inv_fx3=inv(fx3)
```

desc3 =

$[0.3527336860670198, -0.03527336860670277, 0.056437389770723545, 0.05643738977072488]$

```
desc3=inv_fx3*g3
```

x3 =

$[0.7054673721340352, -0.07054673721340349, 0.1128747795414462, 0.1128747795414462]$

```
x3=x2-desc3
```

$$1.2408828073106595$$

```
org_f(x3) #第三次迭代点的函数值
```

4. 第四次迭代

将整个流程整合起来

```
md"""
## 4. 第四次迭代
将整个流程整合起来
"""
```

$$0.245112653295932$$

```
begin
    g4=substitute(∇f,xk(x3))
    fx4=H(x3)
    inv_fx4=inv(fx4)
    desc4=inv_fx4*g4
    x4=x3-desc4
    org_f(x4)
end
```

minf (generic function with 1 method)

```
function minf(x)
    # 上面函数包装一下
    g=substitute(∇f,xk(x))
    fx=H(x)
    inv_fx=inv(fx)
    desc=inv_fx*g
    return new_x=x-desc

end
```

第五次迭代

$$0.04841731423129499$$

```
begin
    #使用包装方法
    minf(x4)|>org_f
```

```
end
```

定义牛顿方法

```
md"""
## 定义牛顿方法
"""
```

newtons_method (generic function with 2 methods)

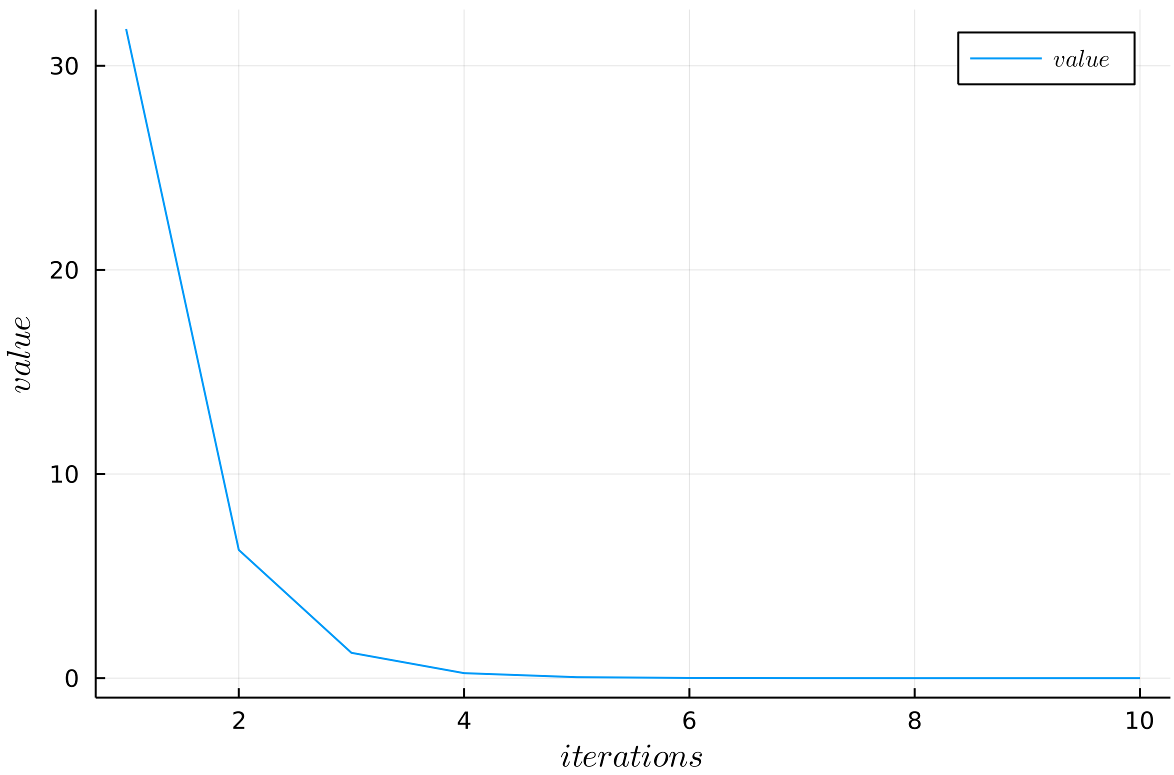
```
function newtons_method(∇f, H, x, k_max, ε=0.005)
    k, Δ = 1, fill(Inf, length(x))

    while norm(Δ) > ε && k ≤ k_max
        Δ = H(x) \ ∇f(x)
        x -= Δ
        k += 1
    end
    return x
end
```

get_value (generic function with 1 method)

```
get_value(k)=newtons_method(grad,H,x₀,k)|>get_val|>d->Symbolics.value(d)|>d->round(d,digits=3)
```

arr = [31.802, 6.282, 1.241, 0.245, 0.048, 0.01, 0.002, 0.0, 0.0, 0.0]

```
arr=[get_value(k) for k in 1:10]
```




```
begin
    span=1:1:length(arr)
    plot(span,arr,label=L"value",xlabel=L"iterations",ylabel=L"value")
end
```

$$Powell(X) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$