6.5. Olinjära minsta kvadratmetoden

Vi skall börja med att i korthet beskriva den statistiska princip, som ligger till grund för den tidigare behandlade minsta kvadratmetoden.

Antag, att vi önskar bestämma parametervektorn $x = \{x_i, i = 1, 2, \ldots, n\}$ i *modellekvationen*

$$y(t) = f(t, x)$$

utgående från observationerna

$$y_i = y(t_i) + e_i \quad (i = 1, 2, \ldots, m) \qquad (m > n),$$

där *mätfelen* e_i antas vara *oberoende*, *normalfördelade* stokastiska variabler med medelvärdet 0 och standardavvikelsen σ_i.

Detta innebär, att sannolikheten för mätvärdet y_i har normalfördelningen

$$P(y_i) = \frac{1}{\sigma_i\sqrt{2\pi}}\exp\left\{-\tfrac{1}{2}\frac{[y_i - f(t_i, x)]^2}{\sigma_i^2}\right\}$$

(Gauss' "klockkurva"). Om mätvärdena är oberoende, så kommer sannolikheten för att erhålla en bestämd serie observationsdata $y_i, i = 1, 2, \ldots, m$ att vara produkten av de enskilda sannolikheterna

$$P_y = P(y_1)P(y_2)\ldots.$$

P_y är givetvis i allmänhet en funktion av parametrarna x_i $(i = 1, 2, \ldots, n)$. Genom att införa beteckningen

$$\chi^2(x) = \sum_{i=1}^{m} \frac{1}{\sigma_i^2}[y_i - f(t, x_i)]^2,$$

så finner man, att den *totala* sannolikheten kan uttryckas

$$P_y(x) = \frac{1}{\sigma_1\sigma_2\ldots\sigma_m(2\pi)^{m/2}} e^{-\chi^2/2}.$$

Enligt den s.k. maximeringsprincipen (principle of maximum likelihood) bör parametrarna väljas så, att P_y blir så stor som möjligt. Detta leder till villkoret $\chi^2 = minimum$, vilket är ekvivalent med minsta kvadratmetoden[1].

[1] C.F. Gauss: *Theoria Motus...*, Art. 179

Observera, att denna härledning bygger på, att mätvärdena är normalfördelade, vilket inte alltid behöver vara fallet.

Som ett uttryck för anpassningens godhet använder man ofta testfunktionen

$$S = \sum_{k=1}^{m} w_k e_k^2,$$

istället för χ^2. w_k betecknar mätningarnas vikter ($w_k \propto \frac{1}{\sigma_k^2}$).

I matrisform kan denna funktion uttryckas

$$S = e^T W e,$$

där W är en diagonal viktsmatris ($W_{ij} = w_i \delta_{ij}$), och e en felvektor ($e_i = y_i - f(t_i, x)$).

Om ifrågavarande minsta kvadratproblem är *linjärt*, så kan felvektorn uttryckas i formen

$$e = y - Ax,$$

där A är en $m \times n$–matris med *konstanta* koefficienter (observera, att $m > n$).

Villkoret för att S skall ha en stationär punkt är $\nabla S = 0$. Genom att tillämpa detta villkor på testfunktionen S fås

$$0 = \frac{\partial S}{\partial x_k} = -2 \sum_{i=1}^{m} w_i (y_i - \sum_{j=1}^{n} A_{ij} x_j) A_{ik},$$

som leder till ekvationssystemet

$$\sum_{i=1}^{m} \sum_{j=1}^{n} A_{ik} w_i A_{ij} x_j = \sum_{i=1}^{m} A_{ik} w_i y_i \quad (k = 1, 2, \ldots, n)$$

(Gauss' normalekvationer).

Dessa ekvationer kan också framställas i matrisform

$$A^T W A x = A^T W y.$$

Om matrisen $A^T W A$ (som är en symmetrisk $n \times n$–matris) inte är singulär, så är

$$x_0 = (A^T W A)^{-1} A^T W y$$

en stationär punkt av S.

Genom direkt uträkning (om vi deriverar en gång till) finner vi dessutom, att Hesses matris H är

$$H = 2A^T W A,$$

varav följer, att x_0 är ett minimum, ifall matrisen $A^T W A$ är positivt definit.

Ifall f beror *olinjärt* av x, så kan minsta kvadratproblemet (i princip) lösas genom successiva approximationer, där man vid varje steg löser ett *linjärt* minsta kvadratproblem.

Låt oss nu anta, att y' är en vektor, som beräknats ur en serie parametervärden $x'_j,\ j = 1, 2, \ldots, n$, så att

$$y'_i = f(t_i, x'),\ i = 1, 2, \ldots, m$$

Om vi ger ett litet tillskott Δx till parametervektorn x', så att den nya parametervektorn blir $x'' = x' + \Delta x$, så finner vi den motsvarande y–vektorns element ur ekvationen

$$y'' = f(t, x'') = f(t, x' + \Delta x).$$

Om $f(t, x'')$ utvecklas i Taylors serie omkring x', så fås

$$y'' = f(t, x'') = y' + g^T \Delta x + \tfrac{1}{2}(\Delta x)^T H \Delta x + \ldots.$$

Om endast termerna t.o.m. första ordningen medtas, fås ett linjärt ekvationssystem:

$$\begin{cases} y_i'' - y_i' = g^T(t_i, x')\Delta x \qquad (i = 1, 2, \ldots, m) \\ g^T(t_i, x') = \left[\dfrac{\partial}{\partial x_1}, \dfrac{\partial}{\partial x_2}, \ldots, \dfrac{\partial}{\partial x_n}\right] f(t_i, x)|_{x=x'} \end{cases}$$

Genom att definiera en matris J (Jakobianen av f):

$$J_{ij} = g_j^T(t_i, x),$$

så kan ovanstående ekvation uttryckas i matrisform

$$y'' - y' = J\Delta x.$$

Låt y_0 beteckna en vektor, vars element utgörs av observationsdata. Då kan felvektorn e skrivas

$$e = y'' - y_0 = y' - y_0 + y'' - y' = y' - y_0 + J\Delta x.$$

Problemet är sålunda *linjäriserat*, och vi kan i analogi med det linjära minsta kvadratproblemets lösning finna korrektionsvektorn Δx genom att lösa ekvationssystemet

$$J^T W J \Delta x = -J^T W y$$

under antagandet, att $J^T W J$ är positivt definit. Observera här, att matrisen J har variabla element!

Om den linjära approximationen för y'' skulle vara exakt, så skulle man finna lösningen till det olinjära minsta kvadratproblemet helt enkelt genom att addera x' och Δx. I praktiken gäller detta *inte*, och vi måste därför antingen beräkna flera successiva approximationer till x (dvs iterera), eller också inkludera flera termer i Taylor–serien för f. Den förstnämnda metoden är mer praktisk, eftersom man inte behöver beräkna högre derivator.

Om matrisen $J^T W J$ är positivt definit, så kan man beräkna parameterkorrektionerna ur ekvationen

$$\Delta x \approx -[J^T W J]^{-1} J^T W y.$$

Om matrisen inte är positivt definit, eller om parameterändringarna tenderar att bli för stora, uppstår det problem. I det senare fallet kan man försöka minska parameterändringarna genom att multiplicera dem med en faktor α, $0 < \alpha < 1$. Den nya parametervektorn blir då

$$x^{(k+1)} = x^{(k)} + \alpha \Delta x^{(k)}.$$

Denna metod kallas för en dämpad minsta kvadratmetod.

Mera allmänt skulle man kunna uttrycka den nya parametervektorn i formen

$$x^{(k+1)} = x^{(k)} + \alpha v.$$

Parametern α kan man beräkna genom att minimera den endimensionella funktion $\bar{S}$, som man kan definiera genom följande identitet:

$$\bar{S}(\alpha) \equiv S(x^{(k)} + \alpha v) = S(x^{(k+1)}).$$

Om vi serieutvecklar $\bar{S}$ efter potenser av α med $\alpha = 0$ som utvecklingscentrum, vilket svarar mot $x = x^{(k)}$, får vi

$$\bar{S} = S(x^{(k)}) + \alpha \left.\frac{d\bar{S}}{d\alpha}\right|_{\alpha=0} + \frac{\alpha^2}{2!} \left.\frac{d^2\bar{S}}{d\alpha^2}\right|_{\alpha=0} + \dots.$$

Villkoret för en stationär punkt blir då

$$\frac{d\bar{S}}{d\alpha} = \left.\frac{d\bar{S}}{d\alpha}\right|_{\alpha=0} + \alpha \left.\frac{d^2\bar{S}}{d\alpha^2}\right|_{\alpha=0} + \dots = 0.$$

Om vi utnyttjar definitionen för $\bar{S}$, så fås följande uttryck för de två första derivatorna:

$$\frac{d\bar{S}}{d\alpha} = \sum_{i=1}^{n} \frac{\partial S}{\partial x_i} v_i \equiv v^T g,$$

$$\frac{d^2\bar{S}}{d\alpha^2} = \sum_{i,j=1}^{n} \frac{\partial^2 S}{\partial x_i \partial x_j} v_i v_j \equiv v^T G v,$$

där g är gradientvektorn av S och G betecknar Hesses matris av S.

För att approximativt finna ett värde av α kan man avbryta serieutvecklingen av $\frac{d\bar{S}}{d\alpha}$ efter andra termen, och vi finner då

$$\alpha = -\frac{\left.\frac{d\bar{S}}{d\alpha}\right|_{\alpha=0}}{\left.\frac{d^2\bar{S}}{d\alpha^2}\right|_{\alpha=0}} = -\frac{v^T g}{v^T G v},$$

där g och G beräknas i punkten $x = x^{(k)}$.

För att kunna använda detta uttryck måste vi göra vissa antaganden. Om v har formen

$$v = -Hg,$$

så får vi efter substitution

$$\alpha = \frac{g^T H^T g}{g^T H^T G H g}.$$

Vilket värde hessianen H har beror av den använda metoden. För "steepest descent" har vi $H = I$, och därmed $\alpha = g^T g / g^T G g$; och för Newton–Raphson gäller $H = G^{-1}$, varav följer att $\alpha = 1$.

För att komma underfund med den *bästa riktningen* av v kan vi substituera $\left.\frac{d^2 \bar{S}}{d\alpha^2}\right|_{\alpha=0} = -\frac{1}{\alpha} \left.\frac{d\bar{S}}{d\alpha}\right|_{\alpha=0}$ (som följer av uttrycket för α) i serieutvecklingen av $\bar{S}$, varvid vi finner

$$\bar{S} = S(x^{(k)}) + \frac{\alpha}{2} \left.\frac{d\bar{S}}{d\alpha}\right|_{\alpha=0}.$$

Emedan $-\alpha \left.\frac{d\bar{S}}{d\alpha}\right|_{\alpha=0} = \frac{v^T g}{v^T G v} \cdot v^T g = \frac{(v^T g)^2}{v^T G v}$, så får vi en stationär punkt ur villkoret

$$\frac{\partial}{\partial v} \left[\frac{(v^T g)^2}{v^T G v} \right] = 0.$$

Genom att utföra deriveringen fås

$$2\frac{v^T g}{v^T G v} g - \left[\frac{(v^T g)^2}{(v^T G v)^2}\right] 2Gv = 0,$$

dvs $g = v^T g/(v^T G v) Gv = -\alpha Gv = -G(x^{(k+1)} - x^{(k)}) = -G\Delta x$. Detta är definitionen för Newton-Raphsons metod, som alltså (i teorin åtminstone) är den "bästa" metoden, med avseende på den noggrannhet vi utfört serieutvecklingen.

I praktiken är däremot Newton-Raphsons metod inte alltid den bästa möjliga. Som redan nämnts, så kan parameterändringarna bli alltför stora (man s.a.s. "skjuter över målet"). *Levenberg* föreslog år 1944[2] en metod att överkomma denna svårighet. Han föreslog, att man skulle modifiera S genom att tillägga en positivt definit kvadratisk form av parameterändringarna, så att man istället för S skulle minimera funktionen

$$\tilde{S} = wS + \tfrac{1}{2}\sum_{i=1}^{n} a_i \Delta x_i^2 \quad (w, a_i > 0)$$

[2] K. Levenberg: *A method for the solution of certain non–linear problems in least squares*, Q. appl. Math. **2**, 164-168 (1944)

Som vi ser, blir Hesses matris av $\tilde{S}$ $w(G + A)$, där A är en diagonal matris med elementen $A_{ij} = w^{-1}a_i\delta_{ij}$. Newton–Raphsons ekvation för denna modifierade funktion blir alltså

$$(G + A)(x^{(k+1)} - x^{(k)}) \equiv (G + A)\Delta x = -g.$$

Parametern w kommer huvudsakligen att bestämma dämpningsgraden, medan koefficienterna a_i egentligen bara fungerar som skalningsfaktorer. För att bestämma ett värde för w, utvecklade Levenberg $\tilde{S}$ efter potenser av w kring $w = 0$, och fann då approximativt (i en omgivning av $x^{(k)}$)

$$\tilde{S} = \tilde{S}_0 + w \left.\frac{d\tilde{S}}{dw}\right|_0 .$$

Om man antar att $\tilde{S} \ll \tilde{S}_0$ gäller följaktligen

$$w \approx -\frac{\tilde{S}_0}{\left.\frac{d\tilde{S}}{dw}\right|_0}.$$

Då $w \approx 0$, så kan ekvationssystemet $(G + A)\Delta x = -g$ approximativt lösas på följande sätt:

$$w^{-1} a_i \Delta x_i \approx -g_i,$$

dvs $\Delta x_i \approx -g_i a_i^{-1} w$. Således gäller

$$\left.\frac{d\Delta x_i}{dw}\right|_{w=0} = -g_i a_i^{-1}.$$

Med hjälp av kedjeregeln finner vi då

$$\frac{d\tilde{S}}{dw} = \sum_{i=1}^{n} \frac{\partial \tilde{S}}{\partial \Delta x_i} \frac{d\Delta x_i}{dw} = -\sum_{i=1}^{n} g_i^2 a_i^{-1},$$

varav följer att Levenbergs approximation för w är

$$w = -\tilde{S}_0 / \left.\frac{d\tilde{S}}{dw}\right|_0 = \frac{\tilde{S}_0}{g^T v},$$

där vektorn v ges av formeln

$$v^T = [g_1 a_1^{-1}, \ldots, g_n a_n^{-1}].$$

På grund av att gradientvektorn är "stor" i början av iterationerna, och avtar med tiden, kommer w att ha sitt minsta värde i början, och småningom växa mot oändligheten. Gränsfallet svarar mot en vanlig Newton–Raphson iteration, vilket vi lätt inser.

Det enklaste sättet att skala variablerna är att välja alla $a_i = 1$. Då $w \approx 0$ kommer i detta fall iterationsriktningen att nära nog sammanfalla med "steepest descent"–riktningen. Parameterändringarna kan då beräknas ur ekvationen

$$(G + \lambda I)\Delta x = -g,$$

och parametern λ $(= 1/w)$ blir i detta fall $(v = g)$

$$\lambda = \frac{g^T g}{\tilde{S}_0}.$$

Parametern λ, som ingår i ekvationen ovan kallas ibland Marquardts parameter, efter D.W. Marquardt, som år 1963 föreslog en liknande ekvation[3], fastän han beräknade sina värden av λ enligt en empirisk regel. Marquardt rekommenderade, att man skulle skala variablerna så, att diagonalelementen i G blir lika med 1.

[3] Marquardt, D.W. : *An algorithm for least-squares estimation of non-linear parameters*, J. Soc. ind. appl. Math. **11**, 431-441 (1963)

Samma effekt fås med Levenbergs metod genom att man väljer $a_i = G_{ii}\,(i = 1, 2, \ldots, n)$, dvs man ersätter diagonalelementen i G med $G_{ii}(1 + \lambda)$, så att ekvationssystemet blir

$$(G + \lambda\,\mathrm{diag}(G))\Delta x = -g,$$

där diag(G) betecknar diagonalelementen av G. Levenbergs metod att uppskatta parametern λ leder i detta fall till

$$\lambda = \frac{g^T v}{\tilde{S}_0}, \quad v^T = [g_1 G_{11}^{-1}, \ldots, g_n G_{nn}^{-1}].$$

Nedan visas ett exempel på en MATLAB–rutin, som baserar sig på Marquardts metod:

```
function [p,fp,iter,conv] = marq(funk,dfunk,p0,data,iw,tol,m,n,niter)
% Marquardt method, minimization or least squares (equal weights!)
% initialize parameters:
conv = 0; iter = 0; p = p0; restore = 0;
while iter <= niter & conv == 0
   yp=feval(funk,p,m,n); yp=yp';
   if iw == 0
      yp1 = yp;
   else
      yp1 = yp - data;
   end
   fp = yp1'*yp1;
```

```
   dyp=feval(dfunk,p,m,n);
% test for convergence:
   if iter > 0
      disp(sprintf('iter=%4.0f fp=%15.6f lamda=%15.6f dpn=
      %15.6f',iter,fp,lamda,dpn));
      dfp = fp - fp1;
      if dpn > tol*pn & gn > tol
         if dfp > 0
% fp increases: restore old parameters and increase lamda
            p=p0; lamda=lamda*2; restore = 1;
         else
% fp decreases: update parameters and decrease lamda
            fp1 = fp;  p0 = p;  lamda = lamda/3; restore = 0;
         end
      else
         conv = 1;  % process has converged!
      end
   end
   if restore == 0
% compute g, A:
      g=-dyp*yp1; gn = sqrt(g'*g) ; A = dyp*dyp';
      d=sqrt(diag(A)); dd=d*d'; g1=g./d ;
   end
   if iter == 0
```

```
% initialize lamda:
      lamda = g1'*g1; lamda = lamda/fp ; fp1 = fp;
      disp(sprintf('iter=%4.0f fp=%15.6f lamda=%15.6f',iter,fp,lamda));
   end
% compute parameter changes:
   U=A./dd+lamda*eye(m);
   dp=U\g1 ; dp = dp./d ; dpn =sqrt(dp'*dp); p=p+dp ; pn=sqrt(p'*p);
% next iteration:
   iter = iter+1;
end
```

Som vi ser, kommer värdet av Marquardts parameter λ att divideras med 3, ifall godhetsfunktionen avtar, men multipliceras med 2, ifall den växer. I det senare fallet kommer λ att tillämpas på den senast beräknade matrisen A (som sparats). Som exempel har vi tillämpat metoden på Rosenbrocks bananfunktion, som förekom i en räkneövning.

```
p1 = [-1.2 ; 1.0]
>> [p,fp,iter,conv] = marq('rosenb','drosen',p1,data,0,1e-4,2,2,50)
iter=   0 fp=      24.200000 lamda=       1.632236
iter=   1 fp=       8.536768 lamda=       1.632236 dpn=       0.130521
iter=   2 fp=       4.612457 lamda=       0.544079 dpn=       0.080935
iter=   3 fp=       4.362478 lamda=       0.181360 dpn=       0.021615
...                 ...                   ...                 ...
iter=  33 fp=       0.001927 lamda=       0.000414 dpn=       0.144111
iter=  34 fp=       0.000012 lamda=       0.000138 dpn=       0.040506
iter=  35 fp=       0.000000 lamda=       0.000046 dpn=       0.004329
iter=  36 fp=       0.000000 lamda=       0.000015 dpn=       0.000157
iter=  37 fp=       0.000000 lamda=       0.000005 dpn=       0.000002
p = [0.99999999998573 ; 0.99999999997143]
fp =   1.222360946254525e-017
```