7 Constrained fits

Benno List

Exercise 7.1: Particle decay

The explicit penalty function $f = \chi^2$ is given by

$$f\,(E_1^*, E_2^*) = \frac{(\hat{E}_1 - E_1^{\mathrm{meas}})^2}{(1-\rho^2)\sigma_1^2} - \frac{2\rho(\hat{E}_1 - E_1^{\mathrm{meas}})(\hat{E}_2 - E_2^{\mathrm{meas}})}{(1-\rho^2)\sigma_1\sigma_2} + \frac{(\hat{E}_2 - E_2^{\mathrm{meas}})^2}{(1-\rho^2)\sigma_2^2}\,.$$

The two equations given by $\mathbf{0} = \boldsymbol{\nabla} f\,(E_1^*, E_2^*) + \lambda^* \boldsymbol{\nabla} c\,(E_1^*, E_2^*)$ plus the third equation $0 = \partial/\partial\lambda^*\ (f\,(E_1^*, E_2^*) + \lambda^* c\,(E_1^*, E_2^*))$ are given by

$$0 = \frac{\partial(f + \lambda^* c)}{\partial \hat{E}_1} = \frac{2(\hat{E}_1 - E_1^{\mathrm{meas}})}{(1-\rho^2)\sigma_1^2} - \frac{2\rho(\hat{E}_2 - E_2^{\mathrm{meas}})}{(1-\rho^2)\sigma_1\sigma_2} + 2\lambda^* \hat{E}_2(1-\cos\psi)\,, \tag{7.1}$$

$$0 = \frac{\partial(f + \lambda^* c)}{\partial \hat{E}_2} = -\frac{2\rho(\hat{E}_1 - E_1^{\mathrm{meas}})}{(1-\rho^2)\sigma_1\sigma_2} + \frac{2(\hat{E}_2 - E_2^{\mathrm{meas}})}{(1-\rho^2)\sigma_2^2} + 2\lambda^* \hat{E}_1(1-\cos\psi), \tag{7.2}$$

$$0 = \frac{\partial(f + \lambda^* c)}{\partial \lambda^*} = 2\hat{E}_1\hat{E}_2(1-\cos\psi) - m_{\pi^0}^2\,. \tag{7.3}$$

a) The set of equations for the one step of the Newton–Raphson method is given by equation (7.62) in the book:

$$\begin{pmatrix} -\mathbf{g}_{\boldsymbol{\theta}}^{(\nu)} \\ -\mathbf{c}^{(\nu)} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{\theta\theta}^{(\nu)} & \mathbf{A}_{\theta}^{(\nu)} \\ \mathbf{A}_{\theta}^{T(\nu)} & 0 \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{\theta}^{(\nu+1)} \\ \Delta\boldsymbol{\lambda}^{(\nu+1)} \end{pmatrix}, \tag{7.4}$$

with

$$-\mathbf{g}_{\boldsymbol{\theta}}{}^{(\nu)} = \begin{pmatrix} -\frac{2(\hat{E}_1^{(\nu)}-E_1^{\text{meas}})}{(1-\rho^2)\sigma_1^2} + \frac{2\rho(\hat{E}_2^{(\nu)}-E_2^{\text{meas}})}{(1-\rho^2)\sigma_1\sigma_2} - 2\lambda^{*(\nu)}\hat{E}_2^{(\nu)}(1-\cos\psi) \\ \frac{2\rho(\hat{E}_1^{(\nu)}-E_1^{\text{meas}})}{(1-\rho^2)\sigma_1\sigma_2} - \frac{2(\hat{E}_2^{(\nu)}-E_2^{\text{meas}})}{(1-\rho^2)\sigma_2^2} - 2\lambda^{*(\nu)}\hat{E}_1^{(\nu)}(1-\cos\psi) \end{pmatrix} \tag{7,5}$$

$$-\mathbf{c}^{(\nu)} = \begin{pmatrix} -2\hat{E}_1^{(\nu)}\hat{E}_2^{(\nu)}(1-\cos\psi) + m_{\pi^0}^2 \end{pmatrix}, \tag{7.6}$$

$$\mathbf{L}_{\theta\theta}^{(\nu)} = \begin{pmatrix} \frac{2}{(1-\rho^2)\sigma_1^2} & -\frac{2\rho}{(1-\rho^2)\sigma_1\sigma_2} + 2\lambda^{*(\nu)}(1-\cos\psi) \\ \text{symm.} & \frac{2}{(1-\rho^2)\sigma_2^2} \end{pmatrix}, \tag{7.7}$$

$$\mathbf{A}_{\theta}^{(\nu)} = \begin{pmatrix} 2\hat{E}_2^{(\nu)}(1-\cos\psi) \\ 2\hat{E}_1^{(\nu)}(1-\cos\psi) \end{pmatrix}, \tag{7.8}$$

$$\Delta\boldsymbol{\theta}^{(\nu+1)} = \begin{pmatrix} \hat{E}_1^{(\nu+1)} - \hat{E}_1^{(\nu)} \\ \hat{E}_2^{(\nu+1)} - \hat{E}_2^{(\nu)} \end{pmatrix}, \tag{7.9}$$

$$\Delta\boldsymbol{\lambda}^{(\nu+1)} = \begin{pmatrix} \lambda^{*(\nu+1)} - \lambda^{*(\nu)} \end{pmatrix}. \tag{7.10}$$

b) The solution is shown in figure 7.1.

```
// Solution to Exercise 7.1b of
// B. List: "Constrained Fits", chapter 7 of
// O. Behnke, K. Kroeninger, G. Schott and T. Schoerner-Sadenius:
// "Data Analysis in High Energy Physics", Wiley-VCH, 2013.

// This C++ routine uses RooT classes, see http://root.cern.ch/

// sample interactive root session:
// root[1] .L exercise_7_1b.C
// root[2] double e1=5, e2=3, l=0;
// root[3] newtonit (1, e1, e2, l, 6, 7, 1, 1.5, -0.5, 1.570796326794897, sqrt(50.0));
// root[4] cout << "e1=" << e1 << ", e2=" << e2 << ", l=" << l << endl;
// e1=6.22807, e2=4.26316, l=0.304094

int newtonit (int nit, double& e1, double& e2, double& l,
              double e1m, double e2m, double s1, double s2, double r, double psi, double m) {
  double rfact = 1./(1-r*r);
  double pfact = 1-cos(psi);
  int n = 0;
  while (n++ < nit) {
    TVectorD vec(3);
    vec(0) =  -2*(e1-e1m)*rfact/(s1*s1) + 2*r*(e2-e2m)*rfact/(s1*s2) - 2*l*e2*pfact;
    vec(1) =  -2*(e2-e2m)*rfact/(s2*s2) + 2*r*(e1-e1m)*rfact/(s1*s2) - 2*l*e1*pfact;
    vec(2) =                                                          - 2*e1*e2*pfact + m*m;
    TMatrixDSym mat(3);
    mat(0,0) =             2*rfact/(s1*s1);
    mat(0,1) = mat(1,0) = -2*r*rfact/(s1*s2) + 2*l*pfact;
    mat(1,1) =             2*rfact/(s2*s2);
    mat(0,2) = mat(2,0) = 2*e2*pfact;
    mat(1,2) = mat(2,1) = 2*e1*pfact;
    mat(2,2) =             0;
    bool ok;
    TDecompLU lu(3);
    lu.SetMatrix(mat);
    ok = lu.Decompose();
    if (!ok) return 2;
    ok = lu.Solve(vec);
    if (!ok) return 1;
    e1 += vec[0];
    e2 += vec[1];
    l  += vec[2];
  }
  return 0;
}
```

Figure 7.1 Solution to exercise 7.1b).

c) The solution is shown in figures 7.2 and 7.3.

```
// Solution to Exercise 7.1c of
// B. List: "Constrained Fits", chapter 7 of
// O. Behnke, K. Kroeninger, G. Schott and T. Schoerner-Sadenius:
// "Data Analysis in High Energy Physics", Wiley-VCH, 2013.

// This C++ routine uses RooT classes, see http://root.cern.ch/

// sample interactive root session:
// root[1] .L exercise_7_1b.C
// root[2] .L exercise_7_1c.C
// root[3] h = plotimprovement(2);
// root[4] h->Draw("COLZ");

TH2F *plotimprovement (int nit,
                       double nx=160, double xl=0, double xr=16,
                       double ny=100, double yb=2, double yt=12) {
  stringstream title;
  title << "Log_{10} (Distance Ratio) for " << nit << " Iterations;E_{1};E_{2}";
  TH2F *h = new TH2F ("improvement", title.str().c_str(), nx, xl, xr, ny, yb, yt);
  for (int i = 1; i <= h->GetNbinsX(); ++i) {
    for (int j = 1; j <= h->GetNbinsY(); ++j) {
      double e1 = h->GetXaxis()->GetBinCenter (i);
      double e2 = h->GetYaxis()->GetBinCenter (j);
      double d0 = sqrt(pow (e1-6.14827646488922230, 2)+pow (e2-4.06618019582670698, 2));
      double l = 0;
      newtonit (nit, e1, e2, l, 6, 7, 1, 1.5, -0.5, 1.570796326794897, sqrt(50.0));
      double d1 = sqrt(pow (e1-6.14828, 2)+pow (e2-4.06618, 2));
      h->SetBinContent (i, j, log10(d1/d0));
    }
  }
  return h;
}
```

Figure 7.2 Solution to exercise 7.1c).

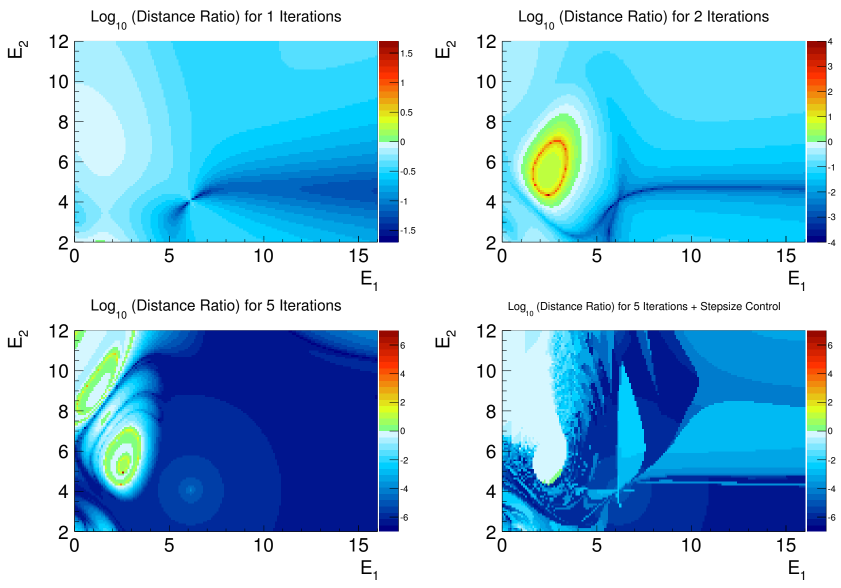

Figure 7.3 Plots for exercises 7.1c) and d). Plotted is $\log_{10}(d^{(\nu)}/d^{(0)})$, where $d^{(0)}$ ($d^{(\nu)}$) are the distances to the solution before (after) 1, 2, or 5 Newton steps. Regions with good convergence are light to dark blue, regions with no convergence are green to red. Convergence is good around the solution ($E_1 = 6.14828, E_2 = 4.06618$), but bad in an annular region around $E_1 \approx 2.5, E_2 \approx 4.5$. Stepsize control (lower right plot) improves convergence in that region, but slows down convergence in other regions, in part due to the Maratos effect.

d) The solution is shown in figures 7.4 and 7.3.

```
// Solution to Exercise 7.1d of
// B. List: "Constrained Fits", chapter 7 of
// O. Behnke, K. Kroeninger, G. Schott and T. Schoerner-Sadenius:
// "Data Analysis in High Energy Physics", Wiley-VCH, 2013.
// This C++ routine uses RooT classes, see http://root.cern.ch/
// sample interactive root session:
// root[1] .L exercise_7_1b.C
// root[2] .L exercise_7_1d.C
// root[3] h = plotimprovement2(2);
// root[4] h->Draw("COLZ");

int newtonit2 (int nit, double& e1, double& e2, double& l,
               double e1m, double e2m, double s1, double s2, double r, double psi, double m) {
  for (int n = 0; n < nit; ++n) {
    double e1try = e1, e2try=e2, ltry=l;
    int notok = newtonit (1, e1try, e2try, ltry, e1m, e2m, s1, s2, r, psi, m);
    if (notok) return 1;
    // get the direction vector p = (p1, p2) for the full Newton step
    double p1 = e1try-e1, p2 = e2try-e2;
    double mu = fabs(ltry);
    double phi1start = phi1 (e1, e2, e1m, e2m, s1, s2, r, psi, m, mu);
    double Dphi1calc = Dphi1 (e1, e2, e1m, e2m, s1, s2, r, psi, m, mu, p1, p2);
    double alpha = 1;
    double c1 = 0.1;
    // Armijo condition Eq. (7.86)
    while ((alpha > 1E-6) &&
           (phi1 (e1+alpha*p1, e2+alpha*p2, e1m, e2m, s1, s2, r, psi, m, mu) > phi1start+c1*alpha*Dphi1calc)) {
      alpha *= sqrt(0.5);   }
    e1 += alpha*p1;     e2 += alpha*p2;
    l += alpha*(ltry-l);    }
  return 0;    }
// Calculate the the merit function phi_1 = f + mu*|c|
double phi1 (double e1, double e2,
             double e1m, double e2m, double s1, double s2, double r, double psi, double m,
             double mu) {
  return (pow((e1-e1m)/s1, 2) - 2*r*(e1-e1m)*(e2-e2m)/(s1*s2) + pow((e2-e2m)/s2, 2))/(1-r*r)
         + mu*fabs (2*e1*e2*(1-cos(psi))-m*m);         }
// Calculate the directional derivative D(phi_1, p) for directions p that satisfy A*p = -c
double Dphi1 (double e1, double e2,
              double e1m, double e2m, double s1, double s2, double r, double psi, double m,
              double mu, double p1, double p2) {
  return 2*(    ((e1-e1m)/(s1*s1) - r*(e2-e2m)/(s1*s2))*p1 +
            (- r*(e1-e1m)/(s1*s2) +   (e2-e2m)/(s2*s2))*p2)/(1-r*r)
         - mu*fabs (2*e1*e2*(1-cos(psi))-m*m);         }
TH2F *plotimprovement2 (int nit,
                        double nx=160, double xl=0, double xr=16,
                        double ny=100, double yb=2, double yt=12) {
  stringstream title;
  title << "Log_{10} (Distance Ratio) for " << nit << " Iterations + Stepsize Control;E_{1};E_{2}";
  TH2F *h = new TH2F ("improvement2", title.str().c_str(), nx, xl, xr, ny, yb, yt);
  for (int i = 1; i <= h->GetNbinsX(); ++i) {
    for (int j = 1; j <= h->GetNbinsY(); ++j) {
      double e1 = h->GetXaxis()->GetBinCenter (i);
      double e2 = h->GetYaxis()->GetBinCenter (j);
      double d0 = sqrt(pow (e1-6.14827646488922230, 2)+pow (e2-4.06618019582670698, 2));
      double l = 0;
      newtonit2 (nit, e1, e2, l, 6, 7, 1, 1.5, -0.5, 1.570796326794897, sqrt(50.0));
      double d1 = sqrt(pow (e1-6.14828, 2)+pow (e2-4.06618, 2));
      h->SetBinContent (i, j, log10(d1/d0));     }     }
  return h;   }   }
```

Figure 7.4 Solution to exercise 7.1d).

e) This exercise is of an exploratory nature. Use the code from the preceding exercises, insert print statements and try to follow the iterations of the Newton algorithm. Try starting at points with good convergence, and with bad convergence. Read section 7.3.1 carefully and try to understand why the Newton method goes into a particular direction for a particular length, and why that direction and length are completely wrong sometimes.

Exercise 7.2: W decay

a) Equations (7.21) and (7.21) read

$$0 = c_1(\mathbf{x}) = \sum_j E_{\mathrm{j}}^{\mathrm{T}} \cos\phi_{\mathrm{j}} + E_{\mathrm{e}}^{\mathrm{T}} \cos\phi_{\mathrm{e}} + E_\nu^{\mathrm{T}} \cos\phi_\nu \,, \tag{7.11}$$

$$0 = c_2(\mathbf{x}) = \sum_j E_{\mathrm{j}}^{\mathrm{T}} \sin\phi_{\mathrm{j}} + E_{\mathrm{e}}^{\mathrm{T}} \sin\phi_{\mathrm{e}} + E_\nu^{\mathrm{T}} \sin\phi_\nu \,. \tag{7.12}$$

They imply

$$E_\nu^{\mathrm{T}} = \left(\left[\sum_j E_{\mathrm{j}}^{\mathrm{T}} \cos\phi_{\mathrm{j}} + E_{\mathrm{e}}^{\mathrm{T}} \cos\phi_{\mathrm{e}} \right]^2 + \left[\sum_j E_{\mathrm{j}}^{\mathrm{T}} \sin\phi_{\mathrm{j}} + E_{\mathrm{e}}^{\mathrm{T}} \sin\phi_{\mathrm{e}} \right]^2 \right)^{1/2} , \tag{7.13}$$

$$\tan\phi_\nu = \frac{-\sum_j E_{\mathrm{j}}^{\mathrm{T}} \sin\phi_{\mathrm{j}} - E_{\mathrm{e}}^{\mathrm{T}} \sin\phi_{\mathrm{e}}}{-\sum_j E_{\mathrm{j}}^{\mathrm{T}} \cos\phi_{\mathrm{j}} - E_{\mathrm{e}}^{\mathrm{T}} \cos\phi_{\mathrm{e}}} \,, \tag{7.14}$$

where the quadrant of ϕ_ν is determined by the signs of the denominator and the numerator of the right-hand side. This part of the solution is guaranteed to exist and is unambiguous (apart from the obvious fact that one could add any integer multiple of 2π to the azimuthal angle ϕ_ν).
Equation (7.22) reads

$$\begin{aligned} 0 &= c_3(\mathbf{x}) && (7.15)\\ &= M_W^2 - (E_{\mathrm{e}}^{\mathrm{T}} \cosh\eta_{\mathrm{e}} + E_\nu^{\mathrm{T}} \cosh\eta_\nu)^2 + (E_{\mathrm{e}}^{\mathrm{T}} \cos\phi_{\mathrm{e}} + E_\nu^{\mathrm{T}} \cos\phi_\nu)^2 \\ &\quad + (E_{\mathrm{e}}^{\mathrm{T}} \sin\phi_{\mathrm{e}} + E_\nu^{\mathrm{T}} \sin\phi_\nu)^2 + (E_{\mathrm{e}}^{\mathrm{T}} \sinh\eta_{\mathrm{e}} + E_\nu^{\mathrm{T}} \sinh\eta_\nu)^2 \,. && (7.16) \end{aligned}$$

We introduce the W boson's transverse momentum

$$p_{\mathrm{t,W}} = \sqrt{(E_{\mathrm{e}}^{\mathrm{T}} \cos\phi_{\mathrm{e}} + E_\nu^{\mathrm{T}} \cos\phi_\nu)^2 + (E_{\mathrm{e}}^{\mathrm{T}} \sin\phi_{\mathrm{e}} + E_\nu^{\mathrm{T}} \sin\phi_\nu)^2}$$

and

$$\begin{aligned} E_{\mathrm{e}} &= E_{\mathrm{e}}^{\mathrm{T}} \cosh\eta_{\mathrm{e}} \,, && (7.17)\\ p_{\mathrm{z,e}} &= E_{\mathrm{e}}^{\mathrm{T}} \sinh\eta_{\mathrm{e}} \,, && (7.18)\\ E_\nu &= E_\nu^{\mathrm{T}} \cosh\eta_\nu \,, && (7.19)\\ p_{\mathrm{z},\nu} &= E_\nu^{\mathrm{T}} \sinh\eta_\nu \,, && (7.20) \end{aligned}$$

and get from equation (7.15)

$$\begin{aligned} M_W^2 + p_{\mathrm{t,W}}^2 &= (E_{\mathrm{e}} + E_\nu)^2 - (p_{\mathrm{z,e}} + p_{\mathrm{z},\nu})^2 && (7.21)\\ &= E_{\mathrm{e}}^2 + 2E_{\mathrm{e}}E_\nu + E_\nu^2 - p_{\mathrm{z,e}}^2 - 2p_{\mathrm{z,e}}p_{\mathrm{z},\nu} - p_{\mathrm{z},\nu}^2 \,, && (7.22) \end{aligned}$$

$$M_W^2 + p_{\mathrm{t,W}}^2 - (E_{\mathrm{e}}^{\mathrm{T}})^2 - (E_\nu^{\mathrm{T}})^2 = 2E_{\mathrm{e}}E_\nu - 2p_{\mathrm{z,e}}p_{\mathrm{z},\nu}\,. \tag{7.23}$$

We abbreviate the left-hand side with $a = M_W^2 + p_{\mathrm{t,W}}^2 - (E_{\mathrm{e}}^{\mathrm{T}})^2 - (E_\nu^{\mathrm{T}})^2$, noting that a has the dimension of squared energy, but may be zero or negative because of the triangle inequality $p_{\mathrm{t,W}}^2 \leq (E_{\mathrm{e}}^{\mathrm{T}})^2 + (E_\nu^{\mathrm{T}})^2$, and get

$$\begin{aligned} a + 2p_{\mathrm{z,e}}p_{\mathrm{z},\nu} &= 2E_{\mathrm{e}}E_\nu\,, & (7.24)\\ a^2 + 4ap_{\mathrm{z,e}}p_{\mathrm{z},\nu} + 4p_{\mathrm{z,e}}^2p_{\mathrm{z},\nu}^2 &= 4E_{\mathrm{e}}^2((E_\nu^{\mathrm{T}})^2 + p_{\mathrm{z},\nu}^2)\,, & (7.25)\\ 4(E_{\mathrm{e}}^{\mathrm{T}})^2p_{\mathrm{z},\nu}^2 - 4ap_{\mathrm{z,e}}p_{\mathrm{z},\nu} - a^2 + 4E_{\mathrm{e}}^2(E_\nu^{\mathrm{T}})^2 &= 0 & (7.26)\end{aligned}$$

with the solution

$$p_{\mathrm{z},\nu} = \frac{-ap_{\mathrm{z,e}} \pm E_{\mathrm{e}}\sqrt{a^2 - 4(E_{\mathrm{e}}^{\mathrm{T}})^2(E_\nu^{\mathrm{T}})^2}}{2(E_{\mathrm{e}}^{\mathrm{T}})^2}\,. \tag{7.27}$$

b) Evidently, the solution is neither unique (except for the special case when the radicand is zero) nor does it always exist. The fundamental reason for this is that the W boson's longitudinal momentum appears only squared in the mass constraint, which leads to the sign ambiguity.

c) In the absence of further constraints or measured values it is to be expected that the constrained-fit problem may also have two equally viable solutions. In order to explore both solutions, a possible strategy is to use both solutions of equation (7.27) as starting values for the iterative process.
If one uses the measured values to estimate the neutrino's parameters from equations (7.13, 7.14, 7.27), it is quite possible that due to measurement errors the radicand in equation (7.27) becomes negative and no solution exists.
If that happens, a solution for $p_{\mathrm{z},\nu}$ that is close to fulfilling the constraint c_3 should be chosen, for instance the solution that results from setting the negative radicand to zero: $p_{\mathrm{z},\nu} = \frac{-ap_{\mathrm{z,e}}}{2(E_{\mathrm{e}}^{\mathrm{T}})^2}$.

Exercise 7.3: Linear fit

a) For M measurements, we introduce M measured quantities $\hat{y}_{\mathrm{m}}$, $m = 1\ldots M$, plus $U = 2$ unmeasured quantities $\hat{a}$ and $\hat{b}$. The objective function f is given by

$$f = \sum_{m=1}^{M}(y_{\mathrm{m}} - \hat{y}_{\mathrm{m}})^2, \tag{7.28}$$

and there are $K = M$ constraints:

$$c_{\mathrm{k}} = \hat{y}_{\mathrm{k}} - ax_{\mathrm{k}} - b\,. \tag{7.29}$$

The Lagrange function L is then given by

$$L = \sum_{m=1}^{M} (y_\mathrm{m} - \hat{y}_\mathrm{m})^2 + \sum_{k=1}^{K} \lambda_\mathrm{k}(\hat{y}_\mathrm{k} - a x_\mathrm{k} - b)\,. \tag{7.30}$$

b) The $M + 2 + K = 2M + 2$ equations are

$$\begin{aligned}
0 = \tfrac{\partial L}{\partial \hat{y}_\mathrm{m}} &= -2(y_\mathrm{m} - \hat{y}_\mathrm{m}) + \lambda_\mathrm{m}\,, && m = 1 \ldots M,\\
0 = \tfrac{\partial L}{\partial a} &= -\sum_{k=1}^{K} \lambda_\mathrm{k} x_\mathrm{k}\,, &&\\
0 = \tfrac{\partial L}{\partial b} &= -\sum_{k=1}^{K} \lambda_\mathrm{k}\,, &&\\
0 = \tfrac{\partial L}{\partial \lambda_\mathrm{k}} &= (\hat{y}_\mathrm{k} - a x_\mathrm{k} - b)\,, && k = 1 \ldots K.
\end{aligned}$$

c) The standard solution is

$$\begin{aligned}
\hat{a} &= \frac{M \sum x_\mathrm{m} y_\mathrm{m} - \sum x_\mathrm{m} \sum y_\mathrm{m}}{M \sum x_\mathrm{m}^2 - (\sum x_\mathrm{m})^2}\,,\\
\hat{b} &= \sum y_\mathrm{m}/M - \hat{a} \sum x_\mathrm{m}/M\,,\\
\hat{y}_\mathrm{m} &= \hat{a} x_\mathrm{m} + \hat{b}\,.
\end{aligned}$$

Inserting this into the first set of equations yields

$$0 = -2(y_\mathrm{m} - \hat{y}_\mathrm{m}) + \lambda_\mathrm{m} \tag{7.31}$$

which implies

$$\lambda_\mathrm{m} = 2(y_\mathrm{m} - \hat{y}_\mathrm{m}) = 2(y_\mathrm{m} - \hat{a} x_\mathrm{m} - \hat{b})\,. \tag{7.32}$$

Inserting this into the next equation yields

$$\begin{aligned}
-\sum_{k=1}^{K} \lambda_\mathrm{k} x_\mathrm{k} &= -2 \sum_{k=1}^{K} (y_\mathrm{k} x_\mathrm{k} - \hat{a} x_\mathrm{k}^2 - \hat{b} x_\mathrm{k})\\
&= -2 \sum y_\mathrm{m} x_\mathrm{m} + 2\hat{a} \sum x_\mathrm{m}^2 + 2\hat{b} \sum x_\mathrm{m}\\
&= -2 \sum y_\mathrm{m} x_\mathrm{m} + 2\hat{a} \sum x_\mathrm{m}^2 + 2 \sum y_\mathrm{m} \sum x_\mathrm{m}/M - 2\hat{a} \sum x_\mathrm{m} \sum x_\mathrm{m}/M\\
&= -2 \left(\sum y_\mathrm{m} x_\mathrm{m} - \sum y_\mathrm{m} \sum x_\mathrm{m}/M\right) + 2\hat{a} \left(\sum x_\mathrm{m}^2 - \sum x_\mathrm{m} \sum x_\mathrm{m}/M\right)\\
&= 0
\end{aligned}$$

after inserting the solution for $\hat{a}$, as expected. For the next equation, we get

$$\begin{aligned}
-\sum_{k=1}^{K} \lambda_\mathrm{k} &= -2 \sum_{k=1}^{K} (y_\mathrm{k} - \hat{a} x_\mathrm{k} - \hat{b})\\
&= -2 \sum y_\mathrm{m} + 2\hat{a} \sum x_\mathrm{m} + 2M\hat{b}\\
&= -2 \sum y_\mathrm{m} + 2\hat{a} \sum x_\mathrm{m} + 2M \sum y_\mathrm{m}/M - 2M\hat{a} \sum x_\mathrm{m}/M\\
&= 0\,.
\end{aligned}$$

The last set of equations is evidently fulfilled by the definition of $\hat{y}_\mathrm{m}$.

d) A different penalty function would result in a different definition of f and its partial derivatives; for instance, one could use the negative logarithm of the Cauchy distribution as penalty function:

$$f = \sum_{\mathrm{m}} \log\left((y_{\mathrm{m}} - \hat{y}_{\mathrm{m}})^2 + (\Gamma/2)^2\right) . \tag{7.33}$$

That would only necessitate an iterative solution of the system of equations arising from the Lagrange method.
If one wanted to use the minimal distance between the measured point and the interpolating line for the penalty, one would introduce another set of unknowns: $\hat{x}_{\mathrm{m}}$, $m = 1 \ldots M$, and write

$$f = \sum_{\mathrm{m}} (y_{\mathrm{m}} - \hat{y}_{\mathrm{m}})^2 + \sum_{\mathrm{m}} (x_{\mathrm{m}} - \hat{x}_{\mathrm{m}})^2 , \tag{7.34}$$

and define a different set of constraints in terms of the unknown quantities $\hat{y}_{\mathrm{m}}$ and $\hat{x}_{\mathrm{m}}$ rather than $\hat{y}_{\mathrm{m}}$ and the known abscissa values x_{m}:

$$c_{\mathrm{k}} = \hat{y}_{\mathrm{m}} - \hat{a}\hat{x}_{\mathrm{k}} - \hat{b} . \tag{7.35}$$

We note that replacing the linear function $y = ax + b$ by a non-linear function would only change the form of the constraints in this approach and allow a set of measurements with errors in x and y to be fitted to any non-linear function.
These changes would be rather easy to implement in a framework that is able to solve (potentially large) constrained-fit problems.