Automatic numerical differentiation by maximum likelihood estimation of a linear Gaussian state space model

Automatic numerical differentiation by maximum likelihood estimation of a linear Gaussian state space model

Robert Piché[1]

***Abstract*— A linear Gaussian state-space smoothing algorithm is presented for off-line estimation of derivatives from a sequence of noisy measurements. The algorithm uses numerically stable square-root formulas, can handle simultaneous independent measurements and non-equally spaced abscissas, and can compute state estimates at points between the data abscissas. The state space model's parameters, including driving noise intensity, measurement variance, and initial state, are determined from the given data sequence using maximum likelihood estimation computed using an expectation maximisation iteration. In tests with synthetic biomechanics data, the algorithm is found to be more accurate compared to a widely used open source automatic numerical differentiation algorithm, especially for acceleration estimation.**

I. INTRODUCTION

Numerical differentiation (ND) of a sequence of noisy measurements is an important problem in data analysis. For example, one may want to estimate velocity and acceleration from a sequence of displacement measurements. The problem has been well studied; comparative surveys of ND algorithms include [1], [2], [3], [4], [5], [6], [7], [8].

Because differentiation amplifies noise, catastrophically so when the sampling rate is high, an effective ND method must trade off data fidelity with noise smoothing. In most ND algorithms, the trade-off is governed by one or more user-defined parameters, variously called regularisation, smoothing, or bandwidth (cutoff frequency) parameters. Some ND algorithms are "automatic", in the sense that they determine the smoothing parameters for a given time series without knowledge of the true signal values. The surveys [4], [5] assess several automatic ND algorithms. While smoothing and ND codes are widely available, automatic ND codes are not: the fortran code for smoothing splines of [9] from the 1980's is still in use.

Fioretti and Jetto [10], [11] approach numerical differentiation as a standard state space smoothing problem with continuous-time dynamics and discrete-time measurements. Their state space dynamic model is a multiply-integrated stationary Wiener process (IWP), and the measurement error is an additive stationary discrete-time Gaussian white noise. In the target tracking literature this family of state-space models is known as the polynomial motion model [12, §6.2], of which the constant velocity model is the best known example.

This work presents an ND algorithm that further develops the algorithm of Fioretti and Jetto, in the following ways:

- it is able to process non-equally spaced and simultaneous data, and can evaluate at abscissas other than data points ("dense output").
- it uses a square-root smoother for better numerical stability;
- its parameters are computed using an established expectation-maximisation (EM) algorithm for state space model identification [15], [16];
- its MATLAB implementation is freely available [17].

The algorithm derivation is presented in section 2 and the appendix. In section 3, the algorithm's accuracy is compared to that of another open source automatic ND code, using tests with synthetic biomechanics benchmark data. The paper closes with ideas for possible extensions.

II. ALGORITHM

A. Signal model

The underlying signal is assumed to be the $(d-1)$-fold integral of a Wiener process. The linear stochastic differential equation is

$$\mathrm{d}x = Fx\,\mathrm{d}t + qL\,\mathrm{d}w$$

where w is the standard Wiener process, the underlying signal is the first component of the d-component state vector x, its first derivative is the second component, etc., and

$$F = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \quad L = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

The parameter $q > 0$ is the intensity (spectral density) of the driving white noise.

The abscissas for the discrete-time state space model are denoted t_k for $k = 1,2,\ldots$; the sequence $t_1, t_2, \ldots$ is assumed to be monotonically increasing. Denoting $x_k = x(t_k)$ and $\Delta_k = t_{k+1} - t_k$, the discrete-time dynamic model is a linear state space model driven by additive discrete Gaussian white noise

$$x_{k+1} | x_k \sim \mathrm{N}(A_k x_k, Q_k), \quad k = 1,2,\ldots, \tag{1}$$

where $\mathrm{N}(\cdot,\cdot)$ denotes a Gaussian distribution with given mean and covariance, the dynamic transition matrix is

$$A_k = \exp(F\Delta_k) = \begin{bmatrix} 1 & \Delta_k & \frac{1}{2!}\Delta_k^2 & \cdots & \frac{1}{(d-1)!}\Delta_k^{d-1} \\ 0 & 1 & \Delta_k & \cdots & \frac{1}{(d-2)!}\Delta_k^{d-2} \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix},$$

[1]Faculty of Information Technology and Communication Sciences, Tampere University, Tampere, Finland robert.piche@tuni.fi

and the driving noise covariance is $Q_k = q\bar{Q}_k$, where

$$\bar{Q}_k = \int_0^{\Delta_k} \exp(F(\Delta_k-\tau))LL^{\mathsf{T}}\exp(F^{\mathsf{T}}(\Delta_k-\tau))\,\mathrm{d}\tau$$
$$= \mathrm{diag}(\begin{bmatrix}\Delta_k^{\frac{2d-1}{2}}\\ \vdots \\ \Delta_k^{1/2}\end{bmatrix})\begin{bmatrix}\frac{1}{2d-1} & \cdots & \frac{1}{4} & \frac{1}{3}\\ \frac{1}{2d-2} & \cdots & \frac{1}{3} & \frac{1}{2}\\ \vdots & & & \vdots \\ \frac{1}{d} & \cdots & \frac{1}{2} & 1\end{bmatrix}\mathrm{diag}(\begin{bmatrix}\Delta_k^{\frac{2d-1}{2}}\\ \vdots \\ \Delta_k^{1/2}\end{bmatrix}).$$

At each abscissa, there are n_k scalar measurements, denoted $y_{k,1},\ldots,y_{k,n_k}$. Each measurement is modelled as the signal value plus independent additive zero-mean Gaussian noise, that is,

$$y_{k,j}|x_k \sim \mathrm{N}(Hx_k, R), \quad j = 1,\ldots,n_k,$$

where $H = [1,0,\cdots,0]$ and R is the variance. The measurement noise is also independent of process noise. Measurements at time k can be pooled into a single equivalent measurement $y_k = \frac{1}{n_k}\sum_{j=1}^{n_k} y_{k,j}$ having variance $R_k = R/n_k$, for which

$$y_k|x_k \sim \mathrm{N}(Hx_k, R_k). \tag{2}$$

B. Fixed-Interval Smoothing

Let $x_{k|j}$ denote the state conditioned on the measurements at times $t_1,\ldots,t_j$. For the linear Gaussian state space model (1) and (2), and a Gaussian prior distribution

$$x_{1|0} \sim \mathrm{N}(m_{1|0}, P_{1|0}), \tag{3}$$

all posterior states $x_{k|j}$ are jointly Gaussian. *Fixed-interval smoothing* is the computation of the mean and covariance of the states $x_{1|T},\ldots x_{T|T}$, given the model parameters

$$\theta = [q, R, m_{1|0}, P_{1|0}]$$

and the measurements $y_{1:T} = [y_{1,1},\ldots,y_{T,n_T}]$. The Rauch-Tung-Striebel (RTS) smoother computes these states sequentially, with a forward pass (a Kalman filter) that processes the measurements, followed by a backward pass. For better numerical stability, the QR factorisation-based square root RTS algorithm of [16] is used, as follows.

The *forward pass* consists of two stages that are carried out for each $k = 1,2,\ldots,T$. Before the beginning of the forward pass, $P_{1|0}^{1/2}$, the lower triangular Cholesky factor of $P_{1|0}$, is computed. The first stage, the *measurement update*, is the computation of the parameters of the filtering distribution $x_{k|k} \sim \mathrm{N}(m_{k|k}, P_{k|k}^{1/2}P_{k|k}^{\mathsf{T}/2})$ by the formulas

$$\begin{aligned}
\mathscr{R} &= \text{triangular factor of QR decomposition of}\\
&\qquad \begin{bmatrix} R_k^{1/2} & HP_{k|k-1}^{1/2}\\ 0 & P_{k|k-1}^{1/2}\end{bmatrix}^{\mathsf{T}}\\
S_k &= \mathscr{R}_{1,1}^{\mathsf{T}}\mathscr{R}_{1,1}\\
K_k &= \mathscr{R}_{1,2:d+1}^{\mathsf{T}}\mathscr{R}_{1,1}^{-\mathsf{T}}\\
P_{k|k}^{1/2} &= \mathscr{R}_{2:d+1,2:d+1}^{\mathsf{T}}\\
v_k &= y_k - Hm_{k|k-1}\\
m_{k|k} &= m_{k|k-1} + K_k v_k
\end{aligned}$$

The second stage, the *dynamic update*, is the computation of the parameters of the one-step prediction distribution $x_{k+1|k} \sim \mathrm{N}(m_{k+1|k}, P_{k+1|k}^{1/2}P_{k+1|k}^{\mathsf{T}/2})$. The formulas for the dynamic update are

$$\begin{aligned}
m_{k+1|k} &= A_k m_{k|k}\\
\mathscr{R} &= \text{triangular factor of QR decomposition of}\\
&\qquad \begin{bmatrix} P_{k|k}^{\mathsf{T}/2}A_k^{\mathsf{T}}\\ Q_k^{1/2}\end{bmatrix}^{\mathsf{T}}\\
P_{k+1|k}^{1/2} &= \mathscr{R}_{1:d,1:d}^{\mathsf{T}}
\end{aligned}$$

This stage is omitted for $k = T$.

In the *backward pass*, the parameters of the joint smoothing distribution

$$\begin{bmatrix}x_{k+1|T}\\ x_{k|T}\end{bmatrix} \sim \mathrm{N}\Big(\begin{bmatrix}m_{k+1|T}\\ m_{k|T}\end{bmatrix}, \begin{bmatrix}P_{k+1|T} & P_{k+1|T}G_k^{\mathsf{T}}\\ G_kP_{k+1|T} & P_{k|T}\end{bmatrix}\Big)$$

are computed sequentially for $k = T-1,\ldots,1$; the smoothing distribution is then $x_{k|T} \sim \mathrm{N}(m_{k|T}, P_{k|T})$. The backward pass formulas are

$$\begin{aligned}
P_{k|k} &= P_{k|k}^{1/2}P_{k|k}^{\mathsf{T}/2}\\
P_{k+1|k}^{-1} &= P_{k+1|k}^{-\mathsf{T}/2}P_{k+1|k}^{-1/2}\\
G_k &= P_{k|k}A_k^{\mathsf{T}}P_{k+1|k}^{-1}\\
m_{k|T} &= m_{k|k} + G_k(m_{k+1|T} - m_{k+1|k})\\
\mathscr{R} &= \text{triangular factor of QR decomposition of}\\
&\qquad \begin{bmatrix} P_{k|k}^{\mathsf{T}/2}A_k^{\mathsf{T}} & P_{k|k}^{\mathsf{T}/2}\\ Q_k^{\mathsf{T}/2} & 0\\ 0 & P_{k+1|T}^{\mathsf{T}/2}G_k^{\mathsf{T}}\end{bmatrix}\\
P_{k|T}^{1/2} &= \mathscr{R}_{d+1:2d,d+1:2d}
\end{aligned}$$

C. Estimation of model parameters

The maximum likelihood estimate of the model parameters $\theta = [q, R, m_{1|0}, P_{1|0}]$ is the maximiser of the likelihood $p(y_{1:T}|\theta)$, or equivalently the minimiser of the ML cost function

$$\phi(\theta) = -\log p(y_{1:T}|\theta).$$

For fixed θ, the cost function can be computed inside the Kalman filter (the first stage of the forward pass of the smoothing algorithm) using

$$\phi(\theta) = \tfrac{1}{2}\sum_{k=1}^{T}\big(\log\det(2\pi S_k) + v_k^{\mathsf{T}}S_k^{-1}v_k\big). \tag{4}$$

In the Expectation-Maximisation (EM) method the ML estimate is found by iteratively maximizing a lower bound on the likelihood. An EM method for state-space model parameters that uses a smoother to marginalise the state variables is presented in [15], [16]. This method is easily extended for the ND state space model, which has varying dynamic model matrices and a single-parameter process

noise matrix; see the appendix for details. The EM parameter update formulas are

$$q = \frac{1}{(T-1)d}\sum_{k=1}^{T-1}\operatorname{tr}(\hat{Q}_k\bar{Q}_k^{-1}),\; R = \frac{1}{T}\sum_{k=1}^{T} n_k\hat{R}_k, \tag{5a}$$

$$m_{1|0} = \hat{m}_{1|T},\; P_{1|0} = \hat{P}_{1|T}, \tag{5b}$$

where

$$\hat{Q}_k = [I,-A_k]\Big(\begin{bmatrix}\hat{m}_{k+1|T}\\ \hat{m}_{k|T}\end{bmatrix}\begin{bmatrix}\hat{m}_{k+1|T}\\ \hat{m}_{k|T}\end{bmatrix}^{\mathsf{T}} + \begin{bmatrix}\hat{P}_{k+1|T} & \hat{P}_{k+1|T}\hat{G}_k^{\mathsf{T}}\\ \hat{G}_k\hat{P}_{k+1|T} & \hat{P}_{k|T}\end{bmatrix}\Big)[I,-A_k]^{\mathsf{T}}, \tag{6a}$$

$$\hat{R}_k = (y_k - H\hat{m}_{k|T})(y_k - H\hat{m}_{k|T})^{\mathsf{T}} + H\hat{P}_{k|T}H^{\mathsf{T}}, \tag{6b}$$

and the other "hat" variables are computed by the smoother with the previous iterand of $\theta = [q, R, m_{1|0}, P_{1|0}]$.

Formulas (6) can also be written in the form

$$\begin{aligned}\hat{Q}_k = {} & (\hat{m}_{k+1|T} - A_k\hat{m}_{k|T})(\hat{m}_{k+1|T} - A_k\hat{m}_{k|T})^{\mathsf{T}}\\ & + (q^{1/2}A_kG_k\bar{Q}_k^{1/2})(q^{1/2}A_kG_k\bar{Q}_k^{1/2})^{\mathsf{T}}\\ & + \big((I-A_kG_k)(\hat{P}_{k+1|T}^{1/2} + A_k\hat{P}_{k|k}^{1/2})\big)\\ & \quad \big((I-A_kG_k)(\hat{P}_{k+1|T}^{1/2} + A_k\hat{P}_{k|k}^{1/2})\big)^{\mathsf{T}},\end{aligned} \tag{7a}$$

$$\hat{R}_k = (y_k - H\hat{m}_{k|T})(y_k - H\hat{m}_{k|T})^{\mathsf{T}} + (H\hat{P}_{k|T}^{1/2})(H\hat{P}_{k|T}^{1/2})^{\mathsf{T}}. \tag{7b}$$

Each term in (7) is a product of a matrix with its transpose. Linear algebra software libraries include codes to compute products of this form efficiently and with exact preservation of symmetry; for example the multiplication operator $\star$ in MATLAB is overloaded to do this.

D. Dense output

The posterior estimate of the state at an inter-abscissa time $t_{k+\tau} = t_k + \tau\Delta_k$, with $0 < \tau < 1$, conditional on the measurements at times up to and including t_k, is denoted

$$x_{k+\tau|k} \sim \mathrm{N}(m_{k+\tau|k}, P_{k+\tau|k}).$$

Its parameters can be obtained using the RTS smoother forward pass formulas by omitting the measurement update stage and applying a dynamic update stage with the modified dynamic model

$$m_{k+\tau|k} = A_{k,\tau}m_{k|k} \tag{8a}$$

$$P_{k+\tau|k} = A_{k,\tau}P_{k|k}A_{k,\tau}^{\mathsf{T}} + Q_{k,\tau} \tag{8b}$$

where $A_{k,\tau} = \exp(F\tau\Delta_k)$ and $Q_{k,\tau} = q\bar{Q}_{k,\tau}$ with

$$\bar{Q}_{k,\tau} = \int_0^{\tau\Delta_k}\exp(F(\tau\Delta_k - \tau))LL^{\mathsf{T}}\exp(F^{\mathsf{T}}(\tau\Delta_k - \tau))\,\mathrm{d}\tau.$$

That is, the formulas for the modified model matrices are obtained by using $\tau\Delta_k$ in place of Δ_k in the formulas for the dynamic transition matrix and process noise covariance given earlier.

The posterior estimate of the interpolatory state conditional on *all* the measurements,

$$x_{k+\tau|T} \sim \mathrm{N}(m_{k+\tau|T}, P_{k+\tau|T}),$$

is obtained using the backward pass formula to go from t_{k+1} to $t_{k+\tau}$ instead of to t_k:

$$G_{k,\tau} = P_{k+\tau|k}A_{k,1-\tau}^{\mathsf{T}}P_{k+1|k}^{-1} \tag{9a}$$

$$m_{k+\tau|T} = m_{k+\tau|k} + G_{k,\tau}(m_{k+1|T} - m_{k+1|k}) \tag{9b}$$

The functional form of the interpolant can be inferred from these formulas. Substituting (8) and (9a) into (9b) gives

$$\begin{aligned}m_{k+\tau|T} = {} & A_{k,\tau}m_{k|k}\\ & + (A_{k,\tau}P_{k|k}A_{k,\tau}^{\mathsf{T}} + Q_{k,\tau})A_{k,1-\tau}^{\mathsf{T}}P_{k+1|k}^{-1}(m_{k+1|T} - m_{k+1|k}).\end{aligned}$$

Because the coefficients of $A_{k,\tau}$, $A_{k,1-\tau}$ and $Q_{k,\tau}$ are polynomials in Δ_k, so is the interpolant $m_{k+\tau|T}$. In particular, its first component (the displacement) is a polynomial of degree $2d-1$.

E. Initial parameters

Although EM has good theoretical convergence properties, the convergence can be slow. This slowness can be offset by making a reasonably good choice of initial parameter values. In the MATLAB implementation, the initial iterands for the state $m_{1|0}$ and the measurement noise variance R are set by least-squares fitting a straight line through the first 10 abscissas. The covariance $P_{1|0}$ is set to a tiny multiple of the identity matrix. The driving noise intensity q is then set by minimizing the negative log likelihood, a univariate minimization whose cost function (4) is computed using a Kalman filter.

III. TESTS

Corradini et al. [4] compare ND algorithms using five analytically defined test functions that resemble experimental measurements of different kinds of human movement (Fig. 1). They considered different measurement noise levels and sampling rates, and found no large differences in accuracy between the five algorithms that they tested. They identify two algorithms, which they label F1 and F2, as being the most accurate: the smoothing seventh-degree spline of [9] (widely used because its code is freely available) and the fixed-lag Kalman smoother of [10] with three states (whose code is not published). These are also the only algorithms in their tests that are automatic, except that the measurement noise variance needs to be specified.

Table I shows the errors of displacement, velocity, and acceleration estimates for the 94-point time series obtained by sampling the analytical displacement functions used in [4] and adding Gaussian noise. Results are the averages of 20 replications with different random number generator seeds. The estimates reported for method F1 were computed by fitting a cubic splines using a C code conversion [18] of the Fortran package of [9]; The "new method" estimates were computed using the Matlab implementation of the algorithm proposed in this paper.

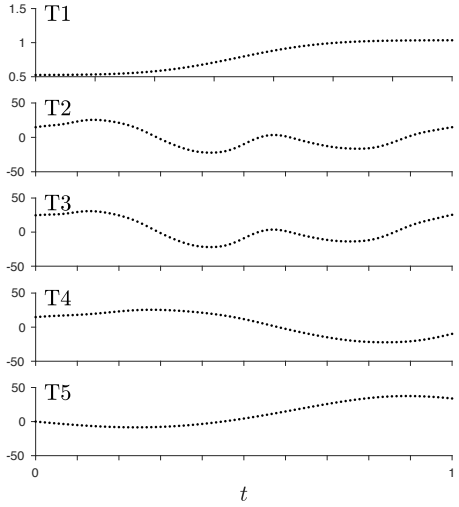

Fig. 1. Test displacement functions from [4] used in tests.

test	method	displ.	vel.	accel.
T1	F1	0.17	4.4	31.9
	new	0.16	3.7	16.3
T2	F1	3.23	11.7	46.9
	new	3.05	9.7	27.8
T3	F1	2.76	13.9	47.2
	new	2.57	11.7	26.9
T4	F1	2.01	11.8	52.1
	new	1.91	9.5	37.0
T5	F1	1.71	12.7	68.8
	new	1.41	7.4	21.0

TABLE I

RELATIVE RMS ERRORS (IN PERCENTAGE) OF MOTION ESTIMATES FOR FIVE SYNTHETIC DISPLACEMENT DATA SETS; AVERAGES FOR 20 NOISE REALISATIONS.

Table I shows the errors of displacement, velocity, and acceleration estimates reported in [4] for 94-point noisy displacement sequences generated from five test functions. The error of the estimate of the derivative sequence is reported as the percentage of RMS error relative to the true sequence's RMS value. Also shown are the errors found with the proposed algorithm with $d = 3$ states. The EM iterations were repeated until the norm of the change in the displacement estimate was less than 0.1% of the norm of the estimate; no more than 3 EM iterations were needed in any of the tests.

In these experiments, the proposed method is slightly better than the reference method for displacement smoothing, somewhat better for velocity estimation, and clearly better for acceleration estimation.

IV. CONCLUSIONS

The proposed automatic ND algorithm is based on the integrated Wiener process, which as argued in [11] is a principled and flexible signal model for estimation of derivatives from noisy time series. The algorithm goes beyond that of [11] in several ways: it uses a numerically stable square-root smoother algorithm, allows non-equally spaced and simultaneous data, and its implementation is freely available. Its maximum likelihood parameters are computed using an established EM iteration. In tests with benchmark data, the proposed automatic ND algorithm displays accuracy as good as or better than a popular ND code based on smoothing splines.

There are a number of open questions and possible further developments.

Fixed-lag smoothing: It would be straightforward to replace the fixed-interval smoother with a fixed-lag smoother, as was done in [10], [11]. The advantages would be less memory complexity and the ability to have near-real time operation. The disadvantages would be more code complexity, more tuning parameters, and suboptimal estimates.

Model degree: In the proposed algorithm the dynamic model degree d is a fixed parameter chosen by the user. Automatic tuning of d might improve the algorithm's usability and accuracy.

Outlier-robustness: Measurements with sporadic outliers could be better handled by replacing the RTS smoother by a Student-t smoother [19].

APPENDIX

The following is an adaptation of the derivations in [20, §12.2.3] and [16]. Substituting the state space model's data log-likelihood

$$\log p(x_{1:T}, y_{1:T} \,|\, \theta) = \log p(x_1 \,|\, \theta) + \sum_{k=1}^{T-1} \log p(x_{k+1} \,|\, x_k, \theta) + \sum_{k=1}^{T} \log p(y_k \,|\, x_k, \theta)$$

into the EM objective function

$$\mathcal{Q}(\theta, \hat{\theta}) = \int p(x_{1:T} \,|\, y_{1:T}, \hat{\theta}) \log p(x_{1:T}, y_{1:T} \,|\, \theta) \, \mathrm{d}x_{1:T}$$

(where $\hat{\theta}$ is the previous iteration's parameter value) gives

$$\begin{aligned}\mathcal{Q}(\theta, \hat{\theta}) = &\int p(x_1 \,|\, y_{1:T}, \hat{\theta}) \log p(x_1 \,|\, \theta) \, \mathrm{d}x_1 \\ &+ \sum_{k=1}^{T-1} \int p(x_{k+1}, x_k \,|\, y_{1:T}, \hat{\theta}) \log p(x_{k+1} \,|\, x_k, \theta) \, \mathrm{d}x_{k+1} \\ &+ \sum_{k=1}^{T} \int p(x_k \,|\, y_{1:T}, \hat{\theta}) \log p(y_k \,|\, x_k, \theta) \, \mathrm{d}x_k \, \mathrm{d}x_k.\end{aligned}$$

This is a sum of expectations of log terms. From (3), (1), (2), the log terms are

$$\begin{aligned}\log p(x_1 \,|\, \theta) &= -\tfrac{1}{2} \log \det(2\pi P_{1|0}) \\ &\quad - \tfrac{1}{2} (x_1 - m_{1|0})^{\mathsf{T}} P_{1|0}^{-1} (x_1 - m_{1|0}), \\ \log p(x_{k+1} \,|\, x_k, \theta) &= -\tfrac{1}{2} \log \det(2\pi q \bar{Q}_k) \\ &\quad - \tfrac{1}{2} (x_{k+1} - A_k x_k)^{\mathsf{T}} (q \bar{Q}_k)^{-1} (x_{k+1} - A_k x_k), \\ \log p(y_k \,|\, x_k, \theta) &= -\tfrac{1}{2} \log \det(2\pi R_k) \\ &\quad - \tfrac{1}{2} (y_k - H x_k)^{\mathsf{T}} R_k^{-1} (y_k - H x_k).\end{aligned}$$

The distributions with respect to which the expectations are taken are

$$\begin{aligned} x_1 &\sim \mathrm{N}(\hat{m}_{1|0}, \hat{P}_{1|0}), \\ \begin{bmatrix} x_{k+1} \\ x_k \end{bmatrix} &\sim \mathrm{N}\Big(\begin{bmatrix} \hat{m}_{k+1|T} \\ \hat{m}_{k|T} \end{bmatrix}, \begin{bmatrix} \hat{P}_{k+1|T} & \hat{P}_{k+1|T}\hat{G}_k^{\mathsf{T}} \\ \hat{G}_k\hat{P}_{k+1|T} & \hat{P}_{k|T} \end{bmatrix} \Big), \\ x_k &\sim \mathrm{N}(\hat{m}_{k|T}, \hat{P}_{k|T}), \end{aligned}$$

where hats indicate values that are computed by the smoothing algorithm applied to the model having parameters $\hat{\theta} = [\hat{q}, \hat{R}, \hat{m}_{1|0}, \hat{P}_{1|0}]$. Computing the expectations gives the formula for the EM objective function as

$$\begin{aligned} \mathscr{Q}(\theta, \hat{\theta}) = &-\tfrac{1}{2}\log\det(2\pi P_{1|0}) - \tfrac{1}{2}\operatorname{tr}\big(P_{1|0}^{-1}\hat{P}_{1|T}\big) \\ &- \tfrac{1}{2}(\hat{m}_{1|T} - m_{1|0})^{\mathsf{T}} P_{1|0}^{-1}(\hat{m}_{1|T} - m_{1|0}) \\ &- \tfrac{1}{2}\sum_{k=1}^{T-1}\Big(\log\det(2\pi Q_k) + \operatorname{tr}\big(Q_k^{-1}\hat{Q}_k\big)\Big) \\ &- \tfrac{1}{2}\sum_{k=1}^{T}\Big(\log(2\pi R_k) + R_k^{-1}\hat{R}_k\Big), \end{aligned}$$

where $\hat{Q}_k$ and $\hat{R}_k$ are given by (6). Using standard matrix differential calculus formulas [21], the partial derivatives of the EM objective function are

$$\begin{aligned} \partial\mathscr{Q}(\theta,\hat{\theta})/\partial q &= \tfrac{1}{2}\sum_{k=1}^{T-1}\operatorname{tr}\Big(Q_k^{-1}\tfrac{\partial Q_k}{\partial q}\big(-I + \hat{Q}_k Q_k^{-1}\big)\Big) \\ &= \tfrac{1}{2}\Big(-\tfrac{(T-1)d}{q} + \tfrac{1}{q^2}\sum_{k=1}^{T-1}\operatorname{tr}(\hat{Q}_k\bar{Q}_k^{-1})\Big), \\ \partial\mathscr{Q}(\theta,\hat{\theta})/\partial R &= \tfrac{1}{2}\sum_{k=1}^{T} R^{-1}\big(-1 + n_k\hat{R}_k R^{-1}\big), \\ \partial\mathscr{Q}(\theta,\hat{\theta})/\partial m_{1|0}^{\mathsf{T}} &= (\hat{m}_{1|T} - m_{1|0})^{\mathsf{T}} P_{1|0}^{-1}, \\ \partial\mathscr{Q}(\theta,\hat{\theta})/\partial P_{1|0} &= \tfrac{1}{2}P_{1|0}^{-1}\Big(-I + \big(\hat{P}_{1|T} \\ &\qquad + (\hat{m}_{1|T} - m_{1|0})(\hat{m}_{1|T} - m_{1|0})^{\mathsf{T}}\big)\Big)P_{1|0}^{-1}. \end{aligned}$$

Setting these to zero and solving gives the EM update formulas (5–6).

The covariance matrix in (6a) can be written as

$$\begin{aligned} &\begin{bmatrix} \hat{P}_{k+1|T} & \hat{P}_{k+1|T}\hat{G}_k^{\mathsf{T}} \\ \hat{G}_k\hat{P}_{k+1|T} & \hat{P}_{k|T} \end{bmatrix} \\ &= \begin{bmatrix} I & 0 \\ \hat{G}_k & I \end{bmatrix} \begin{bmatrix} \hat{P}_{k+1|T} & 0 \\ 0 & \hat{P}_{k|T} - \hat{G}_k\hat{P}_{k+1|T}\hat{G}_k^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} I & 0 \\ \hat{G}_k & I \end{bmatrix}^{\mathsf{T}}. \end{aligned}$$

Substituting the identities

$$\hat{P}_{k|T} = \hat{P}_{k|k} + \hat{G}_k(\hat{P}_{k+1|T} - \hat{P}_{k+1|k})\hat{G}_k^{\mathsf{T}}$$

and

$$\hat{P}_{k+1|k} = A_k\hat{P}_{k|k}A_k^{\mathsf{T}} + q\bar{Q}_k,$$

and applying the Joseph formula, the last element of the diagonal matrix can be rewritten as

$$\begin{aligned} \hat{P}_{k|T} - \hat{G}_k\hat{P}_{k+1|T}\hat{G}_k^{\mathsf{T}} &= \hat{P}_{k|k} - \hat{G}_k\hat{P}_{k+1|k}\hat{G}_k^{\mathsf{T}} \\ &= \hat{P}_{k|k} - \hat{G}_k(A_k\hat{P}_{k|k}A_k^{\mathsf{T}} + q\bar{Q}_k)\hat{G}_k^{\mathsf{T}} \\ &= (I - \hat{G}_kA_k)\hat{P}_{k|k}(I - \hat{G}_kA_k)^{\mathsf{T}} + q\hat{G}_k\bar{Q}_k\hat{G}_k^{\mathsf{T}}. \end{aligned}$$

Formula (7a) is then obtained by replacing the covariance matrices by their Cholesky factorisations.

ACKNOWLEDGEMENT

Siva Sankar Kannan debugged the C code of [18] and did the numerical experiments reported in Table 1.

REFERENCES

[1] J. C. Pezzack, R. W. Norman, and D. A. Winter, "An assessment of derivative determining techniques used for motion analysis," *J. of Biomechanics*, vol. 10, no. 5-6, pp. 377–382, 1977.

[2] R. H. Brown, S. C. Schneider, and M. G. Mulligan, "Analysis of algorithms for velocity estimation from discrete position versus time data," *IEEE Trans. on Industrial Electronics*, vol. 39, no. 1, pp. 11–19, 1992.

[3] M. D'Amico and G. Ferrigno, "Comparison between the more recent techniques for smoothing and derivative assessment in biomechanics," *Medical and Biological Engineering and Computing*, vol. 30, no. 2, pp. 193–204, 1992.

[4] M. L. Corradini, S. Fioretti, and T. Leo, "Numerical differentiation in movement analysis: how to standardise the evaluation of techniques," *Medical and Biological Engineering and Computing*, vol. 31, pp. 187–197, 1993.

[5] G. Giakas and V. Baltzopoulos, "A comparison of automatic filtering techniques applied to biomechanical walking data," *J. of Biomechanics*, vol. 30, no. 8, pp. 847–850, 1997.

[6] J. A. Walker, "Estimating velocities and accelerations of animal locomotion: a simulation experiment comparing numerical differentiation algorithms," *J. of Experimental Biology*, vol. 201, no. 7, pp. 981–995, 1998.

[7] K. Ahnert and M. Abel, "Numerical differentiation of experimental data: local versus global methods," *Computer Physics Communications*, vol. 177, pp. 764–774, 2007.

[8] L. J. Puglisi, R. J. Saltaren, and C. E. G. Cena, "On the velocity and acceleration estimation from discrete time-position signal of linear encoders," *J. of Control Engineering and Applied Informatics*, vol. 17, no. 3, pp. 30–40, 2015.

[9] H. J. Woltring, "A FORTRAN package for generalized, cross-validatory spline smoothing and differentiation," *Advances in Engineering Software*, vol. 8, pp. 104–113, 1986, source code and Matlab interfaces http://isbweb.org/software/sigproc.html.

[10] S. Fioretti and L. Jetto, "Accurate derivative estimation from noisy data: a state-space approach," *International Journal of Systems Science*, vol. 20, no. 1, pp. 33–53, 1989.

[11] ——, "Low a-priori statistical information model for optimal smoothing and differentiation of noisy signals," *Int. J. of Adaptive Control and Signal Processing*, vol. 8, pp. 305–320, 1994.

[12] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation With Applications to Tracking and Navigation*. Wiley, 2001.

[13] S. Fioretti and L. Jetto, "A new algorithm for the sequential estimation of the regularization parameter in the spline smoothing problem," in *Proc. 31st Conf. on Decision and Control*, 1992, pp. 2245–2246.

[14] G. De Nicolao, G. Ferrari-Trecate, and S. G., "Fast spline smoothing via spectral factorization concepts," *Automatica*, vol. 36, pp. 1733–1739, 2000.

[15] R. H. Shumway and D. S. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *J. of Time Series Analysis*, vol. 3, no. 4, pp. 253–264, 1982.

[16] S. Gibson and B. Ninness, "Robust maximum-likelihood estimation of multivariable dynamic systems," *Automatica*, vol. 41, 2005.

[17] R. Piche. (2017, Dec) DEREST. [Online]. Available: https://se.mathworks.com/matlabcentral/fileexchange/63925-derest

[18] D. Meglan. (1994, Apr) GCVSPL in C. [Online]. Available: https://isbweb.org/software/sigproc.html

[19] R. Piché, S. Särkkä, and J. Hartikainen, "Robust outlier-robust filtering and smoothing for nonlinear systems using the multivariate student-t distribution," in *IEEE Int. Workshop on Machine Learning for Signal Processing*, 2012.

[20] S. Särkkä, *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[21] J. Magnus and H. Neudecker, *Matrix Differential Calculus With Applications in Statistics and Econometrics*, 3rd ed. Wiley, 2007.