Linearly Constrained Gaussian Processes with Boundary Conditions

Markus Lange-Hegermann
Department of Electrical Engineering and Computer Science
OWL University of Applied Sciences and Arts
markus.lange-hegermann@th-owl.de

A Proof of Lemma 2.2

Before giving the proof of Lemma 2.2, we recall the definition (if it exists) of the ℓ-th cumulant function $\kappa_\ell(g)$

$$\kappa_\ell(g)\left(x^{(1)},\dots,x^{(\ell)}\right)=\sum_{\pi\in\operatorname{part}(\ell)}(-1)^{|\pi|-1}(|\pi|-1)!\prod_{\tau\in\pi}E\left(\prod_{i\in\tau}g\left(x^{(i)}\right)\right)$$

of a stochastic process g, where $\operatorname{part}(\ell)$ is the set of partitions of ℓ and $|\pi|$ denotes the cardinality of π. In particular, the first two cumulant functions κ_1 resp. κ_2 are equal to the mean resp. covariance function. Furthermore, g is Gaussian iff all but the first two cumulant functions vanish.

The stochastic process B_*g exists, as $\mathcal{F}$ is an R-module and the realizations of g are all contained in $\mathcal{F}$. The compatibility with expectations proves the following formula for the cumulant functions of $\kappa(B_*g)$ of B_*g, where $B^{(i)}$ denotes the operation of B on functions with argument $x^{(i)}\in\mathbb{R}^d$:

$$\begin{aligned}
&\kappa_\ell(B_*g)\left(x^{(1)},\dots,x^{(\ell)}\right)\\
&=\sum_{\pi\in\operatorname{part}(\ell)}(-1)^{|\pi|-1}(|\pi|-1)!\cdot\prod_{\tau\in\pi}E\left(\prod_{i\in\tau}(B_*g)\left(x^{(i)}\right)\right)\\
&=\sum_{\pi\in\operatorname{part}(\ell)}(-1)^{|\pi|-1}(|\pi|-1)!\cdot\prod_{\tau\in\pi}\left(\prod_{i\in\tau}B^{(i)}\right)E\left(\prod_{i\in\tau}g\left(x^{(i)}\right)\right)\\
&\qquad\text{(as } B \text{ commutes with expectation)}\\
&=\sum_{\pi\in\operatorname{part}(\ell)}(-1)^{|\pi|-1}(|\pi|-1)!\cdot\ \widehat{B}\prod_{\tau\in\pi}E\left(\prod_{i\in\tau}g\left(x^{(i)}\right)\right)\\
&\qquad\text{(as } \pi \text{ is a partition; } \widehat{B}:=\textstyle\prod_{i\in\{1,\dots,\ell\}}B^{(i)}\text{)}\\
&=\widehat{B}\sum_{\pi\in\operatorname{part}(\ell)}(-1)^{|\pi|-1}(|\pi|-1)!\cdot\prod_{\tau\in\pi}E\left(\prod_{i\in\tau}g\left(x^{(i)}\right)\right)\\
&\qquad\text{(as } B \text{ is linear)}\\
&=\widehat{B}\,\kappa_\ell(g)\left(x^{(1)},\dots,x^{(\ell)}\right)
\end{aligned}$$

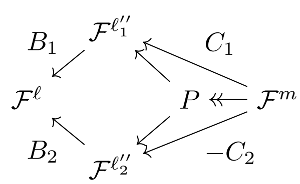
As g is Gaussian, the higher ($\ell\geq 3$) cumulants $\kappa_\ell(g)$ vanish, hence the higher ($\ell\geq 3$) cumulants $\kappa_\ell(B_*g)$ vanish, which implies that B_*g is Gaussian. The formulas for the mean function resp. covariance function follow from the above computation for $\ell=1$ resp. $\ell=2$. □

B Proof of Theorem 5.2

Before giving the proof of Theorem 5.2, we recall some definitions and facts from homological algebra and category theory nLab authors (2020); Mac Lane (1998); Weibel (1994); Cartan and Eilenberg (1999). A collection of two morphisms with the same source $A_2 \xleftarrow{\alpha_1} B \xrightarrow{\alpha_2} A_2$ is a *span* and a collection of two morphisms with the same range $C_2 \xrightarrow{\gamma_1} D \xleftarrow{\gamma_2} C_2$ is a *cospan*. Given a cospan $C_2 \xrightarrow{\gamma_1} D \xleftarrow{\gamma_2} C_2$, an object P together with two morphisms $\delta_1 : P \to C_1$ and $\delta_2 : P \to C_2$ is called a *pullback*, if $\gamma_1 \circ \delta_1 = \gamma_2 \circ \delta_2$ and for every P' with two morphisms $\delta_1' : P' \to C_1$ and $\delta_2' : P' \to C_2$ such that $\gamma_1 \circ \delta_1' = \gamma_2 \circ \delta_2'$ there exists a unique morphism $\pi : P' \to P$ such that $\delta_1 \circ \pi = \delta_1'$ and $\delta_2 \circ \pi = \delta_2'$. Pullbacks are the generalization of intersections. Given a span $A_2 \xleftarrow{\alpha_1} B \xrightarrow{\alpha_2} A_2$, an object P together with two morphisms $\beta_1 : A_1 \to P$ and $\beta_2 : A_2 \to P$ is called a *pushout*, if $\beta_1 \circ \alpha_1 = \beta_2 \circ \alpha_2$ and for every P' with two morphisms $\beta_1' : A_1 \to P'$ and $\beta_2' : A_2 \to P'$ such that $\beta_1' \circ \alpha_1 = \beta_2' \circ \alpha_2$ there exists a unique morphism $\pi : P \to P'$ such that $\beta_1 \circ \pi = \beta_1'$ and $\beta_2 \circ \pi = \beta_2'$. Pullbacks and Pushouts exist in the category of finitely presented modules. Given an R-module M, an epimorphism $M \twoheadleftarrow \mathbb{R}^m$ is a *free cover* of M and a monomorphism $M \hookrightarrow \mathbb{R}^m$ is a *free hull* of M. Every finitely presented R-module has a free cover, but only a free hull iff it corresponds to a controllable system. Given an R-module M, the *contravariant hom-functor* $\hom_R(-, M)$ is the hom-set $\hom_R(A, M) = \{\psi : A \to M \mid \psi\ R\text{-module homomorphism}\}$ when applied to an R-module A and application to an R-module homomorphism $\varphi : A \to B$ gives $\hom_R(\varphi, M) : \hom_R(B, M) \to \hom_R(A, M) : \beta \mapsto \beta \circ \varphi$. If R is a commutative, then $\hom_R(-, M)$ is a functor to the category of R-modules, otherwise it is a functor to the category of Abelian groups.

By Corollary 3.7, the assumptions of Theorem 5.2 ensure that we have a parametrization C of the system defined by B. As C is the nullspace of B, we have $B_1C_1 = -B_2C_2$.

The parametrization of an intersection of parametrizations $B_1\mathcal{F}^{\ell_1''} \cap B_2\mathcal{F}^{\ell_2''}$ is given by the image of the pullback P of the cospan $\mathcal{F}^{\ell_1''} \xrightarrow{B_1} \mathcal{F}^{\ell} \xleftarrow{B_2} \mathcal{F}^{\ell_2''}$ in $\mathcal{F}^{\ell}$ by (Eisenbud, 1995, 15.10.8.a). The approach of Theorem 5.2 computes a subset[1] of this image via a free cover $P \twoheadleftarrow \mathcal{F}^m$ of this pullback P as image of $B_1C_1 = -B_2C_2$, as depicted in the following commutative diagram:



As in Theorem 3.6 and Corollary 3.7, the computation is done dually over the ring R. There, the cospan $R^{1\times\ell_1''} \xrightarrow{C_1} R^{1\times m} \xleftarrow{C_2} R^{1\times\ell_2''}$ defines a free hull $Q \xhookrightarrow{C} R^{1\times m}$ of the pushout Q of the span $R^{1\times\ell_1''} \xleftarrow{B_1} R^{1\times\ell} \xrightarrow{B_2} R^{1\times\ell_2''}$. Then applying the dualizing hom-functor $\hom_R(-, \mathcal{F})$ transforms this to the function space $\mathcal{F}$. □

Even though all operations in this proof are algorithmic (Barakat and Lange-Hegermann, 2011), Theorem 5.2 describes a computationally more efficient algorithm.

C Code

The following computation have been performed in Maple with the OreModules package (Chyzak et al., 2007).

Example C.1 (General Code for GP regression)**.**

[1] To get the full image, we need $\mathcal{F}$ to be an injective module.

```
> # code for GP regression
> GP:=proc(Kf,
>  points,yy,epsilon)
> local n,m,kf,K,s1,s2,alpha,KStar;
>  n:=nops(points);
>  m:=RowDimension(Kf);
>  s1:=map(
>   a->[x1=a[1],y1=a[2],z1=a[3]],
>   points);
>  s2:=map(
>   a->[x2=a[1],y2=a[2],z2=a[3]],
>   points);
>  kf:=convert(Kf,listlist);

>  K:=convert(
>   evalf(
>    map(
>     a->map(
>      b->convert(
>       subs(a,subs(b,kf)),
>       Matrix),
>      s2),
>     s1)),
>   Matrix):
>  alpha:=yy.(K+epsilon^2)^(-1);
>  KStar:=map(
>   a->subs(a,kf),
>   s1):
>  KStar:=subs(
>   [x2=x,y2=y,z2=z],KStar):
>  KStar:=convert(
>   map(op,KStar),Matrix):
>  return alpha.KStar;
> end:
```

Example C.2 (Code for Example 3.9)**.**

```
> restart;
```

```
> with(OreModules):
```

```
> with(LinearAlgebra):
```

```
> Alg:=DefineOreAlgebra(diff=[Dx,x],
>  diff=[Dy,y], diff=[Dz,z],
>  diff=[Dx1,x1], diff=[Dy1,y1],
>  diff=[Dz1,z1], diff=[Dx2,x2],
>  diff=[Dy2,y2], diff=[Dz2,z2],
>  polynom=[x,y,z,x1,x2,y1,y2,z1,z2]):
> A:=«x,Dx>|<y,Dy>|<z,Dz»;
```

$$A := \begin{bmatrix} x & y & z \\ Dx & Dy & Dz \end{bmatrix}$$

```
> # combine
> B:=Involution(
>  SyzygyModule(
>   Involution(A,Alg),
>   Alg),
>  Alg);
```

$$B := \begin{bmatrix} z\,Dy - Dz\,y \\ -Dx\,z + Dz\,x \\ Dx\,y - Dy\,x \end{bmatrix}$$

```
> # check parametrization
> A1:=SyzygyModule(B,Alg):
> ReduceMatrix(A,A1,Alg);
> ReduceMatrix(A1,A,Alg);
```

$$[]$$
$$[]$$

```
> # covariance for
> # parametrizing function
> SE:=exp(-1/2*(x1-x2)^2
>  -1/2*(y1-y2)^2-1/2*(z1-z2)^2):
> Kg:=unapply(
>  DiagonalMatrix([SE]),
>  (x1,y1,z1,x2,y2,z2)):
> # prepare covariance
> P2:=ApplyMatrix(B,
>  [xi(x,y,z)], Alg):
> P2:=convert(P2,list):
> l1:=[x=x1,y=y1,z=z1,
>  Dx=Dx1,Dy=Dy1,Dz=Dz1]:
> l2:=[x=x2,y=y2,z=z2,
>  Dx=Dx2,Dy=Dy2,Dz=Dz2]:
```

```
> # construct covariance
> # apply from one side
> Kf:=convert(
>  map(
>   b->subs(
>    [xi(x1,y1,z1)=b[1]],
>    subs(l1,P2)),
>   convert(
>    Kg(x1,y1,z1,x2,y2,z2),
>    listlist)),
>  Matrix):
> # apply from other side
> Kf:=convert(
>  expand(
>   map(
>    b->subs(
>     [xi(x2,y2,z2)=b[1]],
>     subs(l2,P2)),
>    convert(
>     Transpose(Kf),
>     listlist))),
>  Matrix):
> gp:=unapply(
>  evalf(convert(
>   GP(Kf,
>    [[1,0,0],[-1,0,0]],
>    «0>|<0>|<1>|<0>|<0>|<1»,
>    1e-5),
>   list)),
>  (x,y,z)):
> gp(x,y,z):
> factor(simplify(%));
```

$$
\begin{aligned}
&0.7015\\
&[\,z\left(-\,\mathrm{e}^{x-0.5\,x^2-0.5\,y^2-0.5\,z^2}+\mathrm{e}^{-x-0.5\,x^2-0.5\,y^2-0.5\,z^2}\right),\\
&yz\left(\mathrm{e}^{x-0.5\,x^2-0.5\,y^2-0.5\,z^2}+\mathrm{e}^{-1.0\,x-0.5\,x^2-0.5\,y^2-0.5\,z^2}\right),\\
&-\,y^2\mathrm{e}^{x-0.5\,x^2-0.5\,y^2-0.5\,z^2}+x\mathrm{e}^{x-0.5\,x^2-0.5\,y^2-0.5\,z^2}\\
&-\,y^2\mathrm{e}^{-x-0.5\,x^2-0.5\,y^2-0.5\,z^2}-x\mathrm{e}^{-x-0.5\,x^2-0.5\,y^2-0.5\,z^2}]
\end{aligned}
$$

Example C.3 (Code for Example Code for Example 4.1)**.**

```
> restart;with(LinearAlgebra):
> k:=(x,y)->exp(-1/2*(x-y)^2);
```

$$k := (x, y) \mapsto \mathrm{e}^{-1/2\,(x-y)^2}$$

```
> K:=<
>  <k(0,0),subs(x=0,diff(k(x,0),x)),
>   k(1,0),subs(x=1,diff(k(x,0),x))>|
>  <subs(y=0,diff(k(0,y),y)),
>   subs([x=0,y=0],diff(k(x,y),x,y)),
>   subs(y=0,diff(k(1,y),y)),
>   subs([x=1,y=0],diff(k(x,y),x,y))>|
>  <k(0,1),subs(x=0,diff(k(x,1),x)),
>   k(1,1),subs(x=1,diff(k(x,1),x))>|
>  <subs(y=1,diff(k(0,y),y)),
>   subs([x=0,y=1],diff(k(x,y),x,y)),
>   subs(y=1,diff(k(1,y),y)),
>   subs([x=1,y=1],diff(k(x,y),x,y))>
> >:
> K:=simplify(K);
```

$$K := \begin{bmatrix} 1 & 0 & \mathrm{e}^{-1/2} & -\mathrm{e}^{-1/2} \\ 0 & 1 & \mathrm{e}^{-1/2} & 0 \\ \mathrm{e}^{-1/2} & \mathrm{e}^{-1/2} & 1 & 0 \\ -\mathrm{e}^{-1/2} & 0 & 0 & 1 \end{bmatrix}$$

```
> # posterior covariance
> K_star:=unapply(
> «k(x,0)>|
>  <subs(y=0,diff(k(x,y),y))>|
>  <k(x,1)>|
>  <subs(y=1,diff(k(x,y),y))»,x):
> K_inv:=simplify(K^(-1)):
> d:=denom(K_inv[1,1]):
> K_inv_d:=simplify(d*K_inv):
> 1/d*simplify(
>  («d*k(x,y)»
>   -K_star(x).K_inv_d.
>   Transpose(K_star(y)))[1,1]
> );
```

$$\mathrm{e}^{-\frac{1}{2}(x-y)^2} - \frac{\mathrm{e}^{-\frac{1}{2}x^2-\frac{1}{2}y^2}}{\mathrm{e}^{-2} - 3\,\mathrm{e}^{-1} + 1}.$$

$$\Big((xy-x-y+2)\mathrm{e}^{x+y-1}+(xy+1) \\ +(-2xy+x+y-1)\left(\mathrm{e}^{x+y-2}+\mathrm{e}^{-1}\right) \\ +(xy-y+1)\mathrm{e}^{y-2}+(xy-x+1)\mathrm{e}^{x-2} \\ +(y-x-2)\mathrm{e}^{y-1}+(x-y-2)\mathrm{e}^{x-1}\Big),$$

Example C.4 (Code for Example 5.3)**.**

```
> restart;
> with(OreModules):
> with(LinearAlgebra):
> Alg:=DefineOreAlgebra(
>  diff=[Dx,x], diff=[Dy,y],
>  diff=[Dz,z], polynom=[x,y,z]):
> A1:=«Dx>|<Dy>|<Dz»;
```

$$A1 := \begin{bmatrix} Dx & Dy & Dz \end{bmatrix}$$

```
> B1:=Involution(
>  SyzygyModule(
>   Involution(A1,Alg),
>   Alg),
>  Alg):
> # reorder columns
> B1:=B1.«0,0,-1>|<1,0,0>|<0,-1,0»;
```

$$B1 := \begin{bmatrix} Dz & Dy & 0 \\ 0 & -Dx & Dz \\ -Dx & 0 & -Dy \end{bmatrix}$$

```
> A2:=«x>|<y>|<z»;
```

$$A2 := \begin{bmatrix} x & y & z \end{bmatrix}$$

```
> B2:=Involution(
>  SyzygyModule(
>   Involution(A2,Alg),
>   Alg),
>  Alg):
> # reorder columns
> B2:=B2.«0,0,1>|<-1,0,0>|<0,1,0»;
```

$$B2 := \begin{bmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{bmatrix}$$

```
> MinimalParametrizations(<B1|B2>,Alg):
> C:=%[2]:
> # Normalize Columns
> C:=C.DiagonalMatrix([-1,-1,1]);
```

$$C := \begin{bmatrix} x & Dx & 0 \\ y & Dy & 0 \\ z & Dz & 0 \\ Dx & 0 & x \\ Dy & 0 & y \\ Dz & 0 & z \end{bmatrix}$$

```
> #check, parametrization:
> BB:=SyzygyModule(C,Alg):
> ReduceMatrix(BB,<B1|B2>,Alg);
> ReduceMatrix(<B1|B2>,BB,Alg);
```

$$[]$$
$$[]$$

```
> # B1*C1
> BB1:=Mult(B1,C[1..3,1..3],Alg);
```

$$BB1 := \begin{bmatrix} -zDy + Dz\,y & 0 & 0 \\ Dx\,z - Dz\,x & 0 & 0 \\ -Dx\,y + Dy\,x & 0 & 0 \end{bmatrix}$$

```
> # -B2*C2
> BB2:=-Mult(B2,C[4..6,1..3],Alg);
```

$$BB2 := \begin{bmatrix} -zDy + Dz\,y & 0 & 0 \\ Dx\,z - Dz\,x & 0 & 0 \\ -Dx\,y + Dy\,x & 0 & 0 \end{bmatrix}$$

```
> #For comparison:
> B_old:=«y*Dz-z*Dy,
>   -x*Dz+z*Dx,-y*Dx+x*Dy»;
```

$$B\_old := \begin{bmatrix} -zDy + Dz\,y \\ Dx\,z - Dz\,x \\ -Dx\,y + Dy\,x \end{bmatrix}$$

Example C.5 (Code for Example 5.4).

```
> restart;
> with(Janet):
```

```
> with(OreModules):
> with(LinearAlgebra):
> with(plots):
> Alg:=DefineOreAlgebra(diff=[Dx,x],
>  diff=[Dy,y], diff=[Dz,z],
>  diff=[Dx1,x1], diff=[Dy1,y1],
>  diff=[Dz1,z1], diff=[Dx2,x2],
>  diff=[Dy2,y2], diff=[Dz2,z2],
>  polynom=[x,y,z,x1,x2,y1,y2,z1,z2]):
> # div-free fields on S^2
> B1:=«y*Dz-z*Dy,-x*Dz+z*Dx,-y*Dx+x*Dy»:
> # parametrize equator=0
> B2:=DiagonalMatrix([z$3]);
```

$$B2 := \begin{bmatrix} z & 0 & 0 \\ 0 & z & 0 \\ 0 & 0 & z \end{bmatrix}$$

```
> # combine
> B:=<B1|B2>:
> C:=Involution(
>  SyzygyModule(
>   Involution(B,Alg),
>   Alg),
>  Alg);
```

$$C := \begin{bmatrix} z^2 \\ Dy\, z^2 - Dz\, yz - 2\, y \\ -Dx\, z^2 + x Dz\, z + 2\, x \\ Dx\, yz - Dy\, xz \end{bmatrix}$$

```
> # check parametrization
> BB:=SyzygyModule(C,Alg):
> ReduceMatrix(B,BB,Alg);
```

$$[]$$

```
> ReduceMatrix(BB,B,Alg);
> # new relation!
```

$$\begin{bmatrix} 0 & x & y & z \end{bmatrix}$$

```
> # the new parametrization
> P:=Mult(B1,[[C[1,1]]],Alg);
```

$$P := \left[\begin{array}{c} z\left(-Dy\,z^2 + Dz\,yz + 2\,y\right) \\ z\left(Dx\,z^2 - xDz\,z - 2\,x\right) \\ \left(-Dx\,y + Dy\,x\right)z^2 \end{array}\right]$$

```
> # sanity check = P
> -Mult(B2,C[2..4,1..1],Alg);
```

$$\left[\begin{array}{c} z\left(-Dy\,z^2 + Dz\,yz + 2\,y\right) \\ z\left(Dx\,z^2 - xDz\,z - 2\,x\right) \\ \left(-Dx\,y + Dy\,x\right)z^2 \end{array}\right]$$

```
> # covariance for
> # parametrizing function
> SE:=exp(-1/2*(x1-x2)^2
>  -1/2*(y1-y2)^2-1/2*(z1-z2)^2):
> Kg:=unapply(
>  DiagonalMatrix([SE]),
>  (x1,y1,z1,x2,y2,z2)):
> # prepare covariance
> P2:=ApplyMatrix(P,
>  [xi(x,y,z)], Alg):
> P2:=convert(P2,list):
> l1:=[x=x1,y=y1,z=z1,
>  Dx=Dx1,Dy=Dy1,Dz=Dz1]:
> l2:=[x=x2,y=y2,z=z2,
>  Dx=Dx2,Dy=Dy2,Dz=Dz2]:
> # construct covariance
> # apply from one side
> Kf:=convert(
>  map(
>   b->subs(
>    [xi(x1,y1,z1)=b[1]],
>    subs(l1,P2)),
>   convert(
>    Kg(x1,y1,z1,x2,y2,z2),
>    listlist)),
>  Matrix):
```

```
> # apply from other side
> Kf:=convert(
>  expand(
>   map(
>    b->subs(
>     [xi(x2,y2,z2)=b[1]],
>     subs(l2,P2)),
>    convert(
>     Transpose(Kf),
>     listlist))),
>  Matrix):
> gp:=unapply(
>  piecewise(z<0,[0,0,0],
>  evalf(convert(
>   GP(Kf,[[0,0,1]],<1|0|0>,1e-5),
>   list))),
>  (x,y,z)):
> gp(x,y,z) assuming z>0:
> factor(simplify(%));
```

$$[-0.6065\, z(-z^2+zy^2+2y^2)\, \mathrm{e}^{-0.5\, x^2-0.5\, y^2+z-0.5\, z^2},\\ 0.6065\, xyz(z+2)\, \mathrm{e}^{-0.5\, x^2-0.5\, y^2+z-0.5\, z^2},\\ -0.6065\, xz^2\, \mathrm{e}^{-0.5\, x^2-0.5\, y^2+z-0.5\, z^2}]$$

Example C.6 (Code for Example 6.1).

```
> restart;
> with(OreModules):
> with(LinearAlgebra):
> Alg:=DefineOreAlgebra(diff=[Dx,x],
>  diff=[Dy,y], diff=[Dz,z],
>  diff=[Dx1,x1], diff=[Dy1,y1],
>  diff=[Dz1,z1], diff=[Dx2,x2],
>  diff=[Dy2,y2], diff=[Dz2,z2],
>  polynom=[x,y,z,x1,x2,y1,y2,z1,z2]):
> B1:=«y*Dz-z*Dy,-x*Dz+z*Dx,-y*Dx+x*Dy»;
```

$$B1 := \begin{bmatrix} -zDy + Dz\, y \\ zDx - Dz\, x \\ -Dx\, y + Dy\, x \end{bmatrix}$$

```
> mu:=<0,-z,y>;
```

$$\mu := \begin{bmatrix} 0 \\ -z \\ y \end{bmatrix}$$

```
> #check:
> A1:=Matrix(1,3,[[Dx,Dy,Dz]]):
> A2:=Matrix(1,3,[[x,y,z]]):
> ApplyMatrix(A1,mu,Alg);
> ApplyMatrix(A2,mu,Alg);
```

$$[0]$$

$$[0]$$

```
> # the new parametrization
> P:=Mult(B1,[[z^2]],Alg);
```

$$P := \begin{bmatrix} z\left(-Dy\,z^2 + Dz\,yz + 2\,y\right) \\ z\left(Dx\,z^2 - xDz\,z - 2\,x\right) \\ \left(-Dx\,y + Dy\,x\right)z^2 \end{bmatrix}$$

```
> # covariance for
> # parametrizing function
> SE:=exp(-1/2*(x1-x2)^2
>  -1/2*(y1-y2)^2-1/2*(z1-z2)^2):

> Kg:=unapply(
>  DiagonalMatrix([SE]),
>  (x1,y1,z1,x2,y2,z2)):
> # prepare covariance
> P2:=ApplyMatrix(P,
>  [xi(x,y,z)], Alg):
> P2:=convert(P2,list):

> l1:=[x=x1,y=y1,z=z1,
>  Dx=Dx1,Dy=Dy1,Dz=Dz1]:

> l2:=[x=x2,y=y2,z=z2,
>  Dx=Dx2,Dy=Dy2,Dz=Dz2]:
```

```
> # construct covariance
> # apply from one side
> Kf:=convert(
>  map(
>   b->subs(
>     [xi(x1,y1,z1)=b[1]],
>    subs(l1,P2)),
>   convert(
>    Kg(x1,y1,z1,x2,y2,z2),
>    listlist)),
>  Matrix):
> # apply from other side
> Kf:=convert(
>  expand(
>   map(
>    b->subs(
>     [xi(x2,y2,z2)=b[1]],
>     subs(l2,P2)),
>    convert(
>     Transpose(Kf),
>     listlist))),
>  Matrix):
> p:=[0,0,1]:
> mu_p:=Transpose(
>  subs(
>   [x=p[1],y=p[2],z=p[3]],
>   mu)):
> gp:=unapply(
>  factor(simplify(
>   convert(
>    GP(Kf,[p],<1|0|0>-mu_p,1e-5),
>    list)))
>   +convert(mu,list),
>  (x,y,z)):
> gp(x,y,z);
```

$$[-0.6065\, z(-z^2 + zy^2 + 2y^2)\, \mathrm{e}^{-0.5\, x^2 - 0.5\, y^2 + z - 0.5\, z^2},$$
$$0.6065\, xyz(z+2)\, \mathrm{e}^{-0.5\, x^2 - 0.5\, y^2 + z - 0.5\, z^2} - z,$$
$$-0.6065\, xz^2\, \mathrm{e}^{-0.5\, x^2 - 0.5\, y^2 + z - 0.5\, z^2} + y]$$

Example C.7 (Code for Example 6.2).

```
> restart;
> with(OreModules):
> with(LinearAlgebra):
> Alg:=DefineOreAlgebra(
>  diff=[Dx,x], diff=[Dy,y],
>  diff=[Dx1,x1], diff=[Dy1,y1],
>  diff=[Dx2,x2], diff=[Dy2,y2],
>  polynom=[x,y,x1,x2,y1,y2]):
> A:=«Dx>|<Dy»;
```

$$\begin{bmatrix} Dx & Dy \end{bmatrix}$$

```
> B1:=Involution(
>  SyzygyModule(
>   Involution(A,Alg),
>   Alg),
>  Alg);
```

$$\begin{bmatrix} Dy \\ -Dx \end{bmatrix}$$

```
> mu:=<1,0>;
```

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

```
> B2:=«(x-1)*x,0>|<0,(y-1)*y»;
```

$$\begin{bmatrix} (x-1)\,x & 0 \\ 0 & (y-1)\,y \end{bmatrix}$$

```
> # combine
> B:=<B1|B2>:
> C:=Involution(
>  SyzygyModule(
>   Involution(B,Alg),
>   Alg),
>  Alg);
```

$$\begin{bmatrix} x^2y^2 - x^2y - xy^2 + xy \\ -Dy\,y^2 + Dy\,y - 2\,y + 1 \\ Dx\,x^2 - Dx\,x + 2\,x - 1 \end{bmatrix}$$

```
> # the new parametrization
> P:=Mult(B1,C[1,1],Alg);
```

$$\begin{bmatrix} x\left(-1 + Dy\,y^2 + (-Dy+2)\,y\right)(x-1) \\ -\,(y-1)\,y\left(-1 + Dx\,x^2 + (-Dx+2)\,x\right) \end{bmatrix}$$

```
> # covariance for
> # parametrizing function
> SE:=exp(-1/2*(x1-x2)^2
> -1/2*(y1-y2)^2):

> Kg:=unapply(
> DiagonalMatrix([SE]),
> (x1,y1,x2,y2)):

> # prepare covariance
> P2:=ApplyMatrix(P,
> [xi(x,y)], Alg):
> P2:=convert(P2,list):

> l1:=[x=x1,y=y1,
> Dx=Dx1,Dy=Dy1]:

> l2:=[x=x2,y=y2,
> Dx=Dx2,Dy=Dy2]:

> # construct covariance
> # apply from one side
> Kf:=convert(
>  map(
>   b->subs(
>    [xi(x1,y1)=b[1]],
>    subs(l1,P2)),
>   convert(
>    Kg(x1,y1,x2,y2),
>    listlist)),
>  Matrix):
> # apply from other side
> Kf:=convert(
>  expand(
>   map(
>    b->subs(
>     [xi(x2,y2)=b[1]],
>     subs(l2,P2)),
>    convert(
>     Transpose(Kf),
>     listlist))),
>  Matrix):
```

```
> # code for GP regression
> GP:=proc(Kf,
>  points,yy,epsilon)
> local n,m,kf,K,s1,s2,alpha,KStar;
>  n:=nops(points);
>  m:=RowDimension(Kf);
>  s1:=map(
>   a->[x1=a[1],y1=a[2]],
>   points);
>  s2:=map(
>   a->[x2=a[1],y2=a[2]],
>   points);
>  kf:=convert(Kf,listlist);
>  K:=convert(
>   evalf(
>    map(
>     a->map(
>      b->convert(
>       subs(a,subs(b,kf)),
>       Matrix),
>      s2),
>    s1)),
>   Matrix):
>  alpha:=yy.(K+epsilon^2)^(-1);
>  KStar:=map(
>   a->subs(a,kf),
>   s1):
>  KStar:=subs(
>   [x2=x,y2=y],KStar):
>  KStar:=convert(
>   map(op,KStar),Matrix):
>  return alpha.KStar;
> end:
```

```
> p:=[1/2,1/2]:
> mu_p:=Transpose(
> subs(
>   [x=p[1],y=p[2]],
>   mu)):
> gp:=unapply(
>   factor(simplify(
>    convert(
>     GP(Kf,[p],<0|1>-mu_p,1e-5),
>     list)))
>   +convert(mu,list),
>   (x,y));
```

$$(x,y) \mapsto \mathrm{e}^{-0.25+0.5x-0.5x^2+0.5y-0.5y^2} \cdot \\ [1+16x\left(y^4(x-1)+y^3(x-2.5)(x-1)+y^2(0.5x+1-1.5x^2)-y(x-1)(x-2.33)+(x-1)^2\right), \\ -16y\left(x^4(y-1)+x^3(y-2.5)(y-1)+x^2(0.5y+1-1.5y^2)-x(y-1)(y-2.33)+(y-1)^2\right)]$$

References

Barakat, M. and Lange-Hegermann, M. (2011). An axiomatic setup for algorithmic homological algebra and an alternative approach to localization. *J. Algebra Appl.*, 10(2):269–293.

Cartan, H. and Eilenberg, S. (1999). *Homological algebra*. Princeton Landmarks in Mathematics. Princeton University Press, Princeton, NJ. With an appendix by David A. Buchsbaum, Reprint of the 1956 original.

Chyzak, F., Quadrat, A., and Robertz, D. (2007). OreModules: a Symbolic Package for the Study of Multidimensional Linear Systems. In *Applications of time delay systems*, volume 352 of *Lecture Notes in Control and Inform. Sci.*, pages 233–264. Springer, Berlin.

Eisenbud, D. (1995). *Commutative Algebra with a View Toward Algebraic Geometry*, volume 150 of *Graduate Texts in Mathematics*. Springer-Verlag.

Mac Lane, S. (1998). *Categories for the working mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, second edition.

nLab authors (2020). The *n*lab. (`http://ncatlab.org/nlab/`) [Online; accessed 12-April-2020].

Weibel, C. A. (1994). *An introduction to homological algebra*. Cambridge Studies in Advanced Mathematics. Cambridge University Press.