We study the control system of a wind tunnel model. See A. Manitius, *Feedback controllers for a wind tunnel model involving a delay: analyical design and numerical simulations*, IEEE Trans. Autom. Contr. vol. 29 (1984), pp. 1058-1068.

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

We define the Ore algebra *Alg* where *Dt* acts as differentiation w.r.t. time t and δ acts as a time-delay operator of length h. Note that the constants a, ω, ζ, k of the system have to be declared in the definition of the algebra. (The syntax of *DefineOreAlgebra* is the same as the syntax of *Ore_algebra*[*skew_algebra*].)

```
>  Alg := DefineOreAlgebra(diff=[Dt,t], diff=[delta,s], polynom=[t,s],
>  comm=[a, omega, zeta, k], shift_action=[delta,t,h]):
```

Define the matrix which corresponds to the linear differential time-delay system:

```
>  R := evalm([[Dt+a, -k*a*delta, 0, 0], [0, Dt, -1, 0],
>  [0, omega^2, Dt+2*zeta*omega, -omega^2]]);
```

$$R := \begin{bmatrix} Dt+a & -k\,a\,\delta & 0 & 0 \\ 0 & Dt & -1 & 0 \\ 0 & \omega^2 & Dt+2\,\zeta\,\omega & -\omega^2 \end{bmatrix}$$

1 Computation of the first extension module "by hand"

In this subsection we demonstrate how to compute the first extension module with values in *Alg* of the module associated with the transposed matrix of R. Note that we are working with the transposed matrix because the system has constant coefficients so that it can actually be defined over the commutative polynomial ring with indeterminate Dt. We follow Section 5 of F. Chyzak, A. Quadrat, D. Robertz, *Effective algorithms for parametrizing linear control systems over Ore algebras*, INRIA report 5181.

```
>  R_adj := linalg[transpose](R);
```

$$R\_adj := \begin{bmatrix} Dt+a & 0 & 0 \\ -k\,a\,\delta & Dt & \omega^2 \\ 0 & -1 & Dt+2\,\zeta\,\omega \\ 0 & 0 & -\omega^2 \end{bmatrix}$$

In order to find the syzygy module of the module N spanned by the rows of *R_adj*, we adjoint second members μ_1, ..., μ_4 to the equations given by the entries of *R_adj* $(\lambda_1 : \lambda_2 : \lambda_3)^T$ and compute a Gröbner basis G w.r.t. an elimination ordering that eliminates the λ_i:

```
>  G := Integrability(R_adj, Alg);
```

$$G := [\omega^2\,Dt\,\mu_2 + \omega^2\,k\,a\,\delta\,\mu_1 + \omega^2\,a\,\mu_2 + \omega^2\,Dt^2\,\mu_3 + \omega^2\,a\,Dt\,\mu_3 + Dt^3\,\mu_4 + 2\,Dt^2\,\zeta\,\omega\,\mu_4 \\ + a\,Dt^2\,\mu_4 + Dt\,\omega^2\,\mu_4 + 2\,a\,Dt\,\zeta\,\omega\,\mu_4 + a\,\omega^2\,\mu_4,\ \lambda_3\,\omega^2 + \mu_4, \\ \omega^2\,\lambda_2 + Dt\,\mu_4 + \omega^2\,\mu_3 + 2\,\zeta\,\omega\,\mu_4, \\ \omega^2\,\lambda_1\,k\,a\,\delta + Dt^2\,\mu_4 + \omega^2\,Dt\,\mu_3 + 2\,Dt\,\zeta\,\omega\,\mu_4 + \omega^2\,\mu_2 + \omega^2\,\mu_4,\ \lambda_1\,Dt + \lambda_1\,a - \mu_1]$$

Now, the syzygy module that we are interested in is generated by all elements in G which do not contain any λ_i. This is only one element. Writing it in the form of a matrix, we find exactly the result of *SyzygyModule*:

```
>  S := SyzygyModule(R_adj, Alg);
```

$$S := \Big[\omega^2\, k\, a\, \delta\,,\ Dt\,\omega^2 + a\,\omega^2\,,\ \omega^2\, Dt^2 + \omega^2\, a\, Dt\,,\ Dt\,\omega^2 + a\,\omega^2 + Dt^3 + 2\, Dt^2\,\zeta\,\omega + a\, Dt^2 + 2\, a\, Dt\,\zeta\,\omega\Big]$$

Hence, we started a free resolution of the module N spanned by the rows of *R_adj*. By definition of the extension module, we have to dualize the modules that we obtained so far, so that matrices are transposed. We compute the syzygy module of the module which we get by dualizing S:

```
>  Integrability(linalg[transpose](S), Alg);
```

$$[-\omega^2\,\mu_4 + Dt\,\mu_3 + \omega^2\,\mu_2 + 2\,\zeta\,\omega\,\mu_3,\ -\mu_3 + Dt\,\mu_2,\ Dt\,\mu_1 - k\,a\,\delta\,\mu_2 + a\,\mu_1,\ \lambda_1\,\omega^2\,k\,a\,\delta - \mu_1,\ \lambda_1\,Dt\,\omega^2 + \lambda_1\,a\,\omega^2 - \mu_2]$$

Analogous to the computation above, the syzygy module is generated by all elements which do not involve any λ_i. By means of *SyzygyModule* we obtain this generating set in one step:

```
>  L := SyzygyModule(linalg[transpose](S), Alg);
```

$$L := \begin{bmatrix} Dt + a & -k\,a\,\delta & 0 & 0 \\ 0 & \omega^2 & Dt + 2\,\zeta\,\omega & -\omega^2 \\ 0 & Dt & -1 & 0 \end{bmatrix}$$

Finally, to obtain the first extension module of N with values in *Alg*, we compute the quotient module of the module generated by the rows of L modulo the module generated by the rows of R:

```
>  Q := Quotient(L, R, Alg);
```

$$Q := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The ith column in Q contains the elements of *Alg* by which the ith row of L is to be multiplied to obtain an element in *Alg*-span of the rows of R. Since Q is the identity matrix, we see that actually all rows of L lie in the *Alg*-span of the rows of R which means that the modules spanned by the rows of the respective matrices are the same. Hence, the first extension module of N with values in *Alg* is zero.

We check controllability and parametrizability of the wind tunnel system. Using an involution of *Alg*, we compute the formal adjoint of R:

```
>  R_adj := Involution(R, Alg);
```

$$R\_adj := \begin{bmatrix} -Dt + a & 0 & 0 \\ k\,a\,\delta & -Dt & \omega^2 \\ 0 & -1 & -Dt + 2\,\zeta\,\omega \\ 0 & 0 & -\omega^2 \end{bmatrix}$$

We compute ext^1 with values in *Alg* of *R_adj* in one step:

```
>  st := time():  Ext1 := Exti(R_adj, Alg, 1); time()-st;
```

$$\mathit{Ext1} := \left[\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} Dt + a & -k\,a\,\delta & 0 & 0 \\ 0 & \omega^2 & Dt + 2\,\zeta\,\omega & -\omega^2 \\ 0 & Dt & -1 & 0 \end{bmatrix}, \right.$$

$$\left. \begin{bmatrix} -\omega^2\,k\,a\,\delta \\ -Dt\,\omega^2 - a\,\omega^2 \\ -\omega^2\,Dt^2 - \omega^2\,a\,Dt \\ -Dt^3 - 2\,Dt^2\,\zeta\,\omega - a\,Dt^2 - Dt\,\omega^2 - 2\,a\,Dt\,\zeta\,\omega - a\,\omega^2 \end{bmatrix} \right]$$

$$0.279$$

Since *Ext1*[1] is the identity matrix, we conclude that ext^1 with values in *Alg* of the module N defined by *R_adj* is the zero module. Therefore, the module M which is associated with the system R is torsion-free. It follows that the system is controllable and, equivalently, parametrizable. A parametrization of R is given in *Ext1*[3]. A necessary condition for *Ext1*[3] being a parametrization is that (R o *Ext1*[3]) = 0:

```
>  map(expand, Mult(R, Ext1[3], Alg));
```

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

We compute ext^2 with values in *Alg* of *R_adj* in order to check whether or not the system R is flat:

```
>  Ext2 := Exti(R_adj, Alg, 2);
```

$$\mathit{Ext2} := [\begin{bmatrix} \delta \\ Dt + a \end{bmatrix}, \begin{bmatrix} 1 \end{bmatrix}, \mathrm{SURJ}(1)]$$

Since the first matrix in *Ext*[2] is not an identity matrix, ext^2 of *R_adj* is not the zero module, and we conclude that the module M which is associated with the system R is torsion-free, but not free. Hence, the system is not flat.

The formal obstructions of flatness are defined by the π-polynomial

```
>  PiPolynomial(R, Alg);
```

$$[\delta,\ Dt + a]$$

Finally, we can find a polynomial π in the variable δ such that the system is π-free.

```
>  pi := PiPolynomial(R, Alg, [delta]);
```

$$\pi := [\delta]$$

By definition of the π-polynomial (see H. Mounier, *Propriétés structurelles des systèmes linéaires à retards: aspects théoriques et pratiques*, PhD Thesis, University of Orsay, France, 1995), this means that if we introduce the time-advance operator in the system of the wind tunnel, then it becomes a flat system. Hence, the module associated with this system is a free module (over the Ore algebra which is obtained by adjoining the advance operator δ^{-1} to *Alg*, and we are going to find a basis for this module below.

Let us remark that the fact that the wind tunnel is not a flat system (without advance operator) is coherent with the fact that the full row-rank matrix R does not admit a right-inverse. We remember that a full row-rank matrix R admits a right-inverse if and only if the module which is associated with it is projective. By the theorem of Quillen-Suslin, for modules over commutative polynomial rings, projectiveness is the same as freeness. This remark applies to our situation.

```
>  RightInverse(R, Alg);
```

$$[]$$

The fact that the system is not flat is also coherent with the fact that its parametrization *Ext1*[3] does not admit a left-inverse. Indeed, a linear system is flat if and only if it is parametrizable and one of its parametrization admits a left-inverse.

```
>  LeftInverse(Ext1[3], Alg);
```

$$[]$$

We finish by computing a basis of the free module M_2 which is associated to the wind tunnel system, when introducing the time-advance operator. In the terminology of control, such a basis is called a *flat output*. We apply *LocalLeftInverse* to the parametrization *Ext1*[3] of the system and we allow the algorithm to invert δ:

```
>  S := LocalLeftInverse(Ext1[3], [delta], Alg);
```

$$S := \begin{bmatrix} -\frac{1}{\delta\,\omega^2\,k\,a} & 0 & 0 & 0 \end{bmatrix}$$

By construction, the matrix S is a left-inverse of *Ext1*[3]:

```
>  simplify(evalm(S &* Ext1[3]));
```

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

Hence, we obtain a basis $z = S\,(x_1 : x_2 : x_3 : x_4)^T$ of the $Alg[\delta^{-1}]$-module M_2 associated with R because we have $(x_1 : x_2 : x_3 : u)^T = Ext1[3]\ z$ and $z = -\delta^{-1}\,x_1\ /\ (\omega\hat{\ }2\ k\ a)$ is an element of the module.

Let us finally point out that we can also substitute $z = -\delta^{-1}\,x_1\ /\ (\omega\hat{\ }2\ k\ a)$ into the parametrization *Ext1*[3] of the system in order to express the system variables in terms of x_1:

```
>  F := map(factor, simplify(evalm(Ext1[3] * S[1,1])));
```

$$F := \begin{bmatrix} 1 \\ \frac{Dt + a}{k\,a\,\delta} \\ \frac{Dt\,(Dt + a)}{k\,a\,\delta} \\ \frac{(\omega^2 + Dt^2 + 2\,\zeta\,\omega\,Dt)\,(Dt + a)}{\omega^2\,k\,a\,\delta} \end{bmatrix}$$

We obtain $(x_1 : x_2 : x_3 : u)^T = F\ x_1$, and thus, x_1 is also a basis of the $Alg[\delta^{-1}]$-module M_2, and thus, a flat output of the wind tunnel system over $Alg[\delta^{-1}]$. From the last component of F, we obtain that the input $\mathrm{u}(t-h)$ of the system is defined by:

```
>  v := ApplyMatrix(delta*F[4,1], [x1(t)], Alg)[1];
```

$$v := \frac{\mathrm{x1}(t)}{k} + \frac{(\omega^2 + 2\,a\,\zeta\,\omega)\,(\frac{d}{dt}\,\mathrm{x1}(t))}{\omega^2\,k\,a} + \frac{\frac{d^3}{dt^3}\,\mathrm{x1}(t)}{\omega^2\,k\,a} + \frac{(2\,\zeta\,\omega + a)\,(\frac{d^2}{dt^2}\,\mathrm{x1}(t))}{\omega^2\,k\,a}$$

By shifting the time, we obtain that the input $\mathrm{u}(t)$ is defined by:

```
>  w := ApplyMatrix(F[4,1], [x1(t)], Alg)[1];
```

$$w := \frac{\mathrm{x1}(t+h)}{k} + \frac{(\mathrm{D}^{(3)})(\mathit{x1})(t+h)}{\omega^2\,k\,a} + \frac{(2\,\zeta\,\omega + a)\,(\mathrm{D}^{(2)})(\mathit{x1})(t+h)}{\omega^2\,k\,a}$$
$$+ \frac{(\omega^2 + 2\,\zeta\,\omega\,a)\,\mathrm{D}(\mathit{x1})(t+h)}{\omega^2\,k\,a}$$

Finally, the input $u : t \mapsto w(t)$ is defined by:

```
>  u := t -> w;
```

$$u := t \to w$$

Hence, by imposing a certain desired trajectory *x1_ ref*, we finally obtain an open-loop control $u : t \mapsto u(t)$ which follows the desired trajectory *x1_ ref*. Let us consider the numerical values of the parameters given in A. Manitius, *Feedback controllers for a wind tunnel model involving a delay: analyical design and numerical simulations*, IEEE Trans. Autom. Contr. vol. 29 (1984), pp. 1058-1068.

```
>  a := 1/1.964:  k := -0.0117:  h := 0.32:  zeta := 0.8:  omega := 6:
>  eval(u(t));
```

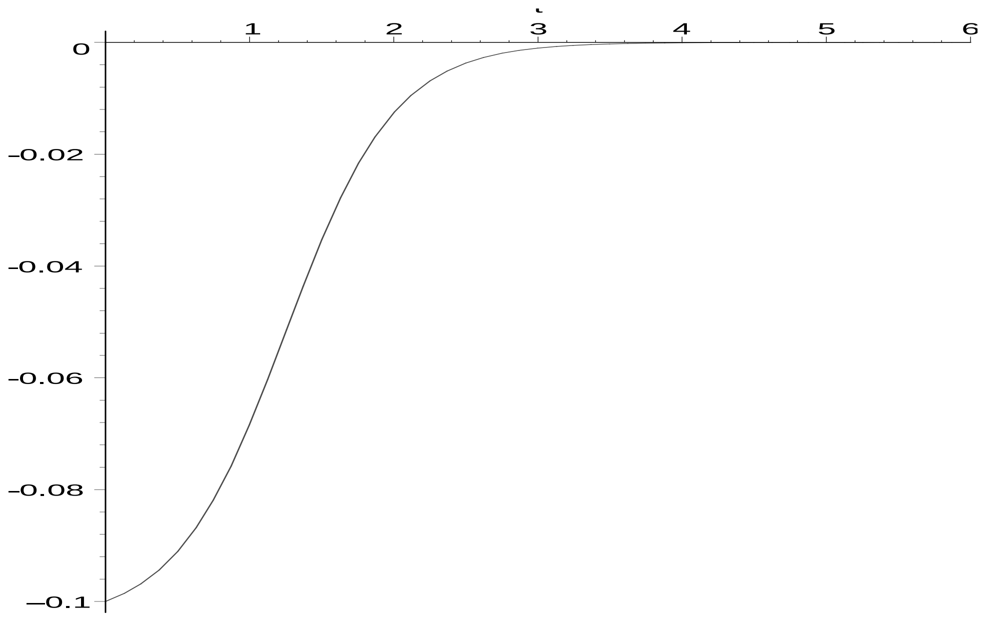
$$-85.47008547\,\mathrm{x1}(t+0.32) - 4.662867997\,(\mathrm{D}^{(3)})(\mathit{x1})(t+0.32)$$
$$-\,47.13770181\,(\mathrm{D}^{(2)})(\mathit{x1})(t+0.32) - 190.6552706\,\mathrm{D}(\mathit{x1})(t+0.32)$$

As in the paper (A. Manitius, *Feedback controllers for a wind tunnel model involving a delay: analyical design and numerical simulations*, IEEE Trans. Autom. Contr. vol. 29 (1984), pp. 1058-1068), let us try to go from the initial condition *x1*(0) = -0.1 to *x1*(6) = 0.

```
>  f := t -> tanh(1.32*(t-1.25));
```

$$f := t \to \tanh(1.32\,t - 1.6500)$$

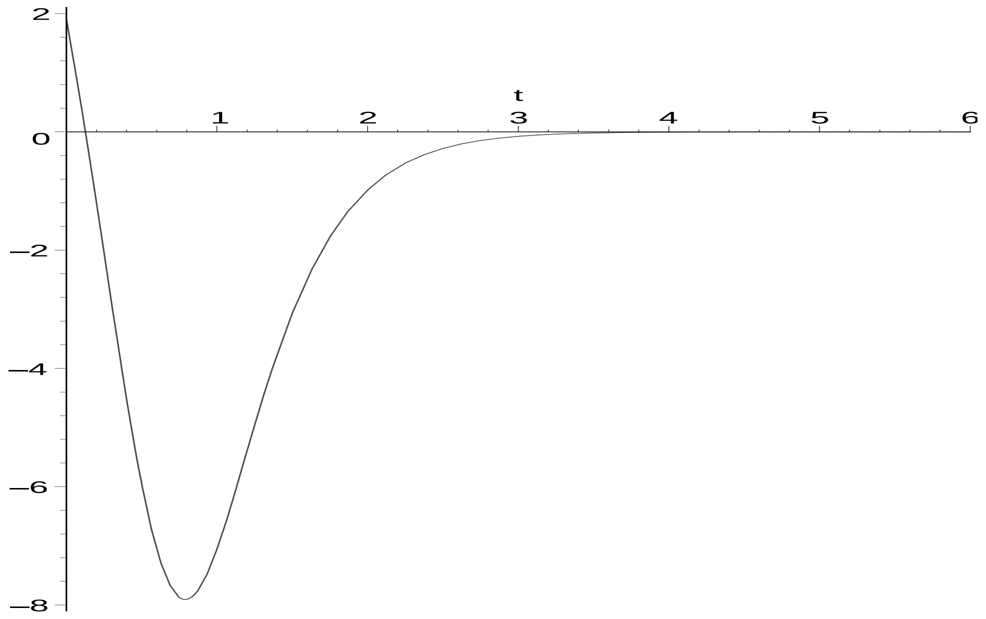
```
>  x1 := t -> -0.1*(1-f(t))/(1-f(0)):
>  plot(x1(t), t=0..6);
```



```
>  simplify(evalf(u(t)));
```

$$-7.504233378 + 4.085080047\,\%1 + 8.599358396\,\%1^2 + 3.335999628\,\%1^4 - 8.516204693\,\%1^3$$
$$\%1 := \tanh(1.320000000\,t - 1.227600000)$$

```
>  plot(u(t), t=0..6);
```



Compare with the simulation obtained in A. Manitius, *Feedback controllers for a wind tunnel model involving a delay: analyical design and numerical simulations*, IEEE Trans. Autom. Contr. vol. 29 (1984), pp. 1058-1068.