Numerical investigation of structural minimality for structures of uncontrolled linear switching systems with Maple

OR, a subplot that has taken over the story:

"Maple's 'fsolve' applied to root finding in sums of exponentials, for good or ill (-conditioned?)"

Maple Conference 2021 (online)

Jason M. Whyte (jason.whyte@unimelb.edu.au)

Australian Research Council Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS), School of Mathematics and Statistics, and Centre of Environmental and Economic Research (CEER), School of BioSciences, University of Melbourne, Victoria, Australia

Last update: 12th October 2021

Abstract:

One path to understanding a physical system is to represent it by a model structure (collection of related models). Suppose our system is not subject to external influences, and depends on unobservable state variables ($x$), and observables ($y$). Then, a suitable uncontrolled, state-space model structure $S$ is defined by relationships between $x$ and $y$, involving parameters $\theta \in \Theta$. That is, each parameter vector in parameter space $\Theta$ is associated with a particular model in $S$.

Before using $S$ for prediction, we require system observations for parameter estimation. This process aims to determine $\theta$ values for which predictions “best” approximate the data (according to some objective function). The result is some number of estimates of the true parameter vector, $\theta^*$. Multiple parameter estimates are problematic when these cause $S$ to produce dissimilar predictions beyond our data's range. This can render us unable to confidently make predictions, resulting in an uninformative study.

Non-uniqueness of parameter estimates follows when $S$ lacks the property of structural global identifiability (SGI). Fortunately, we may test $S$ for SGI prior to data collection. The absence of SGI encourages us to rethink our experimental design or model structure.

Before testing $S$ for SGI we should check that it is structurally minimal. If so, we cannot replace $S$ by a structure of fewer state variables which produces the same output.

Most testing methodology is applicable to structures which employ the same equations for all time. These methods are not appropriate when, for example, a process has an abrupt change in its dynamics. For such a situation, a structure of linear switching systems (LSSs) may be suitable. The structure has a collection of linear time-invariant state-space systems, and a switching function which determines the system in effect at each instant. As such, we face a novel challenge in testing an LSS structure for SGI.

We will consider the case of an uncontrolled LSS structure of one switching event (a ULSS-1 structure). In this setting, we may approach the structural minimality problem via the Laplace transform of the output function on each time interval. Each rational function yields conditions for pole-zero cancellation. If these conditions are not satisfied for almost all $\theta \in \Theta$, then $S$ is structurally minimal.

Analytical approaches can be quite laborious. However, a numerical approach may quickly provide useful insights. For example, if pole-zero cancellation occurs for almost all of a sufficiently large number of parameter values sampled from $\Theta$, then structural minimality is possible. This result may encourage us to prove the existence of structural minimality.

We shall use Maple 2020-2 to conduct a numerical investigation of structural minimality for a test case ULSS-1 structure applicable to flow-cell biosensor experiments used to monitor biochemical interactions.

```maple
restart(); with(LinearAlgebra): with(plots): with(Statistics):
kernelopts(maxdigits);
kernelopts(version);
```

38654705646

Maple 2020.2, APPLE UNIVERSAL OSX, Nov 11 2020, Build ID 1502365
1 Preliminaries

Much of the preliminaries is drawn from (or adapted from) my earlier works, especially:
Whyte, J.M. "Global A Priori Identifiability of Models of Flow-Cell Optical Biosensor Experiments", PhD thesis, School of Mathematics and Statistics, University of Melbourne (2016).
This work follows from an early consideration of the problem in my thesis (Appendix F) which used Maple 2015.2 on a Linux machine.
I understand that fsolve was changed in Maple 2018.

An Introduction to State-Space Model Structures

For brevity, we shall omit many details that will not adversely affect the consideration of our motivating problem.

A controlled state-space structure is a collection of mathematical systems having particular properties which relate a state vector $x$, and possibly, input $u$, to output $y$ on some time set $T \subseteq \mathbb{R}^+$. State variables evolve over time following some ordinary differential equation system, and the output(s) depends on some function of state variables and input(s).

It is useful to demonstrate features of models in structure $S$ by showing the model in $S$ corresponding to some unspecified parameter $\theta \in \Theta$, the "representative system $\tilde{S}(\theta)$" which has the form (noting that $\dot{x} = \frac{d}{dt}(x)$):

\[
\begin{align*}
\dot{x} &= f(x, u, t; \theta), \\
x(x, u, 0; \theta) &= x_0(\theta), \\
y &= g(x, u, t; \theta).
\end{align*}
\]

At any time $t \in T$:
- the state vector belongs to some state space: $x(t) \in X \subseteq \mathbb{R}^n$,
- the input vector belongs to some input space: $u(t) \in U \subseteq \mathbb{R}^k$,
- the output vector belongs to some output space: $y(t) \in Y \subseteq \mathbb{R}^m$.

A variant of a controlled state-space structure that lacks any inputs is an uncontrolled state-space structure.

Key Definitions: Types of Structures/Systems

A controlled linear time-invariant (LTI) state-space structure is a structure where coefficients in ODEs and the output equation are fixed for all time:

\[
\begin{align*}
\dot{x}(x, u, t; \theta) &= A(\theta) \cdot x(x, u, t; \theta) + B(\theta) \cdot u(t), \\
x(x, u, 0; \theta) &= x_0(\theta), \\
y(x, u, t; \theta) &= C(\theta) \cdot x(x, u, t; \theta).
\end{align*}
\]

A subclass is the uncontrolled LTI state-space structure, having representative system of the form:

\[
\begin{align*}
\dot{x}(x, t; \theta) &= A(\theta) \cdot x(x, t; \theta), \\
x(x, 0; \theta) &= x_0(\theta), \\
y(x, t; \theta) &= C(\theta) \cdot x(x, t; \theta).
\end{align*}
\]

In physical applications we tend to consider non-negative quantities (e.g. mass, concentration), and so a subclass of the above is appropriate.

A positive LTI system has $X \subseteq \mathbb{R}^n_+$, $U \subseteq \mathbb{R}^k_+$, $Y \subseteq \mathbb{R}^m_+$.

A compartmental LTI system is a positive system which also satisfies conservation of mass conditions: All elements of $B$ and $C$ are non-negative;
Regarding $A$:

$$
\begin{align*}
 a_{i,j} & > 0 \quad i \in \{1, \ldots, n\}, i \neq j, \\
 a_{i,j} & \leq -\sum_{j=1, j \neq i}^{n} a_{j,j} \quad i \in \{1, \ldots, n\}.
\end{align*}
$$

An Overview of Structural Global Identifiability (SGI)

A structural property of a model structure is one that holds "almost everywhere" in the parameter space (possibly excepting sets of measure zero).

In introducing the testing of a structure for SGI, we note that there are alternatives: we show how favourable these are using a "traffic light" colour scheme.

For simplicity, we start with a consideration of uncontrolled structures.

**Definition 1.** Suppose we have (uncontrolled) model structure $M$ having parameter set $\Theta$ where models in $M$ are defined on time set $T \subseteq [0, \infty)$. We illustrate $M$ by the "representative model" $M(\theta)$ for which we use unspecified $\theta \in \Theta$, output function $y(t; \theta)$ $(t \in T)$. For some finite time $\tau > 0$, we consider the set

$$
I(M) = \left\{ \theta \in \Theta : y(t; \theta^*) = y(t; \theta) \forall t \in [0, \tau] \right\}.
$$

If, for almost all $\theta \in \Theta$:

1. $I(M) = \{\theta\}$, $M$ is structurally globally identifiable;
2. the elements of $I(M)$ are denumerable, $M$ is structurally locally identifiable;
3. the elements of $I(M)$ are not denumerable, $M$ is structurally unidentifiable;

We note that Definition 1 is not appropriate in all cases, e.g. when the initial state is an equilibrium point and response is atypically uninformative as it is constant for all time.

Definition 1 is not so easy to work with in practice. However, we have an alternative.

In general (but not always!) the output of system $M(\theta)$ features "observational parameters" (or "invariants") $\phi(\theta)$ which define the time course of output. This allows us to rewrite Definition 1 in a more useful form.

**Definition 2.** For some arbitrary $\theta \in \Theta$, define the set

$$
I(M, \phi) = \left\{ \theta' \in \Theta : \phi(\theta') = \phi(\theta) \right\} \equiv I(M).
$$

It follows that determination of $I(M, \phi)$, allows classification of $M$ according to Definition 1.

The algebraic conditions of Definition 2 are easier to address than the functional relationship of Definition 1.

In testing a structure for SGI, we may draw on other structural properties.

Either implicitly or explicitly, testing a structure for SGI requires us to consider whether our structure has the property of "structural minimality".

A structure $S$ of $n$ states is structurally minimal if we cannot represent it by some other structure $S'$ which is structurally output equivalent to $S$ and has $n' < n$ states.

Various tests of structures for structural minimality make assumptions that suggest that they are not suited to the analysis of positive structures.

In considering our LTI compartmental structures, we address this matter by choosing a method of testing a structure for SGI that includes an appropriate
method for addressing the issue of structural minimality. We shall outline the method in the next section, and proceed to apply this to an example later in the worksheet.

The Laplace Transform (or Transfer Function) approach to testing an LTI structure for SGI

The Laplace Transform approach is particularly straightforward for an uncontrolled LTI system. Recalling the form of the general response equation $y = Cx$, and the state evaluation equation $\dot{x} = Ax$, then applying the Laplace transform with complex variable $s$:

$$\mathcal{L}(y)[s] = C \mathcal{L}(x)[s] \quad \text{and} \quad \mathcal{L}(\dot{x})[s] = s \mathcal{L}(x)[s] - s x(0) = A \mathcal{L}(x)[s],$$

where all transforms exist as each component of $x$ (and hence $y$) is a sum of exponentials with non-positive exponents as a consequence of the properties of $A$.

Then solving for $\mathcal{L}(x)[s]$ in the second equation above, substituting this into the first, and simplifying, we see that the Laplace transform of $y$ is:

$$\mathcal{L}(y)[s] = C \cdot (sI - A)^{-1} \cdot x_0 = H_2$$

on some domain of convergence for $s$.

The invariants required by our SGI test are held in the matrix $H_2$.

Each element is a rational function in $s$ called a "transfer function", where coefficients of $s$ depend on parameters. We require the "canonical form" of each transfer functions.

Classically, we obtain this by:

1. cancelling any factors common to the denominator and numerator (pole-zero cancellation). [If such cancellation can occur, the structure may not be generically minimal.]
2. ensuring that the leading term of the denominator has the coefficient 1. [We shall allow the user to decide whether or not to enforce this condition.]

Then, the coefficients of $s$ in the processed transfer function contribute to our collection of invariants.

Structures of Uncontrolled Linear Switching Systems of One Transition (ULSS-1 structures)
Definition 3. For some time interval $T$, define the deterministic switching function $\gamma(\cdot)$ by $\gamma : T \rightarrow \Gamma : t \mapsto \gamma(t) \in \Gamma$, $\Gamma \subset \mathbb{N}$.

Consider $\Gamma = \{1, \ldots, q\} \subset \mathbb{N}$ for $q > 1$. Further, consider unspecified parameter vectors $\theta_i \in \Theta_i$, $i = 1, \ldots, q$, combined to give $\theta \triangleq (\theta_1, \ldots, \theta_q)$; a concatenation with redundancy removed. An uncontrolled linear switching system structure of $q$ component systems defined for time interval $T$ is a collection of state-space systems for which state variables $x$ and outputs $y$ at time $t \in T$ are related as shown in the representative system

\[
\dot{x}(t, \theta) = A_{\gamma(t)}(\theta_{\gamma(t)})x(t, \theta), \quad x(0, \theta) = x_0(\theta_1),
\]
\[
y(t, \theta) = C_{\gamma(t)}(\theta_{\gamma(t)})x(t, \theta).
\]

The value of $\gamma$ at any time dictates which of the $q$ component uncontrolled LTI state-space systems is in effect at that time. The $i$-th ($i \in \Gamma$) component system has the form

\[
\dot{x}(t, \theta) = A_i(\theta_i)x(t, \theta),
\]
\[
y(t, \theta) = C_i(\theta_i)x(t, \theta).
\]

Suppose the state and output spaces are $X = \mathbb{R}^n$ and $Y = \mathbb{R}^k$ respectively. Then for $i \in \Gamma$, $A_i \in \mathbb{R}^{n \times n}$ and $C_i \in \mathbb{R}^{k \times n}$. The structure’s representative system (3) has the property that all matrices have elements that are either zero or an expression involving the parameters.

An ULSS structure is characterized by the pattern of non-zero elements appearing in $x_0$ (as in (3)) and each of the $A_i$ and $C_i$ for $i = 1, \ldots, q$ that appear in a representative system of an LTI structure, as in (4).

Definition 4. A ULSS-1 structure is a type of uncontrolled linear switching system structure as in Definition 3 which has two component systems (that is, $q = 2$). Further, the switching function $\gamma(\cdot)$ has $\Gamma = \{1, 2\}$ and is defined by

\[
\gamma(t) = \begin{cases}
1, & 0 \leq t < t_1; \\
\vdots & \\\end{cases}
\]
It can be useful to treat an ULSS—1 structure as two connected LTI structures, with dependence through the state vector. This allows us to adapt the techniques for LTI structures for this setting.

Outline of an approach to investigating structural minimality for a ULSS—1 structure
One analytical approach: substitute each pole of a transfer function—denominator zero—into the transfer function numerator to check if this yields zero. (That is, pole—zero cancellation can occur.) We can investigate this for the first LTI structure of the ULSS—1. However, for the second structure, the initial conditions depend on the switching time. This takes the problem away from being a standard LTI problem, and analytical approaches may be quite laborious.

Numerical approach: suppose that the dynamics after the switch are controlled by an n x n matrix. We substitute each pole of a transfer function into the numerator to obtain a system of cancellation conditions of the form:

\[ f(t) = a_n(t) + a_{n-1}(t) \cdot \exp(\lambda_n(t) \cdot t) + \ldots + a_1(t) \cdot \exp(\lambda_1(t) \cdot t) \]

We will investigate the solutions for t of such a condition by generating random values for the parameter vector \( \theta \).

If we cannot find \( f(t) = 0 \) for feasible t, then we have some indication that pole—zero cancellation cannot occur.

Our motivating example: the "two-state conformational change" structure of biomolecular interactions on a flow-cell optical biosensor (structure \( \mathcal{C} \))
A BIAcore is a leading brnd of flow—cell optical biosensor used to study the dynamics of biochemical interactions in real time. Most simply, analyte in solution is made to flow through a flow chamber, where it binds to ligand immobilised on a chip surface. The change in refractive index near the surface as a result of binding is translated into a sensor response.

We shall consider the experiments comprised of an "association" phase (analyte present in flow, net formation of complex) followed by a "dissociation" phase (no analyte in flow, net dissociation of complex). We shall present the ODE system describing the data obtained from a BIAcore experiment under a particular interaction mechanism below.

Structure \( \mathcal{C} \) Definition
The "two-state conformational change" interaction model is summarised by a reaction scheme relating analyte A, ligand B, complex (AB) and an alternative form of this complex, (AB)*:

\[ k_2 \quad k_1 \quad k_2 \quad k_1 \quad k_2 \quad k_1 \]

forward reaction rate constants

reverse reaction rate constants

The system has initial state \( \{ [B], [\text{(AB)}], [\text{(AB)*}] \} \) for \( t=0 \).

Forming the parameter vector \( \theta = \{ k_1, k_2, k_2, k_2, k_2 \} \).

The differential equation system for the state variables is:

\[ \frac{d}{dt} [B](t, \theta) = -k_2 [A(t)] [B](t, \theta) + k_2 [\text{(AB)}](t, \theta) \]

\[ \frac{d}{dt} [\text{(AB)}](t, \theta) = k_2 [A(t)] [B](t, \theta) - (k_2 + k_2) [\text{(AB)}](t, \theta) + k_2 [\text{(AB)*}](t, \theta) \]
\[ \frac{d}{dt} [(AB)^*(t, \theta)] = k_2 \cdot [(AB)](t, \theta) - k_{-2} \cdot [(AB)^*(t, \theta)] \]

with analyte concentration \([A(t)]=\frac{\alpha}{\theta}(0 \leq t < t_1, 0 \geq t \geq t_1)\),

where \(t_1\) is the switching time.

By definition, rate constants \(k_a\), \(k_d\), \(k_2\), and \(k_{-2}\) are positive. The initial amount of functional free ligand present in the system, \(\beta_1\), is non-negative. However, as no reaction can proceed for \(\beta_1 = 0\), we assume that \(\beta_1\) is positive.

We model the biosensor response by \(y(t, \theta) = \frac{\alpha}{1 + \frac{k_d}{k_a + k_d} x_1(t, \theta)} \cdot [(AB)](t, \theta)\).

Let us use these relationships to compose the system \(C(\theta)\), the representative system of structure \(C\).

Following the approach given in Chapter 3 of Whyte (2016), ULSS-1 \(C(\theta)\) gives rise to two linear time-invariant systems.

The behaviour of the ULSS-1 prior to \(t_1\) (the association phase of an experiment) is encapsulated by \(C(1)\) having state vector \(x^{(1)}\), initial state \(x_0^{(1)}\) and output \(y^{(1)}\).

The behaviour of the ULSS-1 from \(t_1\) onwards (the dissociation phase) is given by \(C(2)\) having state vector \(x^{(2)}\), initial state \(x_0^{(2)}\) and output \(y^{(2)}\).

\[ \text{Defining the association phase structure } C_1 \text{ and solving for the time course of } x \]

We begin by defining the system matrices; \(A_1\) defines the dynamics of the interactions, \(C\) defines how the biosensor output depends on state variables:
\[
A_1 = \begin{bmatrix}
-\alpha & k_d & 0 \\
\alpha & -k_d - k_2 & k_2 \\
0 & k_2 & -k_{-2}
\end{bmatrix}; \quad C = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}
\]

To form the ODEs which describe the state — variable dynamics, we defining the state vector:
\[
X = \text{VectorCalcul} \{ \text{PositionVector} [\{x[1][t], x[2][t], x[3][t]\}, \text{cartesian}[x, y, z]]
\]
\[
X := \begin{bmatrix} x_1(t) \\
x_2(t) \\
x_3(t) \end{bmatrix}
\]

and its time derivative:
\[
Xdot := \text{diff}(X, t);
\]
We use $A_1$ in forming the right-hand side of the first-order ODEs describing the rates of change of concentrations of interacting species.

$$\text{eqn\_rhs} := \text{MatrixMatrixMultiply}(A_1, X);$$

$$\begin{bmatrix}
-k_a x_1(t) + k_d x_2(t) \\
k_a x_1(t) + (-k_d - k_2) x_2(t) + k_2 x_3(t) \\
k_2 x_2(t) - k_2 x_3(t)
\end{bmatrix}$$

Let us solve for the states, subject to the initial conditions

$$\text{ics} := x_1(0) = \beta_1, x_2(0) = 0, x_3(0) = 0$$

Define the dissociation phase structure $C_2$ and preparation for testing it for structural minimality

Define the system matrices for the dissociation phase structure; analyte concentration is set to zero from the start of the phase ($\alpha = 0$). (C is common to both phases.)

As the initial conditions of $C_2$ depend on the duration of the association phase modelled by $C_1$, it is easier to use unspecified initial conditions at this point.

$$A_2 := \text{subs}([\alpha = 0], A_1)$$

$$X_0 := \text{Matrix}(3, 1, [\beta_1, \beta_2, \beta_3]);$$

```plaintext
Let us solve for the states, subject to the initial conditions

$$\text{ics} := x_1(0) = \beta_1, x_2(0) = 0, x_3(0) = 0$$

This is the collection of parameters:

$$\theta := [k_a, k_d, k_2, k_{22}, \beta_1]$$

Defining the dissociation phase structure $C_2$ and preparation for testing it for structural minimality

Define the system matrices for the dissociation phase structure; analyte concentration is set to zero from the start of the phase ($\alpha = 0$). (C is common to both phases.)

As the initial conditions of $C_2$ depend on the duration of the association phase modelled by $C_1$, it is easier to use unspecified initial conditions at this point.
Here we load a procedure from my Maple Conference 2020 presentation (see the Springer book) which will allow us to obtain the transfer function for $E_2$:

```maple
> read "process_matrix_fn.mpl";
process_matrix := proc(sort_order::list(symbol), transfer_matrix::Matrix, canonical_form::truefalse, s::symbol := "s", $)
    local i, j, colMAX, rowMAX, current_element, leading_denom_coeff, processed_matrix, new_numer, new_denom;
    description
    "Prepare matrices of transfer functions for extraction of invariants. This involves pole-zero cancellation, and conversion of transfer functions into their canonical form (by ensuring denominators are monic) if "canonical_form" is set to true."
    i := 0;
    j := 0;
    rowMAX, colMAX := LinearAlgebra[Dimensions](transfer_matrix);
    processed_matrix := Matrix(rowMAX, colMAX);
    for i to rowMAX do
        for j to colMAX do
            current_element := normal(transfer_matrix[i, j]);
            leading_denom_coeff := coeff(denom(current_element), s);
            if canonical_form = true and leading_denom_coeff <> 1 then
                new_numer := eval(numer(current_element) / leading_denom_coeff); new_denom := eval(denom(current_element) / leading_denom_coeff)
            else
                new_numer := numer(current_element); new_denom := denom(current_element)
            end if;
            new_numer := sort(collect(new_numer, s), sort_order, plex);
            new_denom := sort(collect(new_denom, s), sort_order, plex);
            processed_matrix[i, j] := [new_numer, new_denom]
        end do;
    end do;
    return processed_matrix
end proc
```

The result of the block is a list of two elements, the numerator and denominator of the transfer function, respectively.

```maple
> n := LinearAlgebra[RowDimension](A2);
```

\[
A_2 := \begin{bmatrix}
0 & k_d & 0 \\
0 & -k_d & k_2 \\
0 & k_2 & -k_2
\end{bmatrix}
\]

\[
x_02 := \begin{bmatrix}
\xi_1 \\
\xi_2 \\
\xi_3
\end{bmatrix}
\]
We need to check if the zeroes of the denominator polynomial ("poles") satisfy the transfer function numerator polynomial. First, solving for the poles:

\[
\text{poles} := \text{solve}(H2\_proc[1][1][2] = 0, s) : \lambda_2 = \text{poles}[1] ; \lambda_1 = \text{poles}[2] ;
\]

\[
\lambda_2 = \frac{k_j}{2} - \frac{k_2}{2} - \frac{k_2}{2} + \sqrt{\frac{k_j^2 - 2 k_2 k_s - 2 k_2 k_s + k_s^2}{2}} - \frac{k_s}{2} + \frac{k_2}{2},
\]

\[
\lambda_1 = \frac{k_j}{2} - \frac{k_2}{2} - \frac{k_2}{2} - \sqrt{\frac{k_j^2 - 2 k_2 k_s - 2 k_2 k_s + k_s^2}{2}} - \frac{k_s}{2} + \frac{k_2}{2},
\]

(3.4)

As we are considering a compartmental model structure, we have \(\lambda_1 < \lambda_2 < 0\).

The apparent cancellation condition (we shall refine this later) is the equation where the transfer function numerator \(3.3\) = 0 for \(s = \lambda_1, \lambda_2\):

\[
\text{numerator} := \text{collect}(H2\_proc[1][1][1], [s, xi_2, xi_3]);
\]

\[
\text{numerator} := (\xi_2 + \xi_3) s + (\xi_2 + k_2) \xi_2 + (k_2 + k_2) s + k_2 k_2
\]

(3.5)

We see that the cancellation condition is satisfied for \(\xi_2 = \xi_3 = 0\).

However, this means that no complex of either form is produced by the association phase. This could only happen for an association phase of duration zero, which would not give us any data to analyse. As such, we can easily discard this trivial solution. The question remains: is there some \(t > 0\) for which the cancellation condition is satisfied?

In my PhD thesis I showed analytically (in relatively few lines) that the cancellation condition was not satisfied for \(s = \lambda_2\). Showing the same result for \(s = \lambda_1\) was rather more trouble!

\section*{Form the cancellation condition derived from (3.5) we will investigate (for \(s = \lambda_1\) from (3.4))}

Substituting into (3.5) the expressions for \(s = \lambda_1\) and the state variables from \(C_1\) we obtain the first expression:

\[
\text{cancellation\_condition} := \text{simplify}(\text{expand}(\text{subs}\{s = \text{poles}[2], \xi_2 = \text{rhs}(\text{Xsol}[2]), \xi_3 = \text{rhs}(\text{Xsol}[3]), \text{numerator}\}, \text{power, symbolic})
\]

(3.1.1)
We can show the sum of exponentials form more clearly with a rearrangement:

\[
\begin{align*}
\text{cancellation condition} & \implies - \frac{1}{\sqrt{k_d^2 + (2 k_o \alpha - 2 k_z + 2 k_j) k_d + (-k_o \alpha + k_z + k_d)^2}} \left( \alpha \beta_j k_o \left( \left( -k_z - k_j \right) \sqrt{k_d^2 + (2 k_z - 2 k_j) k_d + (k_j - k_z)^2} + k_j^2 + (k_j + 2 k_z) k_j - k_j \left( k_z \right) \right) \right) \\
+ 2 k_j \left( k_j - k_z \right) \left( k_j \right) & \left( k_j \right)
\end{align*}
\]

\[
\left( k_j \right) \left( k_j \right)
\]

(3.1.1)

We can show the sum of exponentials form more clearly with a rearrangement:

\[
\begin{align*}
\text{collect(cancellation_condition, indets (cancellation_condition, function))}
\end{align*}
\]

\[
\begin{align*}
\frac{1}{\sqrt{k_d^2 + (2 k_o \alpha - 2 k_z + 2 k_j) k_d + (-k_o \alpha + k_z + k_d)^2}} \left( \alpha \beta_j k_o \left( \left( -k_z - k_j \right) \sqrt{k_d^2 + (2 k_z - 2 k_j) k_d + (k_j - k_z)^2} + k_j^2 + (k_j + 2 k_z) k_j - k_j \left( k_z \right) \right) \right) & \\
\left( k_j \right) \left( k_j \right) & \\
\end{align*}
\]

(3.1.2)
\[ -\frac{\alpha \beta}{4} k_x \left( -2 \left( -k_x - k_z \right) \sqrt{k_x^2 + (2 k_x - 2 k_z) k_x + (k_x + k_z)^2} - 2 k_x^2 - 2 \left( k_x + 2 k_z \right) k_x + 2 k_x \left( k_x - k_z \right) \right) \]

We check that the expression obeys the expected initial condition:

\[ \text{simplify} \{ t = 0, \text{cancellation\_condition} \} \]

\[ \text{denom} \{ \text{cancellation\_condition} \} \]

\[ 4 \sqrt{k_x^2 + (2 k_x - 2 k_z) k_x + (k_x + k_z)^2} + \left( -k_x \alpha + k_z \right) \left( -k_x \alpha + k_z \right)^2 \left( \alpha k_x k_z + \alpha k_z k_x + k_x k_z \right) \]

Maple has simplified the expression into factored form, which will allow us to simplify our problem. We also note the denominator:

\[ \text{simplified\_num} := \text{sort} \{ \text{simplify} \{ \text{num} \{ \text{cancellation\_condition} \}, \text{sort\_order} \} \}

\[ \text{collect} \{ \text{simplified\_num} \} \]

Which is always positive in our parameter space, so we neglect this and focus our attention on the numerator of (3.1.2)

\[ \text{simplified\_num} := \text{collect} \left\{ \text{simplify} \left\{ \text{simplified\_num} \right\}, \text{indets} \{ \text{simplified\_num, function} \} \right\} \]

\[ -2 \left( -k_x \alpha + k_z \right) \left( -k_x \alpha + k_z \right)^2 \left( \alpha k_x k_z + \alpha k_z k_x + k_x k_z \right) \]

We may simplify (3.1.5) further by neglecting another constant that will not change sign:

\[ \text{simplified\_num} := \text{collect} \left\{ \text{simplify} \left\{ \text{simplified\_num} \right\}, \text{indets} \{ \text{simplified\_num, function} \} \right\} \]

\[ -2 \left( -k_x \alpha + k_z \right) \left( -k_x \alpha + k_z \right)^2 \left( \alpha k_x k_z + \alpha k_z k_x + k_x k_z \right) \]

\[ -2 \left( -k_x \alpha + k_z \right) \left( -k_x \alpha + k_z \right)^2 \left( \alpha k_x k_z + \alpha k_z k_x + k_x k_z \right) \]

\[ \text{collect} \{ \text{simplified\_num} \} \]
For reference, here are the coefficients of

$$\expcoeffs{\text{simplified\_numer}, \text{function}}{\text{dependent}(t)}, \text{list};$$

$$T := \text{convert}(\\text{indets}(\text{simplified\_numer}, \text{function})(\text{dependent}(t)), \text{list});$$

$$T := \left\{ \begin{array}{ll}
\expcoeffs{\text{evalb}(\text{op}(T[1])/t = \text{true}) = \text{false}}{\text{orderedT}[1], \text{orderedT}[2]} & \\
\expcoeffs{\text{evalb}(\text{op}(T[2])/t = \text{true}) = \text{true}}{\text{orderedT}[2], \text{orderedT}[1]} & \\
\expcoeffs{\text{evalb}(\text{op}(T[2])/t = \text{true}) = \text{false}}{\text{orderedT}[1], \text{orderedT}[2]} & \\
\expcoeffs{\text{evalb}(\text{op}(T[2])/t = \text{true}) = \text{true}}{\text{orderedT}[2], \text{orderedT}[1]} & \\
\end{array} \right\}$$

$$\expcoeffs{\text{collect}(\text{simplified\_numer}, \text{orderedT}), \text{orderedT}};$$

For reference, here are the coefficients of (3.1.6):

$$c_0 = \expcoeffs{1 \text{ \_ print( )};} \quad c_1 = \expcoeffs{2 \text{ \_ print( )};} \quad c_3 = \expcoeffs{3 \text{ \_ print( )};}$$

$$c_0 = \left( -2 \left( -k_2 - k_2 \right) \sqrt{k_2^2 + \left( -2 k_2 + 2 k_2 \right) k_2 + \left( k_2 + k_2 \right)^2 \right) - 2 k_2^2 - 2 \left( k_2 + 2 k_2 \right) k_2 + 2 \left( k_2 - k_2 \right) k_2 \right) \sqrt{k_2^2 + \left( 2 \alpha \alpha + 2 k_2 - 2 k_2 \right) k_2 + \left( -\alpha k_2 + k_2 + k_2 \right)^2 \right)}$$

$$c_1 = \left( k_2^2 + \left( k_2 + 2 k_2 \right) k_2 - \left( k_2 - k_2 \right) k_2 + \left( k_2 - k_2 \right) \sqrt{k_2^2 + \left( -2 k_2 + 2 k_2 \right) k_2 + \left( k_2 + k_2 \right)^2 \right) \right) \sqrt{k_2^2 + \left( 2 \alpha \alpha + 2 k_2 - 2 k_2 \right) k_2 + \left( -\alpha k_2 + k_2 + k_2 \right)^2 \right) - \left( k_2^2 + \left( k_2 - k_2 \right) k_2 + \left( k_2 - k_2 \right) \sqrt{k_2^2 + \left( -2 k_2 + 2 k_2 \right) k_2 + \left( k_2 + k_2 \right)^2 \right) \right) \sqrt{k_2^2 + \left( 2 \alpha \alpha + 2 k_2 - 2 k_2 \right) k_2 + \left( -\alpha k_2 + k_2 + k_2 \right)^2 \right)}$$

$$c_3 = \left( k_2^2 + \left( k_2 + 2 k_2 \right) k_2 - \left( k_2 - k_2 \right) k_2 + \left( k_2 - k_2 \right) \sqrt{k_2^2 + \left( -2 k_2 + 2 k_2 \right) k_2 + \left( k_2 + k_2 \right)^2 \right) \right) \sqrt{k_2^2 + \left( 2 \alpha \alpha + 2 k_2 - 2 k_2 \right) k_2 + \left( -\alpha k_2 + k_2 + k_2 \right)^2 \right) + \left( k_2^2 + \left( k_2 - k_2 \right) k_2 + \left( k_2 - k_2 \right) \sqrt{k_2^2 + \left( -2 k_2 + 2 k_2 \right) k_2 + \left( k_2 + k_2 \right)^2 \right) \right) \sqrt{k_2^2 + \left( 2 \alpha \alpha + 2 k_2 - 2 k_2 \right) k_2 + \left( -\alpha k_2 + k_2 + k_2 \right)^2 \right)}$$

Placing these coefficients in a list data structure here avoids having to convert later

$$\text{CC\_coefficients} := \left[ \expcoeffs{\text{\_ \_ print( )};} \right];$$

$$\text{CC\_coefficients} := \left[ \expcoeffs{\text{\_ \_ print( )};} \right]$$

(3.1.7)

(3.1.8)

(3.1.9)

(3.1.10)

(3.1.11)
A numerical exercise to determine the solutions of (3.1.6) (related to the cancellation condition) for a range of parameter values

**Remarks:** previously I tried to solve the cancellation condition in an inefficient manner:
1. Nested do loops (now: "map")
2. generating each pseudo-random parameter value as required (now: generate all values at the start),
3. Attempted to solve $K^2 - K^2 \xi_3 = 0$, where each term was calculated individually (expect that this lead to many unnecessary function evaluations and simplifications) (now: consider an expression that is already a simplified sum of exponentials)
4. Didn't attempt to dispense with terms that will not influence the location of any possible zeros.

In the following, we attempt to determine the solutions of the cancellation condition for a variety of parameter values.

**Preparation and defining procedures**

Define uniform random variables to pseudo-randomly generate feasible values of the parameters.

```plaintext
> upper := [1.0*10^4, 1.0*10^8, 1, 100]; U := Vector[row](5); for i from 1 to 5 do; U[i] := RandomVariable('Uniform'(0, upper[i])); od;
upper := [10000.0, 1, 10000.0, 1, 100]
U := [0 0 0 0 0]
U_1 := _R
U_2 := _R0
U_3 := _R1
U_4 := _R2
U_5 := _R3
```

# Set the conditions of the exercise, set the seed of the pseudo-random number generator for reproducibility of the runs to follow.

```plaintext
> N := 10^5; working_digits := 10; seed := 1; randomize(seed);

# Form a matrix of parameter values
thetaMat := Matrix(N, numelems(theta));
for j from 1 to nops(theta) do; thetaMat[1..N, j] := Sample(U[j], N); od;
```

# Form the replacement lists that we use to form the specific cancellation conditions for a randomly generated parameter vector
```plaintext
parameter_lists := [seq(convert(thetaMat[m, ..], list), m = 1..N)];
```
A procedure to generate and solve a condition for an input list of parameter values

```plaintext
cc_solve_preparation := proc(parameter_list :: list, theta :: list, working_digits :: integer, alpha_cond :: list, cancellation_condition, $)
    local eqn, solution;
    Digits := working_digits;
    eqn := subs(alpha_cond, theta = parameter_list, cancellation_condition) = 0;
    solution := fsolve(eqn, t, 1.0 .. 1000, fulldigits);
    if {type(solution, float) = true} or {type(solution, list) = true} then solution :
        # elif {is(solution = { }) = true} then new_eqn := fi;
        elif {numelems({solution}) = 0} then freeze(lhs(eqn)) : else solution : fi;
    fi
end proc;
```

A procedure to process the results of procedure "cc_solve_preparation".

A procedure to assist us in plotting the results of the root-finding exercise:

Application of routines: zero finding and results processing

```plaintext
working_digits := 10; seed := 101; randomize(seed):
    tstart := time():
    working_digits := 10
    seed := 1
    alpha_cond := \[ \alpha = 0.1 \]
```

```plaintext
# Condition on the injected analyte concentration
alpha_cond := [alpha = 0.1];
N := 10000
working_digits := 10
seed := 1
1
alpha_cond := \[ \alpha = 0.1 \]
```

```plaintext
end proc;
```

```plaintext
# A procedure to process the results of procedure "cc_solve_preparation"
# A procedure to assist us in plotting the results of the root-finding exercise:
```

Application of routines: zero finding and results processing

```plaintext
results := map(cc_solve_preparation, parameter_lists, theta, working_digits, alpha_cond, simplified_numer):
calculated_duration := time() - tstart;
```

```
(4.1.2)

A procedure to generate and solve a condition for an input list of parameter values

```plaintext
cc_solve_preparation := proc(parameter_list :: list, theta :: list, working_digits :: integer, alpha_cond :: list, cancellation_condition, $)
    local eqn, solution;
    Digits := working_digits;
    eqn := subs(alpha_cond, theta = parameter_list, cancellation_condition) = 0;
    solution := fsolve(eqn, t, 1.0 .. 1000, fulldigits);
    if {type(solution, float) = true} or {type(solution, list) = true} then solution :
        # elif {is(solution = { }) = true} then new_eqn := fi;
        elif {numelems({solution}) = 0} then freeze(lhs(eqn)) : else solution : fi;
    fi
end proc;
```

A procedure to process the results of procedure "cc_solve_preparation"

Application of routines: zero finding and results processing

```
```

```plaintext
working_digits := 10; seed := 101; randomize(seed):
    tstart := time():
    working_digits := 10
    seed := 101
```

```
(4.2.2)
```

```plaintext
(4.1.1.1)
```
Process the results to understand some features:

```maplet
> results_processing(N, results):
```

```
"#cases with at least one numerical solution=8177"
"# cases of multiple solutions=0"
"#failures to solve equation=1823, failure percentage=18.2%"
```

The following histograms show the log-10 transformed values of the roots (unsigned, and the constituents of this list, the negative, and non-negative log10-transformed roots respectively) found for the cancellation condition.

```maplet
> result_plots := plot_zeroes_results(results, working_digits); display(result_plots);
```

```maplet
> working_digits := 30; randomize(seed); tstart := time():
> results_30digits := map(cc_solve_preparation, parameter_lists, theta, working_digits, alpha_cond, simplified_numer):
> calculated_duration:= time() - tstart;
> results_processing(N, results_30digits):
```

```
working_digits := 30
101
tstart := 599.672
calculated_duration := 114.937
"#cases with at least one numerical solution=9954"
"# cases of multiple solutions=0"
"#failures to solve equation=46, failure percentage=.460%"
```

```maplet
> result_plots_30digits := plot_zeroes_results(results_30digits, working_digits); display(result_plots_30digits);
```
> working_digits := 70; randomize(seed); tstart := time();
> results_70digits := map(cc_solve_preparation, parameter_lists, theta, working_digits, alpha_cond, simplified_numer);
> processed_70digit_results := results_processing(N, results_70digits):
> calculated_duration := time() - tstart;
> result_plots_70digits := plot_zeroes_results(results_70digits, working_digits) : display(result_plots_70digits);
What can we learn from when fsolve does not return a result?

Are we working on a problem that is ill-conditioned for some parameter values? Can we lessen this by scaling the equation we intend to solve?

```plaintext
local eqn, T, exponents, scaling, new_eqn, solution;
Digits := working_digits;
eqn := subs(alpha_cond, theta = parameter_list, cancellation_condition = 0);
T := convert(indices(lhs(eqn), function(dependent(t))), list);
exponents := [subs(t = 1, map(op, T))];
if (which_exponent = "min") then scaling := min(exponents);
elif (which_exponent = "max") then scaling := max(exponents):
# print(min_exponent);
new_eqn := simplify(lhs(eqn) - exp(abs(scaling) - t)) = 0;
# print(new_eqn);
solution := fsolve(new_eqn, t = 1.0..1000, fulldigits);
if (type(solution, float) = true) or (type(solution, list)) then solution :=
# elf (is(solution = ( )) = true) then new_eqn :=
elif (numelems(solution) = 0) then freeze(lhs(new_eqn)):
else solution :=
end proc;
```

 Commentary: for D working digits, the returned solutions tend to cluster more closely around $10^{-D}$ for $D = 10, 30, 70$. This supports a conjecture that the only solution to the cancellation condition is at $t = 0$. However, we are not certain to decrease the number of times fsolve fails to find a root as we increase D!
Digits := working_digits;
eqn := subs(alpha_cond, \[\{\theta = \alpha, \text{parameter_list}\}, \text{cancellation_condition}) = 0;
T := convert(indets(lhs(eqn), function(dependent(t))), list);
exponents := \{subs(t=1, map(op, T))\};
if which_exponent = "min" then scaling := \min(exponents) elif which_exponent = "max" then scaling := \max(exponents) end if;
new_eqn := simplify(lhs(eqn) * exp(abs(scaling) * t)) = 0;
solution := fsolve(new_eqn, t, \[1.0..1000\), fulldigits];
if \text{type}(\text{solution}, \text{float}) = true or \text{type}(\text{solution}, \text{list}) then solution else solution end if
end proc

Let's apply the function to see if scaling by \(\exp(2t)\) (where 2 is the largest of the two negative exponents) or by \(\exp(t)\) (where 1 is the smallest of the two negative exponents) makes any difference to the solutions we observe.

We begin by scaling our cancellation condition by \(\exp(2t)\):

\begin{verbatim}
> working_digits := 10; randomize(seed);
tstart := time();
which_exponent := "max";
Apply the equation setup and solve function:
> results_scaled_lambda2_eqn := map(cc_solve_preparation_with_scaling, parameter_lists, \theta, working_digits, alpha_cond, simplified_numer, which_exponent);
> calculated_duration := time() - tstart;
working_digits := 10
101
tstart := 2243.406
which_exponent := "max"
calculated_duration := 383.619
\end{verbatim}

\begin{verbatim}
> any_sol_indices, multi_sol_indices := results_processing(N, results_scaled_eqn):
"#cases with at least one numerical solution=8438"
"# cases of multiple solutions=0"
"#failures to solve equation=1562, failure percentage=15.6%"
\end{verbatim}

Here we check on how many of the #failures to solve shown above are due to fsolve issues we planned to catch:

\begin{verbatim}
> fsolve_non_solutions := thaw(select(type, results_scaled_eqn, \{\text{function}, \text{symbol}\}) : numelems(fsolve_non_solutions);
1562
\end{verbatim}

\begin{verbatim}
> scaled_lambda2_results := plot_zeroes_results(results_scaled_eqn, working_digits):
Results obtained from scaling the cancellation condition by \(\exp(\lambda_2 t)\) produce graphs that are visually indistinguishable.
\end{verbatim}

Compare results obtained for working_digits=10 for the unscaled (top row) and scaled (bottom row) equation:

Commentary: We see some noticeable differences result from such a "cheap" change!

\begin{verbatim}
> display(result_plots); display(scaled_lambda2_results);
\end{verbatim}
OPEN QUESTION: What can we learn from the function calls returned (the particular sums of exponentials) when fsolve cannot find a zero?

Back to the script: can we uncover useful relationships between coefficients by generating some using random parameter values?

```plaintext
> numerical_coefficients := proc (parameter_values, CC_coefficients, alpha_cond, $) :: list;
subs (theta =~ parameter_values, alpha_cond, CC_coefficients)
end proc;
```
numeral_coefficients := proc(parameter_values, CC_coefficients, alpha_cond, S): list; subs('-' =~ ') (theta, parameter_values, alpha_cond, CC_coefficients) end proc

> evaluated_coeffs := map(numerical_coefficients, parameter_lists, CC_coefficients, alpha_cond);

> p1 := pointplot3d( evaluated_coeffs, title = "Calculated coefficients", axes = normal, labels = ["c0", "c1", "c2"]));

Calculated coefficients

Further investigation of \(c_1\) and \(c_2\) values:

> evaluated_coeffs_matrix := convert(evaluated_coeffs, Matrix);

\[ c_{1\text{ negative vals}} := \text{numelems} (\text{select}[\text{flatten}] (x \rightarrow x < 0 \text{, evaluated_coeffs_matrix}[.., 2])); \]

\[ c_{1\text{ positive vals}} := \text{numelems} (\text{select}[\text{flatten}] (x \rightarrow x > 0 \text{, evaluated_coeffs_matrix}[.., 2])); \]

\[ c_{2\text{ positive vals}} := \text{numelems} (\text{select}[\text{flatten}] (x \rightarrow x > 0 \text{, evaluated_coeffs_matrix}[.., 3])); \]

Commentary: It seems to be more than coincidence that for all 10000 runs \(c_2 > 0\) and \(c_1 < 0\). This should give us something to exploit in a proof.