Continuous Time Identification in Laplace Domain

Phanindra Jampana * **Ketan Detroja** **

* *Department of Chemical Engineering, Indian Institute of Technology Hyderabad, Medak, AP 502205 India (e-mail: pjampana@iith.ac.in).*
** *Department of Electrical Engineering, Indian Institute of Technology Hyderabad, Medak, AP 502205 India (e-mail: ketan@iith.ac.in).*

Abstract: We give a simple and accurate method for estimating the paramters of continuous time systems under the constraint that all the poles of the system lie to the left of the line $s = -1$. The method relies on the simple solution of a linear system of equations in the complex domain. We demonstrate by the use of simulation that the proposed methods gives accruate estimates when compared to existing methods. Methods for obtaining sparse solutions, which help in determining the order of the system are also given.

Keywords: Continuous time, least squares, laplace domain

1. INTRODUCTION

Developing an accurate model for a process is one of the most important issue in designing and tuning control algorithms. Many of the previously studied methods for system identification are in the discrete-time. For example, some of the standard discrete time identification techniques are discussed in (Ljung, 1987). However, in many cases it is desirable to estimate a continuous model for the process.

Continuous Time Identification (CTI) has gathered interest of late and there is considerable literature on the subject. For example, see (Young, 1981),(Unbenhauen and Rao, 1990), (Rao et al., 2006) and the references therein. CTI methods can be divided into two main cateogries: 1) Indirect and 2)Direct methods. Indirect methods first estimate a discrete time model and then compute the continuous time model from the discrete model. Direct methods estimate the continuous model from input output data directly.

The main challenge with indirect methods is that accurate conversion of models from discrete to continuous time is difficult when the sampling time is too small or large. Also, the zeros of the discrete time model cannot be easily transferred to continuous time (Garnier et al., 2003).

In direct methods, the most fundamental problem is the estimation of the time derivatives. Ofcourse, if the derivatives can be estimated accurately from sampled data then the solution can be obtained by an application of simple least squares. However, numerical estimation of time derivatives is not always accurate. The issue of derivative estimation is resolved in different ways in the literature. To highlight the differences between proposed method and existing methods, we describe the existing methods briefly. For details and a nice review of these methods please see (Garnier et al., 2003). The existing methods can be broadly classified into 1) Pre-filtering methods 2) Integration based methods and 3) Modulating function based methods.

Pre-filtering is done so as to reduce the errors in derivative estimation. The filter has to be designed so that it covers the range of frequencies of the process. At the same time it has to suppress high frequency components correspoding to noise. Some of the filters used in practice are State Variable Filters (SVF) and Generalized Poisson Moment Functionals (GPMF).

Methods based on integration, integrate the basic differential equation, so as to avoid computing derivatives. In this method, typically, n^{th} order integrals have to be performed.

Finally, in the Modulating Function approach, the derivatives of the input and output are transferred to the derivatives of the modulating functions (which are smooth), thereby bypassing the need to estimate derivatives from sampled data. Equations resulting from many such modulating functions are used in estimating an Equation Error model using least squares.

The proposed method is based on the observation that the Laplace *transformed* differential equation contains much information as the transformed equation is valid for (uncountably) infinitely many values of s. By exploiting this fact, many equations (in fact, as many as required) can be written involving the parameter vector. The resulting equations can then be directly solved. In this paper we only consider systems of the form $a_n \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + a_1 \frac{dy}{dt} + a_0 y(t) = b_0 u(t)$.

The paper examines the optimal choice of s's. The resulting matrix for any choice of s's is the Vandermonde matrix, infamous for being highly ill-conditioned. The optimal Vandermonde matrix in terms of condition number is the Discrete Fourier Transform matrix. It is this choice of s's on the unit circle that constrains the location of the

dominant pole of the process and the input to be to the left of the line $s = -1$.

2. PROPOSED METHOD

2.1 Noiseless Case

Consider a noiseless system described by

$$a_n \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + a_1 \frac{dy}{dt} + a_0 y(t) = u(t) \quad (1)$$

which when transformed into the laplace domain (assuming zero initial conditions) gives

$$(a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0) y(s) = u(s) \quad (2)$$

This equation is only valid in the intersection of the region of convergences of the Laplace transforms of both y and u. Writing the above equation for $n+1$ *different* values of s in this region, we get the matrix-vector equation given by $Ax = b$, where

$$A = \begin{pmatrix} 1 & s_1 & s_1^2 & \cdots & s_1^n \\ 1 & s_2 & s_2^2 & \cdots & s_2^n \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & s_{n+1} & s_{n+1}^2 & \cdots & s_{n+1}^n \end{pmatrix}$$

$$x = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} \quad (3)$$

$$b = \begin{pmatrix} u(s_1)/y(s_1) \\ u(s_2)/y(s_2) \\ u(s_3)/y(s_3) \\ \vdots \\ u(s_{n+1})/y(s_{n+1}) \end{pmatrix}$$

The matrix A in equation (3) is known as *Vandermonde* matrix and has the interesting property that if $s_1, s_2, \cdots, s_n$ are all distinct then the matrix is invertible (or has full rank). Therefore, if $s_1, s_2, s_3, \cdots, s_{n+1}$ are chosen so that they are all distinct, (3) has a unique solution, which should coincide with the original parameter vector.

2.2 Numerical Issues

Typically, only $y(t)$ $\forall t \geq 0$ is known and not its laplace transform. However, the laplace transform can be found numerically from the time domain data, i.e.

$$\int_0^\infty e^{-st} y(t) dt \approx \delta \sum_{k=0}^{T} e^{-sk\delta} y(k\delta) \quad (4)$$

where δ is the sample time. Note that for values of s outside the region of convergence, the laplace transform will diverge so the choice of s is important. In the following, let $\mathcal{R}$ denote the intersection of the region of convergences of y and u.

If all $s_1, s_2, s_3, \cdots, s_{n+1}$ are chosen to lie in $\mathcal{R}$, theoretically, equation (3) has a unique solution. However, in practice a few numerical issues need to be addressed to obtain accurate solution. The two main problems that might arise are :

(1) Bad conditioning of A which results in inaccurate estimates of the parameter vector
(2) Errors in the estimating the laplace transforms $u(s), y(s)$ numerically

These two problems are addressed in the following. Vandermonde matrices are known to be highly ill conditioned and therefore the values of s need to be chosen carefully. The Discrete Fourier Transform Matrix given by

$$D = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-\frac{i2\pi}{n+1}} & e^{-\frac{i4\pi}{n+1}} & \cdots & e^{-\frac{i2n\pi}{n+1}} \\ 1 & e^{-\frac{i4\pi}{n+1}} & e^{-\frac{i8\pi}{n+1}} & \cdots & e^{-\frac{i4n\pi}{n+1}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & e^{-\frac{i2n\pi}{n+1}} & e^{-\frac{i4n\pi}{n+1}} & \cdots & e^{-\frac{in^2 2\pi}{n+1}} \end{pmatrix} \quad (5)$$

is a Vandermonde matrix with the choices $s_k = e^{-\frac{i2k\pi}{n+1}}, k = 0, 1, \cdots n$. The condition number of D denoted $\kappa(D)$ is equal to one, i.e. $\kappa(D) = \|D\|_2 \|D^{-1}\|_2 = 1$ as D is an orthogonal matrix.

Therefore, $A = D$ is chosen so that A has perfect conditioning. Finding the Vandermonde matrix A of a given size with the smallest condition number (with real entries) can only be posed as an optimization problem as closed form solutions do not yet exist. For example, see Walter (1975) for explicit constructions of Vandermonde matrices for small sizes and a nonlinear programming algorithm for large sizes. Here, the condition number used is based on the 1-norm, i.e., $\kappa(A) = \|A\|_1 \|A^{-1}\|_1$.

The matrix D contains values of s which lie on the unit circle. As all the choices of s need to lie in $\mathcal{R}$, complex numbers s with real parts greater than -1 should also lie in $\mathcal{R}$ so that the choice $A = D$ is meaningful. This choice of A also places a constraint on the input u. For example, u cannot be chosen to be a step as the ROC of the laplace transform of u is $Re(s) > 0$. Instead if $u(t) = e^{-at}$ where $a > 1$, then $Re(s) > 1$ is a part of ROC and the choice $A = D$ can be substantiated.

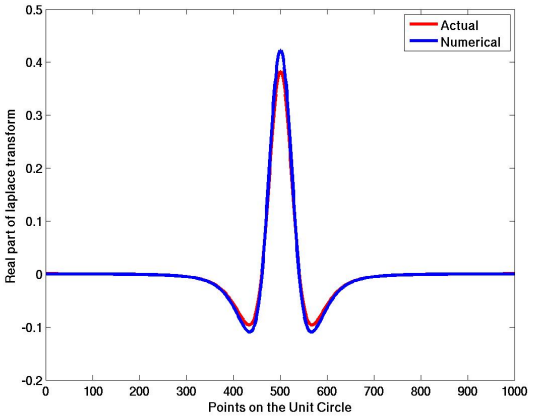
Another important consequence of the choice is that the poles of the system also need to lie to the left of the line $Re(s) = -1$. Finally, with the choice $A = D$, the solution vector can simply be calculated as $(n+1)^{-1} A^T b$.

Errors in the numerical estimation of the laplace transform also have an impact on the performance of the algorithm. If the dominant pole of the transfer function between y and u, i.e., $\frac{1}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0}$ lies far away to the left of the line $s = -1$, then the errors should be small. This is because $y(t)$ decays much faster in this case and hence errors for large values of t will be minimal.

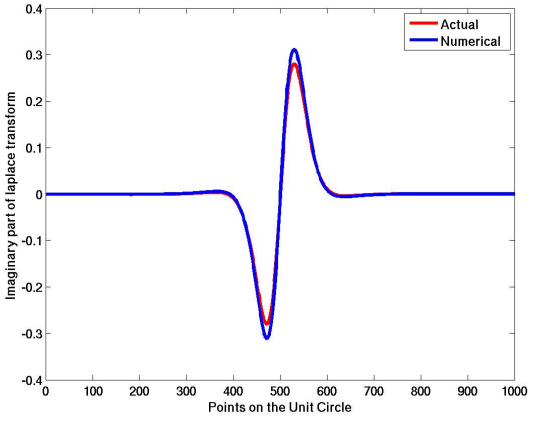
Figure 1 shows the comparison of the numerical and actual laplace transforms for $y(s)$ given by:

$$\begin{aligned} G(s) &= \frac{1}{(s+2) * (s+2.2) * (s+2.1)} \\ &\quad * \frac{1}{(s+2.8) * (s+3.0) * (s+2.1)} \\ u(s) &= \frac{1}{s+1.5} \\ y(s) &= G(s) u(s) \end{aligned} \quad (6)$$

In this system, it can be seen that the real parts of the poles are less than $s = -1$. This is required for previously described reasons.



(a) Real part



(b) Imaginary part

Fig. 1. Comparison of numerical and actual laplace transforms

The numerical laplace transforms are calculated using (4), width $\delta = 0.1, T = 100$. It can be seen from the figure that the difference between the two is minimal. However, this breaks the theoretical guarantee that the solution obtained is exactly equal to the parameter vector.

Therefore, in practice one can expect that the computed solution to differ slightly from the original if the matrix A is well conditioned. If the s's are chosen randomly in (3) the condition number of A has been found to be very high (e.g. of the order of 10^{10}), which makes the inversion process unstable even for small changes in the observation vector $u(s)/y(s)$. This observation reiterates the fact that Vandermonde matrices are typically ill-conditioned for arbitray values of $s_1, s_2, \cdots, s_n$.

2.3 Noisy Case

Consider the system described by

$$\begin{aligned} u(t) &= a_n \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + a_1 \frac{dy}{dt} + a_0 y(t) \\ z(t) &= y(t) + e(t) \end{aligned} \tag{7}$$

where $z(t)$ is the observed data and $e(t)$ is noise such that $|e(t)| < \epsilon \ \forall t$. The laplace transform of this system is

$$\begin{aligned} z(s) &= y(s) + e(s) \\ z(s) &= u(s)/(a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0) + e(s) \end{aligned}$$

Therefore, $(a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0) = \frac{u(s)}{z(s)} + (a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0) \frac{e(s)}{z(s)}$. For various values of s, this can be written as $Ax = b + e'$, where e' contains the values of $(a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0) \frac{e(s)}{z(s)}$ for the chosen values of s. With further simplification, this can be written as $Ax = b + BAx$. where B is a diagonal matrix containing the values $\frac{e(s)}{z(s)}$.

The bound on e' is first computed as this gives the magnitude of the error in the equation $Ax = b + e'$.

$$\begin{aligned} \|e'\|_2 &= \|BAx\|_2 \\ &\leq \|B\|_2 \|A\|_2 \|x\|_2 \\ &\leq \|B\|_2 \|A\|_2 \|x\|_2 \\ &\leq \left| \frac{e(s_{\max})}{z(s_{\max})} \right| \|A\|_2 \|x\|_2 \end{aligned} \tag{8}$$

where $s_{\max}$ is the value for which $\left|\frac{e(s)}{z(s)}\right|$ attains its maximum among the chosen values of s. Note here that the above is meaningful for $A = D$ only if the Laplace transform of the noise $e(s)$ *converges* for $\mathrm{Re}(s) > -1$. In this case, $\|e'\| \leq \left| \frac{e(s_{\max})}{z(s_{\max})} \right| \sqrt{n} \|x\|_2$.

If true Brownian motion (Mörters et al., 2010) is considered a model for $e(t)$, it can be shown that the Laplace transform converges (almost everywhere, in the sense of measures) only for $\mathrm{Re}(s) > 0$. Therefore, the above choice of $A = D$ is not applicable. However, in most simulations performed using MATLAB, for e.g. using the uniform random number generator, it was found that the numerical Laplace transform of noise indeed converges for $\mathrm{Re}(s) > -1$. We shall avoid explicit calculations of the convergence of the Laplace Transform of the continuous noise processes and assume in the following that it converges in $\mathrm{Re}(s) > -1$.

In the noisy case, it is more helpful to consider many observations than dictated by the order of the system – the number of rows of A can be chosen to be much larger than the columns. In this case, the solution can be obtained by standard least squares. For instance, for a sixth order system the matrix A can be chosen to have dimensions 20×7 by constructing the matrix from the first 7 columns of the 20×20 DFT matrix (5). Note that this ensures that $A^T A = cI$ where c is a constant and $\kappa(A) = 1$. The least squares solution is then given by $x_{\mathrm{ls}} = (A^T A)^{-1} A^T b$.

2.4 Numerical Experiments

Experiments on first, second order systems and the example in the previous section are presented here.

(1) **First order system**: Let the process transfer function be given by $G(s) = \frac{1}{s+2}$. We first obtain the response $(y(t))$ of this system to an exponential input $(u(t) = e^{-1.5t})$ using simulink (The exponential input

can be programmed using S-functions in MATLAB). The identification algorithm is provided with this data. In practice, one does not know the order of the system before hand. Therefore, we give some leeway and choose $n = 3$. That is, we are solving for a third order system even though the data have been produced by a first order response. The result from the algorithm is $x = (2.00, 1.20, 0.11, 0.0)^T$, i.e. $a_0 = 2.00, a_1 = 1.20, a_2 = 0.11, a_3 = 0.00$.

It can be said that the identification method closely approximates the true system. The third and fourth elements a_2, a_3 are small compared to the first two suggesting that the original system might indeed be first order.

(2) **Second order system**: Consider $G(s) = \frac{1}{s^2+2*1.4*s+1}$ and $u(s) = \frac{1}{s+1.5}$ as before. The solution obtained for $n = 3$ is $x = [1.18, 2.72, 1.46, 0.06]$ and for $n = 4$ is $x = [1.36, 2.46, 1.62, 0.00, -0.15]$. In both the cases the higher order coefficients are small compared to a_0, a_1, a_2. However, it is not accurate to say that system is second order based on the obtained solution as $a_5 = -0.1533$ does not seem to be small enough. This issue is analyzed later in Section 4.

(3) **Sixth order system**: Consider the system described in equation (6). The solution from the algorithm when $n = 6$ is $x = [163, 423.35, 456.18, 260.92, 83.51, 14.17, 1.0]$. The original parametor vector after expanding the terms is $x_0 = [163, 423, 456, 261, 84, 14, 1]$. For $n = 7$ the solution obtained is
$x = [163, 439, 500, 309, 112, 24, 2.8, 0.2]$.

3. COMPARISON

We compare the proposed method with the SVF and the GPMF methods mentioned in the introduction. For the sixth order system above, Table 1 shows the coefficients obtained from these methods. Note that, the Laplace transform for the input $\frac{1}{s+1.5}$ has been directly used in the simulation instead of calculating it from the input data. For the output however, the Laplace transform has been computed from the data as described previously. Table 1 has been obtained by using the *lssvf* and *lsgpmf* functions in the CONTSID toolbox. It can be observed that the estimates from the proposed method are closer to the actual values.

Table 1. Comparision of the proposed method with SVF and GPMF in Noiseless case

Coefficient	Actual	SVF	GPMF	Proposed
a_0	162.99	175.52	175.52	162.98
a_1	423.36	455.68	455.69	423.37
a_2	456.19	490.56	490.52	456.17
a_3	260.97	280.15	280.06	260.92
a_4	83.57	89.33	89.16	83.52
a_5	14.2	15.05	14.97	14.16
a_6	1.0	1.01	0.97	1.01

Table 2 shows the results when noise has been added to the data. The output is corrupted using additive noise $z(t) = y(t) + e(t)$ where $e(t)$ is a uniform random variable such that $|e(t)| < \epsilon$ as in (8). ϵ has been chosen such that the noise to signal ratio is equal to 0.05, i.e. 5%. The results of the algorithm are then obtained by giving as input $u(t), z(t)$ to the algorithms. For bias correction, *ivsvf* and *ivgpmf* methods have been used in the results shown in Table 2. The step response comparison has been shown in Figure 2. It can be seen that the estimates of the proposed method are closer to the true system.

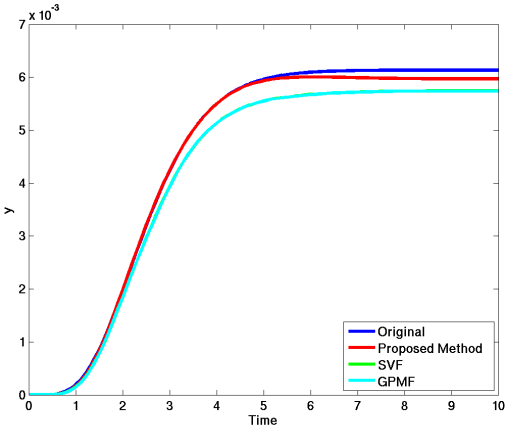


Fig. 2. Comparison of the step responses of proposed method with the SVF and GMPF methods

From Figure 3 it can be seen that the Nyquist plots of SVF and GPMF methods coincide away from the original system response and the proposed method is closer to the original system.

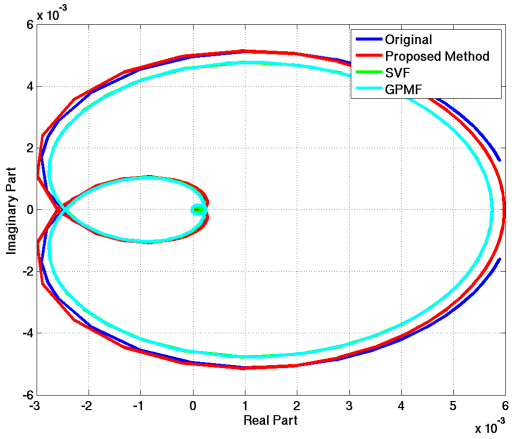


Fig. 3. Comparison of Frequency Response

We conclude from these comparisons that the proposed method performs satisfactorily when compared to existing methods such as SVF and GPMF.

Table 2. Comparision of the proposed method with SVF and GPMF in the Noisy case

Coefficient	Actual	SVF	GPMF	Proposed
a_0	162.99	174.24	174.39	167.47
a_1	423.36	456.57	456.41	419.61
a_2	456.19	493.33	495.24	457.12
a_3	260.97	284.05	283.23	266.19
a_4	83.57	98.97	101.25	86.08
a_5	14.2	16.13	15.73	16.86
a_6	1.0	2.90	3.44	1.92

4. SOLVING FOR SPARSE SOLUTIONS

The least squares solution to $Ax = b$ may contain elements which are close to zero but not exactly zero due to error in computing b (i.e. in estimating the numerical Laplace transform). Note that zero elements in the solution of (3) are desirable because they provide us with a better understanding of the order of the system. For example, we want a_5 to be exactly equal to zero in the second order example above.

This combinatorial problem can be posed as, find x such that

$$\|Ax - b\|_2 \leq \epsilon, \|x\|_0 \text{ is minimized} \tag{9}$$

where $\|x\|_0$ is defined as the number of non-zero elements of x. Obtaining a solution to this problem is computationally hard. A convex relaxed verion of this problem known as Basis Pursuit Denoising (BPDN) is typically solved instead:

$$\|Ax - b\|_2 \leq \sigma, \|x\|_1 \text{ is minimized} \tag{10}$$

Large literature exists on the convex relaxed problem above. When the number of rows of A is smaller than the number of its columns, this problem is studied in the area of Compressed Sensing. When the number of rows of A is larger than the number of its columns, the BPDN is closely related to Lasso due to Tibshirani (Tibshirani, 1996) which has been well established in the statistical commnunity.

For solving (10), a value for σ is needed. σ can be calculated based on (8). Indeed, if $\text{Re}(s_{\max}) > 0$, then

$$\begin{aligned} \|e'\| &\leq \left|\frac{e(s_{\max})}{z(s_{\max})}\right| \sqrt{n}\|x\|_2 \\ &\leq \frac{\epsilon\sqrt{n}}{\text{Re}(s_{\max})|z(s_{\max})|}\|x\|_2 \end{aligned} \tag{11}$$

The minimum value for $\text{Re}(s)|z(s)|$ over the chosen values of s can be used as to estimate $\|e'\|$. However, in practice it was found that this bound is too loose, allowing the zero vector to be a solution, i.e. $\sigma > \|b\|_2$. Therefore, σ is calculated in a different way as explained below.

Let x_{ls} be the least squares solution to $Ax = b$ with A chosen as in Section 2.3. The value for σ needs to be atleast as large as $\|Ax_{\text{ls}} - b\|$. Therefore, $\sigma = \|Ax_{\text{ls}} - b\| + \gamma$ is chosen, where $\gamma > 0$. With a value of $\gamma = 10$, the solution obtained by solving (10) is shown in Table below (the solution is denoted by x_{l1}).

Coefficient	x_{ls}	x_{l1}	x_{ls}^6
a_0	167.47	164.83	167.47
a_1	419.61	416.96	419.61
a_2	457.12	454.48	457.12
a_3	266.19	263.55	266.19
a_4	86.08	83.44	86.08
a_5	16.86	14.22	16.86
a_6	1.92	0	0
a_7	-1.35	0	0
a_8	0.94	0	0
a_9	1.30	0	0
a_{10}	2.53	0	0

It can be seen that the solution x_{l1} contains many zeros for the higher order terms. Even though the solution is accurate for $a_0, a_1, \cdots, a_5$ the procedure fails to obtain the correct value for a_6. However, using this solution a very close estimate of the order of the system can be obtained. The results as presented here were implemented using SPGL1 MATLAB code with the function `spg_bpdn`.

A problem with the BPDN algorithm is that the value of σ has to be known. In the previous example, $\gamma = 10$ was chosen but this was based on trial and error. Another method (called Sparse Least Squares below) for finding sparse solutions is described below. This method has been detailed elsewhere.

Let $x_{\text{ls}} = (A^TA)^{-1}A^Ty$ be the least squares solution as above. The best k-sparse approximation to x_{ls} is defined as $x_{\text{ls}}^k = \text{argmin}_x\|x - x_{\text{ls}}\|, \|x\|_0 = k$. In other words, x_{ls}^k consists of the first k maximum values of x_{ls} in the absolute sense. The rest of the elements in x_{ls}^k are equal to zero.

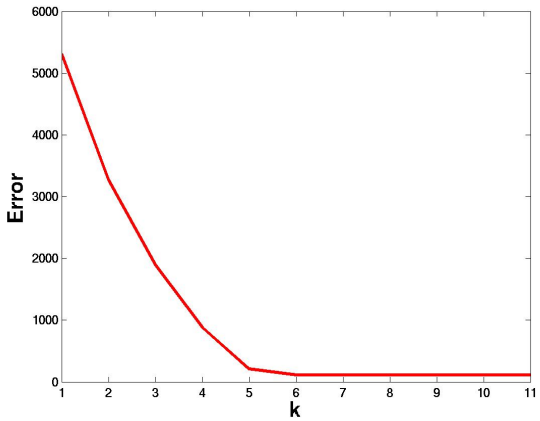


Fig. 4. Plot of k vs $\|Ax_{\text{ls}}^k - b\|$ (= Error)

From Figure 4 it is possible to guess that the order of the system is around 6 as the error plot is essentially constant after $k = 6$. Typically, such plots are obtained by solving the least squares solution for many values of k, the model order. Here, the least squares solution is obtained only once. The solution obtained for this order is shown in the Table. It can be seen that the solution is just a truncation of the least squares solution.

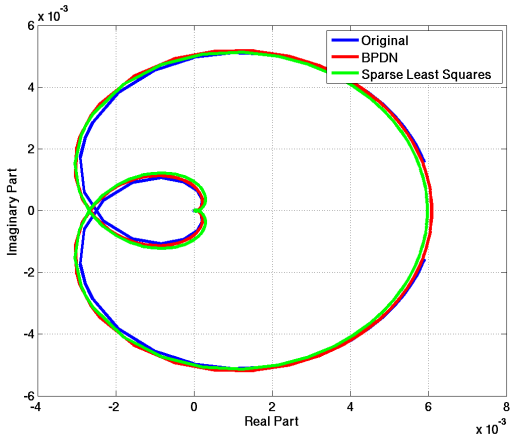


Fig. 5. Comparision of Nyquist plots for the sparse methods

Figure 5 compares the nyquist plots of the Original system with the sparse systems obtained from x_{l1}, x_{ls}^6. It can be seen that even though the sparse methods estimated a system of one lower order the nyquist plots are in close agreement.

5. CHALLENGES IN THE GENERAL CASE

In the general case (for example, for step inputs and stable systems) it can only be assumed that the Laplace Transforms for y, u will only converge for $\mathrm{Re}(s) > 0$. In order to cope with this using the proposed method, we need a choice of $s_1, s_2, \cdots, s_n$ so that $\mathrm{Re}(s_i) > 0 \ \forall i > 0$ and the resulting matrix have good condition number. However, it seems that (based on Monte Carlo simulations) with such choice of s_i's obtaining matrix with a good condition number is very hard.

Therefore, we look for other paradigms where the inversion process could be stable. One such approach is to use an underdetermined system of equations via the methods of Compressed Sensing. For example, it is known that (Akcakaya and Tarokh, 2008) if A is a $N \times M, M < N$ Vandermonde Matrix with distinct *nodes* (i.e. s_i's) and there exists a solution x to $A^T x = b$ such that $\|x\|_0 < \frac{N}{2}$ then this solution can be reconstructed exactly from the knowledge of b. The authors in the above referenced paper give a method for this reconstruction based on Euclid Algorithm. It has been observed that the algorithm does not result in the correct solution vector even if there are minute changes in the vector b. In other words, the inversion (or reconstruction) process is unstable.

In the realm of Compressed Sensing, properties such as Coherence and Restricted Isometry Property (RIP) play an important role in stability of inversion. We shall focus on Coherence here as RIP is NP-hard to check. Coherence of a matrix A is defined as

$$\mu(A) = \max \frac{|\langle u_i, u_j \rangle|}{\|u_i\| \|u_j\|}$$

where the maximum is taken over all $i \neq j$ and u_i's are columns of A.

Matrices with small Coherence allow stable reconstruction. The Coherence of Vandermonde Matrices is directly related to the Turan's problem (Bourgain et al., 2011). Turan's problem is to find the values of $s_1, s_2, \cdots, s_n$ so that

$$\max \left| \sum_{j=1}^{n} s_j^k \right|$$

is minimised. This can be seen to exactly equal to (upto a constant) the coherence defined above when the s_i's lie on the unit circle. Turan's problem does not have a generic solution and is considered an important open problem. For the purpose of System Identification in the general case, an additional constraint that $\mathrm{Re}(s_i) > 0 \ \forall i$ also needs to imposed in the Turan's problem. Therefore, it can be argued that fundamental results in other areas are needed to apply the proposed method in the general case.

6. CONCLUSION

In this paper we have presented a new method based on the simple solution to a system of equations which can be used to obtain the coefficients of a differential equation. In the noisy case, bounds on the Laplace transform of the error have been derived and the least squares solution is shown to give accurate results. When the order of the system is not known, sparse methods such as Basis Pursuit Denoising and another method due to the first author have been shown to aid in estimating the unknown order.

To be able to solve the problem in the general case, one needs fundamental results in other areas of mathematics (such as *positive* solutions to the Turan's problem). In this paper, initial conditions and orders of numerators greater than one are not considered. We believe that these problems too can be tackled using the proposed framework. This work will be pursued in future.

ACKNOWLEDGEMENTS

The first author would like to thank Dr. Challa Sastry and Dr. N. Aravind at IIT Hyderabad for helpful discussions.

REFERENCES

Akcakaya, M. and Tarokh, V. (2008). A frame construction and a universal distortion bound for sparse representations. *IEEE Transactions on Signal Processing*, 56(6), 2443–2450.

Bourgain, J., Dilworth, S., Ford, K., Konyagin, S., and Kutzarova, D. (2011). Explicit construction of rip matrices and related problems. *Duke Math. J.*, 159(1), 145–185.

Garnier, H., Mensler, M., and Richard, A. (2003). Continuous-time model identification from sampled data: Implementation issues and performance evaluation. *International Journal of Control*, 13, 1337–1357.

Ljung, L. (1987). *System Identification: Theory for the User*. Prentice-Hall Information and System Sciences Series. Pearson Education Canada.

Mörters, P., Peres, Y., Schramm, O., and Werner, W. (2010). *Brownian Motion*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Rao, G., Unbenhauen, H., and Heinz, D. (2006). Identification of continuous-time systems. *IEE Proceedings of Control Theory and Applications*, 185–220.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*

Unbenhauen, H. and Rao, G. (1990). Continuous-time approaches to system identification-a survey. *Automatica*, 23–35.

Walter, G. (1975). Optimally conditioned vandermonde matrices. *Numerische Mathematik*, 24, 1–12.

Young, P. (1981). Parameter estimation for continuous-time models. *Automatica*, 23–39.