

## A fast algorithm for linear least-squares smoothing and boundary value problems using number-theoretic transforms

Jin-Jen Hsue, Andrew E. Yagle\*

*Department of Electrical Engineering and Computer Science, Electrical Engineering and Computer Science Building, The University of Michigan, Ann Arbor, MI 48109-2122, USA*

Received 25 March 1993; revised 26 August 1993

---

### Abstract

A fast algorithm for linear least-squares smoothing and boundary value problems using number-theoretic transforms (NTT) is presented. The algorithm utilizes the fact that the fixed interval smoother can be imbedded in a boundary value problem, which can be reformulated as a problem of solving a perturbed-circulant system of equations. The new algorithm solves this perturbed-circulant system of equations by decomposing the solution into a circular deconvolution filter, which can be implemented using NTT, and a small-kernel FIR filter, which only involves a small matrix inversion. The major advantages of this algorithm are (1) it avoids roundoff error and attendant conditioning problems, (2) no storage or computation of complex, irrational roots of unity is required, and (3) computations involving large numbers are broken up into computations involving smaller numbers, which can be performed faster and in parallel.

### Zusammenfassung

Ein schneller Algorithmus zur Linearen Least-Squares Glättung und für Randwertprobleme mit Hilfe zahlentheoretischer Transformationen (ZTT) wird vorgestellt. Der Algorithmus nützt die Tatsache aus, daß die Glättung über ein festes Intervall in ein Randwertproblem eingebettet werden kann, das wiederum in ein Problem umgeschrieben werden kann, das die Lösung eines gestörten zirkulanten Gleichungssystems darstellt. Der Algorithmus löst dieses System, indem die Lösung in ein zirkulares Entfaltungsfiter, das mit Hilfe der ZTT realisiert wird, und ein kurzes FIR Filter zerlegt wird, bei dem nur eine kleine Matrix invertiert werden muß. Die Hauptvorteile des Algorithmus sind: (1) Er vermeidet Rundungsfehler und daraus entstehende Konditionierungsprobleme. (2) Man braucht komplexe, irrationale Einheitswurzeln weder zu berechnen noch zu speichern. (3) Berechnungen, bei denen große Zahlen auftreten, werden in Berechnungen überführt, bei denen kleinere Zahlen verwendet werden, wobei man dann parallel und deshalb schneller arbeiten kann.

### Résumé

Nous présentons dans cet article un algorithme rapide pour l'adoucissement linéaire aux moindres carrés et pour des problèmes de valeurs limites basé sur les transformations de type théorie des nombres (NTT). Cet algorithme utilise le fait que l'adoucisseur à intervalle fixe peut être immergé dans un problème de valeurs limites, qui peut être reformulé en tant que problème de résolution d'un système circulant perturbé d'équations. Cet algorithme nouveau résoud ce système

---

\* Corresponding author. Email: aey@eecs.umich.edu. Fax: 313-763-1503.

circulant perturbé d'équations en décomposant la solution en un filtre de déconvolution circulaire, qui peut être implanté à l'aide de la NTT, et en un petit filtre FIR noyau, qui implique seulement l'inversion d'une petite matrice. Les avantages majeurs de cet algorithme sont: (1) il évite les erreurs d'arrondi et les problèmes liés de conditionnement; (2) aucun stockage ou calcul de racines complexes irrationnelles de l'unité n'est requis; (3) les calculs impliquant des nombres élevés sont divisés en calcul impliquant des nombres plus petits, qui peuvent être exécutés plus vite et en parallèle.

*Key words:* Number-theoretic transforms; Linear least-squares; Fast algorithms

## 1. Introduction

Since the introduction of Kalman's recursive filter [6, 7], numerous algorithms for linear least-squares estimation have emerged [5, 8, 10]. Most of these algorithms are recursive and rely on solving a Riccati equation or equivalent recursive equations.

Recently, new algorithms for fixed interval smoothing, solution of Riccati equations, and block filtering problems that arise in linear least-squares estimation theory for discrete, time-invariant systems, have been proposed [4]. These algorithms reformulate the above three problems into a two-point boundary value problem, and then into a perturbed-circulant system of equations. They have several interesting features which make them attractive in the context of modern digital signal processing: they are nonrecursive; they are based on the fast Fourier transform (FFT); and they seem to be less sensitive to roundoff and truncation errors. As a result, they could be used to process large data sets efficiently in parallel, and they would be well suited for VLSI architecture.

Our goal in this paper is to develop a fast and error-free algorithm for boundary value problems or, equivalently, a solution of a perturbed-circulant system of equations. Our algorithm is similar to that of [4], but operates in a Galois field. Instead of using Fourier transforms, we use number-theoretic transforms (NTTs) [2] to implement circular convolutions. This creates two advantages: (1) since NTTs only involve integer operations, there is no roundoff error; and (2) if the Mersenne or Fermat NTT is used, no multiplications are required, since multiplication by a power of two can be implemented using bit shifting. Otherwise, Cooley–Tukey decompositions can be used to implement

the NTTs efficiently, provided the transform length is properly chosen.

Although NTTs have these advantages over FFTs, and map convolutions to multiplications as FFTs do, replacing the FFTs in [4] with NTTs entails the following problems. (i) NTTs can only be used in a Galois field, which is a field with a finite number of elements. Hence, the data must be scaled into integers. However, since the boundary value problem is nonlinear, scaling at the beginning will only complicate the problem. (ii) In a Galois field, division is much more expensive than multiplication. Hence, an efficient algorithm for division is necessary. (iii) Since all solutions in the Galois field are integers, they must be meaningfully related to the solutions in the original real or complex field.

The first problem is overcome by reformulating the two-point boundary value problem as a problem of solving a perturbed-circulant system of equations. The procedure to compute all entries of the system of equations involves nonlinear operations; therefore, it must be done in the real field. Once the system of equations is obtained, all entries can be scaled into integers; the problem can then be solved using an NTT-based algorithm. The second problem is solved by using the Chinese remainder theorem and the Euclidean algorithm to compute divisions in the Galois field [3]. Finally, the third problem of mapping integers into rationals is discussed in [3].

The advantages of our NTT-based algorithm over a direct floating-point implementation of the algorithm of [4] are as follows:

1. NTTs using two as a root of unity in a Galois field require no multiplications. NTTs using three or some other integer as a root of unity have the following advantages over a floating-point FFT: (i) no storage or computation of

complex roots of unity  $e^{j2\pi i/N}$ ,  $i = 1, \dots, N$ , is necessary; and (ii) no roundoff errors in computing or storing complex roots of unity, which lead to a nonorthonormal transform, are incurred.

2. Since the algorithm uses only integers, there is no roundoff error in any computation. This can be significant in poorly conditioned problems. There are word length (overflow) constraints, but these can be made arbitrarily large using residue number systems.
3. The use of a residue number system entails the following advantages: (i) computations for each modulus can be performed entirely in parallel; (ii) computations for each modulus involve small word lengths (i.e., fewer bits to represent smaller integers), so that additions and multiplications can be performed faster; and (iii) overflows for each modulus constitute no problem, as long as the final solution constructed by combining results from different moduli does not overflow.

It should also be noted that while our algorithm is adapted from the algorithm of [4], it is not merely an NTT-based implementation of it, since there are several nontrivial issues that arise in the adaption. These issues are summarized in the conclusion.

This paper is organized as follows. Section 2 briefly reviews the algorithm of [4]. Section 3 presents our algorithm for solving the perturbed-circulant system of equations. Section 4 presents and discusses some illustrative numerical examples. Section 5 concludes the paper with a summary.

## 2. Review of the FFT algorithm for linear least-squares smoothing problems

### 2.1. Problem formulation

Consider the following discrete-time multichannel ARMA system model:

$$u_{k+1} = \sum_{i=1}^{\gamma} A_i u_{k+1-i} + \varepsilon_k, \tag{1}$$

$$z_k = \sum_{i=1}^{\gamma} C_i u_{k+1-i} + \eta_k;$$

$$E[\varepsilon_k \varepsilon_l^T] = K \delta_{k,l}, \quad E[\eta_k \eta_l^T] = R \delta_{k,l}, \tag{2}$$

where  $u_k \in \mathbb{R}^m$ ,  $\varepsilon_k \in \mathbb{R}^m$ ,  $z_k \in \mathbb{R}^q$ ,  $\eta_k \in \mathbb{R}^q$ , and  $\{\varepsilon_k\}$  and  $\{\eta_k\}$  are zero-mean white processes uncorrelated with each other and with the random variable  $u_0$ . This system can be rewritten in state variable form as

$$\begin{aligned} u_{k+1} &= Au_k + B\varepsilon_k, \\ z_k &= Cu_k + \eta_k, \\ E[u_0 u_0^T] &= Q_0, \end{aligned} \tag{3}$$

where

$$u_k = \begin{bmatrix} u_{k+1-\gamma} \\ \vdots \\ u_k \end{bmatrix}, \quad A = \begin{bmatrix} 0 & I_m & & 0 \\ \vdots & & \ddots & \\ 0 & 0 & & I_m \\ A_\gamma & \dots & & A_1 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ I_m \end{bmatrix}, \quad C^T = \begin{bmatrix} C_\gamma \\ \vdots \\ C_1 \end{bmatrix}.$$

The fixed interval smoothing problem for this system is to find the linear least-squares estimate  $x_k = \hat{u}_k$  of  $u_k$  for  $k = 0, \dots, N$ , given the observations  $\{z_k, k = 0, \dots, N\}$ . This is known to be equivalent to solving the following two-point boundary value problem [1]:

$$\begin{aligned} x_{k+1} &= Ax_k + BK B^T \lambda_{k+1}, \\ \lambda_k &= A^T \lambda_{k+1} + C^T R^{-1} (z_k - Cx_k), \\ x_0 &= Q_0 \lambda_0, \\ \lambda_{N+1} &= 0. \end{aligned} \tag{4}$$

Here  $x_k$  denotes the smoothing estimate of  $u_k$ .

This boundary value problem can be solved with the fast Fourier transform by using a generalization of the circular decomposition algorithm [4]. Basically, this method decomposes the solution into a circular deconvolution filter, which can be implemented using the FFT, and a small-kernel FIR filter which operates only on the boundary observations and the boundary outputs of the FFT filter. This avoids solving a Riccati equation, which is the standard method for solving (4) [9].

2.2. Solution using the algorithm of [4]

The algorithm in [4] for solving the two-point boundary value problem (4), and thus the smoothing problem for (1), can be summarized as follows.

Step 1. Define the following quantities:

$$A_0 = -I_m, \quad C_0 = 0,$$

$$\tilde{x}^i = \begin{bmatrix} x_{-\gamma+1} \\ \vdots \\ x_0 \end{bmatrix}, \quad \hat{x}^i = \begin{bmatrix} x_1 \\ \vdots \\ x_\gamma \end{bmatrix},$$

$$\tilde{x}^l = \begin{bmatrix} x_{N-\gamma+1} \\ \vdots \\ x_N \end{bmatrix}, \quad \hat{x}^l = \begin{bmatrix} x_{N-2\gamma+1} \\ \vdots \\ x_{N-\gamma} \end{bmatrix},$$

$$D_{i,j} = A_i^T K^{-1} A_j + C_i^T R^{-1} C_j,$$

$$\tilde{A}_l = \sum_{i=l}^{\gamma} D_{i,i-l} \text{ for } l \geq 0; \quad \tilde{A}_l = \tilde{A}_{-l}^T \text{ for } l < 0.$$

$H$  = a block circulant matrix of order  $N - \gamma$  whose first row is

$$[\tilde{A}_0, \tilde{A}_1, \dots, \tilde{A}_\gamma, 0, \dots, 0, \tilde{A}_{-\gamma}, \dots, \tilde{A}_{-1}],$$

$$S = Q_0^{-1},$$

$$G_{i,l}^0 = \begin{cases} \sum_{j=1-l+\gamma}^{\gamma} D_{l+j-i,j} + S_{i,l} & \text{if } 1 \leq l \leq i, \\ \sum_{j=1-l+\gamma}^{i-l+\gamma} D_{l+j-i,j} + S_{i,l} & \text{if } \gamma \geq l > i; \end{cases}$$

$$G_{i,l}^1 = \begin{cases} -\sum_{j=0}^{i-l} D_{l+j-i+\gamma,j} & \text{if } 1 \leq l \leq i, \\ 0 & \text{if } l > i; \end{cases}$$

$$E_{i,l} = \begin{cases} -\tilde{A}_{-\gamma-i+l} & \text{if } i \leq l, \\ 0 & \text{if } i > l; \end{cases}$$

$$G^l = E^T.$$

Define  $V_0 = [0, \dots, 0, K]$ ,  $V_{i+1} = [V_i A^T]$ , where the  $V_i$  are  $m \times n$  matrices ( $n = m\gamma$ ):

$$W_i = V_i C^T R^{-1}, \quad i = 0, \dots, \gamma - 1,$$

$$T_{i+1,j}^0 = \begin{cases} \sum_{l=j}^{j+i} W_{i+j-l} C_{\gamma-l+1} + A_{\gamma-i-j} & \\ \text{if } 1 \leq j \leq \gamma - i, & \\ \sum_{l=j}^{\gamma} W_{i+j-l} C_{\gamma-l+1} & \\ \text{if } \gamma - i < j \leq \gamma, 0 \leq i < \gamma; & \end{cases}$$

$$T_{i,j}^1 = \begin{cases} 0 & \text{if } j < \gamma - i + 1, \\ A_\gamma & \text{if } j = \gamma - i + 1, \\ A_{j-i+1} - \sum_{l=1}^{j-\gamma+i-1} W_{i-1+j-\gamma-l} C_{\gamma-l+1} & \\ \text{if } j > \gamma - i + 1; & \end{cases}$$

$$\Psi_{11} = E(G^0)^{-1} G^1, \quad \Psi_{12} = -E,$$

$$\Psi_{21} = -E^T, \quad \Psi_{22} = E^T (T^0)^{-1} T^1.$$

Step 2. Compute the inverse  $H^{-1}$  of the block circulant matrix  $H$ . Then compute the LU decomposition of the  $2n \times 2n$  matrix (note that  $n = m\gamma \ll N$ ; see (1))

$$\Phi = I_{2n} - J^T H^{-1} J \Psi,$$

where

$$J^T = \begin{bmatrix} I_n & \dots & 0 & 0 \\ 0 & \dots & 0 & I_n \end{bmatrix}.$$

Step 3. FIR-filter the observations  $z_k$  to obtain the sequence

$$\tilde{z}_k^0 = \sum_{j=1}^{\gamma} C_j^T R^{-1} z_{k+j-1}, \quad 1 \leq k \leq N - \gamma.$$

This is the convolution of the observations  $\{z_k\}$  with a FIR filter  $\hat{C}_k$  defined as

$$\hat{C}_k = \begin{cases} C_{-k+1}^T R^{-1}, & -\gamma + 1 \leq k \leq 0, \\ 0 & \text{otherwise.} \end{cases}$$

Step 4. Extract the initial and terminal variables  $\hat{z}^i$  and  $\hat{z}^t$ :

$$\hat{z}_k^i = \sum_{j=1}^k C_{\gamma-k+j}^T R^{-1} z_{j-1}, \quad 1 \leq k \leq \gamma,$$

$$\hat{z}_k^t = \sum_{j=1}^k W_{k-j} z_{N-j+1}, \quad 1 \leq k \leq \gamma.$$

Step 5. Compute

$$\tilde{z}^b = J \begin{bmatrix} E(G^0)^{-1} \hat{z}^i \\ E^T(T^0)^{-1} \hat{z}^t \end{bmatrix}, \quad \tilde{z} = \tilde{z}^b + \tilde{z}^0.$$

Step 6. Solve the block circulant system of equations

$$Hy = \tilde{z}.$$

Step 7. Compute

$$b = \begin{bmatrix} \hat{x}^i \\ \hat{x}^t \end{bmatrix} = \Phi^{-1} J^T y.$$

Step 8. Compute the estimate  $x$  of  $u$  as

$$x = y + H^{-1} J \Psi b$$

using circular deconvolution.

Step 9. Compute the boundary value estimates (if desired) as

$$\tilde{x}^i = (G^0)^{-1} (G^1 \hat{x}^i + \hat{z}^i),$$

$$\tilde{x}^t = (T^0)^{-1} (T^1 \hat{x}^t + \hat{z}^t).$$

### 3. New algorithm for linear smoothing and boundary value problems using NTT

#### 3.1. Initialization

Our goal is to implement the algorithm of [4], summarized in the previous section, in a Galois field. The advantages of such an implementation include achieving higher computational efficiency and avoiding roundoff error. If the transform length and moduli are properly chosen, NTTs can be much more efficient than FFTs; e.g., the Fermat and Mersenne NTTs [2] are actually multiplication free. In the algorithm of [4], circular convolutions (or deconvolutions), which are computed in [4] using the FFT, are one of the major computa-

tional burdens. Hence, replacing FFTs by NTTs in the algorithm of [4] should reduce computational cost, avoid roundoff error, and gain the advantages discussed in the introduction.

Although NTTs have these advantages over FFTs, and map convolutions to multiplications as FFTs do, replacing the FFTs in the algorithm of Section 2 with NTTs entails several problems. First of all, NTTs can only be used in a Galois field, which is a field with a finite number of integers. A naive solution to this problem is to scale all numbers in the algorithm (including model coefficients and observation data) into integers and then run the algorithm. However, this will not work, due to the nonlinearity of the algorithm.

To solve this problem, we decompose the algorithm into two parts. The first part, which includes Step 1 to Step 5 of the algorithm in Section 2, includes all the nonlinear initializations and data processing. Most procedures in this part are small summations of products of model coefficients or small FIR filtering operations on observation data. Since no large circular convolutions are required in this part, running it in the real field is almost as efficient as running it in a Galois field.

#### 3.2. Solution of the perturbed-circulant system of equations

The second part of the algorithm amounts to solving the perturbed-circulant system of equations

$$[H - J\Psi J^T]x = \tilde{z}, \tag{5}$$

where  $H$  is a circulant matrix. All entries of the system matrix and the right-hand side are computed during the first part of the algorithm. Since we are dealing with a linear system of equations, scaling both sides of the system of equations by the same factor will not change the final solution.

Using the matrix inversion formula, the solution to (5) can be written as

$$x = H^{-1}\tilde{z} + H^{-1}J\Psi(I - J^T H^{-1}J\Psi)^{-1}J^T H^{-1}\tilde{z}. \tag{6}$$

Note that the first term  $H^{-1}\tilde{z}$  is a circular deconvolution which can be computed quickly using an

NTT. The second term requires only the inversion of the  $2n \times 2n$  matrix  $(I - J^T H^{-1} J \Psi)$  and two other circular deconvolutions. Recall that  $n = m\gamma \ll N$ , since  $m$  is the size of the vector  $u_k$  in (1) (not (3)),  $\gamma$  is the AR model order, and  $N$  is the length of the series of data. Since (6) is the major computation burden of the algorithm of Section 2, replacing FFTs with NTTs should gain some computational saving.

Solving a system of equations in a Galois field  $\text{GF}(p)$  requires implementation of divisions in the Galois field. Computing the division  $b/a$  in  $\text{GF}(p)$  is equivalent to solving the equation  $ax \equiv b \pmod{p}$ . An efficient implementation of division reported in [3] will be restated here. Since  $a$  and  $p$  are relatively prime, solving the problem  $ax \equiv b \pmod{p}$  is equivalent to solving  $ax + pk = 1$  for  $x$  and  $k$ ; the latter equation can be solved using the Euclidean algorithm. The number of integer divisions (computation of quotient and remainder) required by the Euclidean algorithm is bounded by  $2 \log_2 p$ .

### 3.3. Computation of the determinant

The third problem of solving (5) in  $\text{GF}(p)$  is to meaningfully relate the solution in  $\text{GF}(p)$  to the solution in the original real field. One possible solution of this problem is to compute the determinant of the system matrix; if this is known, the solution in the Galois field can be used to find the solution in the real field, since the determinant is the common denominator of the solution elements in the real field. Residue number systems can be used to ensure no overflow occurs in either the numerator or the determinant.

Computing determinants is difficult in general. Here we propose an efficient method to compute the determinant of the system matrix in (5), using the computation involved in the main algorithm. The determinant of the system matrix in (5) can be represented as

$$\begin{aligned} \det(H - J\Psi J^T) &= \det H \det(I - H^{-1} J\Psi J^T) \\ &= \det H \det(I - J^T H^{-1} J\Psi). \end{aligned} \quad (7)$$

Although  $\det(I - J^T H^{-1} J\Psi)$  may not be an integer, we know in advance that  $\det(H - J\Psi J^T)$  is an

integer, so that it can serve as a common denominator of the final solution. Note that both  $\det H$  and  $\det(I - J^T H^{-1} J\Psi)$  in (7) can be obtained as by-products of the main algorithm. Specifically,  $\det H$  can be computed as the product of the NTTs of the first row of  $H$ , and  $\det(I - J^T H^{-1} J\Psi)$  is obtained when the inverse of the  $2n \times 2n$  matrix  $(I - J^T H^{-1} J\Psi)$  (see (6)) is computed using Gaussian elimination (multiply the diagonal elements of the triangular factor).

The last problem of solving (5) in  $\text{GF}(p)$  is to avoid overflow in either the numerator or denominator of the final result. This can be done using residue number systems as follows [2]. First, choose as moduli  $m_i$  some primes, all of which permit number-theoretic transforms having the same transform length. For example, if the desired transform length is a power of 2, Fermat NTTs can be used. In general, if the desirable transform length is  $L$ , the  $m_i$  should be chosen so that  $L$  is a common divisor of the  $(m_i - 1)$ . Then solve (5) in the  $\text{GF}(m_i)$ ; note that computations in different  $\text{GF}(m_i)$  can be performed in parallel. Finally, the final solution is computed using the Chinese remainder theorem.

To prevent overflow, enough moduli  $m_i$  should be used to ensure that the integer  $|\det(H - J\Psi J^T)| < \prod m_i$ . A crude but simple bound on matrix determinants that is often employed [2] when residue number systems are used to solve systems of equations is the Hadamard inequality  $|\det A|^2 \leq \sum_i \sum_j a_{i,j}^2$ , where the  $a_{i,j}$  are the elements of the matrix  $A$ . Once the matrix  $H$  is computed in the initialization Step 1 above (which becomes Part 1 below), the Hadamard inequality can be used to ensure  $\prod m_i$  is large enough. This inequality is loose enough so that the numerators of the elements of  $x$ , as well as their denominators  $\det(H - J\Psi J^T)$ , will be smaller in magnitude than  $\prod m_i$ .

### 3.4. The new algorithm

The new algorithm for solving the smoothing problem for the scalar ARMA system (3), and the associated two-point boundary value problem (4), is as follows:

**Part 1 (floating-point operations).** This part includes all the nonlinear initializations.

$K$  = variance of the driving noise  $\varepsilon_k$ ;  
 $E[\varepsilon_k \varepsilon_l^T] = K \delta_{k,l}$ .  
 $R$  = variance of observation noise  $\eta_k$ ;  
 $E[\eta_k \eta_l^T] = R \delta_{k,l}$ .  
 $Q_0$  = covariance matrix of  $u_0$ ;  $E[u_0 u_0^T] = Q_0$ ,  
 $S = Q_0^{-1}$ .  
 $z_k$  = observation data for  $k = 0, \dots, N$ .  
 $A_i$  = AR coefficients for  $i = 1, \dots, \gamma$ .  
 $C_i$  = observation matrix coefficients for  $i = 1, \dots, \gamma$ .

*Initialization:*

$$D_{i,j} = A_i^T K^{-1} A_j + C_i^T R^{-1} C_j,$$

$$\tilde{A}_l = \sum_{i=l}^{\gamma} D_{i,i-l} \quad \text{for } l \geq 0,$$

$$\tilde{A}_l = \tilde{A}_{-l}^T \quad \text{for } l < 0.$$

$H$  = a block circulant matrix of order  $N - \gamma$  whose first row is

$$(\tilde{A}_0, \tilde{A}_1, \dots, \tilde{A}_\gamma, 0, \dots, 0, \tilde{A}_{-\gamma}, \dots, \tilde{A}_{-1}).$$

Compute the following sums:

$$G_{i,l}^0 = \begin{cases} \sum_{j=1-l+\gamma}^{\gamma} D_{l+j-i,j} + S_{i,l} & \text{if } 1 \leq l \leq i, \\ \sum_{j=1-l+\gamma}^{i-l+\gamma} D_{l+j-i,j} + S_{i,l} & \text{if } \gamma \geq l > i; \end{cases}$$

$$G_{i,l}^1 = \begin{cases} -\sum_{j=0}^{i-l} D_{l+j-i+\gamma,j} & \text{if } 1 \leq l \leq i, \\ 0 & \text{if } l > i; \end{cases}$$

$$E_{i,l} = \begin{cases} -\tilde{A}_{-\gamma-i+l} & \text{if } i \leq l, \\ 0 & \text{if } i > l; \end{cases}$$

$$G^l = E^T.$$

Define  $V_0 = [0, \dots, 0, K]$ ,  $V_{i+1} = [V_i A^T]$ , where the  $V_i$  are  $m \times n$  matrices:

$$W_i = V_i C^T R^{-1}, \quad i = 0, \dots, \gamma - 1;$$

$$T_{i+1,j}^0 = \begin{cases} \sum_{l=j}^{j+i} W_{i+j-l} C_{\gamma-l+1} A_{\gamma-i-j} & \\ \text{if } 1 \leq j \leq \gamma - i, & \\ \sum_{l=j}^{\gamma} W_{i+j-l} C_{\gamma-l+1} & \\ \text{if } \gamma - i < j \leq \gamma, 0 \leq i < \gamma; & \end{cases}$$

$$T_{i,j}^1 = \begin{cases} 0 & \text{if } j < \gamma - i + 1, \\ A_\gamma & \text{if } j = \gamma - i + 1, \\ A_{j-i+1} - \sum_{l=1}^{j-\gamma+i-1} W_{i-1+j-\gamma-l} C_{\gamma-l+1} & \\ \text{if } j > \gamma - i + 1; & \end{cases}$$

$$\Psi_{11} = E(G^0)^{-1} G^1, \quad \Psi_{12} = -E,$$

$$\Psi_{21} = -E^T, \quad \Psi_{22} = E^T (T^0)^{-1} T^1.$$

*Data processing:*

$$\hat{z}_k^0 = \sum_{j=k}^{\gamma} C_j^T R^{-1} z_{k+j-1}, \quad 1 \leq k \leq N - \gamma,$$

$$\hat{z}_k^i = \sum_{j=1}^{i-k} C_{\gamma-k+j}^T R^{-1} z_{j-1}, \quad 1 \leq k \leq \gamma,$$

$$\hat{z}_k^l = \sum_{j=1}^k W_{k-j} z_{N-j+1}, \quad 1 \leq k \leq \gamma.$$

$$\hat{z}^b = J \begin{bmatrix} E(G^0)^{-1} \hat{z}^i \\ E^T (T^0)^{-1} \hat{z}^l \end{bmatrix},$$

$$\tilde{z} = \hat{z}^b + \hat{z}^0.$$

Note that  $\hat{z}_k^0$  is the sum of only  $\gamma$  terms, and  $\hat{z}_k^i$  and  $\hat{z}_k^l$  are each computed for only  $\gamma$  values of  $k$ . Hence, there would be little advantage in using NTTs here anyway.

*Output:* The output of Part 1 includes the first row of  $H$ ,  $\tilde{z}$  and  $\Psi$ .

**Part 2 (integer operations).** The goal of this part is to solve the perturbed-circulant system of equations

$$[H - J\Psi J^T]x = \tilde{z},$$

where  $H$ ,  $\Psi$  and  $\tilde{z}$  are obtained from Part 1 and are represented in fixed-point notation. The first step is to scale all elements of  $H$ ,  $\Psi$  and  $\tilde{z}$  to integers. The solution is obtained by decomposing the inverse of the system matrix using the matrix inversion formula

$$x = H^{-1}\tilde{z} + H^{-1}J\Psi(I - J^T H^{-1}J\Psi)^{-1}J^T H^{-1}\tilde{z}$$

and computing circular deconvolutions via NTT. Addition and multiplication operations are performed modulo  $p$ . A division  $b/a$  is implemented by solving the equation  $ax \equiv b \pmod{p}$  using the Euclidean algorithm.

*Step 1.* Compute NTT of  $H_1$ , the first row of  $H$ , and compute the determinant of  $H$  by

$$\det H = \prod_i \text{NTT}\{H_1\}(i) \pmod{p}.$$

*Step 2.* Solve the circulant system of equations  $Hy = \tilde{z} \pmod{p}$ .

*Step 3.* Solve the circulant systems of equations  $H\tilde{\Phi} = J\Psi \pmod{p}$  (compute each column of matrix  $\tilde{\Phi}$  separately from each column of matrix  $J\Psi$ ). Compute  $\Phi = I - J^T\tilde{\Phi}$ .

*Step 4.* Solve the  $2n \times 2n$  system of equations  $\Phi b = J^T y \pmod{p}$  using Gaussian elimination in  $\text{GF}(p)$ , as in [2] (recall  $2n = 2m\gamma \ll N$ ). Compute  $\det \Phi$  by multiplying the diagonal elements of the triangular factor of  $\Phi$ .

*Step 5.* Compute the integer solution  $x$  using  $x = y + H^{-1}J\Psi b \pmod{p}$ .

*Step 6.* Compute the actual solution  $x$  using  $x = \frac{x \det \Phi \det H \pmod{p}}{\det \Phi \det H \pmod{p}}$ .

If residue number systems are used, replace  $p$  with  $m_i$  above. Note that computations in different  $\text{GF}(m_i)$  are completely independent.

#### 4. Examples

Consider the state space model (3) of a scalar ARMA process with model coefficients

$$[A_1 \ A_2] = [1.5 \ -0.8], \quad [C_1 \ C_2] = [0 \ 1],$$

$$K = R = 1, \quad Q_0 = \begin{bmatrix} 9.09 & 7.58 \\ 7.58 & 9.09 \end{bmatrix},$$

and observation data  $z = \{1.6592, 1.5951, 3.5445, -0.2941, -2.5365, -3.7380, -3.3166, -3.1045, -1.5057\}$ . Our goal is to compute the linear least-squares estimate  $x_k$  of  $u_k$ , given all of the observations  $\{z_k, k = 0, \dots, 8\}$ .

**Part 1.** The output of this part includes the first row of  $H$ ,  $\tilde{z}$  and  $\Psi$ :

$$H_1 = [4.89 \ -2.7 \ 0.8 \ 0 \ 0.8 \ -2.7],$$

$$\tilde{z} = [4.5917 \ -0.9458 \ -2.5365 \ -3.7380$$

$$\ -4.5212 \ -0.8459],$$

$$\Psi = \begin{bmatrix} 1.5847 & -0.4819 & 0.8 & -2.7 \\ -0.4819 & 0.1836 & 0 & 0.8 \\ 0.8 & 0 & -0.64 & 1.2 \\ -2.7 & 0.8 & 1.2 & -2.89 \end{bmatrix}.$$

**Part 2.** Scaling the output of Part 1 by a factor of 100 and rounding the results to the nearest integers, we obtain the following:

$$H_1 = [489 \ -270 \ 80 \ 0 \ 80 \ -270],$$

$$\tilde{z} = [459 \ -95 \ -254 \ -374 \ -452 \ -85],$$

$$\Psi = \begin{bmatrix} 158 & -48 & 80 & -270 \\ -48 & 18 & 0 & 80 \\ 80 & 0 & -64 & 120 \\ -270 & 80 & 120 & -289 \end{bmatrix}.$$

In practice, a much larger scaling factor would be used, so that more significant figures can be kept. We deliberately use a small factor, to keep the numbers in this illustrative example small. Choosing the modulus as 2593, and performing the algorithm in  $\text{GF}(2593)$ , yields the following.

*Step 1.*

$$H_1 = [489 \ 2323 \ 80 \ 0 \ 80 \ 2323],$$

$$\text{NTT}\{H_1\} = [109 \ 139 \ 679 \ 1189 \ 679 \ 139],$$

$$\det H = 109 \times 139 \times 679 \times 1189 \times 679$$

$$\times 139 \pmod{2593} = 484.$$

Step 2.

$$\Phi = \begin{bmatrix} 336 & 790 & 2428 & 1808 \\ 1336 & 223 & 1027 & 967 \\ 1011 & 2455 & 2112 & 2046 \\ 1900 & 1691 & 449 & 2383 \end{bmatrix}$$

Step 3.

$$y = [941 \ 903 \ 1778 \ 1438 \ 43 \ 1051]^T.$$

Step 4.

$$b = [1943 \ 1194 \ 2560 \ 176], \quad \det \Phi = 1073.$$

Step 5.

$$x = [1943 \ 1194 \ 1179 \ 1076 \ 2560 \ 176]^T.$$

Step 6.

$$x = [1312 \ 167 \ 2152 \ 1953 \ 1774 \ 1775]^T/732.$$

We now use residue number systems to avoid overflows. We choose as moduli  $m_i$  some primes which admit six-point number-theoretic transforms in  $GF(m_i)$ , i.e., 6 is a factor of  $m_i - 1$ . The least primitive root (LPR) for each  $m_i$  is also listed (see Table 1);  $(LPR)^{(m_i-1)/6}$  is a sixth root of unity in  $GF(m_i)$ , and is used as the base for the six-point NTT in  $GF(m_i)$ . Note that if the LPR is two, the

Table 1

$i$	$m_i$	$m_i - 1$	LPR
1	2593	$2^5 \cdot 3^4$	7
2	2917	$2^2 \cdot 3^6$	5
3	4093	$2^2 \cdot 3 \cdot 11 \cdot 31$	2
4	16381	$2^2 \cdot 3^2 \cdot 5 \cdot 7 \cdot 13$	2
5	1048573	$2^2 \cdot 3^3 \cdot 7 \cdot 19 \cdot 73$	2

NTT can be implemented without any multiplications (multiplication by a power of two can be performed using bit shifting).

Running the algorithm in each of the Galois fields  $GF(2593)$ ,  $GF(2917)$ ,  $GF(4093)$ ,  $GF(16381)$ , and  $GF(1048573)$  yields Table 2. The final solution is recovered from its residues using the Chinese remainder theorem.

$$x(1) = \frac{3\ 969\ 970\ 019\ 572\ 186}{2\ 039\ 392\ 752\ 946\ 238} \approx 1.9466,$$

$$x(2) = \frac{33\ 631\ 873\ 639\ 622}{2\ 039\ 392\ 752\ 946\ 238} \approx 0.0165,$$

$$x(3) = \frac{-4\ 631\ 406\ 586\ 600\ 928}{2\ 039\ 392\ 752\ 946\ 238} \approx -2.2710,$$

$$x(4) = \frac{-7\ 234\ 116\ 975\ 642\ 248}{2\ 039\ 392\ 752\ 946\ 238} \approx -3.5472,$$

$$x(5) = \frac{-6\ 437\ 206\ 468\ 837\ 182}{2\ 039\ 392\ 752\ 946\ 238} \approx -3.1564,$$

$$x(6) = \frac{-2\ 705\ 822\ 042\ 153\ 665}{2\ 039\ 392\ 752\ 946\ 238} \approx -1.3268.$$

For comparison, the floating-point solution computed using the algorithm of Section 2 is

$$x = [1.9519 \ 0.0204 \ -2.2682 \ -3.5457 \\ -3.1558 \ -1.3261].$$

Note the small discrepancies between these two answers. This is due to the rounding to integers after the results of Part 1 were scaled by 100; from that point the problem is solved exactly. This rounding error may be made arbitrarily small by using more moduli  $m_i$ , which permits representation of larger integers. Computations for each modulus involve relatively small integers, so they

Table 2

$m$	$x(1)$	$x(2)$	$x(3)$	$x(4)$	$x(5)$	$x(6)$	det
2593	1312	167	2152	1953	1774	1775	732
2917	384	727	2791	2078	2673	92	1848
4093	281	3457	3311	245	854	432	3923
16381	4736	14546	2984	3195	5161	8839	6643
1048573	629038	909137	909137	232652	419457	858345	17296

may be performed quickly, with small word lengths and bandwidths for inter-chip communications.

The purpose of this illustrative example was to demonstrate the algorithm and its operation. A larger example would demonstrate its advantages over a floating-point implementation of the algorithm of [4], but could not be presented here due to lack of space.

## 5. Conclusions

This paper has presented an NTT-based fast algorithm for linear least-squares smoothing problems and associated two-point boundary value problems. The algorithm adapts the algorithm of [4] to computation in Galois fields. This allows the usual advantages of residue number systems and integer computations to apply to the linear least-squares smoothing problem.

Problems that were solved in adapting the algorithm of [4] included: recognizing the necessity of performing nonlinear computations first (Part 1); applying the matrix inversion to (5) to transform the solution of a large perturbed-circulant system of equations into the solution of a circulant system (which can be performed using NTTs) and the solution of a much smaller system; and computation of  $\det(H - J\Psi J^T)$  to obtain the common denominator of the solution to the large perturbed circulant system. These were in addition to the usual problems of performing divisions in a Galois field and using residue number systems to avoid overflows.

A simple example illustrated the operation of the algorithm. However, this example (which was

small, due to lack of space) did not demonstrate the advantages of this algorithm over the algorithm of [4]. These advantages were listed at the end of the introduction, and will not be repeated here.

## 6. Acknowledgments

This work was supported by the National Science Foundation under grant #MIP-8858082.

## 7. References

- [1] B.D.O. Anderson and J. Moore, *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, NJ, 1979.
- [2] R.T. Gregory and E.V. Krishnamurthy, *Methods and Applications of Error-Free Computation*, Springer, New York, 1984.
- [3] J.J. Hsue and A.E. Yagle, "Fast algorithms for solving Toeplitz systems of equations using number-theoretic transforms", *Signal Processing*, Submitted.
- [4] A.K. Jain and J. Jasiulek, "Fast Fourier transform algorithms for linear estimation, smoothing, and Riccati equations", *IEEE Trans. Acoust. Speech Signal Process.*, Vol. ASSP-31, 1983, pp. 1435–1446.
- [5] T. Kailath, "A view of three decades of linear filtering theory", *IEEE Trans. Inform. Theory*, Vol. IT-20, 1974, pp. 146–181.
- [6] R.E. Kalman, "A new approach to linear filtering and prediction problems", *Trans. ASME J. Basic Engrg.*, Ser. D., Vol. 82, 1960, pp. 35–45.
- [7] R.E. Kalman and R.S. Bucy, "New results in linear filtering and prediction theory", *Trans. ASME J. Basic Engrg.*, Ser. D., Vol. 83, 1961, pp. 95–108.
- [8] D.G. Lainiotis, "Estimation: A brief survey", *Inform. Sci.*, Vol. 7, 1974, pp. 191–202.
- [9] D.Q. Mayne, "A solution to the smoothing problem for linear dynamic systems", *Automatica*, Vol. 4, 1966, pp. 73–92.
- [10] J.S. Meditch, "A survey of data smoothing for linear and nonlinear dynamic systems", *Automatica*, Vol. 9, 1973, pp. 151–162.