Fall 12-18-2014

Discrepancy Principle and Stable Parameter Estimation in Avian Influenza

Linda DeCamp

DISCREPANCY PRINCIPLE AND STABLE PARAMETER ESTIMATION IN AVIAN INFLUENZA

by

LINDA DECAMP

Under the Direction of Alexandra Smirnova, PhD

ABSTRACT

In the case of a linear ill-posed problem with noisy data, a version of an *a posteriori* parameter selection discrepancy principle (DP) [1] is justified for an arbitrary regularization strategy under very general assumptions on the operator and the stabilizer. Its efficiency is demonstrated for a practically important inverse problem in avian influenza. We refer to our result as an abstract discrepancy principle (ADP), which shows that applicability of the DP largely depends on the level of noise in the data rather than the method used for the construction of a specific regularization procedure.

INDEX WORDS: epidemiology, regularization, discrepancy principle.

DISCREPANCY PRINCIPLE AND STABLE PARAMETER ESTIMATION IN AVIAN INFLUENZA

by

LINDA DECAMP

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the College of Arts and Sciences

Georgia State University

2014

DISCREPANCY PRINCIPLE AND STABLE PARAMETER ESTIMATION IN AVIAN INFLUENZA

by

LINDA DECAMP

Committee Chair: Alexandra Smirnova

Committee: Vladimir Bondarenko

Michael Stewart

Changyong Zhong

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

December 2014

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PART 1 Introduction

Avian influenza belongs to a group of viruses known as Influenza A. The influenza A virus has many subtypes which are based on two surface proteins called hemagglutinin (referred to as H) and neuraminidase (N). The highly pathogenic avian influenza (HPAI) is of subtype H5N1. The virus occurs naturally in the wild bird population where it causes only mild symptoms. In this case it is referred to as low pathogenic avian influenza (LPAI). However, among domestic bird populations - chickens, ducks and turkeys - this virus exhibits as a highly contagious and virulent strain. The highly pathogenic avian influenza (HPAI) can kill up to 90-100% of the entire poultry flock within 48 hours [4]. Other species may be infected with HPAI through contact with secretions/excretions of infected birds. Horses, pigs, cats and humans have contracted the disease; for these species, the disease exhibits a high mortality rate [5]. The first documented case of HPAI of subtype H5N1 human infection was in 1997; the infected boy died. As of December 2013, 648 human cases have been reported to the World Health Organization (WHO) of which 384 have died, a mortality rate in excess of 60% [2, 6].

In the last century, the world has faced 5 major influenza pandemics. The most notable among these is the 1918 Spanish Flu pandemic which originated from avian influenza viruses. These pandemics occur when new strains of influenza are created by mutation (reassortment) of an existing, animal based strain. When humans are effective transmitters of the virus and generally have minimal immunity against the disease, a pandemic results [7, 8]. Though H5N1 is currently zoonotic, the mutability of influenza A viruses coupled with the high human mortality rate are significant motivators to investigate the parameters of this disease.

Several authors have analyzed the effectiveness of the today's control measures taken for avian influenza [9–11].

One of the major challenges in the study of HPAI is to estimate the transmission rate accurately so that the government agencies can develop adequate control strategies and safety measures. The transmission rate of a virus is equal to the transmission probability times the number of contacts with infected individuals. However, measuring the probability of a contact resulting in a disease is extremely difficult since it depends on a number of factors such as age, genetics, immunity, etc. For avian influenza of subtype H5N1 this probability varies in time due to temperature fluctuations, wild bird migrations, and other environmental changes [12]. In our study of avian influenza, instead of directly measuring the probability of the transmission, we propose to use the information available on human and poultry HPAI outbreaks, as well as other related data, to approximate a bird-to-human transmission rate by solving the underlying inverse problem [13]. To that end, this paper is organized as follows. The disease model for HPAI H5N1 is described in the next section. The necessity of using regularization methods for computing the transmission rate is explained. Four regularizations algorithms for solving linear inverse problems are described and related to the model. Particular emphasis is given to *a priori* and *a posteriori* stopping rules for these methods. Numerical experiments are then performed, first on simulated data for a prototype problem and then on real data. The qualitative and quantitative study of the computed solutions indicates that for real data, our numerical results capture a number of important properties that the highly pathogenic avian influenza (HPAI) transmission rate does, indeed, exhibit. The simulation study shows that the method introduced in this paper performs satisfactorily. Results of these experiments point to generalized stopping rule for all methods and this idea is formalized with a theorem and proof. Conclusions and discussions of anticipated future work complete the paper.

PART 2 Avian Influenza Model

Up until recently, models developed with annualized data primarily utilized constant bird-to-human transmission rates [9, 10, 14–16]. With the advent of more timely and frequent reporting, the data can be seen to ebb and swell over time. Following Martcheva and Tuncer [11, 12], we replace transmission rate with a time-dependent function and describe the avian influenza dynamics by the SI (susceptible-infected) model. According to this model, the human population is divided into two classes: susceptible humans, $S(t)$, and infected humans, $I(t)$. Assuming that humans recover with no immunity to the disease, we do not include the recovery class in the model. Similarly, the domestic bird population is divided into susceptible birds, $S_b(t)$, and birds infected with highly pathogenic avian influenza, $I_b(t)$. For poultry, the recovery class is omitted due to an extremely high virulence. In the proposed SI model, the rate of change in the number of susceptible humans is given by the following non-autonomous differential equation

$$\frac{dS}{dt} = \Lambda - \beta(t)I_b(t)S(t) - \mu S(t). \tag{2.1}$$

For susceptible humans, μ and $\beta(t)$ are the natural death rate and the bird-to-human HPAI transmission rate, respectively, which results in the force of infection being $\beta(t)I_b(t)S(t)$. The remaining parameter, Λ, is the growth rate of humans which combines the birth of new individuals as well as recruitment from the recovery class. These parameters are listed in Table 2.1. For this paper, μ and Λ are held as constant. Integrating $\beta(\tau)I_b(\tau)S(\tau)$ from the initial time, 0, to time equal to t, one obtains the cumulative number of HPAI human cases

over the period from 0 to t:

$$C(t) = \int_0^t \beta(\tau) S(\tau) I_b(\tau)\ d\tau + C(0), \qquad t \in (0, T). \tag{2.2}$$

The inverse problem of practical importance is to evaluate the time-dependent transmission rate, $\beta(t)$, given the number of poultry outbreaks, $I_b(t)$, the number of confirmed H5N1 human cases, $C(t)$, and the constant parameters Λ, μ and $S(0)$. Both $I_b(t)$ and $C(t)$ are measured with some level of noise.

Table (2.1) *Definitions of the Parameters in the Model*

Parameter	Meaning
$S(t)$	Susceptible humans
$I_b(t)$	Birds infected with HPAI
Λ	Birth/recruitment rate of humans
μ	Natural death rate of humans
$C(t)$	Cumulative number of human cases
$\beta(t)$	Transmission rate from birds to humans

There are two basic solution approaches. We may solve equation (2.1) for $S = S(\beta(t), t)$

$$S(\beta(t), t) = e^{-\int_0^t (\beta(\tau) I_b(\tau) + \mu) d\tau} \left\{ \Lambda \int_0^t e^{\int_0^u (\beta(\tau) I_b(\tau) + \mu) d\tau} du + S(0) \right\}. \tag{2.3}$$

Substituting this into the integral equation (2.2), one obtains

$$\int_0^t S(\beta(\tau), \tau) I_b(\tau) \beta(\tau) d\tau = C(t) - C(0). \tag{2.4}$$

In this solution method, we benefit from the lack of noise in the kernel. However, we are presented with the necessity of utilizing an iterative method to solve this nonlinear equation to obtain $\beta(t)$. Alternatively, we may solve equation (2.1) for $S = S(C(t), t)$. Equation (2.2)

combined with (2.1) implies

$$\frac{dS}{dt} = \Lambda - \mu S - \frac{dC}{dt}. \tag{2.5}$$

If $C(t)$ is known, we can view (2.5) as a linear ODE with respect to $S = S(C(t), t)$ (susceptible humans). We can solve this linear nonhomogeneous problem to obtain

$$S(C(t), t) = \left\{ S_0 + \int_0^t \left[\Lambda - \frac{dC}{d\tau} \right] e^{\mu\tau} d\tau \right\} e^{-\mu t} \tag{2.6}$$

or, alternatively, by expanding and integrating

$$S\left(C(t), t\right) = \left(S_0 - \frac{\Lambda}{\mu} + C(0) \right) e^{-\mu t} + \frac{\Lambda}{\mu} - C(t) + \mu \int_0^t C(\tau) e^{-\mu(t-\tau)} d\tau. \tag{2.7}$$

The last expression is preferable because it does not contain differentiation of noisy data. Using (2.7), one can find $\beta(t)$, bird-to-human transmission rate, from the following linear Volterra integral equation of the first kind [13]

$$A\beta(t) := \int_0^t K(C(\tau), \tau)\beta(\tau) d\tau = f(t), \quad f(t) := C(t) - C(0),$$

$$K(C(\tau), \tau) := S(C(\tau), \tau) I_b(\tau), \qquad t \in (0, T). \tag{2.8}$$

For now, assume that $A : \mathcal{X} \to \mathcal{Y}$ with $\mathcal{X} = \mathcal{Y} = L_2(0, T)$ over the field $\mathbb{R}$ or $\mathbb{C}$. As one can see from (2.8), the linearity comes at the expense of noise contamination both in the kernel of the operator and in the right hand side.

Note that the kernel in (2.8) does not depend on t so that we can view the equation as a differentiation problem. Then one will have to carry out numerical differentiation of noisy data, which is known to be an unstable procedure. A seemingly 'natural' approach that avoids numerical differentiation is to discretize equation (2.8) and get a linear system with a triangular matrix.

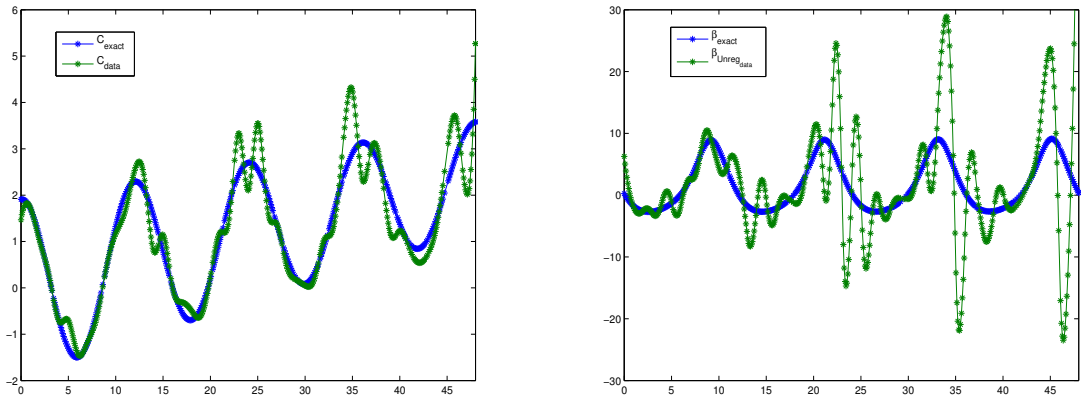

Figure (2.1) *Noisy $C(t)$ and the resulting $\beta(t)$ (unregularized) - Simulated Data*

Consider a simple example. Assume

$$C(t) = \eta t + \frac{\sigma \sin(\gamma t) + \gamma \cos(\gamma t)}{(\sigma^2 + \gamma^2)\exp(\sigma t)}. \tag{2.9}$$

This yields the following exact solution to our inverse problem:

$$\beta(t) = \frac{(\eta \exp(\sigma t) - \sin(\gamma t))\gamma}{I_b(t)\{S_0\gamma + 1 - \cos(\gamma t)\}}. \tag{2.10}$$

We take

$$I_b(t) = \sigma(\sin(\gamma t) + 2), \tag{2.11}$$

and set parameter values as follows: $S_0 = 10$, $\eta = .05$, $\sigma = .01$, $\gamma = \pi/6$. The time interval is 48 months. Function (2.9) possesses some (but not all) features of the actual model for the cumulative number of human cases (seasonal fluctuations, for example). The graph of $C(t)$ with no noise and with 25% noise is shown in Figure **??**. Figure **??** shows the graph for the corresponding analytic solution β and the graph of the solution obtained by solving the discretized equation $A\beta = f_\delta$ directly with the noisy data f_δ. The analytic solution illustrates a near-periodic transmission rate; the solution obtained with noisy data fails to accurately depict the true solution.

One would think that increasing the number of grid points for numerical integration would improve accuracy. However, in practice, such an approach exacerbates the problem. This can be shown by comparing the true solution, $\hat{\beta} = A^{-1}f$, with the computed solution, $\beta_\delta = A^{-1}f_\delta$, which gives us $\|\hat{\beta} - \beta_\delta\| \leq \|A^{-1}\|\|f - f_\delta\|$. Dividing both sides by $\|\hat{\beta}\|$ gives

$$\frac{\|\hat{\beta} - \beta_\delta\|}{\|\hat{\beta}\|} \leq \|A^{-1}\| \frac{\|f - f_\delta\|}{\|\hat{\beta}\|}, \tag{2.12}$$

the relative error of β, which is bounded above. Now

$$\|f\| = \|A\hat{\beta}\| \leq \|A\|\|\hat{\beta}\| \quad \text{implies} \quad \frac{1}{\|\hat{\beta}\|} \leq \frac{\|A\|}{\|f\|}. \tag{2.13}$$

Substituting estimate (2.13) into (2.12) implies

$$\frac{\|\hat{\beta} - \beta_\delta\|}{\|\hat{\beta}\|} \leq \underbrace{\|A^{-1}\|\|A\|}_{cond(A)} \frac{\|f - f_\delta\|}{\|f\|} \tag{2.14}$$

showing that the relative error of $\hat{\beta}$ is bounded above by the condition number of the matrix A times the relative error of f. As Table 2.2 illustrates, decreasing the step size in the discretization of the integral produces an increasing condition number of the corresponding matrix. As a result the noise in the data becomes magnified and negatively impacts the quality of the computed solution. This points to the necessity of incorporating a regulariza-

Table (2.2) *Relative errors for $\beta(t)$ and matrix condition numbers for decreasing h*

h	Relative Error	Cond(A)
0.200	1.97442259	1069
0.100	2.06413754	2179
0.050	2.11201994	4399
0.010	2.15146989	22166
0.005	2.15646443	44375

tion procedure into numerical algorithm, i.e., sacrificing some accuracy to achieve stability. Instead of computing $\beta_\delta = A^{-1}f_\delta$, we evaluate $\beta_{\alpha,\delta} := R_\alpha f_\delta$, replacing A^{-1} with R_α, its

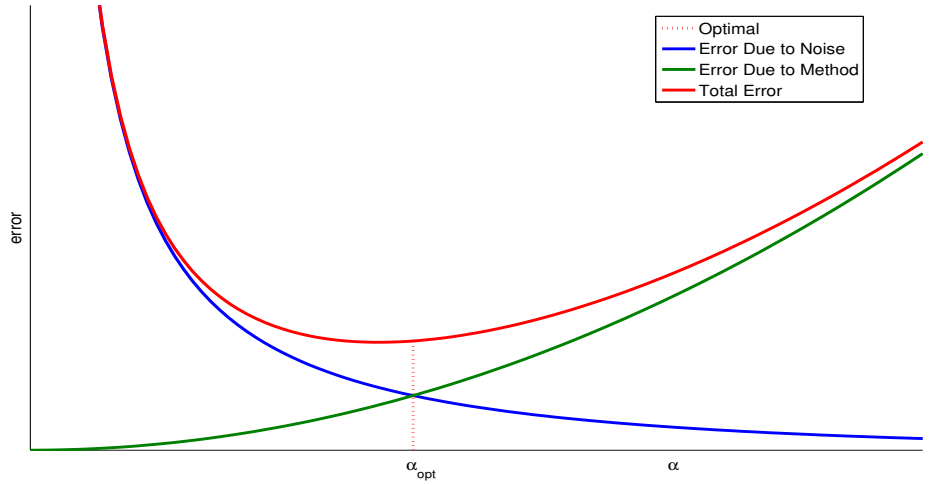

Figure (2.2) *Behavior of Error*

approximate, to gain stability. With this replacement, the error estimate changes as follows

$$\begin{aligned}
\frac{\|\beta_{\alpha,\delta} - \hat{\beta}\|}{\|\hat{\beta}\|} &= \frac{\|R_\alpha f_\delta - \hat{\beta}\|}{\|\hat{\beta}\|} \\
&= \frac{\|R_\alpha f_\delta \pm R_\alpha f - \hat{\beta}\|}{\|\hat{\beta}\|} \leq \frac{\|R_\alpha f_\delta - R_\alpha f\|}{\|\hat{\beta}\|} + \frac{\|R_\alpha f - \hat{\beta}\|}{\|\hat{\beta}\|} \\
&\leq \|R_\alpha\|\|A\|\frac{\|f_\delta - f\|}{\|f\|} + \frac{\|A\|\|R_\alpha A\hat{\beta} - \hat{\beta}\|}{\|f\|} \\
&\leq \|R_\alpha\|\|A\|\frac{\delta}{\|f\|} + \frac{\|A\|\|(R_\alpha A - I)\hat{\beta}\|}{\|f\|} := E_1(\alpha) + E_2(\alpha), \qquad (2.15)
\end{aligned}$$

and it accumulates two sources of error $E(\alpha) = E_1(\alpha) + E_2(\alpha)$: the first term is the error due to noise and the second term is the error due to the numerical method. It has been shown [17] that if A is not boundedly invertible, $\|R_\alpha\|$ cannot be uniformly bounded. As a result, it is noted that $\|R_\alpha\|\|f_\delta - f\| \to \infty$ and $\|R_\alpha A\beta - \beta\| \to 0$ as $\alpha \to 0$ and there is an optimal α that minimizes this sum (see Figure **??** for example). In the next chapter, we select R_α by Tikhonov's, Lavrentiev's and local regularization algorithms for solving integral equation (2.8). Additionally, we will use Legendre polynomial approach for stable numerical differentiation.

PART 3 Regularization Strategies and Discussion

3.1 The Tikhonov Regularization Algorithm

Tikhonov's regularization [18, 19] is one of the most known regularizing algorithms. For this method, instead of solving $A\beta = f_\delta$, we consider the following minimization problem

$$\min_{\beta}\{\|f_\delta - A\beta\|^2 + \alpha\|\beta\|^2\}, \quad \alpha > 0, \tag{3.1}$$

which results in the normal equation

$$A^*A\beta + \alpha\beta = A^*f_\delta \qquad \text{and} \qquad R_\alpha = (A^*A + \alpha I)^{-1}A^*. \tag{3.2}$$

While the penalty term $\alpha||\beta||^2$ incorporates smoothness (that we expect to have in our case), different penalty functionals can be used for other types of *a priori* information, like sparsity, discontinuity, etc. Tikhonov's regularization is one of those stabilizing algorithms that will never let you down considering the properties of A^*A. However, this regularization method does not conserve the Volterra structure of our original operator, and results in a dense matrix at the discrete level.

For Tikhonov's algorithm, setting $\hat{\beta}$ as the exact solution, we have

$$E(\alpha) = \|[A^*A + \alpha I]^{-1}A^*\|\|A\|\frac{\delta}{\|f\|} + \frac{\|A\|\|([A^*A + \alpha I]^{-1}A^*A - I)\hat{\beta}\|}{\|f\|} \tag{3.3}$$

as the total error for the method, analogous to (2.15). For the error due to noise, we have

$$E_1(\alpha) = \|[A^*A + \alpha I]^{-1}A^*\|\|A\|\frac{\delta}{\|f\|} \leq \sup_{\lambda\in\sigma(A^*A)}\frac{\sqrt{\lambda}}{(\lambda+\alpha)}\|A\|\frac{\delta}{\|f\|} \leq \frac{\|A\|\delta}{2\sqrt{\alpha}\|f\|} \tag{3.4}$$

and it is expected that

$$E_1(\alpha) \to \infty \quad \text{as} \quad \alpha \to 0, \tag{3.5}$$

where $\sigma(B)$ denotes the spectrum of the operator B. Addressing the error due to the numerical method, we notice that A defined in (2.8) is one-to-one. Hence the range $A^*(\mathcal{Y})$ is dense in $\mathcal{X}$. So, for $\epsilon > 0$, there is $\tilde{\beta} = A^*\tilde{z} \in A^*(\mathcal{Y})$ such that $\|\hat{\beta} - \tilde{\beta}\| < \frac{\epsilon\|f\|}{2\|A\|}$. Thus one has

$$\begin{aligned}
E_2(\alpha) &= \frac{\|([A^*A + \alpha I]^{-1}A^*A - I)\hat{\beta}\|\|A\|}{\|f\|} \\
&\leq \frac{\|([A^*A + \alpha I]^{-1}A^*A - I)(\hat{\beta} - \tilde{\beta})\|\|A\|}{\|f\|} + \frac{\|([A^*A + \alpha I]^{-1}A^*A - I)A^*\tilde{z}\|\|A\|}{\|f\|} \\
&\leq \|[A^*A + \alpha I]^{-1}A^*A - [A^*A + \alpha I]^{-1}[A^*A + \alpha I]\|\frac{\|\hat{\beta} - \tilde{\beta}\|\|A\|}{\|f\|} \\
&\quad + \frac{\|\left([A^*A + \alpha I]^{-1}A^*A - [A^*A + \alpha I]^{-1}[A^*A + \alpha I]\right)A^*\|\|\tilde{z}\|\|A\|}{\|f\|} \\
&\leq \frac{\alpha\|A^*A + \alpha I]^{-1}\|\|\hat{\beta} - \tilde{\beta}\|\|A\|}{\|f\|} + \frac{\alpha\|[A^*A + \alpha I]^{-1}A^*\|\|\tilde{z}\|\|A\|}{\|f\|} \\
&\leq \frac{\epsilon}{2} + \frac{\sqrt{\alpha}\|\tilde{z}\|\|A\|}{2\|f\|} \leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon \qquad \text{for sufficiently small } \alpha.
\end{aligned} \tag{3.6}$$

This proves that $E_2(\alpha) \to 0$ as $\alpha \to 0$ although one can observe from above that convergence of $E_2(\alpha)$ to 0 can be arbitrarily slow. However, if we additionally assume that $\hat{\beta} \in A^*(\mathcal{Y})$, i.e. $\tilde{\beta} = \hat{\beta}$, then we have

$$E_2(\alpha) \leq \frac{\sqrt{\alpha}\|\hat{z}\|\|A\|}{2\|f\|}. \tag{3.7}$$

And the total error for the method is

$$E(\alpha) \leq \frac{\|A\|\delta}{2\sqrt{\alpha}\|f\|} + \frac{\sqrt{\alpha}\|\hat{z}\|\|A\|}{2\|f\|} = \frac{\|A\|}{2\|f\|}\left[\frac{\delta}{\sqrt{\alpha}} + \sqrt{\alpha}\|\hat{z}\|\right], \tag{3.8}$$

$$\alpha_{opt} = \frac{\delta}{\|\hat{z}\|} \qquad \text{and} \qquad E(\alpha_{opt}) = \frac{\|A\|}{\|f\|}\sqrt{\|\hat{z}\|\delta}. \tag{3.9}$$

We can see that one can minimize $E(\alpha)$ to obtain an optimal value of α. However, this determination will depend on $\|\hat{z}\|$ which is not available *a priori*. Though we will not seek to describe the error in the descriptions of the remaining methods, it has been shown that prior knowledge about the exact solution is necessary to determine the optimal α in each of these methods [13, 20]. Rather than pursuing such an *a priori* method, we will instead use an *a posteriori* algorithm for choosing α that depends on the noise level only.

3.2 Lavrentiev's Stabilizing Scheme

In the attempt to keep the lower triangular form of A, we apply Lavrentiev's regularization procedure [21]

$$A\beta + \alpha\beta = f_\delta, \tag{3.10}$$

which allows for the following solution

$$\beta = (A + \alpha I)^{-1} f_\delta, \tag{3.11}$$

and the regularizer being $R_\alpha = (A + \alpha I)^{-1}$. The structure of the operator is maintained in this algorithm, and the penalty term can potentially increase stability of the equation by shifting the spectrum of the operator away from zero. Lavrentiev's method has been justified for non-negative and self-adjoint linear operators. Clearly, A defined in (2.8) does not satisfy these conditions, therefore, we use Lavrentiev's method with reservations. However, for some data sets, the stability stands to improve by shifting the spectrum while the structure is preserved.

3.3 The Local Regularization Approach

As opposed to Lavrentiev's and Tikhonov's regularization schemes, in the local regularization method [22] the penalty term is extracted from the original operator rather than

added to it. To illustrate the idea of local regularization, we consider a general Volterra-type integral equation

$$A\beta = f_\delta, \quad A\beta(t) := \int_0^t K(t,\tau)\beta(\tau)d\tau, \tag{3.12}$$

$A : L_2(0,T) \to L_2(0,T)$. Following [23], suppose that (3.12) is satisfied on the extended domain $[0, T+\alpha]$, $\alpha > 0$. If necessary, we can reduce the value of T. To regularize, we write this equation in the form

$$\int_0^t K(t+\rho,\tau)\beta(\tau)d\tau + \int_t^{t+\rho} K(t+\rho,\tau)\beta(\tau)d\tau = f_\delta(t+\rho), \tag{3.13}$$

$$t \in (0,T), \quad \rho \in (0,\alpha),$$

and modify (3.13) by assuming that $\beta(\tau) = \beta(t)$ for all $\tau \in [t, t+\rho]$. That results in a Volterra integral equation of the second kind

$$\int_0^t K(t+\rho,\tau)\beta(\tau)d\tau + \int_t^{t+\rho} K(t+\rho,\tau)d\tau\beta(t) = f_\delta(t+\rho), \tag{3.14}$$

where ρ can be viewed as a regularization parameter. However since f_δ is noise contaminated, it is beneficial to take its average over the interval $[t, t+\alpha]$ to approximate $f(t)$. To this end, we integrate both sides of (3.14) to obtain

$$\frac{1}{\alpha}\int_0^t\int_0^\alpha K(t+\rho,\tau)d\rho\,\beta(\tau)d\tau + \frac{1}{\alpha}\int_0^\alpha\int_t^{t+\rho} K(t+\rho,\tau)d\tau\,d\rho\,\beta(t) \tag{3.15}$$

$$= \tfrac{1}{\alpha}\int_0^\alpha f_\delta(t+\rho)d\rho, \qquad t \in (0,T).$$

From the above, one arrives at a local regularization procedure in the form

$$\mathcal{A}_\alpha\beta + \lambda_\alpha\beta = g_\alpha, \qquad \lambda_\alpha := \frac{1}{\alpha}\int_0^\alpha\int_t^{t+\rho} K(t+\rho,\tau)d\tau\,d\rho, \tag{3.16}$$

$$\mathcal{A}_\alpha\beta := \int_0^t \mathcal{K}_\alpha(t,\tau)\beta(\tau)d\tau, \quad \text{and} \quad \mathcal{K}_\alpha(t,\tau) := \tfrac{1}{\alpha}\int_0^\alpha K(t+\rho,\tau)d\rho,$$

where $\mathcal{A}_\alpha$ and g_α are the averaged operator and the averaged data, respectively. As one can see, this regularization does not change the structure of the operator. Additionally, since λ_α is allowed to be time-dependent, the penalty is more sensitive, thus helping to prevent over-regularizing of computed solutions. One concludes from (2.8) that in the case of our particular inverse problem, the kernel of the operator A does not depend on the variable t. This yields $\mathcal{A}_\alpha = A$, and the locally regularized equation takes the form

$$A\beta + \lambda_\alpha\beta = g_\alpha, \qquad g_\alpha = \frac{1}{\alpha}\int_0^\alpha f_\delta(t+\rho)d\rho, \tag{3.17}$$

$$\lambda_\alpha(t) = \frac{1}{\alpha}\int_0^\alpha \int_t^{t+\rho} K(C(\tau),\tau)d\tau d\rho = \frac{1}{\alpha}\int_0^\alpha K(C(s+t),s+t)(\alpha-s)ds. \tag{3.18}$$

3.4 Stable Numerical Differentiation with Legendre Polynomials

Our next routine for the estimation of $\beta(t)$ has been largely motivated by [24, 25], where it was argued that the singular system of the integral operator has limitations preventing it from accurately approximating the derivatives of some (even trivial) analytic functions. Based on this observation, it has been suggested that regularization algorithms applied to Volterra's integral equation of the first kind are not always beneficial for stable numerical differentiation of noise contaminated partial data. Thus in [25], it is recommended to look for some other basis, which does not have the restrictions that the singular system of the integral operator would unavoidably impose. Since in our case $C_\delta(t)$ is the cumulative number of infections from birds to humans, it results from reporting methods that may not be consistently timely and noise-free.

In this section, in place of considering Volterra's equation (2.8), we solve for $\beta(t)$ using

expressions (2.7)-(2.8):

$$\beta(t) = \frac{(C_\delta(t))'}{I_b(t)\left[\left(S_0 - \frac{\Lambda}{\mu} + C_\delta(0)\right)e^{-\mu t} + \frac{\Lambda}{\mu} - C_\delta(t) + \mu\int_0^t C_\delta(\tau)e^{-\mu(t-\tau)}d\tau\right]}. \tag{3.19}$$

Differentiation of noisy data is ill-posed since any small perturbation in the function may lead to large errors in the computed derivative. So, any measured data requires a regularization procedure to be used along with a numerical differentiation scheme. One can try to smooth $C_\delta(t)$ and differentiate in a stable manner by using finite differences with a step size being a noise-dependent regularization parameter. Alternatively, prior to differentiation, the noisy data $C_\delta(t)$ can be splined and then approximated by a finite combination of orthogonal polynomials. Then the number of polynomials can be interpreted as a reciprocal of the regularization parameter. For example, if one takes Legendre polynomials [20, 25, 26], then

$$P^{(n)}(t) = \frac{1}{2^n n!}\frac{d^n}{dt}[(t^2-1)^n],$$

which are L^2-orthogonal on $[-1, 1]$:

$$\int_{-1}^{1} P^{(n)}(t)P^{(m)}(t)\,dt = \frac{2}{2n+1}\delta^*_{n,m}.$$

The Legendre expansion of $C(t)$, re-scaled to $[-1, 1]$, is given by

$$C(t) = \sum_{n=0}^{\infty} \mathbb{C}^{(n)}P^{(n)}(t), \quad \text{with} \quad \mathbb{C}^{(n)} = \frac{2n+1}{2}\int_{-1}^{1} C(t)P^{(n)}(t)\,dt.$$

Hence, for stable numerical differentiation to be carried out, the noisy data can be filtered as

$$C_\delta(t) \approx \sum_{n=0}^{m} \mathbb{C}^{(n)}_\delta P^{(n)}(t).$$

For this approach, the regularization parameter $\alpha^*(\delta) = \frac{1}{m(\delta)}$ with m being the number of terms in the summation.

3.5 The Discrepancy Principle

The main idea of the discrepancy principle states that solving the operator equation with error less than the noise is not reasonable since that would give a solution to the noisy equation rather than the exact one. For an ill-posed problem, that is not a desirable situation, in general. Finding a regularization parameter from the discrepancy principle (DP) for a linear ill-posed problem $A\beta = f$ was first proposed by V. Morozov [1, 27, 28], who suggested computing $\alpha = \alpha(\delta, f_\delta) > 0$ so that the corresponding Tikhonov [18, 19] solution $\beta = \beta_{\alpha(\delta, f_\delta)}$, i.e. the solution to the equation

$$\alpha\beta + A^*A\beta = A^*f_\delta, \tag{3.20}$$

satisfies the condition

$$||A\beta_{\alpha(\delta,f_\delta)} - f_\delta|| = \delta. \tag{3.21}$$

It has been established in [1, 27] that

(a) $\beta_{\alpha(\delta,f_\delta)} \longrightarrow \hat{\beta}$ as $\delta \longrightarrow 0$ and

(b) $||\beta_{\alpha(\delta,f_\delta)} - \hat{\beta}|| \leq 2\sqrt{\delta E}$ if $\hat{\beta} = A^*\hat{z} \in A^*(H)$ with $||\hat{z}|| \leq E$,

provided that A is a linear, compact and one-to-one operator with the dense range in a Hilbert space H, $||f - f_\delta|| \leq \delta \leq ||f_\delta||$, and E is a constant. It was also shown in [1, 27] that for Tikhonov's regularization, the DP guarantees optimal convergence rates up to a certain saturation point (i.e. for $\nu \in (0, 1/2]$ where $\hat{\beta} \in \mathcal{R}((A^*A)^\nu)$) and sub-optimal rates above that point. This indicates that the order of convergence $O(\sqrt{\delta})$ is the best possible for the Tikhonov solution with a stabilizing parameter selected by the DP. For stochastic white noise, in a finite-dimensional setting with the error in each element of f_δ having standard deviation δ, a similar estimate holds "in expectation" as the size of the space approaches infinity [29, 30].

In [31–33], the discrepancy principle in the form

$$\|A\beta_{\alpha(\delta,f_\delta)} - f_\delta\| = \delta^{1-\epsilon}, \quad 0 < \epsilon < 1 \tag{3.22}$$

has been justified for the case of a regularized solution constructed through a special filter

$$R_\alpha f_\delta := \theta(A^*A, \alpha)A^* f_\delta, \tag{3.23}$$

where $\theta = \theta(\sigma, \alpha)$ with $\alpha > 0$, $\sigma \in [0, ||A||^2]$, and the following conditions hold

$$\sup_{\sigma\in[0,||A||^2]} |\theta(\sigma,\alpha)\sqrt{\sigma}| \leq C_1\alpha^{-1/2}, \tag{3.24}$$

$$\sup_{\sigma\in[0,||A||^2]} |\theta(\sigma,\alpha)\sigma - 1|\sigma^p \leq C_2(p)\alpha^p, \quad p \geq \frac{1}{2}, \tag{3.25}$$

$$\sup_{\sigma\in[0,||A||^2]} |\theta(\sigma,\alpha)\sigma - 1| \leq C_3. \tag{3.26}$$

Here C_1 and C_3 are constants and C_2 is a function of p. This generalization allowed the use of a noisy operator (see also [34]), as well as the development of saturation-free regularizing algorithms based on the DP. For example, the DP for the spectral cut-off filter

$$\theta(\sigma,\alpha) := \begin{cases} \frac{1}{\sigma}, & \sigma \geq \alpha, \\ 0, & 0 \leq \sigma < \alpha, \end{cases} \tag{3.27}$$

yields optimal convergence rates [28, 35] for any $\nu > 0$. Additionally, for a self-adjoint operator A, discrepancy principle (3.22) has been investigated for a special regularizer $R_\alpha f_\delta := \theta(A, \alpha)f_\delta$, with $\theta = \theta(\sigma, \alpha)$ defined on the spectrum of A rather than A^*A in its first variable [31, 32]. This family of methods includes Lavrentiev's regularization [21] among other procedures. Since then there has been an enormous effort to further develop the original discrepancy principle while still retaining optimal orders of convergence both for linear [36–42] and nonlinear [19, 40, 42–45] inverse problems.

In the numerical experiments that follow, the discrepancy principle (3.22) will be applied

to all methods consistently. The results of these experiments have motivated the formulation and justification of a theorem on Abstract Discrepancy Principle applied to regularization strategies without any specific structure.

PART 4 Numerical Results

In what follows, we present two sets of experiments. In the first subsection, we test the algorithms and the stopping rule on simulated noisy data and compare our reconstructions with the exact analytic solution. Then, numerical experiments with real data are conducted and analyzed in the next subsection.

4.1 Test Experiments with Simulated Data

For the initial testing and comparison, we conduct numerical simulations for $C(t)$, $\beta(t)$ and $I_b(t)$ given in (2.9)-(2.11) As a reference point, for the numerical differentiation approach, we also look at

$$C'(t) = \eta - \sin(\gamma t)\exp(-\sigma t). \tag{4.1}$$

Employing all four regularization methods to noisy data values results in the graph shown in Figure **??**. Using the discrepancy principle (3.22) with $\varepsilon = 0.13$ for $\delta = 0.25\|f\|$, we stop the regularization when $\|A\beta_{\alpha^*} - f_\delta\| \leq \delta^{1-\varepsilon}$ for the first time. Table 4.1 gives the progression of discrepancy for each method pending the changes in the appropriate regularization parameter. From Figure **??**, we see that all methods improve the un-regularized solution. With the exception of the Lavrentiev stabilizing method, the other three schemes do a credible job of approximating actual $\beta(t)$ for this model function. The unsatisfactory performance of Lavrentiev's regularization is not surprising here, since the A is not self-adjoint in the case of a Volterra type integral operator.

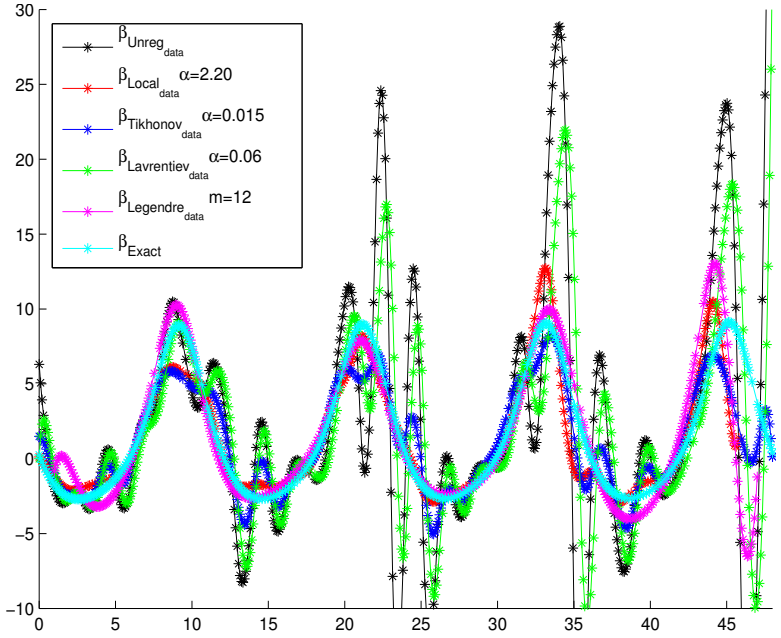

Figure (4.1) *Computation of β using Discrepancy Principle - Simulated data*

Table (4.1) *Finding regularization parameter by Discrepancy Principle - Simulated data*

Tikhonov		Lavrentiev		Local		Legendre	
α	Discrepancy	α	Discrepancy	α	Discrepancy	m	Discrepancy
0.030	0.37567	0.075	0.341159	2.50	0.326267	9	0.583911
0.025	0.353222	0.070	0.326477	2.40	0.316249	10	0.432386
0.020	0.327423	0.065	0.311227	2.30	0.3063	11	0.430368
0.015	0.296755	0.060	0.295342	2.20	0.296307	12	0.241102

4.2 Reconstruction with Real Data

In this subsection, we experiment with two different data sets collected by N. Tuncer [13]. In the first case, we work with data obtained from all countries where highly pathogenic avian influenza (HPAI) of subtype H5N1 has been observed. In the second case, we focus on Indonesia, one of the countries where most cases are seen, and use data for Indonesia only. In both experiments, we evaluate the time-dependent transmission rate $\beta(t)$ of the avian influenza model using cumulative number of infected H5N1 human cases and the number of infected poultry.

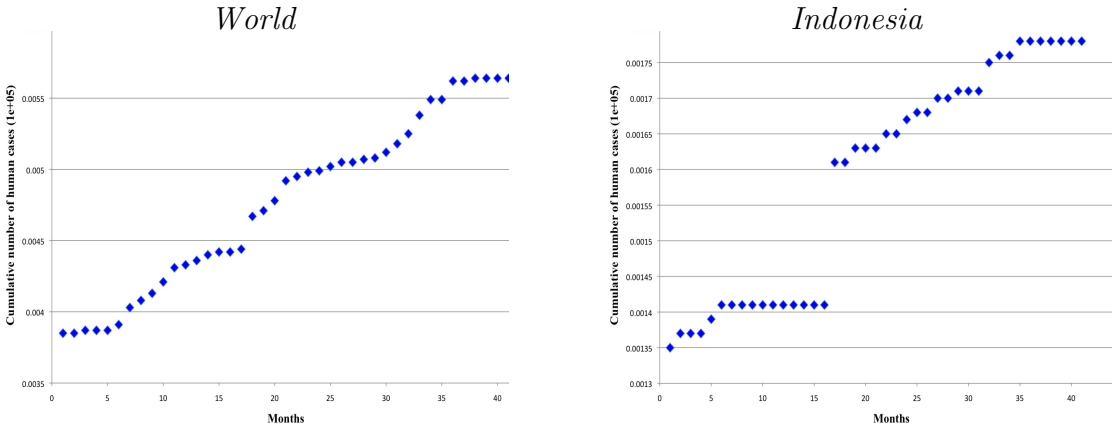

Figure (4.2) *Cumulative human infections H5N1 [2]*

Table (4.2) *Cumulative number of confirmed H5N1 human cases [2]*
W=World $(07/08 - 12/11)$ *and I=Indonesia* $(07/08 - 06/11)$

Month	W	I	Month	W	I	Month	W	I	Month	W	I
1	385	135	12	433	141	23	498	165	34	549	176
2	385	135	13	436	141	24	499	165	35	549	176
3	387	137	14	440	141	25	502	167	36	562	178
4	387	137	15	442	141	26	505	168	37	562	–
5	387	137	16	442	141	27	505	168	38	564	–
6	391	139	17	444	141	28	507	170	39	564	–
7	403	141	18	467	161	29	508	170	40	564	–
8	408	141	19	471	161	30	512	171	41	564	–
9	413	141	20	478	163	31	518	171	42	564	–
10	421	141	21	492	163	32	525	171			
11	431	141	22	495	163	33	538	175			

Current world population is estimated to be 7 billion. Data related to humans is given in units of 10^5 individuals, and so human population is set to 70,000 [11, 12]. The average life expectancy is approximately 70 years and t is given in months, therefore $\mu = 1/(70 * 12)\text{month}^{-1}$. Since in a pre-pandemic scenario the total world population remains essentially unchanged, the estimate above tells us $\frac{\Lambda}{\mu} \approx 70,000$, which results in $\Lambda = 1000/12$ births per month given in 10^5 individuals. We assume that initially all humans are susceptible so that $S(0) = 70,000$ for the world population experiment. For Indonesia, $S(0) = 242,325,638/10^5$.

The data for cumulative number of H5N1 human cases is given by the World Health Organization (WHO) web site. However, at that web site, only data from August 2011 to

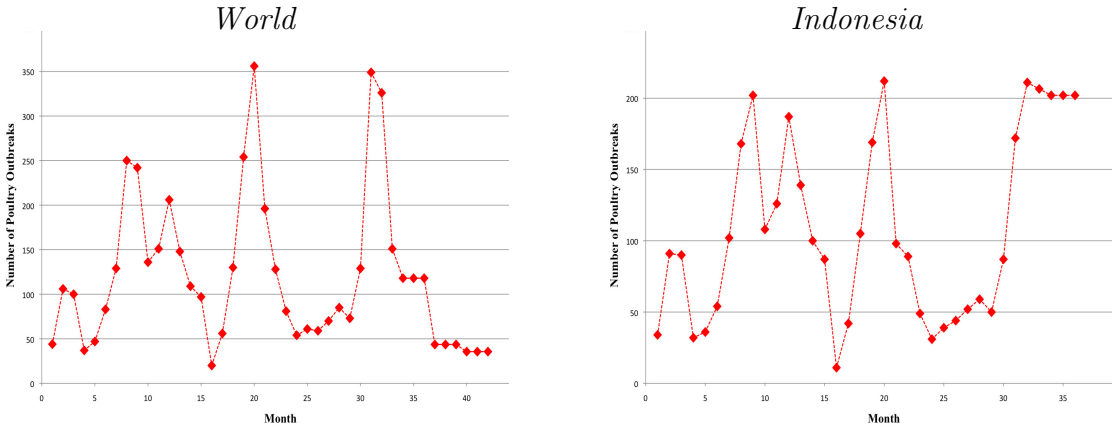

Figure (4.3) *Poultry Outbreaks [3]*

Table (4.3) *Poultry Outbreaks [3]*
W=World $(07/08-12/11)$ *and I=Indonesia* $(07/08-06/11)$.

Month	W	I	Month	W	I	Month	W	I	Month	W	I
1	44	34	12	206	187	23	81	49	34	118	202
2	106	91	13	148	139	24	54	31	35	118	202
3	100	90	14	109	100	25	61	39	36	118	202
4	37	32	15	97	87	26	59	44	37	43.6	–
5	47	36	16	20	11	27	70	52	38	43.6	–
6	83	54	17	56	42	28	85	59	39	43.6	–
7	129	102	18	130	105	29	73	50	40	35.6	–
8	250	168	19	254	169	30	129	87	41	35.6	–
9	242	202	20	356	212	31	349	172	42	35.6	–
10	136	108	21	196	98	32	326	211			
11	151	126	22	128	89	33	151	206.5*			

August 2012 was available. After several email communications, N. Tuncer obtained the archived data $C(t)$ from January 2004 to December 2011 (see Table 4.2 and Figure 4.2).

The Food and Agriculture Organization of the United Nations (FAO) provides information about the number of poultry outbreaks starting from July 2008 to December 2011. This information is presented in Table 4.3 and Figure 4.3. The total poultry population was 20.4 billion poultry units worldwide in 2008 [3], and data related to poultry is given in units of 10^7. For the model 2.8, we require the number of infected poultry for each geographical set. The FAO reports the total number of outbreaks. Thus, the number of infected poultry, $I_b(t)$,

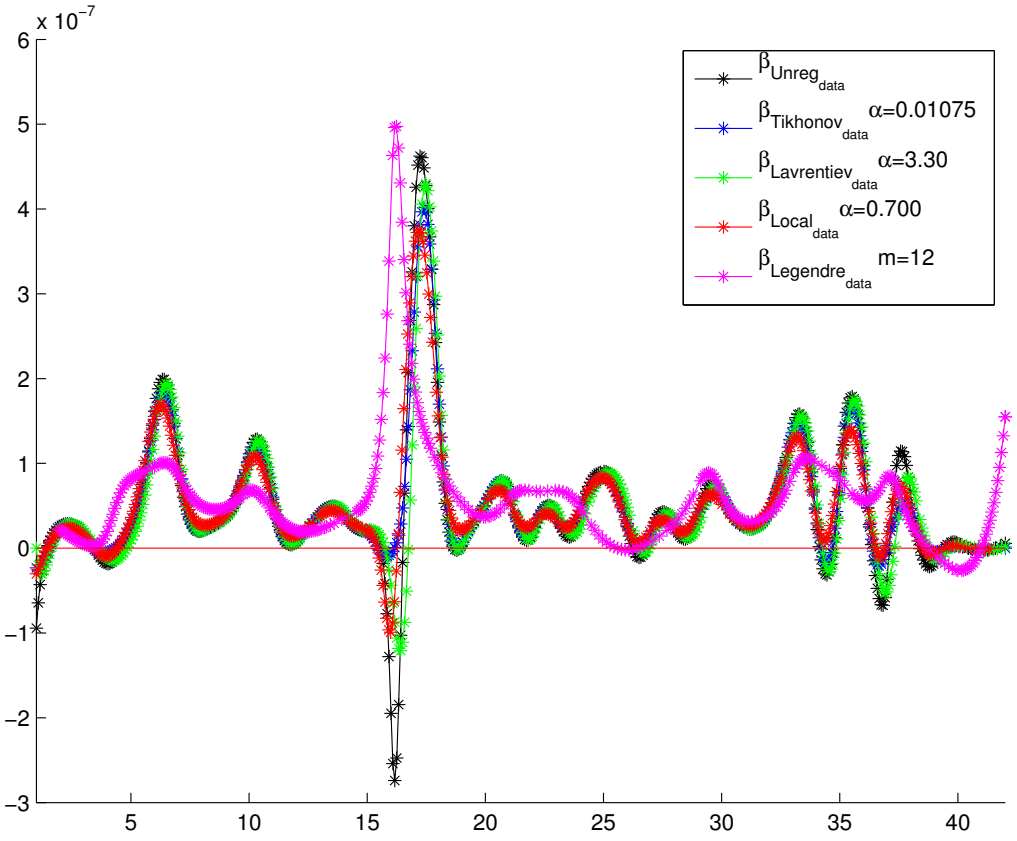

Figure (4.4) *Determining $\beta(t)$ using various regularization methods - World*

Table (4.4) *Determination of Regularization Parameter by Discrepancy - World.*

Tikhonov		Lavrentiev		Local		Legendre	
α	Discrepancy	α	Discrepancy	α	Discrepancy	m	Discrepancy
0.0125	0.0054325	3.575	0.0053487	0.77000	0.0055791	8	0.0090900
0.012	0.0053127	3.5	0.0052477	0.75000	0.0054104	9	0.0072021
0.0115	0.0051896	3.425	0.0051474	0.73000	0.0052426	10	0.0055490
0.011	0.0050631	3.35	0.0050479	0.71000	0.0050756	11	0.0052604
0.01075	0.0049984	3.3	0.0049820	0.70000	0.0049922	12	0.0049884

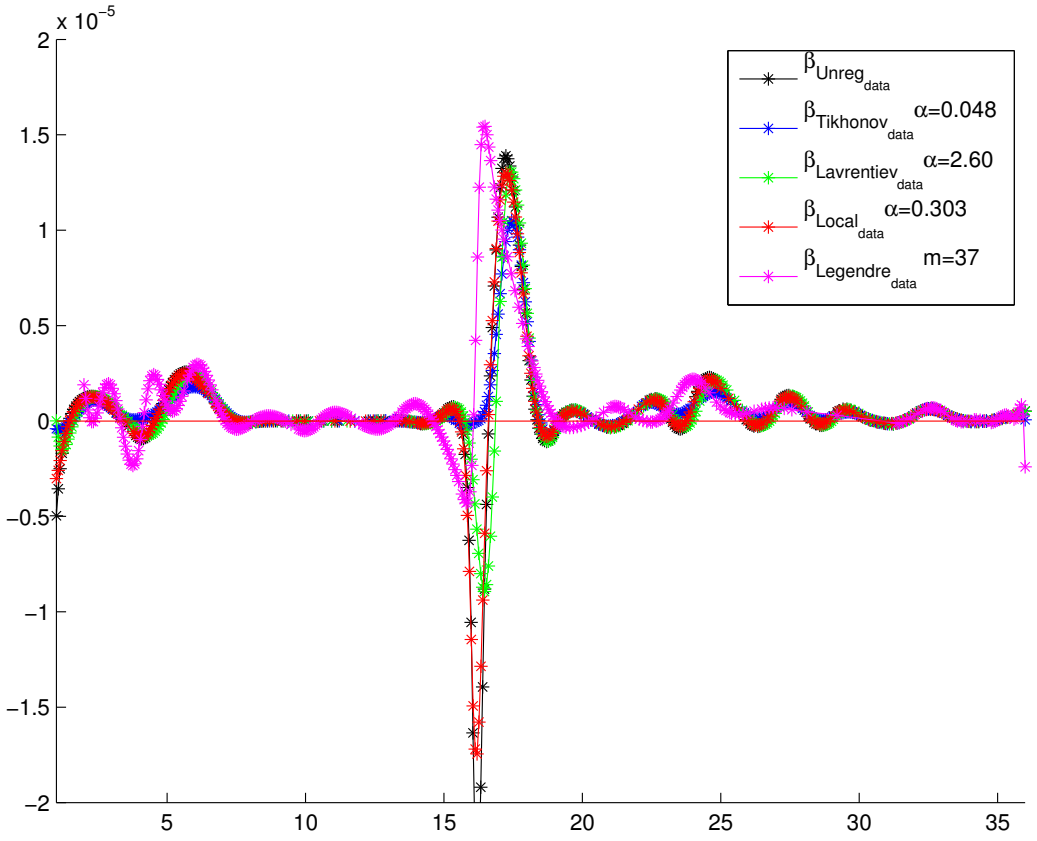

Figure (4.5) *Determining $\beta(t)$ using various regularization methods - Indonesia*

Table (4.5) *Determination of Regularization Parameter by Discrepancy - Indonesia.*

Tikhonov		Lavrentiev		Local		Legendre	
α	Discrepancy	α	Discrepancy	α	Discrepancy	m	Discrepancy
0.055	0.0056013	2.95	0.0055021	0.313	0.0053755	33	0.0062549
0.053	0.0054053	2.85	0.0053413	0.31	0.0052539	34	0.0059120
0.051	0.0052115	2.75	0.0051784	0.307	0.0051338	35	0.0058171
0.049	0.0050199	2.65	0.0050133	0.304	0.0050152	36	0.0057698
0.048	0.0049250	2.6	0.0049299	0.303	0.0049760	37	0.0041646

can be computed by estimating the average poultry farm size and multiplying this by the number of outbreaks. Worldwide farm size varies from hundreds to several thousands in the countries affected by the H5N1 virus. We take the average worldwide farm size to be 1000. Poultry farm size in Indonesia has a much smaller range: over 97.5% of farms in Indonesia are in the range of 222 to 834. We take the weighted average farm size for Indonesia to be 682 [13].

For each method, we maintain the same discretization step size, $h = 0.01$, to generate the lower triangular matrix A. Due to the values used in constructing A, we are presented with different scales for each data set. To facilitate comparison, the following procedure is used. For Indonesia data, the matrix A has a maximum value of 0.7244; for the World data set, this maximum is 26.3448. The minimum in both cases is 0. Our regularization parameter for Lavrentiev's stabilizing algorithm is scaled through dividing by this maximum in both cases. The effect of this division is to present α as the shifting of the spectrum of the matrix whose values lie in the interval $[0, 1]$. For the Tikhonov regularization method, we employ a similar procedure. In this case, we use maximum and minimum entries of A^*A. For Indonesia, this maximum is 566.9428; for World it is 1.391×10^6. The minimum values for this matrix are 0.0138 and 3.0028, respectively, and can be disregarded in the scaling. Finally, for local regularization, α is the length of the interval, where the unknown function β remains unchanged.

In the numerical experiments using simulated data, all methods, with the exception of Lavrentiev's regularization algorithm, reconstructed the model solution in a stable and accurate manner. In contrast, with real data, Lavrientiev's algorithm was among the consistent performers. Tikhonov's algorithm was the top performer with all data sets, both simulated and real, and was best at reducing the significant negative component of the unregularized solution occurring at the approximate middle range of the graphs for real data (Figures 4.4 and 4.5). We do not expect the bird to human transmission rate to be negative. Stable numerical differentiation with Legendre polynomials performs poorly for real data, while for the simulated data it is one of the best. We suspect this has to do with discontinuity of

real data, especially for Indonesia, which was not reporting infected human cases in calendar year 2009, and then reported them all at once at the end of December 2009 [3, 13]. For all methods, this reporting anomaly results in an *artificial* jump in the number of H5N1 infected human cases (see Figure 4.2). As a result, all regularizing algorithms show a rapid increase in the bird-to-human transmission rate around December 2009 that is not entirely appropriate. The remaining oscillations can be attributed to seasonality due to temperature fluctuations, environmental changes, and other natural factors.

PART 5 Abstract Discrepancy Principle

In this section, we formulate the main theoretical result of this thesis, Theorem 2.1 [46], for a linear irregular operator equation

$$A\beta = f, \quad A : \mathcal{X} \to \mathcal{Y}, \tag{5.1}$$

with noise contaminated data $f_\delta,\ ||f - f_\delta|| \leq \delta$.

Theorem 2.1. Let the operator A between two Banach spaces $\mathcal{X}$ and $\mathcal{Y}$ be linear, one-to-one and bounded with a dense range. Suppose also that in the noise-free case, the exact equation, $A\beta = f$, is solvable and $\hat{\beta}$ is a solution. Assume that a family of operators $R_\alpha : \mathcal{Y} \to \mathcal{X}$, $\alpha > 0$, generates a regularization strategy in the following sense

$$\lim_{\alpha \to 0^+} R_\alpha A\beta = \beta \quad \text{for all } \beta \in \mathcal{X}, \tag{5.2}$$

and the operator function R_α is strongly continuous with respect to α for any $\alpha > 0$ such that

$$\lim_{\alpha \to \infty} ||AR_\alpha|| = 0, \quad \sup_{\alpha \geq 0} ||I - AR_\alpha|| = c < \infty, \quad \sup_{\alpha \geq 0} ||R_\alpha||\,||(A - AR_\alpha A)\hat{\beta}|| = D < \infty, \tag{5.3}$$

where c and D are constants. Let f be given by its δ-approximation

$$||f - f_\delta|| \leq \delta, \tag{5.4}$$

and a regularized solution to (5.1) be defined as $\beta_\alpha := R_\alpha f_\delta$. Suppose $\alpha = \alpha(\delta, f_\delta)$ is selected by the Discrepancy Principle:

$$||A\beta_{\alpha(\delta,f_\delta)} - f_\delta|| = \delta^{1-\varepsilon}, \quad 1 > \varepsilon > 0, \tag{5.5}$$

where we assume $\delta^{1-\varepsilon} < ||f_\delta||$. Then $\alpha = \alpha(\delta, f_\delta)$ satisfying equation (5.5) exists, and

$$\lim_{\delta\to 0} ||R_{\alpha(\delta,f_\delta)} f_\delta - \hat{\beta}|| = 0. \tag{5.6}$$

Proof of Theorem 2.1. First, we verify that

$$\lim_{\alpha\to\infty} ||A\beta_\alpha - f_\delta|| = ||f_\delta||, \tag{5.7}$$

$$\lim_{\alpha\to 0^+} ||A\beta_\alpha - f_\delta|| = 0, \tag{5.8}$$

$$\text{and the mapping} \quad \alpha \longrightarrow ||A\beta_\alpha - f_\delta|| \quad \text{is continuous.} \tag{5.9}$$

Indeed, since by (5.3) $\lim_{\alpha\to\infty} ||AR_\alpha|| = 0$, one concludes that

$$\lim_{\alpha\to\infty} ||A\beta_\alpha - f_\delta|| \le \lim_{\alpha\to\infty} \Big\{ ||AR_\alpha||\, ||f_\delta|| + ||f_\delta|| \Big\} = ||f_\delta||,$$

and

$$\lim_{\alpha\to\infty} ||A\beta_\alpha - f_\delta|| \ge \lim_{\alpha\to\infty} \Big\{ ||f_\delta|| - ||AR_\alpha||\, ||f_\delta|| \Big\} = ||f_\delta||,$$

which yields (5.7). Now fix any $\epsilon > 0$. Since the range of A is dense in $\mathcal{Y}$, there exists $\tilde{\beta} \in \mathcal{Y}$ with $||A\tilde{\beta} - f_\delta|| \le \frac{\epsilon}{3(c+1)} \le \frac{\epsilon}{3}$. By (5.2), there is $\bar{\alpha} > 0$, such that for any $\alpha \in (0, \bar{\alpha}]$, $||R_\alpha A\tilde{\beta} - \tilde{\beta}|| \le \frac{\epsilon}{3||A||}$. Hence, for any $\alpha \in (0, \bar{\alpha}]$, one gets

$$||A\beta_\alpha - f_\delta|| = ||AR_\alpha(f_\delta - A\tilde{\beta} + A\tilde{\beta}) - f_\delta)|| \le ||AR_\alpha||\, ||f_\delta - A\tilde{\beta}||$$

$$+||A(R_\alpha A\tilde{\beta} - \tilde{\beta} + \tilde{\beta}) - f_\delta|| \le ||AR_\alpha||\, ||f_\delta - A\tilde{\beta}|| + ||A||\, ||R_\alpha A\tilde{\beta} - \tilde{\beta}||$$

$$+||A\tilde{\beta} - f_\delta|| \le (c+1)\frac{\epsilon}{3(c+1)} + ||A||\frac{\epsilon}{3||A||} + \frac{\epsilon}{3} \le \epsilon.$$

Continuity of $\phi(\alpha) := ||A\beta_\alpha - f_\delta||$ follows from strong continuity of R_α. Since $0 \le \delta^{1-\varepsilon} < ||f_\delta||$, Equation (5.5) is solvable for α.

At the next step, we prove relationship (5.6), the main part of the theorem. One has

$$||R_{\alpha(\delta,f_\delta)}f_\delta - \hat{\beta}|| \le ||R_{\alpha(\delta,f_\delta)}||\,\delta + ||R_{\alpha(\delta,f_\delta)}A\hat{\beta} - \hat{\beta}||.$$

Let us show that

$$||R_{\alpha(\delta,f_\delta)}A\hat{\beta} - \hat{\beta}|| \to 0 \quad \text{as} \quad \delta \to 0. \tag{5.10}$$

Suppose

$$\limsup_{\delta\to 0} \alpha(\delta, f_\delta) = \tilde{\alpha}. \tag{5.11}$$

From (5.7), one derives that $\tilde{\alpha} < \infty$. Indeed, assume the opposite. Let $\{\delta_m\}$, $\lim_{m\to\infty}\delta_m = 0$, be a sequence such that $\lim_{m\to\infty}\alpha_m = \infty$, where $\alpha_m := \alpha(\delta_m, f_{\delta_m})$. Then (5.3) and (5.11) yield

$$\lim_{m\to\infty} ||A\beta_{\alpha_m} - f_{\delta_m}|| \ge \lim_{\delta\to 0}||f_\delta|| - \lim_{\alpha\to\infty}||A\beta_\alpha|| = ||f||.$$

On the other hand, by (5.5) $\lim_{\delta\to 0}||A\beta_{\alpha(\delta,f_\delta)} - f_\delta|| = \lim_{\delta\to 0}\delta^{1-\varepsilon} = 0$. This contradiction shows that $\tilde{\alpha} < \infty$. Two cases are possible:

(i) $\tilde{\alpha} = 0$. Then (5.10) follows from (5.2).

(ii) $0 < \tilde{\alpha} < \infty$. Take an arbitrary sequence $\{\delta_n\}$ such that $\lim_{n\to\infty}\delta_n = 0$. If the sequence $\{\alpha_n\}$, $\alpha_n := \alpha(\delta_n, f_{\delta_n})$, is convergent then $\lim_{n\to\infty}\alpha_n = \hat{\alpha} \le \tilde{\alpha}$. By (5.3), one obtains

$$\lim_{n\to\infty}||A\beta_{\alpha_n} - f_{\delta_n}|| = \lim_{n\to\infty}||AR_{\alpha_n}f_{\delta_n} - f_{\delta_n}|| \le \lim_{n\to\infty}\Big\{||(AR_{\alpha_n} - I)f|| + ||AR_{\alpha_n} - I||\delta_n\Big\} = ||AR_{\hat{\alpha}}f - f||,$$

and

$$\lim_{n\to\infty}||A\beta_{\alpha_n} - f_{\delta_n}|| \ge \lim_{n\to\infty}\Big\{||(AR_{\alpha} - I)f|| - ||AR_{\alpha_n} - I||\delta_n\Big\} = ||AR_{\hat{\alpha}}f - f||.$$

At the same time, by (5.5)

$$\lim_{n\to\infty} ||A\beta_{\alpha_n} - f_{\delta_n}|| = \delta_n^{1-\varepsilon} = 0.$$

Therefore, $||AR_{\hat{\alpha}}f - f|| = 0$ and $R_{\hat{\alpha}}f = \hat{\beta}$. So, from the strong continuity of the operator R_α,

$$\lim_{n\to\infty} ||R_{\alpha_n}A\hat{\beta} - \hat{\beta}|| = \lim_{n\to\infty} ||R_{\alpha_n}f - R_{\hat{\alpha}}f|| = 0.$$

Hence, if the sequence $\{\alpha_n\}$ is convergent, then $\lim_{\delta\to 0} ||R_{\alpha(\delta,f_\delta)}A\hat{\beta} - \hat{\beta}|| = 0$. If $\{\alpha_n\}$ does not converge, then $||R_{\alpha(\delta,f_\delta)}A\hat{\beta} - \hat{\beta}||$ still goes to zero. Indeed, assume

$$\limsup_{n\to\infty} ||R_{\alpha_n}A\hat{\beta} - \hat{\beta}|| = d, \quad 0 < d \leq \infty. \tag{5.12}$$

There is a subsequence $\{\alpha_k\}$, $\alpha_k := \alpha(\delta_{n_k}, f_{\delta_{n_k}})$, such that $\lim_{k\to\infty} ||R_{\alpha_k}A\hat{\beta} - \hat{\beta}|| = d$. Now, if one takes $\{\alpha_{k_j}\}$, $\alpha_{k_j} := \alpha(\delta_{n_{k_j}}, f_{\delta_{n_{k_j}}})$, which converges to some $\breve{\alpha} \leq \tilde{\alpha}$, then, by the above argument, $\lim_{j\to\infty} ||R_{k_j}A\hat{\beta} - \hat{\beta}|| = 0$. On the other hand, by (5.12) this limit is equal to d. This contradiction proves that (5.10) holds.

The last step of the proof is to check that $\lim_{\delta\to 0} ||R_{\alpha(\delta,f_\delta)}||\,\delta = 0$. Conditions (5.3)-(5.5) imply

$$||A\hat{\beta} - AR_\alpha A\hat{\beta}|| \geq ||AR_\alpha f_\delta - f_\delta|| - ||I - AR_\alpha||\,||f - f_\delta|| \geq \delta^{1-\varepsilon} - c\delta.$$

From the above, one derives

$$D\delta^\varepsilon \geq ||R_{\alpha(\delta,f_\delta)}||\,||A\hat{\beta} - AR_{\alpha(\delta,f_\delta)}A\hat{\beta}||\,\delta^\varepsilon \geq \delta(1 - c\delta^\varepsilon)||R_{\alpha(\delta,f_\delta)}||. \tag{5.13}$$

Estimate (5.13) completes the proof. □

Remark 2.2 In order to verify the third condition in (5.3), one has to use some *a priori* information about the true solution $\hat{\beta}$. Oftentimes, the mere existence is enough, while in some cases, additional requirements are to be imposed.

Remark 2.3 For a number of important algorithms, the regularization parameter α takes

discrete values. For instance, when stable numerical differentiation with Legendre polynomials is used, $\alpha = 1,\ 1/2,\ 1/3,...1/n,...$. Suppose α^* and α^{**} are the two values of α such that

$$||A\beta_{\alpha^*(\delta,f_\delta)} - f_\delta|| \leq \delta^{1-\varepsilon} < ||A\beta_{\alpha^{**}(\delta,f_\delta)} - f_\delta||. \tag{5.14}$$

Let us take the value of α equal to α^*. Clearly, $\lim_{\delta\to 0} ||R_{\alpha^*(\delta,f_\delta)}A\hat{\beta} - \hat{\beta}|| = 0$, since in the proof of this part the identity $||A\beta_{\alpha(\delta,f_\delta)} - f_\delta|| = \delta^{1-\varepsilon}$ can easily be replaced with the estimate from above, $||A\beta_{\alpha^*(\delta,f_\delta)} - f_\delta|| \leq \delta^{1-\varepsilon}$. In order to justify

$$\lim_{\delta\to 0} ||R_{\alpha^*(\delta,f_\delta)}||\delta = 0, \tag{5.15}$$

let us additionally assume that for any two consecutive values of α, $\alpha^{(1)} > \alpha^{(2)}$, one has

$$\frac{||R_{\alpha^{(1)}}||}{||R_{\alpha^{(2)}}||} \geq \nu > 0. \tag{5.16}$$

Then, following (5.13), one concludes

$$D\delta^\varepsilon \geq ||R_{\alpha^{**}(\delta,f_\delta)}||\,||A\hat{\beta} - AR_{\alpha^{**}(\delta,f_\delta)}A\hat{\beta}||\,\delta^\varepsilon \geq \delta(1-c\delta^\varepsilon)||R_{\alpha^{**}(\delta,f_\delta)}||$$

$$= \delta(1-c\delta^\varepsilon)||R_{\alpha^*(\delta,f_\delta)}||\frac{||R_{\alpha^{**}(\delta,f_\delta)}||}{||R_{\alpha^*(\delta,f_\delta)}||} \geq \delta(1-c\delta^\varepsilon)||R_{\alpha^*(\delta,f_\delta)}||\nu, \tag{5.17}$$

which yields (5.15).

This observation serves as an important practical tool even for the algorithms where the regularization parameter α is continuous, since in the process of numerical implementation it allows verification of condition (5.5) on a discrete collection of grid points $\{\alpha_i\}$, $i = 1, 2, ..., I$, (and satisfy the inequality rather than identity) without solving nonlinear equation (5.5) by, say, Newton's method.

If the first condition in (5.3) no longer makes sense in a discrete case (see subsection 3.4), then one can assume that $R_\alpha = 0$ for sufficiently large values of α.

PART 6 Conclusions and Future Work

As one can see from Figure **??** and Table 4.1, with the exception of the Lavrentiev regularization, the three remaining schemes provide a very accurate and stable reconstruction of the model solution. The Lavrentiev method is not justified in our case, and we essentially take our chances with it. However, for the real data, Lavreniev's reconstruction is consistent with other regularized solutions and seems to perform satisfactory. The Tikhonov stabilizing algorithm is a clear winner in case of the real data (a textbook example of what regularization should and should not do). Surprisingly, stable numerical differentiation with Legendre polynomials fails for real data, while for for the simulated data it is one of the best. We suspect it has to do with discontinuity of real data, especially for Indonesia, which was not reporting infected human cases in calendar year 2009, and then reported them all at once at the end of December 2009 [3, 13]. This results in an *artificial* jump in the number of H5N1 infected human cases (see Figure 4.2). Consequently, all regularizing algorithms show a rapid increase in the bird-to-human transmission rate around December 2009 that is not entirely accurate, as shown in Figures 4.4 and 4.5. The remaining oscillations can be attributed to seasonality due to temperature fluctuations, environmental changes, and other natural factors. The next step in this approach will be to generalize ADP in the case of nonlinear ill-posed problems and to further explore its numerical efficiency.

REFERENCES

[1] V. A. Morozov, "Choice of parameter for the solution of functional equations by the regularization method," in *Soviet Math. Doklady*, vol. 8, 1967, pp. 1000–1003.

[2] "Confirmed human cases of avian influenza A(H5N1)," World Health Organization. [Online]. Available: http://www.who.int/influenza/human_animal_interface/H5N1_cumulative_table_archives/en/index.html

[3] Food and Agriculture Organization of the United Nations, "FAO statistics, CountrySTAT," www.fao.org.

[4] D. Kaye and C. R. Pringle, "Avian influenza viruses and their implication for human health," *Clinical infectious diseases*, vol. 40, no. 1, pp. 108–112, 2005.

[5] R. Webster, K. Shortridge, and Y. Kawaoka, "Influenza: interspecies transmission and emergence of new pandemics," *FEMS Immunology & Medical Microbiology*, vol. 18, no. 4, pp. 275–279, 1997.

[6] D. M. Morens and A. S. Fauci, "The 1918 influenza pandemic: insights for the 21st century," *Journal of Infectious Diseases*, vol. 195, no. 7, pp. 1018–1028, 2007.

[7] R. J. Garten, C. T. Davis, C. A. Russell, B. Shu, S. Lindstrom, A. Balish, W. M. Sessions, X. Xu, E. Skepner, V. Deyde *et al.*, "Antigenic and genetic characteristics of swine-origin 2009 A (H1N1) influenza viruses circulating in humans," *Science*, vol. 325, no. 5937, pp. 197–201, 2009.

[8] J. Oxford, "Influenza a pandemics of the 20th century with special reference to 1918: virology, pathology and epidemiology," *Reviews in medical virology*, vol. 10, no. 2, pp. 119–133, 2000.

[9] S. Iwami, Y. Takeuchi, A. Korobeinikov, and X. Liu, "Prevention of avian influenza epidemic: What policy should we choose?" *Journal of theoretical biology*, vol. 252, no. 4, pp. 732–741, 2008.

[10] S. Iwami, Y. Takeuchi, and X. Liu, "Avian flu pandemic: Can we prevent it?" *Journal of theoretical biology*, vol. 257, no. 1, pp. 181–190, 2009.

[11] N. Tuncer and M. Martcheva, "Modeling seasonality in avian influenza H5N1," *Journal of Biological Systems*, vol. 21, no. 04, 2013.

[12] M. Martcheva, "Avian influenza: Modeling and implications for control," Math. Mod. Nat. Phenom., to appear.

[13] A. Smirnova and N. Tuncer, "Estimating time-dependent transmission rate of avian influenza via stable numerical algorithm," *Journal of Inverse and Ill-Posed Problems*, vol. 0, no. ISSN (Online) 1569–3945, 2013.

[14] S. Iwami, Y. Takeuchi, and X. Liu, "Avian–human influenza epidemic model," *Mathematical biosciences*, vol. 207, no. 1, pp. 1–25, 2007.

[15] K. I. Kim, Z. Lin, and L. Zhang, "Avian-human influenza epidemic model with diffusion," *Nonlinear Analysis: Real World Applications*, vol. 11, no. 1, pp. 313–322, 2010.

[16] J. Lucchetti, M. Roy, and M. Martcheva, "An avian influenza model and its fit to human avian influenza cases," *Advances in Disease Epidemiology*, pp. 1–30, 2009.

[17] A. Kirsch, *An introduction to the mathematical theory of inverse problems*. Springer, 2011, vol. 120.

[18] A. N. Tikhonov, *Numerical methods for the solution of ill-posed problems*. Kluwer Academic Publishers, 1995.

[19] A. Tikhonov, A. Leonov, and A. Yagola, "Nonlinear ill-posed problems," *Nonlinear ill-posed problems, London: Chapman & Hall, 1998, 2 vols. Applied mathematics and mathematical computation, v. 14, ISBN 0412786605*, vol. 1, 1998.

[20] Z. Zhao, "A truncated legendre spectral method for solving numerical differentiation," *International Journal of Computer Mathematics*, vol. 87, no. 14, pp. 3209–3217, 2010.

[21] M. M. Lavrent'ev, V. G. Romanov, and S. P. Shishatskiĭ, *Ill-posed problems of mathematical physics and analysis*. AMS Bookstore, 1986, vol. 64.

[22] P. K. Lamm, "Full convergence of sequential local regularization methods for volterra inverse problems," *Inverse problems*, vol. 21, no. 3, p. 785, 2005.

[23] C. D. Brooks and P. K. Lamm, "A generalized approach to local regularization of linear volterra problems in lp spaces," *Inverse Problems*, vol. 27, no. 5, p. 055010, 2011.

[24] J. Flemming, B. Hofmann, and P. Mathé, "Sharp converse results for the regularization error using distance functions," *Inverse Problems*, vol. 27, no. 2, p. 025006, 2011.

[25] S. Lu, V. Naumova, and S. V. Pereverzev, "Legendre polynomials as a recommended basis for numerical differentiation in the presence of stochastic white noise," *Journal of Inverse and Ill-posed Problems*, vol. 21, no. 2, pp. 193–216, 2013.

[26] T. Dolgopolova and V. K. Ivanov, "On numerical differentiation," *USSR Computational Mathematics and Mathematical Physics*, vol. 6, no. 3, pp. 223–232, 1966.

[27] V. A. Morozov, "The error principle in the solution of operational equations by the regularization method," *USSR Computational Mathematics and Mathematical Physics*, vol. 8, no. 2, pp. 63–87, 1968.

[28] V. A. Morozov, Z. Nashed, and A. Aries, *Methods for solving incorrectly posed problems*. Springer-Verlag New York, 1984.

[29] C. R. Vogel, *Computational methods for inverse problems*. Siam, 2002, vol. 10.

[30] F. Bauer and M. A. Lukas, "Comparing parameter choice methods for regularization of ill-posed problems," *Mathematics and Computers in Simulation*, vol. 81, no. 9, pp. 1795–1841, 2011.

[31] A. B. Bakushinskii, "Principle of the residual in the case of a perturbed operator for general regularizing algorithms," *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, vol. 22, no. 4, pp. 989–993, 1982.

[32] A. Bakushinsky and A. Goncharsky, *Ill-posed problems: theory and applications*. Springer Netherlands, 1994.

[33] A. Leonov and A. Yagola, "Special regularizing methods for ill-posed problems with sourcewise represented solutions," *Inverse Problems*, vol. 14, no. 6, p. 1539, 1998.

[34] A. Goncharskii, A. S. Leonov, and A. G. Yagola, "A generalized discrepancy principle," *USSR Computational Mathematics and Mathematical Physics*, vol. 13, no. 2, pp. 25–37, 1973.

[35] C. W. Groetsch, *The theory of Tikhonov regularization for Fredholm equations of the first kind*. Pitman Boston, 1984, vol. 105.

[36] H. Engl, "Discrepancy principles for tikhonov regularization of ill-posed problems leading to optimal convergence rates," *Journal of optimization theory and applications*, vol. 52, no. 2, pp. 209–215, 1987.

[37] H. W. Engl and A. Neubauer, *Optimal Discrepancy Principles for the Tikh0n0v Regularization of Integral Equations of the First Kind*. Springer, 1985.

[38] H. Gfrerer, "An a posteriori parameter choice for ordinary and iterated tikhonov regularization of ill-posed problems leading to optimal convergence rates," *Mathematics of computation*, vol. 49, no. 180, pp. 507–522, 1987.

[39] A. Neubauer, "An a posteriori parameter choice for tikhonov regularization in hilbert scales leading to optimal convergence rates," *SIAM journal on numerical analysis*, vol. 25, no. 6, pp. 1313–1326, 1988.

[40] B. Kaltenbacher, A. Neubauer, and O. Scherzer, *Iterative regularization methods for nonlinear ill-posed problems*. Walter de Gruyter, 2008, vol. 6.

[41] U. Hämarik, R. Palm, and T. Raus, "A family of rules for parameter choice in tikhonov regularization of ill-posed problems with inexact noise level," *Journal of Computational and Applied Mathematics*, vol. 236, no. 8, pp. 2146–2157, 2012.

[42] Q. Jin, "Further convergence results on the general iteratively regularized gauss-newton methods under the discrepancy principle," *Mathematics of Computation*, vol. 82, no. 283, pp. 1647–1665, 2013.

[43] A. S. Leonov, "Choice of regularization parameter for non-linear ill-posed problems with approximately specified operator," *USSR Computational Mathematics and Mathematical Physics*, vol. 19, no. 6, pp. 1–15, 1979.

[44] A. Leonov, "Accuracy-order optimality of the generalized residual principle and some algorithms of solving nonlinear ill-posed problems with approximate data," *Siberian Mathematical Journal*, vol. 29, no. 6, pp. 940–947, 1988.

[45] Q. Jin and U. Tautenhahn, "On the discrepancy principle for some newton type methods for solving nonlinear inverse problems," *Numerische Mathematik*, vol. 111, no. 4, pp. 509–558, 2009.

[46] A. Smirnova, A. B. Bakushinskii, and L. deCamp, "On application of asymptotic generalized discrepancy principle to the analysis of epidemiology models," *Applicable Analysis*, 2014, under revision.

Appendix A Matlab Code - Main Program

```
function beta_local_tk_lv_leg_real
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  beta_local_tk_lv_real is calling program for 4 subroutines that
%  determine appropriate parameters for solving the inverse problem of
%  determining the beta transmission rate for avian bird flu.
%
%  The main program requests user input for computations for either World
%  or Indonesia.  Values for the following parameters are set based on this
%  choice:
%
%  Parameter
%  mu - Natural death rate of humans - set at 1/(70*12)
%  S0 - Initial population of Susceptible humans - set at 70000 and given
%             in units of 10^5
%  A - Birth rate of humans - set at 1000/12 - keeping susceptible
%             population stable for time frame used
%  delta - estimated noise - 0.005 for both World and Indonesia
%  Tau - multiplier for delta (used more for simulated)
%  a,b - Start and End months corresponding to data available
%  sw - this is switch parameter to notify subprograms as to whether real
%       or simulated data is being used
```

```
%  C_data - given for either World or Indonesia and is the cumulative
%             number of human infections
%  I_data - given for either World or Indonesia and is number of infected
%             poultry
%  For each method, a starting alpha (m) and a step size is given.  For
%  Tikhonov, Lavrentiev, and Local we have the appendages of TK, LV and LC
%  respectively.  For Legendre, we have the appendage LG.
%  For each data set, the grid is set for plotting
%
%
%  Calling subroutines:
%  This program calls the routine for determining parameter using the 4
%  different regularization methods.  In general, these routines operate as
%  follows:
%  Method is called and the data sets and parameter values are passed to
%  the subroutine.  The spline for determined beta transmission rate, the
%  determined alpha (m), and in the case of Tikhonov and Lavrentiev, the
%  maximum value of the matrix used is passed back so that upon plotting
%  and displaying these methods display comparable alpha values.
%  The call to the Tikhonov method also passes back the unregularized
%  solution which is common to all methods.
%
%  Plotting takes place after all methods are called and are displayed in
%  one plot.
%
%  This program can be modified to work with simulated data in the
%  following fashion:
%
```

```
%  1)  Comment out the input portion of the code
%
%  2)  Set parameter values for the following under the model problem for
%  which an exact solution can be analytically determined (see paper)
%       mu, S0, A, gamma
%  Set beginning and ending time
%       a and b
%  Set delta for amount of noise in data
%       delta
%  Set initial alpha (m) values and step sizes for methods
%     (The setup used in paper for simulated is coded and commented out)
%
%  3) Include these program lines before subprogram calls:
%  C_data = C_exact(t_data,A,mu,gamma).*(1 + delta*(1-2*rand(1,N)));
%  for i=1:N
%     I_data(i)=.01.*(sin(gamma.*i)+2);
%  end
%
%  4) Uncomment plot command for 'beta_exact'
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all

fprintf(1,'Choose W for World data, I for Indonesia.\n');
     fprintf(1,'               \n');
     p=input(' ','s');
     if p== 'W'
```

```
    mu = 1/(70*12);
    S0 = 70000;
    A = 1000/12;
    delta = 0.005;
    Tau = 1;
    a = 1; b = 42;
    sw = 1;
    C_data=C_dataW;
    I_data=I_dataW;
    alphaLV = 94.84128;          %World Lavrentiev initial
    alphaLVstep = .5*1.31724;    %World Lavrentiev step size alpha
    alphaTK = 19474.00;          %World Tikhonov initial
    alphaTKstep = .5*695.5;      %World Tikhonov step size alpha
    mLG = 6;                     %World Legendre initial
    mLGstep = 1;                 %World Legendre step size m
    LGsw = 1;                    %Switch to modify data vec Real
    alphaLC = 0.78;              %World Local initial
    alphaLCstep = 0.005;         %World Local step size alpha
    ax = [a b -3*10^-7 6*10^-7];
    t_data = a:1:b;                      %time data vector set up
    N = length(t_data);                  %length of time data vector


else
    mu = 1/(70*12);
    S0 =  242325638/10^5;
    A = 1000/12;
    delta = 0.005;
    Tau = 1;
```

```
        a = 1; b = 36;
        sw = 1;
        C_data=C_dataI;
        I_data=I_dataI;
        alphaLV = 2.1732;           %Indonesia  Lavrentiev initial
        alphaLVstep = 0.03622;      %Indonesia Lavrentiev step size alpha
        alphaTK = 31.181854;        %Indonesia Tikhonov initial
        alphaTKstep = 0.566943;     %Indonesia Tikhonov step size alpha
        mLG = 30;                   %Indonesia Legendre initial
        mLGstep = 1;                %Indonesia Legendre step size m
        LGsw = 1;                   %Switch to modify data vec Real
        alphaLC = 0.315;            %Indonesia Local initial
        alphaLCstep = 0.001;        %Indonesia Local step size alpha
        ax = [a b -2*10^-5 2*10^-5];
        t_data = a:1:b;                         %time data vector set up
        N = length(t_data);                     %length of time data vector
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%         %%%% For use with simulated data
%         %%%% Uncomment these parameter definitions, change as desired
%
%
%         mu = .01;
%         S0 = 10;
%         A = .05;
%         gamma = pi/6;
%         delta = 0.5;
%         Tau = 0.6;
```

```
%         a = 0; b = 4;
%         sw = 0;
%         alphaLV = 0.1;                %World Lavrentiev initial
%         alphaLVstep = 0.005;          %World Lavrentiev step size alpha
%         alphaTK = 0.05;               %World Tikhonov initial
%         alphaTKstep = 0.005;          %World Tikhonov step size alpha
%         mLG = 6;                      %World Legendre initial
%         mLGstep = 1;                  %World Legendre step size m
%         LGsw = 0;                     %Switch to NOT modify data vec
%         alphaLC = 3;                  %World Local initial
%         alphaLCstep = 0.1;            %World Local step size alpha
%         ax = [a b -10 30];
%         t_data = a:1:b;                       %time data vector set up
%         N = length(t_data);                   %length of time data vector
%         C_data = zeros(1,N);
%         I_data = zeros(1,N);
%         C_data = C_exact(t_data,A,mu,gamma).*(1 + delta*(1-2*rand(1,N)));
%          for i=1:N
%               I_data(i)=.01.*(sin(gamma.*i)+2);
%          end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%% Calling Sub-programs for Methods %%%%%


[pp_data_tk,pp_data_unreg,alphatk,maxMvtk] = ...
    beta_tikhonov_real_data_sub(C_data,I_data,...
    mu,S0,A,a,b,alphaTK,alphaTKstep,Tau,delta,sw);
```

```
[pp_data_lv,alphalv,maxMvlv] = ...
    beta_lavrentiev_real_data_sub(C_data,I_data,...
    mu,S0,A,a,b,alphaLV,alphaLVstep,Tau,delta,sw);
[pp_data_local,alpha] = ...
    beta_local_real_data_sub(C_data,I_data,...
    mu,S0,A,a,b,alphaLC,alphaLCstep,Tau,delta);
[pp_data_lg,m] = beta_legendre_real_data_modified_sub(C_data,I_data,...
    mu,S0,A,a,b,mLG,mLGstep,Tau,delta,LGsw);

%%%%%  Plot Sequence for Beta  %%%%%

figure
hold all

fplot(@(x)[ ppval(pp_data_unreg,x)],[a b],'*-k');
[~,~,~,current_entries] = legend;
    legend([current_entries ...
        {sprintf('\\beta_{Unreg_{data}}')}],'Location','NorthEast');

fplot(@(x)[ppval(pp_data_tk,x)],[a b],'*-b');
[~,~,~,current_entries] = legend;
    legend([current_entries ...
        {sprintf('\\beta_{Tikhonov_{data}}\\alpha=%3.3f',...
        alphatk/maxMvtk)}],'Location','NorthEast');

fplot(@(x)[ppval(pp_data_lv,x)],[a b],'*-g');
[~,~,~,current_entries] = legend;
    legend([current_entries ...
```

```
        {sprintf('\\beta_{Lavrentiev_{data}}\\alpha=%2.2f',...
        alphalv/maxMvlv)}],'Location','NorthEast');

fplot(@(x)[ ppval(pp_data_local,x)],[a b-alpha],'*-r');
[~,~,~,current_entries] = legend;
    legend([current_entries ...
        {sprintf('\\beta_{Local_{data}}\\alpha=%3.3f',...
        alpha)}],'Location','NorthEast');

fplot(@(x)[ppval(pp_data_lg,x)],[a+1 b],'*-m');
[~,~,~,current_entries] = legend;
    legend([current_entries ...
        {sprintf('\\beta_{Legendre_{data}} m=%2.0f',...
        m)}],'Location','NorthEast');

%fplot(@(x)[beta_exact(x,A,mu,gamma,S0)],[a b],'*-c');
%[~,~,~,current_entries] = legend;
%    legend([current_entries ...
%           {sprintf('\\beta_{Exact}')}],'Location','NorthEast');

fplot(@(x)[0],[a b],'-r');
axis(ax);
   hold off
   fprintf(' %12.8f   %12.8f\n', maxMvlv, maxMvtk); %Comment out if desired
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  The following functions can be used when modifying the calling program
%  to use simulated data




function y = C_exact(t,A,mu,gamma)
       y = A.*t+(mu.*sin(gamma.*t)+gamma.*cos(gamma.*t))./...
           ((mu^2+gamma^2).*exp(mu.*t));

function y = C_spline(t,t_data,C_data)
       pp = spline(t_data,C_data);
       y = ppval(pp,t);

function y = beta_exact(t,A,mu,gamma,S0)
            y = (A.*exp(mu.*t)-sin(gamma.*t))./(Ib(gamma,t).*...
                (S0-(cos(gamma.*t)-1)./gamma));

function y = Ib(t,t_data,I_data)
       pp = spline(t_data,I_data);
       y = ppval(pp,t);

%%%%%   Real Data %%%%%
  % ------------------------------------------------%
function IHd = I_dataW
```

```
%IHd = ones(1,length(y));
IHd(1) = 0.00440000000000000;
IHd(2) = 0.0106000000000000;
IHd(3) = 0.0100000000000000;
IHd(4) = 0.00370000000000000;
IHd(5) = 0.00470000000000000;
IHd(6) = 0.00830000000000000;
IHd(7) = 0.0129000000000000;
IHd(8) = 0.0250000000000000;
IHd(9) = 0.0242000000000000;
IHd(10) = 0.0136000000000000;
IHd(11) = 0.0151000000000000;
IHd(12) = 0.0206000000000000;
IHd(13) = 0.0148000000000000;
IHd(14) = 0.0109000000000000;
IHd(15) = 0.00970000000000000;
IHd(16) = 0.00200000000000000;
IHd(17) = 0.00560000000000000;
IHd(18) = 0.0130000000000000;
IHd(19) = 0.0254000000000000;
IHd(20) = 0.0356000000000000;
IHd(21) = 0.0196000000000000;
IHd(22) = 0.0128000000000000;
IHd(23) = 0.00810000000000000;
IHd(24) = 0.00540000000000000;
IHd(25) = 0.00610000000000000;
IHd(26) =  0.00590000000000000;
IHd(27) = 0.00700000000000000;
```

```
IHd(28) = 0.00850000000000000;

IHd(29) = 0.00730000000000000;

IHd(30) = 0.0129000000000000;

IHd(31) = 0.0349000000000000;

IHd(32) = 0.0326000000000000;

IHd(33) = 0.0151000000000000;

IHd(34) = 0.0118000000000000;

IHd(35) = 0.0118000000000000;

IHd(36) = 0.0118000000000000;

IHd(37) = 0.00436000000000000;

IHd(38) = 0.00436000000000000;

IHd(39) = 0.00436000000000000;

IHd(40) = 0.00356000000000000;

IHd(41) = 0.00356000000000000;

IHd(42) = 0.00356000000000000;



% ------------------------------------------------%

% ------------------------------------------------%

function Cexp = C_dataW

Cexp(1) = 0.00385000000000000;

Cexp(2) = 0.00385000000000000;

Cexp(3) = 0.00387000000000000;

Cexp(4) = 0.00387000000000000;

Cexp(5) = 0.00387000000000000;

Cexp(6) = 0.00391000000000000;

Cexp(7) = 0.00403000000000000;

Cexp(8) = 0.00408000000000000;
```

```
Cexp(9) = 0.00413000000000000;

Cexp(10) = 0.00421000000000000;

Cexp(11) = 0.00431000000000000;

Cexp(12) = 0.00433000000000000;

Cexp(13) = 0.00436000000000000;

Cexp(14) = 0.00440000000000000;

Cexp(15) = 0.00442000000000000;

Cexp(16) = 0.00442000000000000;

Cexp(17) = 0.00444000000000000;

Cexp(18) = 0.00467000000000000;

Cexp(19) = 0.00471000000000000;

Cexp(20) = 0.00478000000000000;

Cexp(21) = 0.00492000000000000;

Cexp(22) = 0.00495000000000000;

Cexp(23) = 0.00498000000000000;

Cexp(24) = 0.00499000000000000;

Cexp(25) = 0.00502000000000000;

Cexp(26) = 0.00505000000000000;

Cexp(27) = 0.00505000000000000;

Cexp(28) = 0.00507000000000000;

Cexp(29) = 0.00508000000000000;

Cexp(30) = 0.00512000000000000;

Cexp(31) = 0.00518000000000000;

Cexp(32) = 0.00525000000000000;

Cexp(33) = 0.00538000000000000;

Cexp(34) = 0.00549000000000000;

Cexp(35) = 0.00549000000000000;

Cexp(36) = 0.00562000000000000;
```

```
Cexp(37) = 0.00562000000000000;

Cexp(38) = 0.00564000000000000;

Cexp(39) = 0.00564000000000000;

Cexp(40) = 0.00564000000000000;

Cexp(41) = 0.00564000000000000;

Cexp(42) = 0.00564000000000000;



% ------------------------------------------------%

function IHd = I_dataI

%IHd = ones(1,length(y));

IHd(1) = 0.0023188;

IHd(2) = 0.0062062;

IHd(3) = 0.006138;

IHd(4) = 0.0021824;

IHd(5) = 0.0024552;

IHd(6) = 0.0036828;

IHd(7) = 0.0069564;

IHd(8) = 0.0114576;

IHd(9) = 0.0137764;

IHd(10) = 0.0073656;

IHd(11) = 0.0085932;

IHd(12) = 0.0127534;

IHd(13) = 0.0094798;

IHd(14) = 0.00682;

IHd(15) = 0.0059334;

IHd(16) = 0.0007502;

IHd(17) = 0.0028644;
```

```
IHd(18) = 0.007161;

IHd(19) = 0.0115258;

IHd(20) = 0.0144584;

IHd(21) = 0.0066836;

IHd(22) = 0.0060698;

IHd(23) = 0.0033418;

IHd(24) = 0.0021142;

IHd(25) = 0.0026598;

IHd(26) = 0.0030008;

IHd(27) = 0.0035464;

IHd(28) = 0.0040238;

IHd(29) = 0.00341;

IHd(30) = 0.0059334;

IHd(31) = 0.0117304;

IHd(32) = 0.0143902;

IHd(33) = 0.0140833; % missing data, average taken

IHd(34) = 0.0137764;

IHd(35) = 0.0137764;

IHd(36) = 0.0137764;



% ------------------------------------------------%

% ------------------------------------------------%

function Cexp = C_dataI


Cexp(1) = 0.00135;

Cexp(2) = 0.00135;

Cexp(3) = 0.00137;
```

```
Cexp(4) = 0.00137;
Cexp(5) = 0.00137;
Cexp(6) = 0.00139;
Cexp(7) = 0.00141;
Cexp(8) = 0.00141;
Cexp(9) = 0.00141;
Cexp(10) = 0.00141;
Cexp(11) = 0.00141;
Cexp(12) = 0.00141;
Cexp(13) = 0.00141;
Cexp(14) = 0.00141;
Cexp(15) = 0.00141;
Cexp(16) = 0.00141;
Cexp(17) = 0.00141;
Cexp(18) = 0.00161;
Cexp(19) = 0.00161;
Cexp(20) = 0.00163;
Cexp(21) = 0.00163;
Cexp(22) = 0.00163;
Cexp(23) = 0.00165;
Cexp(24) = 0.00165;
Cexp(25) = 0.00167;
Cexp(26) = 0.00168;
Cexp(27) = 0.00168;
Cexp(28) = 0.00170;
Cexp(29) = 0.00170;
Cexp(30) = 0.00171;
Cexp(31) = 0.00171;
```

```
Cexp(32) = 0.00171;

Cexp(33) = 0.00175;

Cexp(34) = 0.00176;

Cexp(35) = 0.00176;

Cexp(36) = 0.00178;

%Cexp(37) = 0.00178;

%Cexp(38) = 0.00178;

%Cexp(39) = 0.00178;

%Cexp(40) = 0.00178;

%Cexp(41) = 0.00178;

%Cexp(41)= 0.00178;
```

Appendix B Matlab Code - Tikhonov's algroithm subprogram

```
function [pp_data,pp_data_unreg,alpha,maxMv] = ...
    beta_tikhonov_real_data_sub(C_data,I_data,...
    mu,S0,A,a,b,alpha0,alphastep,Tau,delta,sw)

%.....................................................
% Inverse linear problem for transmission coefficient beta(t)
%.....................................................
%
%  Called from beta_local_tk_lv_leg_real.m
%
% Inputs:
% C(t) = supplied human incidence
% IHd(t) = number of birds infected
% S0 = initial number of susceptible humans
% mu = natural death rate of humans
% A = birth rate of humans
% a b = start and end time
% alpha0 = initial alpha
% alphastep = step size to change alpha
% Tau = Multiplier for given delta
% delta = noise
% sw = switch: 1 for real data, 0 for simulated
```

```
%
%.....................................................
% Real data for the inverse problem
%.....................................................

t_data = a:1:b;
h = 0.01;
z = a+h:h:b;
K = length(z);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  This method builds a lower triangular matrix - the A matrix described in
%  the paper.  The (1,1) cell of the matrix is set calculating the K of the
%  volterra integral described.  The iteration then proceeds building
%  subsequent rows and RHS from 2 to K (A is a K x K matrix). The method
%  requires the build of A, the multiplication of A' with A and the RHS and
%  the addition of the penalty term, alpha*Identity to A'*A.  We use \ to
%  solve for beta. The resulting vector for beta is splined using the K
%  values of the z vector and passed back to calling program.  In addition,
%  maximum values of the A'*A matrix are also passed back so that
%  comparable alpha's may be generated from the different methods.
%
%
        C_d = C_spline(z,t_data,C_data);     %spline all points - data
        I_b = Ib(z,t_data,I_data);           %spline all points - data bird
        Vector_for_matrix_data = zeros(1,K);  %zero vector for matrix build
        Q_d(1,1) = quad(@(tau)(C_spline(tau,...
```

```
              t_data,C_data).*exp(mu.*tau)),a,z(1));  %set (1,1) value

discrepancy = 100;  M = 0;
alpha=alpha0;

disp('___________________________________')
disp('M       alpha        discrepancy ')
disp('___________________________________')

    while discrepancy - Tau*delta > 0 && M < 25

           f_data = zeros(K,1);      %zero RHS vector
           Matrix_data = zeros(K);   %zero A matrix

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%   Build A matrix, by rows   %%%%%

          for j = 1:K

                if j >= 2;
                    Q_d(1,j) = Q_d(1,j-1) + quad(@(tau)(C_spline(tau,...
                        t_data,C_data).*exp(mu.*tau)),z(j-1),z(j));
                end
                %%%%% Vector_for_matrix_data holds row data and additional
                %%%%% calculations replace zeros and are then used to
                %%%%% replace rows of A matrix
                Vector_for_matrix_data(1,j) = I_b(j).*((S0-A/mu + ...
```

```
            C_spline(a,t_data,C_data) + ...
            mu.*Q_d(1,j)).*exp(-mu.*z(j))+A/mu-C_d(j)).*h;
        f_data(j,1) = C_d(j)-C_spline(a,t_data,C_data);
        Matrix_data(j,1:j) = Vector_for_matrix_data(1,1:j);

end
 V=Matrix_data'*Matrix_data;
 if sw == 1
     maxMv=max(max(V,[],2));
 else
     maxMv=1;
 end

 f_data_tr = Matrix_data'*f_data;
 Matrix_data_tr = Matrix_data'*Matrix_data + ...
     alpha*eye(K);  %the method

 w = Matrix_data_tr\f_data_tr;          %solve for beta
 pp_data = spline(z,w);                 %set spline for beta

 %set A to beta matrix for use in discrepancy
 A_to_beta = (Matrix_data*w)';
 %calculate for use in discrepancy
 A_data = spline(z,A_to_beta);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
discrepancy = sqrt(quad(@(x)(C_spline(x,t_data,C_data) - ...
    C_spline(a,t_data,C_data)-ppval(A_data,x))...
    .^2,a,b)./quad(@(x)(C_spline(x,t_data,C_data) - ...
    C_spline(a,t_data,C_data)).^2,a,b));

M = M + 1;

  fprintf('%6.2f   %12.8f   %12.8f\n', M, alpha, discrepancy);

alpha = alpha - alphastep;

    end
alpha = alpha + alphastep;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%  Calculation unregularized solution

            f_data = zeros(K,1);

            for j = 1:K

                 if j >= 2;

                 Q_d(1,j) = Q_d(1,j-1) + quad(@(tau)(C_spline(tau,...
                     t_data,C_data).*exp(mu.*tau)),z(j-1),z(j));
                 end
```

```
                Vector_for_matrix_data(1,j) = I_b(j).*((S0-A/mu + ...
                    C_spline(a,t_data,C_data) + ...
                    mu.*Q_d(1,j)).*exp(-mu.*z(j))+A/mu-C_d(j)).*h;
                f_data(j,1) = C_d(j)-C_spline(a,t_data,C_data);
                Matrix_data(j,1:j) = Vector_for_matrix_data(1,1:j);

            end

            w = Matrix_data\f_data;
            pp_data_unreg = spline(z,w);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Matrix_data'*Matrix_data(1,1);



% ------------------------------------------------%
% FUNCTION DEFINITIONS
% ------------------------------------------------%

   function y = C_spline(t,t_data,C_data)
       pp = spline(t_data,C_data);
       y = ppval(pp,t);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   function y = Ib(t,t_data,I_data)
       pp = spline(t_data,I_data);
```

```
y = ppval(pp,t);
```

Appendix C Matlab Code - Lavrentiev's algroithm subprogram

```
function [pp_data,alpha,maxMv]=beta_lavrentiev_real_data_sub(C_data,...
    I_data,mu,S0,A,a,b,alpha0,alphastep,Tau,delta,sw)

%.....................................................
% Inverse linear problem for transmission coefficient beta(t)
%.....................................................
%
%  Called from beta_local_tk_lv_leg_real.m
%
% Inputs:
% C(t) = supplied human incidence
% IHd(t) = number of birds infected
% S0 = initial number of susceptible humans
% mu = natural death rate of humans
% A = birth rate of humans
% a b = start and end time
% alpha0 = initial alpha
% alphastep = step size to change alpha
% Tau = Multiplier for given delta
% delta = noise
% sw = switch: 1 for real data, 0 for simulated
%
```

```
%.....................................................
% Real data for the inverse problem
%.....................................................

t_data = a:1:b;
h = 0.01;
z = a+h:h:b;
K = length(z);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  This method builds a lower triangular matrix - the A matrix described in
%  the paper.  The (1,1) cell of the matrix is set calculating the K of the
%  volterra integral described.  The iteration then proceeds building
%  subsequent rows and RHS from 2 to K (A is a K x K matrix).  To A, we add
%  the penalty term, alpha*Identity matrix and use \ to solve for beta. The
%  resulting vector for beta is splined using the K values of the z vector
%  and passed back to calling program.  In addition, maximum values of the
%  A matrix are also passed back so that comparable alpha's may be
%  generated from the different methods.
%
%

      C_d = C_spline(z,t_data,C_data);   %spline all points - data
      I_b = Ib(z,t_data,I_data);         %spline all points - data bird
      Vector_for_matrix_data = zeros(1,K);   %zero vector for matrix build
      Q_d(1,1) = quad(@(tau)(C_spline(tau,t_data,C_data).*exp(mu.*tau))...
          ,a,z(1));   %set (1,1) value
```

```
discrepancy = 100;  M = 0;
alpha = alpha0;

disp('_____________________________________')
disp('M       alpha        discrepancy ')
disp('_____________________________________')


    while discrepancy - Tau*delta > 0 && M < 25

           f_data = zeros(K,1);     %zero RHS vector
           Matrix_data = zeros(K);  %zero A matrix

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%   Build A matrix, by rows   %%%%%

          for j = 1:K

                if j >= 2;
                   Q_d(1,j) = Q_d(1,j-1) + quad(@(tau)(C_spline(tau,...
                       t_data,C_data).*exp(mu.*tau)),z(j-1),z(j));
                         %cummulate
                end
                %%%%% Vector_for_matrix_data holds row data and additional
                %%%%% calculations replace zeros and are then used to
                %%%%% replace rows of A matrix
```

```
        Vector_for_matrix_data(1,j) = I_b(j).*((S0-A/mu + ...
            C_spline(a,t_data,C_data) + ...
            mu.*Q_d(1,j)).*exp(-mu.*z(j))+A/mu-C_d(j)).*h;
        f_data(j,1) = C_d(j)-C_spline(a,t_data,C_data);
               %Calculate RHS
        Matrix_data(j,1:j) = Vector_for_matrix_data(1,1:j);
               %incorporate for new row calculated

end
 V=Matrix_data;
 if sw == 1
     maxMv=max(max(V,[],2));
 else
     maxMv=1;
 end
 f_data_tr = f_data;
 Matrix_data_tr = Matrix_data + alpha*eye(K);   %the method

 w = Matrix_data_tr\f_data_tr;     %solve for beta
 pp_data = spline(z,w);            %set spline for beta

 %set A to beta matrix for use in discrepancy
 A_to_beta = (Matrix_data*w)';
 %calculate for use in discrepancy
 A_data = spline(z,A_to_beta);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

discrepancy = sqrt(quad(@(x)(C_spline(x,t_data,C_data) - ...
    C_spline(a,t_data,C_data)-ppval(A_data,x))...
    .^2,a,b)./quad(@(x)(C_spline(x,t_data,C_data) - ...
    C_spline(a,t_data,C_data)).^2,a,b));

M = M + 1;

  fprintf('%6.2f   %12.8f   %12.8f\n', M, alpha, discrepancy);
  alpha=alpha-alphastep;

    end
    alpha = alpha + alphastep;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%  Calculation unregularized solution

            f_data = zeros(K,1);
            for j = 1:K

                 if j >= 2;

                 Q_d(1,j) = Q_d(1,j-1) + quad(@(tau)(C_spline...
                     (tau,t_data,C_data).*exp(mu.*tau)),z(j-1),z(j));
                 end

                 Vector_for_matrix_data(1,j) = I_b(j).*((S0-A/mu + ...
```

```
                    C_spline(a,t_data,C_data) + ...
                    mu.*Q_d(1,j)).*exp(-mu.*z(j))+A/mu-C_d(j)).*h;
                f_data(j,1) = C_d(j)-C_spline(a,t_data,C_data);
                Matrix_data(j,1:j) = Vector_for_matrix_data(1,1:j);

            end

            w = Matrix_data\f_data;
            pp_data_unreg = spline(z,w);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



% ------------------------------------------%
% FUNCTION DEFINITIONS
% ------------------------------------------%

   function y = C_spline(t,t_data,C_data)
       pp = spline(t_data,C_data);
       y = ppval(pp,t);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   function y = Ib(t,t_data,I_data)
       pp = spline(t_data,I_data);
       y = ppval(pp,t);
```

Appendix D Matlab Code - Local regularization algroithm subprogram

```
function [pp_data2,alpha] = beta_local_real_data_sub(C_data,I_data,...
    mu,S0,A,a,b,alpha0,alphastep,Tau,delta);
h=.01;
alpha=0;
t_data = a:1:b;
N = length(t_data);
alpha_0 = 0; %or some other fixed value
%set ups and basic information.
%h will determine size of matrix and discretization of problem
%40 is partition size for first set of integrals
%much of allocation and setup can be done initially:
%lambda, g, vector for matrix exact/data are sized
%Matrices for A_exact and A_data are sized
%A weight and evaluation vector for quadrature is obtained
%C_exact and C_spline and Ib(t) are determined
%Integrals for C_exact and C_data used in kernal are obtained
%These are used to create the A matrix for both exact and data
%Matrix_data_base is used in discrepancy loop, only calculated once
%g_alpha exact is determined and w vector for this exact is calculated
%(splined for appx to beta)
%testing code is commented out, expose to view rel error in partition sizes
```

```
step=h/40;
z = a+h:h:b;
K = length(z);

C_d = C_spline(z,t_data,C_data);
I_b = Ib(z,t_data,I_data);
Vector_for_matrix_data = zeros(1,K);
Matrix_data_base=zeros(K);
[s, w1, m] = mpquad(a,b,step);
Q_d = cumsum(C_spline(s,t_data,C_data).*exp(mu.*s).*w1,2);
Q_d = Q_d(40:40:end);
Vector_for_matrix_data = I_b.*((S0-A/mu+C_spline(a,t_data,C_data) + ...
    mu.*Q_d).*exp(-mu.*z)+A/mu-C_d).*h;
Matrix_data_base = tril(ones(length(Vector_for_matrix_data),1)*...
    Vector_for_matrix_data);

    discrepancy = 100;
    M = 0;
    alpha = alpha0;
disp('___________________________________')
disp('M       alpha        discrepancy  ')
disp('___________________________________')

    while discrepancy    - Tau*delta > 0 && M < 25,

    lambda_data = zeros(K,1);
    g_alpha_data = zeros(K,1);
    step2=(alpha)/(40);
```

```
z2 = repmat(z,[40 1]);
%set up matrix for longer vector for s quad  dim:  40 x K
s1 = [step2/2:step2:alpha-step2/2];
%vector for mp integration of s from 0 to alpha
s2 = transpose(repmat(s1, [K 1]));
%replicate the vector for mp integration of s on K columns
s2a = s2(:)';
%convert s matrix to vector 40*K
z2=z2+s2;
%add matrix of t values plus s values
z2 = z2(:)';
%convert matrix to vector
[s, w1, m] = mpquad(0,alpha,step2);
%determine weight vector for quadrature
w2 = transpose(repmat(w1, [K 1]));
%replicate weight vector 40 x K
w2 = w2(:)';
%convert weight matrix to vector
Vector_for_matrix_data_s = zeros(1,length(z2));   %set up vector for s data

C_ds = C_spline(z2,t_data,C_data);
%calculate C data on t plus s vector of length 40*K
I_bs = Ib(z2,t_data,I_data);;
%calculate Ib vector on t plus s vector of length 40*K
Q_ds = C_spline(z2,t_data,C_data).*exp(mu.*z2).*w2;
%calculate inner integral of S(C(t),t) on t plus s vector
```

```
%calculate integral S(C(t+s),t+s)Ib(t+s)(alpha-s)
Vector_for_matrix_data_s = (I_bs.*((S0-A/mu+C_spline(a,t_data,C_data) + ...
    mu.*Q_ds).*exp(-mu.*z2)+A/mu-C_ds).*(alpha-s2a).*step2);
%calculate lambda data - penalty
lambda_data = accumarray( reshape(repmat(1:K,40,1),{},1),...
    Vector_for_matrix_data_s)./alpha;

g_alpha_data = zeros(K,1);
Matrix_data=Matrix_data_base;

% calculate RHS
if alpha == 0;
    g_alpha_data = (C_d-C_spline(a,t_data,C_data)*ones(1,length(z)))';
    lambda_data = zeros(K,1);
else
    for j = 1:K
        g_alpha_data(j,1) = quad(@(rho)(C_spline(z(j) + ...
            rho,t_data,C_data)-C_spline(a,t_data,C_data)),0,alpha)./alpha;
    end
end
    %solve
    Matrix_data = Matrix_data_base + diag(lambda_data);
    w = Matrix_data\g_alpha_data;
    pp_data2 = spline(z,w);
    %set A to beta matrix for use in discrepancy
    A_to_beta2 = (g_alpha_data - lambda_data.*w)';
    %calculate for use in discrepancy
    A_data = spline(z,A_to_beta2);
```

```
    %calculate discrepancy
    discrepancy = sqrt(quad(@(x)(C_spline(x,t_data,C_data) - ...
        C_spline(a,t_data,C_data)-ppval(A_data,x))...
         .^2,a,b-alpha)./quad(@(x)(C_spline(x,t_data,C_data) - ...
         C_spline(a,t_data,C_data)).^2,a,b-alpha));

M = M + 1;

  fprintf('%6.2f   %12.8f   %12.8f     \n', M, alpha, discrepancy);
alpha = alpha - alphastep;
    end
    alpha = alpha + alphastep;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    function [s, w, m] = mpquad(a,b,step)
        s = a+step/2:step:b-step/2;
        m = length(s);
        h = (b-a)/m;
        w = h*ones(1,m);

    function y = C_spline(t,t_data,C_data)
       pp = spline(t_data,C_data);
       y = ppval(pp,t);

    function y = Ib(t,t_data,I_data)
       pp = spline(t_data,I_data);
```

```
y = ppval(pp,t);
```

Appendix E Matlab Code - Legendre approximation subprogram

```
function [ppdata,m]=beta_legendre_real_data_modified_sub(C_dataD,I_data,...
    mu,S0,A,a,b,minit,mstep,Tau,delta,LGsw);

%.....................................................
% Inverse linear problem for transmission coefficient beta(t)
%.....................................................
%
% Inputs:
% C(t) = supplied human incidence
% IHd(t) = number of birds infected
% S0 = initial number of susceptible humans
% mu = natural death rate of humans
% A
% a b
% minit = initial number of polynomials
% mstep = step size to change m
%
%.....................................................
% Real data for the inverse problem
%.....................................................

t_data = a:1:b;
```

```
h = 0.01;
z = a+h:h:b;
K = length(z);
C_d = C_spline(z,t_data,C_dataD);
I_b = Ib(z,t_data,I_data);
m=minit;
discrepancy = 100;  M = 0;

disp('_________________________________')
disp('  M       m       discrepancy   ')
disp('_________________________________')

    while discrepancy - Tau*delta > 0 && M < 15,

m = m + mstep;
c = zeros(m+1,1);
if LGsw == 1
    C_data = 1E3*C_dataD;
else
    C_data = C_dataD;
end
for j = 1:m+1

%c(j,1) = quad(@(t)fun_(t,j-1,t_data,C_data,a,b),-1,1, 1E-13);
c(j,1) = quad(@(t)fun_(t,j-1,t_data,C_data,a,b),-1,1, 1E-9); %use this m>18

end
```

```
discrepancy = sqrt(quad(@(x)(C_spline(x,t_data,C_data) - ...
    approx_(x,m,c,a,b)).^2,a,b)./...
    quad(@(x)(C_spline(x,t_data,C_data)).^2,a,b));

M = M + 1;

  fprintf('%6.2f  %6.2f  %12.8f\n',M,m,discrepancy);

    end

ppdata=spline(z,beta_approx_spline(z,m,c,a,b,A,mu,S0,t_data, I_data,LGsw));



% ------------------------------------------------%
% FUNCTION DEFINITIONS
% ------------------------------------------------%

   function y = C_spline(t,t_data,C_data)
       pp = spline(t_data,C_data);
       y = ppval(pp,t);


    function P = leg(x,j)

                if j == 0
                    P1 = 1; P = P1;
                elseif j == 1
                     P2 = x; P = P2;
```

```
        else
                P1 = 1; P2 = x;
                for k = 2:j
          P3 = ((2*(k-1)+1).*x.*P2 - (k-1).*P1)./k;
          P1 = P2; P2 = P3;
                end
                P = P3;
        end
        P = P.*sqrt((2*j+1)/2);



function y = fun_(t,j,t_data,C_data,a,b)
x = ((b - a).*t + a + b)./2;
y =  C_spline(x,t_data,C_data).*leg(t,j);

function y = approx_(x,m,c,a,b)
t = (2.*x - a - b)./(b - a);
y = 0;
    for j = 1:m+1
    y = y + c(j).*leg(t,j-1);
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 function P_prime = leg_prime(x,j)

        if j == 0
            P1_prime = 0; P_prime = P1_prime;
        elseif j == 1
```

```
                P2_prime = 1; P_prime = P2_prime;
            else
                    P1_prime = 0; P2_prime = 1;
                    P1 = 1; P2 = x;
                    for k = 2:j
              P3_prime = ((2*(k-1)+1).*(x.*P2_prime + P2) - ...
                  (k-1).*P1_prime)./k;
              P1_prime = P2_prime; P2_prime = P3_prime;
              P3 = ((2*(k-1)+1).*x.*P2 - (k-1).*P1)./k;
                        P1 = P2; P2 = P3;
                    end
                    P_prime = P3_prime;
            end
            P_prime = P_prime.*sqrt((2*j+1)/2);



function y = approx_derivative_spline(x,m,c,a,b)
        t = (2.*x - a - b)./(b - a);
        y = 0;
         for j = 1:m+1
         y = y + c(j).*leg_prime(t,j-1);
         end
         y = 2.*y./(b - a);

function y = Ib(t,t_data,I_data)
     pp = spline(t_data,I_data);
     y = ppval(pp,t);
```

```
function y = beta_approx_spline(t,m,c,a,b,A,mu,S0,t_data, I_data,LGsw)
        n = length(t);
        y = zeros(1,n);
        if LGsw == 1
            I_data = 1E3*I_data;
        for i = 1:n
            %y(1,i) = approx_derivative_spline(t(i),m,c,a,b)./...
            %(Ib(gamma,t(i)).*(S0.*exp(-mu.*t(i))+quad(@(tau)...
            %((A-approx_derivative_spline(tau,m,c,a,b)).*...
            %exp(-mu.*(t(i)-tau))),a,t(i))));
            y(1,i) = approx_derivative_spline(t(i),m,c,a,b)./...
                (Ib(t(i),t_data,I_data).*((S0-A/mu+1E-3*...
                approx_(a,m,c,a,b)).*exp(-mu.*t(i))+A/mu-1E-3*...
                approx_(t(i),m,c,a,b)+mu.*quad(@(tau)(1E-3*...
                approx_(tau,m,c,a,b).*exp(-mu.*(t(i)-tau))),a,t(i))));
        end
        else
            I_data = I_data;
            for i = 1:n
            y(1,i) = approx_derivative_spline(t(i),m,c,a,b)./...
            (Ib(t(i),t_data,I_data).*((S0-A/mu+approx_(a,m,c,a,b))...
            .*exp(-mu.*t(i))+A/mu-approx_(t(i),m,c,a,b)+mu.*...
            quad(@(tau)(approx_(tau,m,c,a,b).*exp(-mu.*(t(i)-...
            tau))),a,t(i))));
            end
        end
```