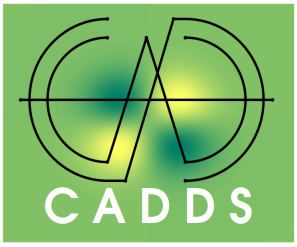
Technical Report 0.2:

WEIGHTED UNIVERSAL ALGEBRAIC CONTROLLERS FOR EPIDEMIC DYNAMICS IDENTIFICATION AND CONTROL



Fredy Vides
Scientific Computing Innovation Center, UNAH &
Centre for Analysis of Data-Driven Systems, Honduras
E-mail: fredy.vides@unah.edu.hn

Norman Sabillón
Scientific Computing Innovation Center, UNAH &
Centre for Analysis of Data-Driven Systems, Honduras
sabillon_rey2004@hotmail.com

CONTENTS

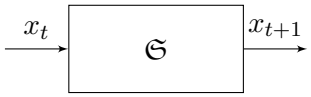
ABSTRACT. In this document, the research work in progress, corresponding to an application of universal algebraic controllers (in the sense of [5]) to the computation of predictive models for epidemic propagation in Honduras, is presented. In order to reduce the forcasting fluctuations due to non-uniform data mesurement techniques, UAC weighted shooting methods are implemented. Some data-driven numerical predictive simulations for the propagation of COVID-19 in Honduras during the first semester of 2020, are outlined.

1. INTRODUCTION

The purpose of this document is to present some theoretical and computational techniques for constrained approximation of data-driven predictive models for the propagation of COVID-19 in Honduras during the first semester of 2020. These models can be interpreted as discrete-time systems that can be *partially* described using the transition block diagram

(1.1), where the *black-box device* $\mathfrak{S}$ needs to be determined in such a way that it can be used to transform the *present state* x_t into the *next state* x_{t+1}, according to (1.2).

(1.1)



In this study each entry $x_{t,j}$ of the state vector x_t corresponds to the known/predicted number of confirmed cases of COVID-19 in Department j, for $1 \leq j \leq 18$, for instance $x_{t,1}$ is the estimated number of confirmed cases in Atlántida at stage t. The number $x_{t,19}$ represents the estimated number of confirmed cases nationwide at stage t. We will first approach the computation of the *state-transtion* maps corresponding to the device (1.1), applying the algebraic methods developed in [5] and [3] to compute the state-transition matrices that correspond to *matrix solvents* of difference equations of the form

$$\Sigma : \begin{cases} x_{t+1} = T_t x_t, \ \ t \geq 1 \\ x_1 \in \Sigma \subseteq \mathbb{R}^{19n} \end{cases} \tag{1.2}$$

where $\Sigma \subseteq \mathbb{R}^{19n}$ is the set of *valid* propagation states for the system with $n \in \mathbb{Z}$ fixed, and where the matrices $T_t \in \mathbb{R}^{18n \times 18n}$ are to be determined by the relations (1.2) and in some cases, in addition, need to satisfy the following structural constraints.

$$\begin{cases} T_t = \prod_{j=1}^{19n} \left(I + \hat{e}_{j,19n} (\tau_{(t,j)} - \hat{e}_{j,19n})^\top \right) \\ K_j \circ \tau_{(t,j)}^\top = \tau_{t,j}, \ \ 1 \leq j \leq 19n \end{cases} \tag{1.3}$$

where $\circ$ denotes the Hadamard product, K_j is the jth-row of a connectivity matrix determined by the geographic configuration of Honduras territory under consideration, the matrices $\tau_{(t,j)} \in \mathbb{R}^{19n \times 1}$ are to be determined by (1.2) and 1.3, and where $\hat{e}_{j,n}$ denotes the matrices in $\mathbb{C}^{n \times 1}$ representing the canonical basis of $\mathbb{C}^n$ (the j-column of the $n \times n$ identity matrix I), that are determined by the expression

$$\hat{e}_{j,n} = \begin{bmatrix} \delta_{1,j} & \delta_{2,j} & \cdots & \delta_{n-1,j} & \delta_{n,j} \end{bmatrix}^\top \tag{1.4}$$

for each $1 \leq j \leq n$, where $\delta_{k,j}$ is the Kronecker delta determined by the expression.

$$\delta_{k,j} = \begin{cases} 1, \ \ k = j \\ 0, \ \ k \neq j \end{cases} \tag{1.5}$$

Once we have computed a pair of systems Σ_D, Σ_P of the form (1.2), a predictor-descriptor system Σ_{PD} based on Σ_D, Σ_P will be computed, applying the techniques presented in §2. Some algorithms for the computation of Σ_{PD} will be presented in §3. Some numerical experiments are presented in §4.

2. Universal Algebraic Controllers for the Propagation Model

2.1. **Connectivity Matrices.** Based on the COVID-19 propagation behavior data available thus far. Let us consider the connectivity matrix $K \in \mathbb{R}^{18 \times 18}$ determined by the expression.

$$K = I + adj(G) \tag{2.1}$$

Where $adj(G) = [a_{jk}]$ denotes the adjacency matrix of a graph $G = (V_G, E_G)$ determined by the rules.

$$a_{jk} = \begin{cases} 1, \ \text{if} \ [v_j, v_k] \in E_G, \ \ v_j, v_k \in V_G \\ 0, \ \text{otherwise} \end{cases} \tag{2.2}$$

The graph G is determined by the geographical configuration of the Honduras territory, and belongs to the class represented by graphs like the ones in figura 2.1.

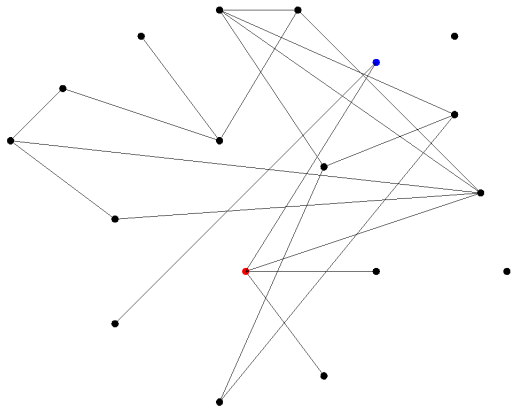


FIGURE 2.1. Geographical connectivity graph corresponding to Honduras departments roads configuration. The red dot represents Francisco Morazán, the blue dot represents Cortés.

2.2. **UAC Computation.**

2.2.1. *A geographically constrained sequential UAC Descriptor.* We start considering a geographically constrained switched UAC model of the form.

$$\begin{cases} x_{t+1} = A_t x_t \\ x_0 \in \mathbb{R}^{19n\times 1} \end{cases}, t \geq 0 \tag{2.3}$$

Where the matrices $A_t \in \mathbb{R}^{19n\times 19n}$ satisfy the constraints (1.2) and (1.3), and are computed according to the observed propagation's behavior by applying lemma 2.1.

Lemma 2.1. *Let us consider two propagation states $x_t, x_{t+1} \in \Sigma$ and the connectivity matrix $K \in \mathbb{R}^{19n\times 19n}$ determined by* (2.1)*. There is a matrix $T_t \in \mathbb{R}^{19n\times 19n}$ that satisfies* (1.2) *and* (1.3)*, if and only if for each $1 \leq j \leq 19n$, there is $\tau_{(t,j)} \in \mathbb{R}^{19n\times 1}$ such that $\tau_{(t,j)}^\top x_t = x_{t+1,j}$ and $K_j \circ \tau_{(t,j)} = \tau_{(t,j)}$, with $x_{t+1} = [x_{t+1,j}]$.*

Proof. Let us consider the matrix.

$$E_{\tau_{(t,j)}} = I + \hat{e}_{j,19n}(\tau_{(t,j)} - \hat{e}_{j,19n})^\top \tag{2.4}$$

Given $x = [x_j] \in \mathbb{R}^{19n\times 1}$, we will have that.

$$\begin{aligned} E_{\tau_{(t,j)}} x &= (I + \hat{e}_{j,19n}(\tau_{(t,j)} - \hat{e}_{j,19n})^\top x \\ &= \begin{cases} \tau_{(t,j)}^\top x, & k = j \\ x_k, & k \neq j \end{cases} \end{aligned} \tag{2.5}$$

Let us set $T_t = \prod_{j=1}^{19n} E_{\tau_{(t,j)}}$ by (1.3). By (2.4) and (2.5), we will have that the matrix $T_t \in \mathbb{R}^{19n\times 19n}$ that satisfies (1.2) and (1.3), if and only if for each $1 \leq j \leq 19n$, there is $\tau_{(t,j)} \in \mathbb{R}^{19n\times 1}$ such that $\tau_{(t,j)}^\top x_t = x_{t+1,j}$ and $K_j \circ \tau_{(t,j)} = \tau_{(t,j)}$. This completes the proof. □

By applying UAC techniques developed in [5] one can compute the best linear time invariant approximation of system (2.3), determined by the expression.

$$\begin{cases} x_{t+1} = Ax_t \\ x_0 \in \mathbb{R}^{19n\times 1} \end{cases}, t \geq 0 \tag{2.6}$$

The dynamics matrix A in (2.6) can be computed using algorithm 1.

2.2.2. *A geographically free UAC Predictor*. A geographically free UAC model of the form.

$$\begin{cases} x_{t+1} = T_t x_t \\ x_0 \in \mathbb{R}^{19n\times 1} \end{cases}, t \geq 0 \tag{2.7}$$

Where the matrices $T_t \in \mathbb{R}^{19n\times 19n}$ are computed according to the observed propagation's behavior using the techniques developed in [5, §3.2]. In particular, by applying UAC techniques developed in [5] one can compute the best linear time invariant approximation of system (2.7), determined by the expression.

$$\begin{cases} x_{t+1} = Sx_t \\ x_0 \in \mathbb{R}^{19n\times 1} \end{cases}, t \geq 0 \tag{2.8}$$

The dynamics matrix A in (2.8) can be computed using algorithm 2.

2.2.3. *Weighted Universal Algebraic Controllers.* In this section we will build on the ideas for real-time forecasting of epidemic trajectories that were presented in [1]. In order to derive a descriptor system of the form,

$$y_{t+1} = W \begin{bmatrix} S \\ A \end{bmatrix} y_t \tag{2.9}$$

where A and S are dynamic matrix solvents of the system identification problems determined by (2.6) and (2.8), respectively. The matrix $W \in \mathbb{R}^{19n\times 38n}$ is a structured nonnegative matrix of the form.

$$W = \begin{bmatrix} u_1 & 0 & \cdots & 0 & v_1 & 0 & \cdots & 0 \\ 0 & u_2 & \ddots & \vdots & 0 & v_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & u_{19n} & 0 & \cdots & 0 & v_{19n} \end{bmatrix} \tag{2.10}$$

Where the coefficients $u_j, v_j \geq 0$ are to be determined and need to satisfy the constraints,

$$(u_j, v_j) = \arg\min \sum_{k=1}^{N-1} (\hat{e}_{j,19n}^{\top}((uSy_{t_k} + vAz_{t_k}) - x_{t_{k+1}}))^2 \tag{2.11}$$

for some times $t_1, \ldots, t_N$ at which the variable x_{t_k} has been observed or measured, and where y_{t_k}, z_{t_k} have been generated using systems (2.7) and (2.3), respectively.

2.3. **Weighted UAC shooting methods.** One can apply shooting methods to improve predictions computed with weighted UAC predictor-descriptor schemes. In this study we will only consider three terms weighted UAC methods. Although the technique implemented can be extended to higher number of terms. Let us consider three weighted UAC predictor-descriptor systems of the form:

$$y_{t+1}^{(0)} = W_0 \begin{bmatrix} S \\ A \end{bmatrix} y_t^{(0)}$$
$$y_{t+1}^{(1)} = W_1 \begin{bmatrix} S \\ A \end{bmatrix} y_t^{(1)}$$
$$y_{t+1}^{(2)} = W_2 \begin{bmatrix} S \\ A \end{bmatrix} y_t^{(2)}$$

A weighted UAC shooting method is based on the computation of a nonnegative matrix $\hat{W} \in \mathbb{R}^{19n \times 57n}$ that has the following representation

$$\hat{W} = \begin{bmatrix} \hat{W}_0 & \hat{W}_1 & \hat{W}_2 \end{bmatrix} \tag{2.12}$$

where each $\hat{W}_j$ has the form.

$$\hat{W}_j = \begin{bmatrix} \hat{w}_1^{(j)} & 0 & \cdots & 0 \\ 0 & \hat{w}_2^{(j)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \hat{w}_{19n}^{(j)} \end{bmatrix} \tag{2.13}$$

The coefficients $\hat{w}_k^{(j)} \geq 0$ satisfy the constaints.

$$(\hat{w}_1^{(j)}, \hat{w}_2^{(j)}, \hat{w}_3^{(j)}) = \arg\min \sum_{k=1}^{N-1} (\hat{e}_{j,19n}^{\top}((\hat{w}_0 y_{t_k}^{(0)} + \hat{w}_1 y_{t_k}^{(1)} + \hat{w}_2 y_{t_k}^{(2)}) - x_{t_k}))^2 \tag{2.14}$$

for some of the most recent times $t_1, \ldots, t_N$ at which the variable x_{t_k} has been observed or measured

3. Algorithms

We can apply lemma 2.1 combined with the techniques developed in [5] and [3], in order to derive three prototypical data-driven approximation algorithms for the propagation model that are described by algoritmo 1, algoritmo 2.

4. Numerical Experiments

We have created two spreadsheets named `COVID19HNHistory.xlsx` and `HNConnect0.xlsx`, where we have collected the data corresponding to a section of the time series of confirmed COVID-19 cases in Honduras thus far as reported in [2], and the geographical configuration of Honduran Departments, respectively.
We have written a GNU Octave program named `UACDescriptor.m` that implements algorithm algoritmo 1 based on the data in `COVID19HNHistory.xlsx` and `HNConnect0.xlsx`. The GNU Octave code of `UACDescriptor.m` is presented below.

Algorithm 1 First Data-driven (descriptor-corrector) approximation algorithm

Data: REAL NUMBER $\varepsilon > 0$, STATE DATA HISTORY: $\{x_t\}_{1\leq t\leq T}, T \in \mathbb{Z}^+$ CONNECTIVITY MATRIX: $K \in \mathbb{R}^{19n\times 19n}$

Result: APPROXIMATE DYNAMIC MATRIX: $A \in \mathbb{R}^{19n\times 19n}$ of $\tilde{\Sigma}$

(1) Compute $\tau_j \in \mathbb{R}^{19n\times 1}$ such that $K_j \circ \tau_j^\top = \tau_j^\top$ and

$$(\tau_1, \ldots, \tau_{19n}) = \arg\min \sum_{j=1}^{19n} (x_{t+1,j} - \tau_j^\top x_t)^2$$

for each $1 \leq t \leq T$ and, with $x_t, x_{t+1} \in \Sigma$

(2) Set $A = \prod_{j=1}^{19n} E_{\tau_j}$, with E_{τ_j} defined by (2.4).

return A

Algorithm 2 Second Data-driven (predictor) approximation algorithm

Data: REAL NUMBER $\varepsilon > 0$, STATE DATA HISTORY: $\{x_t\}_{1\leq t\leq T}, T \in \mathbb{Z}^+$

Result: APPROXIMATE STATE TRANSITION MATRIX: $\mathbf{S} \in \mathbb{R}^{19n\times 19n}$ of $\tilde{\Sigma}$

(1) Set $H = [x_{t_1} \cdots x_{t_1+S}]$ with $t_1 \geq 1$ and $t_1 + S \leq T - 1$

(2) Compute the lower rank reduced singular value decomposition $H = U_r S_r V_r$

(3) Compute the state-transition matrix $\mathbf{S} = U_r \hat{S} U_r^*$ determined by [5, Theorem 3.11.] applying [5, §4.1: Algorithm 4].

return $\mathbf{S}$

```
## Copyright (C) 2020 Fredy Vides
##
## This program is free software: you can redistribute it and/or modify it
## under the terms of the GNU General Public License as published by
## the Free Software Foundation, either version 3 of the License, or
## (at your option) any later version.
##
## This program is distributed in the hope that it will be useful, but
## WITHOUT ANY WARRANTY; without even the implied warranty of
## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
## GNU General Public License for more details.
##
## You should have received a copy of the GNU General Public License
## along with this program.  If not, see
## <https://www.gnu.org/licenses/>.

##
## function [x,A]=UACDescriptor(ss)
##
## Example:
## A=UACDescriptor(48);

## Author: fredy <fredy@HPCLAB>
```

```
## Created: 2020-03-28

function [x,A]=UACDescriptor(ss)
pkg load io;
x=xlsread ('COVID19HNHistoryFull.xlsx');
   K=xlsread ('HNConnect0.xlsx');
[p,m]=size(x);
x=x(2:p,:);
x0=x;
ss=min([ss m-1]);
y0=x0(:,1:ss);
y1=x0(:,2:(ss+1));
E=eye(p-1);
A=E;
K=K+E;
for k=1:(p-1)
w=find(K(k,:));
A0=E;
A0(k,w)=lsqnonneg((y0(w,:)).',y1(k,:).').';
A=A0*A;
end
end
```

We have written a GNU Octave program named `UACPredictor.m` that implements algoritmo 2 based on the data in `COVID19HNHistory.xlsx`. The GNU Octave code of `UACPredictor.m` is presented below.

```
## Copyright (C) 2020 Fredy Vides
##
## This program is free software: you can redistribute it and/or modify it
## under the terms of the GNU General Public License as published by
## the Free Software Foundation, either version 3 of the License, or
## (at your option) any later version.
##
## This program is distributed in the hope that it will be useful, but
## WITHOUT ANY WARRANTY; without even the implied warranty of
## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
## GNU General Public License for more details.
##
## You should have received a copy of the GNU General Public License
## along with this program.  If not, see
## <https://www.gnu.org/licenses/>.

##
## function [x,A]=UACPredictor(ss,tol)
##
## Example:
## A=UACPredictor(48,1e-19);
```

```
## Author: fredy <fredy@HPCLAB>
## Created: 2020-03-28

function [x,A]=UACPredictor(ss,tol)
pkg load io;
x=xlsread ('COVID19HNHistoryFull.xlsx');
   [p,m]=size(x);
ss=min([ss m-1]);
x=x(2:p,1:(ss+1));
[u,s,v]=svd(x,0);
   rk=max(find(diag(s)>=tol));
P=u(:,1:rk);
x0=P'*x;
y0=x0(:,1:ss);
y1=x0(:,2:(ss+1));
for k=1:rk
A(k,:)=lsqnonneg(y0.',y1(k,:).').';
end
   A=P*A*P';
end
```

We have also written GNU Octave programs named `UACWeight.m` and `UACWeightedShooting.m` that can be used to compute the nonnegative matrices (2.10) and (2.12). The Octave code of `UACWeight.m` and `UACWeightedShooting.m` is presented below.

```
## Copyright (C) 2020 Fredy Vides
##
## This program is free software: you can redistribute it and/or modify it
## under the terms of the GNU General Public License as published by
## the Free Software Foundation, either version 3 of the License, or
## (at your option) any later version.
##
## This program is distributed in the hope that it will be useful, but
## WITHOUT ANY WARRANTY; without even the implied warranty of
## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
## GNU General Public License for more details.
##
## You should have received a copy of the GNU General Public License
## along with this program.  If not, see
## <https://www.gnu.org/licenses/>.

##
## function W=UACWeight(A0,A1,x0)
##
## Example:
## [x,A0]=UACPredictor(48,1e-10);
## [x,A1]=UACDescriptor(48);
## W=UACWeight(A0,A1,x(:,47:49));
## A2=[A0;A1];
```

```
## [x(:,49) A0*x(:,48) A1*x(:,48) W*A2*x(:,48)]


## Author: fredy <fredy@HPCLAB>
## Created: 2020-03-28

function W=UACWeight(A0,A1,x0)
[m,N]=size(x0);
y0=x0(:,1);
y1=x0(:,1);
for k=1:(N-1)
y0=[y0 A0*y0(:,k)];
y1=[y1 A1*y1(:,k)];
end
for j=1:m
c(j,:)=lsqnonneg([y0(j,:).' y1(j,:).'],x0(j,:).').';
end
W=[diag(c(:,1)) diag(c(:,2))];
end
```



```
##
## function W=UACWeightedShooting(x,yl,ym,yh)
##

## Author: fredy <fredy@HPCLAB>
## Created: 2020-03-28

function W=UACWeightedShooting(x,yl,ym,yh)
[m,N]=size(x);
for j=1:m
c(j,:)=lsqnonneg([yl(j,:).' ym(j,:).' yh(j,:).'],x(j,:).').';
end
```

```
W=[diag(c(:,1)) diag(c(:,2)) diag(c(:,3))];
end
```

The weighted UAC predictor-descriptor systems in §2.2.3 and §2.3 can be computed using the program `WeightedUACPredictor.m`. The Octave code of `WeightedUACPredictort.m` is presented below.

```
## Copyright (C) 2020 Fredy Vides
##
## This program is free software: you can redistribute it and/or modify it
## under the terms of the GNU General Public License as published by
## the Free Software Foundation, either version 3 of the License, or
## (at your option) any later version.
##
## This program is distributed in the hope that it will be useful, but
## WITHOUT ANY WARRANTY; without even the implied warranty of
## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
## GNU General Public License for more details.
##
## You should have received a copy of the GNU General Public License
## along with this program.  If not, see
## <https://www.gnu.org/licenses/>.

##
## function [t,x]=WeightedUACPredictor(W,A0,A1,x0,T)
##
##  Example:
##  [x,A0]=UACPredictor(72,1e-10);
##  [x,A1]=UACDescriptor(72);
##  W0=UACWeight(A0,A1,x(:,70:73));
##  W1=UACWeight(A0,A1,x(:,71:73));
##  W2=UACWeight(A0,A1,x(:,72:73));
##  [t,y0]=WeightedUACPredictor(W0,A0,A1,x(:,72),16);
##  [t,y1]=WeightedUACPredictor(W1,A0,A1,x(:,72),16);
##  [t,y2]=WeightedUACPredictor(W2,A0,A1,x(:,72),16);
##  W=UACWeightedShooting(x(:,[72 73 88 89]),y0(:,[1 2 15 16]),...
##  y1(:,[1 2 15 16]),y2(:,[1 2 15 16]));
##  yt=W*[y0;y1;y2];
##  subplot(311);
##  j=6;plot([0 1 14 15],x(j,[72 73 88 89]),'c.','markersize',...
##  16,t,y0(j,:),'r.-',t,y2(j,:),'b.-',t,y1(j,:),'g.-',t,...
##  yt(j,:),'k.-');
##  axis tight
##  subplot(312);
##  j=8;plot([0 1 14 15],x(j,[72 73 88 89]),'c.','markersize',...
##  16,t,y0(j,:),'r.-',t,y2(j,:),'b.-',t,y1(j,:),'g.-',t,...
##  yt(j,:),'k.-');
##  axis tight
##  subplot(313);
```

```
##  j=19;plot([0 1 14 15],x(j,[72 73 88 89]),'c.','markersize',...
##  16,t,y0(j,:),'r.-',t,y2(j,:),'b.-',t,y1(j,:),'g.-',t,...
##  yt(j,:),'k.-');
##  axis tight

## Author: fredy <fredy@HPCLAB>
## Created: 2020-03-28

function [t,x]=WeightedUACPredictor(W,A0,A1,x0,T)
x=x0(:,1);
A=[A0;A1];
for k=1:(T-1)
x=[x W*A*x(:,k)];
end
t=0:(T-1);
end
```

One can run program `WeightedUACPredictor.m` using the following command lines in GNU Octave.

```
>> [x,A0]=UACPredictor(72,1e-10);
>> [x,A1]=UACDescriptor(72);
>> W0=UACWeight(A0,A1,x(:,70:73));
>> W1=UACWeight(A0,A1,x(:,71:73));
>> W2=UACWeight(A0,A1,x(:,72:73));
>> [t,y0]=WeightedUACPredictor(W0,A0,A1,x(:,72),16);
>> [t,y1]=WeightedUACPredictor(W1,A0,A1,x(:,72),16);
>> [t,y2]=WeightedUACPredictor(W2,A0,A1,x(:,72),16);
>> W=UACWeightedShooting(x(:,[72 73 88 89]),y0(:,[1 2 15 16]),...
>  y1(:,[1 2 15 16]),y2(:,[1 2 15 16]));
>> yt=W*[y0;y1;y2];
>> subplot(311);
>> j=6;plot([0 1 14 15],x(j,[72 73 88 89]),'c.','markersize',...
>  16,t,y0(j,:),'r.-',t,y2(j,:),'b.-',t,y1(j,:),'g.-',t,...
>  yt(j,:),'k.-');
>> axis tight
>> subplot(312);
>> j=8;plot([0 1 14 15],x(j,[72 73 88 89]),'c.','markersize',...
>  16,t,y0(j,:),'r.-',t,y2(j,:),'b.-',t,y1(j,:),'g.-',t,...
>  yt(j,:),'k.-');
>> axis tight
>> subplot(313);
>> j=19;plot([0 1 14 15],x(j,[72 73 88 89]),'c.','markersize',...
>  16,t,y0(j,:),'r.-',t,y2(j,:),'b.-',t,y1(j,:),'g.-',t,...
>  yt(j,:),'k.-');
>> axis tight
```

The previous lines produce the graphical outputs illustrated in figura 4.1.

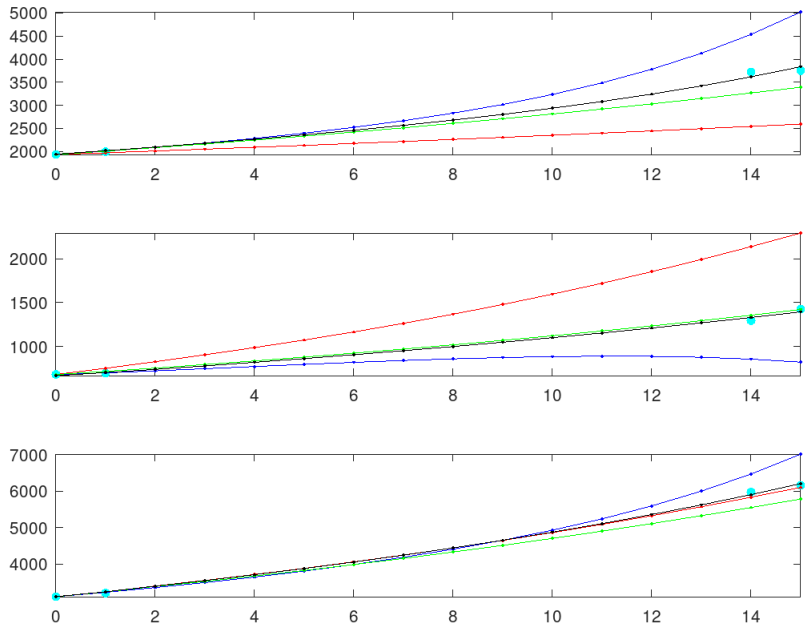


FIGURE 4.1. 16 days forecast for confirmed cases, the UAC weighted shooting method's prediction is represented by the black dotted line, the cyan dots represent observed shooting data. Cortés cases forecast (top). Francisco Morazán cases forecast (middle). Nationwide cases forecast (bottom).

The octave programs developed thus far for this project, together with the data spreadsheets are available on the project's repository at [4].

5. CONCLUSION AND FUTURE DIRECTIONS

The results in §2 can be used to derive predictive numerical simulation algorithms like algoritmo 1, algoritmo 2.
Thus far, given the present state and conditions of available data, by applying algorithms 1 and 2, one can get very good predictions in a time span that ranges from two to three weeks in the future.
Once more accurate COVID-19 behavior data become available, we plan to extend algoritmo 1 and algoritmo 2, to describe other aspects of the COVID-19 propagation in Honduras, for longer time periods. An extension of the ideas presented in this document to more complex geographical configuration graphs will be the subject of future communications.
The application of the techniques developed in [5] together with the algorithms presented in §3 to the study of other subjects of national interest, will also be the subject of future communications.

ACKNOWLEDGMENT

The structure preserving matrix computations needed to implement algoritmo 1 and algoritmo 2., were performed with the technology of universal algebraic controllers developed in the Scientific Computing Innovation Center (**CICC-UNAH**) of the National Autonomous University of Honduras.
The first author would like to thank Rosible Pacheco, Josué Molina, Luis Flores and Fabricio Murillo, for several interesting comments that have been very helpful for the preparation of this document.

REFERENCES

[1] Gerardo Chowell, R Luo, K Sun, Kimberlyn Roosa, Amna Tariq, and C Viboud. Real-time forecasting of epidemic trajectories using computational dynamic ensembles. *Epidemics*, 30:100379, 12 2019.
[2] Sinager. Despacho de comunicaciones y estrategia presidencial, 2020. https://covid19honduras.org/.
[3] F. Vides. On uniform connectivity of algebraic matrix sets. *Banach J. Math. Anal.*, 13(4):918–943, 2019.
[4] F. Vides. Gnu octave function and spreadsheets for the predictive simulation of covid-19 propagation in honduras, 2020. https://github.com/cadds-lab/EpidemicDynamics.
[5] F. Vides. Universal algebraic controllers and system identification. *Submitted*, 2020.