On Data-Driven Models for Epidemic Propagation in Honduras

Fredy Vides
Scientific Computing Innovation Center
School of Mathematics and Computer Science
Universidad Nacional Autónoma de Honduras, UNAH
Honduras
fredy.vides@unah.edu.hn

***Abstract*—In this document, an application of universal algebraic controllers (in the sense of [1]) to the computation of predictive models for epidemic propagation in Honduras, is presented. Special attention to COVID-19 propagation, is given.**

Some data-driven numerical predictive simulations for the COVID-19 propagation in Honduras, are outlined.

***Index Terms*—System identification, state transition matrix, structured matrices, switched closed-loop systems.**

I. 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 quarter of 2020. These models can be interpreted as discrete-time systems that can be *partially* described using the transition block diagram (I.1) as a *black-box device* $\mathfrak{S}$, that 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 (I.2).

$$\xrightarrow{x_t} \boxed{\quad \mathfrak{S} \quad} \xrightarrow{x_{t+1}} \tag{I.1}$$

In this study each entry $x_{t,j}$ of the state vector x_t corresponds to the known/predicted number of infected people in Department j, where the index j coincides with the Department's identification number, for instance $x_{t,1}$ is the estimated number of infected people in Atlántida at stage t. We will approach the computation of the *state-transtion* maps corresponding to the device (I.1), applying the algebraic methods developed in [1] and [2] to compute the state-transition matrices that correspond to *matrix solvents* of difference equations of the form

$$\Sigma : \begin{cases} x_{t+1} = T_t x_t, & t \geq 1 \\ x_1 \in \Sigma \subseteq \mathbb{R}^{18n} \end{cases} \tag{I.2}$$

where $\Sigma \subseteq \mathbb{R}^{18n}$ 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 (I.2), and in addition need to satisfy the following structural constraints.

$$\begin{cases} T_t = \prod_{j=1}^{18n} \left(I + \hat{e}_j(\tau_{(t,j)} - \hat{e}_j)^\top\right) \\ K_j \circ \tau_{(t,j)}^\top = \tau_{t,j}, \ \ 1 \leq j \leq 18n \end{cases} \tag{I.3}$$

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 $\tau_{(t,j)} \in \mathbb{R}^{18n \times 1}$ are to be determined by (I.2) and I.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

$$\hat{e}_{j,n} = \begin{bmatrix} \delta_{1,j} & \delta_{2,j} & \cdots & \delta_{n-1,j} & \delta_{n,j} \end{bmatrix}^\top \tag{I.4}$$

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

$$\delta_{k,j} = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases} \tag{I.5}$$

II. Universal Algebraic Controllers for the Propagation Model

A. 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.

$$K = I + adj(G) \tag{II.1}$$

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

$$a_{jk} = \begin{cases} 1, & \text{if } [v_j, v_k] \in E_G, \ v_j, v_k \in V_G \\ 0, & \text{otherwise} \end{cases} \tag{II.2}$$

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 fig. 1.

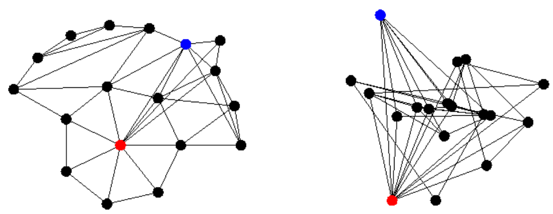

Figure 1. Homomorphic connectivity graphs corresponding to Honduras departments geographical confuguration. The red dot represents Francisco Morazán, the blue dot represents Cortés.

B. UAC Computation

1) A sequential UAC Descriptor-Predictor: We start considering a geographically constrained switched UAC model of the form.

$$\left\{\begin{array}{l} x_{t+1} = A_t x_t \\ x_0 \in \mathbb{R}^{18n\times 1} \end{array}\right. ,t \geq 0 \tag{II.3}$$

Where the matrices $A_t \in \mathbb{R}^{18n\times 18n}$ are computed according to the observed propagation's evolution by applying lemma II.1.

Lemma II.1. *Let us consider two propagation states $x_t, x_{t+1} \in \Sigma$ and the connectivity matrix $K \in \mathbb{R}^{18n\times 18n}$ determined by* (II.1). *There is a matrix $T_t \in \mathbb{R}^{18n\times 18n}$ that satisfies* (I.2) *and* (I.3)*, if and only if for each $1 \leq j \leq 18n$, there is $\tau_{(t,j)} \in \mathbb{R}^{18n\times 1}$ 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.

$$E_{\tau_{(t,j)}} = I + \hat{e}_j(\tau_{(t,j)} - \hat{e}_j)^\top \tag{II.4}$$

Given $x = [x_j] \in \mathbb{R}^{18n\times 1}$, we will have that.

$$\begin{aligned} E_{\tau_{(t,j)}} x &= (I + \hat{e}_j(\tau_{(t,j)} - \hat{e}_j)^\top x \\ &= \left\{\begin{array}{l} \tau_{(t,j)}^\top x, \ \ k = j \\ x_k, \ \ k \neq j \end{array}\right. \end{aligned} \tag{II.5}$$

Let us set $T_t = \prod_{j=1}^{18n} E_{\tau_{(t,j)}}$ by (I.3). By (II.4) and (II.5), we will have that the matrix $T_t \in \mathbb{R}^{18n\times 18n}$ that satisfies (I.2) and (I.3), if and only if for each $1 \leq j \leq 18n$, there is $\tau_{(t,j)} \in \mathbb{R}^{18n\times 1}$ 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. ∎

2) A geographically free Predictor: A geographically free UAC model of the form.

$$\left\{\begin{array}{l} x_{t+1} = T_t x_t \\ x_0 \in \mathbb{R}^{18n\times 1} \end{array}\right. ,t \geq 0 \tag{II.6}$$

Where the matrices $T_t \in \mathbb{R}^{18n\times 18n}$ are computed according to the observed propagation's evolution using the techniques developed in [1, §3.2].

3) A Discrete Diffusion Predictor: A geographically constrained diffusion model of the form,

$$\left\{\begin{array}{l} x_{t+1} = (I - d(x,t)(I-K))x_t \\ x_0 \in \mathbb{R}^{18n\times 1} \end{array}\right. ,t \geq 0 \tag{II.7}$$

with $K \in \mathbb{R}^{18n\times 18n}$ determined by the expression

$$K_{j,k} = \frac{|\mathrm{sign}(x_{t,j})|\mathrm{adj}(G)_{j,k}}{\sum_{p=1}^{18n}\mathrm{adj}(G)_{p,k}} \tag{II.8}$$

and where the coefficients $d(x,t)$ of the discrete diffusion matrices $I - d(x,t)(I-K) \in \mathbb{R}^{18n\times 18n}$ are computed according to the observed propagation's evolution and...

Lemma II.2. *Given $t \in \mathbb{Z}_0^+$, if $d(x,t) = \mathrm{argmin}\|x_{t+1} - x_t + d(I-K)x_t\|_2^2$, then we will have that.*

$$d(x,t) = -\frac{(x_{t+1} - x_t)^\top (I-K)x_t}{\|(I-K)x_t\|_2^2} \tag{II.9}$$

Proof. Let us set $f(d) = \|y + dz\|_2^2$ for $y = x_{t+1} - x_t$ and $z = (I-K)x_t$. We will have that $\mathrm{argmin} f(d) = \mathrm{argmin}\|x_{t+1} - x_t + d(I-K)x_t\|_2^2$. Let us consider the equation.

$$0 = f'(d) = -2(y + dz)^\top z$$

This implies that.

$$d = -\frac{y^\top z}{z^\top z} = -\frac{y^\top z}{\|z\|_2^2}$$

This completes the proof. ∎

III. Algorithms

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

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: $K \in \mathbb{R}^{18n\times 18n}$

Result: Approximate matrix realizations: $\{T_t\}_{t=1}^{T-1} \subset \mathbb{R}^{18n\times 18n}$ of $\tilde{\Sigma}$

1) For each $1 \leq t \leq T-1$
   a) Compute $\tau_{(t,j)} \in \mathbb{R}^{18n\times 1}$ such that $K_j \circ \tau_{(t,j)}^\top = \tau_{(t,j)}^\top$ and $|x_{t+1,j} - \tau_{(t,j)}^\top x_t| \leq \varepsilon$ for each $1 \leq j \leq 18n$ and , with $x_t, x_{t+1} \in \Sigma$
   b) Set $T_t = \prod_{j=1}^{18n} E_{\tau_{(t,j)}}$, with $E_{\tau_{(t,j)}}$ defined by (II.4).

return $\{T_t\}_{t=1}^{T-1}$

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: $\mathbf{T} \in \mathbb{R}^{18n\times 18n}$ 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 reduced singular value decomposition $H = USV$
3) Compute the perturbation $H_\varepsilon = US_\varepsilon V$ of H according to [1, §3.2: (3.38)].
4) Compute the state-transition matrix $\mathbf{T}$ determined by [1, Corollary 3.8.] according to [1, §3.2: (3.44)].

return T

Algorithm 3 Third Data-driven diffusive (predictor) approximation algorithm

Data: Real number $\varepsilon > 0$, State data history: $\{x_t\}_{1\leq t\leq T}$, $T \in \mathbb{Z}^+$ Connectivity matrix: $K \in \mathbb{R}^{18n\times 18n}$

Result: Approximate diffusive state transition matrix: $\mathbf{D} \in \mathbb{R}^{18n\times 18n}$ of $\tilde{\Sigma}$

1) Set $x = x_{t_1}, y = x_{t_2}$ with $t_1 \geq 1 \leq t_2 \leq T - 1$
2) Compute coefficient d according (II.9)
3) Compute the (diffusive) state-transition matrix $\mathbf{D} = I - d(I - K)$ according to (II.8).;

return D

IV. Numerical Experiments

We have created two spreadsheets named `COVID19History.xlsx` and `HNConnect.xlsx`, where we have collected the data corresponding to observed COVID-19 propagation history in Honduras thus far and to the geographical configuration of Honduran Departments, respectively.

We have written a GNU Octave program named `COVID19.m` that implements algorithm 1 based on the data in `COVID19History.xlsx` and `HNConnect.xlsx`. The GNU Octave code of `COVID19.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 [K,T,x0,x]=COVID19(m,n,tol,graph)
##
## Example:
## [K,T01,x0,x1]=COVID19(0,1,eps);
## [K,T12,x1,x2]=COVID19(1,2,eps);
## [K,T23,x2,x3]=COVID19(2,3,eps);
## [K,T03,x0,x3]=COVID19(0,3,eps);
## norm(x3-T03*x0,1)+norm((T23*T12*T01-T03)*x0,1)

## Author: fredy <fredy@HPCLAB>
## Created: 2020-03-17

function [K,T,x0,x]=COVID19(m,n,tol,graph)
m=m+1;
n=n+1;
pkg load io;
COVIDHist=xlsread ('COVID19History.xlsx');
HNConnect=xlsread ('HNConnect.xlsx');
A=HNConnect (1:18,1:18);
[M,N]=size(A);
E=eye(M,N);
K=A+E;
if nargin<=3
graph=1;
end
if graph==1
r=.5;
z1=(r*exp(2*pi*i*(0:6)/7)).';
z2=(2.0*r*exp(20*pi*i*(0:8)/(9*21))).';
z3=2.4*r*exp((pi+.1)*i/4);
xy=zeros(M,2);
xy([15 18 4 12 17 2 7],:)=[real(z1),imag(z1)];
xy([9 3 1 6 16 5 14 13 10],:)=[real(z2),imag(z2)];
xy(11,:)=[real(z3),imag(z3)];
subplot(211);
gplot (A,xy,'k-');
hold on;
plot(xy(:,1),xy(:,2),'k.','markersize',20,...
xy(8,1),xy(8,2),'r.','markersize',20,xy(6,1)...
```

```
,xy(6,2),'b.','markersize',20);
hold off;
axis off;
axis square;
subplot(212);
XY=randn(M,2);
gplot (A,XY,'k-');
hold on;
plot(XY(:,1),XY(:,2),'k.','markersize',20,XY(8,1),
XY(8,2),'r.','markersize',20,XY(6,1),XY(6,2),...
'b.','markersize',20);
hold off;
axis off;
axis square;
end
x0=COVIDHist (1:18,m);
f0=find(abs(x0)<=tol);
x=COVIDHist (1:18,n);
f1=find(abs(x)<=tol);
f2=find(abs(x)>tol);
x0(f0)=0;
x(f1)=0;
T=E;
for k=f2
T0=E;
T0(k,:)=K(k,:).*(x(k)/x0);
T=T0*T;
end
T0=ones(M,1);
y0=T*x0;
T0(f2)=x(f2)./y0(f2);
T=diag(T0)*T;
K=A+E;
end
```

One can run program `COVID19.m` using the following command lines in GNU Octave.

```
>> [K,T01,x0,x1]=COVID19(0,1,eps);
>> [K,T12,x1,x2]=COVID19(1,2,eps);
>> [K,T23,x2,x3]=COVID19(2,3,eps);
>> [K,T03,x0,x3]=COVID19(0,3,eps);
>> norm(x3-T03*x0,1)+norm((T23*T12*T01-T03)*x0,1)
ans =    3.0531e-15
```

We have written a GNU Octave program named `UACPredictor.m` that implements algorithm 3 based on the data in `COVID19History.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 [Xh,T,EIHUB,EIHLB,EGHUB,EGHLB]=
# UACPredictor(n,r,tol)
##
## Example:
## [Xh,T,EIHUB,EIHLB,EGHUB,EGHLB]=...
## UACPredictor(9,12,1e-12);

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

function [Xh,T,EIHUB,EIHLB,EGHUB,EGHLB]=...
UACPredictor(n,r,tol)
pkg load io;
Xh=xlsread ('COVID19History.xlsx');
[p,m]=size(Xh);
Xh=Xh(1:(p-1),(n+1):(r+1));
[p,m]=size(Xh);
Xh0=Xh(:,1:(m-1));
[uh,sh,vh]=svd(Xh0,0);
sh0=diag(sh);
f=find(sh0<=tol);
sh0(f)=tol;
sh0=diag(sh0);
T=Xh0\Xh(:,m);
T=[[zeros(1,m-2);eye(m-2)] T];
T=uh*sh0*vh'*T*(vh/sh0)*uh';
EIHUB=Xh(:,1);
EGHUB=Xh(:,1);
EIHLB=EIHUB;
EGHLB=EGHUB;
for k=1:(m-1)
EIHUB = [EIHUB (Xh(:,k+1)>0).*ceil(T*Xh(:,k))];
EIHLB = [EIHLB (Xh(:,k+1)>0).*floor(T*Xh(:,k))];
EGHUB = [EGHUB (Xh(:,k+1)>0).*ceil(T*EGHUB(:,k))];
EGHLB = [EGHLB (Xh(:,k+1)>0).*floor(T*EGHLB(:,k))];
```

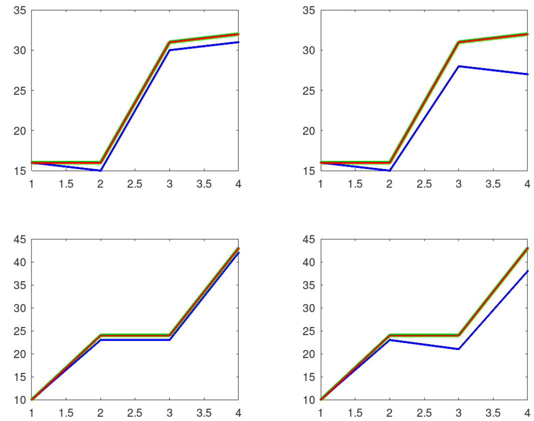

Figure 2. Four stages forcast for Francisco Morazán (top): Local time estimates (left) and Global time estimates (right). Four stages forcast for Cortés (bottom): Local time estimates (left) and Global time estimates (right). Green dotted lines represent observed values, blue dotted lines represent lower bounds for expected-predicted values, and red dotted lines represent upper bounds for expected-predicted values

```
end
end
```

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

```
>> s=9;R=12;
>> [Xh,T,EIHUB,EIHLB,EGHUB,EGHLB]=...
UACPredictor(s,R,1e-12);
>> t=1:(R-s+1);
>> subplot(221),plot(t,Xh(8,:),'k.-',...
'linewidth',6,t,EIHLB(8,:),'r.-',...
'linewidth',2,t,EIHUB(8,:),'b.-',...
'linewidth',2)
>> subplot(222),plot(t,Xh(8,:),'k.-',...
'linewidth',6,t,EGHLB(8,:),'r.-',...
'linewidth',2,t,EGHUB(8,:),'b.-',...
'linewidth',2)
>> subplot(223),plot(t,Xh(6,:),'k.-',...
'linewidth',6,t,EIHLB(6,:),'r.-',...
'linewidth',2,t,EIHUB(6,:),'b.-',...
'linewidth',2)
>> subplot(224),plot(t,Xh(6,:),'k.-',...
'linewidth',6,t,EGHLB(6,:),'r.-',...
'linewidth',2,t,EGHUB(6,:),'b.-',...
'linewidth',2)
```

The previous lines produce the graphical outputs illustrated in fig. 2.

We have written a GNU Octave program named `DiffCOVID19.m` that implements algorithm 2 based on the data in `COVID19History.xlsx`, `HNConnect0.xlsx` and `HNConnect1.xlsx`. The GNU Octave code of `DiffCOVID19.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/>.
##
##
## [K1718,A,x17,x18,xx,yy,zz,f]=...
## DiffCOVID19(17,18,1,eps,1);
##
## Example:
## [K1718,A,x0,x,xx,yy,zz,f]=...
## DiffCOVID19(17,18,1,eps,1);

## Author: fredy <fredy@HPCLAB>
## Created: 2020-04-01

function [K,A,x0,x,xx,yy,zz,f]=...
DiffCOVID19(m,n,p,tol,graph)
m=m+1;
n=n+1;
pkg load io;
COVIDHist=xlsread ('COVID19History.xlsx');
%HNConnect=xlsread ('HNConnect0.xlsx');
HNConnect=xlsread ('HNConnect1.xlsx');
A=HNConnect (1:18,1:18);
[M,N]=size(A);
E=eye(M,N);
if nargin<=4
graph=1;
end
```

```
x0=COVIDHist (1:18,m);
x=COVIDHist (1:18,n);
K=diag(x0>0)*A*diag(1./sum(A));
K=E-K;
d=K*x0;
d=-(x-x0)'*d/(d'*d);
K=E-d*K;
[xx,yy]=meshgrid(-1.5:3/30:1.5);
f=[];
r=.5;
z1=(r*exp(2*pi*i*(0:6)/7)).';
z2=(2.0*r*exp(20*pi*i*(0:8)/(9*21))).';
z3=2.4*r*exp((pi+.1)*i/4);
xy=zeros(M,2);
xy([15 18 4 12 17 2 7],:)=[real(z1)...
,imag(z1)];
xy([9 3 1 6 16 5 14 13 10],:)=...
[real(z2),imag(z2)];
xy(11,:)=[real(z3),imag(z3)];
for k=1:18
z=sqrt((xx-xy(k,1)).^2+(yy-xy(k,2)).^2);
m=min(min(z));
f=[f;find(z==m)];
end
  [X,Y]=meshgrid (-1.5:3/150:1.5);
  zz=zeros(size(xx));
  zz(f)=Mxvec(K,x0,p);
  Z=interp2(xx,yy,zz,X,Y,'spline');
if graph==1
  figure;
  contour(X,Y,Z,64);
  colormap summer;
  hold on;
  gplot (A,[xx(f) yy(f)],'k-');
  plot(xx(f),yy(f),'k.','markersize',...
  20,xx(f(8)),...
  yy(f(8)),'r.','markersize',20,...
  xx(f(6)),yy(f(6)),...
  'b.','markersize',20);
  hold off;
  axis off;
  axis square;
end
end
function y=Mxvec(A,b,n)
y=b;
for k=1:n
y=A*y;
end
end
```

One can run program `DiffCOVID19.m` using the following command lines in GNU Octave.

```
>> [K1718,A,x17,x18,xx,yy,zz,f]=...
DiffCOVID19(17,18,1,eps,1);
```

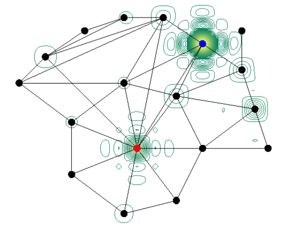

Figure 3. Diffusion map generated by the diffusive model.

The previous lines produce the graphical outputs illustrated in fig. 3.

One can now combine all predictive algorithms using the following command lines in GNU Octave.

```
>> [K,T1718,x17,x18]=COVID19(17,18,eps,0);
>> [K,T1819,x18,x19]=COVID19(18,19,eps,0);
>> [Xh,T,EIHUB,EIHLB,EGHUB,EGHLB]=...
UACPredictor(17,19,1e-18);
>> [K1718,A,x17,x18,xx,yy,zz,f]=...
DiffCOVID19(17,18,1,eps,0);
>> [x19 T1718^2*x17 T^2*x17 K1718^2*x17]
ans =

    16.00000     11.00000     11.01820     11.04176
     2.00000      2.25486      2.00331      2.00328
    20.00000     20.00000     20.03310     19.98064
     2.00000      2.01402      2.15116      1.05549
     1.00000      1.00000      1.00165      1.00138
   195.00000    196.36313    195.32268    194.77527
     0.00000      0.00000      0.00000      0.00000
    54.00000     53.37141     53.97481     46.99580
     0.00000      0.00000      0.00000      0.00000
     0.00000      0.00000      0.00000      0.00000
     0.00000      0.00000      0.00000      0.00000
     1.00000      1.12800      1.00165      1.00471
     4.00000      4.49258      4.00662      4.00288
     0.00000      0.00000      0.00000      0.00000
     0.00000      0.00000      0.00000      0.00000
     8.00000      8.05616      8.01324      8.04501
     0.00000      0.00000      0.00000      0.00000
     9.00000      8.05546      8.01324      8.05306
```

The spreadsheet data files together with a copy of the program `COVID19.m` are available at [3].

V. Conclusion and Future Directions

The results in §II can be used to derive predictive numerical simulation algorithms like algorithm 1, algorithm 2 and algorithm 3.

Thus far, given the present state and conditions of available data, by applying algorithm 1, algorithm 2 and algorithm 3, one can get very good predictions in a time span that ranges from two to five days in the future.

Once more accurate COVID-19 behavior data become available, we plan to extend algorithm 1, algorithm 2 and algorithm 3, 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.

Acknowledgment

The structure preserving matrix computations needed to implement algorithm 1 and algorithm 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.

I would like to thank Josué Molina, Norman Sabillón, Luis Flores and Fabricio Murillo, for several interesting comments that have been very helpful for the preparation of this document.

References

[1] F. Vides, "Universal algebraic controllers and system identification," *Submitted*, 2020.

[2] ——, "On uniform connectivity of algebraic matrix sets," *Banach J. Math. Anal.*, vol. 13, no. 4, pp. 918–943, 2019.

[3] F. Vides. (2020) Gnu octave function and spreadsheets for the predictive simulation of covid-19 propagation. Https://fredyvides.github.io/projects.html.