Finite Orbit decomposition of endomaps

by N. MARTINS-FERREIRA AND M. GASPAR

Centre for Rapid and Sustainable Product Development,
Polytechnic Institute of Leiria, Marinha Grande, Portugal

Abstract: In this work we present a vectorized Matlab algorithm for the decomposition of an endomap into its finite orbits.

Keywords:

1 Introduction

Every endomap $f\colon X \to X$ induces a partition of X into disjoint orbits. Such decomposition is in general not unique. However, in this text we provide a vectorized Matlab implementation for an algorithm which can be considered as a canonical decomposition of the domain of an endomap into its finite (disjoint) orbits.

To do that we have to consider three different kind of (finite) orbits that we will call initial orbits, linking orbits and cyclic orbits.

The following definitions are useful in establishing the three different kind of orbits.

Definition Let $f\colon X \to X$ be an endomap. An element $x \in X$ is said to be:

1. a *initial point* if $card(f^{-1}(x)) = 0$;
2. a *merging point* if $card(f^{-1}(x)) > 1$;
3. a *linking point* if $card(f^{-1}(x)) = 1$.

Here $card$ stands for cardinality and $f^{-1}(x)$ is the set of all elements $x' \in X$ with $f(x') = x$.

For the purposes of this note a sequence of $f\colon X \to X$ is any subset $S \subseteq X$ together with a distinguished element $x_1 \in S$ and a map

$$\eta\colon S \to \mathbb{N}$$

such that

$$\eta(x_1) = 1$$

and for any other $x \in S$, different from x_1, there exists a unique $x' \in S$ with

$$x = f(x') \quad \text{and} \quad \eta(x) = \eta(x') + 1.$$

Each sequence (S, x_1, η) can be ordered as

$$x_1, x_2, \ldots, x_n, \ldots$$

with $x_{i+1} = f(x_i)$. The element x_1 is called the starting element and, in case S is finite, say of length n, then x_n is called the last element in the sequence.

We are now in position to define the three different kind of orbits which will be considered in this note.

Initial Orbits An initial orbit for an endomap $f\colon X \to X$ is a finite sequence (S, x_1, η) where x_1 is a initial point and $f(x_n)$, the image by f of the last element in the sequence, is a merging point. All the other points are required to be linking points.

Linking orbits A linking orbit of $f\colon X \to X$ is a finite sequence of f, say (S, x_1, η), such that both the initial point and the image by f of the last point are merging points and all the others are linking points. Observe that the image by f of the last point is no longer an element in the sequence.

Cyclic orbits A cyclic orbit of f, is any finite sequence (S, x_1, η), in which every element is a linking point and moreover $x_1 = f(x_n)$, with x_n the last point in the sequence.

Infinite orbits In this study, for practical reasons, we are excluding the infinite orbits: the ones where x_1 is an initial point and all the other are linking points.

2 Partitioning a finite domain of an endomap in its disjoint orbits

When X is not finite there is no guarantee that an endomap $f\colon X \to X$ can induce a partition on it into classes of disjoint initial, linking and cyclic orbits. However, if X is a finite set then we always have such a partition.

Proposition Let $f\colon X \to X$ be an endomap. The initial, linking and cyclic orbits of f are all disjoint. Moreover, when X is finite, they induce a partition of X.

The remaining part of this text is devoted to a vectorized (see also [1]) Matlab implementation of an algorithm to decompose a finite set X into the disjoint orbits of an endomap.

2.1 The initial and linking orbits

For practical reasons we assume that

$$X = \{1, \ldots, n\}$$

is the set of natural numbers between 1 and n and hence an endomap of X is just a vector with length n whose all entries are numbers in the range `1:n`.

The following example will be used for the purpose of illustrating the concepts:

```
f = [6, 7, 10, 2, 4, 12, 12, 13, 10, 8,  10,
    13, 13, 15, 16, 14]
```

The notion of orbit, or sequence in the sense defined above, is best illustrated by picturing the endomap $f\colon X \to X$ as a directed graph

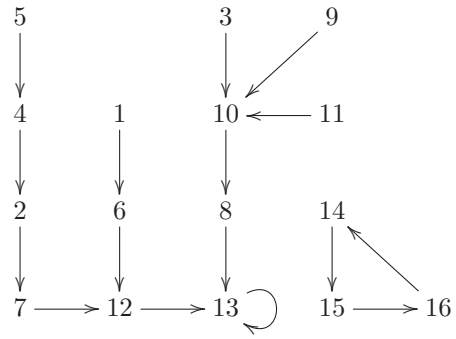
$$X \underset{f}{\overset{1_X}{\rightrightarrows}} X$$

whose vertices are the elements of X, the edges are also the elements of X which are interpreted as follows: an element $x \in X$ is considered as an arrow (or edge)

$$x \xrightarrow{x} f(x)$$

whose domain (or source) is x and whose codomain (or target) is $f(x)$.

Our example can so be pictured as a graph in the following manner:



As we see it is now a simple task to visually identify the initial points:

$$1, 3, 5, 9, 11,$$

and the merging points:

$$10, 12, 13.$$

All the others are linking points.

The initial orbits are the ones which start with an initial point and run over until a merging point is found in the sequence (stopping immediately before the merging point is reached):

```
1, 6
3
5, 4, 2, 7
9
11
```

The linking orbits are the ones which start with a merging point and run over until a merging point is found in the sequence (stopping immediately before the merging point is reached):

```
10, 8
12
13
```

The remaining elements, that are not being used in any of the initial or linking orbits are organized into cyclic orbits. This is a general fact and may be stated as follows, where I is the set of all inital points, M is the set of all merging points, $IM = I \cup M$ and O_x is the set of elements in the orbit whose first element is x.

Proposition Let $f\colon X \to X$ be an endomap. If

$$Y = \bigcup_{x \in IM} O_x$$

then the restriction of f to the set $Z = X \setminus Y$ is a well defined endomap

$$f\colon Z \to Z$$

and moreover it is a bijection.

The Matlab vectorized code to obtain the initial and merging points is the following:

```
% Input: t
t=t(:); % force t to be a column vector
% check if t represents an endomap
try
    t(t);
catch ME
    error(ME.message)
end
nA=length(t);
% computing the initial points
I=setdiff(1:nA,t);
% computing the merging points
ts=sort(t);
dst=diff(ts);
M=ts(dst==0);
```

3 The cyclic orbits and its vectorized implementation

Once all the initial and linking orbits are obtained, the previous result tells us that the elements in X that where not used in the initial or linking orbits are organized into cyclic orbits. The following Matlab code is a vectorized implementation of the simple procedure of decomposing a permutation into its disjoint cycles.

```
function Orb=orbits(p)
% input: a vector p which is a
% permutation on the set A={1,...,nA};
% Output: Orb, a matrix with the
% orbits of p, one orbit in each line,
% without repeating the cycles

nA=length(p);
% force p to be a column vector
p=reshape(p,nA,1);

% Check that it is a permutation
idA=(1:nA)';
invp(p)=idA;
if ~isequal(p(invp),idA)
    disp('The input must be a permutation!')
    Orb=[];
    return
end

% the vector A will be equal to p^x,
% on each iteration x
A=p(idA);
x=1;
```

The following block of code can be improved by replacing `nA-1` by `nA/2` with the due adjustments on the rest of the code; the point is that if we arrive to the iteration `x=nA/2` then, from that moment on, there is only one cycle.

```
Z=[idA, zeros(nA,nA-1)];

L=(A~=Z(:,1));
while any(L)
    Z(L,x+1)=A(L);
    % current iteration
    A=p^{}
    A(L)=p(A(L));
    x=x+1;
    % L0 is the previous L to be
    % compared with the current one
    L0=L;
    L(L)=(A(L)~=Z(L,1));
    if ~isequal(L,L0)
        R=reduce(Z(:,1:x),L0);
        Z(R,:)=0;
        A(R)=0;
        L(R)=0;
    end
end

Orb=Z(Z(:,1)~=0,1:x);
```

The auxiliary function `reduce` is described below.

```
function R=reduce(Z,L)
% auxiliar function:
% Z and L are as above;
% R indicates the elements to be
% removed from the begining of a
% cycle because they are already
% being used in another cycles

% elements to be removed,
% initialized as false
R=false(size(Z(:,1)));

while any(L)
    Rfi=find(L,1,'first');
    % all the elements in a cycle,
    % except the starting one are
    % removed, so that they do not
    % start new cycles
    R(Z(Rfi,2:end))=true;
    L(Z(Rfi,1:end))=false;
end
```

The procedure for computing the initial and linking orbits is similar with the only difference that instead of checking whether the current iteration is equal to the starting one, one has to check if it is any one of the merging points.

An example of a (non-vectorized) implementation for computing the initial and linking orbits may be obtained as follows.

```
% I and M are computed as above
% initial merging points to be used
Orb=[]; IM=[];
labels=[]; currlab=1; % current label
for u=I
    while ~ismember(u,IM)
        Orb(end+1)=u;
        if ismember(u,M)
            currlab=currlab+1;
            IM(end+1)=u;
        end
        labels(end+1)=currlab;
        u=t(u);
    end
    currlab=currlab+1;
end
```

At this point we have computed all the initial and linking orbits, what remains is organized in cyclic orbits and hence we can use the function `perm2orb` which transforms a permutation into its disjoint orbits (see [2]).

```
% From here on we use the perm2orb
x=setdiff(t,Orb);
% find p such that t(x)=x(p) using digraph
[~,p]=digraph(x(:),t(x));
% get the orbits of p
orb=orbits(p);
% label the orbits of p
orb=orb';
[sorb1,sorb2]=size(orb);
nz=orb~=0;
orblabels=repmat(1:sorb2,sorb1,1);
orblabels=orblabels(nz);
v=x(orb(nz));

%output
s=[Orb(:); v(:)];
labels=[labels(:); orblabels(:)];
```

One last issue has to be solved for a complete implementation. We have to be able to transform the restriction of the original endomap from its original domain into the set of linking points which do not belong to any initial or linking orbit. This is done with the simple procedure of transforming any directed graph with any set of edges into a set of indexed labels, as follows.

```
function [d,c]=digraph(dom,cod)
% [d,c]=digraph(dom,cod)
% creates linear indexes d and c for dom
% and cod if dom and cod are matrices

if size(dom)~=size(cod)
    error('dom and cod must be of same size')
```

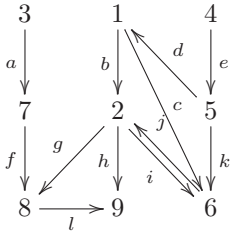
```
 8  end

10  nA=size(dom,1);

12  [~,~,dc3]=unique([dom;cod],'rows');
13  % indexing the dom and cod as a linear
14  % vector of unique entries where the
15  % first nA are from dom while the last
16  % ones (nA+1:2*nA) are from cod
17  d=dc3(1:nA);
18  c=dc3(nA+(1:nA));
```

Finally we may present the output of the example considered before, which is:

Orb	Label
1	1
6	1
12	2
13	3
3	4
10	5
8	5
5	6
4	6
2	6
7	6
9	7
11	8
14	9
15	9
16	9

4 Applications Examples

Every directed graph induces an endomap. For example, the endomap f introduced in section 2.1, is derived from the graph depicted in the following diagram:



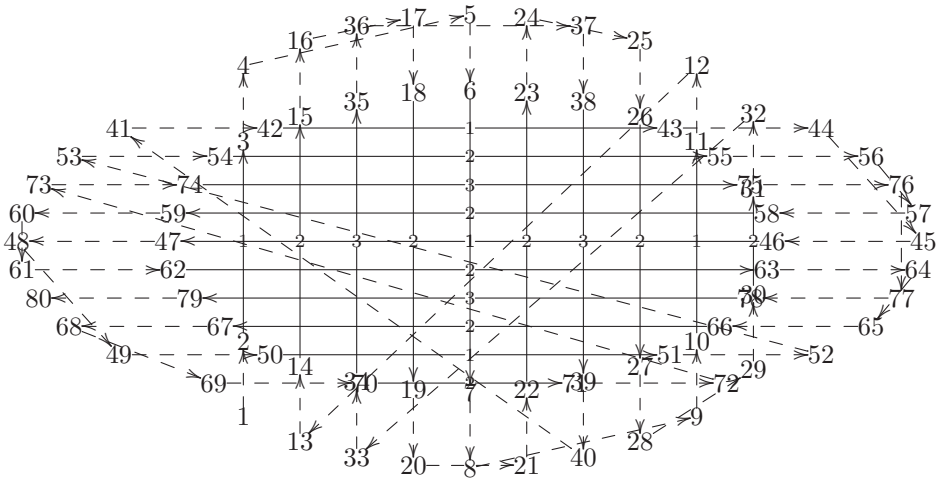
The procedure to do so is very simple: for each vertex, the incoming and outgoing edges are listed and matched. Whenever the number of incoming edges is less than the matching outgoing ones, an identity arrow is inserted. This is the case for the vertex 9, which has two incoming edges and no outgoing ones. Correspondingly, in f a loop is present (labeled as 13).

If $f\colon A \to A$ is a given endomap, then every map $g\colon A \to \mathbb{C}$, from the domain of the given endomap into the complex plane, induces a planar curve which is a realization of the graph depicting the endomap.

References

[1] M. Gaspar and N. Martins-Ferreira, *The Matlab vectorization paradigm*, CSEI, 2012.

[2] M. Gaspar and N. Martins-Ferreira, *Curves and Permutations, GTLab Technical Report*, 2012.



Toolpath trajectory for a multi-material extrusion of a cylindrical scaffold with orthogonal orientation from layer to layer.