Generation of Members of the Symplectic Group Sp(2n)

Author: Michel Barbeau, Carleton University

Version: June 21, 2019

This a companion MATLAB Live Script to the paper:

M. Barbeau, *Secure Quantum Data Communications Using Classical Keying Material*, First International Workshop on Quantum Technology and Optimization Problems (QTOP), Garching b. München, Germany, March 2019.

In the following, $\mathbb{F}_2$ represents the two-element field made of 0 and1. When operands are in $\mathbb{F}_2$, additions (+) are modulo two. This is also equivalent to a logical exclusive or $\oplus$.

The following tests the MATLAB implementation of symplectic group members generation in $\mathrm{Sp}(n)$. The generation is exhaustive, i.e., all members of $\mathrm{Sp}(n)$ are generated and verified.

```
clear;
%%% usage examples in n=1
% n = 1;
% d =  sorder(n)
% S_0 = symplectic(0,n)
% S_1 = symplectic(1,n)
% S_2 = symplectic(2,n)
%%% usage examples in n=2
n = 2; % greater than 2 takes a long time
mysymplecticQA(n);
```

```
Generating 720 symplectics
720 symplectics generated
uniqueness test passed
```

```
% d =  sorder(n) % oder of the group
% a few examples
% S = symplectic(0,n)
% L = Lambda(n)
% apply a transvection to the matrix
% S = mxvection([0 1 1 1]',S)
% mod(S*L*S',2)
% symplectic(1,n)
% symplectic(2,n)
% a=xvection([0 1 1 1]',[1 0 0 0]')
```

Symplectic group background

Let $\Lambda(n)$ be the $2n \times 2n$ block diagonal matrix:

$$\begin{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & 0 & \cdots & 0 \\ 0 & \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{pmatrix}$$

In MATLAB:

```
function [L] = Lambda(n)
% returns the Lambda(n) matrix
    L = kron(eye(n),[0 1; 1 0]);
end
```

Let *v* and *w* be two column vectors in the field $\mathbb{F}_2^{2n}$. Their **symplectic inner product** is defined as

$$< v, w >= v^T \Lambda(n) w.$$

In MATLAB:

```
function [p] = sip(v,w)
% symplectic inner product
   p = mod(v'*Lambda(length(v)/2)*w,2);
end
```

Let *h* be a column vector in $\mathbb{F}_2^{2n}$. The vector *h* is used to define the symplectic transvection Z_h,

with domain and co-domain $\mathbb{F}_2^{2n}$, i.e.,$Z_h : \mathbb{F}_2^{2n} \mapsto \mathbb{F}_2^{2n}$. Its application on a column vector v in $\mathbb{F}_2^{2n}$ is defined as $Z_h v = v + < v, h > h$. In MATLAB:

```
function [w] = xvection(h,v)
% Applies the transvection "Z_h" to column vector "v"
   w = mod(v + sip(v,h)*h,2);
end
```

This function applies a transvection to all the columns of a matrix:

```
function [sigma] = mxvection(h,sigma)
    for i=1:length(sigma)
        % apply transvections to column "i"
        sigma(:,i) = xvection(h,sigma(:,i));
    end
end
```

Lemma 5 (Lemma 2 in [Koenig and Smolin, 2014])

Given two non-zero vectors *x* and *y* in $\mathbb{F}_2^{2n} \backslash \{0\}$, we have that either

$$y = Z_h x \text{ for a } h \in \mathbb{F}_2^{2n}$$

or

$$y = Z_{h_1} Z_{h_2} x \text{ for } h_1, h_2 \in \mathbb{F}_2^{2n}.$$

That is to say, vector *x* can be mapped to vector *y* using one or two transvections h_1 and h_2. Here is a MATLAB implementation with justifications:

```
function [h1,h2] = findxvection(x,y)
% Gigen two colum vectors "x" and "y",
% find h1 and h2 such that y = Z_h1 Z_h2 x
    % initialize transvection vectors
    h1 = zeros(length(x),1);
    h2 = zeros(length(x),1);
```

If *x* and *y* are equal, then h_1 and h_2 are zero vectors:

```
    if isequal(x,y) % vectors are equal?
        return; % h1 and h2 are zero vectors
    end
```

If the symplectic inner product $\langle x, y \rangle$ is equal to one, then $h_1 = x \oplus y$ and h_2 is a zero vector:

```
    if sip(x,y) % symplectic inner product is one?
        h1 = xor(x,y); % h1 = x + y, h2 is a zero vector
        return;
    end
```

If the symplectic inner product $\langle x, y \rangle$ is equal to zero, then find a column vector *z* in $\mathbb{F}_2^{2n}$ such that $\langle x, z \rangle = \langle y, z \rangle = 1$:

```
    z = zeros(length(x),1); % init "z"
```

Firstly, try to find an index $j \in 2, 4, \cdots, 2n$ where $(x_{j-1}, x_j) \neq (0,0)$ and $(y_{j-1}, y_j) \neq (0,0)$:

```
    for k=2:2:length(x)
        if (x(k-1)||x(k)) && (y(k-1)||y(k)) % pair is found!
```

Find values for a pair (z_{j-1}, z_j) such that $x_{j-1} z_j \oplus x_j z_{j-1} = y_{j-1} z_j \oplus y_j z_{j-1} = 1$, with all other elements of *z* set to null:

```
            for i=1:3
                v = de2bi(i,2); % v = [0 1], [1 0], [1 1]
                if xor(x(k-1)*v(2),x(k)*v(1)) && xor(y(k-1)*v(2),y(k)*v(1))
                    z(k-1)= v(1); z(k) = v(2);
                    break;
                end
            end
            h1 = xor(x,z);
            h2 = xor(y,z);
            return;
        end
```

```
    end
```

Note here that we have both $\langle x, z\rangle = 1$ and $\langle y, z\rangle = 1$. Furthermore, making $h_1 = x \oplus z$ and $h_2 = y \oplus z$, so are $\langle x, h_1\rangle = 1$ and $\langle z, h_2\rangle = 1$, because $1 = \langle y, z\rangle = \langle z, y\rangle = \langle z, y \oplus z\rangle = \langle z, h_2\rangle$. We get that

$$x \oplus h_1 \oplus h_2 = z \oplus h_2 = z \oplus (y \oplus z) = y$$

hence

$$z = Z_{h_2} Z_{h_1} x.$$

Else, find indices $j, k \in 2, 4, \cdots, 2n$ where $(x_{j-1}, x_j) \neq (0,0)$ and $(y_{j-1}, y_j) = (0,0)$ and $(x_{k-1}, x_k) = (0,0)$ and $(y_{k-1}, y_k) \neq (0,0)$. Such indices exist because columns vectors *x* and *y* are non zero:

```
    % find index "j"
    for j=2:2:length(x)
        if (x(j-1)||x(j)) && ~(y(j-1)||y(j))
            break;
        end
    end
    % find index "k"
    for k=2:2:length(x)
        if ~(x(k-1)||x(k)) && (y(k-1)||y(k))
            break;
        end
     end
```

Find values for a pair (z_{j-1}, z_j) such that $x_{j-1} z_j \oplus x_j z_{j-1} = 1$ and pair (z_{k-1}, z_k) such that $y_{k-1} z_k \oplus y_k z_{k-1} = 1$, with all other elements of *z* set to null:

```
    for i=1:3
        v = de2bi(i,2); % v = [0 1], [1 0], [1 1]
        if xor(x(j-1)*v(2),x(j)*v(1))
            z(j-1)= v(1); z(j) = v(2);
            break;
        end
    end
    for i=1:3
        v = de2bi(i,2); % v = [0 1], [1 0], [1 1]
        if xor(y(k-1)*v(2),y(k)*v(1))
            z(k-1)= v(1); z(k) = v(2);
            break;
        end
    end
    h1 = xor(x,z);
    h2 = xor(y,z);
    return;
end
```

Again, we have $\langle x, z\rangle = 1$ and $\langle y, z\rangle = 1$. Furthermore, making $h_1 = x \oplus z$ and $h_2 = y \oplus z$, so are $\langle x, h_1\rangle = 1$ and $\langle z, h_2\rangle = 1$. We get that $x \oplus h_1 \oplus h_2 = z \oplus h_2 = z \oplus (y \oplus z) = y$ and $z = Z_{h_2} Z_{h_1} x$.

Order of group $\mathrm{Sp}(2n)$

The order of $Sp(2n)$ is $2^{n^2} \cdot \prod_{j=1}^{n} (4^j - 1)$.

```
function [m] = sorder(n)
% Returns the order of symplectic group Sp(2n)
    m = 1;
    for j=1:n
        m = m * (power(4,j) - 1);
    end
    m = power(2,power(n,2)) * m;
end
```

Cardinal of set S_i

The cardinal of S_i is ratio $|Sp(2i)|/|Sp(2(i-1))|$.

```
function [m] = scardinal(i)
% Returns the cardinal of set "S_i"
    if i==1
        m = sorder(1);
    else
        m = sorder(i)/sorder(i-1);
    end
end
```

Mapping an index to a symplectic

The following function maps an index $i \in 0, \ldots, |S_n| - 1$ to a symplectic in the group $Sp(2n)$.

```
function [sigma] = symplectic(k,i)
% Given an index "k" and dimension "i", returns the
% k-th symplectic in group Sp(2i).
```

The elements of $Sp(2i)$ are indexed in the range $0, \ldots, |Sp(2i)| - 1$.

```
    % validation of inputs
    if mod(k,1) || k<0
        error('1st argument must be non negative integer i=%d', k)
    end
    if mod(i,1) || i<=0
        error('2nd argument must be a positive integer n=%d', i)
    end
    if k<0 || k>=sorder(i) % symplectics indexed in 0...|Sp(2i)|-1
        error('symplectic index out of range n=%d i=%d', i, k)
    end
    % order of symplectic pair set "S_i"
```

```
    d = scardinal(i);
```

There are a symplectic pair and four transvections calculated by the MATALB function indextosp. The elements of $Sp(2i)$ are indexed in the range $0, \ldots, |Sp(n)| - 1$. A symplectic index *k* can be decomposed into two numbers

$$k = l \cdot |S_i| + m \text{ with } 0 \leq l < |Sp(2(i-1))| \text{and } 0 \leq m < |S_i|.$$

The number *l* (mod(k,d) in MATLAB) is recursively used as the index of the symplectic $\sigma_{2(i-1)}$. The number *m* (floor(k/d) in MATLAB) is used to index the element of S_i, i.e., the symplectic pair determining the matrix σ_{2i}.

```
    % find symplectic pair and transvections for index "i"
    [v, w, t, h0, h1, h2] = indextosp(mod(k,d),i);
```

Given a symplectic pair (v, w) in S_1, the corresponding symplectic in $Sp(2)$ is the matrix

$$\sigma_2 = \begin{pmatrix} v_1 & w_1 \\ v_2 & w_2 \end{pmatrix}.$$

```
    if i==1 % done!
        sigma = [ v w ];
        return;
    else
```

For *i* greater than one, given a pair (v, w) in S_i and a symplectic $\sigma_{2(i-1)}$ in $Sp(2i-1))$, the corresponding symplectic in $Sp(2i)$ is the matrix resulting from the composition

$$\sigma_{2i} = Z_s Z_{h_0} Z_{h_1} Z_{h_2} \begin{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \sigma_{2(i-1)} & \\ 0 & & & \end{pmatrix}.$$

Let $I_{*,j}$ denote the *j*-th column of the identity matrix of dimension $2i$. The transvections Z_{h_1} and Z_{h_2} are such that their application to $I_{*,1}$ is equal to *v*, i.e., $Z_{h_1} Z_{h_2} I_{*,j}$ is equal to *v*. The transvections Z_{h_1} and Z_{h_2} are calculated applying Lemma 5. Their application transvects the first column $I_{*,1}$ into *v*. By construction (detailed in the sequel), Z_{h_0} has no effect on *v*. Because it is defined using *v*, the transvection Z_s has no effect on *v*. Result, the first column of σ_{2i} is *v*. The composition $Z_s Z_{h_0} Z_{h_1} Z_{h_2} I_{*,2}$ yields the column vector *w*. Result, the second column of σ_{2i} is *w*. The transvections Z_s, Z_{h_0}, Z_{h_1}, and Z_{h_2} are applied to the remaining $2(i-1)$ columns of the matrix embedding the matrix $\sigma_{2(i-1)}$.

```
        %disp(symplectic(floor(k/d),i-1))
        %disp([v, w, t, h0, h1, h2])
        sigma = compose(eye(2),symplectic(floor(k/d),i-1));
        % apply transvections to all columns
        sigma = mxvection(t,mxvection(h0,mxvection(h1,...
                mxvection(h2,sigma))));
```

```
    end
end
```

Mapping an index to a symplectic pair

The following function maps an index $m \in 0, \ldots, |S_i| - 1$ to a symplectic pair in the set S_i and calculate four transvections as specified for MTALAB function symplectic.

```
function [v,w,s,h0,h1,h2] = indextosp(m,i)
% Given an index "i" and dimension "n", returns the
% i-th symplectic pair in set "S_n" and four transvections
% index validation
    if m<0 || m>=scardinal(i)
        error('symplectic pair index out of range n=%d i=%d', i, m)
    end
```

Applying Lemma 6, the symplectic pair index *m* can be decomposed into two numbers

$$m = r \cdot 2^{2i-1} + s \text{ with } 0 \leq r < (2^{2i} - 1) \text{and } 0 \leq s < 2^{2i-1}.$$

```
    % decompose index "m" into two factors
    d = power(2,2*i-1);
    r = floor(m/d);
    s = mod(m,d);
```

The binary expansion of $r + 1$ over $2i$ bits becomes the column vector *v*.

```
    % "v" becomes the binary expansion of "r + 1"
    v = de2bi(r+1,2*i,'left-msb')';
```

The transvections Z_{h_1} and Z_{h_2} are calculated applying Lemma 5.

```
    % find transvections that yield "v" from "I_{*,1}"
    [h1,h2] = findxvection([ 1 zeros(1,length(v)-1)]',v);
```

Let *b* be the binary representation of *s* over $2i - 1$ bits.

```
    % binary expansion of "s"
    b = de2bi(s,2*i-1,'left-msb');
```

It used to obtain the column vector h_0 in the transvection Z_{h_0}, that is,

$$h_0 = Z_{h_1} Z_{h_2} (I_{*,1} + b_2 I_{*,3} + \ldots b_{2i-1} I_{*,2i}).$$

```
    % apply xvections to I_{*,1}+b(2)I_{*,3}+...+b(2*n-1)I_{*,2n}
    I = eye(2*i); % identity matrix of dimension 2i
    h0 = xvection(h1,xvection(h2, sum(I(:,find([1 0 b(2:end)])),2) ));
```

The transvection Z_s is defined with *s* resulting from the composition $\neg b_1 v$.

```
    % construct transvection "s"
```

```
    s = ~b(1)*v;
```

The vector *w* is defined as $w = Z_s Z_{h_0} Z_{h_1} Z_{h_2} I_{*,2}$.

```
    % generate "w"
    w =  xvection(s,xvection(h0,xvection(h1,...
        xvection(h2,[ 0 1 zeros(1,length(v)-2)]'))));
end
```

Composition of matrices

```
function [m3] = compose(m1,m2)
% Input:
%    m1, m2 = two square matrices
% Output: m3 equal to
%     ( m1 0  )
%     ( 0  m2 )
    m3 = vertcat( ...
    horzcat(m1,zeros(length(m1),length(m2))),...
    horzcat(zeros(length(m2),length(m1)), m2) );
end
```

Quality assurance code

```

function [m] = corder(n)
% Returns the order of the Cliffor group C(n)
m = 1;
for j=1:n
    m = m * (power(4,j) - 1);
end
m = power(2,power(n,2)+2*n) * m;
end

function findxvectionQA(n)
% Verifies function findxvection for all non null binary column vectors of
% length "n"
% max value on a vector of length 2n
max = power(2,2*n)-1;
for i=1:max
    for j=1:max
        v = de2bi(i,2*n,'left-msb')';
        w = de2bi(j,2*n,'left-msb')';
        [h1,h2] = findxvection(v, w);
        if ~isequal(w,xvection(h1,xvection(h2,v)))
            disp(v); disp(w); disp(h1); disp(h2);
            error('findxvection errror i=%d j=%d', i, j);
        end
    end
end
end

function indextospQA(n)
```

```
% Verifies function indextosp
for i=0:scardinal(n)-1
    [v, w, t, h0, h1, h2] = indextosp(i,n);
    % disp([v, w, t, h0, h1, h2]);
    if ~sip(v,w)
        disp([v, w, t, h0, h1, h2]);
        error('indextosp errror n=%d i=%d', n, i);
    end
end
end

function mysymplecticQA(n)
% Verifies function mysymplectic
% generate the Lambda(n) matrix
L = Lambda(n);
% order of Sp(2n)
d =  sorder(n);
% exhaustive verification
fprintf('Generating %d symplectics\n', d);
for i=0:d-1
    sigma(:,:,i+1) = symplectic(i,n);
    if ~isequal(mod(sigma(:,:,i+1)*L*sigma(:,:,i+1)',2),L)
        disp(sigma(:,:,i+1));
        error('symplectic test failed n=%d i=%d', n, i);
    end
end
fprintf('%d symplectics generated\n', d);
% verification of uniqueness
for i=0:d-1
    if i<d-1
        % compare with every othe matrics from that indx
        for j=i+1:d-1
            if isequal(sigma(:,:,i+1),sigma(:,:,j+1))
               disp(sigma(:,:,i+1),sigma(:,:,j+1));
               error('repetition n=%d i=%d j=%d',n, i,j);
            end
        end
    end
end
fprintf('uniqueness test passed\n')
end
```