Dilations of unitary tuples - supplementary file

Malte Gerhold, Satish Pandey, Orr Shalit and Baruch Solel

This file acompanies the paper Dilations of Unitary Tuples by Malte Gerhold, Satish Pandey, Orr Shalit and Baruch Solel. The purpose of these MATLAB computations is to find an estimate for the largest dilation constant for three q-commuting unitaries.

Recall that a pair u, v of unitaries is said to q***-commute*** it $vu = quv$. We will write $q = e^{i\theta}$ and alternatively treat θ as the parameter. We will consider "rational" θ, meaning that $\theta = 2\pi \frac{m}{n}$. A triple of unitaries u_1, u_2, u_3 is said to q***-commute*** if each one of the pairs (u_1, u_2) , (u_1, u_3) and (u_2, u_3) is q-commuting. We write c_θ for the ***dilation constant*** corresponding to this tuple, i.e., the smallest constant such that $u \prec c_\theta z$, where where z is 3-tuple of **commuting** unitaries.

Below we define functions make2qCommuting(m,n) and make3qCommuting(m,n) for making a pair or a triple of q-commuting unitaries (where $q = e^{i\theta}$ and $\theta = 2\pi \frac{m}{n}$). When q is a root of unity, one can prove that every irreducible representation of q commuting pairs or $q$commuting tuples can be obtained from these pairs or triples by applying a guage automorphism (i.e., multiplying each unitary by some scalar of modulus one).

Here is an example:

```
format short
U = make3qCommuting(1,3);
U(:,:,1)
```

```
ans = 3×3 complex
  -0.5000 + 0.8660i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i  -0.5000 - 0.8660i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   1.0000 - 0.0000i
```

```
U(:,:,2)
```

```
ans = 3×3 complex
   0.0000 + 0.0000i  -0.5000 + 0.8660i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.5000 - 0.8660i
   1.0000 - 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
```

```
U(:,:,3)
```

```
ans = 3×3
     0     1     0
     0     0     1
     1     0     0
```

By Proposition 6.6 in the paper, we have that:

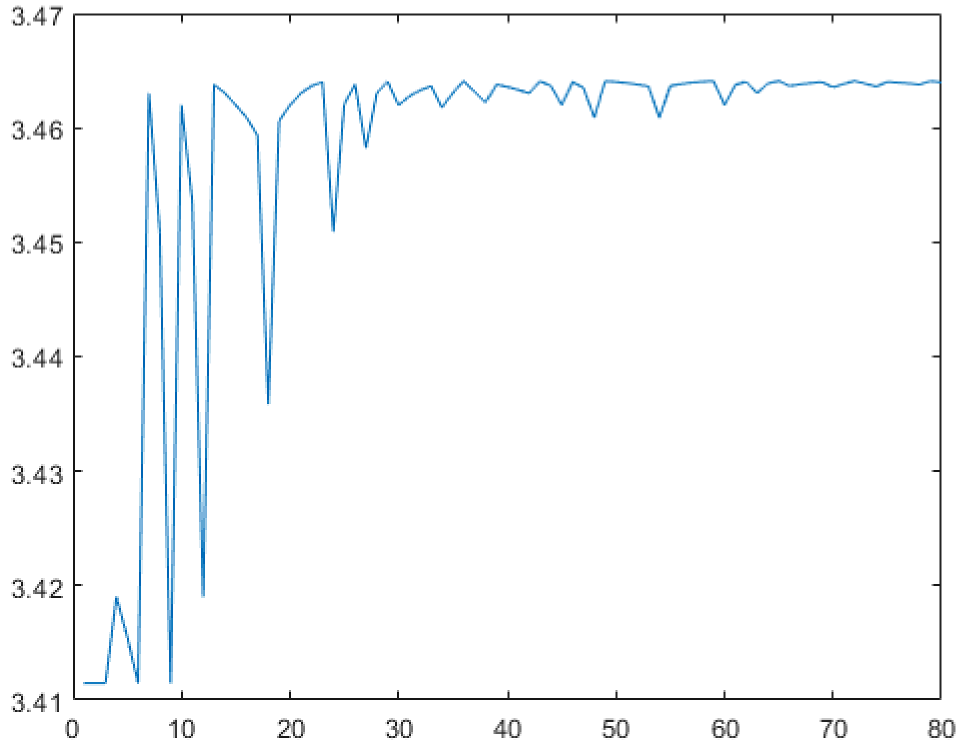
$$c_\theta = \frac{6}{\|u_1 + u_1^* + u_2 + u_2^* + u_3 + u_3^*\|}$$

As all the representations of $C^*(u)$ are given by $u_i \mapsto a_i U_i$, where U = make3qCommuting(m,n) (recall $\theta = 2\pi \frac{m}{n}$) and $a_i \in \mathbb{T}$, we can calculate an approximation (from below) of the the norm of the denominator by the function norm3tuple(U,N) (defined below).

We have $\|u_1 + u_1^* + u_2 + u_2^* + u_3 + u_3^*\| = \sup\{\|H(t) : t \in [0,1]^3\}$ where $H(t) = \sum_{j=1}^{3} e^{2\pi i t_j} U_j + e^{-2\pi i t_j} U_j^*$ and $t = (t_1, t_2, t_3) \in [0,1]^3$ and $U = make3qCommuting(m,n)$. The function norm3tuple(U,N) approximates this supremum for the input $U = (U_1, U_2, U_3)$ by computing $H(t)$ from t selected from a uniformly spaced lattice in the cube $[0,1]^3$ where the points lie ot the vertices of cubes of size $1/N$. (The default value of N is 10, to allow a quick and dirty estimation of the norm).

Here is a little experiment to see how the calculated norm depends on the parameter N (the fineness of the grid).

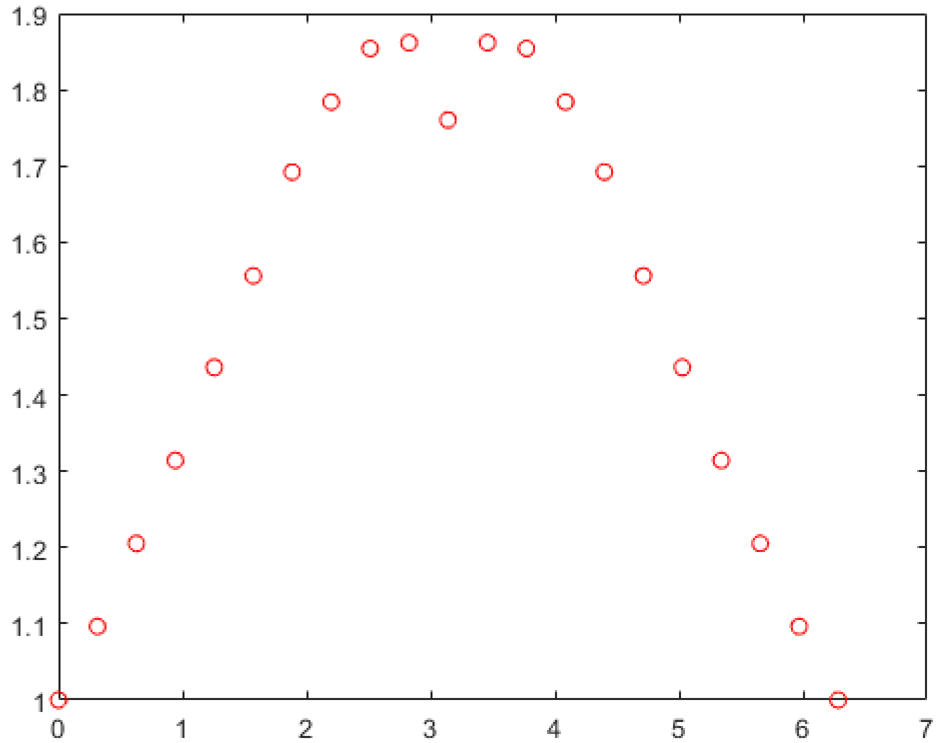
```
nrm = zeros(80,1);
for i=1:80
    nrm(i) = norm3tuple(U,i);
end
plot(nrm)
```



It is interesting to note, first, that the norm ideed changes with different changes of gauge, and, second, that the estimate seems to stabilize, and, third, that the relative variation in the value looks like 2%, so using low value of N can give us a rough estimate with which to start.

The following loop quickly scans values of θ, searching for a value that ***minimizes*** the norm $\|u_1 + u_1^* + u_2 + u_2^* + u_3 + u_3^*\|$, so that it will maximizes the value of c_θ.

```
N = 20;
c_vals = zeros(N+1,1);
for m=0:N
    if m > N/2;
        c_vals(m+1) = c_vals(N+1-m);% A_\theta is isomorphic to A_{-\theta}
    end
    % Find reduced fraction for m/N
    g = gcd(m,N);
    mr = round(m/g);
    nr = round(N/g);
    % create the q-commuting 3-tuple and compute norm of sum_i u_i + u_i^*
    U = make3qCommuting(mr,nr);
    nrm = norm3tuple(U);
    c_vals(m+1) = 6/nrm;
end
thetas = 2*pi*[0:1/N:1];
plot(thetas,c_vals,'ro')
```




```
[max_c, ind] = max(c_vals)
```

```
max_c = 1.8616
ind = 10
```

```
best_theta = thetas(ind)
```

```
best_theta = 2.8274
```

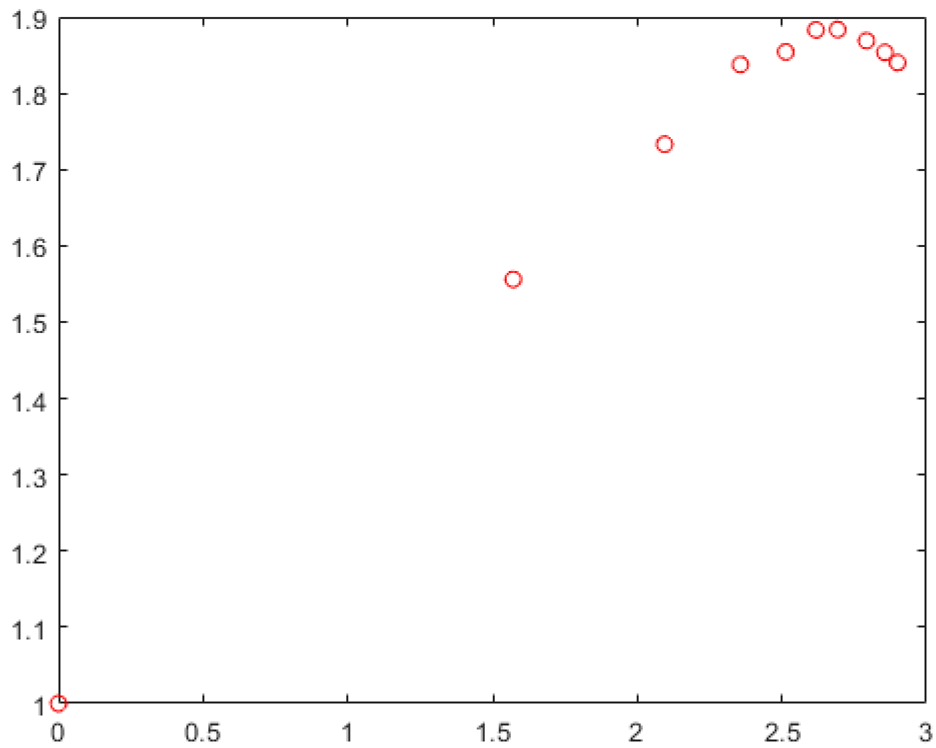
```
biggest_norm = 6/max_c
```

```
biggest_norm = 3.2230
```

Having found that $\theta = 2\pi \frac{9}{20}$ is a reasonable guess for the value that gives the largest dilation cosntant, we will try some other fractions with low denominator that are slightly less than $1/2$.

```
N = 13;
c_vals = zeros(N,1);
thetas = zeros(N,1);
for n=1:N
    m = ceil(n/2)-1;
    % Find reduced fraction for m/N
    g = gcd(m,n);
    mr = round(m/g);
    nr = round(n/g);
    thetas(n) = 2*pi*mr/nr;
    % create the q-commuting 3-tuple and compute norm of sum_i u_i + u_i^*
```

```
    U = make3qCommuting(mr,nr);
    X = zeros(size(U(:,:,1)));
    nrm = norm3tuple(U);
    c_vals(n) = 6/nrm;
end
plot(thetas,c_vals,'ro')
```




```
[max_c, ind] = max(c_vals)
```

```
max_c = 1.8835
ind = 7
```

```
best_fraction = (ceil(ind/2)-1)/ind
```

```
best_fraction = 0.4286
```

```
best_theta = thetas(ind)
```

```
best_theta = 2.6928
```

```
biggest_norm = 6/max_c
```

```
biggest_norm = 3.1855
```

So now we know that it is a good idea to try $\theta = 2\pi\frac{3}{7}$. So we compute its norm more exactly, by using a finer grid, with N=1000. This will make the norm estimate for $\|u_1 + u_1^* + u_2 + u_2^* + u_3 + u_3^*\|$ more precise - larger - and hence make our estimate of the value of c_θ for this angle θ more preicse - smaller. This takes significantly longer, since we are going to construct and then compute the norm of one billion matrices. Lucky for us their size is only 7×7.

```
U = make3qCommuting(3,7);
N = 1000;
nrm = norm3tuple(U,N)
```

```
nrm = 3.1882
```

```
c_theta = 6/nrm
```

```
c_theta = 1.8819
```

Error estimate

Up to here we found

$$\max_\theta c_\theta \geq c_{6\pi/7}$$

and we have an estimate

$$c_{6\pi/7} \approx 1.8819 \text{ .}$$

This is interesting because it is larger than the currently know lower bound $C_3 \geq \sqrt{3} = 1.7321$. However our estimate comes from a numerical approximation, and we now wish to give a perfectly reliable lower bound for $c_{6\pi/7}$.

Write $\nu = \|u_1 + u_1^* + u_2 + u_2^* + u_3 + u_3^*\|$. We have $\nu = \sup\{\|H(t) : t \in [0,1]^3\}$ where $H(t) = \sum_{j=1}^3 e^{2\pi i t_j} U_j + e^{-2\pi i t_j} U_j^*$ and $U = make3qCommuting(3,7)$. Let $\nu^* = norm3tuple(U,N)$, so it is obtained like the above supremum, but only over a grid of fineness $\delta := \frac{1}{N} \times 2\pi$. It is easy to see that $\nu \leq \nu^* + 6\delta$. Therefore

$$c_\theta = \frac{6}{\nu} \geq \frac{6}{\nu^* + 6\delta} = \frac{6}{\nu^*} \times \left(1 - \left(\frac{6\delta}{\nu^*}\right) + \left(\frac{6\delta}{\nu^*}\right)^2 - \ldots\right) \geq \frac{6}{\nu^*} \times \left(1 - \left(\frac{6\delta}{\nu^*}\right)\right)$$

Plugging in $3.18 \leq \nu^* \leq 3.19$ (counting on the matlab computation of the 7×7 matrix to be precise up to 3 digits) and the value of δ we find the right hand side is great or equal than the following lower bound:

```
N = 1000;
d = 2*pi/N;
lower_bound = (6/3.19)*(1 - (6*d)/(3.18))
```

```
lower_bound = 1.8586
```

To conclude, we found the reliable lower bound $c_{6\pi/7} \geq 1.8586$, which gives the new lower bound $C_3 \geq 1.8586$.

Here are the functions that were used in the above script:

```
function [u,v] = make2qCommuting(m,n)
% make2qCommuting - create the canonical pair of q-commuting matrices,
% where q=exp(i*theta), and theta = 2*pi*m/n

D = 1:n;
u = diag(exp(2*pi*i*D*m/n));
v = circshift(eye(n),1,2);

end
```

```
function [u1,u2,u3] = make3qCommuting(m,n)
% make3qCommuting - create the "canonical" triple of q-commuting matrices,
% where q=exp(i*theta), and theta = 2*pi*m/n
% if nargout ==1 then the 3-tuple is all recorded in u1, as a stack of
% matrices

[u,v] = make2qCommuting(m,n);
u1 = u;
u2 = u*v;
u3 = v;

if nargout == 1
    u1 = cat(3,u1,u2,u3);
end

end
```

```
function max_norm = norm3tuple(U,N)
% norm3tuple - computes the norm of h = sum u_i+u_i^*
% U - nXnX3 3-tuple of q-commuting unitaries (output of make3qCommuting)
% N - optional input argument for fineness of approximation (default value is 10)

arguments
    U (:,:,3) double
    N double = 10
end

dt = 1/N;

t = 2*pi*[0:dt:1];
t = exp(1i*t);
len = length(t);
sz1 = size(U,1);
max_norm = 0;

for i=1:len
```

```
    for j=1:len
        for k=1:ceil(len/sz1)
            X1 = t(i)*U(:,:,1);
            X2 = t(j)*U(:,:,2);
            X3 = t(k)*U(:,:,3);
            max_norm = max(max_norm,norm(X1+X1'+X2+X2'+X3+X3'));
        end
    end
end

end
```