```
function [K0,K1,H] = Cartan(G,l_space,m_space,h_space)
suGen;
global g;
global g_m;
global g_l;
global basis_m;
global basis_l;
global m;
global v;
global K_min;
basis_m = m_space;
basis_l = l_space;
G = G/expm(1i*angle(G(1,1)));
g = logm(G);    %g is the generator in su(2^n) of G.
g_l = 0;    %g_l and g_m are the components of g along the subspaces l_space and m_space
g_m = 0;
for j = 1:size(l_space,3)
    g_l = g_l+trace(g'*basis_l(:,:,j))/trace(basis_l(:,:,j)'*basis_l(:,:,j))*basis_l(:,:,j);
end
for j = 1:size(m_space,3)
    g_m = g_m+trace(g'*basis_m(:,:,j))/trace(basis_m(:,:,j)'*basis_m(:,:,j))*basis_m(:,:,j);
end
lim = 10^6;
eps1 = inf;
m = g_m;
num = 0;
while eps1 > 0.15 && num < lim
    num = num+1;
    k = PolQ(m);
    G_trial = expm(k)*expm(m);
    G_actual = expm(g_l+g_m);
    for i = 1:length(G_trial);
        if abs(G_trial(i,1))>0.1
            break
        end
    end
    G_trial = G_trial/exp(1i*angle(G_trial(i,1)));
    G_actual = G_actual/exp(1i*angle(G_actual(i,1)));
    eps1 = sum(sum(abs(G_actual-G_trial)));
    m = logm(expm(-k)*expm(g_m+g_l));
    m_m = 0;
for j = 1:size(basis_m,3)
    m_m = m_m+real(trace(m'*basis_m(:,:,j)))/trace(basis_m(:,:,j)'
    *basis_m(:,:,j))*basis_m(:,:,j)+real(trace(m))/length(m)*eye(size(m));
end
    m = m_m;
end
init_mat = m;
init_pos = to_vect(init_mat,basis_m);
options = optimoptions('fsolve','Algorithm','levenberg-marquardt','MaxFunctionEvaluations'
,3e5,'OptimalityTolerance',1e-4,'TolX',1e-50,'TypicalX',1*ones(size(basis_m,3),1));
m_vect = fsolve(@map,init_pos,options);
m = 0;
for k = 1:size(basis_m,3)
    m = m + m_vect(k)*basis_m(:,:,k);
```

```
end
K0 = G*expm(-m);
v = 0;
for k = 1:size(h_space,3)
    v = v + (pi/2)^(k-1)*h_space(:,:,k);
end
K_init = zeros(size(basis_l,3),1);
options = optimset('MaxIter',4000,'TolX',1e-30,'TolFun',1e-10,'MaxFunEvals'
,1e6,'TypicalX',1e-3*ones(size(basis_l,3),1));
K_min = fminunc(@f_vm, K_init, options);
K1_gen = 0;
for k = 1:size(basis_l,3)
    K1_gen = K1_gen + K_min(k)*basis_l(:,:,k);
end
K1 = expm(K1_gen);
h = K1'*m*K1;
H = expm(h);
end
function [A,B,K,F1,F2,H,M,N1,N2] = ModKhanejaGlaser(U)
suGen;
[K0,K1,H] = Cartan(U,su3_l,su3_m,su3_h);
h = logm(H);
[Ka1,Kb1,F1] = Cartan(K0*K1,su3_l0,su3_l1,su3_f);
K(:,:,1) = Ka1*Kb1;
K(:,:,2) = Kb1';
f1 = logm(F1);
[Ka2,Kb2,F2] = Cartan(K1',su3_l0,su3_l1,su3_f);
K(:,:,3) = Ka2*Kb2;
K(:,:,4) = Kb2';
f2 = logm(F2);
for i = 1:4
    [A0,B0] = kron_div(K(:,:,5-i),2);
    A(:,:,i) = round(A0,4);
    B(:,:,i) = round(B0,4);
    M(i) = round(trace(h'*su3_h(:,:,i))/trace(su3_h(:,:,i)'*su3_h(:,:,i)),6);
end
for i =1:3
    N2(i) = round(trace(f1*su3_f(:,:,i))/trace(su3_f(:,:,i)'*su3_f(:,:,i)),6);
    N1(i) = round(trace(f2*su3_f(:,:,i))/trace(su3_f(:,:,i)'*su3_f(:,:,i)),6);
end
end
function str = decompose(Unitaries)
str = [];
init = ['init = QuantumCircuit(3)\n'];   %This can be modified later, in the Jupyter Notebook
str = [str,sprintf(init)];
meas = ['\n\nmeas = QuantumCircuit(3,3)\n' ...
        'meas.measure(0,0)\n' ...
        'meas.measure(1,1)\n' ...
        'meas.measure(2,2)\n\n']
str = strcat(str,sprintf(meas));   %Print meas circuit to file
suGen;
function str = composeCircuit(U,name,str)
suGen;
global su2_l;
global su2_m;
```

```
global su2_h;
[A,B,K,F1,F2,H,M,N1,N2] = ModKhanejaGlaser(U);
CNOT = [1,0,0,0;0,1,0,0;0,0,0,1;0,0,1,0];
U_leftMA = kron(expm(1i*pi/4*Z),I)*CNOT;
U_rightMA = kron(expm(-1i*pi/4*Z),I)*CNOT;
A(:,:,2) = U_leftMA*A(:,:,2);
H = H/kron(U_leftMA,I);
A(:,:,3) = A(:,:,3)*U_rightMA;
H = kron(U_rightMA,I)\H;
formatSpecA = ['\n\ncircuit_A%u = QuantumCircuit(3)\n' ...
               'circuit_A%u.u3(%4.3f,%4.3f,%4.3f,2)\n' ...
               'circuit_A%u.u3(%4.3f,%4.3f,%4.3f,1)\n' ...
               'circuit_A%u.rz(pi/2,1)\n' ...
               'circuit_A%u.cx(1,2)\n' ...
               'circuit_A%u.rz(-2*(%4.3f)+pi/2,2)\n' ...
               'circuit_A%u.ry(pi/2-2*(%4.3f),1)\n' ...
               'circuit_A%u.cx(2,1)\n' ...
               'circuit_A%u.ry(2*(%4.3f)-pi/2,1)\n' ...
               'circuit_A%u.cx(1,2)\n' ...
               'circuit_A%u.rz(-pi/2,2)\n' ...
               'circuit_A%u.u3(%4.3f,%4.3f,%4.3f,2)\n' ...
               'circuit_A%u.u3(%4.3f,%4.3f,%4.3f,1)'];
position_index = [1,2,6,10,11,12,14,16,17,19,20,21,25];
x_all = [];
for k = 1:4
    x = [];
    for j = position_index
        x(j) = k;
    end
    [K0,K1,N0] = Cartan(A(:,:,k),su2_l,su2_m,su2_h);
    A01A02 = K1';
    A03A04 = K0*K1;
    [A01,A02] = kron_div(A01A02,2);
    [A03,A04] = kron_div(A03A04,2);
    [theta,phi,lambda] = squ(A01);
    x(3:5) = [theta,phi,lambda];
    [theta,phi,lambda] = squ(A02);
    x(7:9) = [theta,phi,lambda];
    anglesN0 = to_vect(logm(N0),su2_h);
    x(13) = anglesN0(3);
    x(15) = anglesN0(1);
    x(18) = anglesN0(2);
    [theta,phi,lambda] = squ(A03);
    x(22:24) = [theta,phi,lambda];
    [theta,phi,lambda] = squ(A04);
    x(26:28) = [theta,phi,lambda];
    x_all = [x_all,x'];
end
x_all(isnan(x_all)) = 0;
str = strcat(str,sprintf(formatSpecA,x_all));
formatSpecN = ['\n\ncircuit_N%u = QuantumCircuit(3)\n' ...
               'circuit_N%u.cx(2,0)\n' ...
               'circuit_N%u.rz(2*(%4.3f),0)\n' ...
               'circuit_N%u.cx(1,0)\n' ...
               'circuit_N%u.rz(2*(%4.3f),0)\n' ...
```

```
              'circuit_N%u.cx(2,0)\n' ...
              'circuit_N%u.rz(2*(%4.3f),0)\n' ...
              'circuit_N%u.cx(1,0)'];
x_all = [];
N(:,1) = N1;
N(:,2) = N2;
position_index = [1,2,3,5,6,8,9,11];
for k = 1:2
    x = [];
    for j = position_index
        x(j) = k;
    end
    x(4) = N(2,k);
    x(7) = N(1,k);
    x(10) = N(3,k);
    x_all = [x_all,x'];
end
x_all(isnan(x_all)) = 0;
str = strcat(str,sprintf(formatSpecN,x_all));
formatSpecM = ['\n\ncircuit_M = QuantumCircuit(3)\n' ...
              'circuit_M.cx(2,0)\n' ...
              'circuit_M.ry(2*(%4.3f),2)\n' ...
              'circuit_M.cx(1,2)\n' ...
              'circuit_M.ry(-2*(%4.3f),2)\n' ...
              'circuit_M.cx(1,2)\n' ...
              'circuit_M.cx(2,0)\n' ...
              'circuit_M.rz(pi/2,0)\n' ...
              'circuit_M.cx(1,0)\n' ...
              'circuit_M.ry(-2*(%4.3f),0)\n' ...
              'circuit_M.cx(1,0)\n' ...
              'circuit_M.ry(-2*(%4.3f),0)\n' ...
              'circuit_M.rz(-pi/2,0)'];
x_all = [M(1),M(2),M(3),M(4)]';
x_all(isnan(x_all)) = 0;
str = strcat(str,sprintf(formatSpecM,x_all));
formatSpecB = ['\n\ncircuit_B%u = QuantumCircuit(3)\n' ...
              'circuit_B%u.rz(%4.3f,0)'];
x_all = [];
for k = [2,4]
    x = [];
    x(1) = k;
    x(2) = k;
    x(3) = angle(B(2,2,k));
    x_all = [x_all,x'];
end
x_all(isnan(x_all)) = 0;
str = strcat(str,sprintf(formatSpecB,x_all));
str = strcat(str,sprintf(['\n\n' name ' = circuit_A1+circuit_N1+circuit_B2+circuit_A2
+circuit_M+circuit_A3+circuit_N2+circuit_B4+circuit_A4']));
end
for k = 1:size(Unitaries,3)
    str = composeCircuit(Unitaries(:,:,k),['circuit_' num2str(k)],str);
end
circuits_str = '\n\ncircuits = [';
for k=1:size(Unitaries,3)-1
```

```
    circuits_str = [circuits_str,'init+circuit_',num2str(k),'+meas,'];
end
circuits_str = [circuits_str,'init+circuit_',num2str(size(Unitaries,3)),'+meas];'];
str = strcat(str,sprintf(circuits_str));
end
function y = f_vm(K_vect)
global m;
global v;
global basis_l;
global basis_m;
K_gen = 0;
for k = 1:size(basis_l,3)
    K_gen = K_gen + K_vect(k)*basis_l(:,:,k);
end
K1 = expm(K_gen);
m_rot = K1'*m*K1;
y = real(to_vect(v,basis_m)*to_vect(m_rot,basis_m)');
end
global su2;
global su2_m;
global su2_l;
global su2_h;
global su3_m;
global su3_l;
global su3_l0;
global su3_l1;
global su3_h;
global su3_f;
global X;
global Y;
global Z;
global I;
X = [0,1;1,0];
Y = [0,-1i;1i,0];
Z = [1,0;0,-1];
I = eye(2);
su2(:,:,1) = 1i*kron(X,X);
su2(:,:,2) = 1i*kron(Y,Y);
su2(:,:,3) = 1i*kron(Z,Z);
su2(:,:,4) = 1i*kron(X,Y);
su2(:,:,5) = 1i*kron(Y,Z);
su2(:,:,6) = 1i*kron(Z,X);
su2(:,:,7) = 1i*kron(X,Z);
su2(:,:,8) = 1i*kron(Y,X);
su2(:,:,9) = 1i*kron(Z,Y);
su2(:,:,10) = 1i*kron(X,I);
su2(:,:,11) = 1i*kron(Y,I);
su2(:,:,12) = 1i*kron(Z,I);
su2(:,:,13) = 1i*kron(I,X);
su2(:,:,14) = 1i*kron(I,Y);
su2(:,:,15) = 1i*kron(I,Z);
su2_m(:,:,1) = 1i*kron(X,X);
su2_m(:,:,2) = 1i*kron(Y,Y);
su2_m(:,:,3) = 1i*kron(Z,Z);
su2_m(:,:,4) = 1i*kron(X,Y);
```

```
su2_m(:,:,5) = 1i*kron(Y,Z);
su2_m(:,:,6) = 1i*kron(Z,X);
su2_m(:,:,7) = 1i*kron(X,Z);
su2_m(:,:,8) = 1i*kron(Y,X);
su2_m(:,:,9) = 1i*kron(Z,Y);
su2_l(:,:,1) = 1i*kron(X,I);
su2_l(:,:,2) = 1i*kron(Y,I);
su2_l(:,:,3) = 1i*kron(Z,I);
su2_l(:,:,4) = 1i*kron(I,X);
su2_l(:,:,5) = 1i*kron(I,Y);
su2_l(:,:,6) = 1i*kron(I,Z);
su2_h(:,:,1) = 1i*kron(X,X);
su2_h(:,:,2) = 1i*kron(Y,Y);
su2_h(:,:,3) = 1i*kron(Z,Z);
su3_h(:,:,1) = 1i*kron(X,kron(X,X));
su3_h(:,:,2) = 1i*kron(Y,kron(Y,X));
su3_h(:,:,3) = 1i*kron(Z,kron(Z,X));
su3_h(:,:,4) = 1i*kron(I,kron(I,X));
su3_f(:,:,1) = 1i*kron(Z,kron(Z,Z));
su3_f(:,:,2) = 1i*kron(Z,kron(I,Z));
su3_f(:,:,3) = 1i*kron(I,kron(Z,Z));
su3_m(:,:,1) = 1i*kron(I,kron(I,X));
su3_m(:,:,17) = 1i*kron(I,kron(I,Y));
for j = 1:15
    su3_m(:,:,j+1) = kron(su2(:,:,j),X);
    su3_m(:,:,j+17) = kron(su2(:,:,j),Y);
    su3_l0(:,:,j) = kron(su2(:,:,j),I);
    su3_l1(:,:,j) = kron(su2(:,:,j),Z);
    su3_l(:,:,1) = 1i*kron(kron(I,I),Z);
    su3_l(:,:,j+1) = kron(su2(:,:,j),Z);
    su3_l(:,:,j+16) = kron(su2(:,:, j),I);
end
function y = PolP(m)
global g
global g_m
global g_l
global basis_m
k = PolQ(m);
y = logm(expm(g_l+g_m)/(expm(k)*expm(m)));
end
function k_l = PolQ(m)
global g;
global basis_l;
global g_m;
global g_l;
k = logm(expm(g_m+g_l)*expm(-m));
k_l = 0;
for j = 1:size(basis_l,3)
    k_l = k_l+real(trace(k'*basis_l(:,:,j)))/trace(basis_l(:,:,j)'
    *basis_l(:,:,j))*basis_l(:,:,j);
end
end
function [theta,phi,lambda] = squ(A)
alpha = angle(A(1,1));
A = A/exp(1i*alpha);
```

```
theta = 2*acos(A(1,1));
lambda = angle(-A(1,2)/sin(theta/2));
phi = angle(A(2,1)/sin(theta/2));
end
function [A,B] = kron_div(C,n)
m = size(C,1)/n;
B = C(1:n,1:n);
DB = B*B';
modB = sqrt(DB(1,1));
B = B/modB;
B = B/exp(1i*angle(B(1,1)));
eps = 0.0001;
A = zeros(m);
for i = 1:n;
    if abs(B(i,1))>eps
        d = i;
        break
    end
end
for i = 1:m;
    for j = 1:m;
        A(i,j) = C((i-1)*n+d,(j-1)*n+1)/B(d,1);
    end
end
DA = A*A';
modA = sqrt(DA(1,1));
A = A/modA;
end
function u = map(v1)
global basis_m;
global basis_l;
mat = 0;
for k = 1:size(basis_m,3)
    mat = mat + v1(k)*basis_m(:,:,k);
end
n = PolP(mat);
for j = 1:size(basis_m,3)
    u(j) = trace(n*basis_m(:,:,j));
end
u = real(u);
end
function z = brac(x,y)
z = x*y-y*x;
end
function v = to_vect(M,basis)
for k=1:size(basis,3)
    v(k) = trace(M'*basis(:,:,k))/trace(basis(:,:,k)'*basis(:,:,k));
end
end
```