CMSC 643
RESEARCH PROBLEM 1
QUANTUM TOMOGRAPHY FOR 2 QUBITS

PATRICK TRINKLE
tri1@umbc.edu

Problem

Given some fixed state 2-qubit ket $|\psi>$ and an a positive real number ε, estimate the amplitudes a and b such that the estimates are within ε of the amplitudes.

Background

Given some black box M that executes some quantum measurement with two inputs and two outputs the following is known. Given some input observable O and some quantum state ket $|\psi>$, M will output an eigenvalue λ_j of O, and $|\psi_j> = P_j|\psi> / \text{sqrt}(<\psi|P_j|\psi>)$ with probability $<\psi|P_j|\psi>$.

Solution

To complete this problem in Maple two routines are fairly efficient. The first subroutine needs to measure an arbitrary 2-qubit ket $|\psi>$ and an arbitrary observable O. This subroutine, hereafter Meas, returns a random eigenvalue λ_j of O and a corresponding eigenket $|\lambda_j>$. In the event there is only 1 eigenvalue Meas returns the original $|\psi>$. The other routine I need is called QTomography which will call Meas for the various observables required. QTomography will attempt to estimate the amplitudes of the input quantum state represented by the ket $|\psi>$ within a certain ε. Without loss of generality we can assume that a_{00} is real, and b_{00} is 0. For this problem I am also assuming all amplitudes are reals. Therefore we only need to solve a_{00}, a_{01}, b_{01}, a_{10}, b_{10}, a_{11}, b_{11}, subject to the constraint $a_{00}^2 + a_{01}^2 + b_{01}^2 + a_{10}^2 + b_{10}^2 + a_{11}^2 + b_{11}^2 = 1$. Therefore we only need to concern oursevles with 6 degress of freedom, as opposed to the 15 possible. Also, note that measuring against the Identity Matrix, I, will provide no interesting information (it only has 1 eigenvalue).

$\sigma_0 = (1, 1, 1, 1)$	$\sigma_1 = (0, 1, 1, 0)$	$\sigma_2 = (0, -i, i, 0)$	$\sigma_3 = (1, 0, 0, -1)$

Table 1 - Pauli spin Operators

To estimate the amplitudes we measure the state with respect to the tensor combinations of the Pauli spin operators σ_0, σ_1, σ_2, and σ_3, see table 1. The following observables are used: $\sigma_{01}=\sigma_0\otimes\sigma_1$; $\sigma_{02}=\sigma_0\otimes\sigma_2$; $\sigma_{03}=\sigma_0\otimes\sigma_3$; $\sigma_{10}=\sigma_1\otimes\sigma_0$; $\sigma_{20}=\sigma_2\otimes\sigma_0$; $\sigma_{30}=\sigma_3\otimes\sigma_0$. Each observable has two possible eigenvalues, +1 and -1. Therefore we can focus on only two eigenspaces, V^+ and V^-, and the corresponding projection operators.

+1	-1
$P_{+1} = (1/2)\ (1, 1, 1, 1)$	$P_{-1} = (1/2)(1, -1, -1, 1)$
$P_{+2} = (1/2)(1, -i, i, 1)$	$P_{-2} = (1/2)(1, i, -i, 1)$
$P_{+3} = (1, 0, 0, 0)$	$P_{-3} = (0, 0, 0, 1)$

Table 2 - 2x2 Projectors

The corresponding projection operators are labeled as follows: P_{+1}, P_{-1}, P_{+2}, P_{-2}, P_{+3},

P_{-3}, see table 2. These correspond to the following equations: $\sigma_{01}=\sigma_0\otimes P_{+1} - \sigma_0\otimes P_{-1}$. Since we are focusing on the how often we receive the eigenvalue +1, we will use the projection operators: P_{+10}, P_{+01}, P_{+20}, P_{+02}, P_{+30}, P_{+03}, where $P_{+10}=P_{+1}\otimes\sigma_0$, $P_{+01}=\sigma_0\otimes P_{+1}$, $P_{+20}=P_{+2}\otimes\sigma_0$, $P_{+02}=\sigma_0\otimes P_{+2}$, $P_{+30}=P_{+3}\otimes\sigma_0$, $P_{+03}=\sigma_0\otimes P_{+3}$. The probability for each measurement to return the +1 eigenvalue is represented in the following equations (each indexed to match the Observable used): $p_{10}=<\psi|P_{+10}|\psi>$, $p_{01}=<\psi|P_{+01}|\psi>$,$p_{20}=<\psi|P_{+20}|\psi>$, $p_{02}=<\psi|P_{+02}|\psi>$, $p_{30}=<\psi|P_{+30}|\psi>$, $p_{03}=<\psi|P_{+03}|\psi>$. To determine these values experimentally, QTomography executes the measurement for each Observable 100-1000 times depending on desired preciseness. Each measurement increments a counter, and each returned +1 increments another counter. This provides with the probability of returning a specific eigenket state.

Given the list of unknowns, and the list of equations, we task Maple with determining the estimates. Then we compare the results with the actual input, to determine how precise our estimates were. If they were not precise enough the experiments can repeat until we are within ε of the amplitudes.

Conclusions

After going through the process of measuring the quantum states with varying observables I gained a better understanding of the entire measurement process, as well as a better understanding of quantum tomography. While working on the project I had serious problems using the maple on the UMBC computers; it was always running out of memory (even if it was doing very trivial things). Also, in the QTomography subroutine there is a bug that I could not fix related to building the projection operators. Also, I had a hard time with the very last step; comparing the estimates with the input to determine correctness. Since the solver didn't work--I determined it wasn't going to give me a good solution, but not in time to try an algebraic estimate.

Towards the end of the project everything started to make sense in the context of the diagram you presented the class in the beginning, which has the quantum black box and the system represented in kets and in density operators. During the project, I kept referring to the diagram to better understand what equations and the process of determining what the input was (with probability) based on the outputs. This all comes together when we know the probability and we know the projection operator and the returned eigenket from the measurement process.

Appendix A - Meas() subroutine in Maple Code

```
Meas:=proc(psi, Ob)

local a, b, x, eigenspaceCnt, ListEigenVects, i, j, vCount, vIdx, VectorSet, vnext, ProjA,
ProjB, ProbA, ProbB, StateA, StateB, output, normal;

# Given some arbitrary 2-qubit ket psi and an arbitrary observable Ob, output
# a random eigenvalue of Observable and a corresponding eigenket.

# obtain eigenvalue information
ListEigenVects := [eigenvects(evalm(Ob))];
eigenspaceCnt := nops(ListEigenVects);

# Separate into Eigenspaces
for i from 1 to eigenspaceCnt do
  eigenspace||i := op(i, ListEigenVects);
```

```
od;

# Extract the Eigenvalues
for i from 1 to eigenspaceCnt do
  eigen||i := op(1, eigenspace||i);
od;

# Extract the Eigenvector counts
for i from 1 to eigenspaceCnt do
  count||i := op(2, eigenspace||i);
od;

# Setup Counters
vCount := 0;
vIdx := 1;

# Extract the Eigenvectors
for i from 1 to eigenspaceCnt do
  for j from 1 to count||i do
    v||vIdx := op(j, op(3, eigenspace||i));
    vCount := eval(vCount + 1);
    vIdx := eval(vIdx + 1);
  od;
od;

# make them orthogonal if necessary (max set of 2)
vIdx := 1;
for i from 1 to eigenspaceCnt do
  if (count||i > 1) then
    vnext := eval(vIdx + 1);
    VectorSet := GramSchmidt({v||vIdx,v||vnext});
    ww||vIdx := op(1, VectorSet);
    ww||vnext := op(2, VectorSet);
    vIdx := eval(vIdx + 2);
  else
    ww||vIdx := v||vIdx;
    vIdx := eval(vIdx + 1);
  fi;
od;

# normalize the Eigenkets
for i from 1 to vCount do
  w||i:=convert(evalm( (1/norm(ww||i, 2)) * ww||i ), matrix):
od;

# compute the alpha values so we can compute the resulting state for each eigenvalue
for i from 1 to vCount do
  alpha||i := evalm(htranspose(w||i) &* psi)[1,1];
od;

# compute the Projection Operators piece-wise
for i from 1 to vCount do
  Proj||i:=evalm( w||i &* htranspose(evalm(w||i)));
od;
```

```
# compute the Projection Operators for the two Eigenspaces
# I do realize this isn't generic; below is a description of how to attempt this generically.
# I broke it into the special cases, 4 and 2--basically.
if (vCount = 4) then
  ProjA := evalm(Proj1 + Proj2);
  ProjB := evalm(Proj3 + Proj4);
else
  ProjA := evalm(Proj1);
  ProjB := evalm(Proj2);
fi:

# compute the Probabilities
if (vCount = 4) then
  ProbA := abs(alpha1)^2 + abs(alpha2)^2;
  ProbB := abs(alpha3)^2 + abs(alpha4)^2;
else
  ProbA := abs(alpha1)^2;
  ProbB := abs(alpha2)^2;
fi:

normal := 1;

# Was there only 1 eigenvalue?
if (eigenspaceCnt = 1) then
  normal := 0;
  output := [eigen1, psi];
fi:

# compute the Resulting States
# If there can only be 1 resulting state, map it such that it will return the correct one
regardless of rand()
if (vCount = 4) then
  if (ProbA <> 1 and ProbB <> 1) then # Kind of the normal 4 vector case
    StateA := map(simplify, evalm( (1/sqrt(abs(alpha1)^2 + abs(alpha2)^2)) *
evalm(alpha1*w1 + alpha2*w2) ));
    StateB := map(simplify, evalm( (1/sqrt(abs(alpha3)^2 + abs(alpha4)^2)) *
evalm(alpha3*w3 + alpha4*w4) ));
  else
    normal := 0;
    if (ProbA = 1) then
      StateA := map(simplify, evalm( (1/sqrt(abs(alpha1)^2 + abs(alpha2)^2)) *
evalm(alpha1*w1 + alpha2*w2) ));
      StateB := StateA;
      output := [eigen1, evalm(StateA)];
    else
      StateB := map(simplify, evalm( (1/sqrt(abs(alpha3)^2 + abs(alpha4)^2)) *
evalm(alpha3*w3 + alpha4*w4) ));
      StateA := StateB;
      output := [eigen2, evalm(StateB)];
    fi:
  fi:
else # (vCount = 2) (this won't effect output if eigenspaceCnt is 1)
  StateA := evalm(w1);
```

```
   StateB := evalm(w2);
fi:

# HOW TO BUILD PROJECTION OPERATORS FOR EIGENSPACES GENERICALLY:
# So, somehow given some eigenvalue, I need to know which eigenkets are associated with
it.
# Well, if I know the order of the eigenvalues, and the counts of each group, then i know.
# Especially if i know each eigenspace and the order the eigenspaces, because everything
just walks in order.
# Therefore, if we have 3 eigenspaces, with the following count||i's, 1, 2, 3-->this tells us
that
# eigen1, w1, and alpha1 are related,
# eigen2, w2, w3, and alpha2 and alpha3 are related,
# eigen3, w4, w5, w6, and alpha4, alpha5 and alpha6 are related.
# Would it be worthwhile to determine this pairing and group them in some deterministic
way?
# Or should I try to just group them on the fly?  : Note, if I group them on the fly I can
pretty much
# just randomly select a group index and return that group.

if (normal = 1) then
   #Determine Randomly Which Eigenvalue and Resulting State to Return
   a:=rand(0..1);
   b:= a();

   # if it's less than 0.5 (0) then ProbPlus, otherwise ProbMinus
   if (b < 0.5) then
      output := [eigen1, evalm(StateA)];
   else
      output := [eigen2, evalm(StateB)];
   fi:
fi:

RETURN(output);

end:
```

Appendix B - QTomography routine in Maple Code

Code could not paste properly; see attached maple worksheet file.

Appendix C - Required Header Code in Maple Code

```
with(linalg):  with(group):
kr:=proc(A,B)
   # This procedure computes the Kronecker product of two matrices
   local mm,nn,ans,ans1;
   ans:=[];
   for mm from 1 to rowdim(A) do
      for nn from 1 to coldim(A) do
         ans:=[op(ans), scalarmul(B, A[mm,nn] )];
      od;
   od;
   ans1:=blockmatrix(rowdim(A),coldim(A), ans );
```

```
  RETURN(evalm(ans1));
end:
sigma||0:=diag(1,1):
sigma||1:=matrix(2,2,[0,1,1,0]):
sigma||2:=matrix(2,2,[0,-I,I,0]):
sigma||3:=matrix(2,2,[1,0,0,-1]):
ket||0:=matrix(2,1,[1,0]):
ket||1:=matrix(2,1,[0,1]):

#
for i from 0 to 1 do
  for j from 0 to 1 do
    ket||i||j:=kr(ket||i,ket||j);
  od;
od;

#
for i from 0 to 1 do
  for j from 0 to 1 do
    for k from 0 to 1 do
      ket||i||j||k:=kr(ket||i||j, ket||k);
    od;
  od;
od;

#
for i from 0 to 1 do
  for j from 0 to 1 do
    for k from 0 to 1 do
      for m from 0 to 1 do
        ket||i||j||k||m:=kr(ket||i||j||k, ket||m);
      od;
    od;
  od;
od;
```