APPENDIX

MARIO KUMMER AND BÜŞRA SERT

1. Source Code for Computations in Macaulay2 and SageMath

In this section we provide codes we used during the tests and computations we proccessed for the classification of matroids on at most 8 elements.

1.1. SoS Tests for HPP of Matroids in Macaulay2.

Operations we do require `Macaulay2` packages "SumsOfSquares" and "Matroids". They can be called by using the following commands:

```
needsPackage("SumsOfSquares");
needsPackage("Matroids");
```

We start with obtaining the list of all non-isomorphic matroids on 8 elements, then continue with taking those with rank 3 or 4 that are simple and connected.

```
L8=allMatroids 8;
--List of all non-isomorphic matroids with 8 elements

-- #L8=1724
L={};--List of matroids on 8 elements with rank 3 or 4
for M in L8 do (
    if rank M==3 or rank M==4
       then L=append(L,M);
       );
--#L=1265
S={}; --List of simple matroids
for M in L do (
    if isSimple M
       then S=append(S,M);
       );
--#S=685
C={}; --List of simple and connected matroids
for M in S do (
    if #components M==1
       then C=append(C,M);
       );
--#C=659
```

We define the forbidden minors for the half-plane property for matroids on seven elements as folllows:

```
F=specificMatroid "fano"; -- $F$
NF=specificMatroid "nonfano"; -- $F^{-}$
F2=relaxation(NF,set{0,1,6}); --$F^{--}$
K4e=relaxation(F2,set{2,1,5});--$M(K4)+e$
```

2010 *Mathematics Subject Classification.* Primary: 05B35; Secondary: 90C22.

```
preF3=relaxation(NF,set{0,2,4});
F3=relaxation(preF3,set{0,3,5});--$F^{-3}$

DF=dual F; --dual of $F$
DNF=dual NF; --dual of $F^{-}$
DF2=dual F2; --dual of $F^{--}$
DK4e=dual K4e; --dual of $M(K4)+e$
DF3=dual F3; --dual of $F^{-3}$
```

We exclude the forbidden minors defined above as follows:

```
H={};
--List of matroids without the forbidden minors of rank 3

for M in C do (
    if not hasMinor(M,F) and not hasMinor(M,NF)
    and not hasMinor(M,F2) and not hasMinor(M,F3)
    and not hasMinor(M,K4e)
       then H=append(H, M);
       );

Hpcand={};
-- List of matroids that don't have any of the forbidden minors
--We recommend to save this list in a file

for M in H do (
    if rank M==4 and not hasMinor(M,DF) and not hasMinor(M,DNF)
    and  not hasMinor(M,DF2) and not hasMinor(M,DF3)
    and not hasMinor(M,DK4e)
       then Hpcand=append(Hpcand, M);
    if rank M==3
       then Hpcand=append(Hpcand, M);
       );
--#Hpcand=309
```

Now, we have the list of matroids that are candidates to have the half-plane property. In order to make a sums of squares test on the Rayleigh difference of the basis generating polynomials of those matroids, we define the following functions:

```
R=QQ[x_0 .. x_7]; --Ring of the basis generating polynomials

BgP=method();
--Function for finding the basis generating polynomial
BgP(Matroid,Ring):= RingElement => (M,R) ->(
    h:=0_R; B:=bases(M);
    for i from 0 to (#B-1) do (
       L:=toList(B_i); N:=1;
       for j from 0 to (#L-1) do (
            N=N*x_(L_j); --monomials
            );
     h=h+N
        );
     return h
     );
 --Creating list of indices for Rayleigh difference
```

```
S1=toList(0..6);
S2=toList(0..7);
A=set(S1)**set(S2);
I=toList A;
J={};--the list of distinct pairs of indices i,j
for i from 0 to (#I-1) do (
    a:=I_i;
    if a_0!=a_1
      then J=append(J,a)
      );

RDSos=method();
-- Function for SoS test on Rayleigh differences for a matroid
RDSos(Matroid,List,Ring):= Boolean =>(M,J,R)-> (
     h:=BgP(M,R); --Basis generating polynomial
     SS={};
     for a in J do(
         i:=a_0; j:=a_1;
         rayl=diff(x_i,h)*diff(x_j,h)-h*diff(x_i,diff(x_j,h));
         --Rayleigh difference
         sol=solveSOS rayl; --Sos test
         if status(sol)=="SDP solved, primal-dual feasible"
            then SS=append(SS,{i,j});
            );
      if #SS==0
         then return false
      else
         return true
       );
```

A matroid M has the half-plane property if and only if there are some indices i, j for which the Rayleigh difference $\Delta_{ij} h_M$ of the basis generating polynomial h_M is a sum of squares. Thus, at the beginning we only test whether for each matroid in our list there is a pair i, j for which Δ_{ij} is a sums of squares. Please note that the test takes time and a lot of memory. We recommend to split the set of matroids into $2-3$ parts and run the code on those parts.

```
unkwn={};--List of matroids that don't pass SoS test
for i from 0 to #Hpcand-1 do (
    print i; --to keep track of the process
    M=Hpcand_i;--Matroid
    if not RDSos(M,J,R)
       then unkwn=append(unkwn,M)
       );
--It takes time and memory.
--We recommend to run this code on several parts of Hpcand
--instead of all in once
--For example, do it for the first 200 then for the rest
--#unkwn=22
```

After the test, we obtain that there are 22 matroids for which more test are required in order to find out whether they have the half-plane property. Moreover the test shows that other 287 matroids have the half-plane property. In order to certify that

the result of the SOS test implies that they have the half-plane property, we record the ring of the Gram matrix as follows:

```
--List of indices of 22 matroids from unkwn in Hpcand

NotHpp={73,76,77,81,82,83,85,88,92,93,94,95,96,97
,99,100,101,112,113,114,116,117}

Hpp={};
--List of indices of 287 Matroids from Hpcand that have HPP

for i from 0 to #Hpcand-1 do (
    if not member(i,NotHpp)
       then Hpp=append(Hpp,i);
       );

RDGram=method();
--Function that returns the list of rings of Gram matrices
RDGram(Matroid,List,Ring):= List =>(M,J,R) -> (
    h:=BgP(M,R); --Basis generating polynomial
    SS:={};
    for a in J do(
        i:=a_0; j:=a_1;
        rayl=diff(x_i,h)*diff(x_j,h)-h*diff(x_i,diff(x_j,h));
        sol=solveSOS rayl; --Sos test
        if status(sol)=="SDP solved, primal-dual feasible"
           then (G=sol#GramMatrix; SS=append(SS, ring G))
           --ring G is the  ring of the Gram matrix
        else
           continue
             );
    return SS
    );

--Recording the data in a file
f="/home/.../.../Gram"; --Fill in the file location
for i from 0 to #Hpp-1  do (
    print i; j=Hpp_i;
    M=Hpcand_j;
    K=RDGram(M);
    f<< toString({i, K})<< ","<< flush --Writing on the file
    );
f<<close;--Closing the file

--It takes time and memory.
--We recommend to run this code on several parts of Hpp
--instead of all in once

--List of indices of matroids in Hpcand for which
--some Gram matrices have floating point entries
--{42,44,46,48,49,50,52,53,54,59,60,61,70,72,79,120,141}
```

Let $\mathcal{M}$ be the list of all matroids on 8 elements obtained in Macaulay2 using the command **allMatroids 8**). For $\mathcal{M}_k$ where k is in

$$\{393, 395, 397, 399, 400, 401, 403, 404, 405, 410, 411, 412, 421, 423, 433, 664, 717\}$$

the SDP solver finds Gram matrices having floating points as entries (i.e, they are approximate) during the SOS test on $\Delta_{i,j}$ for some indices i, j, but not all.

1.2. **Tests on Hyperbolicity.** A matroid M on n elements has the half-plane property if and only if h_M is hyperbolic with respect to every point $e \in \mathbb{R}^n_{>0}$, i.e., for all $v \in \mathbb{R}^n$, $h_M(et - v) \in \mathbb{R}[t]$ has only real roots. In this section, we consider $e, v \in \{0,1\}^8$ and check whether the basis generating polynomial h_M where M is from the list of the 22 matroids has the property that $h_M(et - v) \in \mathbb{R}[t]$ is real rooted.

```
--Creating the list of vectors with 0-1 entries
K=toList(0..2^8-1);
D={};--List of all directions with 0-1 entries
--Its elements are {0,0,1,0,1,1,1,0} etc.
for i from 0 to 2^8-1 do (
    a=();
    for j from 0 to 7 do(
         if K_i & (2^j) == 0
             then a=append(a,0)
         else a=append(a,1)
         );
     D=append(D,toList(a));
     );
U=CC[t];--Ring of univariate polynomial $h_M(et-v)$
--(over complex numbers as we will check complex solutions)
univP=method();
--Function to create the univariate polynomial $h_{M}(et-v)$
univP(Matroid,List,List,Ring):= RingElement => (M,e,v,U) -> (
    B:=bases(M); p:=0_U;
    for i from 0 to #B-1 do(
        s:=toList(B_i); c:=1;
        for j from 0 to #s-1 do (
            k:=s_j; m=(e_k*t-v_k); c=c*m;
            );
         p=p+c;
         );
    return p
    );
```

Having defined the list of directions we consider and the function to obtain the univariate polynomial, we can now run the test for each matroid and $e, v \in \{0,1\}^8$. We recommend to save the indices and corresponding polynomials with non-real solutions in a separate file, and call them afterwards.

```
--For each matroid, creating a file with polynomials
--with non-real roots

for i from 0 to #unkwn-1 do(
    S={}; --List of tuples ((e,v),univP(M,e,v,U))
    M=unkwn_i;--Matroid from the list of 22 matroids
    for j from 0 to 2^8-1 do(
```

```
        for k from 0 to 2^8-1 do(
           h=univP(M,D_k,D_j,U); --Univariate polynomial
           if h!=0
              then (d=degree h;
              if #d!=0
                 then (F={h}; sys:=solveSystem F;
                 n:=#sys-#(realPoints sys);
                 --Checks the number of non-real roots
                 if n!=0
                    then S=append(S,((D_k,D_j),h))););
               );
            );
      y="/home/.../.../NonRealM"|i;
      --Creates a file (fill in the file location)
      y<< toString(S)<< close; --Writes on the file
      );
```

After the test, we obtain that among the 22 matroids, for 15 of them there are directions $e, v \in \{0,1\}^8$ for which the univariate polynomial $h_M(et - v) \in \mathbb{R}[t]$ has non-real roots. Since roots of a univariate polynomial continuously depend on the coefficients, having such directions at the limit (boundary of the positive orthant) proves that they don't have the half-plane property (see Table 1 for the list of a sample of directions).

Let $\mathcal{M}$ be the list of all non-isomorphic matroids on 8 elements (in the code, it refers to the list $L8$). The 15 matorids we mentioned are $\mathcal{M}_i$ for i in

$$\{435, 437, 439, 443, 450, 455, 460, 461, 465, 466, 467, 548, 549, 570, 575\}.$$

For the remaining 7 matroids we apply the hyperbolicity test again with random $e \in \mathbb{R}^8_{\geq 0}$ and $v \in \mathbb{R}^8$.

```
PD={};--List of random positive directions
for i from 0 to 2^8-1 do (
    b=();
    for j from 0 to 7 do (
        b=append(b,random(0.,100.))
         );
    PD=append(PD,toList(b));
    );

ND={};--List of random directions
for i from 0 to 2^8-1 do (
    b=();
    for j from 0 to 7 do (
        b=append(b,random(-100.,100.))
         );
    ND=append(ND,toList(b));
    );
T={0,1,2,12,13,19}--List of indices of 7 matroids in unkwn
for i in T do(
    S={}; --List of tuples ((e,v),univP(M,e,v,U))
    M=unkwn_i;
    for j from 0 to 2^8-1 do(
        for k from 0 to 2^8-1 do(
           h=univP(M,PD_k,ND_j,U);
```

```
            --Univariate polynomial with random directions
            if h!=0
               then (d=degree h;
               if #d!=0
                  then (F={h}; sys:=solveSystem F;
                  n:=#sys-#(realPoints sys);
                  --Checks the number of non-real roots
                  if n!=0
                     then S=append(S,((PD_k,ND_j),h))););
                );
             );
      print S;--One can also print it in a file as above
      );
```

Using this test, we could not find any e, v for which $h_M(et - v)$ has non-real roots for M among the 7 matroids. On the other hand, for each of the matorids M, we used " Homotopy Continuation Julia" in order to obtain critical points, and found some points $x \in \mathbb{R}^6$ for which $\Delta_{6,7}h_M(x) < 0$. This proves that they don't have the half-plane property, since h_M has the half-plane property if and only if $\Delta_{i,j}h_M(X) \geq 0$ for all i, j (see Table 2 for the list of points).

1.3. **Tests for Determinantal Representability.** In Section 1.1, we obtained that 287 matroids among the 309 matrodis have the half-plane property since they passed the SOS test we applied. At that point we were only interested in knowing whether there are some indices i, j for which $\Delta_{i,j}$ is a sums of square. However, we didn't record if they are SOS-Rayleigh. Such an information provides one of the ways to test determinantal representability. In this section, we test determinantal representability of 287 matroids that have the half-plane property.

Since for a matroid with the half-plane property, having a determinantal representation implies that it is SOS-Rayleigh, matroids for which $\Delta_{i,j}$ is a sums of squares for some i, j but all does not have a determinantal representation. Please note that being SOS-Rayleigh doesn't imply determinantal representability. There are some matroids having that property and not having a determinantal representation. We define the following function and test this property.

```
R=QQ[x_0 .. x_7]; --Ring of the basis generating polynomials

 --Creating list of indices for Rayleigh difference
S1=toList(0..6);
S2=toList(0..7);
A=set(S1)**set(S2);
I=toList A;
J={};--the list of distinct pairs of indices i,j
for i from 0 to (#I-1) do (
    a:=I_i;
    if a_0!=a_1
      then J=append(J,a)
      );

SosDet=method();
--Function for testing determinantal representability
SosDet(Matroid,List, Ring):= Boolean =>(M,J,R) -> (
     h:=BgP(M,R);--Basis generating polynomial
     S:={};
```

```
      for a in J do(
          i:=a_0; j:=a_1;
          rayl=diff(x_i,h)*diff(x_j,h)-h*diff(x_i,diff(x_j,h));
          --Rayleigh difference
          sol=solveSOS rayl; --Sos test
          if status(sol)=="SDP solved, primal-dual feasible"
             then S=append(S,{i,j});
              );
       if #S!=#J and #S>0
          then return false
       --Implies that M doesn't have
       --a determinantal representation
       else
          return true
       --Doesn't imply anything in terms of
       --determinantal representation
          );

--Hpp is the list of indices of matroids with Hpp
--(defined above)
NotDet={};
--List of indices of matroids that don't past SosDet test
--Test is applied on matroids with HPP
for j from 0 to #Hpp-1 do (
    print j;--to keep track of the process
    i=Hpp_j;--Corresponding index in Hpcand
    M=Hpcand_i;
    if not SosDet(M,J,R)
       then  NotDet=append(NotDet,i);
       );
--It takes time. We recommend to split indices into parts
--and to run the test on the parts
--NotDet={58, 62, 63, 64, 66, 67, 68, 84, 86, 90, 106,
--108, 109,110}
--List of indices of matroids in Hpcand that don't have
--determinantal representation
--Hpcand_110 is the Vamos Matroid
```

After the test, we obtain that 14 of the 287 matroids (including the Vamos matroid) does not pass the test so that they don't have a determinantal representation. Let $\mathcal{M}$ be the list of all non-isomorphic matroids on 8 elements (in the code, it refers to the list $L8$). The matroids $\mathcal{M}_k$ for k in

$$\{409, 413, 414, 415, 417, 418, 419, 438, 440, 445, 498, 500, 501, 502\}$$

fail the test.

1.4. **Producing Non-SOS Certificates.** As the result of the tests we did on determinantal representability, we have found that there are some matroids that are not SOS-Rayleigh. In this section we give a way to produce symbolic proof that for such matroids, there are some indices i, j for which the Rayleigh difference is not a sum of squares. Let M be a matroid and fix indices i, j. The Rayleigh difference $\Delta_{i,j}$ is not a sum of squares, if and only if we cannot find a PSD Gram matrix G such that $\Delta_{i,j} = XGX^t$ where X is a vector consisting of multi-affine monomials

of degree $d-1$. In order to show that such a PSD matrix G doesn't exist, we need to show that the Gram spectrahedron S_G given by the matrix inequality

$$G_0 + G_1\lambda_1 + \cdots + G_m\lambda_m \succcurlyeq 0$$

where $\sum G_i\lambda_i = G$ is empty. One can show that if there exists a PD matrix M such that $\mathrm{tr}(MG_i) = 0$ for all $i = 0, \ldots, m$, then S_G is zero. We provide a code to find such a matrix M.

The list of indices of matroids we want to produce the certificates for is

$$\{409, 413, 414, 415, 417, 418, 419, 438, 440, 445, 498, 500, 501, 502\}.$$

We write the following function in order to find the list of indices (i, j) for which $\Delta_{i,j}$ is a sum of squares such that we can exclude that from the list of indices and obtain the indices we want.

```
notSosRay={409, 413, 414, 415, 417, 418, 419, 438, 440, 445, 498,
500, 501,502};--502 is Vamos

RDSosIndx=method();
-- Function for SoS test on Rayleigh differences and return
--the indices that make sos
RDSosIndx(Matroid,List,Ring):= List =>(M,J,R)-> (
     h:=BgP(M,R); --Basis generating polynomial
     SS={}; --List of indices
     for p in J do(
         i:=p_0; j:=p_1;
         rayl=diff(x_i,h)*diff(x_j,h)-h*diff(x_i,diff(x_j,h));
         --Rayleigh difference
         sol=solveSOS rayl; --Sos test
         if status(sol)=="SDP solved, primal-dual feasible"
            then SS=append(SS,(i,j));
            );
      if #SS>0
         then return SS
       );

BgP=method();
--Function for finding the basis generating polynomial
BgP(Matroid,Ring):= RingElement => (M,R) ->(
    h:=0_R; B:=bases(M);
    for i from 0 to (#B-1) do (
       L:=toList(B_i); N=1;
       for j from 0 to (#L-1) do (
           N=N*x_(L_j); --monomials
           );
     h=h+N
        );
     return h
     );

L8=allMatroids 8;
M=L8_417;--A matroid from the list

F1=QQ[x_0..x_7];
--Polynomial ring of the basis generating polynomial
```

```
h=BgP(M,F1);--The basis generating polynomial

S1=toList(0..6);
S2=toList(0..7);
C=set(S1)**set(S2);
I=toList C;
J={};--the list of distinct pairs of indices i,j
for i from 0 to (#I-1) do (
    p:=I_i;
    if p_0!=p_1
      then J=append(J,p)
      );

N=RDSosIndx(M,J,F1);
--List of indices that make the Rayleigh difference Sos
K={};
--List of the indices that fail to make Rayleigh difference Sos
for i in J do (
    if not member(i,N)
       then K=append(K,i);
);

i=K_0_0; j=K_0_1;--Setting indices i,j
rayl=diff(x_i,h)*diff(x_j,h)-h*diff(x_i,diff(x_j,h));
--The Rayleigh difference given by the first index set from K
```

As we have the Rayleigh difference we want to produce the certificate for, we can proceed to construct the Gram matrix with variable entries as follows.

```
n=binomial(6,3);--The number of monomials in the matrix X
k=n*(n+1)//2;--The number of distinct entries of G
RS=QQ[a_0..a_(k-1)][x_0..x_7];--Creating the ring
R=coefficientRing RS;
A=genericSymmetricMatrix(R,a_0,n);
G=map(RS^n,RS^n,(i,j)-> (A_(i,j))_RS);
--For changing the ring of A
--This is the initial Gram Matrix with entries ai

y=toList(0..7)-set{i,j};
--List of indices of variables in the Rayleigh difference

substs=subsets(y,3);--Subsets of indices that can appear

E={};--List of monomials to insert in the matrix X
for i from 0 to #substs-1 do (
    c:=1;  s:=substs_i;
    for j from 0 to #s-1 do(
        t:=s_j;  m=x_t;
        c=c*m; --Monomial
        );
    E=append(E,c)
    );
X=matrix{E};
```

```
--matrix M with entries that are prod. of x_i for each subset
XT=transpose X;
Q=X*G*XT;--Matrix with one polynomial entry
```

We want $\Delta - Q_{00} = 0$ so that G is the Gram matrix of $\Delta_{i,j}$. This equality gives us relations between the variable entries of G. Then, we save the variables that don't vanish which will appear as λ_is in the defining pencil of the Gram spectrahedron. Then, we construct the matrices G_i of the pencil.

```
f=substitute(rayl,RS)-Q_(0,0);--Polynomial rayl-Q_00
Cf=coefficients f;
I=ideal(Cf);--Ideal of the relation of the entries
G=G%I;--Inserting relations in G
varS={};
for i from 0 to n-1 do (
    for j from 0 to n-1 do (
        gij=G_(i,j);--Entry Gij
        T=terms gij;
        for k in T do (
            Lmk=leadMonomial substitute(k,R);
            --Lead monomial of the term
            if member(Lmk,gens R)
                then
                  varS=append(varS,Lmk);
            );
        );
    );

varS=toList(set(varS));--List of used variables (lambdas)

GiS={};--List of matrices Gi

--Creating matrices Gi
IR=ideal(gens R);
G0=substitute(G,R)%IR;--Matrix G_0
GiS=append(GiS,G0);

for i from 0 to #varS-1 do (
    C:=G-G0;
    m:=varS_i;
    C=substitute(C,{m=>1});
    --Substitute 1 for the variable
    C=substitute(C,R)%IR;
    --Substitute 0 for the others
    GiS=append(GiS,C);
    );
```

We want to find a PD matrix M such that $\mathrm{tr}(MG_i) = 0$ for all i. Thus, we need to obtain relations coming from the trace of products MG_i, and create new list of matrices that we will use to search for a matrix with rational entries satisfying the trace condition.

```
Traces={};--List of traces of MGi

for i from 0 to #GiS-1 do (
```

```
    Gg=GiS_i; Z=A*Gg;
    if not member(trace Z, Traces)
      then
       Traces=append(Traces,trace Z);
    );

Matrel={};--Matrices for the trace relation

for i in Traces do(
    Mt=A;--Symmetric matrix with entries ai
    Ti=terms substitute(i,R);
    for j in Ti do(
        Lm=leadMonomial j;
        Lc=leadCoefficient j;
        Mt=substitute(Mt,{Lm=>Lc});
        --Inserting the coefficients from the trace
        );
    Mt=Mt%IR;--Setting other entries zero
    Matrel=append(Matrel,Mt);
    );
```

We use semi-definite programming in order to find a PSD matrix M for which the trace of $MG_i = 0$. Then we round it to a matrix with rational entries.

```
r=#GiS;
C=map(QQ^n,QQ^n,(i,j)->0);--We don't want to minimize something
b=map(QQ^r,QQ^1,(i,j)->0);--We want the trace of M*Gi=0

P=sdp(C,toSequence(GiS),b);--SDP
(X,y,Z,stat) = optimize P;
--X is a point from the solution set


Mvecs={};--List of vectorized matrices

for i from 0 to #Matrel-1 do(
    m=substitute(Matrel_i,QQ); Sv=smat2vec m;
    Mvecs=append(Mvecs,Sv);
    );


TMvecs=transpose matrix{Mvecs};
u=#Mvecs;
bb=map(QQ^u,QQ^1,(i,j)->0);

for i in GiS do (print trace(Mat*i))
(ispsd,Mat) = roundPSDmatrix(X,TMvecs,bb,10^5);
--Rounds the SDP solution, satisfying the trace condition
--ispsd
--Mat--Certification matrix

--for i in GiS do (print trace(Mat*i))
--Checking that all traces are zero
```

```
--ispsd
--Mat
```

1.5. **Non-Rayleigh Matroids.** A matroid M is called *Rayleigh* if for all indices i, j, the Rayleigh difference $\Delta_{i,j}h_M(x)$ is non-negative for all $x \in \mathbb{R}^n_{>0}$. Moreover, for fixed indices i, j, h_M can be written of the form

$$h_M = ax_ix_j + bx_i + cx_j + d$$

where a, b, c, d are polynomials that don't depend on x_i or x_j. Then, $\Delta_{i,j}h_M = bc - ad$. By using "HomotopyContinuation.jl", for some of the matroids that have a forbidden minor for the half-plane property, we could find a positive point p such that, $\Delta_{i,j}h_M(p) < 0$. Rayleigh matroids we list have correlation value smaller than $\frac{8}{7}$ for the points we have found. For full list of the matroids, points and the correlation values, please see Table 3.

```
M=L8_891;--Matroid from the list

R=QQ[x_0..x_7]
h=BgP(M,R);--Basis generating Polynomial

S1=toList(0..6);
S2=toList(0..7);
C=set(S1)**set(S2);
I=toList C;
J={};--the list of distinct pairs of indices i,j
for i from 0 to (#I-1) do (
    p:=I_i;
    if p_0!=p_1
      then J=append(J,p)
      );

K=J_27;--indices (i,j)
--For different L, insert the Jindx value given with L

I=K; k=I_0; t=I_1;
rayl=diff(x_k,h)*diff(x_t,h)-h*diff(x_k,diff(x_t,h));
--Rayleigh difference

Y=toList(set(toList(0..7))-set({k,t}));
--List of indices i of xi that appear in Rayl
e0=Y_0; e1=Y_1; e2=Y_2; e3=Y_3; e4=Y_4; e5=Y_5;

L={1,19/100,141/100,141/100,19/100,19/100}--L8_891-Jindx27
--Point that gives the negatie value
--See below for list of L's

mini1=sub(rayl,{x_(e0)=>L_0, x_(e1)=>L_1, x_(e2)=>L_2,
 x_(e3)=>L_3, x_(e4)=>L_4, x_(e5)=>L_5});
print mini1;--prints the negative value


p={};--List of terms of h without xi and xj
q={};--List of terms of h with xi and xj
r={};--List of terms of h with xi without xj
```

```
s={};--List of terms of h With xj without xi

Trms=terms h;
Ex=exponents h;
for j from 0 to #Ex-1 do (
   I=toList(Ex_j);
   if I_k==0 and I_t==0
      --Without xi and xj
      then
       p=append(p,Trms_j)
    );
for j from 0 to #Ex-1 do (
    I=Ex_j;
    if I_k==1 and I_t==1
       --With xi and xj
       then
        q=append(q,Trms_j)
    );
for j from 0 to #Ex-1 do (
    I=Ex_j;
    if I_k==1 and I_t==0
       --With xi without xj
       then
        r=append(r,Trms_j)
    );
for j from 0 to #Ex-1 do (
    I=Ex_j;
    if I_k==0 and I_t==1
       --With xj without xi
       then
        s=append(s,Trms_j)
    );

polD=sum p;--polynomial d
polA=sum q;--polynomial a*xi*xj
polC=sum s;--polynomial c*xj
polB=sum r;--polynomial b*xi

--Substituting point L in polynomials a,b,c,d
minid=sub(polD,{x_(e0)=>L_0, x_(e1)=>L_1, x_(e2)=>L_2,
 x_(e3)=>L_3, x_(e4)=>L_4, x_(e5)=>L_5});
--print minid;

minia=sub(polA,{x_(e0)=>L_0, x_(e1)=>L_1, x_(e2)=>L_2,
 x_(e3)=>L_3, x_(e4)=>L_4, x_(e5)=>L_5});
--print leadCoefficient minia;

minic=sub(polC,{x_(e0)=>L_0, x_(e1)=>L_1, x_(e2)=>L_2,
 x_(e3)=>L_3, x_(e4)=>L_4, x_(e5)=>L_5});
--print leadCoefficient minib;

minib=sub(polB,{x_(e0)=>L_0, x_(e1)=>L_1, x_(e2)=>L_2,
```

```
 x_(e3)=>L_3, x_(e4)=>L_4, x_(e5)=>L_5});
--print leadCoefficient minic;


dL=leadCoefficient minid;--d(L)
aL=leadCoefficient minia;--a(L)
bL=leadCoefficient minib;--b(L)
cL=leadCoefficient minic;--c(L)
--Checking if ad/bc>8/7
(dL*aL)/(bL*cL)>8/7

-----------------------------------------
--List of points for matroids on the table
-----------------------------------------

--L={1,2302/100000,2302/100000,88/100000,1936/100000,
--1936/100000}
--L8_768-Jindx27
---------------------------------

--L={1,4096/10000,4234/10000,493/10000,1071/10000,373/10000
--L8_816-Jindx2
---------------------------------

--L={1,9348/10000,4050/10000,4050/10000,799/10000,799/10000}
--L8_821-Jindx2
---------------------------------

--L={1,092/1000,1/10,7/1000,52/1000,4/1000}
--L8_825-Jindx2
---------------------------------

--L={1,2673/10000,2822/10000,735/10000,1128/10000,63/10000}
--L8_878-Jindx19
---------------------------------

--L={1,296/100,305/100,640/100,48/100,7/100}
--L8_879-Jindx19
---------------------------------

--L={1,94/1000,102/1000,11/1000,56/1000,2/1000}
--L8_882-Jindx19
---------------------------------

--L={1,13/100,13/100,2/100,4/100,4/1000}
--L8_883-Jindx19
---------------------------------

--L={1,19/100,141/100,141/100,19/100,19/100}
--L8_891-Jindx27
---------------------------------

--L={1,74/1000,73/1000,7/1000,73/1000,3/1000}
```

```
--L8_894-Jindx27
---------------------------------

--L={1,37/1000,36/1000,2/1000,36/1000,1/1000}
--L8_895-Jindx27
---------------------------------

--L={1,3/10000,3/10000,3/10000,3/10000,3/10000}
--L8_896-Jindx27
---------------------------------

--L={1,3/100,3/100,2/1000,3/100,1/1000}
--L8_910-Jindx27
---------------------------------

--L={1,2/100,2/100,1/1000,2/100,1/1000}
--L8_911-Jindx27
---------------------------------

--L={1,25/100000,25/100000,25/100000,25/100000,25/100000}
--L8_912-Jindx27
---------------------------------
```

1.6. **Sage Code for using the Catalog of Matroids.** In this section, we provide code for testing whether some of the matroids we have correspond to some matroids from the catalog of matroids in `SageMath`.

We start with defining a list with matroids from catalog of matroids in Sage. We also extend the list by adding their duals, extensions and coextensions.

```
##Catalog of Matroids from:
##https://doc.sagemath.org/html/en/reference/matroids/sage
##/matroids/matroids_catalog
Cat=[matroids.named_matroids.AG23minus(),
    matroids.named_matroids.AG32prime(),
    matroids.named_matroids.BetsyRoss(),
    matroids.named_matroids.Block_9_4(),
    matroids.named_matroids.Block_10_5(),
    matroids.named_matroids.D16(),
    matroids.named_matroids.ExtendedBinaryGolayCode(),
    matroids.named_matroids.ExtendedTernaryGolayCode(),
    matroids.named_matroids.F8(),
    matroids.named_matroids.Fano(),matroids.named_matroids.J(),
    matroids.named_matroids.K33dual(),
    matroids.named_matroids.L8(),
    matroids.named_matroids.N1(),matroids.named_matroids.N2(),
    matroids.named_matroids.NonFano(),
    matroids.named_matroids.NonPappus(),
    matroids.named_matroids.NonVamos(),
    matroids.named_matroids.NotP8(),
    matroids.named_matroids.O7(),matroids.named_matroids.P6(),
    matroids.named_matroids.P7(),
    matroids.named_matroids.P8(),matroids.named_matroids.P8pp(),
    matroids.named_matroids.P9(),
    matroids.named_matroids.Pappus(),
    matroids.named_matroids.Q6(),
    matroids.named_matroids.Q8(),
    matroids.named_matroids.Q10(),matroids.named_matroids.R6(),
    matroids.named_matroids.R8(),
    matroids.named_matroids.R9A(),matroids.named_matroids.R9B(),
    matroids.named_matroids.R10(),
    matroids.named_matroids.R12(),matroids.named_matroids.S8(),
    matroids.named_matroids.T8(),
    matroids.named_matroids.T12(),
    matroids.named_matroids.TernaryDowling3(),
    matroids.named_matroids.Terrahawk(),
    matroids.named_matroids.TicTacToe(),
    matroids.named_matroids.Vamos(),
    matroids.Wheel(4),matroids.Whirl(4),matroids.AG(2, 3)]

##We define functions to keep track of the name of a matroid
##from the catalog
##while adding more matroids to the list

def Dual(M):
##Function for keeping the name of M while taking its dual
     DM=M.dual()
```

```
     RepM=repr(M)##Name of the matroid from the catalog
     r=RepM.rfind(":")
     DM.rename("Dual("+RepM[:r]+"): "+repr(DM))##Naming the dual
     return (DM)

CatD=[Dual(m) for m in Cat ]
##List of duals of matroids from the catalog

def Ext(M):
##Function for keeping the name of M in its extension
     ExtM=M.extension() ##Extension of the matroid
     RepM=repr(M) ##Name of the matroid from the catalog
     r=RepM.rfind(":")
     ExtM.rename("Ext("+RepM[:r]+"): "+repr(ExtM))
     ##Naming the extension
     return (ExtM)

ExtCat=[Ext(m) for m in Cat]
##List of extension of matroids form the catalog

def CoExt(M):
##Function for keeping the name of M in its co-extension
     CoExtM=M.coextension() ##Co-extension of the matroid
     RepM=repr(M) ##Name of the matroid from the catalog
     r=RepM.rfind(":")
     CoExtM.rename("CoExt("+RepM[:r]+"): "+repr(CoExtM))
    ##Naming the co-extension
     return (CoExtM)

CoextCat=[CoExt(m) for m in Cat]
##List of co-extension of matroids form the catalog

##Putting all the lists together
CAT=Cat+CatD+ExtCat+CoextCat ##Big catalog
```

Now, we run Macaulay2 in Sage in order to call the file that contains the list of bases of matroids that are in the set "Hpcand" in Macaulay2.

```
% macaulay2--Runs Macaulay2 in Sage
----> Switching to Macaulay2 <--
f="/home/.../.../BofHpcand";
--Calls the file: fill in the file location
--Warning: This is the list of 309 matroids,
--NOT the list of all matroids on 8 elements
BofHpcand=value get f; --It calls the list in Macaulay2 syntax
--The syntax is of the form
--{{set{},...,set{}},...,{set{},...,set{}}}
--Ctrl-D
---> Exiting back to Sage <--
```

Then we exit back to Sage and convert the list to Sage syntax as follows:

```
def convert(s):
##Function to convert M2 syntax to Sage syntax for list of Bases
    conv=list(eval(s.replace("{","[").replace("set","").
```

```
replace("}","]")[1:-1]))
    return (conv)

BofMs=convert(str(macaulay2("BofHpcand")))
##List of Bases in Sage syntax
##The syntax is of the form [[[],...,[]],...,[[],...,[]]]
```

We provide the list of indices of matroids we are interested in namely, those that don't have the half-plane property and those that don't have a determinantal representation in the list of 309 matroids (the list Hpcand). Then, we run an isomorphism test to see if some of them are isomorphic to some matroids from the big catalog. Moreover, on the matroids that don't have the half-plane property, we test whether they have some special matroids as minors, and whether some special matroids have them they as a minors.

Please note that indices given in the previous section are in the list of all matroids on 8 element (the list L8 in the code, the list $\mathcal{M}$ in the text).

```
##Indices of 22 matroids that don't have Hpp in the list Hpcand
NotHpp=[73,76,77,81,82,83,85,88,92,93,94,95,96,97,99,100,101,
112,113,114,116,117]

##Isomorphism Test on NotHpp
for i in NotHpp:
     B=BofMs[i]
     M=Matroid(bases=B)
     Iso=[]
     for j in range(len(CAT)):
          Mcat=CAT[j]
          if M.is_isomorphic(Mcat):
               Iso.append(Mcat)
     print("--------------------")
     print(i,Iso)

##Testing which minors they have
for i in NotHpp:
     B=BofMs[i]
     M=Matroid(bases=B)
     Hasminor=[]
     for j in range(len(CAT)):
          Mcat=CAT[j]
          E=Mcat.groundset()
          n=len(E)
          if n<9:
               if M.has_minor(Mcat):
                    Hasminor.append(Mcat)
     print("----------------------")
     print(i,Hasminor)

##Testing which matroids have them as minors
##Loop takes a lot of time, we recommend to do them one by one
i=NotHpp[0]
B=BofMs[i]
M=Matroid(bases=B)
IsminorOf=[]
```

```
for j in range(len(CAT)):
     Mcat=CAT[j]
     E=Mcat.groundset()
     n=len(E)
     if n>8:
          if Mcat.has_minor(M):
               IsminorOf.append(Mcat)
print(i,IsminorOf)

##Isomorphism Test on NotDet
for i in NotDet:
     B=BofMs[i]
     M=Matroid(bases=B)
     Iso=[]
     for j in range(len(CAT)):
          Mcat=CAT[j]
          if M.is_isomorphic(Mcat):
               Iso.append(Mcat)
     print("--------------------")
     print(i,Iso)
##Hpcand_110 is isomorphic to the Vamos matroid.
##None of the other matroids are isomorphic
##to a matroid from CAT
```

2. Tables

Matroids	**$\#(\mathbf{e},\mathbf{v})$ for which $\mathbf{h_M(et-v)}$ has non-real roots**	**One of the pairs $(\mathbf{e},\mathbf{v})$**
$\mathcal{M}_{435}$	4	$((1,1,1,1,1,1,0,1),(1,1,0,0,0,0,1,1))$
$\mathcal{M}_{437}$	8	$((0,0,0,0,1,1,1,1),(1,1,1,1,1,1,0,1))$
$\mathcal{M}_{439}$	4	$((1,0,1,1,1,1,1,1),(1,1,1,1,0,0,0,0))$
$\mathcal{M}_{443}$	2	$((1,1,1,1,0,1,1,1),(0,0,0,0,1,1,1,1))$
$\mathcal{M}_{450}$	4	$((0,1,1,1,1,1,1,1),(1,1,0,0,1,1,0,0))$
$\mathcal{M}_{455}$	2	$((1,1,0,1,0,1,0,1),(0,1,1,0,1,0,1,1))$
$\mathcal{M}_{460}$	2	$((0,0,1,1,1,1,1,1),(1,1,0,0,0,0,1,1))$
$\mathcal{M}_{461}$	12	$((1,1,1,1,0,1,1,1),(0,0,1,1,1,1,0,0))$
$\mathcal{M}_{465}$	10	$((0,1,1,1,1,1,1,1),(1,1,0,0,0,0,1,1))$
$\mathcal{M}_{466}$	38	$((0,0,1,1,1,1,0,1),(1,1,0,0,0,0,1,1))$
$\mathcal{M}_{467}$	62	$((1,1,0,1,1,0,1,1),(0,0,1,1,1,1,0,0))$
$\mathcal{M}_{548}$	6	$((1,1,1,1,0,0,0,1),(0,0,0,0,1,1,1,1))$
$\mathcal{M}_{549}$	4	$((1,1,1,1,0,0,1,1),(1,1,1,1,0,0,1,1))$
$\mathcal{M}_{570}$	78	$((1,1,0,0,1,1,1,1),(0,0,1,1,1,1,0,0))$
$\mathcal{M}_{575}$	158	$((1,1,1,1,1,1,0,1),(0,0,0,1,1,0,1,1))$

Table 1. 15 matroids and a sample of directions for which they fail the hyperbolicity test

Matroids	$\mathbf{x \in \mathbb{R}^6}$ **s.t.** $\mathbf{\Delta_{6,7}(h_M(x)) < 0}$
$\mathcal{M}_{424}$	$(4, 30, 1, 7, -32, -4)$
$\mathcal{M}_{430}$	$(80, 19, -31, -31, -17, -4)$
$\mathcal{M}_{431}$	$(60, 27, -90, -22, 27, 5)$
$\mathcal{M}_{436}$	$(40, 309, -40, -306, 9, 73)$
$\mathcal{M}_{462}$	$(20, 55, -11, -4, -52, -19)$
$\mathcal{M}_{463}$	$(30, 399, -111, -10, -368, -28)$
$\mathcal{M}_{550}$	$(50, -25, 94, -45, -142, 66)$

TABLE 2. 7 matroids and points $x \in \mathbb{R}^6$ that disprove the HPP

	(i,j)	Points p	$\Delta_{i,j}(p)$	$\frac{ad}{bc}(p)$
$\mathcal{M}_{768}$	$(0,7)$	$(x_1,\dots,x_6)$ $(1, \frac{1151}{50000}, \frac{1151}{50000}, \frac{11}{12500}, \frac{121}{6250}, \frac{121}{6250})$	$-\frac{137044575881655277}{244140625.10^{18}}$	$\frac{7087627531101625600}{7086494931300950763}$
$\mathcal{M}_{816}$	$(5,2)$	$(x_0,x_1,x_3,x_4,x_6,x_7)$ $(1, \frac{256}{625}, \frac{2117}{5000}, \frac{493}{10000}, \frac{1071}{10000}, \frac{373}{10000})$	$-\frac{44144993633423974179}{10^{24}}$	$\frac{16778353097134052416 30}{16757331671594326714 31}$
$\mathcal{M}_{821}$	$(5,2)$	$(x_0,x_1,x_3,x_4,x_6,x_7)$ $(1, \frac{2337}{2500}, \frac{81}{200}, \frac{81}{200}, \frac{799}{10000}, \frac{799}{10000})$	$-\frac{244470035602773648449}{25.10^{22}}$	$\frac{2106739084957900875}{2101723183203601984}$
$\mathcal{M}_{825}$	$(5,2)$	$(x_0,x_1,x_3,x_4,x_6,x_7)$ $(1, \frac{23}{250}, \frac{1}{10}, \frac{7}{1000}, \frac{13}{250}, \frac{1}{250})$	$-\frac{428814189}{390625.10^{10}}$	$\frac{17057286016}{17041404009}$
$\mathcal{M}_{878}$	$(0,5)$	$(x_1,x_2,x_3,x_4,x_6,x_7)$ $(1, \frac{2673}{10000}, \frac{1411}{5000}, \frac{147}{2000}, \frac{141}{1250}, \frac{63}{10000})$	$-\frac{7274235177449194527}{10^{24}}$	$\frac{10731828475989312509 30}{10723745992458813404 27}$
$\mathcal{M}_{879}$	$(0,5)$	$(x_1,x_2,x_3,x_4,x_6,x_7)$ $(1, \frac{17}{25}, \frac{61}{20}, \frac{32}{5}, \frac{12}{25}, \frac{7}{100})$	$-\frac{167603609619}{625.10^{8}}$	$\frac{6212735812250}{6206528271153}$
$\mathcal{M}_{882}$	$(0,5)$	$(x_1,x_2,x_3,x_4,x_6,x_7)$ $(1, \frac{47}{500}, \frac{51}{500}, \frac{11}{1000}, \frac{7}{125}, \frac{1}{500})$	$-\frac{1444139463}{125.10^{15}}$	$\frac{574575099022}{574093719201}$
$\mathcal{M}_{883}$	$(0,5)$	$(x_1,x_2,x_3,x_4,x_6,x_7)$ $(1, \frac{13}{100}, \frac{13}{100}, \frac{1}{50}, \frac{1}{25}, \frac{1}{250})$	$-\frac{423857}{15625.10^{8}}$	$\frac{208804463}{208562259}$
$\mathcal{M}_{891}$	$(0,7)$	$(x_1,\dots,x_6)$ $(1, \frac{19}{100}, \frac{141}{100}, \frac{141}{100}, \frac{19}{100}, \frac{19}{100})$	$-\frac{3136366917}{2.10^{11}}$	$\frac{2377183363}{2371329748}$
$\mathcal{M}_{894}$	$(0,7)$	$(x_1,\dots,x_6)$ $(1, \frac{37}{500}, \frac{73}{1000}, \frac{7}{1000}, \frac{73}{1000}, \frac{3}{1000})$	$-\frac{55652830199}{10^{18}}$	$\frac{88756978376279}{88701325546080}$
$\mathcal{M}_{895}$	$(0,7)$	$(x_1,\dots,x_6)$ $(1, \frac{37}{1000}, \frac{9}{250}, \frac{1}{500}, \frac{9}{250}, \frac{1}{1000})$	$-\frac{673510083}{25.10^{16}}$	$\frac{1113113239348}{1112439729265}$
$\mathcal{M}_{896}$	$(0,7)$	$(x_1,\dots,x_6)$ $(1, \frac{3}{1000}, \frac{3}{1000}, \frac{3}{1000}, \frac{3}{1000}, \frac{3}{1000})$	$-\frac{29157813}{10^{25}}$	$\frac{3202760405}{3202400432}$
$\mathcal{M}_{910}$	$(0,7)$	$(x_1,\dots,x_6)$ $(1, \frac{3}{100}, \frac{3}{100}, \frac{1}{500}, \frac{3}{100}, \frac{1}{1000})$	$-\frac{1784831}{25.10^{14}}$	$\frac{5380022943}{5378238112}$
$\mathcal{M}_{911}$	$(0,7)$	$(x_1,\dots,x_6)$ $(1, \frac{1}{50}, \frac{1}{50}, \frac{1}{1000}, \frac{1}{50}, \frac{1}{1000})$	$-\frac{480249}{25.10^{14}}$	$\frac{263808}{263687}$
$\mathcal{M}_{912}$	$(0,7)$	$(x_1,\dots,x_6)$ $(1, \frac{1}{4000}, \frac{1}{4000}, \frac{1}{4000}, \frac{1}{4000}, \frac{1}{4000})$	$-\frac{1}{64.10^{15}}$	$\frac{16010001}{16008001}$

TABLE 3. List of non-Rayleigh matroids, points p and the correlation constants $\frac{ad}{bc}(p)$.

TECHNISCHE UNIVERSITÄT DRESDEN, GERMANY
Email address: mario.kummer@tu-dresden.de

TECHNISCHE UNIVERSITÄT DRESDEN, GERMANY
Email address: buesra.sert@tu-dresden.de