# SAGE computations used in the paper *Orientations of Quaternary Matroids*

April 5, 2021

## 1 Two matroids without consistent orientations

In this section we consider the two rank-3 quaternary matroids whose matrices are shown below. One of these matroids is the Pappus matroid. We use two alternative labelings for the elements of these matroids. In this section we simply check that these labelings are correct. In Section 5.2.3 of our main paper we show that these matroids have no consistently ordered orientations.

```
In [1]: from sage.matroids.advanced import *

        GF4.<w>=GF(4)

        ExcludedR3E9Matrices=[
        matrix(GF4,[[1, 0, 0, 1, 1, w + 1, w, w, 0],
        [0, 1, 0, 1, w, w, 0, 1, 1],
        [0, 0, 1, 1, w + 1, 1, 1, 0, 1]]),
        matrix(GF4,[[1, 0, 0, 1, 1, w + 1, w, w, w],
        [0, 1, 0, 1, w, w, 0, 1, w + 1],
        [0, 0, 1, 1, w + 1, 1, 1, 0, w + 1]])]

        ExcludedR3E9=[]
        for A in ExcludedR3E9Matrices:
            ExcludedR3E9=ExcludedR3E9+[Matroid(A)]

        AlternateLabelings=[
        Matroid(circuit_closures=
                {2:[[1,4,7],[2,3,7],[5,6,7],[1,6,8],[2,5,8],
                    [3,4,8],[1,3,9],[2,6,9],[4,5,9],[7,8,9]],
                 3:[[1,2,3,4,5,6,7,8,9]]}),
        Matroid(circuit_closures=
                {2:[[1,4,7],[2,3,7],[5,6,7],
                    [1,6,8],[2,5,8],[3,4,8],
                    [1,2,9],[3,6,9],[4,5,9]],
                 3:[[1,2,3,4,5,6,7,8,9]]})]

        for i in [0,1]:
            print ExcludedR3E9[i].is_isomorphic(AlternateLabelings[i])
```

```
True
True
```

```
In [2]: matroids.named_matroids.Pappus().is_isomorphic(AlternateLabelings[1])
```

```
Out[2]: True
```

## 2 All rank-3 matroids with consistent orientations

Three rank-3 quaternary matroids which are known to not be orientable are the Fano matroid and $AG(2,3)\backslash e$ (also known as the MacLane matroid). In Section 5.2 of our main paper we also showed that the two matroids of Section 1 have no consistent orientations. The following is a calculation of all rank-3 matroids which

- are quaternary and simple,
- contain a $U_{3,5}$-minor, and
- do not contain any of the Fano matroid, the MacLane matroid and the two matroids of Section 1 as a minor.

We also calculate all of the $GF(5)$-representations of these matroids. We therefore get all of the $GF(4) \times GF(5)$-representations as well as the $\mathbb{C}$-representations of these matroids.

We also test for the existence of a $U_{3,6}$- and $X_7$-minor in each matroid. The matroid $X_7$ is defined in the code below. It is also show in Figure 21 of our main paper.

```
In [1]: from sage.matroids.advanced import *
```

```
In [2]: GF4.<w>=GF(4)

        U35OverF5=[
            Matroid(matrix(GF(5),[[1,0,0,1,1],[0,1,0,1,2],[0,0,1,1,3]])),
            Matroid(matrix(GF(5),[[1,0,0,1,1],[0,1,0,1,2],[0,0,1,1,4]])),
            Matroid(matrix(GF(5),[[1,0,0,1,1],[0,1,0,1,3],[0,0,1,1,4]])),
            Matroid(matrix(GF(5),[[1,0,0,1,1],[0,1,0,1,3],[0,0,1,1,2]])),
            Matroid(matrix(GF(5),[[1,0,0,1,1],[0,1,0,1,4],[0,0,1,1,2]])),
            Matroid(matrix(GF(5),[[1,0,0,1,1],[0,1,0,1,4],[0,0,1,1,3]]))]

        U35OverF4=Matroid(matrix(GF4,[[1,0,0,1,1],[0,1,0,1,w],[0,0,1,1,w+1]]))

        ExcludedR3E9=[
            Matroid(circuit_closures=
                {2:[[1,4,7],[2,3,7],[5,6,7],[1,6,8],[2,5,8],
                    [3,4,8],[1,3,9],[2,6,9],[4,5,9],[7,8,9]],
                 3:[[1,2,3,4,5,6,7,8,9]]}),
            Matroid(circuit_closures=
                {2:[[1,4,7],[2,3,7],[5,6,7],
                    [1,6,8],[2,5,8],[3,4,8],
```

```
                [1,2,9],[3,6,9],[4,5,9]],
             3:[[1,2,3,4,5,6,7,8,9]]})]

        U36=Matroid(matrix(GF4,[[0, 1, 0, 0, 1, 1, 1],
                                [0, 0, 1, 0, 1, w, w+1],
                                [0, 0, 0, 1, 1, w+1, w]])).delete(0)
        X7=Matroid(matrix(GF4,[[0,1,0,1,0,1,1,1],
                               [0,0,1,1,0,0,w,1],
                               [0,0,0,0,1,1,w,1]])).delete(0)
```

In [3]:
```
def HasExcludedMinor(M):
    Result=False
    if not Result:
        for N in ExcludedR3E9:
            Result=M.has_minor(N)
            if Result:
                break
    if not Result:
        Result=M.has_minor(matroids.named_matroids.Fano()) or
                           M.has_minor(matroids.named_matroids.AG23minus())
    return Result

def ExtendGF4AndGF5(R):
    ExtList=get_nonisomorphic_matroids(R[4].linear_extensions(simple=True))
    FinalList=[]
    for N in ExtList:
        if not HasExcludedMinor(N):
            FinalList=FinalList+[{4:N,5:[]}]
    for M in R[5]:
        for N in M.linear_extensions(simple=True):
            for K in FinalList:
                if N.equals(K[4]):
                    K[5]=K[5]+[N]
    return FinalList

def SimplifyMultilist(Multilist):
    FinalList=[]
    for i in range(len(Multilist)):
        for j in range(len(Multilist[i])):
            ijAppearsLater=False
            for k in range(i+1,len(Multilist)):
                for l in range(len(Multilist[k])):
                    ijAppearsLater=Multilist[i][j][4].is_isomorphic(Multilist[k][l][4])
                    if ijAppearsLater:
                        break
                if ijAppearsLater:
                    break
            if not ijAppearsLater:
```

```
                    FinalList=FinalList+[Multilist[i][j]]
        return FinalList
```

```
In [4]: GoldenRank3Catalog={5:[{4:U35OverF4,5:U35OverF5}]}
        for i in range(5,21):
            iElements=GoldenRank3Catalog[i]
            iPlusOneMultilist=[]
            for MRecord in iElements:
                iPlusOneMultilist=iPlusOneMultilist+[ExtendGF4AndGF5(MRecord)]
            iPlusOne=SimplifyMultilist(iPlusOneMultilist)
            print "There are",len(iPlusOne),"matroids on",i+1,"elements"
            k=0
            for MRecord in iPlusOne:
                print("Matroid",k,"has",len(MRecord[5]),
                      "GF(5)-reps. U36-minor and X7-minor",
                      MRecord[4].has_minor(U36),MRecord[4].has_minor(X7))
                k=k+1
            print
            if len(iPlusOne)==0:
                break
            else:
                GoldenRank3Catalog.update({i+1:iPlusOne})
```

```
There are 2 matroids on 6 elements
('Matroid', 0, 'has', 6, 'GF(5)-reps. U36-minor and X7-minor', False, False)
('Matroid', 1, 'has', 6, 'GF(5)-reps. U36-minor and X7-minor', True, False)

There are 4 matroids on 7 elements
('Matroid', 0, 'has', 6, 'GF(5)-reps. U36-minor and X7-minor', False, False)
('Matroid', 1, 'has', 6, 'GF(5)-reps. U36-minor and X7-minor', False, False)
('Matroid', 2, 'has', 2, 'GF(5)-reps. U36-minor and X7-minor', False, True)
('Matroid', 3, 'has', 4, 'GF(5)-reps. U36-minor and X7-minor', True, False)

There are 8 matroids on 8 elements
('Matroid', 0, 'has', 6, 'GF(5)-reps. U36-minor and X7-minor', False, False)
('Matroid', 1, 'has', 6, 'GF(5)-reps. U36-minor and X7-minor', False, False)
('Matroid', 2, 'has', 6, 'GF(5)-reps. U36-minor and X7-minor', False, False)
('Matroid', 3, 'has', 6, 'GF(5)-reps. U36-minor and X7-minor', False, False)
('Matroid', 4, 'has', 2, 'GF(5)-reps. U36-minor and X7-minor', False, True)
('Matroid', 5, 'has', 2, 'GF(5)-reps. U36-minor and X7-minor', False, True)
('Matroid', 6, 'has', 2, 'GF(5)-reps. U36-minor and X7-minor', True, True)
('Matroid', 7, 'has', 3, 'GF(5)-reps. U36-minor and X7-minor', True, False)

There are 9 matroids on 9 elements
('Matroid', 0, 'has', 6, 'GF(5)-reps. U36-minor and X7-minor', False, False)
('Matroid', 1, 'has', 6, 'GF(5)-reps. U36-minor and X7-minor', False, False)
('Matroid', 2, 'has', 6, 'GF(5)-reps. U36-minor and X7-minor', False, False)
('Matroid', 3, 'has', 1, 'GF(5)-reps. U36-minor and X7-minor', False, True)
```

```
('Matroid', 4, 'has', 2, 'GF(5)-reps. U36-minor and X7-minor', False, True)
('Matroid', 5, 'has', 2, 'GF(5)-reps. U36-minor and X7-minor', False, True)
('Matroid', 6, 'has', 2, 'GF(5)-reps. U36-minor and X7-minor', False, True)
('Matroid', 7, 'has', 1, 'GF(5)-reps. U36-minor and X7-minor', True, True)
('Matroid', 8, 'has', 2, 'GF(5)-reps. U36-minor and X7-minor', True, True)

There are 6 matroids on 10 elements
('Matroid', 0, 'has', 6, 'GF(5)-reps. U36-minor and X7-minor', False, False)
('Matroid', 1, 'has', 6, 'GF(5)-reps. U36-minor and X7-minor', False, False)
('Matroid', 2, 'has', 2, 'GF(5)-reps. U36-minor and X7-minor', False, True)
('Matroid', 3, 'has', 1, 'GF(5)-reps. U36-minor and X7-minor', False, True)
('Matroid', 4, 'has', 2, 'GF(5)-reps. U36-minor and X7-minor', True, True)
('Matroid', 5, 'has', 1, 'GF(5)-reps. U36-minor and X7-minor', True, True)

There are 1 matroids on 11 elements
('Matroid', 0, 'has', 1, 'GF(5)-reps. U36-minor and X7-minor', True, True)

There are 0 matroids on 12 elements
```

# 3 Sorting the catalogue according to the existence of $X_7$- and $U_{3,6}$-minors

In this section we partition the catalogue of matroids from Section 2 into four blocks according to the existence of $X_7$- and $U_{3,6}$-minors.

In Section 3.1 we analyze matroids with an $X_7$-minor and no $U_{3,6}$-minor. We build the matroids from $X_7$ using single-element extensions and produce a labeling of the elements for each. These labelings are used in our main paper to construct all consistent orientations of each matroid in the class.

In Section 3.2 we do the same for the class of matroids with both $X_7$- and $U_{3,6}$-minors. The smallest matroid in this class has 8-elements as seen in the calculation of Section 2. We call this matroid $UX_8$. As we will see in this section, all other matroids in the class are built from via single-element extensions from $UX_8$.

In Section 3.3, we analyze the class of matroids with neither an $X_7$- nor $U_{3,6}$-minor. We show that each matroid in the class can be constructed via a sequence of single-element extensions starting with $U_{3,5}$ in which each dual coextension is permeating. Since the dual coextensions are permeating, no orientation of any matroid in the class extends to more than one orientation.

We do not do any calculations for the class of matroids without an $X_7$-minor and with a $U_{3,6}$-minor. This class is addressed in the main paper without need of further calculations here.

```
In [1]: from sage.matroids.advanced import *

In [2]: GF4.<w>=GF(4)

In [3]: ExcludedR3E9=[
        Matroid(circuit_closures=
```

```
                {2:[[1,4,7],[2,3,7],[5,6,7],[1,6,8],[2,5,8],
                    [3,4,8],[1,3,9],[2,6,9],[4,5,9],[7,8,9]],
                 3:[[1,2,3,4,5,6,7,8,9]]}),
        Matroid(circuit_closures=
                {2:[[1,4,7],[2,3,7],[5,6,7],
                    [1,6,8],[2,5,8],[3,4,8],
                    [1,2,9],[3,6,9],[4,5,9]],
                 3:[[1,2,3,4,5,6,7,8,9]]})]

In [4]: U35=Matroid(matrix(GF4,[[0,1,0,0,1,1],
                                [0,0,1,0,1,w],
                                [0,0,0,1,1,w+1]])).delete(0)
        U36=Matroid(matrix(GF4,[[0, 1, 0, 0, 1, 1, 1],
                                [0, 0, 1, 0, 1, w, w+1],
                                [0, 0, 0, 1, 1, w+1, w]])).delete(0)
        X7=Matroid(matrix(GF4,[[0,1,0,1,0,1,1,1],
                               [0,0,1,1,0,0,w,1],
                               [0,0,0,0,1,1,w,1]])).delete(0)
        UX8=Matroid(matrix(GF4,[[0,1,0,0,1,1,1,0,1],
                               [0,0,1,0,1,w,w+1,1,w+1],
                               [0,0,0,1,1,w+1,w,1,w+1]])).delete(0)

In [5]: print U35.circuit_closures()
        print
        print U36.circuit_closures()
        print
        for c in X7.circuit_closures()[2]:
            print list(c)
        print
        for c in UX8.circuit_closures()[2]:
            print list(c)

{3: set([frozenset([1, 2, 3, 4, 5])])}

{3: set([frozenset([1, 2, 3, 4, 5, 6])])}

[3, 4, 7]
[1, 4, 5]
[1, 6, 7]
[1, 2, 3]
[2, 5, 7]

[5, 6, 7]
[2, 3, 7]
[8, 3, 6]
[8, 1, 4, 7]
[8, 2, 5]
```

In [6]:
```
#M is single-element coextension over N in all 3 functions

def LinearClass(M,N):
    LM=set(M.circuits())
    LN=set(N.circuits())
    return LM.intersection(LN)

def NotLinearClass(M,N):
    LM=set(M.circuits())
    LN=set(N.circuits())
    return LN.difference(LM)


def Permeation(M,N):
    EdgeList=[]
    LC=list(LinearClass(M,N))
    NL=list(NotLinearClass(M,N))
    NumVerts=len(NL)
    for i in range(NumVerts):
        j=0
        while j<i:
            IsModular=(2==len(NL[i].union(NL[j]))-N.rank(NL[i].union(NL[j])))
            if IsModular:
                Straddling=False
                for S in subsets(NL[i].union(NL[j])):
                    Straddling=frozenset(S) in LC
                    if Straddling:
                        EdgeList=EdgeList+[(i,j)]
                        break
            j=j+1
    G=Graph(EdgeList)
    return G.is_connected() and G.order()==NumVerts
```

In [7]:
```
def HasFanoOrMacLaneOrNiner(M):
    return (M.has_minor(matroids.named_matroids.Fano()) or
            M.has_minor(matroids.named_matroids.AG23minus()) or
            M.has_minor(ExcludedR3E9[0]) or
            M.has_minor(ExcludedR3E9[1]))


def ExtendThis(M):
    L=[]
    I=get_nonisomorphic_matroids(
        M.linear_extensions(element=M.size()+1,simple=true))
    for N in I:
        if not HasFanoOrMacLaneOrNiner(N):
            L=L+[N]
    return L
```

```
        def ExtendThisU36Free(M):
            L=[]
            I=get_nonisomorphic_matroids(
                M.linear_extensions(element=M.size()+1,simple=true))
            for N in I:
                if not HasFanoOrMacLaneOrNiner(N) and not N.has_minor(U36):
                    L=L+[N]
            return L

        def ExtendThisX7Free(M):
            L=[]
            I=get_nonisomorphic_matroids(
                M.linear_extensions(element=M.size()+1,simple=true))
            for N in I:
                    if not HasFanoOrMacLaneOrNiner(N) and not N.has_minor(X7):
                        L=L+[N]
            return L

        def ExtendThisX7andU36Free(M):
            L=[]
            I=get_nonisomorphic_matroids(
                M.linear_extensions(element=M.size()+1,simple=true))
            for N in I:
                if not HasFanoOrMacLaneOrNiner(N) and not N.has_minor(U36) and not N.has_minor(X
                    L=L+[N]
            return L
```

### 3.1 Matroids with an $X_7$-minor but no $U_{3,6}$-minor

```
In [8]: "First, we examine the class of rank-3 golden-mean matroids which"
        "contain an X7-minor but no U36-minor. We know how many matroids"
        "there are on each number of elements and the number of golden-mean"
        "representations. There are two 8-element matroids in this class."
        X7e8=ExtendThisU36Free(X7)
        for M in X7e8:
            for c in M.circuit_closures()[2]:
                if 8 not in list(c):
                    print list(c)
            for c in M.circuit_closures()[2]:
                if 8 in list(c):
                    print list(c)
            print
```

```
[2, 5, 7]
[1, 6, 7]
[3, 4, 7]
[1, 4, 5]
```

```
[8, 4, 6]
[8, 1, 2, 3]

[1, 6, 7]
[3, 4, 7]
[1, 4, 5]
[2, 5, 7]
[1, 2, 3]
[8, 5, 6]
[8, 2, 4]
```

In [9]:
```
"There are four 9-element matroids in this class. All four matroids on"
"9 elements in this class are single-element extensions of X7e8[0]"
X7e9=ExtendThisU36Free(X7e8[0])
for M in X7e9:
    for c in M.circuit_closures()[2]:
        if 9 not in list(c):
            print list(c)
    for c in M.circuit_closures()[2]:
        if 9 in list(c):
            print list(c)
    print
```

```
[1, 6, 7]
[3, 4, 7]
[1, 4, 5]
[8, 4, 6]
[2, 5, 7]
[9, 5, 6]
[8, 1, 2, 3, 9]

[8, 1, 2, 3]
[2, 5, 7]
[1, 4, 5]
[8, 4, 6]
[1, 6, 7]
[9, 5, 6]
[9, 3, 4, 7]

[2, 5, 7]
[1, 6, 7]
[3, 4, 7]
[1, 4, 5]
[8, 4, 6]
[8, 1, 2, 3]
[8, 9, 7]
```

```
[9, 3, 5]
[9, 2, 6]

[1, 6, 7]
[3, 4, 7]
[1, 4, 5]
[8, 1, 2, 3]
[8, 9, 4, 6]
[9, 2, 5, 7]
```

```
In [10]: "There are two 10-element matroids in this classe. Both 10-element"
         "matroids in this class are single-element extensions of X7e9[0]."

         X7e10=ExtendThisU36Free(X7e9[0])
         for M in X7e10:
             for c in M.circuit_closures()[2]:
                 if 10 not in list(c):
                     print list(c)
             for c in M.circuit_closures()[2]:
                 if 10 in list(c):
                     print list(c)
             print
```

```
[1, 6, 7]
[8, 1, 2, 3, 9]
[1, 4, 5]
[8, 4, 6]
[2, 5, 7]
[9, 10, 5, 6]
[10, 3, 4, 7]

[9, 5, 6]
[1, 6, 7]
[3, 4, 7]
[8, 1, 2, 3, 9]
[1, 4, 5]
[8, 4, 6]
[2, 5, 7]
[10, 3, 5]
[8, 10, 7]
[2, 10, 6]
[9, 10, 4]
```

### 3.2 Matroids with both an $X_7$-minor and $U_{3,6}$-minor

```
In [11]: "Second, we examine the class of rank-3 golden-mean matroids containing a"
         "UX8-minor. There are two single-element extensions of UX8."

         UX8e9=ExtendThis(UX8)
         for M in UX8e9:
             for c in M.circuit_closures()[2]:
                 if 9 not in list(c):
                     print list(c)
             for c in M.circuit_closures()[2]:
                 if 9 in list(c):
                     print list(c)
             print
```

```
[8, 1, 4, 7]
[2, 3, 7]
[5, 6, 7]
[8, 2, 5]
[1, 2, 9]
[9, 4, 5]
[8, 9, 3, 6]

[8, 3, 6]
[2, 3, 7]
[8, 1, 4, 7]
[5, 6, 7]
[8, 2, 5]
[1, 2, 9]
[9, 3, 5]
[9, 4, 6]
```

```
In [12]: "There are two 10-element matroids in this class. They are"
         "all single-element extensions of UX8e9[1]."

         UX8e10=ExtendThis(UX8e9[1])
         for M in UX8e10:
             for c in M.circuit_closures()[2]:
                 if 10 not in list(c):
                     print list(c)
             for c in M.circuit_closures()[2]:
                 if 10 in list(c):
                     print list(c)
             print
```

```
[8, 1, 4, 7]
[2, 3, 7]
```

```
[9, 3, 5]
[5, 6, 7]
[9, 4, 6]
[8, 2, 5]
[1, 2, 10, 9]
[8, 10, 3, 6]
[10, 4, 5]

[1, 2, 9]
[8, 3, 6]
[2, 3, 7]
[8, 1, 4, 7]
[5, 6, 7]
[9, 4, 6]
[8, 2, 5]
[1, 10, 6]
[2, 4, 10]
[9, 10, 3, 5]
```

```
In [13]: "The Betsy Ross matroid is the only 11-element golden-mean matroid"
         "and it contains a UX8-minor."
         BetsyRoss=ExtendThis(UX8e10[0])[0]
         for c in BetsyRoss.circuit_closures()[2]:
             if 11 not in list(c):
                 print list(c)
         for c in BetsyRoss.circuit_closures()[2]:
             if 11 in list(c):
                 print list(c)
```

```
[1, 2, 10, 9]
[8, 10, 3, 6]
[8, 1, 4, 7]
[9, 3, 5]
[5, 6, 7]
[10, 4, 5]
[8, 2, 5]
[9, 11, 4, 6]
[3, 2, 11, 7]
[1, 11, 5]
```

### 3.3 No $U_{3,6}$- and no $X_7$-minor

```
In [14]: "Lastly, we examine the class of rank-3 golden-mean matroids with"
         "a U35-minor but no U36-minor nor X7-minor. Previous computations"
         "show that these matroids all have 12 golden-mean representations."
```

```
        "We know that U35 has 12 golden-mean representations which correspond"
        "to its 12 orienatations. We will show that every other matroid M in"
        "this class is a single-element extension of a matroid N for which the"
        "associated dual coextension is permeating. Thus the 12 orientations"
        "extend uniquely at for each such single-element extension in this class and"
        "so each matroid in this class has 12 orientations corresponding to its"
        "12 golden-mean representaions."

        "Q6 is the only 6-element matroid in this class. The dual coextension"
        "from U35 to Q6 is permeating."

        U35e6=ExtendThisX7andU36Free(U35)
        for M in U35e6:
            print Permeation(M,U35)
```

```
True
```

```
In [15]: "There are two single-element extensions of Q6 within this class."
         "The dual coextensions are both permeating."

         U35e7=ExtendThisX7andU36Free(U35e6[0])
         for M in U35e7:
             print Permeation(M,U35e6[0])
```

```
True
True
```

```
In [16]: "There are four 8-element matroids in this class. They are all"
         "extensions of U35e7[1]. These dual coextensions are all permeating."

         for M in U35e7:
             for N in ExtendThisX7andU36Free(M):
                 print Permeation(N,M)
             print
```

```
True
True
True

True
True
True
True
```

```
In [17]: "There are three 9-element matroids in this class. They are not"
         "all exensions of a single matroids from U35e8, but all dual"
```

```
        "coextensions from 8 to 9 elements are permeating."

        U35e8=ExtendThisX7andU36Free(U35e7[0])
        for M in U35e8:
            for N in ExtendThisX7andU36Free(M):
                print Permeation(N,M)
            print
```

```
True
True

True
True

True
```

In [18]:
```
        "Here are the three 9-element matroids."
        L=[]
        for M in U35e8:
            L=L+ExtendThisX7andU36Free(M)
        U35e9=get_nonisomorphic_matroids(L)
        print len(U35e9)
```

```
3
```

In [19]:
```
        "There are two 10-element matroids in this class. They"
        "are not all exensions of a single matroids from U35e9,"
        "but all dual coextensions from 9 to 10 elements are permeating."

        for M in U35e9:
            for N in ExtendThisX7andU36Free(M):
                print Permeation(N,M)
            print
```

```
True

True

True
```