Matematički odjel
Prirodoslovno–matematički fakultet
Sveučilište u Zagrebu

Vedran Krčadinac

Konstrukcija konačnih ravnina u programskom sustavu *Mathematica*

Mentori:

Juraj Šiftar
Goranka Nogo

Zagreb, travanj 1996.

Sadržaj

1 Uvod

Predmet ovog rada je konstrukcija konačnih projektivnih i afinih ravnina pomoću računala. Korišteni alat je programski sustav *Mathematica*.

Projektivna geometrija razvila se kao samostalna disciplina početkom 19. stoljeća. Ljudi su prvo proučavali realnu projektivnu ravninu, zatim kompleksnu. Kasnije su došle na red projektivne ravnine nad proizvoljnim poljima i tijelima, kao i projektivni prostori više dimenzije. Javljaju se i pokušaji aksiomatizacije projektivne geometrije. Aksiomi su bili dobri za dimenzije veće od dva, u smislu da su definirali točno onu klasu objekata koja se do tada proučavala. Ali za dimenziju dva, tj. za projektivnu ravninu (vidi definiciju 1.2) nikom nije pošlo za rukom to dokazati.

Krajem 19. stoljeća pronađeni su prvi objekti koji zadovoljavaju aksiome, a nisu izomorfni niti jednom od standardnih modela projektivne ravnine (dobivene od tijela). To su bile prve nedesarguesovske ravnine. Početkom 20. stoljeća nađeni su i konačni primjeri. Teorija konačnih projektivnih ravnina počinje se intenzivnije razvijati oko 1950. godine. Do tada su konstruirani još mnogi drugi primjeri konačnih nedesarguesovskih ravnina.

U radu će najprije biti konstruirane klasične projektivne ravnine, nad konačnim poljima. U trećem poglavlju vidjet ćemo da je klasična konstrukcija u nekim slučajevima provediva i nad općenitijom strukturom, kvazipoljem. Konstruirat ćemo dvije velike klase kvazipolja, Hallovu i Andréovu. Tako ćemo dobiti niz nedesarguesovskih ravnina, koje pripadaju važnoj klasi translacijskih ravnina. Na kraju će biti provedena jedna druga konstrukcija projektivnih ravnina. Pomoću nje ćemo dobiti tzv. Hughesove ravnine, koje nisu desarguesovske, a nisu ni translacijske.

Recimo nekoliko riječi i o programskom alatu. *Mathematica* je sustav za računalnu algebru (Computer Algebra System, CAS). To su programi koji omogućuju egzaktno, simboličko računanje. Slični proizvodi su Macsyma, Maple, Axiom i nešto stariji Derive.

U siječnju 1995. godine potpisan je ugovor između Ministarstva znanosti i tehnologije Republike Hrvatske i Wolfram Research, Inc. o korištenju programskog sustava *Mathematica* na hrvatskim sveučilištima. Pri Matematičkom odjelu PMF-a Sveučilišta u Zagrebu osnovan je Referalni centar za *Mathematicu*, čiji sam član postao.

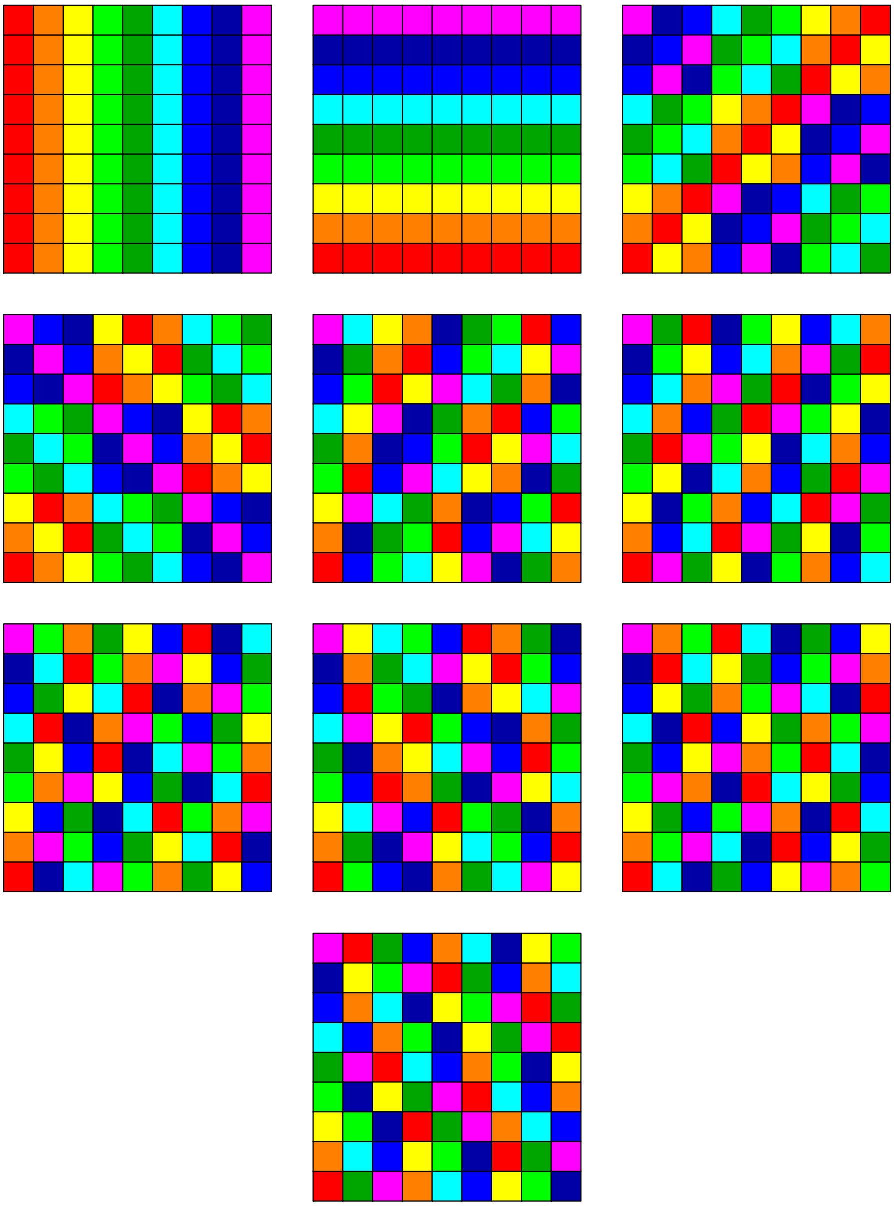
Mathematica je velik i vrlo moćan sustav. Njegova prednost nad klasičnim programskim jezicima je u visokom stupnju apstrakcije. Jedina struktura podataka koju *Mathematica* poznaje je izraz (expression), a sve što *Mathematica* radi je da izraze mijenja prema zadanim pravilima. Upravo zbog toga probleme koji bi npr. u Pascalu zahtjevali više petlji, grananja i slično u *Mathematici* riješavamo sa svega nekoliko definicija ili pravila. Te definicije često možemo upotrijebiti i općenitije nego što smo prvobitno zamislili, dok su klasični programi primjenjivi samo na onu strukturu podataka za koju su pisani.

Zbirke definicija i pravila koje proširuju osnovne mogućnosti *Mathematice* nazivaju se paketi. Wolfram Research stavlja korisnicima na raspolaganje velik izbor paketa s najrazličitijih područja, pod imenom *MathSource*. Svaka datoteka na *MathSourceu* ima jedinstveni broj. *MathSourceu* se pristupa preko globalne informatičke mreže Internet (URL adresa je *http://www.wri.com/mathsource/*), a kopiju te zbirke nudi i Referalni centar za *Mathematicu*, preko računala Matematičkog odjela (URL adresa *http://mrc.math.hr/mrc/mathsource/*).

U ovom radu bit će opisana četiri paketa: *FiniteFieldsV2.m*, *FinitePlanes.m*, *Quasifields.m* i *HughesPlanes.m*. Posljednja tri napisao sam samostalno, dok je prvi prilagođena verzija paketa *FiniteFields.m* (broj 0200-350) s *MathSourcea*. Pakete, kao i elektronsku verziju ovog rada i neke daljnje primjere može se naći na URL asdresi *http://student.math.hr/~krcko/radovi* .

Uz pomoć paketa napravljenih u sklopu ovog rada možemo dobiti velik broj međusobno neizomorfnih konačnih ravnina. Na primjer, moguće je konstruirati sve četiri projektivne ravnine reda devet: desarguesovsku, translacijsku, dualnu translacijsku i Hughesovu. "Brisanjem" pojedinih pravaca dobivamo svih sedam afinih ravnina reda devet.

2 Desarguesovske ravnine



Slika 1. Desarguesovska afina ravnina reda devet.

2.1 Projektivne i afine ravnine

Na početku ovog odjeljka definirat ćemo osnovne pojmove.

Definicija 1.1. **Incidencijska struktura** *je uređena trojka* (P, L, I)*, gdje su* P *i* L *neprazni skupovi, a* $I \subseteq P \times L$ *relacija.*

Elemente skupa P zovemo *točke* i označavamo velikim slovima $A, B, C \ldots$, a elemente skupa L *pravci* i označavamo malim slovima $a, b, c \ldots$ O incidenciji ćemo često govoriti geometrijskim rječnikom, npr. pravac prolazi točkom, točka leži na pravcu.

Definicija 1.2. **Projektivna ravnina** *je incidencijska struktura koja zadovoljava aksiome:*
(P1) *Svake dvije točke incidentne su s točno jednim pravcem.*
(P2) *Svaka dva pravca sijeku se u jedinstvenoj točki.*
(P3) *Postoje četiri točke od kojih nikoje tri ne leže na istom pravcu.*

Definicija 1.3. **Afina ravnina** *je incidencijska struktura koja zadovoljava aksiome:*
(A1) *Kroz svake dvije točke prolazi jedinstveni pravac.*
(A2) *Za svaki pravac* l *i točku* A *koja mu ne pripada postoji jedinstveni pravac* m *koji prolazi kroz* A *i ne siječe se s* l*.*
(A3) *Postoje tri točke koje ne leže na istom pravcu.*

Pravce ćemo često identificirati sa skupom točaka s kojima su incidentni. To znači da pravac l identificiramo sa skupom $\{X \in P \mid X \ I \ l\}$. U toj interpretaciji općenitu relaciju I zamjenjujemo relacijom pripadanja $\in$. Implementacija konačnih ravnina u *Mathematici* upravo je takva: pravci su skupovi točaka s kojima su incidentni. Najmanja projektivna ravnina, koja se sastoji od sedam točaka i sedam pravaca, u *Mathematici* se pamti kao lista

```
{ {2, 4, 6}, {1, 4, 5}, {3, 4, 7}, {1, 2, 3},
  {2, 5, 7}, {1, 6, 7}, {3, 5, 6} }
```

Točke su brojevi $1 \ldots 7$, a pravci skupovi točaka skupljeni u gornju listu.

Strukture čijom konstrukcijom se bavi ovaj rad opisane su definicijama 1.2 i 1.3. Iako definicije imaju smisla i za beskonačne skupove, bavimo se isključivo slučajem kada su P i L konačni. To je, uostalom, neophodno za implementaciju na računalu.

Teorem 1.1. *Neka je* $\mathcal{P}$ *konačna projektivna ravnina. Svaka dva pravca u* $\mathcal{P}$ *incidentna su s istim brojem točaka. Označimo taj broj s* $n+1$*. Onda je* *ukupan broj točaka ravnine $\mathcal{P}$ n^2+n+1, kroz svaku točku prolazi $n+1$ pravac, a ukupan broj pravaca je n^2+n+1.*

Neka je $\mathcal{A}$ afina ravnina. Svaka dva pravca u $\mathcal{A}$ incidentna su s istim brojem točaka. Označimo li taj broj s n, ukupan broj točaka ravnine $\mathcal{A}$ je n^2. Kroz svaku točku prolazi $n+1$ pravac, a ukupan broj pravaca je n^2+n.

U oba slučaja je $n \geq 2$, tj. svaki pravac u $\mathcal{P}$ incidentan je s bar tri, a u $\mathcal{A}$ s bar dvije točke.

Za projektivnu ravninu $\mathcal{P}$ i afinu ravninu $\mathcal{A}$ iz prethodnog teorema kažemo da su *reda* n. Sljedeći teorem govori o vezi projektivnih i afinih ravnina.

Teorem 1.2. *Neka je $\mathcal{P}=(P,L,I)$ projektivna ravnina reda n i $l_\infty \in L$ jedan njezin pravac. Struktura koju dobijemo kada iz L izbacimo l_∞, a iz P sve točke incidentne s l_∞ je afina ravnina reda n.*

Obrnuto, neka je $\mathcal{A}=(A,L,I)$ afina ravnina reda n. Pravce možemo razdijeliti u $n+1$ klasa od kojih svaka sadrži n međusobno disjunktnih (paralelnih) pravaca. Dodajmo svakom pravcu po jednu novu točku. Pravcima unutar jedne klase dodamo istu točku, a pravcima iz različitih klasa različite. Ako još skupu L dodamo novi pravac l_∞ incidentan sa svim novim točkama, dobit ćemo projektivnu ravninu reda n.

Tvrdnja 1.3. (Desarguesov teorem)
Neka su $O, A_1, B_1, C_1, A_2, B_2, C_2$ različite točke projektivne ravnine, a OA_1A_2, OB_1B_2, OC_1C_2 različiti pravci. Neka je C_3 sjecište pravaca A_1A_2 i B_1B_2, B_3 sjecište pravaca A_1A_2 i C_1C_2, te A_3 sjecište B_1B_2 sa C_1C_2. Onda su točke A_3, B_3 i C_3 kolinearne.

Tvrdnja 1.3 naziva se teorem samo iz povijesnih razloga. Ona ne slijedi iz aksioma **(P1)**–**(P3)**, tj. postoje projektivne ravnine u kojima ne vrijedi. Ravnine u kojima vrijedi tvrdnja 1.3 nazivamo *desarguesovskim*, a one u kojima ne vrijedi *nedesarguesovskim*. Desarguesovske su točno one konačne ravnine koje se mogu konstruirati pomoću polja, kako će to biti opisano.

2.2 Osnovno o konačnim poljima

Definicija 2.1. Polje *je skup F s dvije binarne operacije $+$ i $\cdot$ takve da vrijedi:*
(1) $(F,+)$ je Abelova grupa. Neutralni element označimo s 0.
(2) $(F\backslash\{0\},\cdot)$ je Abelova grupa. Neutralni element označimo s 1.
(3) Distributivnost: $a(b+c)=ab+ac$ i $(a+b)c=ac+bc$, za sve a,b,c iz F.
Ako je skup F konačan i ima n članova, kažemo da je polje **reda** *n.*

Kao što je rečeno, polja će nam trebati za konstrukciju desarguesovskih ravnina. Konačna polja zadovoljavaju ovaj vrlo važan teorem:

Teorem 2.1. *Konačno polje reda n postoji ako i samo ako je n potencija prostog broja. Svaka dva konačna polja istog reda su izomorfna.*

Jezgra *Mathematice* ne podržava konačna polja. Međutim, na *MathSourceu* pronašao sam dvije implementacije:

(1) Operations on Finite Fields (broj 0200-350), Ilan Vardi, 1989.

(2) Galois Field Package (broj 0205-894), Ryoh Fuji-Hara, 1993.

U ovom radu koristio sam paket (1) nadopunjen funkcijama `AllElements` i `SaveFF`. Kompletan listing tog paketa nalazi se u poglavlju 4.2.

Kada implementiramo algebarsku strukturu kao što je konačno polje u *Mathematici* ili nekom drugom sustavu za računalnu algebru, imamo tri mogućnosti:

(1) Definirati nove operacije, npr. `FFPlus` i `FFTimes` za zbrajanje i množenje u konačnom polju. Elementima polja proglasimo cijele brojeve $0, 1, 2\ldots$

(2) Zadržati postojeće operacije $+$ i $*$ i uvesti funkciju za promjenu algebre. Ako napišemo `ChangeAlgebra[FiniteField,11]` svi računi će se izvoditi nad konačnim poljem reda 11.

(3) Zadržati postojeće operacije, a konstante i varijable s kojima računamo označiti tako da *Mathematica* zna kojoj algebri pripadaju.

Oba spomenuta paketa koriste treći pristup. Njegova prednost je da u novoj algebarskoj strukturi možemo koristiti mnoge funkcije *Mathematice* koje su originalno pisane samo za realne ili kompleksne brojeve (funkcije za rješavanje jednadžbi, računanje determinante, množenje matrica i polinoma itd.)

Budući da je za konstrukciju konačnih ravnina presudna brzina, ja sam odabrao prvi pristup. Funkcija `SaveFF` koju sam dodao paketu *FiniteFields.m* računa tablice zbrajanja i množenja traženog konačnog polja i pomoću njih definira operacije `FFPlus` i `FFTimes`. Zatim te operacije sprema na disk pod zadanim imenom, da bi se mogle koristiti u paketu *FinitePlanes.m*.

2.3 Konstrukcija ravnina

Funkcije za konstrukciju projektivnih i afinih ravnina nalaze se u paketu *FinitePlanes.m*. Dvije glavne funkcije, `ProjectivePlane` i `AffinePlane` napisao sam na osnovu ovih teorema:

Teorem 3.1. *Neka je F konačno polje reda n, a V trodimenzionalni vektorski prostor nad F. Neka je P skup svih jednodimenzionalnih, a L skup svih* *dvodimenzionalnih potprostora od V. Ako za relaciju incidencije uzmemo inkluziju (sadržavanje) $\subseteq$, uređena trojka $(P, L, \subseteq)$ je projektivna ravnina reda n.*

Teorem 3.2. *Neka je F konačno polje reda n, a A dvodimenzionalni vektorski prostor nad F. Za L uzmimo skup svih jednodimenzionalnih linearnih mnogostrukosti u A, tj. translate jednodimenzionalnih potprostora. Onda je uz relaciju pripadanja $\in$ trojka $(A, L, \in)$ afina ravnina reda n.*

Da bi konstruirali ravninu reda n treba nam odgovarajuće konačno polje. Snimimo ga pomoću paketa *FiniteFieldsV2.m* na disk, kako je opisano u prethodnoj točki. Unutar paketa *FinitePlanes.m* možemo ga učitati s `LoadFF["`*ime*`"]`.

Prvo ću opisati konstrukciju projektivnih ravnina. Za vektorski prostor V uzimamo uređene trojke elemenata iz F s koordinatnim zbrajanjem. Funkciju koja nalazi jednodimenzionalne potprostore od V, točke projektivne ravnine, nazvao sam `Points`. Ona ne vraća cijele potprostore, nego samo vektore koji ih razapinju, što je dovoljno. Ako učitamo polje reda 3 dobit ćemo listu

```
{{0, 0, 1}, {0, 1, 0}, {0, 1, 1}, {0, 1, 2}, {1, 0, 0},
 {1, 0, 1}, {1, 0, 2}, {1, 1, 0}, {1, 1, 1}, {1, 1, 2},
 {1, 2, 0}, {1, 2, 1}, {1, 2, 2}}
```

Svaka od uređenih trojki unutar te liste razapinje jedan potprostor od V. Ovo je funkcija `Points`:

```
Points :=
  Join[{{0,0,1}},Table[{0,1,i},{i,0,FFOrder-1}],
       Flatten[Table[{1,i,j},{i,0,FFOrder-1},{j,0,FFOrder-1}],1]]
```

Funkcija vraća sve vektore kojima je prva nenul komponenta jednaka 1. Svaki jednodimenzionalni potprostor sadrži jedan i samo jedan takav vektor, pa je pronalaženje svih takvih vektora ekvivalentno pronalaženju svih potprostora.

Dvodimenzionalne potprostore ne moramo posebno tražiti, jer su u bijektivnom odnosu s jednodimenzionalnim. Potprostoru razapetom vektorom (a, b, c) pridružimo skup svih vektora (x, y, z) koji zadovoljavaju jednadžbu $ax + by + cz = 0$. Pokazuje se da je to dvodimenzionalni potprostor od V i da je opisano pridruživanje dobro definirana bijekcija. Za provjeru pripadnosti nekom dvodimenzionalnom potprostoru poslužit će nam funkcija

```
FFInner[l1_,l2_] := Inner[FFTimes,l1,l2,FFPlus]
```

Glavna funkcija `ProjectivePlane` je kratka.

```
ProjectivePlane :=
  Module[{po,p,x,i} , po = Points; Do[p[i] = po[[i]]; ,{i,1,Length[po]}];
       Table[Flatten[Position[po,x_List /; FFInner[p[i],x]==0,{1}]],
```

```
            {i,1,Length[po]}]
        ]
```

Algoritam je sljedeći: uzmemo jedan vektor v iz liste koju nam vrati `Points`. Pronađemo sve vektore x u toj listi koji zadovoljavaju jednadžbu `FFInner[`v`,`x`] = 0` (točnije, njihove položaje u listi). To je jedan pravac naše projektivne ravnine. Ponovimo li to za sve vektore v, dobit ćemo cijelu projektivnu ravninu.

Da smo tražili vektore koji zadovoljavaju jednadžbu `FFInner[`x`,`v`] = 0` dobili bi isti rezultat, jer je množenje u polju komutativno. Međutim, kasnije ćemo ovu konstrukciju provoditi nad kvazipoljima, a tamo nemamo uvijek komutativnost. U tom slučaju druga jednadžba će davati projektivnu ravninu dualnu onoj dobivenoj od prve. Zato sam definirao i funkciju

```
DualProjectivePlane :=
  Module[{po,p,x,i} , po = Points; Do[p[i] = po[[i]]; ,{i,1,Length[po]}];
         Table[Flatten[Position[po,x_List /; FFInner[x,p[i]]==0,{1}]],
            {i,1,Length[po]}]
        ]
```

Desarguesovske ravnine su samodualne, pa je logično da dvije funkcije daju isti rezultat za polja.

Pređimo sada na konstrukciju afinih ravnina. Za A uzmimo skup svih uređenih parova elemenata iz F organiziran u vektorski prostor na standardni način. Točke afine ravnine su jednostavno vektori iz A. Problem predstavlja pronalaženje jednodimenzionalnih linearnih mnogostrukosti. Kao što smo rekli, to su translati jednodimenzionalnih potprostora, njihovih *smjerova*. Smjerove nalazimo slično kao što smo prije nalazili točke projektivne ravnine.

```
RightSpan[v_] := Table[FFTimes[v,i],{i,0,FFOrder-1}]
RightDirections :=
  Map[RightSpan,Join[{{0,1}},Table[{1,i},{i,0,FFOrder-1}]]]
```

Algoritam se zasniva na istoj ideji. Razlika je (osim u dimenziji) u tome što nam trebaju cijeli potprostori, a ne samo vektori koji ih razapinju. Zato nam služi funkcija `RightSpan`. Kao kod projektivnih ravnina, i ovdje imamo "lijevu" verziju funkcija.

```
LeftSpan[v_] := Table[FFTimes[i,v],{i,0,FFOrder-1}]
LeftDirections :=
  Map[LeftSpan,Join[{{0,1}},Table[{1,i},{i,0,FFOrder-1}]]]
```

One daju isti rezultat primjenjene na polja, ali će nad kvazipoljima služiti za konstrukciju dualne ravnine.

Linearne mnogostrukosti, tj. pravce afine ravnine dobivamo translacijom smjerova. Ako uzmemo čvrsti smjer i translatiramo ga n puta dobivamo n međusobno disjunktnih (paralelnih) pravaca.

```
Parallels[dir_] :=
  If[MemberQ[dir,{0,1}] ,
     Table[ Map[FFPlus[#,{i,0}] & , dir] , {i,0,FFOrder-1}] ,
     Table[ Map[FFPlus[#,{0,i}] & , dir] , {i,0,FFOrder-1}]
    ]
```

Kada to ponovimo sa svim smjerovima, dobivamo afinu ravninu.

```
AffinePlane := Module[{d}, d = RightDirections;
  Table[Parallels[d[[j]]], {j, 1, FFOrder+1}]]

DualAffinePlane := Module[{d}, d = LeftDirections;
  Table[Parallels[d[[j]]], {j, 1, FFOrder+1}]]
```

Ovo je najmanja afina ravnina, dobivena od polja reda 2.

```
{{{{0, 0}, {0, 1}}, {{1, 0}, {1, 1}}},
 {{{0, 0}, {1, 0}}, {{0, 1}, {1, 1}}},
 {{{0, 0}, {1, 1}}, {{0, 1}, {1, 0}}}}
```

Ako želimo da točke budu prirodni brojevi, kao kod projektivnih ravnina, možemo primjeniti transformacijsko pravilo `Normalize`.

```
Normalize := {x_Integer,y_Integer} -> 1+x+FFOrder*y;
```

Dobijemo pregledniju listu

```
{{{1, 3}, {2, 4}}, {{1, 2}, {3, 4}}, {{1, 4}, {3, 2}}}
```

Vanjska lista sadrži pravce grupirane u klase međusobno paralelnih.

2.4 Analiza korištenih algoritama

Pomoću konstrukcija opisanih u prošloj točki teoretski možemo dobiti ravninu reda n za svaki broj n koji je potencija prostog broja (vidi teorem 2.1). U praksi se, međutim postavljaju ograničenja kao što su memorija računala na kojem izvodimo program i vrijeme. Svrha ove točke je da ustanovimo kako ta ograničenja utječu na red ravnina koje možemo dobiti.

Računala na kojima sam izvodio program su:

(1) Studentski server ("poslužitelj") Matematičkog odjela PMF-a (adresa na CARNetu je *student.math.hr*). To je Hewlett Packard 9000 serija 800 sa 160 Mb RAM-a i procesorom HP PA-RISC 7100LC na 100 MHz.

(2) Paralelno distribuirano računalo IBM SP/2 RISC Centra za napredno računarstvo na Fizičkom odjelu PMF-a. Računalo se sastoji od četiri čvora (*sp01.cnr.pmf.hr* – *sp04.cnr.pmf.hr*). Svaki čvor ima 128 Mb RAM-a i procesor Power 2 na 66.7 MHz.

Vidjeli smo da se projektivne ravnine u *Mathematici* pohranjuju kao liste pravaca. Pravci su liste koje se sastoje od cijelih brojeva, točaka ravnine. Projektivna ravnina reda n ima n^2+n+1 pravaca, od kojih svaki sadrži $n+1$ točaka. Za pamćenje jednog cijelog broja (tip `Integer`) u verziji 2.2 *Mathematice* potrebno je 12 bajtova. Zato za pohranjivanje svih cijelih brojeva u reprezentaciji projektivne ravnine treba

$$12(n^2+n+1)(n+1) = 12(n^3+2n^2+2n+1)$$

bajtova. Memorija se troši i na kontrolne strukture (liste). Iz tablice se vidi da u praksi za pamćenje ravnine reda n treba oko $16(n^3+2n^2+2n+1)$ bajtova, za velike n.

Za pamćenje afine ravnine odgovarajućeg reda treba više memorije. Točke su uređeni parovi cijelih brojeva. Za pamćenje jedne točke treba 48 bajtova. Svaki pravac sadrži n točaka, a pravaca ima $n(n+1)$. Ovdje samo za pohranjivanje svih točaka treba $48n^2(n+1)$ bajtova, a troši se i više memorije na kontrolne strukture, jer su liste dublje ugniježđene. Iz tablice se vidi da je potrošena memorija proporcionalna broju $n^2(n+1)$ s faktorom otprilike 52, za veće n.

Potrošnja memorije (u Kb) na HP 9000/800 i IBM SP/2						
Red	Projektivna ravnina			AfinaRavnina		
n	Veličina	$\frac{(n^3+2n^2+2n+1)}{1024}$	Omjer	Veličina	$\frac{n^2(n+1)}{1024}$	Omjer
2	0.48	0.02	23.43	0.80	0.01	68.33
11	27.55	1.56	17.68	76.56	1.42	53.99
19	126.52	7.44	17.00	374.47	7.05	53.11
31	515.91	31.03	16.63	1581.64	30.03	52.67
41	1164.39	70.67	16.47	3619.73	68.95	52.50
49	1962.73	119.68	16.40	6145.13	117.24	52.42

Ova tablica nam govori samo koliko je memorije potrebno za pohranjivanje gotovih ravnina. Prilikom izvođenja programa troši se još mnogo memorije za međurezultate i pomoćne strukture, kao što su operacije `FFPlus` i `FFTimes`. Na primjer, prilikom konstrukcije afine ravnine reda 49, koja sama zauzima oko 6 Mb, jezgra *Mathematice* poveća se na oko 10 Mb. Treba znati da jezgra inicijalno zauzima 3.5 Mb, a s učitanim paketom *FinitePlanes.m* i poljem reda 49 oko 5 Mb (podaci se odnose na HP 9000/800).

Obratimo sada pažnju na složenost algoritama. Želimo ocijeniti ponašanje funkcije f koja prirodnom broju n pridružuje vrijeme potrebno za konstrukciju ravnine reda n. Pri tom nam nisu bitne konkretne vrijednosti, jer one ovise o računalu na kojem se izvodi program. Za takvu ocjenu pogodna je notacija "veliko O".

Definicija 4.1. *Neka su $f, g\colon \mathbb{N} \to \mathbb{R}$ funkcije. Kažemo da je f* **reda naj-**

više g *i pišemo* $f(n) = O(g(n))$, *ako postoje* $n_0 \in \mathbb{N}$ *i* $c > 0$ *takvi da je* $|f(n)| \leq c \cdot |g(n)|$, *za sve* $n \geq n_0$.

Algoritam za konstrukciju projektivne ravnine sastoji se od dva dijela: pronalaženje točaka ravnine (jednodimenzionalnih potprostora od V) i utvrđivanje incidencije. Funkcija `Points` za nalaženje točaka sastoji se od jedne petlje duljine n i dvije ugniježđene petlje, svake duljine n. Njezina složenost je reda $O(n^2)$.

U drugom dijelu naizgled imamo samo jednu petlju duljine $n^2 + n + 1$. No pri svakom prolasku treba naći sve vektore predstavnike koji zadovoljavaju jednadžbu, što znači da još jednom treba proći kroz listu duljine $n^2 + n + 1$ (to radi funkcija `Position`). Stoga je složenost ovog dijela $O(n^4)$. To je ujedno i složenost ukupnog algoritma za konstrukciju projektivne ravnine.

Algoritam za afine ravnine također se sastoji od dva dijela: pronalaženje smjerova (funkcija `RightDirections`) i njihovo translatiranje. Prvi dio ima složenost $O(n^2)$. Vektora predstavnika ima $n + 1$, a svaki se razapinje pomoću petlje duljine n (funkcija `RightSpan`). Drugi dio ima složenost $O(n^3)$. Glavna funkcija `AffinePlane` $n+1$ puta poziva `Parallels`. Pri svakom pozivu izvršava se petlja duljine n unutar koje je operacija čije vrijeme izvođenja opet ovisi linearno o n: dodavanje konstante na svaku komponentu smjera. Dakle, tu je ukupna složenost $O(n^3)$.

Tablica pokazuje koliko vremena u sekundama treba za konstrukciju pojedinih ravnina na HP9000/800 i IBM SP/2.

Vrijeme izvođenja (u sekundama) na:				
Red	HP 9000 / 800		IBM SP/2	
n	`ProjectivePlane`	`AffinePlane`	`ProjectivePlane`	`AffinePlane`
2	0.05	0.02	0.04	0.02
11	27.14	1.25	17.24	0.83
19	403.39	13.33	227.97	7.37
31	5273.08	115.79	3096.28	71.99
41	23389.47	446.16	14571.24	276.48
49	–	–	39167.18	668.42

Podatke u tablici možemo pokušati modelirati funkcijom oblika $f(n) = C \cdot n^p$. Linearizacijom i primjenom diskretne metode najmanjih kvadrata dobivamo parametre C i p za koje f najbolje aproksimira podatke. Za projektivnu ravninu ispada $p \approx 4.29$ na HP 9000/800, odnosno $p \approx 4.31$ na IBM SP/2. To potvrđuje da je složenost otprilike $O(n^4)$. Za afinu ravninu dobivamo $p \approx 3.27$ na HP 9000/800 i $p \approx 3.13$ na IBM SP/2. I to su dovoljno dobre aproksimacije za složenost $O(n^3)$.

U sva četiri slučaja parametar p nešto je iznad očekivane vrijednosti, što možemo objasniti činjenicom da su računala pod višekorisničkim operacijskim sustavom UNIX. Što je dulje vrijeme izvođenja nekog procesa u UNIX-u, to je njegov prioritet sve manji. Osim toga s vremenom izvođenja raste i udio koji troši sam operacijski sustav, a smanjuje se dio namijenjen korisnom računanju.

Zaključak :
Vidimo da je za reprezentaciju afine ravnine potrebno oko tri puta više memorije nego za projektivnu ravninu istog reda. S vremenom izvođenja situacija je obrnuta: algoritam za afine ravnine ima složenost $O(n^3)$, a algoritam za projektivne $O(n^4)$. Za konstrukciju afine ravnine reda 49 treba oko 11 minuta procesorskog vremena, a za projektivnu čak 11 sati (podaci se odnose na IBM SP/2).

2.5 Ostale funkcije u paketu *FinitePlanes.m*

U ovoj točki ukratko ću opisati ostale funkcije u paketu *FinitePlanes.m*. Možemo ih podijeliti u nekoliko kategorija:

Pretvaranje projektivne ravnine u afinu i obrnuto.

`ProjectiveToAffine[` P, k `]`. Funkcija pretvara projektivnu ravninu P u afinu izbacivanjem k-tog po redu pravca i svih točaka incidentnih s njim.

`AffineToProjective[` A `]`. Pretvara afinu ravninu A u projektivnu tako da joj doda $n+1$ novih točaka i pravac incidentan s njima. Novi pravac nalazi se na kraju liste koja predstavlja dobivenu projektivnu ravninu. Ravnina A se prije transformacije normalizira (pomoću pravila `Normalize`).

Konstrukcija potpunih skupova ortogonalnih latinskih kvadrata.

`AffineToLatin[` A `]`. Pretvara afinu ravninu A (u nenormaliziranom obliku!) u skup međusobno ortogonalnih latinskih kvadrata.

`LatinSquares`. Vraća potpun skup ortogonalnih latinskih kvadrata, tako da primjeni `AffineToLatin` na rezultat od `AffinePlane`.

`OrthoTest[` $MOLS$ `]`. Provjerava da li su kvadrati u skupu $MOLS$ međusobno ortogonalni. Uspoređuje ih dva po dva i vraća `True` ako jesu, a `False` ako nisu. Dobivamo listu logičkih konstanti, koje su sve `True` ako se radi o međusobno ortogonalnim kvadratima.

Vizualizacija afinih ravnina i latinskih kvadrata.

`Paint[` A `]`, `Paint[` $MOLS$ `]`. Crta sliku koja predstavlja afinu ravninu A, odnosno skup latinskih kvadrata $MOLS$. Svaka klasa paralelnih pravaca ravnine A prikazuje se kao šareni kvadrat. Točke koje pripadaju jednom pravcu obojane su istom bojom. Latinski kvadrati se prikazuju položeni jedni na druge, da se vidi ortogonalnost. Slike u ovom radu dobivene su pomoću `Paint`.

Incidencijske matrice.

`IncidenceMatrix[` P `]`, `IncidenceMatrix[` A `]`. Funkcija vraća matricu incidencije projektivne ravnine P, odnosno afine ravnine A. A može biti normalizirana ili u izvornom obliku.

`TestMatrix`. To je kvadratna matrica reda n koja na dijagonali ima broj $n+1$, a izvan dijagonale jedinice (n je red trenutno učitanog konačnog polja). Ona služi za provjeru naše konstrukcije. Incidencijska struktura je projektivna ravnina ako i samo ako njezina incidencijska matrica A zadovoljava $A \cdot A^T =$ `TestMatrix`.

Spajanje točaka i presjeci pravaca.

`Vee[`$ravnina$`,`A`,`B`]`. Vraća redni broj pravca koji prolazi točkama A, B.

`Wedge[`$ravnina$`,`a`,`b`]`. Vraća točku u kojoj se sijeku pravci a i b.

3 Nedesarguesovske ravnine

3.1 Osnovno o kvazipoljima

U prethodnom poglavlju objasnili smo kako se od konačnog polja može dobiti projektivna i afina ravnina. Sada ćemo primjeniti tu konstrukciju na općenitiju strukturu, kvazipolje.

Definicija 1.1. *Neka je G neprazan skup, a $\cdot : G \times G \to G$ binarna operacija. Uređen par $(G, \cdot)$ zovemo* **kvazigrupa** *ako jednadžbe $ax = b$ i $ya = b$ imaju jedinstvena rješenja $x, y \in G$ za sve $a, b \in G$. Kvazigrupa u kojoj postoji neutralni element $1 \in G$ takav da je $1 \cdot x = x \cdot 1 = x$, $\forall x \in G$ zovemo* **petlja**.

Definicija 1.2. **Slabo kvazipolje** *je skup Q s dvije binarne operacije $+$ i $\cdot$ takve da vrijedi:*
(1) $(Q, +)$ je Abelova grupa. Neutralni element označimo s 0.
(2) $(Q \backslash \{0\}, \cdot)$ je petlja. Neutralni element označimo s 1.
(3) Lijeva distributivnost: $x(y + z) = xy + xz$, za sve $x, y, z \in Q$.
(4) $0 \cdot x = x \cdot 0 = 0$, za sve $x \in Q$.
Ako vrijedi još i aksiom
(5) Jednadžba $ax = bx + c$ ima jedinstveno rješenje $x \in Q$ za sve $a, b, c \in Q$, $a \neq b$,
onda uređenu trojku $(Q, +, \cdot)$ zovemo **kvazipolje**.

Očito je svako kvazipolje ujedno i slabo kvazipolje. Ako je skup Q konačan (a taj nas slučaj zanima) vrijedi i obrat.

Definicija 1.3. **Jezgra** *slabog kvazipolja Q je skup svih $k \in Q$ koji zadovoljavaju $(x + y)k = xk + yk$ i $(xy) \cdot k = x \cdot (yk)$, za sve $x, y \in Q$.*

Definicija 1.4. *Kvazipolje Q u kojem je množenje asocijativno, tj. $x \cdot (yz) = (xy) \cdot z$, $\forall x, y, z \in Q$ zovemo* **skoropolje**. *Kvazipolje u kojem vrijedi desna distributivnost $(x + y)z = xz + yz$, $\forall x, y, z \in Q$ zovemo* **prsten s dijeljenjem**. *Ako su ispunjena oba zahtjeva kažemo da je Q* **tijelo**.

Tijelo u kojem je množenje komutativno je polje. Naša konstrukcija projektivnih i afinih ravnina započinjala je od polja F. Ravnine koje smo dobili bile su desarguesovske.

Sljedeći korak je da funkcije `AffinePlane` i `ProjectivePlane` pokušamo primjeniti na kvazipolja. Realizirao sam dvije konstrukcije kvazipolja u *Mathematici*, Hallovu i Andréovu. Paket sam nazvao *Quasifields.m*. Pomoću njega možemo dobiti niz kvazipolja i snimiti ih u datoteke. Možemo ih s `LoadFF` učitati u paket *FinitePlanes.m* i od njih pokušati graditi ravnine.

Tek kod kvazipolja koja nisu polja dolazi do izražaja razlika između `ProjectivePlane` i `DualProjectivePlane`, te `AffinePlane` i `DualAffinePlane`. `ProjectivePlane` i `AffinePlane` konstruiraju translacijske ravnine, a `DualProjectivePlane` i `DualAffinePlane` dualne translacijske.

S funkcijama za konstrukciju projektivnih ravnina treba biti oprezan. Primjenjene na kvazipolja, one općenito neće dati projektivne ravnine. Projektivne ravnine dobivamo za asocijativna kvazipolja, tj. skoropolja.

Ipak i od neasocijativnih kvazipolja možemo dobiti projektivne ravnine, ako primjenimo `AffineToProjective` na rezultat od `AffinePlane`, odnosno `DualAffinePlane`. Te dvije funkcije konstruiraju afine ravnine od svih kvazipolja, ne samo asocijativnih.

Ravnine koje dobivamo od kvazipolja koja nisu polja su nedesarguesovske. One su manje istražene od desarguesovskih. Svaka konačna desarguesovska ravnina izomorfna je nekoj koja je dobivena opisanom konstrukcijom od konačnog polja. Zato teorem 2.2.1 rješava problem egzistencije i klasifikacije konačnih desarguesovskih ravnina. Za nedesarguesovske ravnine taj je problem velikim dijelom otvoren.

Na kraju poglavlja konstruirat ćemo jednu klasu nedesarguesovskih ravnina koje se ne mogu dobiti na opisani način, ravnine Hughesa. To je potpuno nova konstrukcija, koja ne koristi funkcije `ProjectivePlane` i `AffinePlane`. Sve potrebne definicije nalaze se u paketu *HughesPlanes.m*.

3.2 Hallova kvazipolja i ravnine

Teorem 2.1 *Neka je F konačno polje, a $f(v) = v^2 - av - b$ ireducibilni kvadratni polinom nad F. Neka je $H = F \times F$ dvodimenzionalni vektorski prostor nad F. U H definiramo množenje vektora:*

$$(x, 0) \cdot (z, t) = (xz, xt)$$

$$(x, y) \cdot (z, t) = (xz - y^{-1} t f(x), yz - xt + at), \text{ za } y \neq 0$$

Onda je H (uz uobičajeno zbrajanje vektora) kvazipolje. Daljnja svojstva od H su:
(1) H je polje ako i samo ako je F polje reda 2.
(2) H je asocijativno (tj. skoropolje) ako i samo ako je F polje reda 2 ili F polje reda 3 i $f(v) = v^2 + 1$.

Vektore iz H implementirat ćemo u obliku `Hall[ {x,y} ]`, gdje su x i y elementi konačnog polja F. Zbrajanje definiramo koordinatno:

```
Hall /: Hall[a_List] + Hall[b_List] := Hall[ a + b ]
```

Za množenje koristimo operator ** (NonCommutativeMultiply) kojem smo s

```
Unprotect[NonCommutativeMultiply]
ClearAttributes[NonCommutativeMultiply,Flat]
```

uklonili ugrađeno svojstvo asocijativnosti. Definiciju množenja možemo gotovo doslovno prepisati iz teorema.

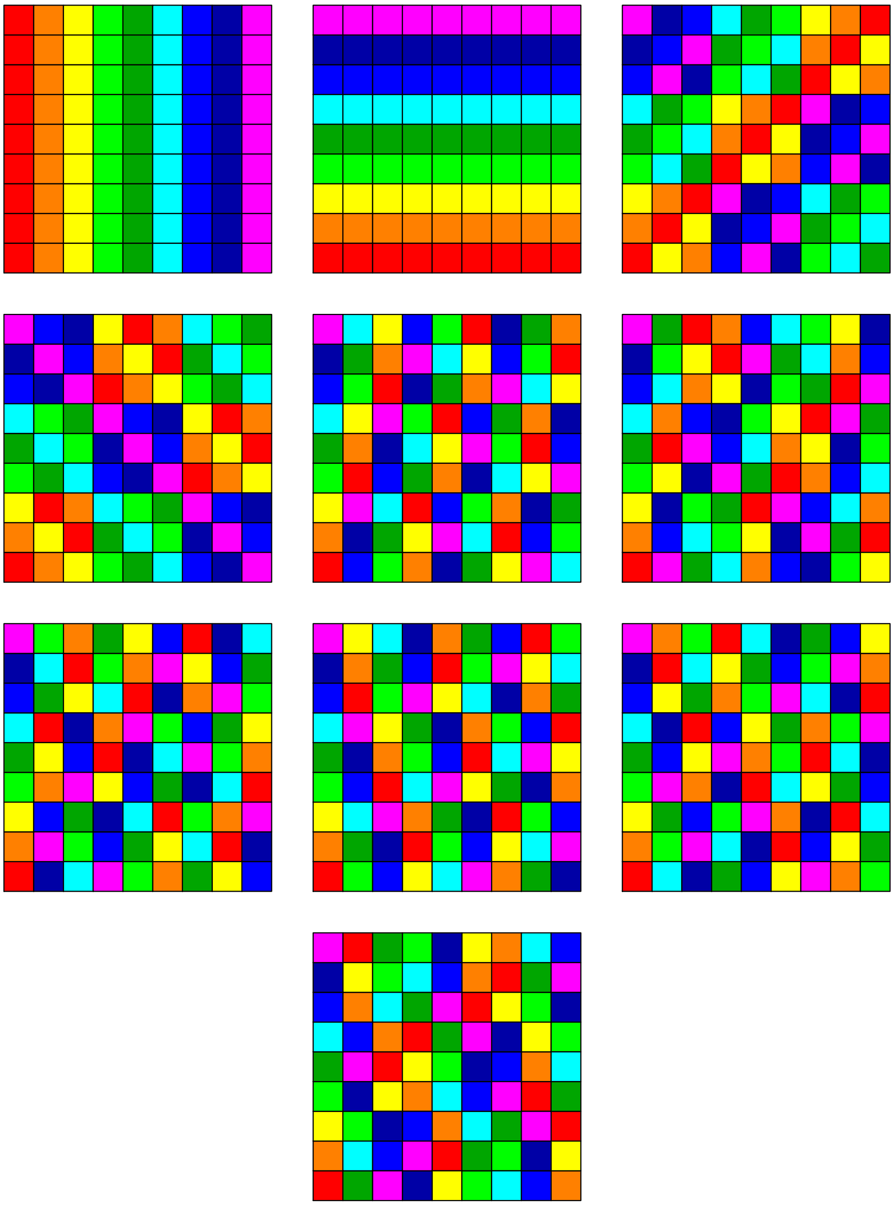
```
Hall /: Hall[{x_FiniteField , FiniteField[0,v_,{p_,n_}] }] **
        Hall[{z_FiniteField , t_FiniteField }]  :=  Hall[{x*z,x*t}]

Hall /: Hall[{x_FiniteField , FiniteField[y_,v_,{p_,n_}] }] **
        Hall[{z_FiniteField , t_FiniteField}]   :=
        Hall[{x*z-(FiniteField[y,v,{p,n}]^(-1))*t*
        IrreducibleQuadratic[x,{p,n}] ,
        FiniteField[y,v,{p,n}]*z - x*t + a[{p,n}]*t }]
```

Da olakšam pronalaženje ireducibilnih kvadratnih polinoma definirao sam funkciju `IrreducibleQuadratic`. Ona pamti prvi ireducibilni kvadratni polinom na kojeg naiđe ispitujući sve mogućnosti za koeficijente a i b. Provjera ireducibilnosti zasniva se na činjenici da je kvadratni polinom nad F ireducibilan ako i samo ako nema nultočku u F.

Sve ireducibilne kvadratne polinome dobivamo s `AllIrreducibleQuadratics`. Nad poljem reda tri ima ih 3, nad poljem reda četiri 6, a nad poljem reda pet 10. Onaj koji želimo možemo i ručno pridružiti funkciji `IrreducibleQuadratic` u odgovarajućem argumentu.

Operacije u Hallovom kvazipolju nad poljem reda p^n snimamo u datoteku sa `SaveHall[`$\{p,n\}$`,"`*ime*`"]`. Zatim ih možemo učitati u paket *FinitePlanes.m* (pomoću `LoadFF`) i konstruirati Hallove konačne ravnine. Ovdje vidimo sliku afine Hallove ravnine reda 9. Konstruirana je pomoću skoropolja dobivenog od polja reda 3 i polinoma $f(v) = v^2 + 1$.



Slika 2. Afina ravnina reda devet dobivena od Hallovog skoropolja.

3.3 Andréova kvazipolja i ravnine

Neka je p prost, s, d prirodni brojevi. Uvedimo još oznake $n = s \cdot d$, $q = p^s$, te $(F, +, \cdot)$ za konačno polje reda p^n. Kao što znamo, ono postoji i jedinstveno je do na izomorfizam (teorem 2.1). Osim toga, F ima jedno i samo jedno potpolje reda q. Označavat ćemo ga s K.

Konstruirat ćemo kvazipolje tako da "promijenimo" množenje u F. Novo množenje, koje ćemo označavati s `**`, bit će lijevo distributivno prema zbrajanju, ali neće uvijek zadovoljavati desnu distributivnost i asocijativnost. Potpolje K bit će sadržano u jezgri tako dobivenog kvazipolja.

Pokazuje se da množenje u konačnim poljima ima jednostavnu strukturu: $(F\backslash\{0\}, \cdot)$ je ciklička grupa reda $p^n - 1$. Označimo s ω jedan njezin generator. To je tzv. *primitivni korijen* u F.

Još jedna ciklička grupa igra važnu ulogu u našoj konstrukciji – grupa automorfizama polja F, $AutF$. Nama će trebati oni automorfizmi koji fiksiraju elemente potpolja K. Skup svih takvih automorfizama čini podgrupu, koju označavamo s $Aut_K F$. Ona je također ciklička – generirana je preslikavanjem $\gamma: F \to F$, $\gamma(x) = x^q$. Njezin red je $d = dim_K F$. To slijedi iz temeljnog teorema Galoisove teorije, koji povezuje podgrupe od $AutF$ s potpoljima od F.

Grupu $Aut_K F$ na prirodan način identificiramo s aditivnom grupom cijelih brojeva modulo d, $(\mathbb{Z}_d, +)$. Izomorfizam je $\Gamma: \mathbb{Z}_d \to Aut_k F$, $\Gamma(k) = \gamma^k$.

Teorem 3.1. *Neka je preslikavanje $\nu: F\backslash\{0\} \to \mathbb{Z}_{q-1}$ zadano s $\nu(\omega^i) = i$ $(mod\ q-1)$, $i \in \{0, 1, \ldots, p^n - 2\}$. Neka je $\Phi: \mathbb{Z}_{q-1} \to \mathbb{Z}_d$ bilo koja funkcija sa svojstvom $\Phi(0) = 0$. Označimo s $\alpha = \Gamma \circ \Phi \circ \nu: F\backslash\{0\} \to Aut_K F$ kompoziciju funkcija ν, Φ i Γ. Definiramo novo množenje elemenata iz F:*

$$0 * *y = 0$$

$$x * *y = x \cdot [\alpha(x)](y), \text{ za sve } x, y \in F,\ x \neq 0$$

Onda je $(F, +, **)$ kvazipolje. Ono zadovoljava:

(1) *asocijativnost, ako i samo ako je $\Phi: \mathbb{Z}_{q-1} \to \mathbb{Z}_d$ homomorfizam aditivnih grupa*

(2) *desnu distributivnost, ako i samo ako je Φ konstantno preslikavanje, $\Phi(k) = 0$, $\forall k \in \mathbb{Z}_{q-1}$. U tom slučaju novo množenje $**$ podudara se sa starim $\cdot$, pa smo dobili polje.*

U ovom teoremu po volji smo mogli izabrati p, s, d, te funkciju Φ. Ako su p, s, d zadani, Φ možemo izabrati na d^{q-2} načina.

Za opis implementacije Andréovih kvazipolja u *Mathematici* treba znati osnovno o paketu *FiniteFieldsV2.m*. Elementi konačnog polja su oblika

`FiniteField[` f, x, $\{p,n\}$ `]`

Pri tom je f polinom nad $\mathbb{Z}_p$ u varijabli x, a p^n je red polja o kojem se radi. Dva takva objekta zbrajaju se tako da se zbroje odgovarajući polinomi (naravno, modulo p). Množenje se vrši modulo nekog ireducibilnog polinoma stupnja n nad $\mathbb{Z}_p$.

Za Andréova kvazipolja uveo sam slične objekte:

$$\texttt{Andre[}\ f,\ \{p,s,d\}\ \texttt{]}$$

Polinom f uvijek je u varijabli v. Simbol v sam zaštitio. Njemu se ne smije pridružiti neka vrijednost tokom konstrukcije kvazipolja. Objekt pripada polju reda $p^{s\cdot d}$ s istaknutim potpoljem K reda p^s. Zbrajanje i množenje uveo sam kao i za `FiniteField` objekte.

Da bi definirali novo množenje $**$, treba nam najprije funkcija ν.

```
Norm[ Andre[f_,{p_,s_,d_}] ] :=
  Mod[Position[CyclicOrder[{p,s,d}] , Andre[f,{p,s,d}]]
      [[1,1]] -1, p^s-1]
```

Funkciju Φ treba odabrati sam korisnik i pridružiti je simbolu `Phi[`$\{p,s,d\}$`]`. Sve mogućnosti dobivamo s `AllPhis`.

```
MakeFunction = Function[i, #1[[i + 1]]] &
AllPhis1[{2,1,d_}] = {{0}}
AllPhis1[{p_, s_, d_}] :=
  Flatten[Array[{1,##1} - 1 & , Table[d, {p^s - 2}]],p^s-3]
AllPhis[{p_,s_,d_}] :=
 Map[MakeFunction,AllPhis1[{p,s,d}]]
```

Ako korisnik to ne učini, *Mathematica* će na slučajan način odabrati jednu od mogućih funkcija Φ.

```
Phi[{p_,s_,d_}] := Phi[{p,s,d}] =
 AllPhis[{p,s,d}][[Random[Integer,{1,Length[AllPhis[{p,s,d}]]}]]]
```

Treba nam i izomorfizam Γ između $\mathbb{Z}_d$ i $Aut_K F$. Budući da je simbol `Gamma` u *Mathematici* rezerviran, dao sam mu ime `Aut`.

```
Aut[i_Integer, {p_,s_,d_}] :=
  Function[x,x^(p^(s*i))]
```

Funkcija α je kompozicija Γ, Φ i ν.

```
Alpha[ Andre[f_,{p_,s_,d_}] ] :=
  Aut[ Phi[{p,s,d}] [ Norm[Andre[f,{p,s,d}]] ], {p,s,d}]
```

Sada možemo definirati množenje u Andréovom kvazipolju.

```
Andre /: Andre[0,{p_,s_,d_}] ** Andre[y_,{p_,s_,d_}] :=
 Andre[0,{p,s,d}]

Andre /: Andre[x_,{p_,s_,d_}] ** Andre[y_,{p_,s_,d_}] :=
 Andre[x,{p,s,d}] * Alpha[Andre[x,{p,s,d}]][Andre[y,{p,s,d}]]
```

Definirao sam i funkciju `SaveAndre` koja snima operacije u datoteku pod zadanim imenom (analogno kao `SaveFF` i `SaveHall`). Sintaksa je `SaveAndre[` $\{p, s, d\}$`, "`*ime*`" ].`

Pomoću ove konstrukcije možemo dobiti skoropolje reda q^2 za svaku neparnu potenciju prostog broja q. Treba samo $\Phi: \mathbb{Z}_{q-1} \to \mathbb{Z}_2$ definirati kao $\Phi(k) = k(mod\ 2)$. To je homomorfizam aditivnih grupa, pa će dobiveno kvazipolje biti asocijativno. Ovo je važan slučaj, jer će nam takva skoropolja trebati za konstrukciju ravnina Hughesa.

Najmanji primjer Andréovog kvazipolja je skoropolje reda 9. Ovo su tablice zbrajanja i množenja u tom skoropolju:

+	0	1	2	3	4	5	6	7	8
0	0	1	2	3	4	5	6	7	8
1	1	2	0	4	5	3	7	8	6
2	2	0	1	5	3	4	8	6	7
3	3	4	5	6	7	8	0	1	2
4	4	5	3	7	8	6	1	2	0
5	5	3	4	8	6	7	2	0	1
6	6	7	8	0	1	2	3	4	5
7	7	8	6	1	2	0	4	5	3
8	8	6	7	2	0	1	5	3	4

**	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8
2	0	2	1	6	8	7	3	5	4
3	0	3	6	2	5	8	1	4	7
4	0	4	8	7	2	3	5	6	1
5	0	5	7	4	6	2	8	1	3
6	0	6	3	1	7	4	2	8	5
7	0	7	5	8	3	1	4	2	6
8	0	8	4	5	1	6	7	3	2

Ono se podudara s Hallovim skoropoljem reda 9, dobivenim od polja reda 3 i ireducibilnog polinoma $f(v) = v^2 + 1$. Pomoću njega je dobivena ravnina na prethodnoj slici.

Sljedeći primjer su Andréova kvazipolja reda 25. Funkciju Φ možemo definirati na $2^{5-2} = 2^3 = 8$ načina. Jedan od njih je konstanta 0, što nam daje konačno polje reda 25. Druga mogućnost je $\Phi(k) = k\ (mod\ 2)$, $k = 0, 1, 2, 3$. Tako dobivamo skoropolje. Preostalih 6 funkcija Φ definiraju neasocijativna kvazipolja.

3.4 Ravnine Hughesa

Vidjeli smo da primjenom Andréove konstrukcije možemo dobiti skoropolje reda q^2 za svaku neparnu potenciju prostog broja q. Takva skoropolja imaju svojstvo da im je jezgra polje reda q i da elementi jezgre komutiraju sa svim ostalim elementima. Zvat ćemo ih *Hughesova skoropolja.*

Neka je N Hughesovo skoropolje reda q^2, a F njegova jezgra. Za konstrukciju Hughesovih ravnina trebat će nam još i regularna matrica A reda tri nad poljem F, takva da je A^{q^2+q+1} skalarna (tj. oblika kI za neki $k \in F$) i da niti jedna niža potencija od A nema to svojstvo. Pokazuje se da takva matrica uvijek postoji.

U konstrukciji Hughesovih ravnina donekle ćemo slijediti uobičajenu konstrukciju pomoću vektorskih prostora. Naš će "vektorski prostor" biti definiran nad skoropoljem N. Točke će biti zadane kao "jednodimenzionalni potprostori", a pravci kao slike određenih skupova točaka pod djelovanjem grupe $\langle A \rangle$.

Teorem 4.1. *Neka je $V = N \times N \times N$ kartezijev produkt skupova. Operacije zbrajanja i množenja skalarom uvodimo na standardni način:*

$$(x_1, x_2, x_3) + (y_1, y_2, y_3) = (x_1 + y_1, x_2 + y_2, x_3 + y_3)$$

$$k(x_1, x_2, x_3) = (kx_1, kx_2, kx_3)$$

Točke ravnine $\mathcal{H}$ neka su skupovi oblika $\langle \mathbf{x} \rangle = \{k\mathbf{x} \mid k \in N\}$, za $\mathbf{x} = (x_1, x_2, x_3) \in V \backslash \{(0,0,0)\}$. Označimo s $L(t)$ skup onih točaka u $\mathcal{H}$ čiji vektori (x_1, x_2, x_3) zadovoljavaju jednadžbu $x_1 + x_2 t + x_3 = 0$. Za pravce ravnine $\mathcal{H}$ uzimamo skupove oblika $A^m L(t) = \{\langle A^m \mathbf{x} \rangle \mid \langle \mathbf{x} \rangle \in L(t)\}$, pri čemu je $0 \le m \le q^2 + q$, $t \in \{1\} \bigcup (N \backslash F)$.

Tako definiran $\mathcal{H}$ je uz inkluziju $\subseteq$ kao relaciju incidencije projektivna ravnina reda q^2.

Najveći dio posla oko implementacije ove konstrukcije obavio sam u paketu *Quasifields.m*. Tamo sam definirao funkcije koje grade skoropolje N, izdvajaju njegovu jezgru F i nalaze matricu A i njezine potencije. Sve to skupa treba snimiti u datoteku, zajedno s funkcijom za pronalaženje lijevih multiplikativnih inverza u N.

Za korisnika je jedina važna funkcija `SaveHughes[{`p, s`},"`*ime*`"]`. Ona obavlja cijeli posao za Hughesovo skoropolje reda q^2, $q = p^s$. Operacije u snimljenom skoropolju zovu se `FFPlus` i `FFTimes`, kao i prije. Funkcija za pronalaženje lijevih inverza snimit će se pod imenom `FFInverse`, a A i njezine potencije kao `A[0]`, `A[1]`, $\ldots$, `A[`q^2+q`]`.

Opisat ću ipak ukratko kako se dobiva matrica A. Ona predstavlja regularni linearni operator na trodimenzionalnom vektorskom prostoru W nad F. Za W uzimamo konačno polje reda q^3 čije smo jedinstveno potpolje reda q identificirali s F. Operator čija je matrica A zapravo je množenje primitivnim korijenom ω

iz W: $Ax = \omega \cdot x, \forall x \in W$. Može se vidjeti da je taj operator linearan, regularan i da ima traženo svojstvo.

Prostor W realizirao sam kao polinome stupnja najviše tri nad F. Na taj način izbjegnut je problem smještenja F u W. Množenje polinoma iz W vrši se modulo nekog ireducibilnog kubičnog polinoma nad F, da bi dobili polje. Trebalo je riješiti i niz drugih problema, kao što je pronalaženje ireducibilnih kubičnih polinoma, nalaženje primitivnog korijena u W, dobivanje matrice našeg linearnog operatora itd.

Nastojao sam da za korisnika paket *HughesPlanes.m* bude što sličniji paketu *FinitePlanes.m*. To nažalost znači da se ne mogu oba koristiti istovremeno. Funkcija `LoadFF` ovdje služi da bi učitali skoropolje s kojim započinjemo konstrukciju i ostale potrebne objekte, potencije matrice A i funkciju `FFInverse`. Funkcija koja konstruira ravninu zove se `HughesPlane`. Ukratko ću opisati kako radi.

Problem pronalaženja točaka ravnine isti je kao prije, pa sam funkciju `Points` mogao prepisati iz paketa *FinitePlanes.m*. Sjetimo se da za svaki jednodimenzionalni potprostor dobivamo samo po jedan vektor, koji ga razapinje. Skupove $L(t)$ dobivamo vrlo jednostavno. Na ovom primjeru vidi se snaga *Mathematice* kao programskog jezika visokog nivoa.

```
L[t_] := L[t] =
  Cases[Points[FFOrder],{x_,y_,z_} /; FFPlus[x,FFTimes[y,t],z] == 0]
```

Morao sam definirati novu funkciju za množenje matrice s vektorom, jer ugrađena ne radi s našim operacijama `FFTimes` i `FFPlus`.

```
FFDot[M_, v_] :=
  Table[Inner[FFTimes, M[[i]], v, FFPlus], {i,1,First[Dimensions[M]]}]
```

Kada pomnožimo vektor x iz liste P koju nam vrati `Points` s potencijom matrice A, može se desiti da rezultat $y = A^k x$ ne bude sadržan u P. Trebali bi pronaći onaj vektor iz P koji razapinje potprostor kojem pripada x. Problem zvuči komplicirano, ali ima jednostavno rješenje. Svi vektori iz liste P imaju prvu nenul komponentu s lijeva jednaku 1. Dovoljno je da i vektor y normaliziramo tako da njegova prva nenul komponenta bude 1. Onda se on mora podudarati s nekim vektorom iz P.

```
  Norm[{0, 0, x_}] := {0, 0, 1}
  Norm[{0, x_, y_}] := {0, 1, FFTimes[FFInverse[x], y]}
  Norm[{x_, y_, z_}] :=
    {1, FFTimes[FFInverse[x], y], FFTimes[FFInverse[x], z]}
```

Ovdje su nam trebali lijevi multiplikativni inverzi (funkcija `FFInverse`). U paketu *FinitePlanes.m* nije se pojavio taj problem, pa je tamo nismo imali. Konačno, evo i funkcije `HughesPlane`.

```
HughesPlane := Map[Norm,
  Join[Table[Map[FFDot[A[m],#] &, L[1]], {m,0,FFOrder+Sqrt[FFOrder]}],
       Flatten[Table[ Map[FFDot[A[m],#] &, L[t]], {m,0,FFOrder+Sqrt[FFOrder]},
               {t,Sqrt[FFOrder],FFOrder-1} ] , 1]
      ], {2}        ]
```

Rezultat koji ćemo dobiti prilično je nepregledan, jer su točke uređene trojke. Sličan problem imali smo kod `AffinePlane`. Rješenje opet leži u normalizacijskom pravilu koje trojke pretvara u prirodne brojeve.

```
Normalize :=
  {x_Integer,y_Integer,z_Integer} :> Position[Points[FFOrder],{x,y,z}][[1,1]]
```

Kao primjer navodim Hughesovu ravninu reda devet u normaliziranom obliku. Točke su brojevi 1,2,...,91, a pravci ovi skupovi točaka:

```
{{4, 13, 21, 29, 46, 54, 62, 70, 78, 86}, {29, 1, 31, 30, 32, 33, 34, 35, 36, 37},
 {30, 2, 12, 21, 84, 75, 66, 57, 48, 39}, {21, 11, 3, 31, 91, 81, 71, 61, 51, 41},
 {31, 4, 20, 12, 64, 45, 53, 88, 69, 77}, {12, 29, 22, 3, 52, 59, 42, 82, 89, 72},
 {3, 30, 13, 20, 60, 43, 50, 90, 73, 80}, {20, 21, 1, 22, 25, 23, 24, 28, 26, 27},
 {22, 31, 2, 13, 40, 49, 58, 67, 76, 85}, {13, 12, 11, 1, 14, 15, 16, 17, 18, 19},
 {1, 3, 4, 2, 6, 7, 5, 9, 10, 8},         {2, 20, 29, 11, 47, 56, 38, 74, 83, 65},
 {11, 22, 30, 4, 87, 79, 68, 63, 55, 44}, {8, 13, 28, 34, 38, 51, 63, 66, 77, 89},
 {10, 13, 27, 32, 44, 47, 61, 69, 82, 84}, {9, 13, 26, 33, 41, 55, 56, 72, 75, 88},
 {5, 13, 25, 37, 39, 53, 59, 65, 81, 87}, {7, 13, 24, 35, 45, 52, 57, 68, 74, 91},
 {6, 13, 23, 36, 42, 48, 64, 71, 79, 83}, {65, 1, 67, 66, 68, 69, 70, 71, 72, 73},
 {83, 1, 85, 84, 86, 87, 88, 89, 90, 91}, {74, 1, 76, 75, 77, 78, 79, 80, 81, 82},
 {38, 1, 40, 39, 41, 42, 43, 44, 45, 46}, {56, 1, 58, 57, 59, 60, 61, 62, 63, 64},
 {47, 1, 49, 48, 50, 51, 52, 53, 54, 55}, {44, 2, 17, 71, 62, 89, 35, 53, 80, 26},
 {55, 2, 19, 91, 37, 46, 82, 73, 28, 64}, {63, 2, 18, 81, 72, 36, 54, 27, 45, 90},
 {68, 2, 14, 41, 77, 50, 23, 86, 59, 32}, {79, 2, 16, 61, 43, 25, 88, 34, 70, 52},
 {87, 2, 15, 51, 24, 69, 60, 42, 33, 78}, {86, 11, 9, 53, 34, 73, 57, 42, 27, 76},
 {78, 11, 8, 64, 39, 32, 90, 26, 67, 52}, {70, 11, 10, 45, 80, 48, 33, 85, 59, 28},
 {62, 11, 6, 77, 72, 24, 49, 37, 43, 84}, {54, 11, 5, 88, 23, 40, 82, 66, 35, 60},
 {46, 11, 7, 69, 58, 89, 25, 50, 75, 36}, {37, 4, 74, 42, 66, 26, 61, 50, 85, 18},
 {36, 4, 65, 52, 41, 84, 28, 76, 17, 60}, {35, 4, 83, 59, 27, 51, 75, 19, 43, 67},
 {34, 4, 47, 72, 80, 58, 15, 39, 23, 91}, {33, 4, 38, 82, 49, 14, 90, 71, 57, 25},
 {32, 4, 56, 89, 16, 73, 40, 24, 81, 48}, {39, 29, 63, 50, 71, 76, 88, 24, 19, 10},
 {48, 29, 44, 60, 77, 91, 67, 18, 9, 25}, {57, 29, 55, 43, 85, 69, 81, 8, 23, 17},
 {66, 29, 87, 80, 27, 16, 7, 41, 49, 64}, {75, 29, 68, 90, 15, 6, 28, 53, 61, 40},
 {84, 29, 79, 73, 5, 26, 14, 58, 45, 51}, {41, 30, 70, 24, 53, 18, 82, 58, 8, 83},
 {51, 30, 86, 25, 72, 64, 17, 10, 74, 40}, {61, 30, 78, 23, 19, 89, 45, 65, 49, 9},
 {71, 30, 46, 27, 85, 5, 56, 77, 15, 52}, {81, 30, 62, 28, 7, 47, 67, 42, 88, 14},
 {91, 30, 54, 26, 38, 76, 6, 16, 59, 69}, {77, 21, 35, 58, 42, 10, 90, 16, 65, 55},
 {69, 21, 37, 40, 80, 52, 9, 83, 63, 14}, {88, 21, 36, 49, 8, 73, 59, 44, 15, 74},
 {53, 21, 32, 85, 19, 38, 79, 72, 7, 60}, {45, 21, 34, 67, 56, 87, 17, 50, 82, 6},
 {64, 21, 33, 76, 68, 18, 47, 5, 43, 89}, {72, 31, 57, 16, 50, 83, 28, 5, 44, 78},
 {89, 31, 39, 14, 27, 60, 74, 55, 70, 6}, {82, 31, 48, 15, 65, 26, 43, 86, 7, 63},
 {42, 31, 84, 19, 8, 68, 54, 80, 56, 25}, {59, 31, 66, 17, 79, 46, 9, 24, 90, 47},
 {52, 31, 75, 18, 62, 10, 87, 38, 23, 73}, {80, 12, 61, 5, 24, 55, 67, 38, 86, 36},
 {73, 12, 41, 6, 85, 25, 63, 78, 35, 47}, {90, 12, 51, 7, 44, 76, 23, 37, 56, 70},
 {50, 12, 91, 8, 65, 62, 33, 27, 79, 40}, {43, 12, 71, 9, 54, 32, 74, 58, 28, 87},
 {60, 12, 81, 10, 34, 83, 46, 68, 49, 26}, {27, 3, 88, 38, 58, 78, 17, 68, 37, 48},
 {26, 3, 77, 47, 19, 40, 70, 36, 57, 87}, {28, 3, 69, 56, 86, 18, 49, 39, 79, 35},
```

```
{24, 3, 64, 65, 44, 34, 75, 85, 54, 14}, {23, 3, 53, 74, 33, 84, 63, 16, 67, 46},
{25, 3, 45, 83, 66, 55, 32, 62, 15, 76}, {85, 20, 82, 68, 16, 36, 9, 62, 39, 51},
{76, 20, 72, 87, 8, 14, 35, 48, 61, 46}, {67, 20, 89, 79, 37, 10, 15, 41, 54, 57},
{58, 20, 52, 44, 86, 66, 81, 19, 33, 6}, {49, 20, 42, 63, 75, 91, 70, 5, 17, 32},
{40, 20, 59, 55, 71, 78, 84, 34, 7, 18}, {19, 22, 90, 62, 5, 48, 74, 34, 41, 69},
{18, 22, 80, 46, 65, 6, 57, 51, 88, 32}, {17, 22, 73, 54, 39, 83, 7, 77, 33, 61},
{16, 22, 60, 86, 37, 71, 45, 8, 75, 47}, {15, 22, 50, 70, 81, 64, 35, 38, 9, 84},
{14, 22, 43, 78, 53, 36, 91, 66, 56, 10}}
```

4 Paketi

4.1 *FinitePlanes.m*

```
(*********************************************************************

FinitePlanes.m

Program by Vedran Krcadinac, April 1996.
Department of Mathematics, University of Zagreb

Email: krcko@student.math.hr

*********************************************************************)



BeginPackage["FinitePlanes`"]



FinitePlanes::usage =
  "FinitePlanes.m is a package that implements finite projective and affine
   planes in Mathematica."

LoadFF::usage =
  StringJoin["LoadFF[",FromCharacterCode[34],"name",FromCharacterCode[34],"]
  loads a finite field into Mathematica. One can also load quasifields."]

FFPlus::usage =
  "FFPlus[x,y] adds the elements x,y in the currently loaded finite field."

FFTimes::usage =
  "FFTimes[x,y] multiplies the elements x,y in the currently loaded finite
   field."

FFOrder::usage =
  "FFOrder is the order of the currently loaded finite field."
```

```
ProjectivePlane::usage =
  "ProjectivePlane returns a projective plane constructed from the currently
   loaded finite field."

DualProjectivePlane::usage =
  "DualProjectivePlane constructs a plane dual to the one obtained from
   ProjectivePlane. If the construction starts from a finite field, the
   two planes coincide."

Points::usage =
  "Points returns a list of points of the projective plane to construct,
   i.e. vectors that span one-dimensional subspaces of the three-dimensional
   vector space over the currently loaded finite field."

AffinePlane::usage =
  "AffinePlane returns an affine plane constructed from the currently loaded
   finite field."

DualAffinePlane::usage =
  "DualAffinePlane returns an affine plane dual to the one obtained from
   AffinePlane. If the construction starts from a finite field, the two
   planes coincide."

LeftDirections::usage =
  "LeftDirections returns a list of one dimensional subspaces of the two-
   dimensional left vector space over the currently loaded finite field."

RightDirections::usage =
  "RightDirections returns a list of one dimensional subspaces of the two-
   dimensional right vector space over the currently loaded finite field."

Normalize::usage =
  "Normalize is a transformation rule that converts ordered pairs
   constituting an affine plane to integers."

Paint::usage =
  "Paint[ap] , Paint[ls] makes a picture of the affine plane ap or the
   list of mutually orthogonal latin squares ls, respectively."

Color::usage =
  "Color is a function that determines how Paint colors pictures of
   affine planes and latin squares."

Color9::usage =
```

```
  "My favourite coloring for the affine plane of order 9."

BlackAndWhite::usage =
  "A black and white coloring for affine planes and latin squares."

ProjectiveToAffine::usage =
  "ProjectiveToAffine[pp,k] converts the projective plane pp to an affine
   plane by deleting the k-th line and all of its points."

AffineToProjective::usage =
  "AffineToProjective[ap] converts the affine plane ap to a projective
   plane by adding a line at infinity."

LatinSquares::usage =
  "LatinSquares returns a complete list of mutually orthogonal latin
   squares constructed from the currently loaded finite field."

AffineToLatin::usage =
  "AffineToLatin[ap] converts the affine plane ap to a complete list of
   mutually orthogonal latin squares."

OrthoTest::usage =
  "OrthoTest[ls] checks if the latin squares in the list ls are mutually
   orthogonal."

IncidenceMatrix::usage =
  "IncidenceMatrix[pp] , IncidenceMatrix[ap] returns the incidence matrix
   of the projective (affine) plane pp (ap), respectively."

TestMatrix::usage =
  "An incidence matrix M represents a projective plane iff
   M.Transpose[M] == TestMatrix."

Vee::usage =
  "Vee[plane,A,B] is the unique line through the points A, B in plane."

Vee::samearg =
  "Attempt to join point with itself."

Wedge::usage =
  "Wedge[plane,a,b] is the intersection of the lines a, b of plane."

Wedge::samearg =
  "Attempt to intersect line with itself."
```

```
Begin["`Private`"]


(*  Load Operations in a Finite Field  *)

LoadFF[name_] :=
 ( Clear[FFPlus,FFTimes]; Get[name]; )


(*  Construction of Projective Planes   *)

Points :=
  Join[{{0,0,1}},Table[{0,1,i},{i,0,FFOrder-1}],
       Flatten[Table[{1,i,j},{i,0,FFOrder-1},{j,0,FFOrder-1}],1]]

FFInner[l1_,l2_] :=
  Inner[FFTimes,l1,l2,FFPlus]

ProjectivePlane :=
  Module[{po,p,x,i} , po = Points; Do[p[i] = po[[i]]; ,{i,1,Length[po]}];
         Table[Flatten[Position[po,x_List /; FFInner[p[i],x]==0,{1}]],
               {i,1,Length[po]}]
        ]

DualProjectivePlane :=
  Module[{po,p,x,i} , po = Points; Do[p[i] = po[[i]]; ,{i,1,Length[po]}];
         Table[Flatten[Position[po,x_List /; FFInner[x,p[i]]==0,{1}]],
               {i,1,Length[po]}]
        ]


(*  Construction of Affine Planes  *)


LeftSpan[v_] := Table[FFTimes[i,v],{i,0,FFOrder-1}]

RightSpan[v_] := Table[FFTimes[v,i],{i,0,FFOrder-1}]

LeftDirections :=
  Map[LeftSpan,Join[{{0,1}},Table[{1,i},{i,0,FFOrder-1}]]]

RightDirections :=
  Map[RightSpan,Join[{{0,1}},Table[{1,i},{i,0,FFOrder-1}]]]
```

```
Parallels[dir_] :=
  If[MemberQ[dir,{0,1}] ,
     Table[ Map[FFPlus[#,{i,0}] & , dir] , {i,0,FFOrder-1}] ,
     Table[ Map[FFPlus[#,{0,i}] & , dir] , {i,0,FFOrder-1}]
    ]

AffinePlane := Module[{d}, d = RightDirections;
    Table[Parallels[d[[j]]], {j, 1, FFOrder+1}]]

DualAffinePlane := Module[{d}, d = LeftDirections;
    Table[Parallels[d[[j]]], {j, 1, FFOrder+1}]]

Normalize := {x_Integer,y_Integer} -> 1+x+FFOrder*y;


(*  Visualisation of Affine Planes  *)

Square[{x_,y_}] := {Polygon[{{x,y},{x+1,y},{x+1,y+1},{x,y+1}}],
 {GrayLevel[0],Line[{{x,y},{x+1,y},{x+1,y+1},{x,y+1},{x,y}}]}}

MakePicture[Af_,color_] :=
  Table[ Graphics[   Table[ {color[j] ,
         Table[ Square[ Af[[i,j,k]] ] , {k,1,FFOrder}] } ,
         {j,1,FFOrder}],AspectRatio->1 ], {i,1,FFOrder+1}  ]

ToMatrix[x_List] :=
  Module[ {l,d},
          l = Length[x];
          d = Ceiling[N[Sqrt[l]]];
          Join[Partition[x,d],{Take[x,Floor[N[l/d]]*d-l]}]
        ]

ToMatrix[x_List ] :=
  Append[Partition[x,3],{Graphics[{}],Last[x],Graphics[{}]}] /; Length[x] == 10

Paint[af_ /; Depth[af] == 5,opt___] :=
 Show[GraphicsArray[ToMatrix[
  MakePicture[af,Color /. {opt} /. Options[Paint] ]]]]

Options[Paint] = { Color -> (Hue[(#-1)/(FFOrder-1)*.8] &) };

BlackAndWhite = (GrayLevel[(#-1)/(FFOrder-1)] &)
```

```
Color9[1] = RGBColor[1., 0., 0.]
Color9[2] = RGBColor[1., 0.5, 0.]
Color9[3] = RGBColor[1., 1., 0.]
Color9[4] = RGBColor[0., 1., 0.]
Color9[5] = RGBColor[0, 0.65, 0]
Color9[6] = RGBColor[0., 1., 1.]
Color9[7] = RGBColor[0., 0., 1.]
Color9[8] = RGBColor[0, 0, 0.65]
Color9[9] = RGBColor[1., 0., 1.]


(*  Relation between Projective and Affine Planes  *)

AffineToProjective[Af_] :=
  Module[ {i,j,a,b},
          a = Af /. Normalize;
          b = Table[Transpose[Join[Transpose[ a[[i]] ],
                              {Table[i+FFOrder^2,{j,1,FFOrder}]}
                                   ]
                              ] , {i,1,FFOrder+1}
                    ];
          a = Flatten[b,1];
          Join[a,{Table[i+FFOrder^2,{i,1,FFOrder+1}]}]
        ]

ProjectiveToAffine[pp_,pos_] :=
  Module[ {line} ,
          line = pp[[pos]];
          DeleteCases[Delete[pp,pos] , x_Integer /;
                      MemberQ[line,x] , -1          ]
        ]


(*  Construction of Latin Squares  *)

AffineToLatin[af_] :=
  Table[MakeSquare[af[[i]] /.
        {x_Integer,y_Integer}->{x+1,y+1}],{i,3,FFOrder+1}]

MakeSquare[x_] :=
  Module[ {sq,i,j},
          sq = Table[0,{i,1,FFOrder},{j,1,FFOrder}];
          For[i=1,i<=FFOrder,i++,
              For[j=1,j<=FFOrder,j++,
```

```
                  sq = ReplacePart[sq,i,x[[i,j]] ]
                 ]
             ];
          sq
        ]

LatinSquares := AffineToLatin[AffinePlane]


(*  Orthogonality Test  *)

Superimpose[sq1_, sq2_] :=
  Table[{sq1[[i,j]], sq2[[i,j]]}, {i, 1, Dimensions[sq1][[1]]},
   {j, 1, Dimensions[sq1][[2]]}]

Test[x_, y_] := Sort[Flatten[Superimpose[x, y], 1]]

OrthoTest[sq_] :=
 Module[{t,l,i,j},
        t = Flatten[Table[{i,j},{i,1,FFOrder},{j,1,FFOrder}],1];
        l = {};
        For[i=1,i<FFOrder-1,i++,
            For[j=i+1,j<=FFOrder-1,j++,
                AppendTo[l,Test[sq[[i]],sq[[j]]] == t] ]
           ];
        l
       ]


(*  Visualisation of Latin Squares  *)

Square[x_,y_,d_] :=
 { Polygon[{{x-d/2,y-d/2},{x+d/2,y-d/2},{x+d/2,y+d/2},
   {x-d/2,y+d/2}}], {GrayLevel[0] , Thickness[.001] ,
    Line[{{x-d/2,y-d/2},{x+d/2,y-d/2},{x+d/2,y+d/2},
    {x-d/2,y+d/2},{x-d/2,y-d/2}}] }
 }

Square[x_,y_,1] :=
  { Polygon[{{x-.5,y-.5},{x+.5,y-.5},{x+.5,y+.5},{x-.5,y+.5}}],
     {GrayLevel[0] , Thickness[.005] , Line[{{x-.5,y-.5},
     {x+.5,y-.5},{x+.5,y+.5},{x-.5,y+.5},{x-.5,y-.5}}] }
  }
```

```
LatinPicture[sq_,color_] :=
  Table[ {color[(sq[[k,j,i]])] ,
          Square[i,j,(FFOrder-k)/(FFOrder-1)]
         } , {i,1,FFOrder},{j,1,FFOrder},{k,1,FFOrder-1}
       ]

Paint[ls_ /; Depth[ls] ==4 , opt___] :=
  Show[Graphics[LatinPicture[ LatinSquares ,
       Color /. {opt} /. Options[Paint] ] ] ]


(*  Incidence Matrices  *)

IncidenceMatrix[plane_] :=
 Module[ {po,pl,i,j},
          po = Union[Flatten[plane,2]];
          pl = If[Depth[plane] > 3,Flatten[plane,1],plane];
          Table[If[MemberQ[pl[[i]],po[[j]]],1,0],
                {j,1,Length[po]}, {i,1,Length[pl]} ]
       ]


TestMatrix := FFOrder*IdentityMatrix[FFOrder^2 + FFOrder + 1] +
   Table[1, {FFOrder^2 + FFOrder + 1}, {FFOrder^2 + FFOrder + 1}]

(* Joining Points and Intersecting Lines *)

Vee[plane_,A_,B_] :=
  If[A==B, Message[Vee::samearg]; $Failed,
           Position[plane, x_ /; MemberQ[x,A] && MemberQ[x,B],{1}][[1,1]] ]

Wedge[plane_,a_,b_] :=
  If[a==b, Message[Wedge::samearg]; $Failed,
           Intersection[plane[[a]],plane[[b]]][[1]] ]

End[]

EndPackage[]
```

4.2 *FiniteFieldsV2.m*

```
(** FiniteFields.m  **)

(* Program by Ilan Vardi, December, 1989.

   Copyright Wolfram Research, Inc.
   P.O. Box 6059, Champaign, IL 61820, 217-398-0700   *)

(* Adjustment to Mathematica version 2 by
   Vedran Krcadinac, November, 1995.         *)

(*
   A finite field of p^n elements can be constructed as polynomials
   f[y] with coefficients (mod p), and the polynomials themselves
   are taken (mod Generator[y]), which is an irreducible polynomial
   of degree n (mod p). Multiplication and addition are just as usual
   but taken (mod p, Generator[y]).

   The only hard part is taking inverses (i.e., 1/f[y]), which ends up
   being equivalent to computing the extended GCD (mod p). In other words,
   given f[y], Generator[y], find a[y], b[y], such that

  a[y] f[y] + b[y] Generator[y] = GCD[Generator[y], f[y], Modulus-> p]

   If GCD = 1, then a[y] is the required inverse. This is the only
   nontrivial part. The rest of the code is just a routine set up of
   finite field rules.

   I set up a data structure FiniteField[f, y, {p, n}] which represents finite
   field elements. Once the rules of addition, multiplication,
   and division have been generalized to this data structure, other
   functions will operate on them like field elements. Thus
   PolynomialQuotient[f, g, x] will give the correct answer when
   applied to polynomials f, g, in x with coefficients FiniteField[f, y, {p, n}].

*)

(* This is the irreducible polynomial (mod p) that generates
   the finite field of p^n elements  *)

Generator[y_, p_, n_] := Generator[y, p, n] =
   First[Select[First /@
```

```
   FactorList[Cyclotomic[p^n-1,y], Modulus -> p],
   Exponent[#, y] == n&]]

(* An element in the finite field of p^n elements is a polynomial
   (mod Generator) and (mod p)                                         *)

FiniteField[f_, y_, {p_, n_}] :=
          FiniteField[PolynomialMod[PolynomialRemainder[PolynomialMod[f, p],
                   Generator[y, p, n], y], p],y, {p, n}] /;
     Exponent[f, y] >= n || Select[CoefficientList[f, y], # >= p&] != {}

(* The field laws for elements in a finite field. The only hard
   part is defining inverses, which is equivalent to finding the
   ExtendedGCD for polynomials (mod p)                             *)

FiniteField /: FiniteField[f_, y_, {p_, n_}] + FiniteField[g_, y_, {p_, n_}] :=
   FiniteField[PolynomialMod[f + g, p], y, {p, n}]

FiniteField /: FiniteField[f_, y_, {p_, n_}] + m_Integer :=
   FiniteField[PolynomialMod[f + m,p], y, {p, n}]

FiniteField /: FiniteField[f_, y_, {p_, n_}] * FiniteField[g_, y_, {p_, n_}] :=
   FiniteField[PolynomialMod[f * g, p] , y, {p, n}]

FiniteField /: FiniteField[f_, y_, {p_, n_}] * m_Integer :=
   FiniteField[PolynomialMod[f * m, p], y, {p, n}]

FiniteField /: FiniteField[0, y_, {p_, n_}] * x_ :=
   FiniteField[0, y, {p, n}]

(*    Taking 1/x, the only hard part    *)

FiniteField /: Power[FiniteField[f_, y_, {p_, n_}], -1] :=
   FiniteField[Last[FFExtendedGCD[Generator[y, p, n], f, y, p]], y, {p, n}]  /;
   PolynomialGCD[Generator[y, p, n], f, Modulus -> p] == 1

FiniteField /: FiniteField[f_, y_, {p_, n_}]^k_ :=
   FiniteField[PolynomialMod[f^k, p], y, {p, n}]    /;  k >= 0


(*    **********************************************************

     Computing ExtendedGCD (mod p). This is a little awkward
     since PolynomialRemainder[f, g, x], and PolynomialQuotient[f, g, x]
```

```
     don't have the option Modulus -> p, so if g is not monic
     you have to compute inverses (mod p) to make them monic.
     For example,

PolynomialRemainder[233 x^5 + 641 x^3 + 5 x + 17, 101 x^3 + 54 x^2 +67, x]

   =>  -(466108906/1030301) + (893999 x)/10201 - (549034337 x^2 )/1030301

    *************************************************************    *)


FFExtendedGCD[f_,g_, y_, p_] :=
    Reverse[FFExtendedGCD[g, f, y, p]] /;
            Exponent[PolynomialMod[f, p], y] < Exponent[PolynomialMod[g, p], y]

FFExtendedGCD[f_, g_, y_, p_Integer] :=
    {1, 0} /;  Exponent[PolynomialMod[g, p], y] == -Infinity

FFExtendedGCD[f_, g_, y_, p_] :=
     Block[{fp = PolynomialMod[f, p], gp = PolynomialMod[g, p],
            extf = {1, 0}, extg = {0, 1},
            q, r = y, leading},
       If[Exponent[gp, y] == 0, {0, PowerMod[gp, -1, p]},
           While[Exponent[r, y] > 0,
                 leading = Coefficient[gp, y, Exponent[gp, y]];
                  monic = PowerMod[leading, -1, p];
            q = PolynomialQuotient[PolynomialMod[monic fp, p],
                                   PolynomialMod[monic gp, p], y];
            r = leading *
                   PolynomialRemainder[PolynomialMod[monic fp, p],
                                       PolynomialMod[monic gp, p], y];
           {fp, gp, extf, extg}  = PolynomialMod[#, p] & /@
                                   {gp, r, extg, extf - q extg};
                 ];
             If[r == 0, Null, PowerMod[r, -1, p] extg]]]



(**************************************************)
(*        Additions to FiniteFields.m            *)
(**************************************************)

(* Vedran Krcadinac, March, 1996. *)
```

```
(* All elements of the field of order p^n *)

Poly[x_, a_] := a-1
Poly[x_, b__,a_] := a-1 + x Poly[x, b]

AllElements[x_,{p_,n_}] := AllElements[x,{p,n}] =
  Array[ FiniteField[Expand[Poly[x,##]],x,{p,n}] &,
 Table[p,{n}] ]  // Flatten

(* Define addition and multiplication in the field with
   p^n elements. For large fields this may take a while *)

Operations[{p_,n_}] :=
  Module[ {ff,i,j,x},
  Clear[FFPlus];
  Attributes[FFPlus]={Flat,Orderless,Listable};
  Clear[FFTimes];
  Attributes[FFTimes]={Listable};
  FFOrder = p^n;
  ff = AllElements[x,{p,n}];
  For[i=1, i<=p^n, ++i,
      For[j=1, j<=p^n, ++j,
  FFPlus[i-1,j-1] = Position[ff,ff[[i]]+ff[[j]]] [[1,1]]-1;
  FFTimes[i-1,j-1] = Position[ff,ff[[i]]*ff[[j]]] [[1,1]]-1
                ]
            ]
        ]


(* Save the operations to a specified file *)

SaveFF[{p_,n_},name_] :=
 ( Operations[{p,n}]; Put[Definition[FFOrder,FFPlus,FFTimes],name] )
```

4.3 *Quasifields.m*

```
(* Quasifields.m *)

(* Vedran Krcadinac, April, 1996. *)


(* Initial setup *)

<<FiniteFieldsV2.m;
Unprotect[NonCommutativeMultiply,v,var];
ClearAttributes[NonCommutativeMultiply,Flat];
Clear[var,v];
Protect[var,v];

(********************************************

           Hall quasifields

 ********************************************)

(* Irreducible quadratic polynomials over finite fields *)

ReducibleQ[el_,i_,j_] :=
  Module[ {m, reducible,k},
          m = Length[el];
          If[ i > m || j > m,True,
              reducible = False;
              k = 1;
              While[ k<= m && Not[reducible] ,
 reducible = SameQ[el[[k]]^2+el[[i]]*el[[k]]+el[[j]], el[[1]] ];
 ++k
                   ];
              reducible
            ]
        ]

IrreducibleQuadratic[{p_,n_}] :=
  IrreducibleQuadratic[{p,n}] =
  Module[ {el,m,i,j},
          el = AllElements[v,{p,n}];
          m = Length[el];
          i = 1;
```

```
          j = 1;
          While[ i<=m && ReducibleQ[el,i,j] ,
                 j = 1;
                 While[j<=m && ReducibleQ[el,i,j], ++j];
                 ++i;
               ];
          --i;
 If[ReducibleQ[el,i,j],
    Message[IrreducibleQuadratic::nonexsistent,p^n]; $Failed,
    var^2 + el[[i]]*var+el[[j]] ]
   ]

IrreducibleQuadratic::nonexsistent =
 "Irreducible quadratic over the field with `1` elements not found."

IrreducibleQuadratic[x_,{p_,n_}] :=
  IrreducibleQuadratic[{p,n}] /. var -> x

AllIrreducibleQuadratics[{p_,n_}] :=
  AllIrreducibleQuadratics[{p,n}] =
  Module[ {el,m,i,j,t},
          el = AllElements[v,{p,n}];
          m = Length[el];
          t = Table[ If[ReducibleQ[el,i,j],Null,var^2+el[[i]]*var+el[[j]] ],
                       {i,1,m},{j,1,m} ]  // Flatten;
          DeleteCases[t,Null]
        ]

(* Definition of addition and multiplication *)

Hall /: Hall[a_List] + Hall[b_List] := Hall[ a + b ]

a[{p_,n_}] :=
 -Coefficient[IrreducibleQuadratic[{p,n}],var,1]

b[{p_,n_}] :=
 -Coefficient[IrreducibleQuadratic[{p,n}],var,0]

Hall /: Hall[{x_FiniteField , FiniteField[0,v_,{p_,n_}] }] **
        Hall[{z_FiniteField , t_FiniteField }]  :=  Hall[{x*z,x*t}]

Hall /: Hall[{x_FiniteField , FiniteField[y_,v_,{p_,n_}] }] **
        Hall[{z_FiniteField , t_FiniteField}]   :=
        Hall[{x*z-(FiniteField[y,v,{p,n}]^(-1))*t*
```

```
IrreducibleQuadratic[x,{p,n}] ,
        FiniteField[y,v,{p,n}]*z - x*t + a[{p,n}]*t }]

(* Find all elements in a Hall quasifield *)

HallAll[{p_,n_}] := HallAll[{p,n}] =
 Map[Hall,Flatten[Table[{AllElements[v,{p,n}] [[j]],
     AllElements[v,{p,n}] [[i]]},{i,1,p^n},{j,1,p^n}],1]]

(* Define addition and multiplication in a Hall quasifiled *)

HallOperations[{p_,n_}] :=
  Module[ {ff,i,j},
   Clear[FFPlus,FFTimes];
   Attributes[FFPlus]={Flat,Orderless,Listable};
   Attributes[FFTimes]={Listable};
   FFOrder = p^(2*n);
           ff = HallAll[{p,n}];
           For[i=1, i<=p^(2*n), ++i,
       For[j=1, j<=p^(2*n), ++j,
           FFPlus[i-1,j-1] = Position[ff,ff[[i]]+ff[[j]]] [[1,1]]-1;
   FFTimes[i-1,j-1] = Position[ff,ff[[i]]**ff[[j]]] [[1,1]]-1
                 ]
              ]
]

(* Save a Hall quasifileld to a file *)

SaveHall[{p_,n_},name_] :=
 ( HallOperations[{p,n}]; Put[Definition[FFOrder,FFPlus,FFTimes],name] )

(**********************************************

          Andre quasifields

 **********************************************)


(* Definition of Andre objects. They should behave as FiniteField
   objects with respect to the operations * and +                   *)

Andre[f_, {p_, s_, d_}] :=
          Andre[PolynomialMod[PolynomialRemainder[PolynomialMod[f, p],
                   Generator[v, p, s*d], v], p], {p, s, d}] /;
```

```
     Exponent[f, v] >= s*d || Select[CoefficientList[f, v], # >= p&] != {}

Andre /: Andre[f_,  {p_, s_, d_}] + Andre[g_,{p_, s_, d_}] :=
   Andre[PolynomialMod[f + g, p], {p, s, d }]

Andre /: Andre[f_,{p_, s_, d_}] + m_Integer :=
   Andre[PolynomialMod[f + m,p],{p,s,d}]

Andre /: Andre[f_, {p_,s_,d_}] * Andre[g_,{p_, s_,d_}] :=
   Andre[PolynomialMod[f * g, p] , {p, s,d}]

Andre /: Andre[f_, {p_, s_,d_}] * m_Integer :=
   Andre[PolynomialMod[f * m, p], {p, s,d}]

Andre /: Andre[0, {p_, s_,d_}] * x_ :=
   Andre[0, {p, s,d}]


Andre /: Power[Andre[f_, {p_, s_,d_}], -1] :=
   Andre[Last[FFExtendedGCD[Generator[v,p,s*d],f,v,p]],{p, s,d}]  /;
   PolynomialGCD[Generator[v, p, s*d], f, Modulus -> p] == 1

Andre /: Andre[f_, {p_, s_,d_}]^k_ :=
   Andre[PolynomialMod[f^k, p], {p, s,d}]    /;  k >= 0


(* All Andre objects for given p,s,d   *)

AndreAll[{p_,s_,d_}] := AndreAll[{p,s,d}] =
   Array[ Andre[Expand[Poly[v,##]],{p,s,d}] &,
  Table[p,{s*d}] ]  // Flatten


(* The primitive root in the Andre finite field *)

PrimitiveQ[Andre[f_,{p_,s_,d_}]] :=
 Length[Union[Table[Andre[f,{p,s,d}]^i,{i,0,p^(s*d)-2}]]] == p^(s*d)-1

PrimitiveRoot[{p_,s_,d_}] := Andre[v,{p,s,d}]

CyclicOrder[{p_,s_,d_}] := CyclicOrder[{p,s,d}] =
 Table[Andre[v,{p,s,d}]^i,{i,0,p^(s*d)-2}]
```

```
(* The norm mapping *)

Norm[ Andre[f_,{p_,s_,d_}] ] :=
 Mod[ Position[CyclicOrder[{p,s,d}] , Andre[f,{p,s,d}]] [[1,1]] -1,
      p^s-1]

Norm2[ Andre[f_,{p_,s_,d_}] ] :=
  Andre[f,{p,s,d}] ^ Sum[p^(i*s),{i,0,d-1}]

Norm3[  Andre[f_,{p_,s_,d_}] ] :=
  Position[CyclicOrder[{p,s,d}],Norm2[Andre[f,{p,s,d}]]][[1,1]]-1


(* Find all possible functions f from {0,1,..,p^s-2} to {0,1,..,d-1}
   such that f[0] = 0. There are altogether d^(p^s-2) of them.
   One should choose one and assign it to Phi[{p,s,d}].            *)

MakeFunction = Function[i, #1[[i + 1]]] &

AllPhis1[{2,1,d_}] = {{0}}

AllPhis1[{p_, s_, d_}] :=
  Flatten[Array[{1,##1} - 1 & , Table[d, {p^s - 2}]],p^s-3]

AllPhis[{p_,s_,d_}] :=
 Map[MakeFunction,AllPhis1[{p,s,d}]]

(* If it is not already defined, randomly choose one Phi[{p,s,d}] *)

Phi[{p_,s_,d_}] := Phi[{p,s,d}] =
 AllPhis[{p,s,d}] [[ Random[Integer,{1,Length[AllPhis[{p,s,d}]]}] ]]

(* An isomorphism between the group {0,1,..,d-1} with addition (mod d)
   and the authomorphism group of the field with p^(s*d) elements
   that fix it's subfield of order p^s.                           *)

Aut[i_Integer, {p_,s_,d_}] :=
  Function[x,x^(p^(s*i))]


(* The mapping Alpha, from the multiplicative group of the field
   of order p^(s*d) to the above described automorphism group.
   The Andre quasifield will be associative if and only if this
   mapping is a homomorphism, i.e. if the chosen Phi is. Norm and
```

```
   Aut are always homomorphisms.                                *)

Alpha[ Andre[f_,{p_,s_,d_}] ] :=
  Aut[ Phi[{p,s,d}] [ Norm[Andre[f,{p,s,d}]] ], {p,s,d}]

(* Finally, the definition of multiplication in an Andre quasifield. *)

Andre /: Andre[0,{p_,s_,d_}] ** Andre[y_,{p_,s_,d_}] :=
 Andre[0,{p,s,d}]

Andre /: Andre[x_,{p_,s_,d_}] ** Andre[y_,{p_,s_,d_}] :=
 Andre[x,{p,s,d}] * Alpha[Andre[x,{p,s,d}]][Andre[y,{p,s,d}]]


(* Define operations in an Andre quasifield *)

AndreOperations[{p_,s_,d_}] :=
  Module[ {ff,i,j},
   Clear[FFPlus,FFTimes];
   Attributes[FFPlus]={Flat,Orderless,Listable};
   Attributes[FFTimes]={Listable};
   FFOrder = p^(s*d);
           ff = AndreAll[{p,s,d}];
           For[i=1, i<=p^(s*d), ++i,
       For[j=1, j<=p^(s*d), ++j,
           FFPlus[i-1,j-1] = Position[ff,ff[[i]]+ff[[j]]] [[1,1]]-1;
   FFTimes[i-1,j-1] = Position[ff,ff[[i]]**ff[[j]]] [[1,1]]-1
                  ]
              ]
]


(* Save an Andre quasifield to a file *)

SaveAndre[{p_,s_,d_},name_] :=
 ( AndreOperations[{p,s,d}]; Put[Definition[FFOrder,FFPlus,FFTimes],name] )



(**********************************************

          Hughes nearfields

 **********************************************)
```

```
(*
    Hughes nearfields are in fact special Andre quasifields.
    They are nearfields of order q^2, where q=p^s is an odd prime power.
    Phi is chosen to be the homomorphism

        Phi[i] = i (mod 2) , i=0,1,..,p^s-2

    The kernel F of a Hughes nearfield N is a finite field of order q.
    Elements of the kernel commute with all elements in N.

    We need Hughes nearfields for the construction of Hughes planes.
    What we also need is a 3*3 matrix A over F such that A^(q^2+q+1)
    is scalar and no smaller power of A has this property.

 *)


(* The subfield of order q=p^s in the field of order q^d = p^(s*d).
   This will be the kernel F.                                          *)

Subfield[{p_,1,d_}] := Subfield[{p,1,d}] = Take[AndreAll[{p,1,d}],{1,p}]

Subfield[{p_,s_,d_}] := Subfield[{p,s,d}] =
 Prepend[ Table[ Evaluate[Andre[v,{p,s,d}]^(j*Sum[p^(s*i),{i,0,d-01}])],
{j,0,p^s-2}],
         Andre[0,{p,s,d}] ]


(* Ordering of the Andre finite field in which its subfield comes first *)

SubfieldFirst[{p_,1,d_}] := SubfieldFirst[{p,1,d}] = AndreAll[{p,1,d}]

SubfieldFirst[{p_,s_,d_}] := SubfiledFirst[{p,s,d}] =
 Join[ Subfield[{p,s,d}],
       DeleteCases[AndreAll[{p,s,d}] ,
   x_Andre /; MemberQ[ Subfield[{p,s,d}],x]
  ]
     ]


(* To obtain the matrix A we need irreducible cubic polynomials over
    the kernel F                                                         *)
```

```
ReducibleQ[el_,i_,j_,k_] :=
  Module[ {m, reducible,l},
          m = Length[el];
          If[ i > m || j > m || k >m,True,
              reducible = False;
              l = 1;
              While[ l<= m && Not[reducible] ,
 reducible = SameQ[el[[l]]^3+el[[i]]*(el[[l]]^2)+el[[j]]*el[[l]]+el[[k]],
                   el[[1]] ];
                     ++l
                   ];
              reducible
            ]
        ]

AllIrreducibleCubics[{p_,s_,d_}] :=
  AllIrreducibleCubics[{p,s,d}] =
  Module[ {el,m,i,j,k,t},
          el = Subfield[{p,s,d}];
          m = Length[el];
          t = Table[ If[ReducibleQ[el,i,j,k],Null,
                        var^3+el[[i]]*var^2+el[[j]]*var+el[[k]]
                       ],
                     {i,1,m},{j,1,m},{k,1,m}
                   ]  // Flatten;
          DeleteCases[t,Null]
        ]


(* A function that checks whether a 3*3 matrix is scalar *)

ScalarQ[{{s_,a_,b_},{c_,s_,d_},{e_,f_,s_}}] := True /;
  Apply[ And , Map[ # == 0 || MatchQ[#,Andre[0,{_,_,_}]] & ,
                   {a,b,c,d,e,f} ] ]

ScalarQ[x_] := False


(* This function returns the least positive integer i such that M^i is scalar *)

MatrixOrder[M_] :=
  Module[ {P,i},
          i=1;  P=M;
          While[Not[ScalarQ[P]], ++i; P = P.M];
```

```
          i
        ]


(* Definition of the matrix A *)

MatrixA[pol_] :=
  Transpose[{Reverse[Take[CoefficientList[-pol,var], {1, 3}]],
             {1, 0, 0}, {0, 1, 0}}]


MatrixA[{p_,s_,d_}] :=  MatrixA[{p,s,d}] =
  Module[{i,A},
         i = 1;
         A = MatrixA[AllIrreducibleCubics[{p,s,d}][[1]]];
         While[ MatrixOrder[A] != (p^(2*s)+p^s+1) , ++i;
                A =  MatrixA[AllIrreducibleCubics[{p,s,d}][[i]]]
              ];
         A
        ]


(* Define operatons in a Hughes nearfield and save it to a file,
   together with A and all its relevant powers. In order to construct
   a Hughes plane we will have to take left multiplicative inverses,
   so they are computed and saved, too.                              *)

DefineA[{p_,s_,d_}] :=
 ( Clear[A]; Do[A[i] = MatrixPower[MatrixA[{p,s,d}],i] /.
 x_ /; MemberQ[Subfield[{p,s,d}],x] :> Position[Subfield[{p,s,d}],x][[1,1]]-1,
                { i, 0, p^(2*s)+p^s}] )

HughesOperations[{p_,s_}] :=
  Module[ {ff,i,j},
          Phi[{p,s,2}] = Function[i,Mod[i,2]];
   Clear[FFPlus,FFTimes];
   Attributes[FFPlus]={Flat,Orderless,Listable};
   Attributes[FFTimes]={Listable};
   FFOrder = p^(s*2);
          ff = SubfieldFirst[{p,s,2}];
          For[i=1, i<=p^(s*2), ++i,
       For[j=1, j<=p^(s*2), ++j,
          FFPlus[i-1,j-1] = Position[ff,ff[[i]]+ff[[j]]] [[1,1]]-1;
   FFTimes[i-1,j-1] = Position[ff,ff[[i]]**ff[[j]]] [[1,1]]-1
```

```
                ]
            ]
]

DefineInverses[{p_,s_}] :=
  Module[ {el,i},
          Clear[FFInverse];
          el = Table[i,{i,1,p^(2*s)-1}];
          Do[ FFInverse[i] = Position[Map[FFTimes[#,i]&,el],1][[1,1]],
              {i,1,p^(2*s)-1} ]
        ]

SaveHughes[{p_,s_},name_] :=
 ( DefineA[{p,s,2}]; HughesOperations[{p,s}]; DefineInverses[{p,s}];
   Put[Definition[FFOrder,FFPlus,FFTimes,FFInverse,A],name] )
```

4.4 *HughesPlanes.m*

```
(*  HughesPlanes.m  *)

(*  Vedran Krcadinac, April, 1996.  *)


(* Load a Hughes nearfield and all necessary objects *)

LoadFF[name_] :=
 ( Clear[FFPlus,FFTimes,FFInverse,A,L]; Get[name];  L[t_] := L[t] =
   Cases[Points[FFOrder] , {x_,y_,z_} /; FFPlus[x,FFTimes[y,t],z] == 0 ]
 )


(* The points in a Hughes plane *)

Points[x_ /; x == FFOrder] := Points[x] =
  Join[{{0,0,1}},Table[{0,1,i},{i,0,FFOrder-1}],
       Flatten[Table[{1,i,j},{i,0,FFOrder-1},{j,0,FFOrder-1}],1]]


(* L(t) is the set of all points (x,y,z) that satisfy the equation
   x + yt + z = 0                                                  *)
```

```
L[t_] := L[t] =
  Cases[Points[FFOrder] , {x_,y_,z_} /; FFPlus[x,FFTimes[y,t],z] == 0 ]


(* Matrix-vector product over the loaded Hughes nearfield  *)

FFDot[M_, v_] :=
  Table[Inner[FFTimes, M[[i]], v, FFPlus], {i, 1, First[Dimensions[M]]}]


(* The construction of Hughes planes  *)

Norm[{0, 0, x_}] := {0, 0, 1}

Norm[{0, x_, y_}] := {0, 1, FFTimes[FFInverse[x], y]}

Norm[{x_, y_, z_}] :=
  {1, FFTimes[FFInverse[x], y], FFTimes[FFInverse[x], z]}

HughesPlane := Map[Norm,
 Join[ Table[ Map[FFDot[A[m],#] &, L[1]], {m,0,FFOrder+Sqrt[FFOrder]}],
       Flatten[ Table[ Map[FFDot[A[m],#] &, L[t]], {m,0,FFOrder+Sqrt[FFOrder]},
              {t,Sqrt[FFOrder],FFOrder-1} ] , 1]
     ],
             {2}]


(* A normalization rule that transforms ordered triplets to integers *)

Normalize := {x_Integer,y_Integer,z_Integer} :>
         Position[Points[FFOrder],{x,y,z}][[1,1]]


(* Transformation of Hughes projective planes to affine planes *)

ProjectiveToAffine[pp_,pos_] :=
  Module[ {plane,line} ,
          plane = pp /. NRule;
          line = plane[[pos]];
          DeleteCases[Delete[plane,pos] , x_Integer /;
                      MemberQ[line,x] , -1         ]
        ]
```

```
(* Compute the incidence matrix of a plane *)

IncidenceMatrix[plane_] :=
  Module[{po, i, j}, po = Union[Flatten[plane, 1]];
    Table[If[MemberQ[plane[[i]], po[[j]]], 1, 0], {j, 1, Length[po]},
     {i, 1, Length[plane]}]]

(* The incidence matrix M of a projective plane should satisfy
   M . Transpose[M] == TestMatrix                                *)

TestMatrix := FFOrder*IdentityMatrix[FFOrder^2 + FFOrder + 1] +
   Table[1, {FFOrder^2 + FFOrder + 1}, {FFOrder^2 + FFOrder + 1}]


(* Joining points and intersecting lines *)

Vee[plane_,A_,B_] :=
  If[A==B, Message[Vee::samearg]; $Failed,
         Position[plane, x_ /; MemberQ[x,A] && MemberQ[x,B],{1}][[1,1]] ]

Vee::samearg =
  "Attempt to join point with itself.";

Wedge[plane_,a_,b_] :=
  If[a==b, Message[Wedge::samearg]; $Failed,
         Intersection[plane[[a]],plane[[b]]][[1]] ]

Wedge::samearg =
  "Attempt to intersect line with itself.";
```

5 Zaključak

Tema ovog rada je konstrukcija projektivnih ravnina i toga sam se uglavnom držao. U poglavlju 2.5 samo sam naznačio što se sve s ravninama može dalje raditi u *Mathematici*. Ovdje bih želio navesti još nekoliko ideja.

Mathematica je velik program koji sadrži funkcije za rješavanje najrazličitijih problema. Napisani su i mnogobrojni paketi koji još više proširuju njezine mogućnosti. Sav taj alat možemo upotrijebiti na objektima konstruiranim u ovom radu, konačnim ravninama.

Prvi smjer analize ravnina je geometrijski. Zainteresirani čitatelj mogao bi napraviti funkcije koje će u projektivnim ravninama tražiti Desarguesove konfiguracije. Pomoću takvih funkcija mogli bi razlikovati desarguesovske od nedesarguesovskih ravnina, a i nedesarguesovske ravnine među sobom. U translacijskim ravninama mogli bi identificirati translacijski pravac, a u dualnim translacijskim translacijsku točku. U Hughesovoj ravnini mogli bi pokazati da ni jedan pravac niti točka nisu translacijski. Daljnja geometrijska analiza uključivala bi pronalaženje ovala i hiperovala, konika, podravnina itd.

Konačne ravnine su i kombinatorni objekti. Za ovaj smjer njihovog proučavanja obećavajuće je što postoje mnogi gotovi paketi za *Mathematicu* koji se bave kombinatornom problematikom.

Još jedan smjer proučavanja ravnina je preko njihovih grupa kolineacija. Nažalost u *Mathematicu* nisu ugrađene funkcije za rad s grupama permutacija, a meni je poznat samo jedan vrlo oskudan paket s tog područja. Ovaj smjer zahtjevao bi vjerojatno više programiranja.

Ovaj rad, kao i eventualni budući koji će nastati na tom području mogu poslužiti u nekoliko svrha. Kao prvo, dobili smo velik izvor primjera i protuprimjera. Do zanimljivih primjera konačnih ravnina teško je doći bez upotrebe računala, jer oni počinju kod reda devet, a to su ipak prilično veliki objekti.

Drugo, *Mathematica* je moćan sustav za vizualizaciju. Dobra slika izrađena u *Mathematici* može nam pomoći da shvatimo neki problem ili da si predočimo neku strukturu.

Ne treba zanemariti niti ulogu koju *Mathematica* može igrati u nastavi. Napredniji tečaj geometrije ili kombinatorike mogao bi koristiti neki od spomenutih paketa, bilo kao ilustraciju obrađenog gradiva, bilo kao materijal za zadatke i vježbe.

Konačno, ovaj i slični radovi mogu poslužiti i za znanstveni rad. Dobivamo mogućnost da svoje hipoteze ispitamo na puno većem broju primjera nego što bi to ikada mogli bez računala. To je svakako korisno ako želimo provjeriti neku tvrdnju, prije nego što se upustimo u dokazivanje.

6 Literatura

[1] D.R.Hughes, F.C.Piper, *Projective Planes*, Springer–Verlag, 1973.

[2] S.Wolfram, *Mathematica: A System for Doing Mathematics by Computer*, drugo izdanje, Addison–Wesley, 1993.

[3] *Guide to Standard Mathematica Packages*, Wolfram Research Technical Report, 1993.