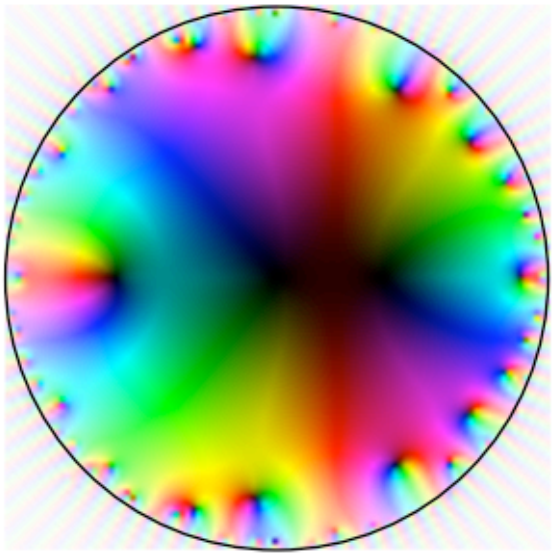
Sage Days 16: Modular Forms

Sage Days 16: Modular Forms

June 24, 2009, William Stein

```
f = EllipticCurve('389a').q_eigenform(100).truncate()
show(circle((0,0),1) + complex_plot(f,(-1,1),(-1,1)), axes=False)
```



Congruence Subgroups

1. Can create $\Gamma_0(N)$, $\Gamma_1(N)$, and $\Gamma_H(N)$ for each subgroup H of $(\mathbf{Z}/N\mathbf{Z})^*$.
2. (Also) non-congruence subgroups
3. Cusp equivalence
4. Generators
5. Coset representatives
6. Genus of corresponding modular curve

Example: We create a couple of different congruence subgroups of various types.

```
Gamma0(11)
```

```
Congruence Subgroup Gamma0(11)
```

```
SL2Z
```

```
Modular Group SL(2,Z)
```

```
Gamma1(13)
```

```
Congruence Subgroup Gamma1(13)
```

```
GammaH(11,[2])
```

```
Congruence Subgroup Gamma_H(11) with H generated by [2]
```

```

```

Example: We create a non-congruence subgroup of index 7.

```
a2 = SymmetricGroup(7)([(1,2),(3,4),(5,6)]); a3 = SymmetricGroup(7)
([(1,3,5),(2,6,7)])
G  = ArithmeticSubgroup_Permutation(a2*a3, ~a2 * ~a3); G
```

```
Arithmetic subgroup corresponding to permutations
L=(1,6)(2,3,4,5,7), R=(1,7,6,3,4)(2,5)
```

```
G.index()
```

```
7
```

```
G.is_congruence()
```

```
False
```

```
G.dimension_cusp_forms(4)
```

```
1
```

Example: We enumerate cusps.

```
Gamma0(20).cusps()
```

```
[0, 1/10, 1/5, 1/4, 1/2, Infinity]
```

```
Gamma1(13).cusps()
```

```
[0, 2/13, 1/6, 1/5, 3/13, 1/4, 4/13, 1/3, 5/13, 6/13, 1/2, Infinity
```

Example: We compute generators for some congruence subgroups.

```
Gamma0(11).gens()
```

```
([1 1]
[0 1], [-1  0]
[ 0 -1], [ 1 -1]
[ 0  1], [ 1  0]
[11  1], [1 1]
[0 1], [-5 -1]
[11  2], [-7 -1]
[22  3], [-8 -1]
```

```
[33  4], [-2 -1]
[11  5], [-9 -1]
[55  6], [-3 -1]
[22  7], [-4 -1]
[33  8], [-6 -1]
[55  9], [-1  0]
[11 -1], [  1   0]
[-11   1])
```

Example: We compute the genus of the corresponding modular curve in some cases.

```
Gamma0(11).genus()
```

```
1
```

```
Gamma1(13).genus()
```

```
2
```

Dirichlet Characters

1. Group of Dirichlet characters over a given ring
2. Factor as product of prime power moduli characters
3. Conductor
4. Extend, restrict
5. Generalized Bernoulli numbers (fast)
6. Kloosterman, Gauss, and Jacobi sums
7. Fast arithmetic
8. (upside down) kronecker character

Example: We enumerate all the Dirichlet characters of a given level (explain notation).

```
G = DirichletGroup(40); G
```

```
Group of Dirichlet characters of modulus 40 over Cyclotomic Field c
order 4 and degree 2
```

```
list(G)
```

```
[[1, 1, 1], [-1, 1, 1], [1, -1, 1], [-1, -1, 1], [1, 1, zeta4], [-1
1, zeta4], [1, -1, zeta4], [-1, -1, zeta4], [1, 1, -1], [-1, 1, -1]
[1, -1, -1], [-1, -1, -1], [1, 1, -zeta4], [-1, 1, -zeta4], [1, -1,
-zeta4], [-1, -1, -zeta4]]
```

```
G.gens()
```

```
([-1, 1, 1], [1, -1, 1], [1, 1, zeta4])
```

```
G.unit_gens()
```

```
[31, 21, 17]
```

```
a = G.0; c = G.2; timeit('a*c')
```

```
    625 loops, best of 3: 8.25 µs per loop
```

```
c(7)
```

```
    zeta4
```

```
timeit('c(7)')
```

```
    625 loops, best of 3: 5.04 µs per loop
```

```

```

Example: We factor a Dirichlet character as a product of prime power characters.

```
G.<a,b,c> = DirichletGroup(40); eps=a*b*c; eps
```

```
    [-1, -1, zeta4]
```

```
D = eps.decomposition(); D
```

```
    [[-1, -1], [zeta4]]
```

```
D[0].parent()
```

```
    Group of Dirichlet characters of modulus 8 over Cyclotomic Field of
    order 4 and degree 2
```

```
D[1].parent()
```

```
    Group of Dirichlet characters of modulus 5 over Cyclotomic Field of
    order 4 and degree 2
```

```
D[0].extend(40) * D[1].extend(40) == eps
```

```
    True
```

```

```

Example: We compute the conductor of a Dirichlet character.

```
G.<a,b,c> = DirichletGroup(40); (a*b*c).conductor()
```

```
    40
```

```
(a*c).conductor()
```

```
    20
```

```
c.conductor()
```

```
    5
```

```
b.conductor()
```

```
    8
```

```
G.<a,b,c> = DirichletGroup(40); eps=a*b*c
```

```

```

```
eps.bernoulli(2)
```

```
    -20*zeta4 + 16
```

Example: We compute generalized Bernoulli numbers associated to a Dirichlet character.

```
time eps.bernoulli(300)
```

```
    120113342813768032741195920158640442126768503462211681546631004145
```

```
885318059442658391016022436062235107932040576627755548574901281234 4
245958599432370934852669110581397638997661831709778889283201706115 6
927380284546124952485839417416396184756978638661860566750322860237 7
409738130092016266548010945422132308191377855431388946342571592763 1
550776492198218562374956068841552437779602051536593552526928787094 9
095918482280459785311526800208502951370638659264200869036230988029 8
990787061703634307486190361807409342493198379774531806750130827833 0
130481678516542691156033925661780745253947737666898625153015267832 0
729759292613442018694776069350770909759782911752333010719480709089 4
297316934080971321329361932399473845325913213955434201982615497825 8
647951766948891897643565467323752074876098450349160804303256806061 8
166227626320349977512253379347559435700 0*zeta4 -
742341283612765755586319662160357529997237332740202793019305000934 7
394987566614740931637153108128767667582104634777851538471153189974 4
817204782138218863999077698632706687249608930875882887476091377384 7
156845173419537796753483712518370761450514652791942103502862556919 0
084475947338034617371627024068132350764863941234108202283257302505 1
331674462151355216199842659857119318869028073785380511039411014233 8
293076694691800421543898828280916373535475493088078601084010953965 8
298930200422214877664847003801454386134162265252631872014110920371 4
711370298509977714544663117879902934276402674852266074499721567709 4
227056739391879401881665443906193672379092507972353803358877287624 2
424583652886993875430811515873271013715869271188896719491344041954 8
425903939631550148970342415952127584324987874501870109329976926874 1
860444722186184737733253633002261973600
Time: CPU 0.18 s, Wall: 0.25 s
```

Example: We compute Kloosterman, Gauss, and Jacobi sums associated to a character.

```
G.<a,b,c> = DirichletGroup(40); eps=a*b*c
```

```
eps.kloosterman_sum()
```

```
-2*zeta40^15 - 4*zeta40^13 + 4*zeta40^11 - 4*zeta40^7 - 2*zeta40^5
4*zeta40
```

```
eps.gauss_sum()
```

```
-2*zeta40^15 - 4*zeta40^13 + 4*zeta40^11 - 4*zeta40^7 - 2*zeta40^5
4*zeta40
```

```
G.<a> = DirichletGroup(7)
a.jacobi_sum(a)
```

```
-zeta42^7 + 3
```

Cusps

1. Equivalence
2. Linear fractional transformation action
3. Galois action

Example: We create two cusps and verify equivalence.

```
alpha = Cusp(2/3); beta = Cusp(2/3 + 10)
```

```
alpha
```

```
2/3
```

```
alpha.parent()
```

```
Set P^1(QQ) of all cusps
```

```
alpha.is_gamma0_equiv(beta, 3)
```

```
True
```

```
```

Example: We act on a cusp by a linear fractional transformation.

```
alpha = Cusp(2/3); alpha.apply([1,-1,1,0])
```

```
-1/2
```

```
```

Dimension Formulas

1. Largest range of formulas implemented anyway
2. This code has evolved since 1990: Shimura's book, then Bruce Kaskel in Pari at Berkeley, then Kevin Buzzard in Pari, then me in C++, then me in Magma, then here in Sage; many additional formulas due to Henri Cohen, Jordi Quer, etc.
3. Dimensions of cusp and eisenstein spaces with and without character, for subgroups Gamma_H
4. For each, dimension of new, old, and p-new subspaces
5. I think half-integral weight formulas would be easy to implement but aren't there yet (see Cohen-Oesterle)

Example: We compute the dimensions of some spaces of modular forms with and without character.

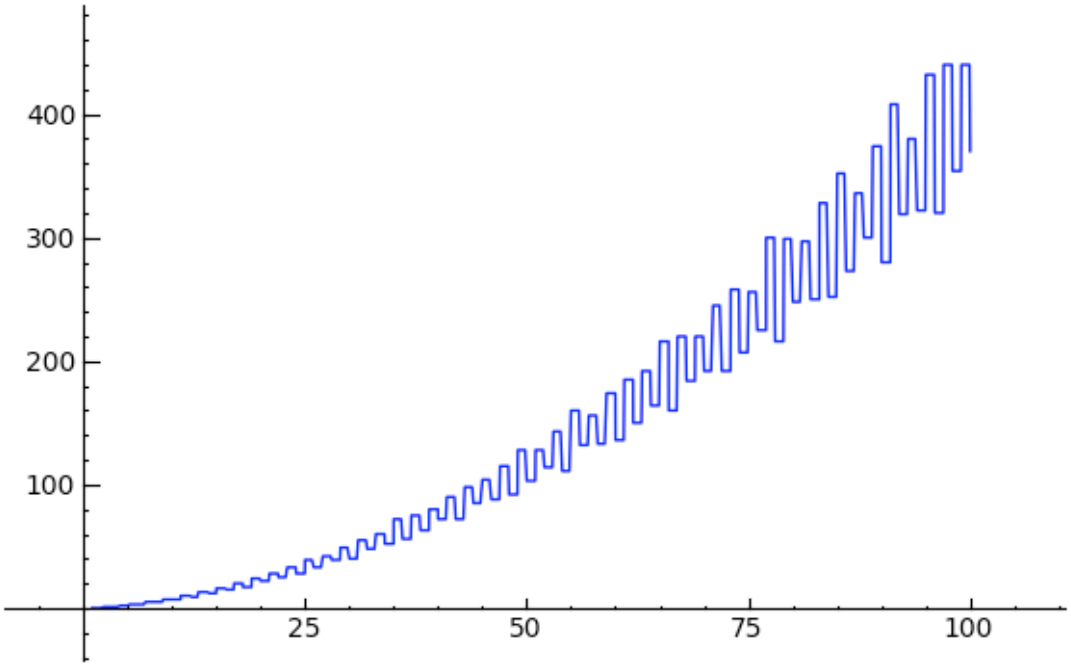
```
dimension_modular_forms(Gamma1(13),2)
```

```
13
```

```
dimension_modular_forms(Gamma1(2009),2)
```

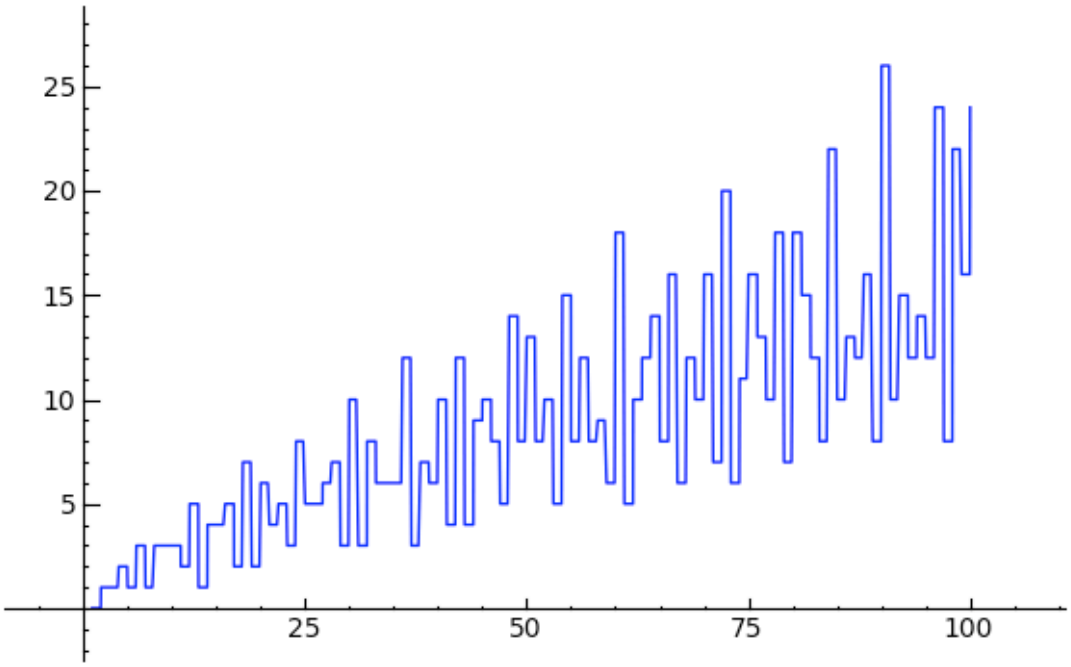
```
167040
```

```
plot(lambda N: dimension_modular_forms(Gamma1(floor(N)),2),
(1,100))
```



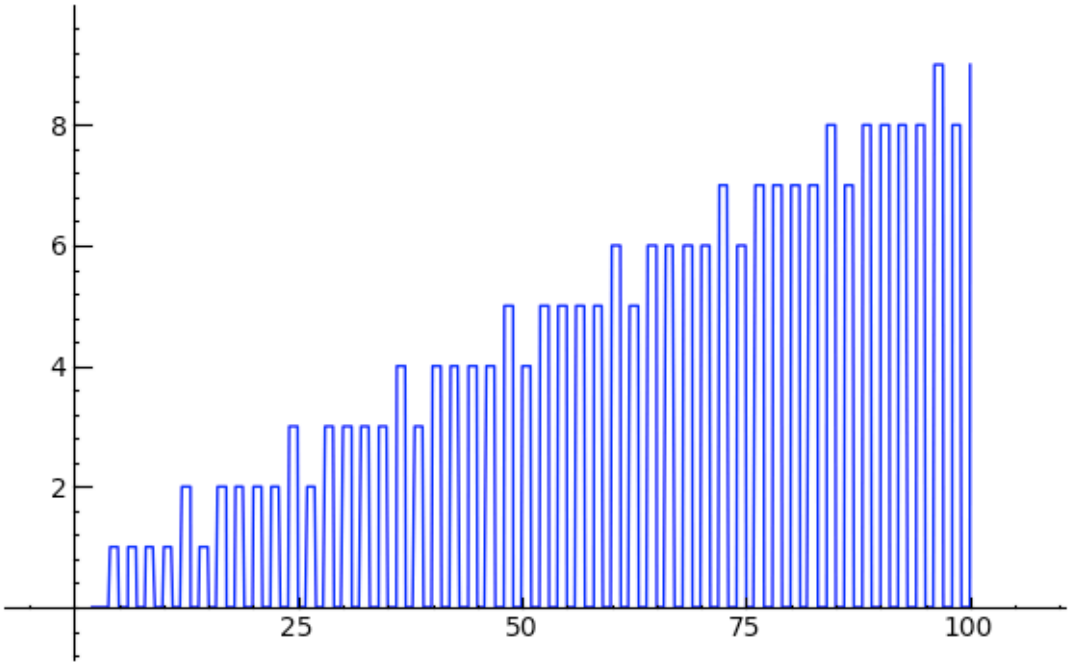

```
@interact
def f(k=(2,4,6,8,10)):
   plot(lambda N: dimension_modular_forms(Gamma0(floor(N)),k),
(1,100)).show()
```

k




```
@interact
def f(N=(1..16)):
   plot(lambda k: dimension_modular_forms(Gamma1(N),floor(k)),
(2,100)).show()
```

N



Example: We compute the dimension of a new subspace and a p-new subspace.

```
dimension_cusp_forms(66)
```

```
9
```

```
dimension_new_cusp_forms(66)
```

```
3
```

```
dimension_new_cusp_forms(66,p=2)
```

```
7
```

```
dimension_new_cusp_forms(66,p=3)
```

```
9
```

```
dimension_new_cusp_forms(66,p=11)
```

```
9
```

Basis Of Modular Forms Spaces

1. Can compute for integral weight > 1 and any subgroup or character
2. Can compute for half-integral weight (less natural)
3. Echelon form basis
4. Also have integral basis in Hermite form when space is defined over $\mathbf{Q}$.
5. Special faster algorithms for level 1

Example: We compute bases for a few space of modular forms.

```
ModularForms(1,12).basis()
```

```
[
q - 24*q^2 + 252*q^3 - 1472*q^4 + 4830*q^5 + O(q^6),
1 + 65520/691*q + 134250480/691*q^2 + 11606736960/691*q^3 +
274945048560/691*q^4 + 3199218815520/691*q^5 + O(q^6)
]
```

```
for f in CuspForms(33,2, prec=10).basis(): view(f)
```

$q - q^5 - 2q^6 + 2q^7 - 2q^8 - q^9 + O(q^{10})$
$q^2 - q^4 - q^5 - q^6 + 2q^7 - q^8 + q^9 + O(q^{10})$
$q^3 - 2q^6 - q^9 + O(q^{10})$

```
for f in ModularForms(Gamma1(13),2).basis():
view(f.q_expansion(16))
```

$q - 4q^3 - q^4 + 3q^5 + 6q^6 - 3q^8 + q^9 - 6q^{10} - 2q^{12} + 2q^{13} + O(q^{16})$
$q^2 - 2q^3 - q^4 + 2q^5 + 2q^6 - 2q^8 + q^9 - 3q^{10} + 3q^{13} - 2q^{15} + O(q^{16})$
$1 + \frac{21060}{19}q^{11} - \frac{36504}{19}q^{12} + \frac{75504}{19}q^{13} - \frac{388440}{19}q^{14} + \frac{595140}{19}q^{15} + O(q^{16})$
$q + \frac{11709}{19}q^{11} - \frac{20687}{19}q^{12} + \frac{43295}{19}q^{13} - \frac{219842}{19}q^{14} + \frac{336765}{19}q^{15} + O(q^{16})$
$q^2 + 262q^{11} - 467q^{12} + 964q^{13} - 4779q^{14} + 7306q^{15} + O(q^{16})$
$q^3 + \frac{918}{19}q^{11} - \frac{1215}{19}q^{12} + \frac{2554}{19}q^{13} - \frac{15203}{19}q^{14} + \frac{23529}{19}q^{15} + O(q^{16})$
$q^4 - \frac{882}{19}q^{11} + \frac{2095}{19}q^{12} - \frac{4330}{19}q^{13} + \frac{18130}{19}q^{14} - \frac{27230}{19}q^{15} + O(q^{16})$
$q^5 - \frac{1287}{19}q^{11} + \frac{2607}{19}q^{12} - \frac{5440}{19}q^{13} + \frac{25201}{19}q^{14} - \frac{38200}{19}q^{15} + O(q^{16})$
$q^6 - \frac{1080}{19}q^{11} + \frac{2024}{19}q^{12} - \frac{4138}{19}q^{13} + \frac{20053}{19}q^{14} - \frac{30520}{19}q^{15} + O(q^{16})$
$q^7 - \frac{675}{19}q^{11} + \frac{1056}{19}q^{12} - \frac{2192}{19}q^{13} + \frac{12355}{19}q^{14} - \frac{19075}{19}q^{15} + O(q^{16})$
$q^8 - \frac{360}{19}q^{11} + \frac{453}{19}q^{12} - \frac{898}{19}q^{13} + \frac{5975}{19}q^{14} - \frac{9350}{19}q^{15} + O(q^{16})$
$q^9 - \frac{153}{19}q^{11} + \frac{98}{19}q^{12} - \frac{204}{19}q^{13} + \frac{2290}{19}q^{14} - \frac{3665}{19}q^{15} + O(q^{16})$
$q^{10} - \frac{54}{19}q^{11} - \frac{9}{19}q^{12} + \frac{4}{19}q^{13} + \frac{597}{19}q^{14} - \frac{994}{19}q^{15} + O(q^{16})$

```

```

Example: We compute a Hermite form integral basis.

```
for f in ModularForms(Gamma1(13),2).integral_basis():
   view(f.q_expansion(16))
```

$1 + 17940q^{13} - 68328q^{14} + 58812q^{15} + O(q^{16})$
$q + 9920q^{13} - 37946q^{14} + 32946q^{15} + O(q^{16})$
$q^2 + 4188q^{13} - 15943q^{14} + 13765q^{15} + O(q^{16})$

$q^3 + 862q^{13} - 3125q^{14} + 2496q^{15} + O(q^{16})$
$q^4 - 634q^{13} + 2602q^{14} - 2495q^{15} + O(q^{16})$
$q^5 - 1021q^{13} + 4015q^{14} - 3631q^{15} + O(q^{16})$
$q^6 - 886q^{13} + 3415q^{14} - 2992q^{15} + O(q^{16})$
$q^7 - 599q^{13} + 2257q^{14} - 1903q^{15} + O(q^{16})$
$q^8 - 340q^{13} + 1241q^{14} - 989q^{15} + O(q^{16})$
$q^9 - 165q^{13} + 574q^{14} - 419q^{15} + O(q^{16})$
$q^{10} - 68q^{13} + 219q^{14} - 139q^{15} + O(q^{16})$
$q^{11} - 23q^{13} + 64q^{14} - 30q^{15} + O(q^{16})$
$q^{12} - 6q^{13} + 12q^{14} - 3q^{15} + O(q^{16})$

Example: We compute a basis for a space of half-integral weight forms.

```
for f in
half_integral_weight_modform_basis(DirichletGroup(16*7).0^2,3,30):
view(f)
```

$q - 2q^2 - q^9 + 2q^{14} + 6q^{18} - 2q^{21} - 4q^{22} - q^{25} + O(q^{30})$
$q^2 - q^{14} - 3q^{18} + 2q^{22} + O(q^{30})$
$q^4 - q^8 - q^{16} + q^{28} + O(q^{30})$
$q^7 - 2q^{15} + O(q^{30})$

Example: We compute the Delta function to high precision:

```
delta_qexp(20)
```

```
q - 24*q^2 + 252*q^3 - 1472*q^4 + 4830*q^5 - 6048*q^6 - 16744*q^7 +
84480*q^8 - 113643*q^9 - 115920*q^10 + 534612*q^11 - 370944*q^12 -
577738*q^13 + 401856*q^14 + 1217160*q^15 + 987136*q^16 -
6905934*q^17 + 2727432*q^18 + 10661420*q^19 + O(q^20)
```

```
time d = delta_qexp(10^6)
```

```
Time: CPU 13.68 s, Wall: 20.93 s
```

```
time victor_miller_basis(500,44)
```

```
[
1 +
5072259795933427154456176117061862204013935652569831200541465923641
2240000000*q^42 +
6357018695355722738844474405421846752652532909969671978222422444501
567892480000000*q^43 + O(q^44),
q +
1820953916246675278123830541924585557242265464032154412055351963326
005268912*q^42 +
2107265275702640444969498266205697402111464858080628365534348164668
64285875592744*q^43 + O(q^44),
q^2 +
6391675096009654638667224571231706878029161816840216585802137329659
9607170*q^42 +
6822036969308680862596569323096117827649662733279852566661011416900
034919235584*q^43 + O(q^44),
```

```
q^3 +
219218957801301554356705068511029059165642830090480993097444964384 9
678464*q^42 +
215548905335089418160895064478737892975886216097229042126143810624 5
19633770507*q^43 + O(q^44),
q^4 +
734182911175009857450312856042680787717540014639358465505439966513 7
3152*q^42 +
664211237026323260936969685914289316971276673851896106461796055434 9
808295424*q^43 + O(q^44),
q^5 +
239934516375122422127020407262493718468402055155924339201930782948 5
000*q^42 +
199466354454137600393836907935897270398205311445001792167870743545 4
44654000*q^43 + O(q^44),
q^6 +
764585794634546120164392176234173972348763236669507668445096020435 0
2*q^42 +
583299556307129186776239613379726486769113062945347698845233907497 5
193664*q^43 + O(q^44),
q^7 +
237393737661873528100671523865231023760225087156118246271845925280 6
*q^42 +
165961438481789420523935626312997909631821418345626956455951206981 5
63596*q^43 + O(q^44),
q^8 +
717573632818421083201994123080097234594664737130980516964835771680*
^42 +
459017065766115064859664713762001231843577105005198964396466558738 1
432*q^43 + O(q^44),
q^9 +
210979786566389738218429350748636157328915841662672343114639597 68*q
42 +
123294741191592329211209347706077146226492842367990036600009406618 8
16*q^43 + O(q^44),
q^10 +
602825584665960506669460774625043889596017647930172328596812855*q^4
+
321303423893704719384568353017370212864865857001335044008469698969 6
*q^43 + O(q^44),
q^11 +
167221553564565677702795664924630312120953336415964855197617 28*q^42
+
811470078025492778906560363408070089891788468555092576080647564711*
^43 + O(q^44),
q^12 +
449870266324945459826970685905810069752577729667187073272832*q^42 +
198387565302271021922902870147607838996781454363977124511838371 84*q
43 + O(q^44),
q^13 +
117243813386920028238836931258343503898673468235489161156 16*q^42 +
468924739007744843125568244415516742038338580822851328824942640*q^4
```

```
+ O(q^44),
q^14 +
29565034880352067998046815204858346674954185200920860680 8*q^42 +
10701878150767325620875506905520239829807213066206995699204096*q^43
+ O(q^44),
q^15 + 72043040197411503987585793266010814401022316865336500 80*q^42
+ 23548380093796936336656201157279274407901465984632760960 3910*q^43
+ O(q^44),
q^16 + 16940524740384456449330457921642272142624658568868864 0*q^42
49880557722173222925603772793666175138805779396408691916 80*q^43 +
O(q^44),
q^17 + 38382239885400556956935947294616095966604658214298888*q^42 +
10154026913648909797114090345635511568428260780858433209 6*q^43 +
O(q^44),
q^18 + 83655625319721735583415547222825867484137297666810*q^42 +
19828245929344296797718054248129019851722311215433646 08*q^43 +
O(q^44),
q^19 + 17508632458124700492312373303553496016652018327 04*q^42 +
37067813808559835009768933560408227082662967512847123*q^43 +
O(q^44),
q^20 + 35120605897939259229581153117730682360354579200*q^42 +
66193972380615412771080423280607762489516454051840 0*q^43 + O(q^44),
q^21 + 67375862352629056696678054100808906648598132 8*q^42 +
11263977467064828122448102406911349240061917372536*q^43 + O(q^44),
q^22 + 12332929474164833973103939263658741128578676*q^42 +
18215643579666122002714207059676866876331786240 0*q^43 + O(q^44),
q^23 + 21484584977144910544028019643651826752262 4*q^42 +
27910547824653204915085228906071157166048188 20*q^43 + O(q^44),
q^24 + 35517273344689978826877861458923459841 92*q^42 +
40382528145750342577652598156921168374267904*q^43 + O(q^44),
q^25 + 55540319879597885046719328075750747000*q^42 +
54961188895393097785367442701111080370650 0*q^43 + O(q^44),
q^26 + 81857733277476433556042155377367470 9*q^42 +
70057744932227262875043508828594325422 08*q^43 + O(q^44),
q^27 + 11324148009261429522838694725846080*q^42 +
83215011032260877451619395681553576620*q^43 + O(q^44),
q^28 + 14635082097013761037222951603046 4*q^42 +
91565891280325423715507612818184601 6*q^43 + O(q^44),
q^29 + 17573223632989099800219933443 04*q^42 +
92687724958667320197045751679502 48*q^43 + O(q^44),
q^30 + 19479762749274799720470922200*q^42 +
85588826800946307979325630054400*q^43 + O(q^44),
q^31 + 19781786426988740913157795 2*q^42 +
71355432565469419034737592307 4*q^43 + O(q^44),
q^32 + 18232978655258810006054 40*q^42 +
53012134892111117771090165 76*q^43 + O(q^44),
q^33 + 15077819930384491326200*q^42 +
34500831474325683088421316*q^43 + O(q^44),
q^34 + 11022324801305253924 6*q^42 + 19212404668107058262835 2*q^43 +
O(q^44),
q^35 + 69834250829726496 0*q^42 + 8843208727571174542 95*q^43 +
O(q^44),
```

```
q^36 + 3728673187198848*q^42 + 31790744996948213 76*q^43 + O(q^44),
q^37 + 16064166491856*q^42 + 7977793210058584*q^43 + O(q^44),
q^38 + 51613491636*q^42 + 9898709483520*q^43 + O(q^44),
q^39 + 101474016*q^42 - 8249681358*q^43 + O(q^44),
q^40 + 15120*q^42 - 26869760*q^43 + O(q^44),
q^41 - 504*q^42 + 72252*q^43 + O(q^44)
]
Time: CPU 0.18 s, Wall: 0.38 s
```

Example: We compute a basis of level 1 modular forms quickly.

```
time victor_miller_basis(500,44)
```

```
[
1 +
5072259795933427154456176117061862204013935652569831200541465923641
2240000000*q^42 +
6357018695355722738844474405421846752652532909969671978222422444501
567892480000000*q^43 + O(q^44),
q +
1820953916246675278123830541924585557242265464032154412055351963326
005268912*q^42 +
2107265275702640444969498266205697402111464858080628365534348164668
64285875592744*q^43 + O(q^44),
q^2 +
6391675096009654638667224571231706878029161816840216585802137329659
9607170*q^42 +
6822036969308680862596569323096117827649662733279852566661011416900
034919235584*q^43 + O(q^44),
q^3 +
2192189578013015543567050685110290591656428300904809930974449643849
678464*q^42 +
2155489053350894181608950644787378929758862160972290421261438106245
19633770507*q^43 + O(q^44),
q^4 +
7341829111750098574503128560426807877175400146393584655054399665137
3152*q^42 +
6642112370263232609369696859142893169712766738518961064617960554349
808295424*q^43 + O(q^44),
q^5 +
2399345163751224221270204072624937184684020551559243392019307829485
000*q^42 +
1994663544541376003938369079358972703982053114450017921678707435454
44654000*q^43 + O(q^44),
q^6 +
7645857946345461201643921762341739723487632366695076684450960204350
2*q^42 +
5832995563071291867762396133797264867691130629453476988452339074975
193664*q^43 + O(q^44),
q^7 +
2373937376618735281006715238652310237602250871561182462718459252806
*q^42 +
1659614384817894205239356263129979096318214183456269564559512069815
63596*q^43 + O(q^44),
```

```
q^8 +
71757363281842108320199412308009723459466473713098051696483577168 0*
^42 +
45901706576611506485966471376200123184357710500519896439646655873 81
432*q^43 + O(q^44),
q^9 +
21097978656638973821842935074863615732891584166267234311463959768*q
42 +
12329474119159232921120934770607714622649284236799003660000940661 88
16*q^43 + O(q^44),
q^10 +
60282558466596050666946077462504388959601764793017232859681285 5*q^4
+
32130342389370471938456835301737021286486585700133504400846969896 96
*q^43 + O(q^44),
q^11 +
16722155356456567770279566492463031212095333641596485519761728*q^42
+
81147007802549277890656036340807008989178846855509257608064756471 1*
^43 + O(q^44),
q^12 +
44987026632494545982697068590581006975257772966718707327283 2*q^42 +
19838756530227102192290287014760783899678145436397712451183837184*q
43 + O(q^44),
q^13 +
11724381338692002823883693125834350389867346823548916115616*q^42 +
46892473900774484312556824441551674203833858082285132882494264 0*q^4
+ O(q^44),
q^14 +
29565034880352067998046815204858346674954185200920860680 8*q^42 +
10701878150767325620875506905520239829807213066206995699204096*q^43
+ O(q^44),
q^15 + 72043040197411503987585793266010814401022316865336500 80*q^42
+ 23548380093796936336656201157279274407901465984632760960391 0*q^43
+ O(q^44),
q^16 + 16940524740384456449330457921642272142624658568868864 0*q^42
49880557722173222925603772793666175138805779396408691916 80*q^43 +
O(q^44),
q^17 + 38382239885400556956935947294616095966604658214298 88*q^42 +
10154026913648909797114090345635511568428260780858433209 6*q^43 +
O(q^44),
q^18 + 83655625319721735583415547222825867484137297666810*q^42 +
19828245929344296797718054248129019851722311215433646 08*q^43 +
O(q^44),
q^19 + 17508632458124700492312373303553496016652018327 04*q^42 +
37067813808559835009768933560408227082662967512847123*q^43 +
O(q^44),
q^20 + 35120605897939259229581153117730682360354579200*q^42 +
66193972380615412771080423280607762489516454051840 0*q^43 + O(q^44),
q^21 + 67375862352629056696678054100808906648598132 8*q^42 +
11263977467064828122448102406911349240061917372536*q^43 + O(q^44),
q^22 + 12332929474164833973103939263658741128578676*q^42 +
```

```
1821564357966612200271420705967686687633178624 00*q^43 + O(q^44),
q^23 + 21484584977144910544028019643651826752262 4*q^42 +
279105478246532049150852289060711571660481882 0*q^43 + O(q^44),
q^24 + 35517273344689978826877861458923459841 92*q^42 +
403825281457503425776525981569211683742679 04*q^43 + O(q^44),
q^25 + 55540319879597885046719328075750747000*q^42 +
54961188895393097785367442701111080370650 0*q^43 + O(q^44),
q^26 + 81857733277476433556042155377367470 9*q^42 +
70057744932227262875043508828594325422 08*q^43 + O(q^44),
q^27 + 11324148009261429522838694725846080*q^42 +
83215011032260877451619395681553576620*q^43 + O(q^44),
q^28 + 14635082097013761037222951603046 4*q^42 +
91565891280325423715507612818184601 6*q^43 + O(q^44),
q^29 + 17573223632989099800219933443 04*q^42 +
92687724958667320197045751679502 48*q^43 + O(q^44),
q^30 + 19479762749274799720470922200*q^42 +
85588826800946307979325630054400*q^43 + O(q^44),
q^31 + 19781786426988740913157795 2*q^42 +
71355432565469419034737592307 4*q^43 + O(q^44),
q^32 + 18232978655258810006054 40*q^42 +
53012134892111117771090165 76*q^43 + O(q^44),
q^33 + 150778199303844913262 00*q^42 +
34500831474325683088421316*q^43 + O(q^44),
q^34 + 1102232480130525392 46*q^42 + 19212404668107058262 8352*q^43 +
O(q^44),
q^35 + 69834250829726496 0*q^42 + 8843208727571174542 95*q^43 +
O(q^44),
q^36 + 3728673187198848*q^42 + 3179074499694821376*q^43 + O(q^44),
q^37 + 16064166491856*q^42 + 7977793210058584*q^43 + O(q^44),
q^38 + 51613491636*q^42 + 9898709483520*q^43 + O(q^44),
q^39 + 101474016*q^42 - 8249681358*q^43 + O(q^44),
q^40 + 15120*q^42 - 26869760*q^43 + O(q^44),
q^41 - 504*q^42 + 72252*q^43 + O(q^44)
]
Time: CPU 0.15 s, Wall: 0.35 s
```

Example: More needs to be implemented!

```
M = ModularForms(GammaH(13,[2])); M.basis()
```

```
Traceback (click to the left for traceback)
...
NotImplementedError: q-expansion basis not implemented for "Cuspida
subspace of dimension 0 of Modular Forms space of dimension 1 for
Congruence Subgroup Gamma_H(13) with H generated by [2] of weight 2
over Rational Field"
```

Hecke Operators On Modular Forms

1. Implemented on all spaces (of integral weight)

Example: We compute the matrix of a Hecke operator on a space of forms for $\Gamma_1(N)$.

```
M = ModularForms(Gamma1(13)); M
```

```
Modular Forms space of dimension 13 for Congruence Subgroup
Gamma1(13) of weight 2 over Rational Field
```

```
T2 = M.hecke_operator(2); T2
```

```
Hecke operator T_2 on Modular Forms space of dimension 13 for
Congruence Subgroup Gamma1(13) of weight 2 over Rational Field
```

```
show(T2.charpoly('x'))
```

$$x^{13} - 12x^{12} + 60x^{11} - 153x^{10} + 177x^9 + 18x^8 - 211x^7 - 177x^6 + 1620x^5 - 4131x^4 + 4779x^3 +$$

```
show(factor(charpoly(T2)))
```

$$(x - 3) \cdot (x - 1) \cdot (x + 1) \cdot (x^2 - 5x + 7) \cdot (x^2 - 4x + 7) \cdot (x^2 - 3x + 3) \cdot (x^2 + 3) \cdot (x^2 + 3x +$$

```
show(M.eisenstein_subspace().hecke_matrix(2))
```

$$\begin{pmatrix} -1461 & 0 & -\frac{4680}{19} & 0 & \frac{96408}{19} & 0 & -\frac{471744}{19} & -\frac{388440}{19} & \frac{443352}{19} & -\frac{939120}{19} & \frac{1332552}{19} \\ -828 & 0 & -\frac{2640}{19} & 0 & \frac{54612}{19} & 0 & -\frac{267385}{19} & -\frac{219842}{19} & \frac{251027}{19} & -\frac{532306}{19} & \frac{755325}{19} \\ -344 & 1 & -56 & 0 & 1182 & 0 & -5823 & -4779 & 5431 & -11550 & 16376 \\ -56 & 0 & -\frac{204}{19} & 0 & \frac{3792}{19} & 0 & -\frac{18211}{19} & -\frac{15203}{19} & \frac{17570}{19} & -\frac{36661}{19} & \frac{52017}{19} \\ 70 & 0 & \frac{215}{19} & 0 & -\frac{4410}{19} & 0 & \frac{22451}{19} & \frac{18130}{19} & -\frac{20222}{19} & \frac{43745}{19} & -\frac{61965}{19} \\ 96 & 0 & \frac{286}{19} & 0 & -\frac{6226}{19} & 0 & \frac{30877}{19} & \frac{25201}{19} & -\frac{28539}{19} & \frac{60950}{19} & -\frac{86432}{19} \\ 76 & 0 & \frac{240}{19} & 1 & -\frac{4944}{19} & 0 & \frac{24496}{19} & \frac{20053}{19} & -\frac{22736}{19} & \frac{48445}{19} & -\frac{68583}{19} \\ 46 & 0 & \frac{150}{19} & 0 & -\frac{3090}{19} & 0 & \frac{14892}{19} & \frac{12355}{19} & -\frac{14210}{19} & \frac{29872}{19} & -\frac{42444}{19} \\ 22 & 0 & \frac{80}{19} & 0 & -\frac{1477}{19} & 0 & \frac{7133}{19} & \frac{5975}{19} & -\frac{6901}{19} & \frac{14489}{19} & -\frac{20524}{19} \\ 8 & 0 & \frac{34}{19} & 0 & -\frac{594}{19} & 0 & \frac{2652}{19} & \frac{2290}{19} & -\frac{2770}{19} & \frac{5575}{19} & -\frac{7938}{19} \\ 2 & 0 & \frac{12}{19} & 0 & -\frac{156}{19} & 1 & \frac{651}{19} & \frac{597}{19} & -\frac{715}{19} & \frac{1420}{19} & -\frac{2007}{19} \end{pmatrix}$$

```
show(M.cuspidal_subspace().hecke_matrix(2))
```

$$\begin{pmatrix} 0 & -3 \\ 1 & -3 \end{pmatrix}$$

```
M = ModularForms(Gamma1(13)); E = M.eisenstein_subspace(); E
```

```
Eisenstein subspace of dimension 11 of Modular Forms space of
dimension 13 for Congruence Subgroup Gamma1(13) of weight 2 over
Rational Field
```

Eisenstein Series

1. Explicit enumeration algorithm (which involves pairs of Dirichlet characters)
2. General and hopefully fast

Example: We compute the Eisenstein series in a space of modular forms, along with the "Eisenstein parameters".

```
es = E.eisenstein_series()
for f in es: view(f)
```

$\frac{1}{2} + q + 3q^2 + 4q^3 + 7q^4 + 6q^5 + O(q^6)$
$-\frac{7}{13}\zeta_6 - \frac{11}{13} + q + (2\zeta_6 + 1)\,q^2 + (-3\zeta_6 + 1)\,q^3 + (6\zeta_6 - 3)\,q^4 - 4q^5 + O(q^6)$
$q + (\zeta_6 + 2)\,q^2 + (-\zeta_6 + 3)\,q^3 + (3\zeta_6 + 3)\,q^4 + 4q^5 + O(q^6)$
$-\zeta_6 + q + (2\zeta_6 - 1)\,q^2 + (3\zeta_6 - 2)\,q^3 + (-2\zeta_6 - 1)\,q^4 + 6q^5 + O(q^6)$
$q + (\zeta_6 + 1)\,q^2 + (\zeta_6 + 2)\,q^3 + (\zeta_6 + 2)\,q^4 + 6q^5 + O(q^6)$
$-1 + q - q^2 + 4q^3 + 3q^4 - 4q^5 + O(q^6)$
$q + q^2 + 4q^3 + 3q^4 + 4q^5 + O(q^6)$
$\zeta_6 - 1 + q + (-2\zeta_6 + 1)\,q^2 + (-3\zeta_6 + 1)\,q^3 + (2\zeta_6 - 3)\,q^4 + 6q^5 + O(q^6)$
$q + (-\zeta_6 + 2)\,q^2 + (-\zeta_6 + 3)\,q^3 + (-\zeta_6 + 3)\,q^4 + 6q^5 + O(q^6)$
$\frac{7}{13}\zeta_6 - \frac{18}{13} + q + (-2\zeta_6 + 3)\,q^2 + (3\zeta_6 - 2)\,q^3 + (-6\zeta_6 + 3)\,q^4 - 4q^5 + O(q^6)$
$q + (-\zeta_6 + 3)\,q^2 + (\zeta_6 + 2)\,q^3 + (-3\zeta_6 + 6)\,q^4 + 4q^5 + O(q^6)$

```
for f in es: print f.parameters()
```

```
([1], [1], 13)
([1], [zeta6], 1)
([zeta6], [1], 1)
([1], [zeta6 - 1], 1)
([zeta6 - 1], [1], 1)
([1], [-1], 1)
([-1], [1], 1)
([1], [-zeta6], 1)
([-zeta6], [1], 1)
([1], [-zeta6 + 1], 1)
([-zeta6 + 1], [1], 1)
```

```
time k = es[1].qexp(1000)
```

```
Time: CPU 0.47 s, Wall: 0.48 s
```

Eta Products

1. New(-ish) code by David Loeffler
2. Basis for space of eta product of a given level
3. Q-expansions of eta products

Example: We compute a basis for a specific level, and output q-expansions for each element.

```
E = EtaGroup(11); E
```

```
    Group of eta products on X_0(11)
```

```
f = E.basis()[0]; f
```

```
    Eta product of level 11 : (eta_1)^12 (eta_11)^-12
```

```
f.degree()
```

```
    5
```

```
show(f.q_expansion(10))    # NB -- perhaps this should go to
O(q^10), for consistency with other modular forms functions
```

$$\frac{1}{q^5} + \frac{-12}{q^4} + \frac{54}{q^3} + \frac{-88}{q^2} + \frac{-99}{q} + 540 - 418q - 648q^2 + 594q^3 + 836q^4 + O(q^5)$$

```
for f in EtaGroup(12).basis():
   view(f.q_expansion(10))
```

$$1 - 2q - 2q^2 + 2q^3 + 6q^4 + 4q^5 - 6q^6 - 16q^7 - 10q^8 + 14q^9 + O(q^{10})$$
$$1 + 2q + 2q^2 - 2q^3 - 6q^4 - 4q^5 + 6q^6 + 16q^7 + 10q^8 - 14q^9 + O(q^{10})$$
$$q + 3q^2 + 7q^3 + 15q^4 + 30q^5 + 57q^6 + 104q^7 + 183q^8 + 313q^9 + 522q^{10} + O(q^{11})$$
$$1 - q + 2q^3 - 3q^5 + 4q^7 - 7q^9 + O(q^{10})$$
$$1 + 6q + 18q^2 + 42q^3 + 90q^4 + 180q^5 + 342q^6 + 624q^7 + 1098q^8 + 1878q^9 + O(q^{10})$$

```
for f in EtaGroup(12).basis():
   view(f.q_expansion(10))
```

$$1 - 2q - 2q^2 + 2q^3 + 6q^4 + 4q^5 - 6q^6 - 16q^7 - 10q^8 + 14q^9 + O(q^{10})$$
$$1 + 2q + 2q^2 - 2q^3 - 6q^4 - 4q^5 + 6q^6 + 16q^7 + 10q^8 - 14q^9 + O(q^{10})$$
$$q + 3q^2 + 7q^3 + 15q^4 + 30q^5 + 57q^6 + 104q^7 + 183q^8 + 313q^9 + 522q^{10} + O(q^{11})$$
$$1 - q + 2q^3 - 3q^5 + 4q^7 - 7q^9 + O(q^{10})$$
$$1 + 6q + 18q^2 + 42q^3 + 90q^4 + 180q^5 + 342q^6 + 624q^7 + 1098q^8 + 1878q^9 + O(q^{10})$$

```

```

```

```

Generators For Ring Of Modular Forms

1. Implementation of straightforward linear algebra algorithms to write forms in terms of forms of lower weight until we appear to get nothing new

2. Useful for experimentation

Example: Here we see that for $\Gamma_0(11)$ taking forms in weights 2 and 4 is enough to generate everything up to weight 12 (and probably everything else).

```
import sage.modular.modform.find_generators as fg
v = fg.modform_generators(Gamma0(11), 12)
for z in v: view(z)
```

$(2, q - 2q^2 - q^3 + 2q^4 + q^5 + 2q^6 - 2q^7 - 2q^9 - 2q^{10} + q^{11} - 2q^{12} + 4q^{13} + 4q^{14} - q^{15} - 4q^{16} -$
$(2, 1 + \frac{12}{5}q + \frac{36}{5}q^2 + \frac{48}{5}q^3 + \frac{84}{5}q^4 + \frac{72}{5}q^5 + \frac{144}{5}q^6 + \frac{96}{5}q^7 + 36q^8 + \frac{156}{5}q^9 + \frac{216}{5}q^{10} + \frac{12}{5}q^{11} + \frac{336}{5}q^{12}$
$(4, 1 + 240q^{11} + 2160q^{22} + O(q^{24}))$

```
dimension_modular_forms(11,2)
```

2

```
dimension_modular_forms(11,4)
```

4

Example: We rediscover the well-known fact that E_4 and E_6 generate at level 1.

```
import sage.modular.modform.find_generators as fg
v = fg.modform_generators(Gamma0(1), 30)
for z in v: view(z)
```

$(4, 1 + 240q + 2160q^2 + 6720q^3 + 17520q^4 + 30240q^5 + O(q^6))$
$(6, 1 - 504q - 16632q^2 - 122976q^3 - 532728q^4 - 1575504q^5 + O(q^6))$

Example: Weight 2 is enough for level 4.

```
import sage.modular.modform.find_generators as fg
v = fg.modform_generators(Gamma0(4), 20)
for z in v: view(z)
```

$(2, 1 + 24q^2 + 24q^4 + 96q^6 + 24q^8 + 144q^{10} + 96q^{12} + 192q^{14} + 24q^{16} + 312q^{18} + 144q^{20} + O(q^$
$(2, q + 4q^3 + 6q^5 + 8q^7 + 13q^9 + 12q^{11} + 14q^{13} + 24q^{15} + 18q^{17} + 20q^{19} + 32q^{21} + O(q^{22}))$

```

```

```

```

Numerical Computation Of Newforms

1. A very fast purely numerical algorithm for computing eigenvalues of newforms to double precision (using numpy)
2. Useful for certain types of computational experiments

Example: We compute numerical eigenvalues for level 389.

```
n = numerical_eigenforms(389); n
```

```
Numerical Hecke eigenvalues for Congruence Subgroup Gamma0(389) of
weight 2
```

```
n.systems_of_eigenvalues(97)
```

```
[
[-2.77612549543, 1.77136382566, -3.66769733872, -2.25035603467,
3.22836879279, 1.501326093, 5.5538125828, 5.56551890432,
3.81141485303, -1.35077598927, 4.44391838222, -0.376574061517,
11.5037747108, 6.42956091853, -10.1466868327, 4.07828917203,
-3.16351039744, 2.77048614067, -5.08320791353, -2.1389725934,
-4.90383511255, -5.50872590675, 4.56777812625, 10.6643317329],
[-2.6575528059, -2.19226519393, -0.219842909208, 2.10301598314,
0.480308031326, -0.03798622437, -4.07540594497, -1.49287135229,
-3.11296829003, 9.01273889204, 7.41770435325, 9.10603017059,
3.38501521218, -0.555667670312, 8.30142299727, -8.61015766969,
5.10599667928, 5.05964130144, -4.58019295664, 10.1761172025,
5.75492373853, 14.5667799834, 8.9211063959, -6.73056890998],
[-2.51395267946, 0.116172717926, 2.95148816349, -3.09324839047,
-4.0818288951, -0.502948449283, 1.68380947804, -6.61803398883,
5.28317512229, -8.76288985498, -7.06664363675, 4.11281270044,
10.8362718252, -10.7779260159, 6.6155853399, -10.7819330237,
1.6388538933, -6.29471589691, 8.84331226889, -0.626375037367,
6.6656516495, -14.2389097492, -5.84982066656, -2.7642513252],
[-2.37490913972, 2.46658426036, 1.28465910961, 4.22851393095,
-0.425189042273, 2.51815419223, -1.39814151287, -1.74376256477,
-4.60597966379, -8.10030606181, -0.660784668504, -2.93177495528,
-0.713761738299, 6.05623131582, 4.45019187436, 11.5274445212,
-4.72561394668, 11.9273366771, -15.3989380314, -7.7458459856,
-3.67454668853, 14.0930085139, -10.1680937047, 10.4238238719],
[-2.0, -2.0, -3.0, -5.0, -4.0, -3.0, -6.0, 5.0, -4.0, -6.0, 4.0,
-8.0, -3.0, 12.0, -2.0, -6.0, 3.0, -8.0, -5.0, -10.0, -7.0, -13.0,
-12.0, -8.0],
[-1.91638399542, -0.605432092793, -1.57895835187, 0.888856534306,
6.49708279052, -2.4973616798, -1.50810305103, -4.38196601125,
5.90532824676, -6.91573276053, -7.01997453181, -0.463001390245,
-8.97393409285, -4.51308400875, -12.6656394788, 8.83328725209,
4.35979185348, -2.8518690959, -14.308620303, 2.21112114198,
14.7933120779, 10.5307971937, -2.4645909561, -14.4533746748],
[-1.76383376439, 1.58760939399, 2.80546537649, -2.97474932766,
6.19933826817, 1.83550359531, -3.25849557754, 6.69067881834,
-5.18988815324, 6.47129445559, 0.24146126473, 9.47912657955,
1.19031200846, 6.89876677549, 4.5815369482, -10.8638508596,
0.627868887644, -8.9882839165, 2.07257898186, -5.35980822801,
4.17510932725, -4.03003290045, 4.0648943419, 7.4979861475],
[-1.67513087057, 1.67513087057, -1.80606343353, -1.0,
-1.19393656647, -3.0, 4.15632517466, -6.76845204171, -7.0253926117,
6.31265034932, 9.59990808693, -9.66291209045, -4.50658691579,
-10.2496463458, 11.9247772164, -6.2496463458, -10.9247772164,
2.71274226238, 6.43136413216, 7.0253926117, 2.73813487408,
```

```
-10.3805789088, 15.9877812199, 2.45087713647],
[-1.64085858659, 0.244124560347, -2.84950368583, 2.53341280273,
-4.68458334658, 6.76553726844, -2.63161598846, 4.83619082784,
8.84089747051, 7.77955259936, -1.91817908531, 7.09663546177,
6.26639192124, -6.8489044225, 9.65258348908, 8.77232052752,
-1.84484424204, -0.523043498362, 9.28352147919, -9.75236955291,
-4.67534479555, 1.11075764675, -5.16165654669, -7.18442042058],
[-1.63341139845, -1.13812122701, 3.42530615528, 3.25152890995,
-1.70002616923, -2.28544873029, 2.81876352205, 3.46348312073,
0.978948623313, 0.112934678403, 10.4127560544, -2.17496830343,
-4.09919495898, 6.75430289827, -10.6910219159, -1.01671188262,
6.29808754848, 8.95551203956, 14.5002154713, 15.2011539155,
-10.0611918029, -3.16494239143, -9.35860373445, -1.64622974008],
[-1.41421356237, -3.41421356237, -1.0, 1.82842712475, -2.0,
-1.82842712475, 6.82842712475, -3.82842712475, 3.41421356237,
1.17157287525, -9.41421356237, 4.82842712475, 1.82842712475,
9.07106781187, 3.65685424949, -8.58578643763, -5.0, -9.07106781187,
-5.82842712475, -12.2426406871, -8.65685424949, -0.171572875254,
-3.07106781187, 4.58578643763],
[-0.871374908306, 3.14416780833, 0.478096798347, -2.49522762219,
-1.83187996416, 6.0488846412, 6.03511514148, 0.476710922342,
7.45140678661, -8.36831171527, -1.90956392741, -9.35477750121,
-5.73104695796, 8.9675520394, -5.67814733453, -4.34896468034,
3.35983553114, -1.47523895662, 13.2828555924, -6.10004754122,
6.89190753244, -7.37259423077, -6.4455228606, -11.3165600607],
[-0.855987000806, -2.31225503915, 2.69727110039, -1.65723740022,
-0.785836781863, 6.69005993944, -7.67559871537, -6.61803398875,
-3.88202851607, 1.41918203578, -3.52875759095, -3.93512176005,
-7.44644607427, -5.80402897992, 0.342658079668, 3.84566253263,
9.41263204644, -6.2624535223, -0.725607104719, -3.92236715064,
0.00902229571127, -0.458744529044, 10.012280365, -8.47654284065],
[-0.539188872811, 0.539188872811, 0.709275359437, -1.0,
-3.70927535944, -3.0, -0.630897613815, 3.04944833269,
-3.61756661843, -3.26179522763, -6.14115707369, 2.18341748201,
2.55251986819, 3.21953481931, -2.6803459465, 7.21953481931,
3.6803459465, 8.87936184606, -15.2328658147, 3.61756661843,
5.4969284645, 4.46799905156, 5.27739364518, 11.3762903106],
[-0.505134199859, 2.75550226843, -2.8223486167, 4.80936476762,
2.72015712937, -4.4809435164, 1.35247379833, 6.19107426741,
-0.152200927175, 2.71381009216, 0.733511464822, 10.0452718592,
-7.31146249161, -1.65960460939, -12.1666840399, -11.8803552689,
5.67722010887, -3.82675959371, -9.88370379041, -7.69135589487,
-9.78301090562, -7.58924302182, 10.4956821055, 3.5396769878],
[-0.23797053375, -1.44573005298, -3.7003019925, 0.279204365642,
2.39532461859, 2.33467103751, 3.66443233207, -1.21553403291,
-7.55986608399, 1.26514138249, 0.890900209463, -0.890399366888,
3.38491520952, 9.22150102268, -1.95749403329, -4.08813793439,
4.6572598674, 7.11901112203, 14.3709176719, -0.557089464283,
14.8365096679, 11.7759630048, -6.61063426701, 4.21593302323],
[-0.229804177743, -2.17870094903, 1.03033706198, -3.19321752468,
4.48475666353, -5.42691406712, 1.01176683523, 5.84292587462,
8.56618149569, 3.95803257705, 3.10193432071, -3.53115855188,
```

```
8.37059485892, -9.72562391657, 7.81918712499, -6.33749038824,
-0.275350409001, 2.46927075261, 15.1385308497, -5.44774029171,
0.569887468038, -4.47295272406, 3.91988377311, -11.6495204601],
[0.251350911656, 1.7271506717, -2.5718072652, -3.23222659807,
2.26349455524, 0.00663464106779, -8.16553210562, -4.38196601125,
4.44652359373, -1.48553245419, 3.60431332178, 3.55620016327,
-2.23229171159, 1.99012071247, 4.68966723204, 10.2119364036,
-9.22131666279, -8.14783822203, 7.28027464306, 6.44470937726,
-14.5477424149, -7.53343977119, -14.3525149844, 8.16071878599],
[0.359690366264, 0.153877958327, 2.13906891638, 2.51164679512,
2.75224443071, 3.56861425251, 4.66055672689, -1.29611362997,
1.0855275727, -1.92973159995, 1.89542468905, 5.10140261609,
-4.98390095291, -1.91495649719, 6.6841865484, -0.555149624785,
-10.6007417916, -8.25423311417, -7.16734744137, -9.48165081764,
-9.74029288083, -1.70409844665, 7.40804107385, -10.7596862438],
[0.751905691471, -1.42195166752, -0.79465729761, 4.9865537681,
-2.84053825564, -5.33300952398, -4.24427874022, -6.61803398875,
2.21688738259, 9.2879797291, -5.34887068244, -11.7399968392,
2.31837818183, -0.980350903088, 0.749960512933, 4.28379464846,
8.45509186921, 2.26537335174, 1.73639680233, -1.86766567686,
-3.67467394495, -8.57285555325, -11.8362217774, 12.712930121],
[0.919097377566, 2.63205023032, 2.12810515973, 1.91405539126,
-5.06352552347, 2.83700418501, -6.96727189823, 3.56016051376,
-5.27818300486, 3.32499358201, 9.25794298899, 0.60392256479,
1.05891447447, -8.31455123614, 0.351302464724, -12.0534865326,
8.44616682281, -6.27043834005, 7.7118056518, 10.3956352153,
13.2842533461, 1.96549497561, -11.9906438279, -9.20888634225],
[1.1004663882, -3.08431118755, -0.916066900677, 1.93532299367,
1.17692516686, 5.9348921803, -4.05365599273, 8.26598698511,
4.51585062467, -6.27746686689, 7.62802712835, -0.223077950462,
-0.427949101218, 8.71860936663, -5.76115297289, 7.96608529988,
12.7373826484, 11.5203435295, -13.6175312928, -1.06728196091,
-7.51063699016, -1.04762320936, 2.42044389794, 5.54054680324],
[1.28306707251, -2.50368459016, 2.29666365082, -1.89269791374,
-3.05237341326, -3.36337492752, 3.90970313414, -4.38196601125,
-8.96988582924, -7.54300669528, 5.35993312003, 5.46910712573,
3.49802187194, 5.08526919504, 2.26776831421, 9.60725218686,
-13.6450529994, -12.7084966146, 10.1742436937, 11.7605773458,
2.75443033698, 7.27315240877, -6.50913198075, -1.17948006621],
[1.33119677362, 2.58121334461, 1.43502873917, -1.54209925451,
4.81115398417, -4.11917366652, 4.49468079795, -1.06697798891,
-1.69456947267, -2.03271907546, -7.06214833178, -2.86690895381,
6.560751973, -2.06226456216, -2.66809704575, -2.09607806787,
10.0412959748, 10.9599164961, 0.36074265656, 3.35533700895,
-11.6113501099, 12.6750641708, 2.95607280869, 9.20603322989],
[1.41421356237, -0.585786437627, -1.0, -3.82842712475, -2.0,
3.82842712475, 1.17157287525, 1.82842712475, 0.585786437627,
6.82842712475, -6.58578643763, -0.828427124746, -3.82842712475,
-5.07106781187, -7.65685424949, -11.4142135624, -5.0, 5.07106781187
-0.171572875254, -3.75735931288, 2.65685424949, -5.82842712475,
11.0710678119, 7.41421356237],
[1.96404766408, -0.655927501903, 4.38447834155, 0.550492107123,
```

```
1.61586515855, -0.758447882785, -7.39647315034, 4.65523283475,
4.72765349953, -2.07911138285, -8.4438657128, -5.70772376386,
6.28815421933, 0.331082082995, -1.07131080137, -2.32629706064,
-6.35272756043, -3.54199395359, -7.8794061205, 12.5688517375,
-1.98900457357, 10.0244746139, -7.25902583048, 9.20303519087],
[2.01219875163, 1.46167794756, -1.98477252807, 3.65228285226,
1.83568252865, 3.62766028749, -1.06801639125, -2.32399054939,
-0.937033806048, -3.57807068207, -3.11664693334, -0.172364562473,
2.09370207991, 3.97112777169, -6.38576357878, 4.19366069145,
-12.1719369855, 4.30092841806, 12.6924280796, 8.08373809164,
16.5354155592, -3.79304546648, 2.15899409967, 3.82072848861],
[2.21431974338, -2.21431974338, -3.90321192591, -1.0,
0.903211925911, -3.0, -1.52542756084, -5.28099629098, 4.64295923013
-5.05085512169, -1.45875101324, 9.47949460844, 8.9540670476,
-6.96988847351, 4.75556873014, -2.96988847351, -3.75556873014,
2.40789589155, -14.1984983175, -4.64295923013, -9.23506333858,
-13.0874201428, 0.734825134934, -9.82716744702],
[2.27802573453, 1.28538083652, 1.3495702151, -0.260450965384,
-4.38552066553, -3.92478260065, 3.02119872197, -3.78799924911,
8.05293064075, 6.58402600579, 9.63200598368, 7.17714173857,
-4.58554122679, 12.0240550953, 1.78346135307, 0.252374082053,
4.67895603949, 3.23074951817, -9.30105002109, -8.90538822057,
3.66847264244, -9.44787358005, 2.39527432312, -8.93030173264],
[2.40575084198, 3.25552915896, -3.7458320007, -3.2672901519,
-2.55490600149, -1.17291262291, -1.08217743888, 8.23446651588,
-3.14885722831, 3.39887116732, 1.18030379195, -4.92704914591,
-7.11775231833, -3.35290005292, 9.59567053103, 3.21267959785,
-9.04282028331, 12.8226144607, -1.64518347721, -2.81567546644,
4.52802809076, 13.2451186563, -2.03339801107, 16.8335714322],
[2.64734306658, -1.68384110024, 0.049693845708, 3.75820799783,
-5.06870746286, -0.380076581696, 2.71895234787, 8.66282505979,
-6.75029930986, -4.97512385351, -7.99736208564, 4.48216779583,
-8.02771252726, -1.67285096662, 2.30038827689, -5.3919424197,
2.20514902551, -0.469530683087, 5.60354089378, -4.95848366699,
-6.42106254071, 10.1595863502, 10.3479153946, -15.9233195645],
[2.67315804566, 0.0398156192124, 0.396556253047, -3.54365801634,
4.01421340286, 2.61443815971, -1.40049891138, -2.51800527757,
-6.6009656268, -5.92977820517, 10.2726601131, -10.9349216697,
8.89579560561, -4.26546535302, 3.00642694718, 9.56576849734,
13.3423264823, -1.78628839978, 0.539423716883, -1.75912348678,
0.125769027693, 5.51488396211, 8.37149244244, -7.59617343345],
[3.0, 4.0, 6.0, 8.0, 12.0, 14.0, 18.0, 20.0, 24.0, 30.0, 32.0, 38.0
42.0, 44.0, 48.0, 54.0, 60.0, 62.0, 68.0, 72.0, 74.0, 80.0, 84.0,
90.0]
]
```

```
n.systems_of_eigenvalues(97)
```

```
[
[-2.77612549543, 1.77136382566, -3.66769733872, -2.25035603467,
3.22836879279, 1.501326093, 5.5538125828, 5.56551890432,
3.81141485303, -1.35077598927, 4.44391838222, -0.376574061517,
11.5037747108, 6.42956091853, -10.1466868327, 4.07828917203,
```

```
-3.16351039744, 2.77048614067, -5.08320791353, -2.1389725934,
-4.90383511255, -5.50872590675, 4.56777812625, 10.6643317329],
[-2.6575528059, -2.19226519393, -0.219842909208, 2.10301598314,
0.480308031326, -0.03798622437, -4.07540594497, -1.49287135229,
-3.11296829003, 9.01273889204, 7.41770435325, 9.10603017059,
3.38501521218, -0.555667670312, 8.30142299727, -8.61015766969,
5.10599667928, 5.05964130144, -4.58019295664, 10.1761172025,
5.75492373853, 14.5667799834, 8.9211063959, -6.73056890998],
[-2.51395267946, 0.116172717926, 2.95148816349, -3.09324839047,
-4.0818288951, -0.502948449283, 1.68380947804, -6.61803398883,
5.28317512229, -8.76288985498, -7.06664363675, 4.11281270044,
10.8362718252, -10.7779260159, 6.6155853399, -10.7819330237,
1.6388538933, -6.29471589691, 8.84331226889, -0.626375037367,
6.6656516495, -14.2389097492, -5.84982066656, -2.7642513252],
[-2.37490913972, 2.46658426036, 1.28465910961, 4.22851393095,
-0.425189042273, 2.51815419223, -1.39814151287, -1.74376256477,
-4.60597966379, -8.10030606181, -0.660784668504, -2.93177495528,
-0.713761738299, 6.05623131582, 4.45019187436, 11.5274445212,
-4.72561394668, 11.9273366771, -15.3989380314, -7.7458459856,
-3.67454668853, 14.0930085139, -10.1680937047, 10.4238238719],
[-2.0, -2.0, -3.0, -5.0, -4.0, -3.0, -6.0, 5.0, -4.0, -6.0, 4.0,
-8.0, -3.0, 12.0, -2.0, -6.0, 3.0, -8.0, -5.0, -10.0, -7.0, -13.0,
-12.0, -8.0],
[-1.91638399542, -0.605432092793, -1.57895835187, 0.888856534306,
6.49708279052, -2.4973616798, -1.50810305103, -4.38196601125,
5.90532824676, -6.91573276053, -7.01997453181, -0.463001390245,
-8.97393409285, -4.51308400875, -12.6656394788, 8.83328725209,
4.35979185348, -2.8518690959, -14.308620303, 2.21112114198,
14.7933120779, 10.5307971937, -2.4645909561, -14.4533746748],
[-1.76383376439, 1.58760939399, 2.80546537649, -2.97474932766,
6.19933826817, 1.83550359531, -3.25849557754, 6.69067881834,
-5.18988815324, 6.47129445559, 0.24146126473, 9.47912657955,
1.19031200846, 6.89876677549, 4.5815369482, -10.8638508596,
0.627868887644, -8.9882839165, 2.07257898186, -5.35980822801,
4.17510932725, -4.03003290045, 4.0648943419, 7.4979861475],
[-1.67513087057, 1.67513087057, -1.80606343353, -1.0,
-1.19393656647, -3.0, 4.15632517466, -6.76845204171, -7.0253926117,
6.31265034932, 9.59990808693, -9.66291209045, -4.50658691579,
-10.2496463458, 11.9247772164, -6.2496463458, -10.9247772164,
2.71274226238, 6.43136413216, 7.0253926117, 2.73813487408,
-10.3805789088, 15.9877812199, 2.45087713647],
[-1.64085858659, 0.244124560347, -2.84950368583, 2.53341280273,
-4.68458334658, 6.76553726844, -2.63161598846, 4.83619082784,
8.84089747051, 7.77955259936, -1.91817908531, 7.09663546177,
6.26639192124, -6.8489044225, 9.65258348908, 8.77232052752,
-1.84484424204, -0.523043498362, 9.28352147919, -9.75236955291,
-4.67534479555, 1.11075764675, -5.16165654669, -7.18442042058],
[-1.63341139845, -1.13812122701, 3.42530615528, 3.25152890995,
-1.70002616923, -2.28544873029, 2.81876352205, 3.46348312073,
0.978948623313, 0.112934678403, 10.4127560544, -2.17496830343,
-4.09919495898, 6.75430289827, -10.6910219159, -1.01671188262,
6.29808754848, 8.95551203956, 14.5002154713, 15.2011539155,
```

```
-10.0611918029, -3.16494239143, -9.35860373445, -1.64622974008],
[-1.41421356237, -3.41421356237, -1.0, 1.82842712475, -2.0,
-1.82842712475, 6.82842712475, -3.82842712475, 3.41421356237,
1.17157287525, -9.41421356237, 4.82842712475, 1.82842712475,
9.07106781187, 3.65685424949, -8.58578643763, -5.0, -9.07106781187,
-5.82842712475, -12.2426406871, -8.65685424949, -0.171572875254,
-3.07106781187, 4.58578643763],
[-0.871374908306, 3.14416780833, 0.478096798347, -2.49522762219,
-1.83187996416, 6.0488846412, 6.03511514148, 0.476710922342,
7.45140678661, -8.36831171527, -1.90956392741, -9.35477750121,
-5.73104695796, 8.9675520394, -5.67814733453, -4.34896468034,
3.35983553114, -1.47523895662, 13.2828555924, -6.10004754122,
6.89190753244, -7.37259423077, -6.4455228606, -11.3165600607],
[-0.855987000806, -2.31225503915, 2.69727110039, -1.65723740022,
-0.785836781863, 6.69005993944, -7.67559871537, -6.61803398875,
-3.88202851607, 1.41918203578, -3.52875759095, -3.93512176005,
-7.44644607427, -5.80402897992, 0.342658079668, 3.84566253263,
9.41263204644, -6.2624535223, -0.725607104719, -3.92236715064,
0.00902229571127, -0.458744529044, 10.012280365, -8.47654284065],
[-0.539188872811, 0.539188872811, 0.709275359437, -1.0,
-3.70927535944, -3.0, -0.630897613815, 3.04944833269,
-3.61756661843, -3.26179522763, -6.14115707369, 2.18341748201,
2.55251986819, 3.21953481931, -2.6803459465, 7.21953481931,
3.6803459465, 8.87936184606, -15.2328658147, 3.61756661843,
5.4969284645, 4.46799905156, 5.27739364518, 11.3762903106],
[-0.505134199859, 2.75550226843, -2.8223486167, 4.80936476762,
2.72015712937, -4.4809435164, 1.35247379833, 6.19107426741,
-0.152200927175, 2.71381009216, 0.733511464822, 10.0452718592,
-7.31146249161, -1.65960460939, -12.1666840399, -11.8803552689,
5.67722010887, -3.82675959371, -9.88370379041, -7.69135589487,
-9.78301090562, -7.58924302182, 10.4956821055, 3.5396769878],
[-0.23797053375, -1.44573005298, -3.7003019925, 0.279204365642,
2.39532461859, 2.33467103751, 3.66443233207, -1.21553403291,
-7.55986608399, 1.26514138249, 0.890900209463, -0.890399366888,
3.38491520952, 9.22150102268, -1.95749403329, -4.08813793439,
4.6572598674, 7.11901112203, 14.3709176719, -0.557089464283,
14.8365096679, 11.7759630048, -6.61063426701, 4.21593302323],
[-0.229804177743, -2.17870094903, 1.03033706198, -3.19321752468,
4.48475666353, -5.42691406712, 1.01176683523, 5.84292587462,
8.56618149569, 3.95803257705, 3.10193432071, -3.53115855188,
8.37059485892, -9.72562391657, 7.81918712499, -6.33749038824,
-0.275350409001, 2.46927075261, 15.1385308497, -5.44774029171,
0.569887468038, -4.47295272406, 3.91988377311, -11.6495204601],
[0.251350911656, 1.7271506717, -2.5718072652, -3.23222659807,
2.26349455524, 0.00663464106779, -8.16553210562, -4.38196601125,
4.44652359373, -1.48553245419, 3.60431332178, 3.55620016327,
-2.23229171159, 1.99012071247, 4.68966723204, 10.2119364036,
-9.22131666279, -8.14783822203, 7.28027464306, 6.44470937726,
-14.5477424149, -7.53343977119, -14.3525149844, 8.16071878599],
[0.359690366264, 0.153877958327, 2.13906891638, 2.51164679512,
2.75224443071, 3.56861425251, 4.66055672689, -1.29611362997,
1.0855275727, -1.92973159995, 1.89542468905, 5.10140261609,
```

```
-4.98390095291, -1.91495649719, 6.6841865484, -0.555149624785,
-10.6007417916, -8.25423311417, -7.16734744137, -9.48165081764,
-9.74029288083, -1.70409844665, 7.40804107385, -10.7596862438],
[0.751905691471, -1.42195166752, -0.79465729761, 4.9865537681,
-2.84053825564, -5.33300952398, -4.24427874022, -6.61803398875,
2.21688738259, 9.2879797291, -5.34887068244, -11.7399968392,
2.31837818183, -0.980350903088, 0.749960512933, 4.28379464846,
8.45509186921, 2.26537335174, 1.73639680233, -1.86766567686,
-3.67467394495, -8.57285555325, -11.8362217774, 12.712930121],
[0.919097377566, 2.63205023032, 2.12810515973, 1.91405539126,
-5.06352552347, 2.83700418501, -6.96727189823, 3.56016051376,
-5.27818300486, 3.32499358201, 9.25794298899, 0.60392256479,
1.05891447447, -8.31455123614, 0.351302464724, -12.0534865326,
8.44616682281, -6.27043834005, 7.7118056518, 10.3956352153,
13.2842533461, 1.96549497561, -11.9906438279, -9.20888634225],
[1.1004663882, -3.08431118755, -0.916066900677, 1.93532299367,
1.17692516686, 5.9348921803, -4.05365599273, 8.26598698511,
4.51585062467, -6.27746686689, 7.62802712835, -0.223077950462,
-0.427949101218, 8.71860936663, -5.76115297289, 7.96608529988,
12.7373826484, 11.5203435295, -13.6175312928, -1.06728196091,
-7.51063699016, -1.04762320936, 2.42044389794, 5.54054680324],
[1.28306707251, -2.50368459016, 2.29666365082, -1.89269791374,
-3.05237341326, -3.36337492752, 3.90970313414, -4.38196601125,
-8.96988582924, -7.54300669528, 5.35993312003, 5.46910712573,
3.49802187194, 5.08526919504, 2.26776831421, 9.60725218686,
-13.6450529994, -12.7084966146, 10.1742436937, 11.7605773458,
2.75443033698, 7.27315240877, -6.50913198075, -1.17948006621],
[1.33119677362, 2.58121334461, 1.43502873917, -1.54209925451,
4.81115398417, -4.11917366652, 4.49468079795, -1.06697798891,
-1.69456947267, -2.03271907546, -7.06214833178, -2.86690895381,
6.560751973, -2.06226456216, -2.66809704575, -2.09607806787,
10.0412959748, 10.9599164961, 0.36074265656, 3.35533700895,
-11.6113501099, 12.6750641708, 2.95607280869, 9.20603322989],
[1.41421356237, -0.585786437627, -1.0, -3.82842712475, -2.0,
3.82842712475, 1.17157287525, 1.82842712475, 0.585786437627,
6.82842712475, -6.58578643763, -0.828427124746, -3.82842712475,
-5.07106781187, -7.65685424949, -11.4142135624, -5.0, 5.07106781187
-0.171572875254, -3.75735931288, 2.65685424949, -5.82842712475,
11.0710678119, 7.41421356237],
[1.96404766408, -0.655927501903, 4.38447834155, 0.550492107123,
1.61586515855, -0.758447882785, -7.39647315034, 4.65523283475,
4.72765349953, -2.07911138285, -8.4438657128, -5.70772376386,
6.28815421933, 0.331082082995, -1.07131080137, -2.32629706064,
-6.35272756043, -3.54199395359, -7.8794061205, 12.5688517375,
-1.98900457357, 10.0244746139, -7.25902583048, 9.20303519087],
[2.01219875163, 1.46167794756, -1.98477252807, 3.65228285226,
1.83568252865, 3.62766028749, -1.06801639125, -2.32399054939,
-0.937033806048, -3.57807068207, -3.11664693334, -0.172364562473,
2.09370207991, 3.97112777169, -6.38576357878, 4.19366069145,
-12.1719369855, 4.30092841806, 12.6924280796, 8.08373809164,
16.5354155592, -3.79304546648, 2.15899409967, 3.82072848861],
[2.21431974338, -2.21431974338, -3.90321192591, -1.0,
```

```
0.903211925911, -3.0, -1.52542756084, -5.28099629098, 4.64295923013
-5.05085512169, -1.45875101324, 9.47949460844, 8.9540670476,
-6.96988847351, 4.75556873014, -2.96988847351, -3.75556873014,
2.40789589155, -14.1984983175, -4.64295923013, -9.23506333858,
-13.0874201428, 0.734825134934, -9.82716744702],
[2.27802573453, 1.28538083652, 1.3495702151, -0.260450965384,
-4.38552066553, -3.92478260065, 3.02119872197, -3.78799924911,
8.05293064075, 6.58402600579, 9.63200598368, 7.17714173857,
-4.58554122679, 12.0240550953, 1.78346135307, 0.252374082053,
4.67895603949, 3.23074951817, -9.30105002109, -8.90538822057,
3.66847264244, -9.44787358005, 2.39527432312, -8.93030173264],
[2.40575084198, 3.25552915896, -3.7458320007, -3.2672901519,
-2.55490600149, -1.17291262291, -1.08217743888, 8.23446651588,
-3.14885722831, 3.39887116732, 1.18030379195, -4.92704914591,
-7.11775231833, -3.35290005292, 9.59567053103, 3.21267959785,
-9.04282028331, 12.8226144607, -1.64518347721, -2.81567546644,
4.52802809076, 13.2451186563, -2.03339801107, 16.8335714322],
[2.64734306658, -1.68384110024, 0.049693845708, 3.75820799783,
-5.06870746286, -0.380076581696, 2.71895234787, 8.66282505979,
-6.75029930986, -4.97512385351, -7.99736208564, 4.48216779583,
-8.02771252726, -1.67285096662, 2.30038827689, -5.3919424197,
2.20514902551, -0.469530683087, 5.60354089378, -4.95848366699,
-6.42106254071, 10.1595863502, 10.3479153946, -15.9233195645],
[2.67315804566, 0.0398156192124, 0.396556253047, -3.54365801634,
4.01421340286, 2.61443815971, -1.40049891138, -2.51800527757,
-6.6009656268, -5.92977820517, 10.2726601131, -10.9349216697,
8.89579560561, -4.26546535302, 3.00642694718, 9.56576849734,
13.3423264823, -1.78628839978, 0.539423716883, -1.75912348678,
0.125769027693, 5.51488396211, 8.37149244244, -7.59617343345],
[3.0, 4.0, 6.0, 8.0, 12.0, 14.0, 18.0, 20.0, 24.0, 30.0, 32.0, 38.0
42.0, 44.0, 48.0, 54.0, 60.0, 62.0, 68.0, 72.0, 74.0, 80.0, 84.0,
90.0]
]
```

Newforms Of Given Weight And Level

1. Given a space of modular forms, can ask for the cuspidal newforms in it you ***must*** give name of generator for coefficient field

Example: We compute a newforms.

```
ModularForms(1,12).newforms('a')
```

```
    [q - 24*q^2 + 252*q^3 - 1472*q^4 + 4830*q^5 + O(q^6)]
```

```
X = ModularForms(Gamma1(13)).newforms('a'); X
```

```
    [q + a0*q^2 + (-2*a0 - 4)*q^3 + (-a0 - 1)*q^4 + (2*a0 + 3)*q^5 +
    O(q^6)]
```

```
f = X[0]; type(f)
```

```
    <class 'sage.modular.modform.element.Newform'>
```

```
parent(f)
```

```
    Modular Forms space of dimension 13 for Congruence Subgroup
    Gamma1(13) of weight 2 over Number Field in a0 with defining
    polynomial x^2 + 3*x + 3
```

```
show(f.q_expansion(15))
```

$$q + a_0 q^2 + (-2a_0 - 4)\, q^3 + (-a_0 - 1)\, q^4 + (2a_0 + 3)\, q^5 + (2a_0 + 6)\, q^6 + (-2a_0 - 3)\, q^8 + (a_0 +$$

```
f.hecke_eigenvalue_field()
```

```
    Number Field in a0 with defining polynomial x^2 + 3*x + 3
```

```
f.coefficients(10)
```

```
    [1, alpha, -2*alpha - 4, -alpha - 1, 2*alpha + 3, 2*alpha + 6, 0,
    -2*alpha - 3, alpha + 1, -3*alpha - 6]
```

```
A = f.abelian_variety(); A
```

```
    Newform abelian subvariety 13aG1 of dimension 2 of J1(13)
```

```
A.cuspidal_subgroup()
```

```
    Finite subgroup with invariants [19, 19] over QQ of Newform abelian
    subvariety 13aG1 of dimension 2 of J1(13)
```

```

```

```

```

Modular Symbols

1. Mostly work of me, Alex Ghitz, Craig Citro, John Cremona, and David Loeffler
2. The core of many modular forms routines
3. Fairly general implementation, e.g., all congruence subgroups and base rings
4. Represented by Manin symbols unless otherwise requested
5. Cremona's modular symbols library that he used to make his huge tables is also included, and there is a rudimentary Cython wrapper
6. Most important algorithm to learn well if you want to understand how Sage computes modular forms, and get more flexibility

Example: Illustrate the various inputs to computing modular symbols spaces, e.g., specifying the sign, over a finite field, etc.

```
ModularSymbols(group=Gamma0(11), weight=2, sign=0, base_ring=QQ,
use_cache=True)
```

```
Modular Symbols space of dimension 3 for Gamma_0(11) of weight 2
with sign 0 over Rational Field
```

```
ModularSymbols(group=Gamma1(13), weight=3, sign=-1, base_ring=GF(7),
use_cache=False)
```

```
Modular Symbols space of dimension 14 for Gamma_1(13) of weight 3
with sign -1 and over Finite Field of size 7
```

```
ModularSymbols(kronecker_character_upside_down(7), weight=3,
sign=-1)
```

```
Modular Symbols space of dimension 2 and level 7, weight 3,
character [-1], sign -1, over Rational Field
```

```

```

Example: We illustrate that computing matrices of Hecke operators on modular symbols can be **vastly faster** than on modular forms spaces. This gives nearly the same linear operator, but ***on a different basis***, hence different information.

```
time t2 = ModularSymbols(Gamma1(13), sign=1, weight=3,
use_cache=False).hecke_matrix(2)
```

```
Time: CPU 0.11 s, Wall: 0.12 s
```

```
show(factor(t2.charpoly()))
```

$$(x^2 - 2x + 17) \cdot (x^4 - 4x^3 - 10x^2 + 28x + 241) \cdot (x^4 + 2x^3 + 5x^2 + 4x + 1) \cdot (x^4 + 4x^3 + 8x^2 -$$

```
time t2 = ModularForms(Gamma1(13), weight=3,
use_cache=False).hecke_matrix(2)
```

```
Time: CPU 1.55 s, Wall: 1.75 s
```

```
show(factor(t2.charpoly()))
```

$$(x^2 - 8x + 17) \cdot (x^2 - 2x + 17) \cdot (x^4 - 16x^3 + 95x^2 - 248x + 241) \cdot (x^4 - 4x^3 - 10x^2 + 28x +$$

Example: Illustrate using Cremona's modular symbols library (which is way more optimized than Sage or Magma for weight 2 on $\Gamma_0(N)$).

```
time M = CremonaModularSymbols(2009)
```

```
Time: CPU 0.03 s, Wall: 0.03 s
```

```
time t = M.hecke_matrix(2)
```

```
    Time: CPU 0.02 s, Wall: 0.02 s
```

```
time M = ModularSymbols(2009, use_cache=False, sign=1,
base_ring=GF(40009))
```

```
    Time: CPU 1.00 s, Wall: 1.10 s
```

```
time t = M.hecke_matrix(2)
```

```
    Time: CPU 0.05 s, Wall: 0.06 s
```

```

```

Systems Of Eigenvalues

1. Clever compact representation of eigenvalues a_2, a_3, a_5, etc. that Cremona came up with (explain with example)
2. Highly optimized except in case of higher weight and nontrivial character (good project!)

Example: We give an example to illustrate the compact representation of eigenvalues.

```
M = ModularSymbols(389,sign=1); D = M.decomposition(); D
```

```
    [
    Modular Symbols subspace of dimension 1 of Modular Symbols space of
    dimension 33 for Gamma_0(389) of weight 2 with sign 1 over Rational
    Field,
    Modular Symbols subspace of dimension 1 of Modular Symbols space of
    dimension 33 for Gamma_0(389) of weight 2 with sign 1 over Rational
    Field,
    Modular Symbols subspace of dimension 2 of Modular Symbols space of
    dimension 33 for Gamma_0(389) of weight 2 with sign 1 over Rational
    Field,
    Modular Symbols subspace of dimension 3 of Modular Symbols space of
    dimension 33 for Gamma_0(389) of weight 2 with sign 1 over Rational
    Field,
    Modular Symbols subspace of dimension 6 of Modular Symbols space of
    dimension 33 for Gamma_0(389) of weight 2 with sign 1 over Rational
    Field,
    Modular Symbols subspace of dimension 20 of Modular Symbols space c
    dimension 33 for Gamma_0(389) of weight 2 with sign 1 over Rational
    Field
    ]
```

```
A = D[-1]; aplist = [A.eigenvalue(p) for p in prime_range(1000)]
aplist[:3]
```

```
    [alpha, -20146763/1097385680*alpha^19 + 20466323/219477136*alpha^18
    + 119884773/274346420*alpha^17 - 753611053/274346420*alpha^16 -
    381358355/109738568*alpha^15 + 3611475535/109738568*alpha^14 +
```

```
6349339639/1097385680*alpha^13 - 56878934241/274346420*alpha^12 +
71555185319/1097385680*alpha^11 + 163330998525/219477136*alpha^10 -
223188336749/548692840*alpha^9 - 169878973265/109738568*alpha^8 +
265944624817/274346420*alpha^7 + 199655892261/109738568*alpha^6 -
1167579836501/1097385680*alpha^5 - 619178000979/548692840*alpha^4 +
261766056911/548692840*alpha^3 + 4410485304/13717321*alpha^2 -
14646077211/274346420*alpha - 1604641167/68586605,
252247073/1097385680*alpha^19 - 89195955/109738568*alpha^18 -
6876716517/1097385680*alpha^17 + 13248231811/548692840*alpha^16 +
3639338697/54869284*alpha^15 - 32041245347/109738568*alpha^14 -
367946611589/1097385680*alpha^13 + 2037515640679/1097385680*alpha^1
+ 816602908511/1097385680*alpha^11 - 368120881159/54869284*alpha^10
- 19595857657/1097385680*alpha^9 + 763403091515/54869284*alpha^8 -
403904463001/137173210*alpha^7 - 1743458092745/109738568*alpha^6 +
5304122556631/1097385680*alpha^5 + 9907751136883/1097385680*alpha^4
- 704659646193/274346420*alpha^3 - 236814032039/109738568*alpha^2 +
88326575941/274346420*alpha + 37821733103/274346420]
```

```
v=A.compact_system_of_eigenvalues(prime_range(3),'a')
```

```
v[0][:4]
```

```
[ 253/149  -12/149    8/149   34/149  -40/149  112/149 -122/149
-80/149  -54/149   64/149 -258/149    4/149    4/149   44/149
-12/149   -4/149   14/149  -18/149   -8/149   -8/149]
```

```
v[1]
```

```
(1, 325011341/5486928400*a^19 - 27850367/137173210*a^18 -
2215759151/1371732100*a^17 + 8226699541/1371732100*a^16 +
9406039723/548692840*a^15 - 39497246359/548692840*a^14 -
480352641313/5486928400*a^13 + 2486905749713/5486928400*a^12 +
557000342801/2743464200*a^11 - 1773928265763/1097385680*a^10 -
337553126739/5486928400*a^9 + 723309957135/219477136*a^8 -
3337045319611/5486928400*a^7 - 4038350495147/1097385680*a^6 +
1440451869583/1371732100*a^5 + 1391219604327/685866050*a^4 -
1538401573037/2743464200*a^3 - 63593424607/137173210*a^2 +
94683919497/1371732100*a + 20111495423/685866050,
245292751/10973856800*a^19 - 108462779/2194771360*a^18 -
1868624441/2743464200*a^17 + 1053348869/685866050*a^16 +
9273188337/1097385680*a^15 - 21495662357/1097385680*a^14 -
600802729783/10973856800*a^13 + 363429296067/2743464200*a^12 +
2157340531277/10973856800*a^11 - 1124044461681/2194771360*a^10 -
2067273806887/5486928400*a^9 + 249161430265/219477136*a^8 +
431666959413/1371732100*a^7 - 1496626555737/1097385680*a^6 -
83692048403/10973856800*a^5 + 4263553489053/5486928400*a^4 -
428651621607/5486928400*a^3 - 23855918389/137173210*a^2 +
28886053387/2743464200*a + 3748686767/342933025,
-459613381/5486928400*a^19 + 173657167/548692840*a^18 +
742092874/342933025*a^17 - 50360117049/5486928400*a^16 -
22971954579/1097385680*a^15 + 118069290279/1097385680*a^14 +
30131613203/342933025*a^13 - 1803671625179/2743464200*a^12 -
500753450637/5486928400*a^11 + 2481390337951/1097385680*a^10 -
653283286489/1371732100*a^9 - 486564939659/109738568*a^8 +
```

```
9246040599551/5486928400*a^7 + 659706892163/137173210*a^6 -
2785987549583/1371732100*a^5 - 14719895868311/5486928400*a^4 +
2568856350817/2743464200*a^3 + 45312991334/68586605*a^2 -
74461873461/685866050*a - 14727623454/342933025,
-675356389/2743464200*a^19 + 114667407/137173210*a^18 +
9312668813/1371732100*a^17 - 136843529487/5486928400*a^16 -
80264238801/1097385680*a^15 + 66552843859/219477136*a^14 +
2092598810219/5486928400*a^13 - 10652485544029/5486928400*a^12 -
5011286803921/5486928400*a^11 + 3881339310391/548692840*a^10 +
1810588193857/5486928400*a^9 - 3250922894157/219477136*a^8 +
7563926394769/2743464200*a^7 + 18754791447223/1097385680*a^6 -
6837042490719/1371732100*a^5 - 53805908702083/5486928400*a^4 +
7477254150821/2743464200*a^3 + 162703319571/68586605*a^2 -
232806734423/685866050*a - 105609911249/685866050,
20146763/2194771360*a^19 - 20466323/438954272*a^18 -
119884773/548692840*a^17 + 753611053/548692840*a^16 +
381358355/219477136*a^15 - 3611475535/219477136*a^14 -
6349339639/2194771360*a^13 + 56878934241/548692840*a^12 -
71555185319/2194771360*a^11 - 163330998525/438954272*a^10 +
223188336749/1097385680*a^9 + 169878973265/219477136*a^8 -
265944624817/548692840*a^7 - 199655892261/219477136*a^6 +
1167579836501/2194771360*a^5 + 619178000979/1097385680*a^4 -
261766056911/1097385680*a^3 - 4396767983/27434642*a^2 +
14097384371/548692840*a + 836613886/68586605,
-3075090289/10973856800*a^19 + 2002355747/2194771360*a^18 +
10848627717/1371732100*a^17 - 151191377053/5486928400*a^16 -
6044369631/68586605*a^15 + 93309291891/274346420*a^14 +
5326851715167/10973856800*a^13 - 3041851886319/1371732100*a^12 -
14324843442163/10973856800*a^11 + 3621733878631/438954272*a^10 +
5886066098613/5486928400*a^9 - 3875875282149/219477136*a^8 +
12625298184297/5486928400*a^7 + 4556376023959/219477136*a^6 -
59036716330773/10973856800*a^5 - 32895692042111/2743464200*a^4 +
17340557389873/5486928400*a^3 + 1573186478983/548692840*a^2 -
1114387437183/2743464200*a - 248437167707/1371732100,
-5751237/1097385680*a^19 + 33173801/2194771360*a^18 +
288853159/2194771360*a^17 - 213914589/548692840*a^16 -
689853699/548692840*a^15 + 4174705151/1097385680*a^14 +
3153897897/548692840*a^13 - 37088035947/2194771360*a^12 -
1540991689/109738568*a^11 + 63326133021/2194771360*a^10 +
61152551939/2194771360*a^9 + 1858420443/109738568*a^8 -
38920811329/548692840*a^7 - 121171768243/1097385680*a^6 +
131801161837/1097385680*a^5 + 233857051319/2194771360*a^4 -
81655506897/1097385680*a^3 - 37057045927/1097385680*a^2 +
2865092227/274346420*a + 1308232839/548692840,
608077081/2194771360*a^19 - 1018743517/1097385680*a^18 -
16846440981/2194771360*a^17 + 3803856889/137173210*a^16 +
91408196917/1097385680*a^15 - 92635054801/274346420*a^14 -
966022777027/2194771360*a^13 + 4753010333053/2194771360*a^12 +
478453927461/438954272*a^11 - 4337956279221/548692840*a^10 -
1235079423911/2194771360*a^9 + 3640094529521/219477136*a^8 -
1529293836059/548692840*a^7 - 10510275246449/548692840*a^6 +
11743445691099/2194771360*a^5 + 24078791531269/2194771360*a^4 -
```

```
204140462182/68586605*a^3 - 2889869194407/1097385680*a^2 +
204939716069/548692840*a + 92256771939/548692840,
1660779427/10973856800*a^19 - 249993983/548692840*a^18 -
48932024013/10973856800*a^17 + 77699674429/5486928400*a^16 +
28862381683/548692840*a^15 - 198406931983/1097385680*a^14 -
3446131715411/10973856800*a^13 + 13448723307911/10973856800*a^12 +
10602082792869/10973856800*a^11 - 5221188858793/1097385680*a^10 -
13442103132533/10973856800*a^9 + 582992895149/54869284*a^8 -
1608514236173/2743464200*a^7 - 14208120958987/1097385680*a^6 +
31601271852929/10973856800*a^5 + 83536430761427/10973856800*a^4 -
2616588046791/1371732100*a^3 - 1989804782611/1097385680*a^2 +
699419510259/2743464200*a + 306825487287/2743464200,
1634020679/10973856800*a^19 - 525682469/1097385680*a^18 -
46593899911/10973856800*a^17 + 79951867483/5486928400*a^16 +
6581133153/137173210*a^15 - 39816619875/219477136*a^14 -
2959207100667/10973856800*a^13 + 13113798690197/10973856800*a^12 +
8242330970053/10973856800*a^11 - 308523356173/68586605*a^10 -
7721373333551/10973856800*a^9 + 1069454541789/109738568*a^8 -
3056974035721/2743464200*a^7 - 12710158719877/1097385680*a^6 +
31938045885093/10973856800*a^5 + 73897036716869/10973856800*a^4 -
2406303819407/1371732100*a^3 - 1770570851039/1097385680*a^2 +
620212682603/2743464200*a + 281751699339/2743464200,
950271899/5486928400*a^19 - 627276581/1097385680*a^18 -
26693393961/5486928400*a^17 + 94495734821/5486928400*a^16 +
59093280791/1097385680*a^15 - 232578859561/1097385680*a^14 -
402679325023/1371732100*a^13 + 7554628417207/5486928400*a^12 +
4245912017673/5486928400*a^11 - 2799010557757/548692840*a^10 -
3177161818601/5486928400*a^9 + 1192594007611/109738568*a^8 -
8315021611579/5486928400*a^7 - 6977120039443/548692840*a^6 +
9173161186689/2743464200*a^5 + 20070555082197/2743464200*a^4 -
1325554239017/685866050*a^3 - 956526414493/548692840*a^2 +
167763340669/685866050*a + 149781303739/1371732100,
-446451063/5486928400*a^19 + 139051847/548692840*a^18 +
12559295087/5486928400*a^17 - 20757871001/2743464200*a^16 -
1747326684/68586605*a^15 + 50500907603/548692840*a^14 +
774767962689/5486928400*a^13 - 3232885357859/5486928400*a^12 -
2157039100221/5486928400*a^11 + 117620413757/54869284*a^10 +
2271444465417/5486928400*a^9 - 490394720149/109738568*a^8 +
955355180499/2743464200*a^7 + 559613331961/109738568*a^6 -
6100548821941/5486928400*a^5 - 15642674414373/5486928400*a^4 +
1872755680641/2743464200*a^3 + 359245310447/548692840*a^2 -
61777656593/685866050*a - 55360600863/1371732100,
1196337671/10973856800*a^19 - 793373679/2194771360*a^18 -
16777470497/5486928400*a^17 + 59641783617/5486928400*a^16 +
4635665869/137173210*a^15 - 73220365231/548692840*a^14 -
2019864117793/10973856800*a^13 + 4742664208739/5486928400*a^12 +
5337518267417/10973856800*a^11 - 7003649381171/2194771360*a^10 -
258882515619/685866050*a^9 + 1485700731113/219477136*a^8 -
4918345411283/5486928400*a^7 - 8646020887677/1097385680*a^6 +
22123803913087/10973856800*a^5 + 24682959870263/5486928400*a^4 -
6416007235397/5486928400*a^3 - 144964601919/137173210*a^2 +
407948718977/2743464200*a + 44447814189/685866050,
```

```
180312963/2743464200*a^19 - 385063707/2194771360*a^18 -
22327470513/10973856800*a^17 + 15507168677/2743464200*a^16 +
13996082323/548692840*a^15 - 82381899993/1097385680*a^14 -
903931173593/5486928400*a^13 + 5822927432911/10973856800*a^12 +
781000556943/1371732100*a^11 - 4715252337821/2194771360*a^10 -
10313813141483/10973856800*a^9 + 1094337202257/219477136*a^8 +
1547755388879/5486928400*a^7 - 3436305110841/548692840*a^6 +
5366904284127/5486928400*a^5 + 41020789066527/10973856800*a^4 -
4370549589139/5486928400*a^3 - 978426443661/1097385680*a^2 +
153076703267/1371732100*a + 153516413637/2743464200,
-169788063/1371732100*a^19 + 25357062/68586605*a^18 +
9929356669/2743464200*a^17 - 3901115163/342933025*a^16 -
23244150721/548692840*a^15 + 78750171173/548692840*a^14 +
689399558343/2743464200*a^13 - 2632433781743/2743464200*a^12 -
265352271684/342933025*a^11 + 503142953667/137173210*a^10 +
701382177851/685866050*a^9 - 110501267044/13717321*a^8 +
179666356799/685866050*a^7 + 5292374155097/548692840*a^6 -
1335287411063/685866050*a^5 - 15264046284051/2743464200*a^4 +
3628004340189/2743464200*a^3 + 88531856917/68586605*a^2 -
249096461859/1371732100*a - 52801221131/685866050,
-4850519049/10973856800*a^19 + 159675843/109738568*a^18 +
135626477421/10973856800*a^17 - 14973304428/342933025*a^16 -
29837282485/219477136*a^15 + 293348055501/548692840*a^14 +
8059253688287/10973856800*a^13 - 37880306002307/10973856800*a^12 -
20912059789963/10973856800*a^11 + 870816056264/68586605*a^10 +
14632993624361/10973856800*a^9 - 2944425104151/109738568*a^8 +
21564403645427/5486928400*a^7 + 34171494507521/1097385680*a^6 -
91842109884313/10973856800*a^5 - 195196356710509/10973856800*a^4 +
1644586545473/342933025*a^3 + 924551941343/219477136*a^2 -
1679228336673/2743464200*a - 722296821179/2743464200,
-96233401/2743464200*a^19 + 267759209/2194771360*a^18 +
10296303451/10973856800*a^17 - 19571519383/5486928400*a^16 -
10592994267/1097385680*a^15 + 2897743051/68586605*a^14 +
127547117743/2743464200*a^13 - 2870003600847/10973856800*a^12 -
502349496319/5486928400*a^11 + 2004239659767/2194771360*a^10 -
482469804809/10973856800*a^9 - 199340196205/109738568*a^8 +
2597629265167/5486928400*a^7 + 2171530756399/1097385680*a^6 -
3808058067179/5486928400*a^5 - 11760429547379/10973856800*a^4 +
1954727511453/5486928400*a^3 + 263814150077/1097385680*a^2 -
16076302946/342933025*a - 35864756149/2743464200,
192125593/2743464200*a^19 - 460801607/2194771360*a^18 -
22826589743/10973856800*a^17 + 9024955561/1371732100*a^16 +
13591074903/548692840*a^15 - 93047861363/1097385680*a^14 -
820030854373/5486928400*a^13 + 6373104097971/10973856800*a^12 +
2551918844517/5486928400*a^11 - 5003604892271/2194771360*a^10 -
6544802036163/10973856800*a^9 + 141227441097/27434642*a^8 -
198389792057/685866050*a^7 - 6954271688587/1097385680*a^6 +
3921685976661/2743464200*a^5 + 41251470563297/10973856800*a^4 -
5193334500029/5486928400*a^3 - 995160593131/1097385680*a^2 +
170809821637/1371732100*a + 157708844807/2743464200,
-3892193777/10973856800*a^19 + 2604321159/2194771360*a^18 +
53853219109/5486928400*a^17 - 24272705863/685866050*a^16 -
```

```
116706676267/1097385680*a^15 + 94386148621/219477136*a^14 +
6154712497421/10973856800*a^13 - 15093527251743/5486928400*a^12 -
15200334843539/10973856800*a^11 + 21962849145773/2194771360*a^10 +
1949024658147/2743464200*a^9 - 4585657995853/219477136*a^8 +
9682708598073/2743464200*a^7 + 26308292343311/1097385680*a^6 -
74049585562559/10973856800*a^5 - 9326021030467/685866050*a^4 +
20537106367139/5486928400*a^3 + 1763821292131/548692840*a^2 -
1289822239939/2743464200*a - 275509624141/1371732100)
```

Example: We give a benchmarketing example. (This is on a 2.6 core 2 duo linux box.)

```
sage: A = ModularSymbols(389,sign=1).decomposition()[4]; A
Modular Symbols subspace of dimension 6 of Modular Symbols space of dimension 33 for Gam
sage: time v = A.compact_system_of_eigenvalues(prime_range(10000), 'a')
CPU times: user 6.03 s, sys: 0.16 s, total: 6.19 s
Wall time: 6.33 s

sage: magma.eval('A := NewformDecomposition(CuspidalSubspace(ModularSymbols(389,2,1)))[4
'Modular symbols space for Gamma_0(389) of weight 2 and dimension 6 over Rational Field'
sage: B = magma('A')
sage: time vv = B.CompactSystemOfEigenvalues(10000)
CPU times: user 0.00 s, sys: 0.00 s, total: 0.00 s
Wall time: 37.98 s

37.98 / 6.33 = 6
```

Modular Symbols: Factorization

1. A basic operation on modular symbols spaces is to write as direct sums of Hecke submodules
2. This involves lots of exact linear algebra, and is perhaps the main bottlekneck in many modular symbols computations

Example: We decompose a space of modular symbols.

```
M = ModularSymbols(Gamma1(13), sign=1); M
```

```
Modular Symbols space of dimension 13 for Gamma_1(13) of weight 2
with sign 1 and over Rational Field
```

```
M.decomposition()
```

```
[
Modular Symbols subspace of dimension 1 of Modular Symbols space of
dimension 13 for Gamma_1(13) of weight 2 with sign 1 and over
Rational Field,
Modular Symbols subspace of dimension 1 of Modular Symbols space of
```

```
dimension 13 for Gamma_1(13) of weight 2 with sign 1 and over
Rational Field,
Modular Symbols subspace of dimension 1 of Modular Symbols space of
dimension 13 for Gamma_1(13) of weight 2 with sign 1 and over
Rational Field,
Modular Symbols subspace of dimension 2 of Modular Symbols space of
dimension 13 for Gamma_1(13) of weight 2 with sign 1 and over
Rational Field,
Modular Symbols subspace of dimension 2 of Modular Symbols space of
dimension 13 for Gamma_1(13) of weight 2 with sign 1 and over
Rational Field,
Modular Symbols subspace of dimension 2 of Modular Symbols space of
dimension 13 for Gamma_1(13) of weight 2 with sign 1 and over
Rational Field,
Modular Symbols subspace of dimension 2 of Modular Symbols space of
dimension 13 for Gamma_1(13) of weight 2 with sign 1 and over
Rational Field,
Modular Symbols subspace of dimension 2 of Modular Symbols space of
dimension 13 for Gamma_1(13) of weight 2 with sign 1 and over
Rational Field
]
```

```
M = ModularSymbols(37); M
```

```
Modular Symbols space of dimension 5 for Gamma_0(37) of weight 2
with sign 0 over Rational Field
```

```
M.decomposition()
```

```
[
Modular Symbols subspace of dimension 1 of Modular Symbols space of
dimension 5 for Gamma_0(37) of weight 2 with sign 0 over Rational
Field,
Modular Symbols subspace of dimension 2 of Modular Symbols space of
dimension 5 for Gamma_0(37) of weight 2 with sign 0 over Rational
Field,
Modular Symbols subspace of dimension 2 of Modular Symbols space of
dimension 5 for Gamma_0(37) of weight 2 with sign 0 over Rational
Field
]
```

Supersingular Module

1. This is the free $\mathbf{Z}$-module generated by supersingular j-invariants in characteristic p.
2. Sage can compute Hecke operators and decompose with respect to their action.
3. One quickly gets very sparse matrices

Example: We compute a supersingular module, then display the graph of a Hecke operator.

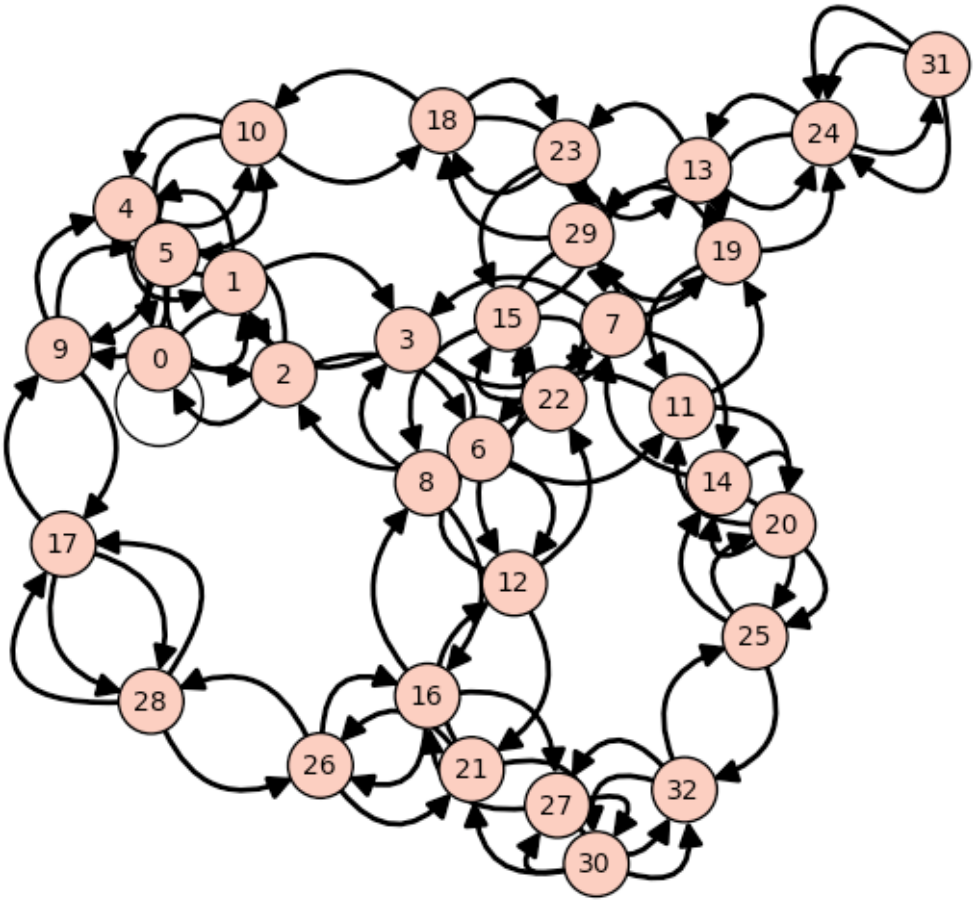
```
X = SupersingularModule(389); X
```

```
Module of supersingular points on X_0(1)/F_389 over Integer Ring
```

```
T2 = X.hecke_matrix(2); show(T2)
```

$$\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

```
DiGraph(T2).plot()
```



Brandt Modules

1. Mostly me, Jon Bober, Gonzalo Tornaria, Alia Hamieh
2. Compute right ideal classes in rational quaternion algebras, along with Hecke action
3. Very new functionality in Sage -- took 2-3 weeks of hard work
4. This was the capability of Magma (by David Kohel) that got me to start using Magma in 1998, and implementing this in Sage was the moment *I finally no longer felt "enslaved"*.

```
Q.<i,j,k> = QuaternionAlgebra(-3,-7)
(2+i-j+k)^3
```

```
-178 - 19*i + 19*j - 19*k
```

```
B = BrandtModule(7, 20); B
```

```
Brandt module of dimension 18 of level 7*20 of weight 2 over
Rational Field
```

```
B.right_ideals()
```

```
(Fractional ideal (2 + 2*j + 32*k, 2*i + 54*k, 4*j + 64*k, 80*k),
Fractional ideal (2 + 2*j + 32*k, 2*i + 8*j + 22*k, 12*j + 192*k,
240*k), Fractional ideal (2 + 2*j + 32*k, 2*i + 32*j + 646*k, 36*j
576*k, 720*k), Fractional ideal (2 + 2*j + 112*k, 2*i + 4*j + 118*k
12*j + 192*k, 240*k), Fractional ideal (2 + 2*j + 112*k, 2*i + 4*j
358*k, 36*j + 576*k, 720*k), Fractional ideal (2 + 10*j + 80*k, 2*i
+ 8*j + 182*k, 12*j + 192*k, 240*k), Fractional ideal (2 + 10*j +
80*k, 2*i + 8*j + 422*k, 36*j + 576*k, 720*k), Fractional ideal (2
10*j + 160*k, 2*i + 4*j + 38*k, 12*j + 192*k, 240*k), Fractional
ideal (2 + 10*j + 640*k, 2*i + 28*j + 422*k, 36*j + 576*k, 720*k),
Fractional ideal (2 + 14*j + 304*k, 2*i + 16*j + 310*k, 36*j +
576*k, 720*k), Fractional ideal (2 + 14*j + 704*k, 2*i + 20*j +
454*k, 36*j + 576*k, 720*k), Fractional ideal (2 + 22*j + 272*k, 2*
+ 20*j + 374*k, 36*j + 576*k, 720*k), Fractional ideal (2 + 22*j +
592*k, 2*i + 16*j + 230*k, 36*j + 576*k, 720*k), Fractional ideal (
+ 26*j + 496*k, 2*i + 28*j + 262*k, 36*j + 576*k, 720*k), Fractiona
ideal (2 + 26*j + 656*k, 2*i + 8*j + 262*k, 36*j + 576*k, 720*k),
Fractional ideal (2 + 34*j + 464*k, 2*i + 32*j + 326*k, 36*j +
576*k, 720*k), Fractional ideal (2 + 34*j + 544*k, 2*i + 4*j + 38*k
36*j + 576*k, 720*k), Fractional ideal (2 + 98*j + 368*k, 2*i + 44*
+ 118*k, 108*j + 1728*k, 2160*k))
```

```
time t2 = B.hecke_matrix(2); show(t2)
```

```
Time: CPU 0.00 s, Wall: 0.00 s
```

$$\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

```
show(t2.charpoly().factor())
```

$$(x-4)\cdot(x-3)\cdot(x-2)\cdot(x+1)\cdot(x+2)\cdot(x-1)^2\cdot(x+3)^2\cdot x^3\cdot(x^2+x-4)\cdot(x^4+x^3-1$$

```
I = B.right_ideals()[3]; show(I.theta_series(20))
```

$$1+2q^3+4q^6+4q^8+4q^{10}+8q^{12}+2q^{13}+8q^{15}+8q^{16}+2q^{17}+4q^{18}+O(q^{20})$$

Example: A benchmarketing example (on core2 64-bit Linux).

```
SAGE
sage: time B = BrandtModule(59,15)
CPU times: user 0.00 s, sys: 0.00 s, total: 0.00 s
Wall time: 0.00 s
sage: time B.hecke_matrix(2)
CPU times: user 9.29 s, sys: 0.27 s, total: 9.56 s
Wall time: 9.57 s
116 x 116 dense matrix over Rational Field
sage: time B.hecke_matrix(3)
CPU times: user 0.03 s, sys: 0.00 s, total: 0.03 s
Wall time: 0.03 s
116 x 116 dense matrix over Rational Field

MAGMA
sage: magma.eval('time B := BrandtModule(59,15);')
'Time: 40.820'
sage: magma.eval('time T2 := HeckeOperator(B,2);')
'Time: 0.330'
sage: magma.eval('time T3 := HeckeOperator(B,3);')
'Time: 0.360'

40.820 / 9.57 = 4.2
```

Modular Abelian Varieties

1. Represent any modular abelian variety
2. Enumeration up to isogeny
3. Computation of exact endomorphism ring
4. Sums and intersections
5. Computation with torsion and cuspidal points
6. Modular degree

Example: We define $J_0(N)$, decompose to obtain some simple modular abelian varieties,

then take intersections.

```
J = J0(43); J
```

```
Abelian variety J0(43) of dimension 3
```

```
D = J.decomposition(); D
```

```
[
Simple abelian subvariety 43a(1,43) of dimension 1 of J0(43),
Simple abelian subvariety 43b(1,43) of dimension 2 of J0(43)
]
```

```
D[0].intersection(D[1])
```

```
(Finite subgroup with invariants [2, 2] over QQ of Simple abelian
subvariety 43a(1,43) of dimension 1 of J0(43), Simple abelian
subvariety of dimension 0 of J0(43))
```

```
G.lattice()
```

```
Traceback (click to the left for traceback)
...
AttributeError: 'DirichletGroup_class' object has no attribute
'lattice'
```

Example: We compute some torsion subgroups of modular abelian varieties

```
J = J0(69); J.decomposition()
```

```
[
Simple abelian subvariety 23a(1,69) of dimension 2 of J0(69),
Simple abelian subvariety 23a(3,69) of dimension 2 of J0(69),
Simple abelian subvariety 69a(1,69) of dimension 1 of J0(69),
Simple abelian subvariety 69b(1,69) of dimension 2 of J0(69)
]
```

```
A = J[3]; A
```

```
Simple abelian subvariety 69b(1,69) of dimension 2 of J0(69)
```

```
A.cuspidal_subgroup()
```

```
Finite subgroup with invariants [2, 2] over QQ of Simple abelian
subvariety 69b(1,69) of dimension 2 of J0(69)
```

```
T = A.rational_torsion_subgroup(); T
```

```
Torsion subgroup of Simple abelian subvariety 69b(1,69) of dimensic
2 of J0(69)
```

```
T.multiple_of_order()
```

```
4
```

```
T.divisor_of_order()
```

```
4
```

```

```

Example: We compute some endomorphism rings of modular abelian varieties

```
E = End(A); E
```

```
Endomorphism ring of Simple abelian subvariety 69b(1,69) of
dimension 2 of J0(69)
```

```
for phi in E.gens(): print phi.matrix()
```

```
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
[ 0  0  2  0]
[ 0  2 -1  1]
[ 2  0 -2  0]
[ 2 -4  2 -4]
```

Tamagawa Numbers And Component Groups

1. Can compute component group order at primes of multiplicative reduction, using Brandt modules, etc.
2. Compute Tamagawa numbers in many cases

Example: We compute the component groups of some modular abelian varieties.

```
A = J0(69)[3]; A
```

```
Simple abelian subvariety 69b(1,69) of dimension 2 of J0(69)
```

```
A.component_group_order(3)
```

```
22
```

```
A.component_group_order(23)
```

```
2
```

```
A.tamagawa_number(3)
```

```
2
```

Example: We compute a Tamagawa number.

```
A.tamagawa_number(23)
```

```
2
```

P-adic Modular Forms

```
M = OverconvergentModularForms(3, 16, 1/2); M
```

```
Space of 3-adic 1/2-overconvergent modular forms of weight-characte
16 over Rational Field
```

```
M.hecke_matrix(2,2)
```

```
[                          1                           0]
[                          0 2335232192130/25949996501]
```

```
show(M.hecke_matrix(2,4))
```

$$\begin{pmatrix} 1 & 0 & 0 & \frac{68100570190}{259499965} \\ 0 & \frac{2335232192130}{25949996501} & \frac{16815597512328}{25949996501} & \frac{47939678128577938377342}{6734023184019122430} \\ 0 & -\frac{1223393579448818065715 16}{673402318401912243001} & \frac{164613271032536908412271 06}{673402318401912243001} & \\ 0 & -\frac{818759176473092913108287736796628937}{1747478780629491061758501 1739501} & -\frac{611020156148263932244827938537631120}{1747478780629491061758501 1739501} & \end{pmatrix}$$

```
f = M.gen(0); f
```

```
3-adic overconvergent modular form of weight-character 16 with
q-expansion 1 - 8160/25949996501*q - 24308640/2359090591*q^2 -
8160/25949996501*q^3 - 8762000678880/25949996501*q^4 -
249023437508160/25949996501*q^5 - 24308640/2359090591*q^6 -
3521827447376640/2359090591*q^7 - 2007784882836000/181468507*q^8 -
8160/25949996501*q^9 - 741840820336808640/2359090591*q^10 -
340863450624317203 20/25949996501*q^11 -
8762000678880/25949996501*q^12 -
379706260904527804 80/2359090591*q^13 -
104915239657350105 60/214462781*q^14 -
249023437508160/25949996501*q^15 -
940812659083011687 2160/25949996501*q^16 -
163338266435804873280/181468507*q^17 - 24308640/2359090591*q^18 -
112616360512525776 48000/2359090591*q^19 + O(q^20)
```

Questions?