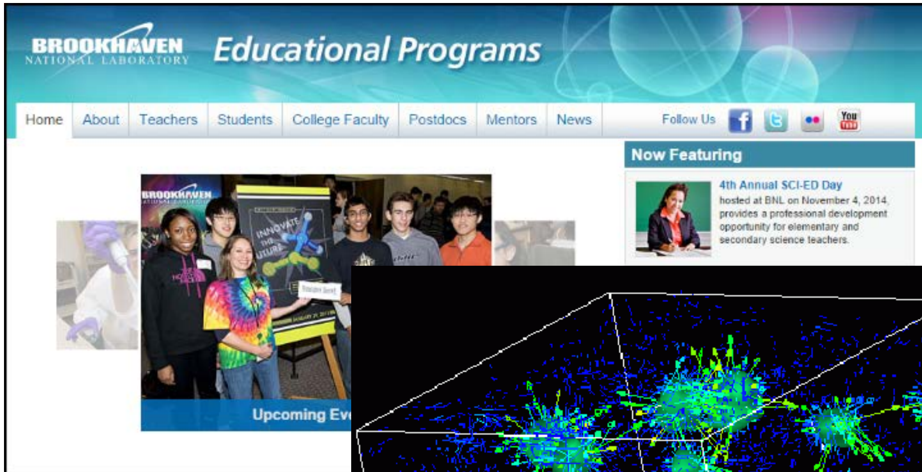
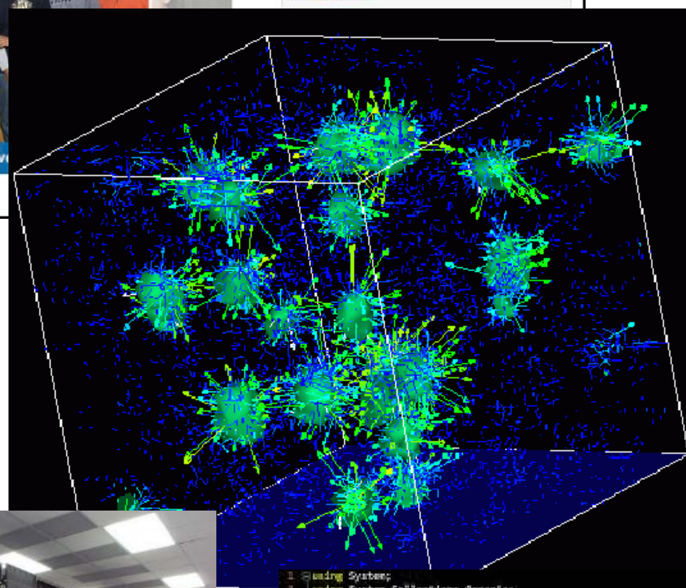
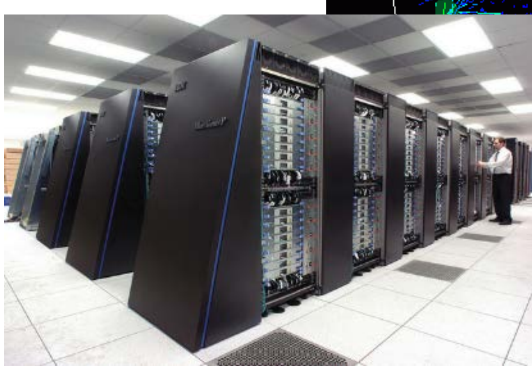






Survey of Scientific Computing
(SciComp 301)

Dave Biersach
Brookhaven National Laboratory
dbiersach@bnl.gov

Session 12
Continued Fractions,
Chi Squared

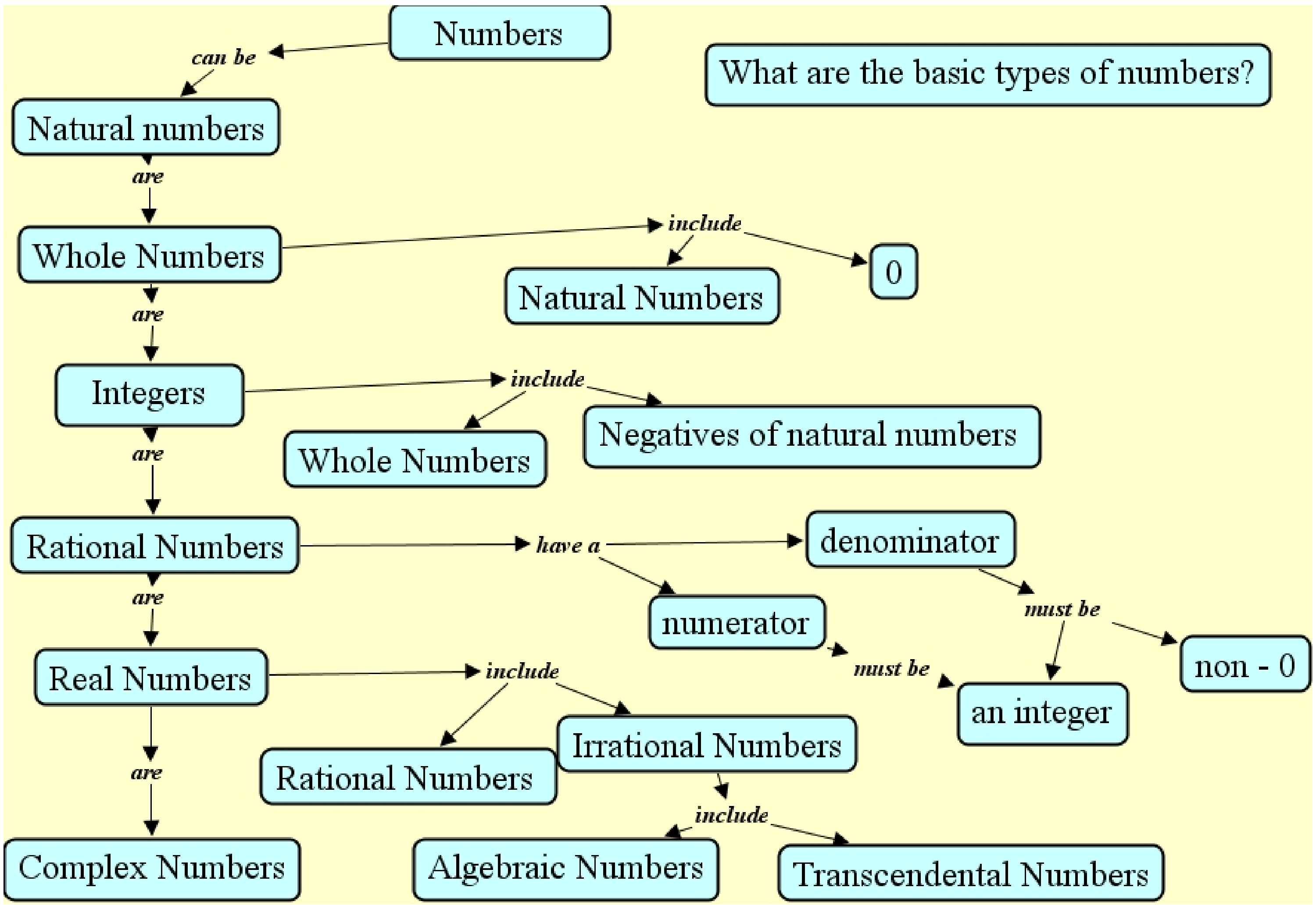
Section Goals

- Gain an appreciation for **Continued Fractions** in nature
- Understand the three types of CFs: 1) finite, 2) infinite with repeating *sequence*, 3) infinite with repeating *pattern*
- Write code to **generate** a generalized CF for a real number, and how to **expand** that CF to produce **convergents** of the original number
- Appreciate the hidden underlying simplicity of the **generalized continued fraction** for π
- Perform a computational mathematical experiment to determine the solutions to **Pell's Equation**

Seminar Goals

- Gain an appreciation for the **Normal Distribution**
- Investigate if a Normal Distribution can be made from a Uniform Distribution using a **Pachinko game**
- Use **chi-squared statistic** to determine if a random sample conforms to a reasonable Normal Distribution

Expanding Your Definition of a “Number”



Continued Fractions

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3}}}$$

$$\mathbf{3.245} = 3 + \cfrac{1}{4 + \cfrac{1}{12 + \cfrac{1}{4}}}$$

Continued Fractions

What is the continued fraction expansion for **0.825** (= **33/40**)?

x	1/x	INT(1/x)	1/x - INT(1/x)
0.82500000	1.21212121	**1**	0.21212121
0.21212121	4.71428571	**4**	0.71428571
0.71428571	1.40000000	**1**	0.40000000
0.40000000	2.50000000	**2**	0.50000000
0.50000000	2.00000000	**2**	0.00000000

A CF is an *encoding* scheme

0.825 = [0; 1, 4, 1, 2, 2]

All **rational** numbers have a CF of **finite** length!

Continued Fractions

How do we expand a given CF?

[0; 1, 4, 1, 2, 2] = ??

a	h	k	h/k	x
0	0	1	0.00000000	0.82500000
1	1	1	1.00000000	0.82500000
4	4	5	0.80000000	0.82500000
1	5	6	0.83333333	0.82500000
2	14	17	0.82352941	0.82500000
2	**33**	**40**	0.82500000	0.82500000

Each row of h & k's give a better and better approximation to the original number

[0; 1, 4, 1, 2, 2] = 0.825 (= **33/40**)

$\sqrt{2}$ to 3,600 digits

1.4142135623730950488016887242096980785696718753769480731766797379907324784621070388503875343276415727350138
462309122970249248360558507372126441214970999358314132226659275055927557999505011527820605714701095599716059
702745345968620147285174186408891986095523292304843087143214508397626036279952514079896872533965463318088296
406206152583523950547457502877599617298355752203375318570113543746034084988471603868999706990048150305440277
903164542478230684929369186215805784631115966687130130156185689872372352885092648612494977154218334204285686
060146824720771435854874155657069677653720226485447015858801620758474922657226002085584466521458398893944370
926591800311388246468157082630100594858704003186480342194897278290641045072636881313739855256117322040245091
227700226941127573627280495738108967504018369868368450725799364729060762996941380475654823728997180326802474
420629269124859052181004459842150591120249441341728531478105803603371077309182869314710171111683916581726889
419758716582152128229518488472089694633862891562882765952635140542267653239694617511291602408715510135150455
381287560052631468017127402653969470240300517495318862925631385188163478001569369176881852378684052287837629
389214300655869568685964595155501644724509836896036887323114389415576651040883914292338113206052433629485317
049915771756228549741438999188021762430965206564211827316726257539594717255934637238632261482742622208671155
839599926521176252698917540988159348640083457085181472231814204070426509056532333398436457865796796519267292
399875366617215982578860263363617827495994219403777753681426217738799194551397231274066898329989895386728822
856378697749662519966583525776198939322845344735694794962952168891485492538904755828834526096524096542889394
538646625744927556381964410316979833061852019379384940057156333720548068540575867999670121372239475821426306
585132217408832382947287617393647467837431960001592188807347857617252211867490424977366929207311096369721608
933708661156734585334833295254675851644710757848602463600834449114818587655554286455123314219926311332517970
608436559704352856410087918500760361009159465670676883605571740076756905096136719401324935605240185999105062
108163597726431380605467010293569971042425105781749531057255934984451126922780344913506637568747760283162829
605532422426957534529028838768446429173282770888318087025339852338122749990812371892540726475367850304821591
801886167108972869229201197599880703818543332536460211082299279293072871780799888099176741774108983060800326
311816427988231171543638696617029999341616148786860180455055539869131151860103863753250045581860448040750241
195184305674533683613674597374423988553285179308960373898915173195874134428817842125021916951875593444387396
189314549999906107587049090260883517636224749757858858368037457931157339802099986622186949922595913276423619
410592100328026149874566599688874067956167391859572888642473463585886864496822386006983352642799056283165613
913942557649062065186021647263033362975075697870606606856498160092718709292153132368281356988937097416504474
590960537472796524477094099241238710614470543986743647338477454819100872886222149589529591187892149179833981
083788278153065562315810360648675873036014502273208829351341387227684176678436905294286984908384557445794095
986260742499549168028530773989382960362133539875320509199893607513906444495768456993471276364507163279154701
597733548638939423257277540038260274785674172580951416307159597849818009443560379390985590168272154034581581
521004936662953448827107292396602321638238266612626830502572781169451035379371568823365932297823192986064679
789864092085609558142614363631004615594332550474493975933999125419532300932175304476533964706627611661753518
754646209676345587386164880198848497479264045065444896910040794211816925796857563784881498986416854994916357
614484047021033989215342377037233353115645944389703653166721949049351882905806307401346862641672470110653463
493916407146285

Continued Fractions

What is the continued fraction expansion for $\sqrt{2}$?

x	**1/x**	**INT(1/x)**	**1/x - INT(1/x)**
	1.41421356	1	0.41421356
0.41421356	2.41421356	2	0.41421356
0.41421356	2.41421356	2	0.41421356
0.41421356	2.41421356	2	0.41421356
0.41421356	2.41421356	2	0.41421356

$\sqrt{2}$ = [1; {2}]

Numbers within {} are repeated

All irrational numbers yield an infinite CF with a repeated *sequence* of finite length!

There is simple order behind the chaos!

Continued Fractions

What fraction best approximates $\sqrt{2}$?

a	h	k	h/k	x
1	1	1	1.00000000	1.41421356
2	3	2	1.50000000	1.41421356
2	7	5	1.40000000	1.41421356
2	17	12	1.41666667	1.41421356
2	41	29	1.41379310	1.41421356
2	99	70	1.41428571	1.41421356
2	239	169	1.41420118	1.41421356
2	577	408	1.41421569	1.41421356
2	1393	985	1.41421320	1.41421356
2	3363	2378	1.41421362	1.41421356
2	8119	5741	1.41421355	1.41421356
2	19601	13860	1.41421356	1.41421356

$\sqrt{2} \approx$ **19,601 / 13,860**

Continued Fractions

What is the continued fraction expansion for $\sqrt{113}$?

x	1/x	INT(1/x)	1/x - INT(1/x)
10.63014581		10	0.63014581
0.63014581	1.58693429	1	0.58693429
0.58693429	1.70376823	1	0.70376823
0.70376823	1.42092235	1	0.42092235
0.42092235	2.37573512	2	0.37573512
0.37573512	2.66144940	2	0.66144940
0.66144940	1.51183144	1	0.51183144
0.51183144	1.95376823	1	0.95376823
0.95376823	1.04847275	1	0.04847275
0.04847275	20.63014581	20	0.63014581
0.63014581	1.58693430	1	0.58693430
0.58693430	1.70376822	1	0.70376822
0.70376822	1.42092237	1	0.42092237
0.42092237	2.37573499	2	0.37573499
0.37573499	2.66145027	2	0.66145027
0.66145027	1.51182945	1	0.51182945
0.51182945	1.95377581	1	0.95377581
0.95377581	1.04846442	1	0.04846442
0.04846442	20.63369395	20	0.63369395

$\sqrt{113}$ = [10; {1,1,1,2,2,1,1,1,20}]

Continued Fractions

What fraction best approximates $\sqrt{113}$?

a	h	k	h/k	x
10	10	1	10.00000000	10.63014581
1	11	1	11.00000000	10.63014581
1	21	2	10.50000000	10.63014581
1	32	3	10.66666667	10.63014581
2	85	8	10.62500000	10.63014581
2	202	19	10.63157895	10.63014581
1	287	27	10.62962963	10.63014581
1	489	46	10.63043478	10.63014581
1	776	73	10.63013699	10.63014581
20	16009	1506	10.63014608	10.63014581
1	16785	1579	10.63014566	10.63014581
1	32794	3085	10.63014587	10.63014581
1	49579	4664	10.63014580	10.63014581
2	131952	12413	10.63014581	10.63014581
2	313483	29490	10.63014581	10.63014581
1	445435	41903	10.63014581	10.63014581
1	758918	71393	10.63014581	10.63014581
1	1204353	113296	10.63014581	10.63014581
20	24845978	2337313	10.63014581	10.63014581

$\sqrt{113} \approx$ **131,952 / 12,413**

e to 3,600 digits

2.718281828459045235360287471352662497757247093699959574966967627724076630353547594571382178525166427427466 3
91932003059921817413596629043572900334295260595630738132328627943490763233829880753195251019011573834187930 7
02154089149934884167509244761460668082264800168477411853742345442437107539077744992069551702761838606261331 3
84583000752044933826560297606737113200709328709127443747047230696977209310141692836819025515108657463772111 2
52389784425056953696770785449969967946864454905987931636889230098793127736178215424999229576351482208269895 1
93668033182528869398496465105820939239829488793320362509443117301238197068416140397019837679320683282376464 8
04295311802328782509819455815301756717361332069811250996181881593041690351598888519345807273866738589422879 2
28499892086805825749279610484198444363463244968487560233624827041978623209002160990235304369941849146314093 4
31738143640546253152096183690888707016768396424378140592714563549061303107208510383750510115747704171898610 6
87396965521267154688957035035402123407849819334321068170121005627880235193033224745015853904730419957777093 5
03660416997329725088687696640355570716226844716256079882651787134195124665201030592123667719432527867539855 8
94489697096409754591856956380236370162112047742722836489613422516445078182442352948636372141740238893441247 9
63574370263755294448337998016125492278509257782562092622648326277933386566481627725164019105900491644998289 3
15056604725802778631864155195653244258698294695930801915298721172556347546396447910145904090586298496791287 4
06870504895858671747985466775757320568128845920541334053922000113786300945560688166740016984205580403363795 3
76452030402432256613527836951177883863874439662532249850654995886234281899707733276171783928034946501434558 8
97071942586398772754710962953741521115136835062752602326484728703920764310059584116612054529703023647254929 6
66938115137322753645098889031360205724817658511806303644281231496550704751025446501172721155519486685080036 8
53228183152196003735625279449515828418829478761085263981395599006737648292244375287184624578036192981971399 1
47564488262603903381441823262515097482798777996437308997038886778227138360577297882412561190717663946507063 3
04527954661855096666185664709711344474016070462621568071748187784437143698821855967095910259686200235371858 8
74856965220005031173439207321139080329363447972735595527734907178379342163701205005451326383544000186323991 4
90705479778056697853358048966906295119432473099587655236812859041383241160722602998330535370876138939639177 9
57454016137223618789365260538155841587186925538606164779834025435128439612946035291332594279490433729908573 1
58029095863138268329147711639633709240031689458636060645845925126994655724839186564209752685082307544254599 3
76917041977780085362730941710163434907696423722294352366125572508814779223151974778060569672538017180776360 3
46245927877846585065605078084421152969752189087401966090665180351650179250461950136658543663271254963990854 9
14442000145747608193022120660243300964127048943903971771951806990869986066365832322787093765022601492910115 17
77635944602023249300280401867723910288097866605651183260043688508817157238669842242201024950551881694803221
00251542649463981287367765892768816359831247788652014117411091360116499507662907794364600585194199856016264 7
90761532103872755712699251827568798930276176114616254935649590379804583818232336861201624373656984670378585 3
30527583333793990752166069238053369887956513728559388349989470741618155012539706464817194670834819721448889 8
79067650379590366967249499254527903372963616265897603949857674139735944102374432970935547798262961459144293 6
45142861715858733974679189757121195618738578364475844842355558105002561149239151889309946342841393608038309 1
66281881150371528496705974162562823609216807515017772538740256425347087908913729172282861151591568372524163 0
77225440633787593105982676094420326192428531701878177296023541306067213604600038966109364709514141718577701 4
180606443636815

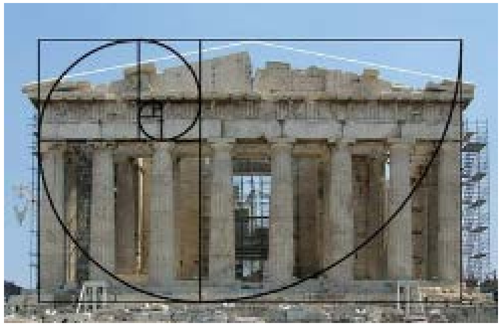
Continued Fraction for *e*

x	1/x	INT(1/x)	1/x - INT(1/x)
2.71828183		2	0.71828183
0.71828183	1.39221119	1	0.39221119
0.39221119	2.54964678	**2**	0.54964678
0.54964678	1.81935024	1	0.81935024
0.81935024	1.22047929	1	0.22047929
0.22047929	4.53557348	**4**	0.53557348
0.53557348	1.86715744	1	0.86715744
0.86715744	1.15319313	1	0.15319313
[illegible]	[illegible]	**6**	0.52770793
[illegible]	[illegible]	1	0.89498763
[illegible]	[illegible]	1	0.11733388
[illegible]	[illegible]	**8**	0.52268767
0.52268767	1.91318841	1	0.91318841
0.91318841	1.09506427	1	0.09506427
0.09506427	10.51919947	**10**	0.51919947
0.51919947	1.92604201	1	0.92604201
0.92604201	1.07986461	1	0.07986461
0.07986461	12.52119027	**12**	0.52119027
0.52119027	1.91868508	1	0.91868508

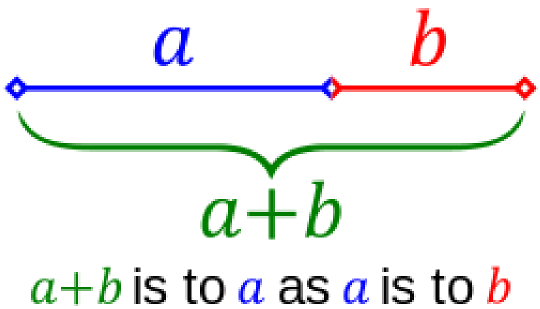
All transcendental numbers yield an infinite CF with a repeated *pattern* of finite length

***e* = [2; {1,2n,1}] for n > 0**

e^2 = [7;2,{1,1,3n,12n+6,3n+2}] for n > 0



The Golden Ratio



$$1 + \frac{1}{\varphi} = \varphi$$

$$\varphi + 1 = \varphi^2$$

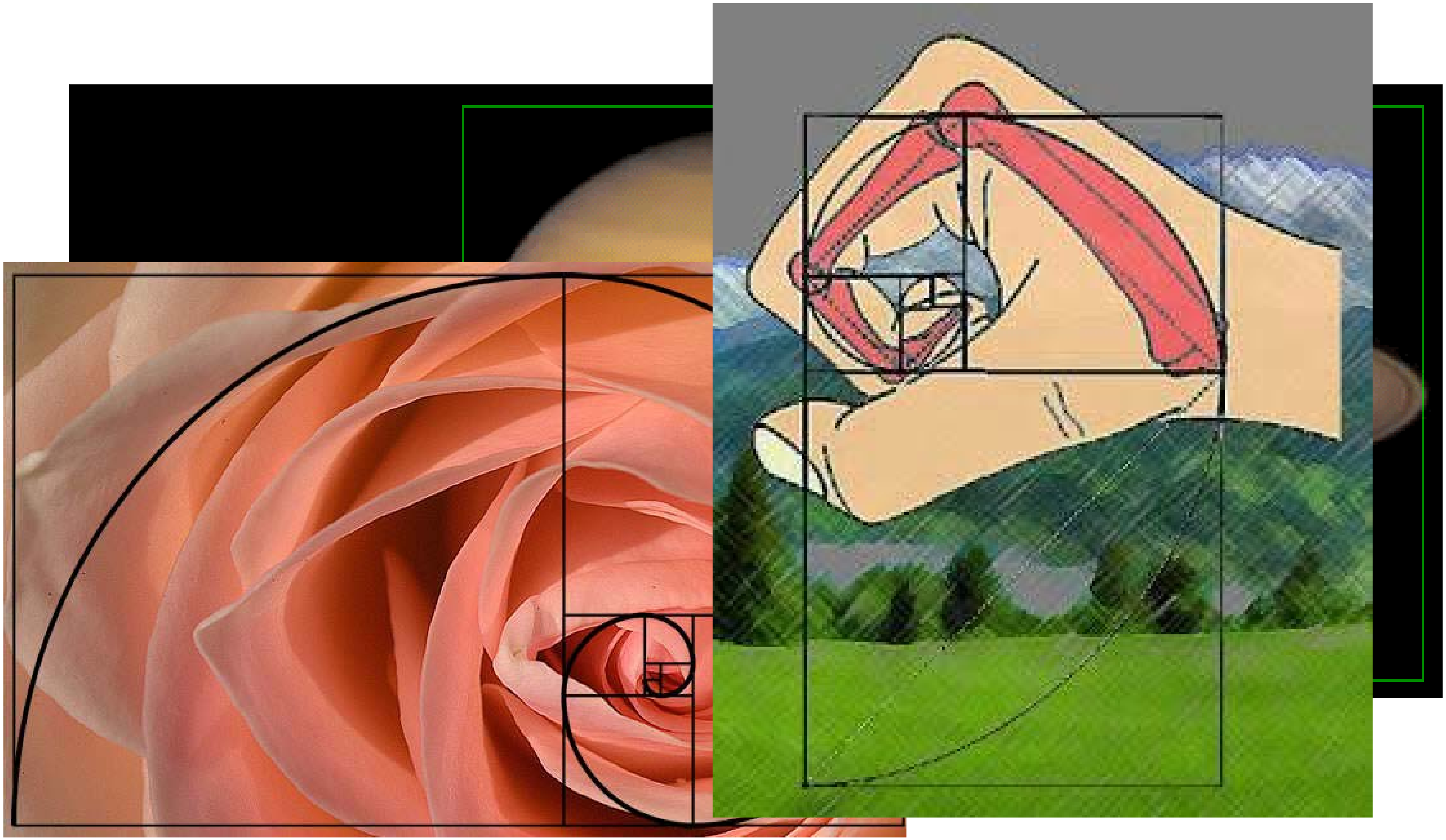
$$\frac{a+b}{a} = \frac{a}{b} = \varphi$$

$$\varphi^2 - \varphi - 1 = 0$$

$$1 + \frac{b}{a} = \frac{a}{b} = \varphi$$

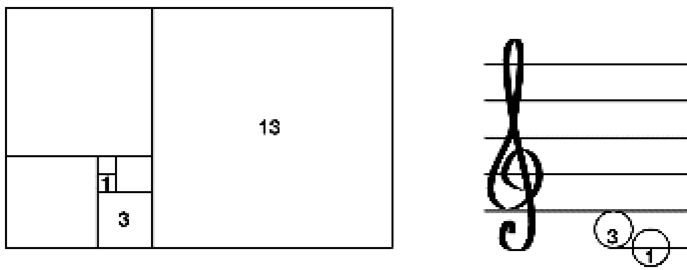
$$\varphi = \frac{1+\sqrt{5}}{2} = 1.6180339887\ldots$$

$$\varphi = \frac{1+\sqrt{5}}{2} = 1.6180339887\ldots$$

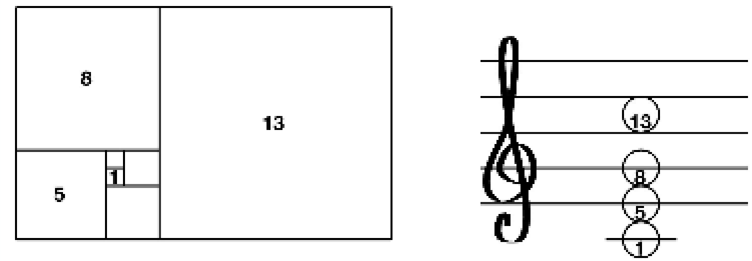


$$\varphi = \frac{1+\sqrt{5}}{2} = 1.6180339887\ldots$$

Whole Step



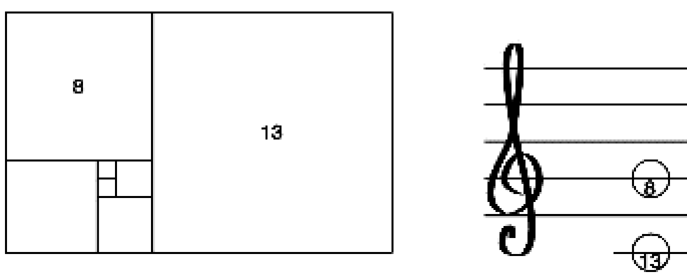
Major Triad



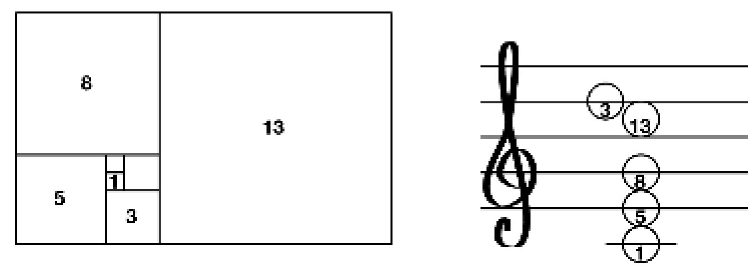
IDEAL PROPORTIONS

IDEAL CHORD STRUCTURE

Perfect Fifth



Major 9



$$\varphi = \frac{1+\sqrt{5}}{2} = 1.6180339887 \ldots$$

The greatest of luthiers, Stradivarius, designed his violins around the golden ratio (ϕ). His violins are the most valuable and precious instruments in the string-playing world because of their exquisite tonal and harmonic qualities, [2]. The Stradivarius violin in Fig. 2 reveals how precisely his instruments are determined by the golden ratio, [3]:

$$\frac{a1+a2}{a2} = \frac{a2}{a1} = \frac{b2}{b1} = \frac{b2}{c2} = \frac{c2}{c1} = \phi$$

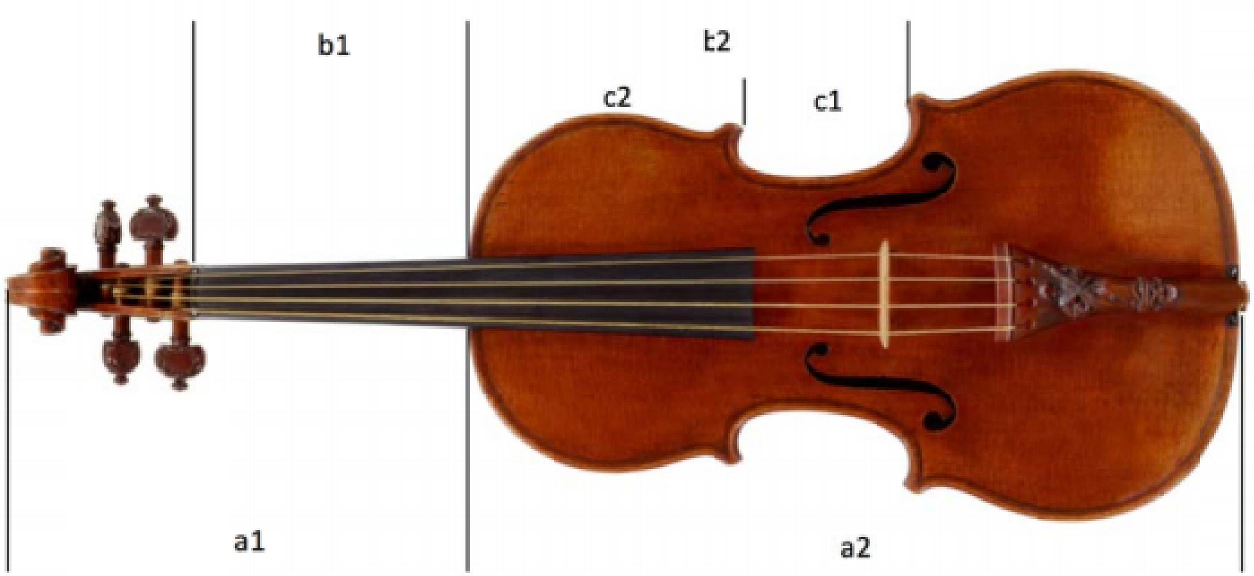


Figure 2. Photo of "Lady Blunt" Stradivarius violin (sold for nearly $16M).

φ to 3,600 digits

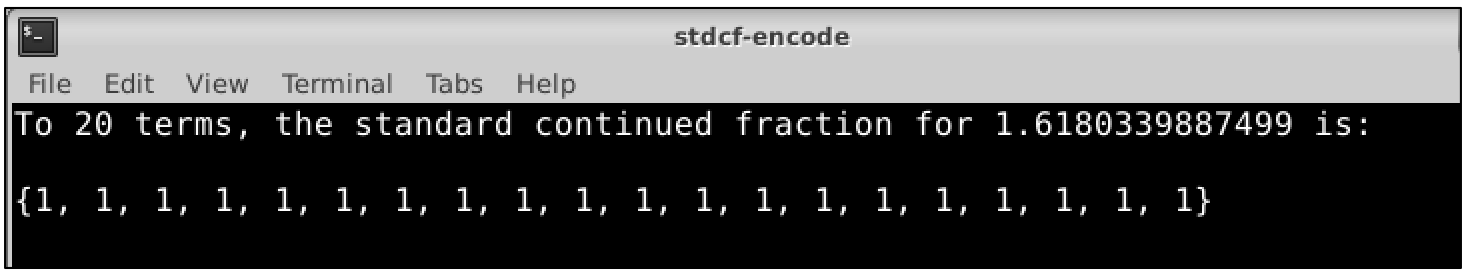
1.61803398874989484820458683436563811772030917980576286213544862270526046281890244970720720418939113748475408807538689175212663386222353693179318006076672635443338908659593958290563832266131992829026788067520876689250171169620703222104321626954862629631361443814975870122034080588795445474924618569536486444924104432077134494704956584678850987433944221254487706647809158846074998871240076521705751797883416625624940758906970400028121042762177111777805315317141011704666599146697987317613560067087480710131795236894275219484353056783002287856997829778347845878228911097625003026961561700250464338243776486102838312683303724292675263116533924731671112115881863851331620384005222165791286675294654906811317159934323597349498509040947621322298101726107059611645629909816290555208524790352406020172799747175342777592778625619432082750513121815628551222480939471234145170223735805772786160086883829523045926478780178899219902707769038953219681986151437803149974110692608867429622675756052317277752035361393621076738937645560606059216589466759551900400555908950229530942312482355212212415444006470340565734797663972394949946584578873039623090375033993856210242369025138680414577995698122445747178034173126453220416397232134044449487302315417676893752103068737880344170093954409627955898678723209512426893557309704509595684401755519881921802064052905518934947592600734852282101088194644544222318891319294689622002301443770269923007803085261180754519288770502109684249362713592518760777884665836150238913493333122310533923213624319263728910670503399282265263556209029798642472759772565508615487543574826471814145127000602389016207773224499435308899909501680328112194320481964387675863314798571911397815397807476150772211750826945863932045652098969855567814106968372884058746103378105444390943683583581381131168993855576975484149144534150912954070050194775486163075422641729394680367319805861833918328599130396072014455950449779212076124785645916160837059498786006970189409886400764436170933417270919143365013715766011480381430626238051432117348151005590134561011800790506381421527093085880928757034505078081454588199063361298279814117453392731208092897279222132980642946878242748740174505540677875708323731097591511776297844328474790817651809778726841611763250386121129143683437670235037111633072586988325871033632223810980901211019899176841491751233134015273384383723450093478604979294599158220125810459823092552872124137043614910205471855496118087642657651106054588147560443178479858453973128630162544876114852021706440411166076695059775783257039511087823082710647893902111569103927683845386333321565829659773103436032322545743637204124406408882673758433953679593123221343732099574988946995656473600729599983912881031974263125179714143201231127955189477817269141589117799195648125580018455065632952859859100090862180297756378925999164994642819302229355234667475932695165421402109136301819472270789012208728736170734864999815625547281137347987165695274890081443840532748378137824669174442296349147081570073525457070897726754693438226195468615331209533579238014609273510210119190218360675097308957528957746814229543394385493155339630380729169175846101460995055064803679304147236572039860073550760902317312501613204843583648177048481810991602442523271672190189334596378608787528701739359303013359011237102391712659047026349402830766876743638651327106280323174069317334482343564531850581353108549733350759966778712449058363675413289086240632456395357212524261170278028656043234942837301725574405837278267996031739364013287627701243679831144643694767053127249241047167001382478312865650649343418039004101780533950587724586655755229391582397084177298337282311152

Open Lab 1 –Standard CF Encoding

```
int main()
{
    double x = (1 + sqrt(5)) / 2;

    int maxTerms = 20;

    cout << "To " << maxTerms << " terms, "
        << "the standard continued fraction for "
        << setprecision(14) << x << " is:\n" << endl
        << "{" << (int)(x) << ", ";

    x = x - int(x);

    for (int terms = 1;terms < maxTerms;terms++)
    {
        cout << (int)(1 / x);
        if (terms < maxTerms - 1) cout << ", ";
        x = 1 / x - (int)(1 / x);
    }

    cout << "}" << endl << endl;

    return 0;
}
```

Run Lab 1 – Standard CF Encoding

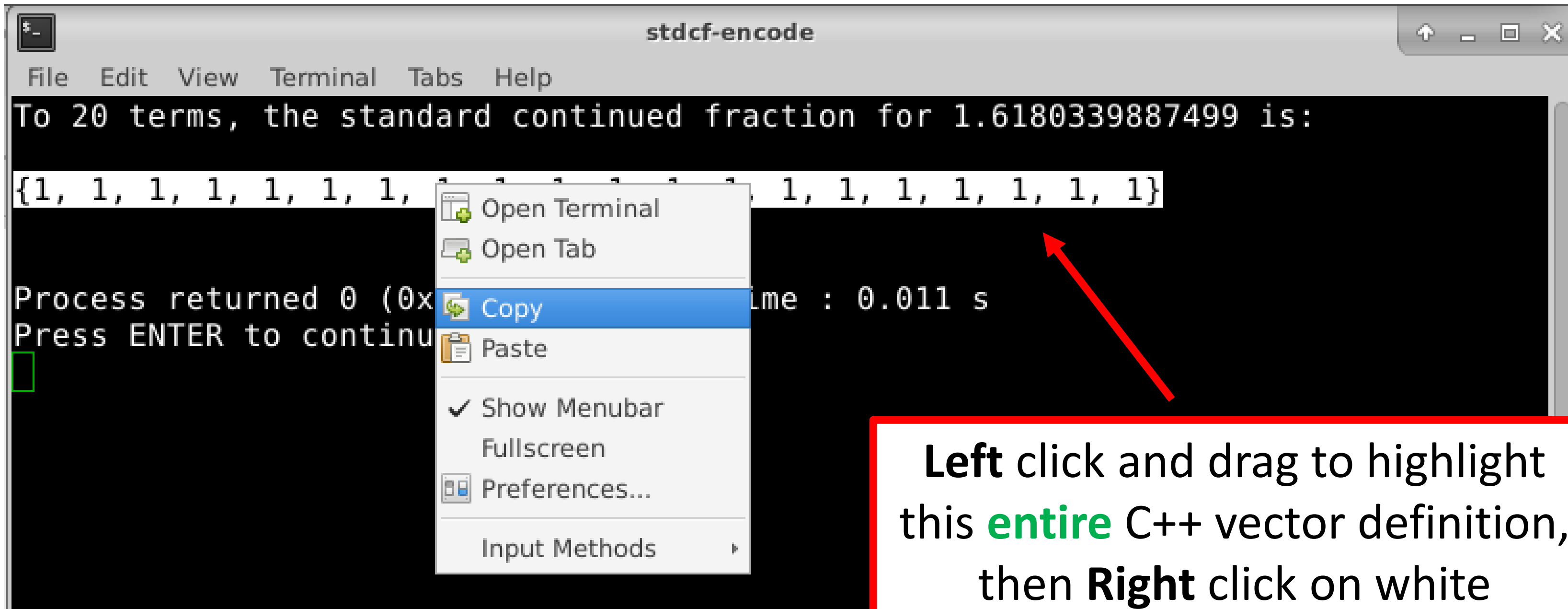
- Generate the Standard CF for the golden ratio $\frac{1+\sqrt{5}}{2}$



$$\boxed{\varphi = [1; \{1\}]}$$

This is Mother Nature's ***Unit***…
It is the most simple infinite CF possible

Check Lab 1 – Standard CF Encoding



Left click and drag to highlight this **entire** C++ vector definition, then **Right** click on white highlighted line, and then select **Copy** to place it in the clipboard

C++ Vector Initialization

```
vector<int> cf{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
```

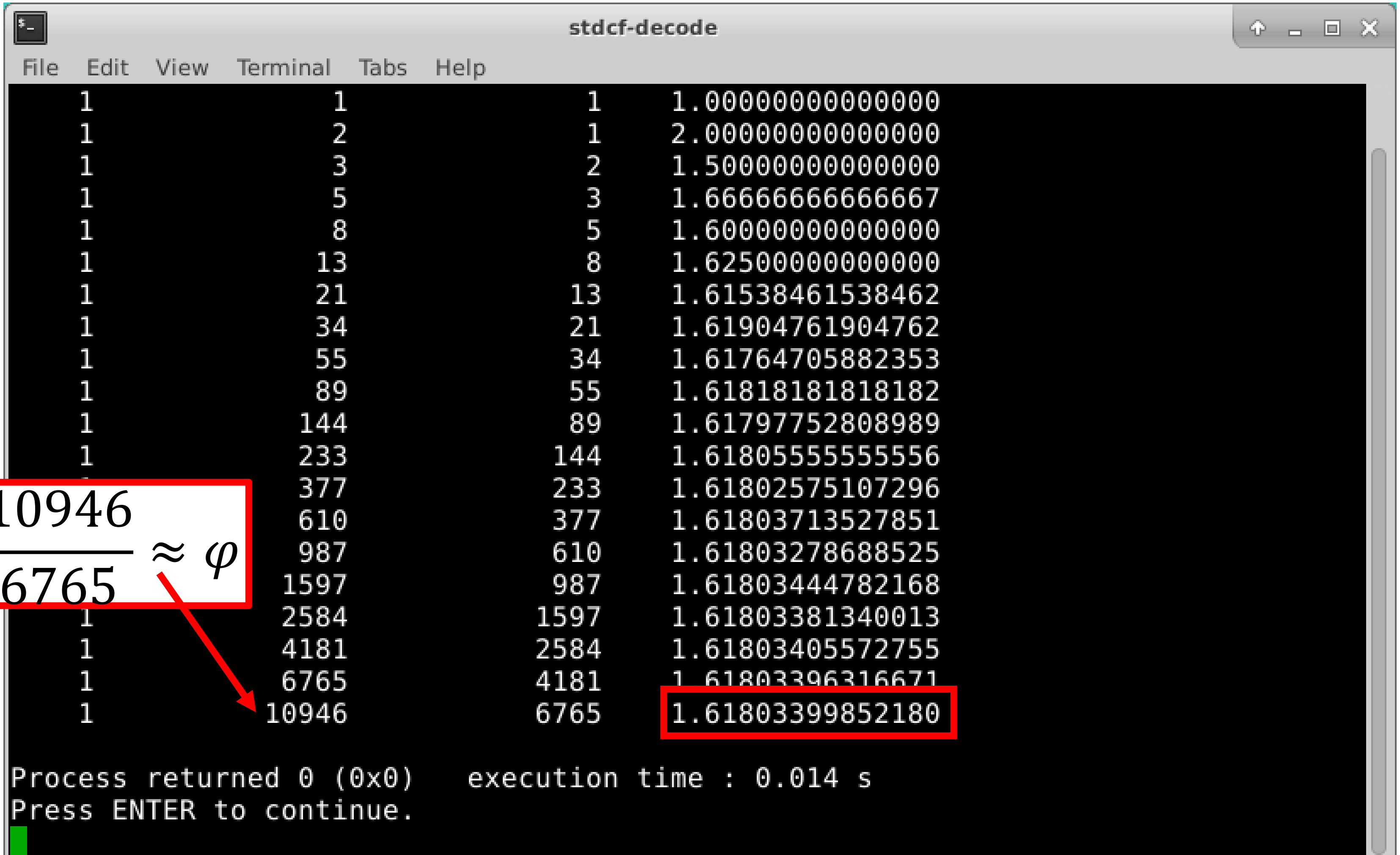
- Vectors can be defined using the **list initializer** syntax
- Elements are comma separated between curly braces
- First item in list goes into index position **0** in the array
- The vector is dynamically sized to match the number of elements in the initializer list
- The curly brace syntax after the variable name **cf** is just exactly the output from **Lab 2!**

Edit Lab 2 – Standard CF Decoding

```
int main()
{
    int maxTerms = 20;

    vector<int> cf{ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };

    vector<double> h(maxTerms + 2);
    vector<double> k(maxTerms + 2);

    h.at(0) = 0; k.at(0) = 1;
    h.at(1) = 1; k.at(1) = 0;

    cout << "Using " << maxTerms << " terms, ";
    cout << "the continued fraction expansion is:" << endl;
    cout << setw(5) << "a";
    cout << right << setw(15) << "h";
    cout << right << setw(15) << "k";
    cout << setw(20) << "convergent" << endl;

    for (int n{ 2 }; n < maxTerms + 2; ++n) {
        double a = cf.at(n - 2);

        h.at(n) = a * h.at(n - 1) + h.at(n - 2);
        k.at(n) = a * k.at(n - 1) + k.at(n - 2);

        double convergent = h.at(n) / k.at(n);

        cout << setprecision(0) << right
            << setw(5) << a << setw(15) << h[n] << setw(15) << k[n]
            << setprecision(14) << fixed << setw(20) << convergent << endl;
    }

    cout << endl;
    system("pause");
    return 0;
}
```

Right click and paste in the output from Lab 4

Run Lab 2 – Standard CF Decoding



$$\frac{10946}{6765} \approx \varphi$$

π to 3,600 digits

3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808
8651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566659
334461284756482337867831652712019091456485669234603486104543266482133936072602491412737245870066063155881748
815209209628292540917153643678925903600113305305488204665213841469519415116094330572703657595919530921861173
819326117931051185480744623799627495673518857527248912279381830119491298336733624406566430860213949463952247
371907021798609437027705392171762931767523846748184676694051320005681271452635608277857713427577896091736371
787214684409012249534301465495853710507922796892589235420199561121290219608640344181598136297747713099605187
072113499999983729780499510597317328160963185950244594553469083026425223082533446850352619311881710100031378
387528865875332083814206171776691473035982534904287554687311595628638823537875937519577818577805321712268066
130019278766111959092164201989380952572010654858632788659361533818279682303019520353018529689957736225994138
912497217752834791315155748572424541506959508295331168617278558890750983817546374649393192550604009277016711
390098488240128583616035637076601047101819429555961989467678374494482553797747268471040475346462080466842590
694912933136770289891521047521620569660240580381501935112533824300355876402474964732639141992726042699227967
823547816360093417216412199245863150302861829745557067498385054945885869269956909272107975093029553211653449
872027559602364806654991198818347977535663698074265425278625518184175746728909777727938000816470600161452491
921732172147723501414419735685481613611573525521334757418494684385233239073941433345477624168625189835694855
620992192221842725502542568876717904946016534668049886272327917860857843838279679766814541009538837863609506
800642251252051173929848960841284886269456042419652850222106611863067442786220391949450471237137869609563643
719172874677646575739624138908658326459958133904780275900994657640789512694683983525957098258226205224894077
267194782684826014769909026401363944374553050682034962524517493996514314298091906592509372216964615157098583
874105978859597729754989301617539284681382686838689427741559918559252459539594310499725246808459872736446958
486538367362226260991246080512438843904512441365497627807977156914359977001296160894416948685558484063534220
722258284886481584560285060168427394522674676788952521385225499546667278239864565961163548862305774564980355
936345681743241125150760694794510965960940252288797108931456691368672287489405601015033086179286809208747609
178249385890097149096759852613655497818931297848216829989487226588048575640142704775551323796414515237462343
645428584447952658678210511413547357395231134271661021359695362314429524849371871101457654035902799344037420
073105785390621983874478084784896833214457138687519435064302184531910484810053706146806749192781911979399520
614196634287544406437451237181921799983910159195618146751426912397489409071864942319615679452080951465502252
316038819301420937621378559566389377870830390697920773467221825625996615014215030680384477345492026054146659
252014974428507325186660021324340881907104863317346496514539057962685610055081066587969981635747363840525714
591028970641401109712062804390397595156771577004203378699360072305587631763594218731251471205329281918261861
258673215791984148488291644706095752706957220917567116722910981690915280173506712748583222871835209353965725
121083579151369882091444210067510334671103141267111369908658516398315019701651511685171437657618351556508849
099898599823873455283316355076479185358932261854896321329330898570642046752590709154814165498594616371802709
819943099244889575712828905923233260972997120844335732654893823911932597463667305836041428138830320382490375
898524374417029132765618093773444030707469211201913020330380197621101100449293215160842444859637669838952286
847831235526582131449576857262433441893039686426243410773226978028073189154411010446823252716201052652272111
847831235526582

Continued Fractions

What is the continued fraction expansion for π ?

x	1/x	INT(1/x)	1/x - INT(1/x)
3.141592654		3	0.141593
0.141593	7.062513	7	0.062513
0.062513	15.996594	15	0.996594
0.996594	1.003417	1	0.003417
0.003417	292.634591	292	0.634591
0.634591	1.575818	1	0.575818
0.575818	1.736659	1	0.736659
0.736659	1.357481	1	0.357481
0.357481	2.797351	2	0.797351
0.797351	1.254153	1	0.254153
0.254153	3.934642	3	0.934642
0.934642	1.069928	1	0.069928
0.069928	14.300420	14	0.300420
0.300420	3.328678	3	0.328678
0.328678	3.042495	3	0.042495
0.042495	23.532265	23	0.532265
0.532265	1.878762	1	0.878762
0.878762	1.137964	1	0.137964
0.137964	7.248258	7	0.248258

π = [3;7,15,1,292,1,1,1,2,1,3,1,14,3,3,23,1,1,7….]

(no repeated *pattern* of finite length ☹!)

Continued Fractions

a	h	k	h/k	x
3	3	1	3	3.141592654
7	22	7	3.142857143	3.141592654
15	333	106	3.141509434	3.141592654
1	355	113	3.14159292	3.141592654
292	103993	33102	3.141592653	3.141592654
1	104348	33215	3.141592654	3.141592654
1	208341	66317	3.141592653	3.141592654
1	312689	99532	3.141592654	3.141592654
2	833719	265381	3.141592654	3.141592654
1	1146408	364913	3.141592654	3.141592654
3	4272943	1360120	3.141592654	3.141592654
1	5419351	1725033	3.141592654	3.141592654
14	80143857	25510582	3.141592654	3.141592654
3	245850922	78256779	3.141592654	3.141592654
3	817696623	260280919	3.141592654	3.141592654
23	19052873251	6064717916	3.141592654	3.141592654
1	19870569874	6324998835	3.141592654	3.141592654
1	38923443125	12389716751	3.141592654	3.141592654
7	2.92335E+11	93053016092	3.141592654	3.141592654

If measuring the circumference of Earth:

22 / 7 = accurate to between this classroom and Washington, DC

355 / 113 = accurate to between this classroom and the main parking lot

If measuring the distance between Earth & Sun:

355 / 113 = accurate to 4 football fields

104348 / 33215 = accurate to the length of my shoe

Generalized Continued Fractions

A generalized continued fraction is an expression of the form

$$x = b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 + \cfrac{a_4}{b_4 + \ddots}}}}$$

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3}}}$$

This is the ***standard*** CF expression where the numerators are all 1's

Generalized Continued Fractions

What is a **generalized** continued fraction expansion for π ?

$$\pi = 3 + \cfrac{1^2}{6 + \cfrac{3^2}{6 + \cfrac{5^2}{6 + \ddots}}}$$

$$\boldsymbol{\pi = [3;\{(2n+1)^2 \mid 6\}]}$$

All the mysterious and unpredictable digits of PI
come from this simple generalized CF !!

Generalized Continued Fractions

What is a **generalized** continued fraction expansion for π ?

index n	n	a	b	h	k	h/k	delta
0	-2			0	1		
1	-1	1		1	0		
2	0	1	3	3	1	3.000000000000	0.141592653590
3	1	9	6	19	6	3.166666666667	-0.025074013077
4	2	25	6	141	45	3.133333333333	0.008259320256
5	3	49	6	1321	420	3.145238095238	-0.003645441648
6	4	81	6	14835	4725	3.139682539683	0.001910113907
		121	6	196011	62370	3.142712842713	-0.001120189123
		169	6	2971101	945945	3.140881340881	0.000711312708
		225	6	50952465	16216200	3.142071817072	-0.000479163482
10	8	289	6	974212515	310134825	3.141254823608	0.000337829982
11	9	361	6	20570537475	6547290750	3.141839618929	-0.000246965340
12	10	441	6	4.75114E+11	1.51242E+11	3.141406718497	0.000185935093
13	11	529	6	1.19223E+13	3.79481E+12	3.141736099261	-0.000143445671

$$h_n=b_n*h_{(n-1)}+a_{(n-1)}*h_{(n-2)}$$
$$k_n=b_n*k_{(n-1)}+a_{(n-1)}*k_{(n-2)}$$

$$\pi = [3;\{(2n+1)^2 \mid 6\}]$$

All the mysterious and unpredictable digits of PI come from this simple generalized CF !!

Pell's Equation

- Your scientist has asked you to write a C++ program (given $x, y, \boldsymbol{n} \in \mathbb{Z}^+$) to find $\boldsymbol{x}$ & $\boldsymbol{y}$ for successive $\boldsymbol{n}$ such that:

$$x^2 - \boldsymbol{n}y^2 = 1$$

- Check all integers $2 \leq \boldsymbol{n} \leq 70$ and $1 \leq x \leq \mathbf{70,000}$

- Why is there is no need to check for $y > \left\lfloor \sqrt{\frac{x^2}{n}} \right\rfloor$?

- Do you see any relationship between the specific $\boldsymbol{x}$ & $\boldsymbol{y}$ values that solve the equation for each successive $\boldsymbol{n}$ value?

Open Lab 3 - Pell's Equation

```
int main()
{
    DisplayHeader();

    const uint64_t xMax = 70000;

    for (int n = 2; n <= 70; n++) {
        cout << setw(4) << n;
        bool foundSolution = false;
        uint64_t x = 1;
        while ((x <= xMax) && !foundSolution) {
            uint64_t xSqr = (uint64_t)x * x;
            uint64_t y = 1;
            uint64_t yMax = sqrt(xSqr / n);
            while ((y <= yMax) && !foundSolution) {
                uint64_t ySqr = (uint64_t)y * y;
                uint64_t lhs = xSqr - (uint64_t)n * ySqr;
                if (lhs == 1) {
                    cout << setw(8) << x
                         << setw(8) << y;
                    foundSolution = true;
                }
                y++;
            }
            x++;
        }
        if (!foundSolution)
            cout << setw(8) << "-"
                 << setw(8) << "-";
        cout << endl;
    }
    return 0;
}
```

$$x^2 - ny^2 = 1$$

As soon as a valid solution is found for the current value of **n**, then stop trying any more **x** & **y** values

Hyphens indicate no solution was found in the allowed search space

Run Lab 3
Pell's Equation

n	x	y
2	3	2
3	2	1
4	-	-
5	9	4
6	5	2
7	8	3
8	3	1
9	-	-
10	19	6
11	10	3
12	7	2
13	649	180
14	15	4
15	4	1
16	-	-
17	33	8
18	17	4
19	170	39
20	9	2
21	55	12
22	197	42
23	24	5
24	5	1
25	-	-
26	51	10
27	26	5
28	127	24
29	9801	1820
30	11	2
31	1520	273
32	17	3
33	23	4
34	35	6
35	6	1
36	-	-
37	73	12
38	37	6
39	25	4
40	19	3
41	2049	320
42	13	2
43	3482	531
44	199	30
45	161	24
46	24335	3588
47	48	7
48	7	1
49	-	-
50	99	14
51	50	7
52	649	90
53	66249	9100
54	485	66
55	89	12
56	15	2
57	151	20
58	19603	2574
59	530	69
60	31	4
61	-	-
62	63	8
63	8	1
64	-	-
65	129	16
66	65	8
67	48842	5967
68	33	4
69	7775	936
70	251	30

Check Lab 3 – Observations

- Which values of **n** have no solution?

$$\boldsymbol{n} = 1, 4, 9, 16, 25, 36, 49, \mathbf{61}, 64, \ldots$$

- Some of the values for **x** & **y** are much bigger than for other close values of **n**:

40	19	3
41	2049	320
42	13	2
43	3482	531
44	199	30
45	161	24
46	24335	3588
47	48	7
48	7	1

- The magnitude of **n** does not seem to be a good predictor about the magnitude of the **x** & **y** values that solve the equation for that specific **n**

Pell's Equation: Period of Standard CF

Small values for x & y

n	x	y
35	6	1
47	48	7
60	31	4
68	33	4

Period = 2

Std. CF

$\sqrt{68} =$ {8, 4, 16, 4, 16,

Large values for x & y

n	x	y
13	649	180
29	9801	1820
41	2049	320
43	3482	531
46	24335	3588
53	66249	9100
61	1766319049	226153980
67	48842	5967

Period = 5

$\sqrt{29} =$ {5, 2, 1, 1, 2, 10, 2, 1, 1, 2, 10,

Period = 5

$\sqrt{53} =$ {7, 3, 1, 1, 3, 14, 3, 1, 1, 3, 14,

Period = 11

$\sqrt{61} =$ {7, 1, 4, 3, 1, 2, 2, 1, 3, 4, 1, 14, 1, 4, 3, 1, 2, 2, 1,

Continued Fractions

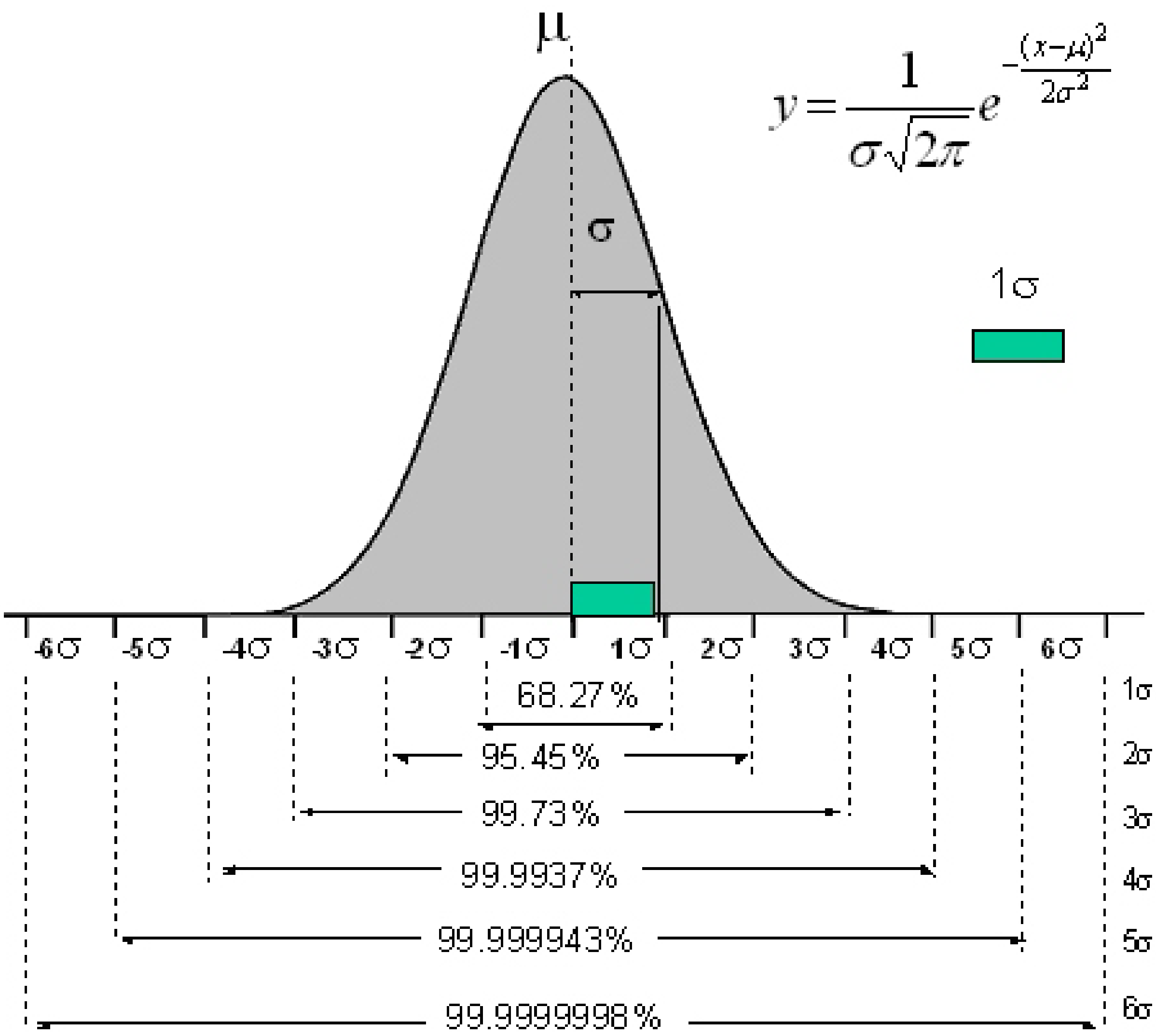
- CFs *may* have their own rich **arithmetic**, **algebra**, and potentially even their own **calculus**
  - How can one directly **divide** two CFs?
  - How can one directly take the **sin()** of a CF?
  - What does the **factorial** of a CF look like?
- In many ways a CF is a “more accurate” representation of an irrational or transcendental number
  - The sum of an infinite series must stop somewhere - after that all the remaining precision digits are lost
  - A CF encodes the entire number **with no loss of precision**
  - What can you discover about CFs?

Normal Distribution

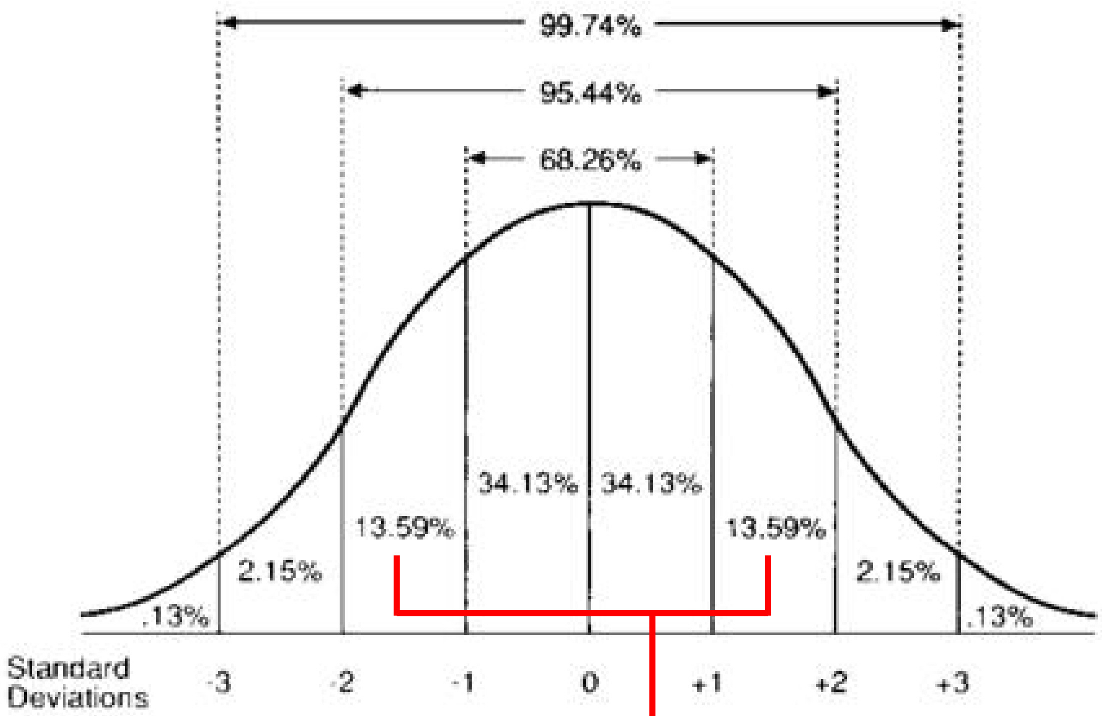
- Until recently, most computer languages only provided a **uniform** pseudo-random number generator
- Growing up I had heard of a **bell curve** and I understood the rationale for **curving** test scores
- However I could not create a **normal distribution** using my 1978 vintage TRS-80 computer using Bill Gate's first **BASIC** language interpreter

… or could I?

Normal Distribution



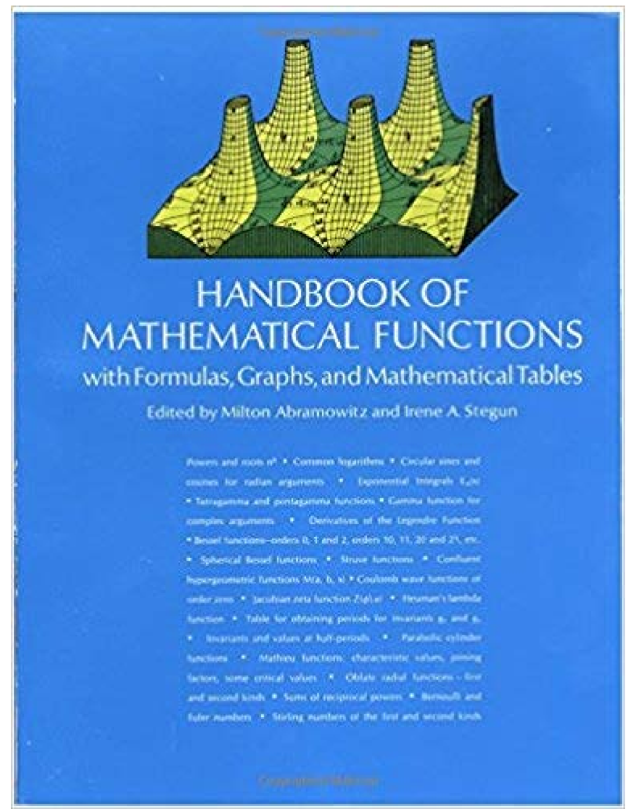
Normal Distribution



Sigma Number	Population Percent
1	68.26%
2	27.18%
3	4.30%
4	0.26%

Normal Distribution

- There are indeed several ways to turn a uniform distribution into a normal distribution
- Developing an accurate functional approximation to a normal curve requires advanced mathematics
- Consider this code from Abramowitz & Stegun's classic Handbook

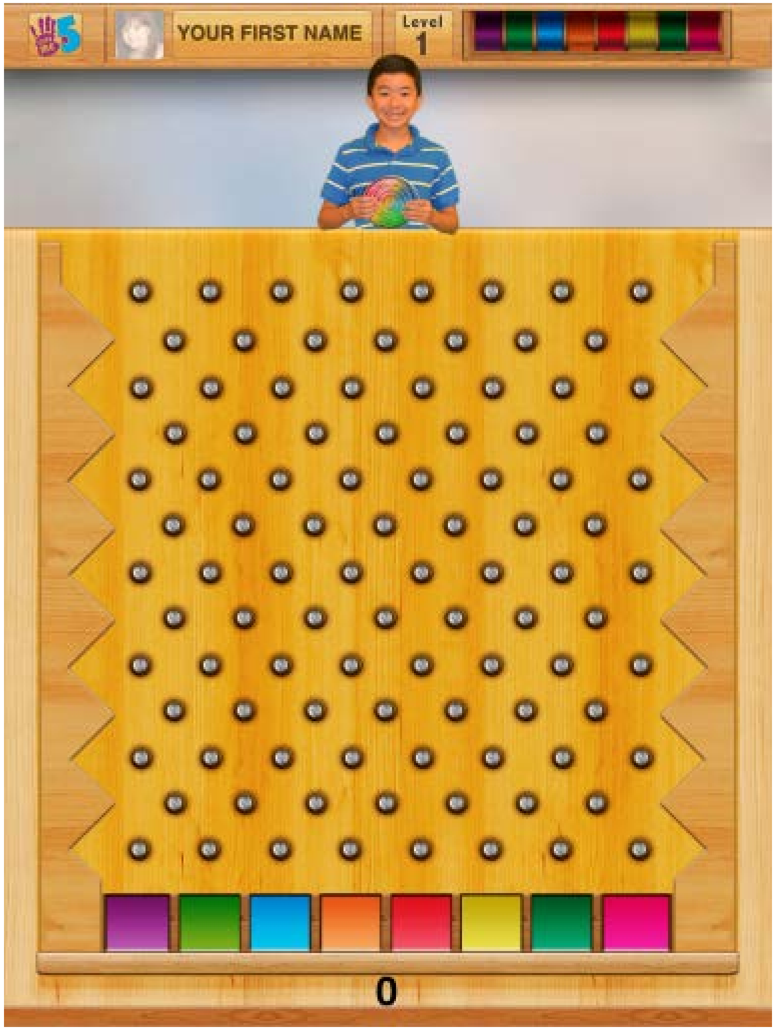


Normal Distribution

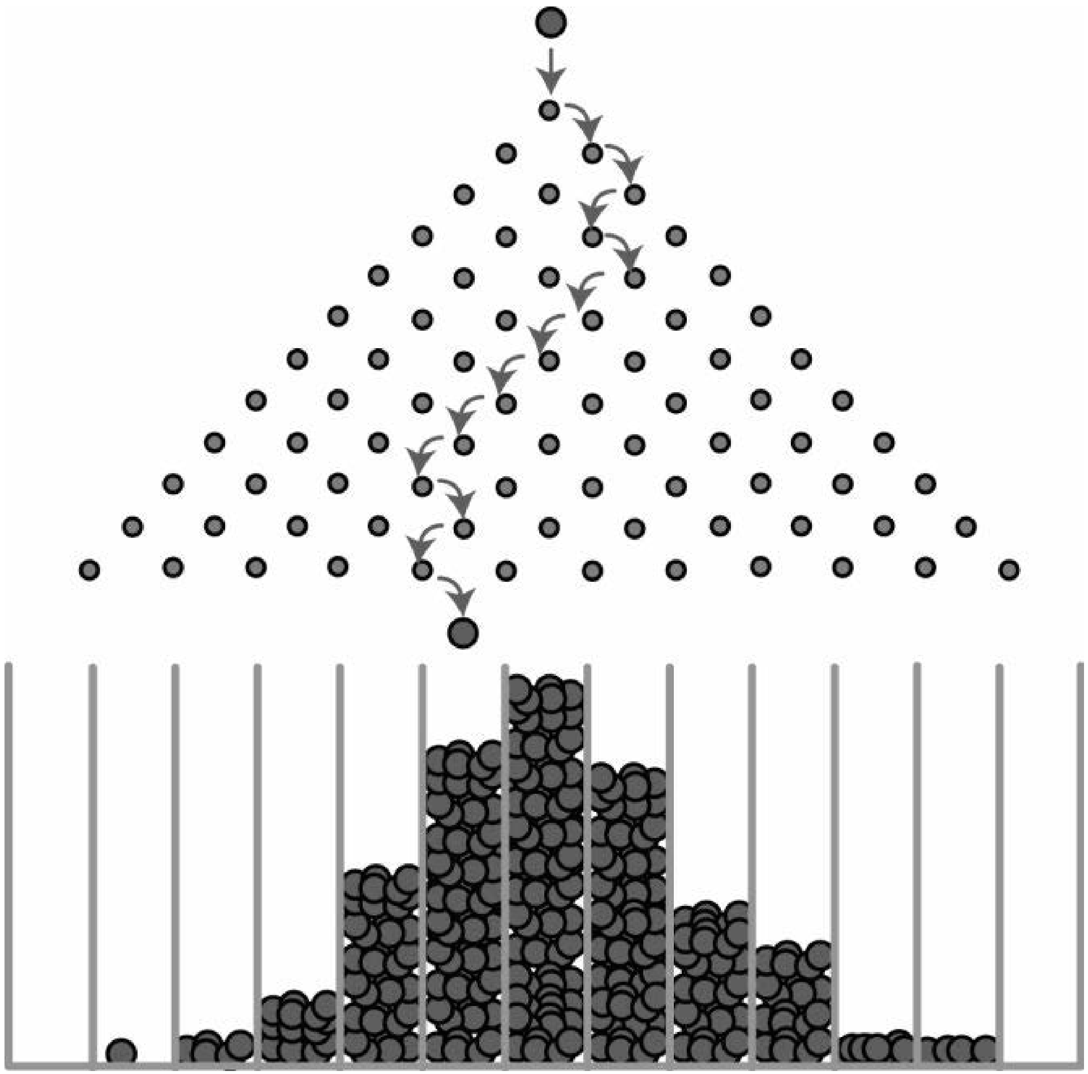
```
double StegunNormal(double mean, double stddev)
{
    double q = 1 - distUniform(generator);
    double p = (q < 0.5) ? q : 1 - q;
    double t = sqrt(log((1 / (p * p))));
    double x = t - (2.515517 + 0.802853 * t + 0.010328 * (t * t)) /
        (1 + 1.432788 * t + 0.189269 * (t * t) + 0.001308 * (t * t * t));
    x = (q < 0.5) ? x : -1 * x;
    return x * stddev + mean;
}
```

- This was neat but I did not understand it at all!
- Where did all those magic numbers come from?
- I wanted to base my approach after something tangible

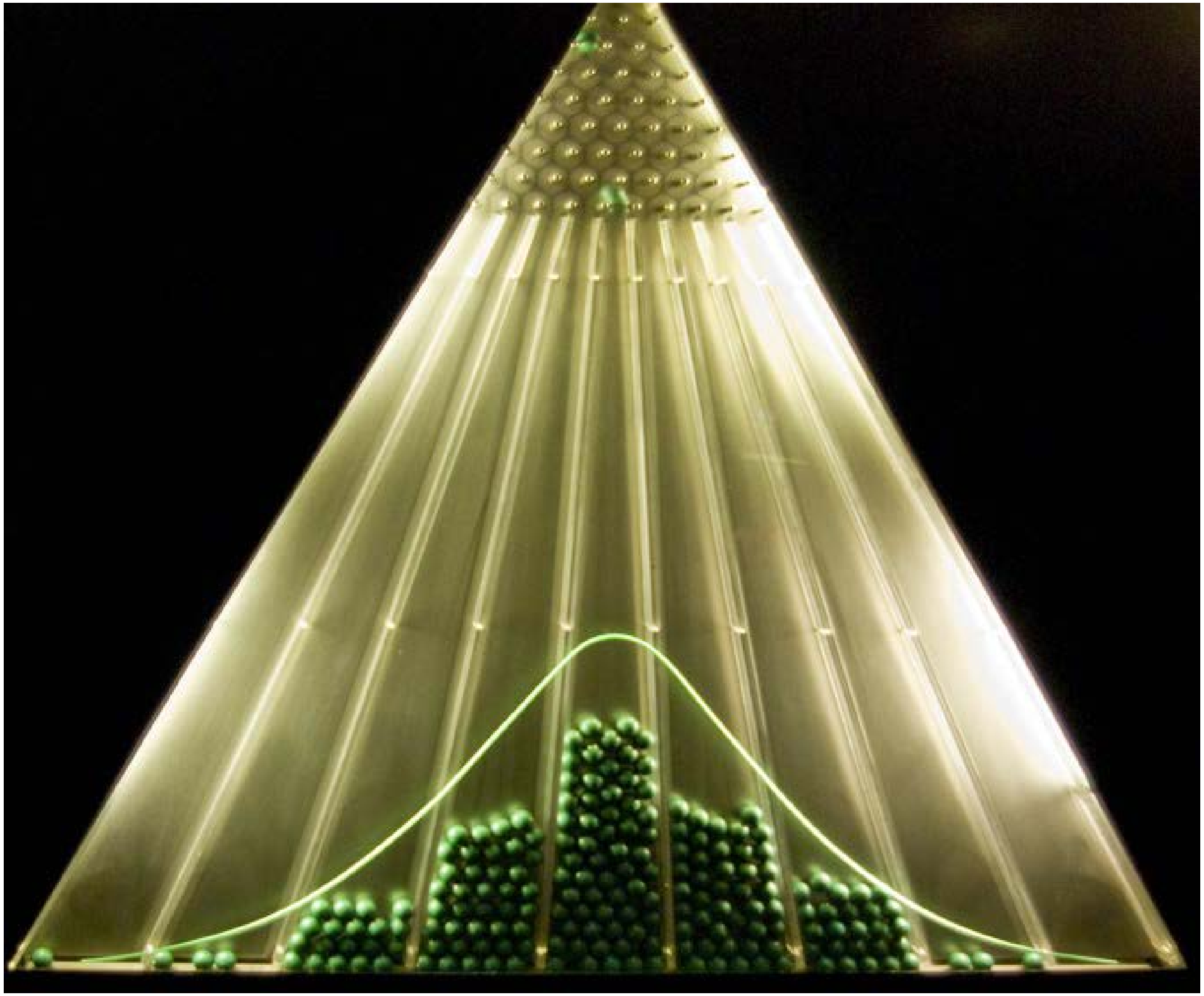
Pachinko Distribution



Pachinko Distribution



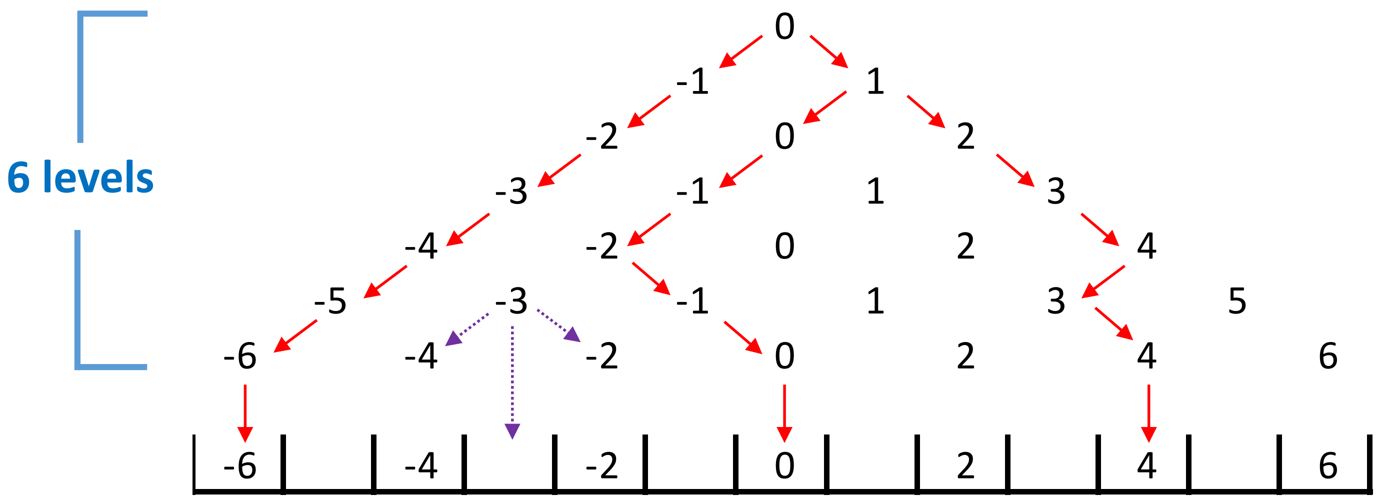
Pachinko Distribution



Normal Distribution

- We can simulate dropping balls down a Pachinko board where at each **level** a ball can move one step to left or right
- If we drop enough balls through enough levels, and we accumulate the **count of balls at each slot** at the bottom, then we should be able to simulate a normal distribution
- We will run a **chi-squared test** to see if our code simulates a distribution that has a reasonable deviation from the perfect (pure) **normal distribution**
- If the discrepancies are ***statistically significant***, then we cannot trust that our algorithm is producing a "good enough" normal distribution to use in scientific simulations

Open Lab 4 – Normal Distribution




```
int DropBall()
{
    int slot{};
    for (int level{}; level < levels; level++) {
        int step = distribution(generator);
        if (step == 0)
            slot--;
        else
            slot++;
    }
    slot = slot / 2;
    return slot;
}
```

```
seed_seq seed{ 2016 };
default_random_engine generator(seed);
uniform_int_distribution<int> distribution(0, 1);
```

The range is ½ the number of levels

View Lab 4 – Normal Distribution

```
const int balls{ 1000 };
const int levels{ 10 };

seed_seq seed{ 2016 };
default_random_engine generator(seed);
uniform_int_distribution<int> distribution(0, 1);
double mean{};
double stddev{};

const int sigmas{ 4 };
vector<int> sigCountPachinko(sigmas);
vector<int> sigCountNormal(sigmas);

double chiSquared{};
```

```
int main()
{
    CalcMeanPachinko();

    ResetPachinkoDistribution();

    CalcStdDevPachinko();

    ResetPachinkoDistribution();

    CountBallsPerSigma();

    DisplayBallsPerSigma();

    return 0;
}
```

```
void ResetPachinkoDistribution()
{
    generator.seed(seed);
    distribution.reset();
}

void CalcMeanPachinko()
{
    for (int ball{}; ball < balls; ball++)
        mean += DropBall();
    mean /= balls;
}

void CalcStdDevPachinko()
{
    double variance{};
    for (int ball{}; ball < balls; ball++)
        variance += pow(DropBall() - mean, 2);
    stddev = sqrt(variance / balls);
}
```

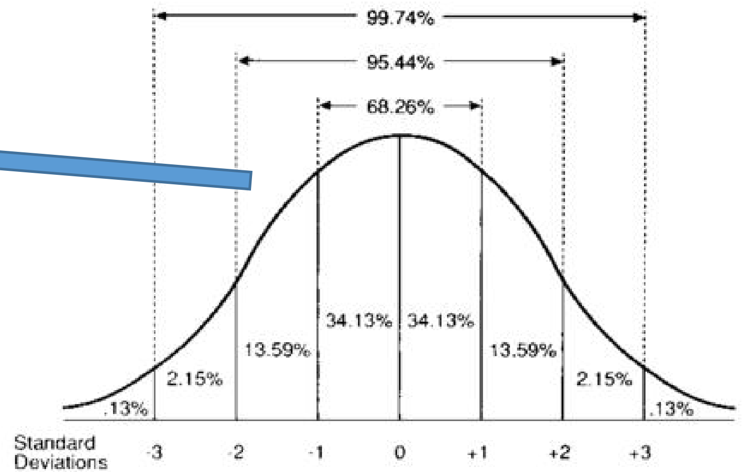
The mean slot *should be* = 0

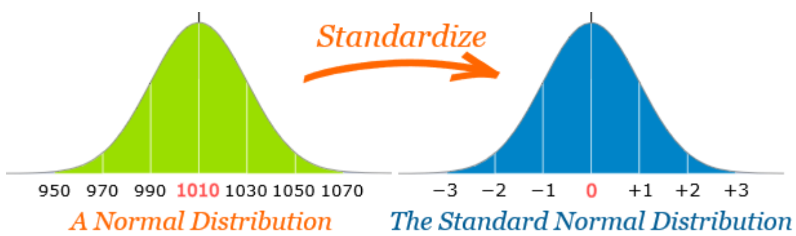
View Lab 4 – Normal Distribution

```
const int balls{ 1000 };
const int levels{ 10 };
```

```
const int sigmas{ 4 };
vector<int> sigCountPachinko(sigmas);
vector<int> sigCountNormal(sigmas);
```

```
void CountBallsPerSigma()
{
    sigCountNormal[0] = 0.6826 * balls;
    sigCountNormal[1] = 0.2718 * balls;
    sigCountNormal[2] = 0.0430 * balls;
    sigCountNormal[3] = 0.0026 * balls;

    for (int ball{}; ball < balls; ball++) {
        int slot = DropBall();
        int sigma = abs(mean + slot) / stddev;
        if (sigma < sigCountPachinko.size())
            sigCountPachinko.at(sigma)++;
    }

    for (int s{}; s < sigmas; s++)
        chiSquared += (pow(
            sigCountNormal[s] -
            sigCountPachinko[s], 2)
            / sigCountNormal.at(s));
}
```





- A **sigma** is a integral multiple of the **standard deviation**
- Each **slot** is belongs to a sigma
- We count the **number of balls** that fall into each sigma

Chi-Squared Test

- Does the Pachinko distribution perform reasonably well compared to a perfect **normal** distribution?

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

```
for (int s{}; s < sigmas; s++)
    chiSquared += (pow(
        sigCountNormal[s] -
        sigCountPachinko[s], 2)
        / sigCountNormal.at(s));
```

Run Lab 4 – Normal Distribution

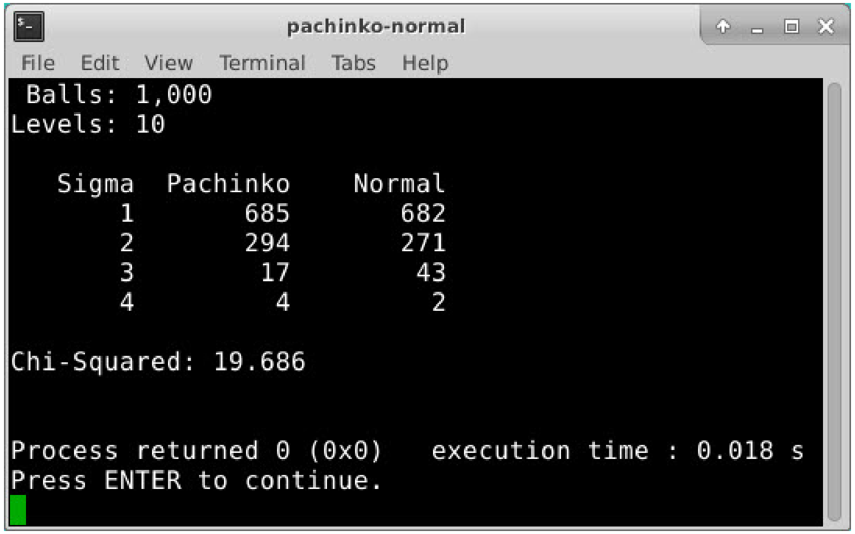
```
pachinko-normal
File  Edit  View  Terminal  Tabs  Help
 Balls: 1,000
Levels: 10

   Sigma  Pachinko     Normal
       1       685        682
       2       294        271
       3        17         43
       4         4          2

Chi-Squared: 19.686
```

For 4 degrees of freedom, the **19.686** deviation is statistically ***significant*** (> 9.49) so the proposed algorithm **is not** generating reasonable normally distributed probabilities! ☹☹☹

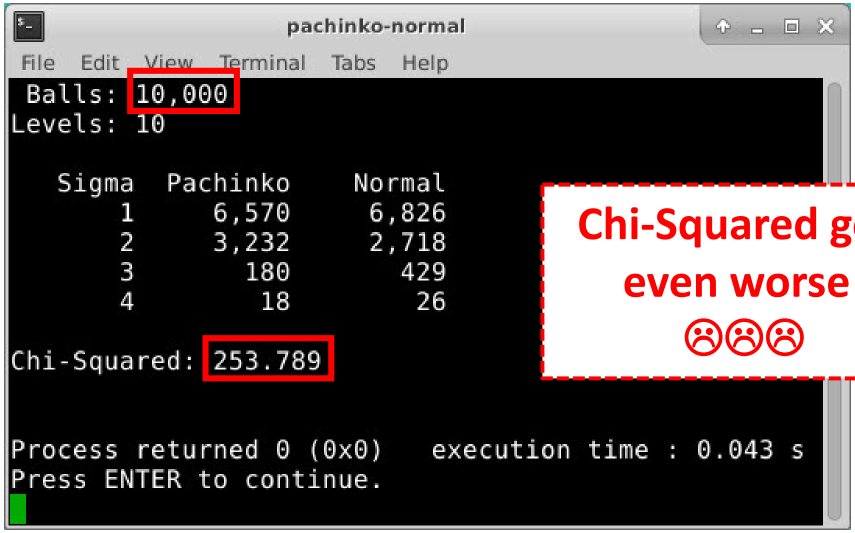
Degrees of Freedom	Probability										
	0.95	0.90	0.80	0.70	0.50	0.30	0.20	0.10	0.05	0.01	0.001
1	0.004	0.02	0.06	0.15	0.46	1.07	1.64	2.71	3.84	6.64	10.83
2	0.10	0.21	0.45	0.71	1.39	2.41	3.22	4.60	5.99	9.21	13.82
3	0.35	0.58	1.01	1.42	2.37	3.66	4.64	6.25	7.82	11.34	16.27
4	0.71	1.06	1.65	2.20	3.36	4.88	5.99	7.78	9.49	13.28	18.47
5	1.14	1.61	2.34	3.00	4.35	6.06	7.29	9.24	11.07	15.09	20.52
6	1.63	2.20	3.07	3.83	5.35	7.23	8.56	10.64	12.59	16.81	22.46
7	2.17	2.83	3.82	4.67	6.35	8.38	9.80	12.02	14.07	18.48	24.32
8	2.73	3.49	4.59	5.53	7.34	9.52	11.03	13.36	15.51	20.09	26.12
9	3.32	4.17	5.38	6.39	8.34	10.66	12.24	14.68	16.92	21.67	27.88
10	3.94	4.86	6.18	7.27	9.34	11.78	13.44	15.99	18.31	23.21	29.59
	Nonsignificant								Significant		

Approaching a Normal Distribution

Maybe we didn't let enough balls drop through to get a good estimate?



What if we used 10x more balls?



Chi-Squared gets even worse!
☹☹☹

Approaching a Normal Distribution

- So can the Pachinko model accurately mimic a normal distribution? ***Yes!*** ...but only under the right circumstances
- It turns out that **it is not just** the **number of balls** that are used in the experiment that matters, but also the **number of levels** in the simulated Pachinko board
- The **levels** affects how wide (displacement from the center slot) a ball can fall left or right from the topmost (first) pin
- We have to ensure we have enough **levels** (therefore enough **width** in the last level) to ensure the balls can **spread out** during their fall to occupy bottom slots that represent the **higher sigma values**

Approaching a Normal Distribution

Increasing the **number of levels** improves the agreement of the **sigma ball count** between the Pachinko and perfect normal distribution, **thereby decreasing the chi-squared value**, until the difference **is no longer statistically significant**

Degrees of Freedom	Probability										
	0.95	0.90	0.80	0.70	0.50	0.30	0.20	0.10	0.05	0.01	0.001
1	0.004	0.02	0.06	0.15	0.46	1.07	1.64	2.71	3.84	6.64	10.83
2	0.10	0.21	0.45	0.71	1.39	2.41	3.22	4.60	5.99	9.21	13.82
3	0.35	0.58	1.01	1.42	2.37	3.66	4.64	6.25	7.82	11.34	16.27
4	0.71	1.06	1.65	2.20	3.36	4.88	5.99	7.78	9.49	13.28	18.47
5	1.14	1.61	2.34	3.00	4.35	6.06	7.29	9.24	11.07	15.09	20.52
6	1.63	2.20	3.07	3.83	5.35	7.23	8.56	10.64	12.59	16.81	22.46
7	2.17	2.83	3.82	4.67	6.35	8.38	9.80	12.02	14.07	18.48	24.32
8	2.73	3.49	4.59	5.53	7.34	9.52	11.03	13.36	15.51	20.09	26.12
9	3.32	4.17	5.38	6.39	8.34	10.66	12.24	14.68	16.92	21.67	27.88
10	3.94	4.86	6.18	7.27	9.34	11.78	13.44	15.99	18.31	23.21	29.59
	Nonsignificant								Significant		

pachinko-normal
File Edit View Terminal Tabs Help

```
 Balls: 1,000
Levels: 10

   Sigma  Pachinko     Normal
       1       685        682
       2       294        271
       3        17         43
       4         4          2

Chi-Squared: 19.686

Process returned 0 (0x0)   exec
Press ENTER to continue.
```

pachinko-normal
File Edit View Terminal Tabs Help

```
 Balls: 1,000
Levels: 100

   Sigma  Pachinko     Normal
       1       644        682
       2       301        271
       3        52         43
       4         3          2

Chi-Squared: 7.822

Process returned 0 (0x0)   exec
Press ENTER to continue.
```

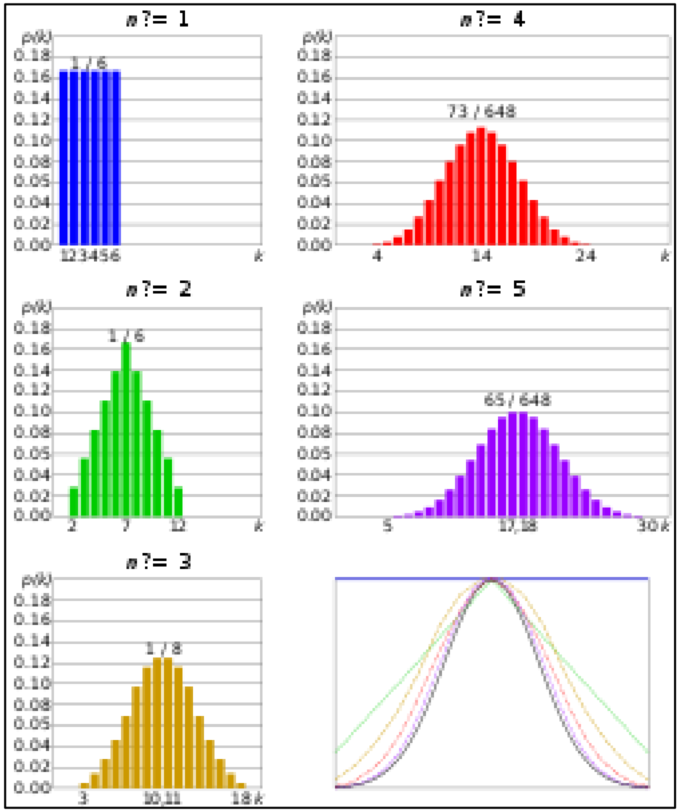
pachinko-normal
File Edit View Terminal Tabs Help

```
 Balls: 1,000
Levels: 1,000

   Sigma  Pachinko     Normal
       1       662        682
       2       295        271
       3        41         43
       4         2          2

Chi-Squared: 2.805

Process returned 0 (0x0)   exec
Press ENTER to continue.
```

Central Limit Theorem

Central limit theorem

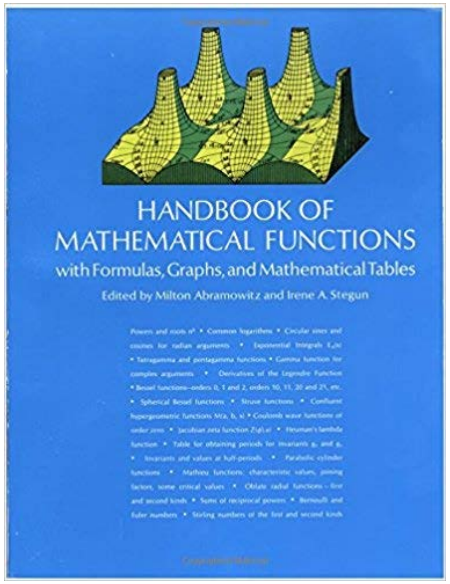
In probability theory, the **central limit theorem** (**CLT**) establishes that, for the most commonly studied scenarios, when independent random variables are added, their sum tends toward a normal distribution (commonly known as a *bell curve*) even if the original variables themselves are not normally distributed. In more precise terms, given certain conditions, the arithmetic mean of a sufficiently large number of iterates of independent random variables, each with a well-defined (finite) expected value and finite variance, will be approximately normally distributed, regardless of the underlying distribution.[1][2] The theorem is a key concept in probability theory because it implies that probabilistic and statistical methods that work for normal distributions can be applicable to many problems involving other types of distributions.



Approximating a Normal Distribution

It is best to use the code from Abramowitz & Stegun, as only **one** uniform random number call is required vs. my Pachinko method

(1000 balls x 1000 levels = **1M** iterations!)




```
double StegunNormal(double mean, double stddev)
{
    double q = 1 - distUniform(generator);
    double p = (q < 0.5) ? q : 1 - q;
    double t = sqrt(log((1 / (p * p))));
    double x = t - (2.515517 + 0.802853 * t + 0.010328 * (t * t)) /
        (1 + 1.432788 * t + 0.189269 * (t * t) + 0.001308 * (t * t * t));
    x = (q < 0.5) ? x : -1 * x;
    return x * stddev + mean;
}
```

The C++ STL has a built-in **normal_distribution<>()**

```
int main()
{
    std::random_device rd{};
    std::mt19937 gen{rd()};

    // values near the mean are the most likely
    // standard deviation affects the dispersion of generated values from the mean
    std::normal_distribution<> d{5,2};

    std::map<int, int> hist{};
    for(int n=0; n<10000; ++n) {
        ++hist[std::round(d(gen))];
    }
    for(auto p : hist) {
        std::cout << std::setw(2)
                  << p.first << ' ' << std::string(p.second/200, '*') << '\n';
    }
}
```

Now you know…

- Rational, Irrational, and Transcendental numbers each have their own style of continued fractions
  - We can take any real number and ***generate*** a CF
  - Given a CF, we can ***expand*** it to regain the original number
- The **convergents** of a CF are excellent approximations to the original number
- The magnitude of the **x** & **y** values in solutions to **Pell's Equation** $\{x^2 - \boldsymbol{n}y^2 = 1\}$ is related to the period of the *standard* continued fraction of $\sqrt{\boldsymbol{n}}$
- Memorizing thousands of digits of $\boldsymbol{\pi}$ is okay – but I'd rather appreciate its beautifully simple GCF: **[3;{(2n+1)^2 | 6}]**

Now you know…

- A perfect **Normal distribution** ensures that **68.26%** of all values fall within **one** (1) standard deviation from the **mean**
  - 99.73% of all values in a perfect normal distribution are within **three (3)** standard deviations from the mean
  - The normal distribution is known as the “**bell curve**”
- There is a way to convert a PRNG created uniform distribution into a normal distribution – but **don’t use the Pachinko method**
- The **chi-squared** test suggests if the discrepancies between the observed and the expected values are <u>statistically significant</u>