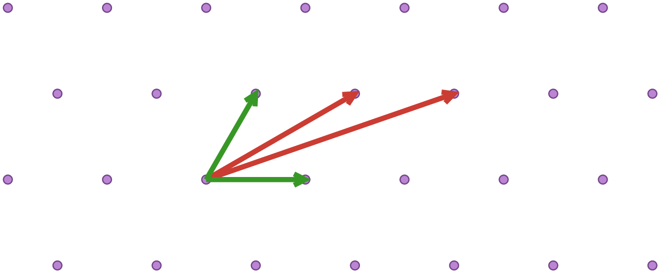
Lattices
From LLL to Quantum Hype in 15 Min

Chris Peel

Silicon Valley Ethereum Meetup



What use are lattice tools?

...other than in cryptanalysis as Joachim will describe?

Use lattice reduction and other lattice tools in places where you'd normally use linear algebra, but you want an integer-valued solution

There are practical uses:

- Post-quantum cryptography (LWE,...)
  - Encrypted ML
- Integer programming
- Digital communication
- Coding theory
- Finding anagrams :-)

And there are theoretical uses:

- Disproving Merten's Conjecture
- Sphere packing
- Diophantine equations
  - Solving $x^3 + y^3 + z^3 = d$
- Spigot algorithms

Outline

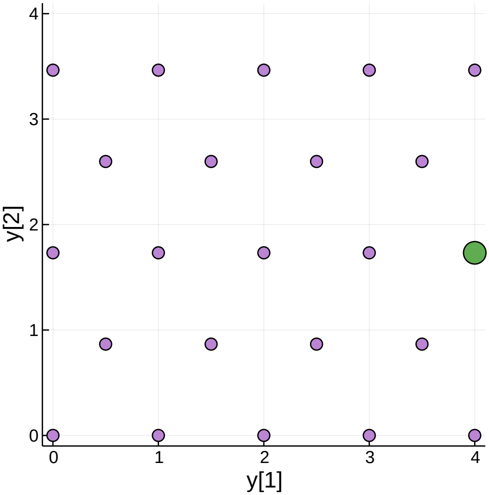
Lattice Basics: Bases

Lattice Reduction: LLL

Lattice Problems: CVP, SVP

Lattice Speculation

What is a Lattice?



A lattice is:

- In group theory lingo, a full-rank discrete additive subgroup
- The space spanned by the product of a fixed matrix and a vector of integers

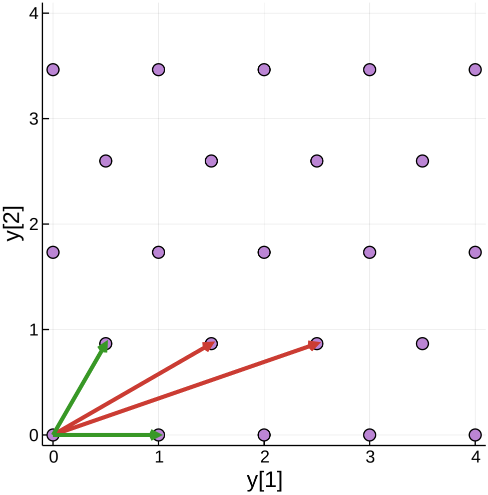
A practical definition

For a given "basis" matrix B, and a vector of integers $\mathbf{z}$, the set of points $\mathbf{y}$ reachable by $\mathbf{y} = B\mathbf{z}$ is a lattice

How did you generate the green lattice point?

```
julia> B=[2.5      1.5
         0.866025 0.866025]  # hexagonal lattice
2×2 Array{Float64,2}:
 2.5       1.5
 0.866025  0.866025
julia> zgreen=[1
               1]
2-element Array{Int64,1}:
 1
 1
julia> ygreen=B*zgreen
2-element Array{Float64,1}:
 4.0
 1.73205
julia> Pkg.add("Plots"); using Plots;
julia> plot([ygreen[1]],[ygreen[2]], markershape = :circle,
            markersize = 10,
            markercolor = RGB(0.376, 0.678, 0.318))
```

For a lattice, how many bases are possible?



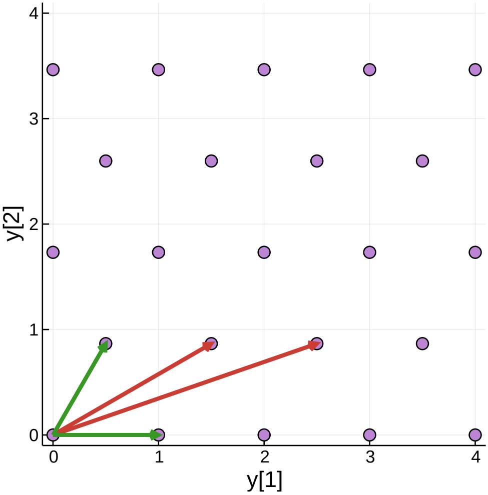
There are an infinite number of bases for a lattice; which one should we use?

$$Br = \begin{bmatrix} 2.5 & 1.5 \\ .86602 & .86602 \end{bmatrix}$$

$$Bg = \begin{bmatrix} 1.0 & .5 \\ 0.0 & .86602 \end{bmatrix}$$

In many problems, we want a short, close-to-orthogonal basis, like the green basis

How are different lattice bases related?



A **unimodular matrix** is square, integer-valued, and has determinant ± 1. Its inverse is also unimodular

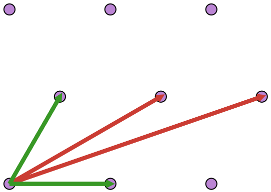
Relation between Bases

Every pair of lattice bases B_1 and B_2 are related by a unimodular matrix T: $B_1 = B_2 T$

For the bases in the figure, $B_{red} = B_{green} T$, where

$$T = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

How can we find a short basis?



Use lattice reduction

Given lattice with basis B_1, the goal of lattice reduction is to find another basis B_2 for the same lattice which has short, closer-to-orthogonal basis vectors

Often, "short" and "orthogonal" are defined according to the Euclidian norm. So $B_2^T B_2$ is closer to diagonal than $B_1^T B_1$, and the diagonal elements of $B_2^T B_2$ are smaller than those of $B_1^T B_1$

'Lattice reduction is like QR for integer problems.'
Jack Poulson

Instead of an orthonormal Q, we have a close-to-orthogonal reduced basis, and instead of a triangular R we have a unimodular matrix: $B_1 = B_2 T$

How does one do lattice reduction?

The most important lattice reduction technique is from Lenstra, Lenstra, and Lovász[1], known as the LLL algorithm

LLL in pseudocode

Input: a basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of a lattice L.
Output: the basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ is LLL-reduced with factor δ.

1: Size-reduce $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$
2: **if** there exists an index j which does not satisfy Lovász' condition
3:     swap $\mathbf{b}_j$ and $\mathbf{b}_{j+1}$, then return to Step 1.
4: **end if**

Lovász' condition is $||\mathbf{b}_{j+1}||^2 \geq (\delta - \mu_{j+1,j}^2)||\mathbf{b}_j||^2$ where the coeficients μ are Gram-Schmidt coefficients from size reduction

[1] A. K. Lenstra; H. W. Lenstra Jr.; L. Lovász; "Factoring polynomials with rational coefficients". Mathematische Annalen 261, 1982.

Size Reduction? Gram-Schmidt? Do I need to know this?

No, most LLL users can skip previous, current, next slides :-)

Size Reduction pseudocode[2]

Input: A basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$ of a lattice L.
Output: A size-reduced basis $(\mathbf{b}_1, \ldots, \mathbf{b}_d)$.

Compute all the Gram–Schmidt coefficients $\mu_{i,j}$
for $i = 2$ to d **do**
    for $j = i - 1$ downto 1 **do**
        $\mathbf{b}_i \longleftarrow \mathbf{b}_i - \lceil \mu_{i,j} \rfloor \mathbf{b}_j$
        for $k = 1$ to j **do**
            $\mu_{i,k} \longleftarrow \mu_{i,k} - \lceil \mu_{i,j} \rfloor \mu_{j,k}$
        end for
    end for
end for

Size reduction is GS with rounding

There are LLL variants which use

- Gram-Schmidt (shown)
- Givens rotations
- Householder rotations
- A Cholesky decomposition (fastest)

[2] The LLL and size reduction pseudocode are from P. Q. Nguyen "Hermite's constant and lattice algorithms," a chapter of The LLL Algorithm, Springer, Berlin, Heidelberg, 2009, pp 19-69

Givens-based LLL in Julia

```
function lll(H::Matrix{Td},δ::Float64=3/4) where {Td<:Number}
    B = copy(H);   N,L = size(B);  _,R = qr(B)
    lx  = 2
    while lx <= L
        for k=lx-1:-1:1
            rk = R[k,lx]/R[k,k]
            mu = round(rk)
            if abs(mu)>0
                B[:,lx]   -= mu * B[:,k]
                R[1:k,lx] -= mu * R[1:k,k]
            end
        end
        nrm = norm(R[lx-1:lx,lx])
        if δ*abs(R[lx-1,lx-1])^2 > nrm^2
            B[:,[lx-1,lx]]     = B[:,[lx,lx-1]]
            R[1:lx,[lx-1,lx]] = R[1:lx,[lx,lx-1]]
            cc = R[lx-1,lx-1] / nrm
            ss = R[lx,lx-1]   / nrm
            Θ = [cc' ss; -ss cc]  # Givens rotation
            R[lx-1:lx,lx-1:end] .= Θ * R[lx-1:lx,lx-1:end]
            lx = max(lx-1,2)
        else; lx = lx+1; end
    end
    return B
end
```

What should I remember about the LLL?

Remember two things:

- LLL is fast; it runs in $O(d^5)$ for bases of size d
- LLL reduces the basis[3]: $||\mathbf{b}_1|| \leq (\frac{2}{\sqrt{4\delta-1}})^{d-1}\lambda_1(\mathcal{L})$

The LLL is **the** core lattice tool. Its polynomial speed and acceptable reduction quality is what brought interest to lattice tools

[3] For a lattice $\mathcal{L}$, the length of the shortest vector is $\lambda_1(\mathcal{L})$

Outline

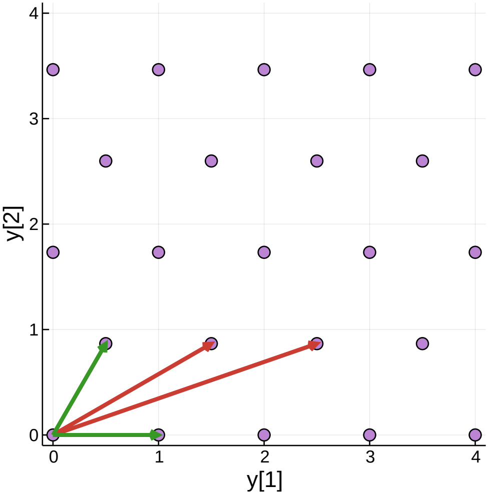
Lattice Basics: Bases

Lattice Reduction: LLL

Lattice Problems: CVP, SVP

Lattice Speculation

Shortest Vector Problem: $\arg\min_{\mathbf{b}\in\mathcal{L},\mathbf{x}\neq\mathbf{0}} ||\mathbf{b}||$



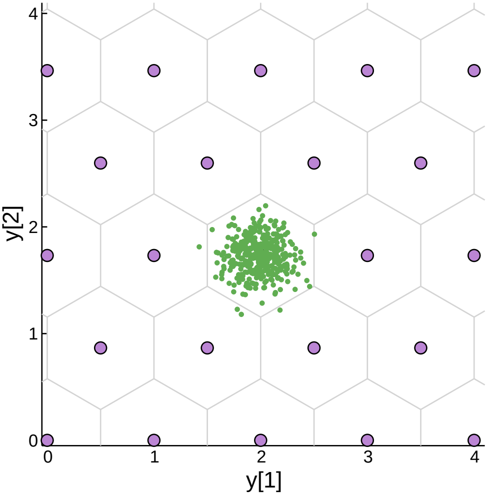
SVP

Find one of the shortest non-zero vectors in the lattice

Solving SVP is so **HARD**, that it is used for post-quantum cryptography

LLL can be used to approximate SVP in polynomial time

Closest Vector Problem: $\arg\min_{\mathbf{x}\in\mathbb{Z}^n} ||B\mathbf{x} - \mathbf{y}||$



CVP

Find the closest point in the lattice to a given vector, which is usually not a lattice point

Again, solving CVP is so **HARD**, that it is used for post-quantum cryptography

(ok, ok, it's not quite CVP or SVP, rather closely related problems that are used for post-quantum crypto)

Outline

Lattice Basics: Bases

Lattice Reduction: LLL

Lattice Problems: CVP, SVP

Lattice Speculation

Could we make a Lattice Blockchain?

Yes. After lattice cryptography is more mature (years from now) we could make a “lattice blockchain”

What could it do?

- Generate hype! Marketing galore!
- Yes, it would hopefully be quantum resistant
- Woah, resistant to quantum attack, that’s great for marketing!
- Fully homomorphic encryption using on-chain keys and functions. Program obfuscation, identity-based encryption, attribute-based encryption, functional encryption

I believe it will take years for non-hype uses to mature

That's all for this quick tour of lattices

Where can I learn more?

- Look at the documents linked from the "What use are lattice tools?" slide earlier in this deck
- Watch videos from a January 2020 Simons Institute worksop
- Read Wikipedia on lattice reduction, lattice problems (CVP, SVP), and on lattice cryptography

What lattice software tools are available?

- `fplll` is an open-source, fast, full-featured C++-based lattice library
- The Number Theory Library has a C++-based LLL
- The Julia package `LLLplus.jl` can handle complex bases, and is easily accessible at Julia's REPL