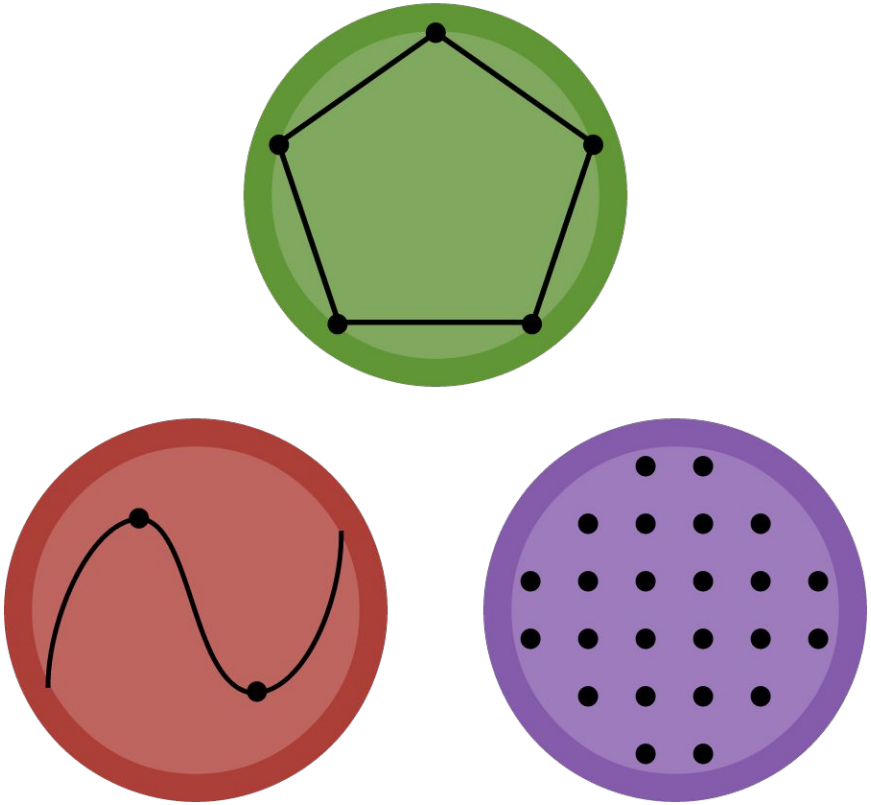


JuliaOpt

Optimization packages in Julia

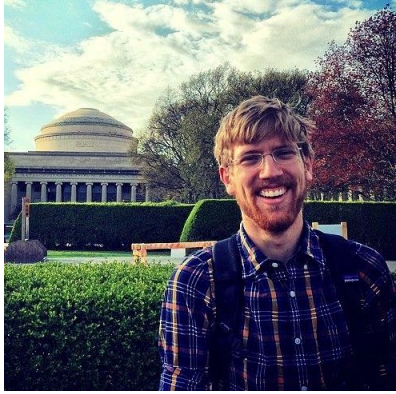
Iain Dunning
Operations Research Center, MIT

http://iaindunning.com
@iaindunning
github.com/IainNZ

7th Annual Scientific Software Days Conference, Austin, Texas, Feb 25-26, 2016

Overview

1. What is **Julia**?
2. What is **optimization**?
3. What is **JuliaOpt**?
4. Modeling with **JuMP**
5. Modeling with **Convex.jl**
6. **Interfaces** - MathProgBase.jl
7. JuliaOpt as an **organization**



Iain Dunning
Fifth (final!) year PhD

OPERATIONS
RESEARCH
CENTER

What is Julia?

“high-level, high-performance dynamic programming language for technical computing, with syntax that is familiar to users of other technical computing environments”

- LLVM JIT, types, multiple dispatch,
- Macros/metaprogramming (Lisp-ish)
- Built-in package manager, plays well with other languages, free/open source (MIT)

```
a_norm(x) = sqrt(sum(x.*x))
```

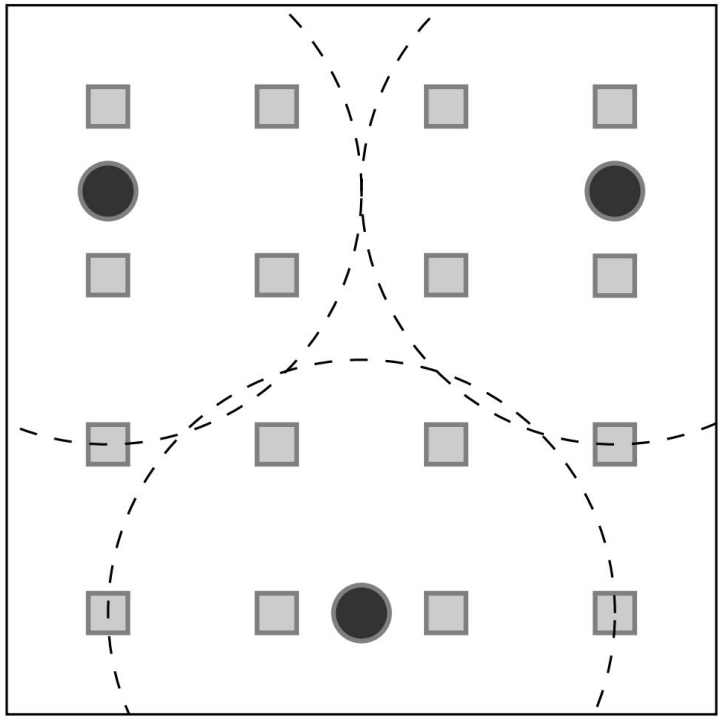
```
function my_norm{T<:Number}(x::Vector{T})
    n = zero(T)
    for i in eachindex(x)
        n += x[i]^2
    end
    return sqrt(n)
end
@show my_norm([-1,0,+1])  # 1.414..

my_map(f, x) = [f(i) for i in x]
@show my_map(abs, -1:+1)  # [1,0,1]
```

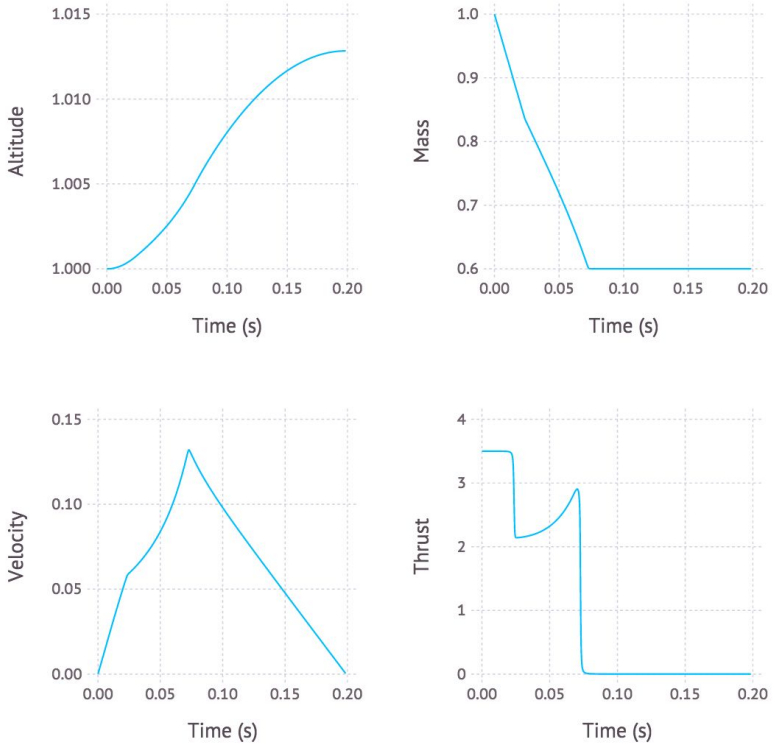
What is Optimization?

$$\min_{x} \quad f(x)$$

$$\text{subject to} \quad g_i(x) \leq 0 \quad \forall i$$

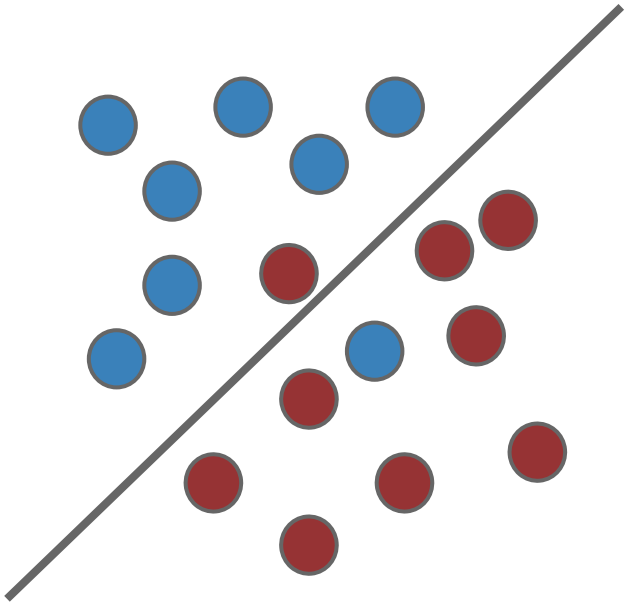
What is Optimization?

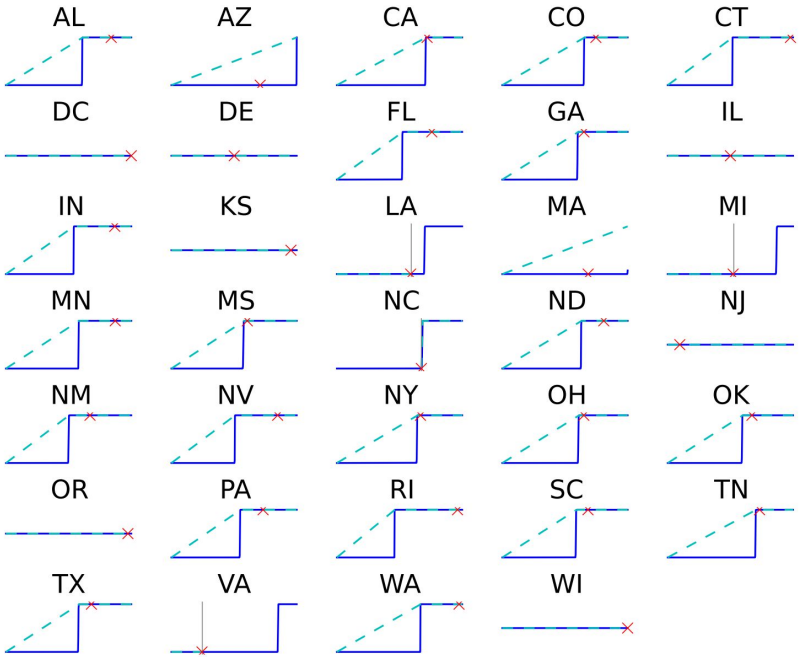




I. Dunning, J. Huchette, and M. Lubin. "JuMP: A modeling language for mathematical optimization."

What is Optimization?

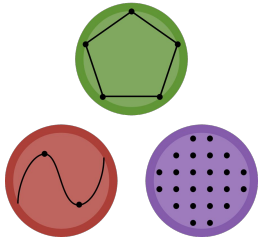




M. Udell, and S. Boyd, “Maximizing a Sum of Sigmoids”

What is JuliaOpt? Packages for...

- **Modeling**: express optimization problems with Julia code
- **Solving**: pure Julia routines, and wrappers for external solvers
- **Abstracting**: the “glue” between modeling, solving, and user code



JuliaOpt Packages

JuMP

Convex.jl

MathProgBase.jl

Cbc.jl

Clp.jl

CPLEX.jl

ECOS.jl

GLPK.jl

Gurobi.jl

Ipopt.jl

KNITRO.jl

Mosek.jl

NLopt.jl

SCS.jl

Optim.jl

LsqFit.jl

CoinOptServices.jl

AmplNLWriter.jl

JuliaOpt as an Organization

- **Standards** for packages:

  binaries, documentation, tests, integration
- Centralized **info** about optimization in Julia

  juliaopt.org julia-opt mailing list
- Including 

notebooks!

Modeling problems with JuMP

$$\min_x \sum_{(i,j)\in E} c_{i,j}x_{i,j}$$

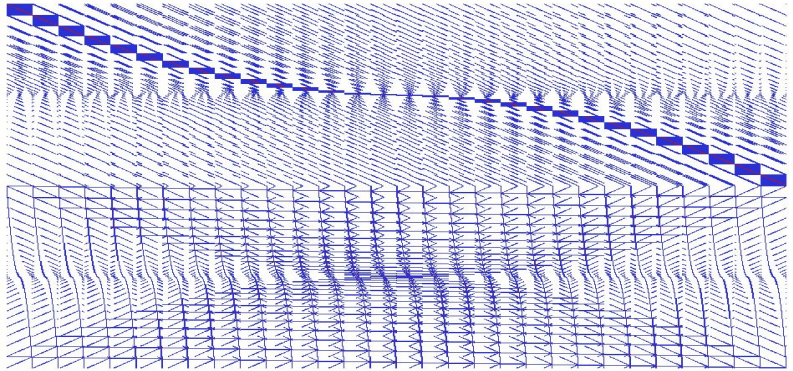
$$\text{s.t.} \sum_{(i,j)\in E} x_{i,j} = \sum_{(j,k)\in E} x_{j,k} \quad \forall j \in V \setminus \{1,n\}$$

$$\sum_{(i,n)\in E} x_{i,n} = 1$$

$$0 \le x_{i,j} \le C_{i,j} \quad \forall (i,j) \in E$$

We have...

But need...

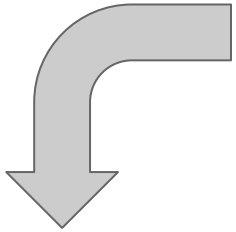


$$\min_x \sum_{(i,j)\in E} c_{i,j}x_{i,j}$$

$$\text{s.t.} \sum_{(i,j)\in E} x_{i,j} = \sum_{(j,k)\in E} x_{j,k} \quad \forall j \in V \setminus \{1,n\}$$

$$\sum_{(i,n)\in E} x_{i,n} = 1$$

$$0 \le x_{i,j} \le C_{i,j} \quad \forall (i,j) \in E$$



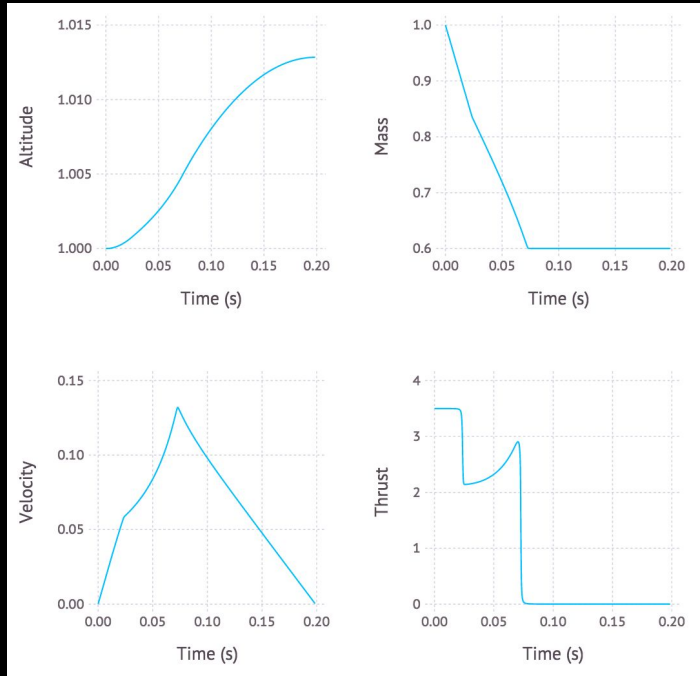
```
immutable Edge
    from; to; cost; capacity
end
edges = [Edge(1,2,1,0.5), Edge(1,3,2,0.4), Edge(1,4,3,0.6),
         Edge(2,5,2,0.3), Edge(3,5,2,0.6), Edge(4,5,2,0.5)]
mcf = Model()
@defVar(mcf, 0 <= flow[e in edges] <= e.capacity)
@addConstraint(mcf, sum{flow[e], e in edges; e.to==5} == 1)
@addConstraint(mcf, flowcon[n=2:4], sum{flow[e], e in edges; e.to==node}
                                 == sum{flow[e], e in edges; e.from==node})
@setObjective(mcf, Min, sum{e.cost * flow[e], e in edges})
```

Julia

JuMP

```julia
using JuMP, Ipopt
mod = Model(solver=IpoptSolver(print_level=0))
@defVar(mod, Δt ≥ 0, start = 1/n)   # Time step
@defNLExpr(t_f, Δt*n)               # Time of flight
# State variables
@defVar(mod, v[0:n] ≥ 0)            # Velocity
@defVar(mod, h[0:n] ≥ h_0)          # Height
@defVar(mod, m_f ≤ m[0:n] ≤ m_0)    # Mass
# Control: thrust
@defVar(mod, 0 ≤ T[0:n] ≤ T_max)
# Objective: maximize altitude at end of time of flight
@setObjective(mod, Max, h[n])
# Forces - drag and gravity
@defNLExpr(mod, drag[j=0:n], D_c*(v[j]^2)*exp(-h_c*(h[j]-h_0)/h_0))
@defNLExpr(mod, grav[j=0:n], g_0*(h_0/h[j])^2)
# Dynamics
for j in 1:n
    # h' = v
    @addNLConstraint(mod, h[j] == h[j-1] + Δt*v[j-1])
    # v' = (T-D(h,v))/m - g(h)
    @addNLConstraint(mod, v[j] == v[j-1] + Δt*(
                        (T[j-1] - drag[j-1])/m[j-1] - grav[j-1]))
    # m' = -T/c
    @addNLConstraint(mod, m[j] == m[j-1] - Δt*T[j-1]/c)
end
```

Reverse-then-forward mode automatic differentiation to codegen derivative-evaluating functions (sparse Hessian), as callbacks that can be used with IPOPT, KNITRO, ...



Modeling with Convex.jl

"Given **polyhedron** $C = \{x \mid a_i^T x \leq b_i,\ i = 1, \ldots, m\}$

find **ellipsoid** $\mathcal{E} = \{Bu + d \mid \|u\|_2 \leq 1\}$ that lies in the

interior of *C* with maximum volume"

$$\begin{array}{ll} \text{maximize} & \log\det B \\ \text{subject to} & \sup_{\|u\|_2 \leq 1} I_C(Bu + d) \leq 0 \end{array}$$

Convex.jl analyzes convexity, reduces to "conic"

Max Volume Inscribed Ellipsoid

$$\begin{array}{ll} \text{maximize} & \log\det B \\ \text{subject to} & \|Ba_i\|_2 + a_i^T d \leq b_i, \quad i = 1, \ldots, m. \end{array}$$

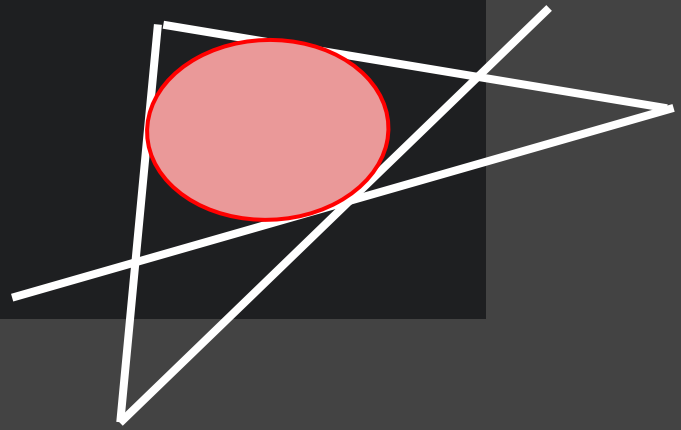
Is that objective function concave?

→ **B** is **positive definite** matrix...

→ **det(B)** = product of eigenvalues of B = **+ve**...

→ **log** of positive *x* = **concave!**

```
using Convex
a = { [ 2, 1], [ 2,-1], [-1, 2], [-1,-2] }
B = Variable(2,2)
d = Variable(2)
p = maximize(logdet(B))
for i in 1:4
  p.constraints += norm(B*a[i]) + dot(a[i],d) <= 1
end
solve!(p)
println(B.value)
println(d.value)
```



MathProgBase.jl

- Standard interface for optimization in Julia
- Crucial to the success of JuliaOpt

Linear Programming	Mixed-integer Programming	Quadratic Programming
Conic Programming	MIP Callbacks	Nonlinear Programming

MathProgBase.jl Design & Benefits

- “Don’t create interface unless you have at least two use cases”
  - Callbacks: “many states, one callback” vs “many states, many callbacks”
  - SDP interface becomes conic interface
- Multiplier effect for shared interface
  - JuMP initial consumer, now also Convex.jl
  - Each added solver benefits all

JuliaOpt + MPB for new solvers

- e.g. “mixed-integer disciplined convex programming”
  - *M. Lubin, E. Yamangil, R. Bent, J.P. Vielma, Extended Formulations in Mixed-integer Convex Programming*
- User code → Convex.jl → MathProgBase.jl → New MIDCP solver → MathProgBase.jl → Any* MILP Solver

JuMP Extensions

JuMPeR - Robust Optimization https://github.com/IainNZ/JuMPeR.jl

JuMPChance - Chance Constraints https://github.com/mlubin/JuMPChance.jl

StochJuMP - Stochastic Optimization https://github.com/joehuchette/StochJuMP.jl

Education, Academia & Industry

- JuliaOpt used for teaching around the world
- And research! For example,
  - Vielma, et al. "Extended Formulations in Mixed Integer Conic QP"
  - Gupta, Tobin, Pavel, “LP Makes Railway Networks Energy-efficient”
  - Gorhan, Mackey. “Measuring Sample Quality with Stein’s Method”
  - Giordano, Broderick, Jordan. “Robust Inference with Variational Bayes”
  - Bertsimas, de Ruiter, “Duality in two-stage adaptive linear optimization: Faster computation and stronger bounds”
- Several **companies** using for “real work” too
- See http://juliaopt.org for latest info, email us!

Thanks to our contributors!


blakejohnson
Blake Johnson


carlobaldassi
Carlo Baldassi


cmcbride
Cameron McBride


davidlizeng
David Zeng


lainNZ
Iain Dunning


JackDunnNZ
Jack Dunn


tkelman
Tony Kelman


jennyhong
Jenny Hong


jfsantos
João Felipe Santos


joehuchette
Joey Huchette


johnmyleswhite
John Myles White


karanveerm
Karanveer Mohan


lindahua
Dahua Lin


ulfworsoe
Ulf Worsøe


madeleineudell
Madeleine Udell


mlubin
Miles Lubin


pjabardo


pkofod
Patrick Kofod Mogensen


stevengj
Steven G. Johnson


timholy
Tim Holy


yeesian
Yeesian Ng