Effective Programming Practices for Economists

Numerical Optimization

Choosing optimization algorithms

Janoś Gabler and Hans-Martin von Gaudecker

Steps for choosing an algorithm

1. Theory (intro here)
2. Experimentation (histories video)
3. Refine until convergence

Relevant problem properties

- **Smoothness**: Differentiable? Kinks? Discontinuities? Stochastic?
- **Convexity**: Are there local optima?
- **Goal**: If not convex, do you need a global solution? How precise does a solution need to be?
- **Size**: 2 parameters? 10? 100? 1,000? Millions? Billions?
- **Constraints**: Bounds? Linear constraints? Nonlinear constraints?
- **Structure**: Nonlinear least-squares, Log-likelihood function

→ Properties guide selection but experimentation is important

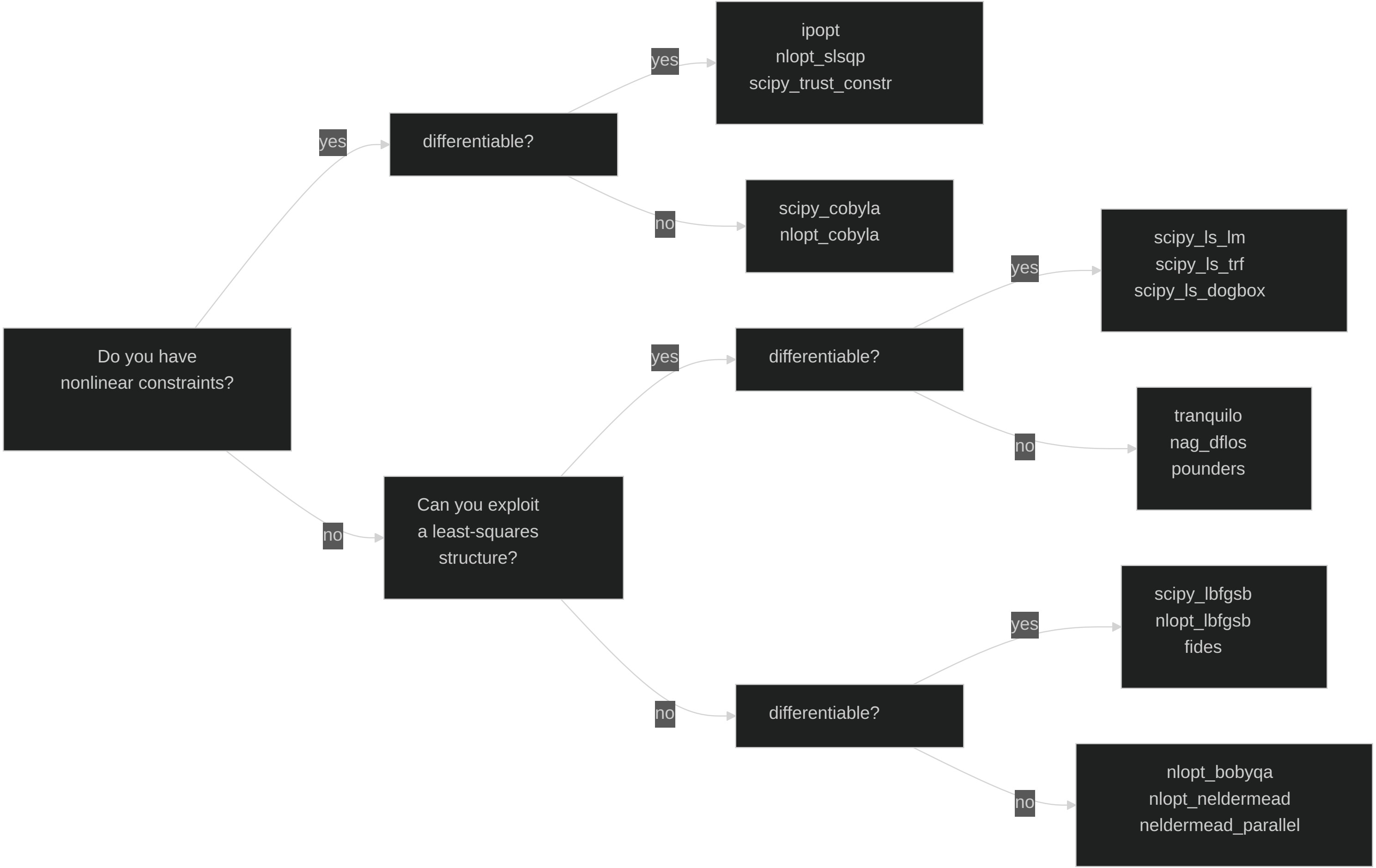
→ Always compare multiple algorithms in a criterion plot

Try to make your problem simpler

- Get derivatives using automatic differentiation (JAX, pytorch)
- Make your function faster
- Make your function more stable
- Make your function smooth

Choosing local optimizers

Next slide has a practical guide, see optimagic docs for more details.



Do you have nonlinear constraints?

yes

differentiable?

yes

ipopt
nlopt_slsqp
scipy_trust_constr

no

scipy_cobyla
nlopt_cobyla

no

Can you exploit a least-squares structure?

yes

differentiable?

yes

scipy_ls_lm
scipy_ls_trf
scipy_ls_dogbox

no

tranquilo
nag_dflos
pounders

no

differentiable?

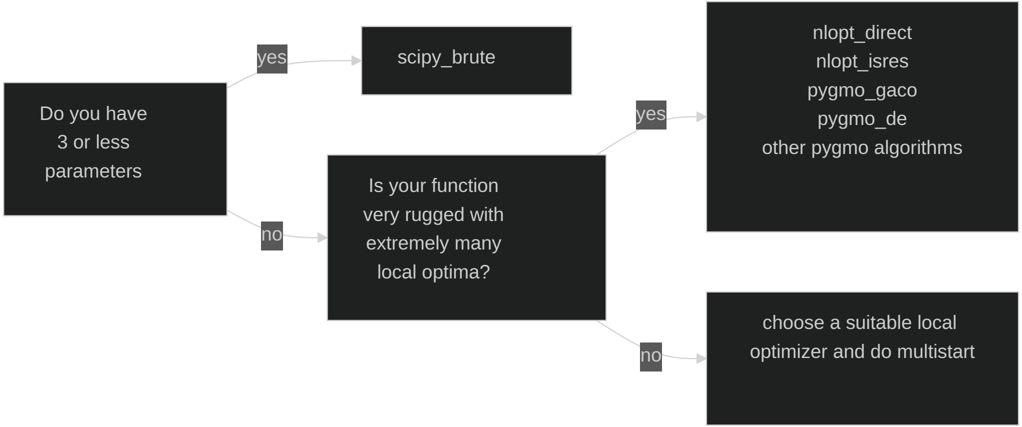
yes

scipy_lbfgsb
nlopt_lbfgsb
fides

no

nlopt_bobyqa
nlopt_neldermead
neldermead_parallel

Choosing a global approach



Do you have 3 or less parameters

yes

scipy_brute

no

Is your function very rugged with extremely many local optima?

yes

nlopt_direct
nlopt_isres
pygmo_gaco
pygmo_de
other pygmo algorithms

no

choose a suitable local optimizer and do multistart

Always refine the result of a global optimizer with a local one