

Disciplined Nonlinear Programming

Stephen Boyd Maximilian Schaller Daniel Cederberg

Stanford University

Outline

Nonlinear programming

Disciplined nonlinear programming

DNLP examples

Conclusions

Nonlinear programming (NLP)

$$\begin{array}{ll}\text{minimize} & f(x) \\ \text{subject to} & c(x) = 0 \\ & \ell \leq x \leq u\end{array}$$

- ▶ objective function $f : \mathbf{R}^n \rightarrow \mathbf{R}$ and constraint function $c : \mathbf{R}^n \rightarrow \mathbf{R}^m$
- ▶ the problem is convex if f is convex and c is affine, *i.e.*, $c(x) = Ax + b$
- ▶ in NLP, we assume f and c are differentiable, but not necessarily convex or affine

Modeling languages and solvers for NLP

- ▶ NLP solvers:
 - open source: Ipopt, Uno
 - commercial: Knitro, Gurobi, fmincon
- ▶ NLP modeling languages:
 - open source: Pyomo & CVXPY (Python), JuMP (Julia), YALMIP (MATLAB), CasADi (C++)
 - commercial: AMPL, GAMS, AIMMS

Modeling languages and solvers for NLP

- ▶ NLP solvers:
 - open source: Ipopt, Uno
 - commercial: Knitro, Gurobi, fmincon
- ▶ NLP modeling languages:
 - open source: Pyomo & CVXPY (Python), JuMP (Julia), YALMIP (MATLAB), CasADi (C++)
 - commercial: AMPL, GAMS, AIMMS
- ▶ main task of an NLP modeling language: supply the solver with derivatives
- ▶ CVXPY also transforms the problem to a solver-friendly form

Modeling languages and solvers for NLP

- ▶ NLP solvers:
 - open source: Ipopt, Uno
 - commercial: Knitro, Gurobi, fmincon
- ▶ NLP modeling languages:
 - open source: Pyomo & CVXPY (Python), JuMP (Julia), YALMIP (MATLAB), CasADi (C++)
 - commercial: AMPL, GAMS, AIMMS
- ▶ main task of an NLP modeling language: supply the solver with derivatives
- ▶ CVXPY also transforms the problem to a solver-friendly form
- ▶ we recommend CVXPY with Ipopt

Common misconceptions

NLP solvers are **not** guaranteed to

- ▶ compute a global minimizer
- ▶ compute a local minimizer
- ▶ compute a feasible point at all

(counterexamples are given below)

Common misconceptions

NLP solvers are **not** guaranteed to

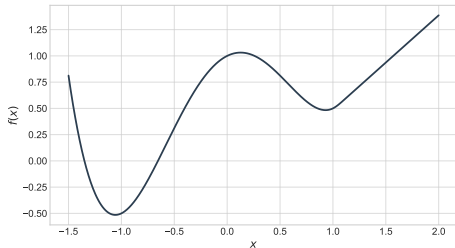
- ▶ compute a global minimizer
- ▶ compute a local minimizer
- ▶ compute a feasible point at all

(counterexamples are given below)

but in practice, NLP solvers often work well (perhaps with decent initialization)

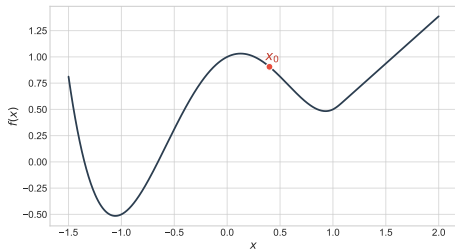
Misconception 1 about NLP

NLP solvers always compute global minimizers



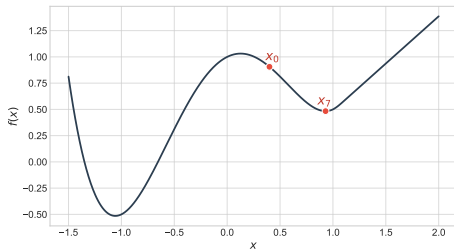
Misconception 1 about NLP

NLP solvers always compute global minimizers



Misconception 1 about NLP

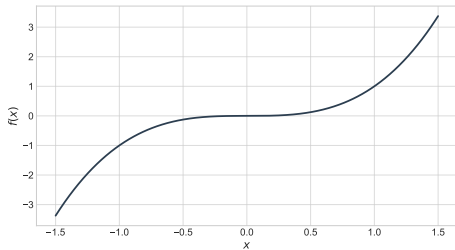
NLP solvers always compute global minimizers



- Ipopt converges to a local (but not global) minimizer

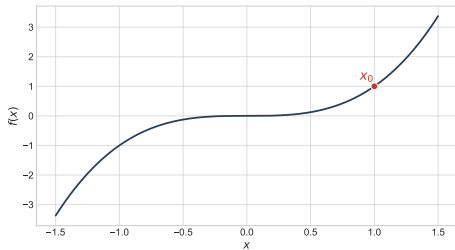
Misconception 2 about NLP

NLP solvers always converge to a local minimizer



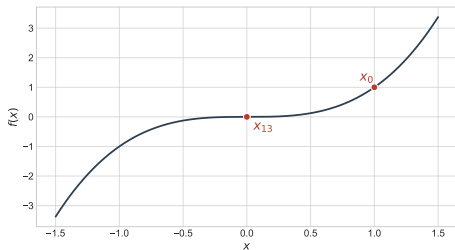
Misconception 2 about NLP

NLP solvers always converge to a local minimizer



Misconception 2 about NLP

NLP solvers always converge to a local minimizer



- Ipopt converges to a saddle point when minimizing $f(x) = x^3$ initialized at $x_0 = 1$

Misconception 3 about NLP

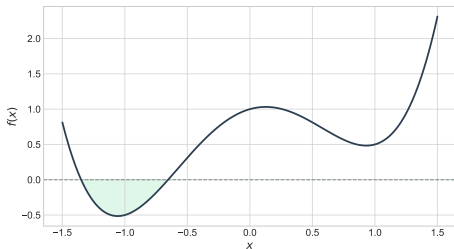
NLP solvers always converge to a 'good' feasible point (KKT point)

Misconception 3 about NLP

$$\begin{array}{ll}\text{minimize} & f(x) = x^4 - 2x^2 + 0.5x + 1 \\ \text{subject to} & f(x) \leq 0\end{array}$$

(Also see Wächter & Bowly, Gurobi, 2026)

NLP solvers always converge to a 'good' feasible point (KKT point)

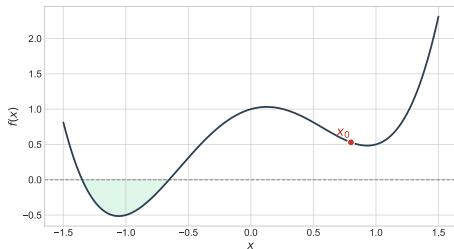


Misconception 3 about NLP

$$\begin{array}{ll}\text{minimize} & f(x) = x^4 - 2x^2 + 0.5x + 1 \\ \text{subject to} & f(x) \leq 0\end{array}$$

(Also see Wächter & Bowly, Gurobi, 2026)

NLP solvers always converge to a 'good' feasible point (KKT point)

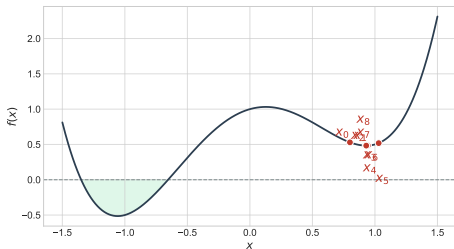


Misconception 3 about NLP

$$\begin{array}{ll}\text{minimize} & f(x) = x^4 - 2x^2 + 0.5x + 1 \\ \text{subject to} & f(x) \leq 0\end{array}$$

(Also see Wächter & Bowly, Gurobi, 2026)

NLP solvers always converge to a 'good' feasible point (KKT point)



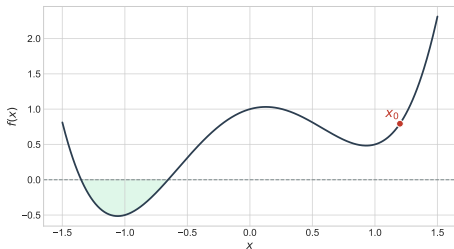
- Ipopt converges to an infeasible point when initialized at $x_0 = 0.8$

Misconception 3 about NLP

$$\begin{array}{ll}\text{minimize} & f(x) = x^4 - 2x^2 + 0.5x + 1 \\ \text{subject to} & f(x) \leq 0\end{array}$$

(Also see Wächter & Bowly, Gurobi, 2026)

NLP solvers always converge to a 'good' feasible point (KKT point)

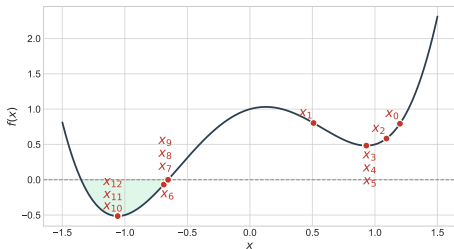


Misconception 3 about NLP

$$\begin{array}{ll}\text{minimize} & f(x) = x^4 - 2x^2 + 0.5x + 1 \\ \text{subject to} & f(x) \leq 0\end{array}$$

(Also see Wächter & Bowly, Gurobi, 2026)

NLP solvers always converge to a 'good' feasible point (KKT point)



- Ipopt converges to global minimizer when initialized at $x_0 = 1.2$

When many NLP modeling languages do a bad job

$$\text{minimize } \|Ax - b\|_2^2 + \lambda \|x\|_1$$

- ▶ variable x , regularization parameter $\lambda > 0$
- ▶ convex problem (lasso) so should be solved gracefully
- ▶ the solution occurs at a point of nondifferentiability
- ▶ objective is treated as a black box and “derivatives” are supplied to the solver
- ▶ ignoring nondifferentiability might cause solvers to struggle

When many NLP modeling languages do a bad job

output from Ipopt when interfacing it using Casadi:

```
This is Ipopt version 3.14.11, running with linear solver MUMPS 5.4.1.
```

```
...
```

iter	objective	inf_pr	inf_du	lg(mu)	d	lg(rg)	alpha_du	alpha_pr	ls
10	1.1126107e+01	0.00e+00	1.71e+00	-1.7	1.29e+08	-7.8	1.00e+00	1.32e-23f	77
11	1.3152054e+10	0.00e+00	1.71e+00	-1.7	4.09e+08	-8.3	1.00e+00	1.00e+00w	1
12	3.5423855e+10	0.00e+00	1.66e+00	-1.7	1.13e+09	-8.8	1.00e+00	1.00e+00w	1
13	1.4112259e+11	0.00e+00	1.58e+00	-1.7	3.39e+09	-9.2	1.00e+00	1.00e+00w	1

```
Cannot call restoration phase at point that is almost feasible (violation 0.0).  
Abort in line search due to no other fall back.
```

```
Number of Iterations.....: 13
```

```
EXIT: Error in step computation!
```

Outline

Nonlinear programming

Disciplined nonlinear programming

DNLP examples

Conclusions

Disciplined nonlinear programming (DNLP)

- ▶ a grammar for specifying nonlinear programs, similar to DCP
- ▶ smooth functions can be freely mixed with nonsmooth functions
- ▶ rules governing how nonsmooth functions can appear
- ▶ smooth = twice continuously differentiable in the interior of their domain

Disciplined nonlinear programming (DNLP)

- ▶ a grammar for specifying nonlinear programs, similar to DCP
- ▶ smooth functions can be freely mixed with nonsmooth functions
- ▶ rules governing how nonsmooth functions can appear
- ▶ smooth = twice continuously differentiable in the interior of their domain

a DNLP-compliant problem can be transformed to a form that

- ▶ only involves smooth functions
- ▶ does not introduce LICQ violations in the transformation

The DNLP grammar

in DNLP, problems are constructed using **atoms** with known attributes

- ▶ atoms are divided into three categories:
 - **smooth**
 - **nonsmooth convex**
 - **nonsmooth concave**
- ▶ DNLP enforces simple rules on how atoms can be composed

The DNLP grammar

in DNLP, problems are constructed using **atoms** with known attributes

- ▶ atoms are divided into three categories:
 - **smooth**
 - **nonsmooth convex**
 - **nonsmooth concave**
- ▶ DNLP enforces simple rules on how atoms can be composed
- ▶ the DNLP rules are very simple if you know the DCP rules:

a problem is DNLP if it is DCP when smooth atoms are treated as linear

Some new DNLP atoms

Atom	Definition
multiply	$\phi(x, y) = xy$
quad_form	$\phi(x) = x^T Q x$ where $Q \in \mathbf{S}^n$
sin	$\phi(x) = \sin x$
sigmoid	$\phi(x) = \frac{1}{1+e^{-x}}$
normal_cdf	$\phi(x) = (1/\sqrt{2\pi}) \int_{-\infty}^x e^{-t^2/2} dt$

- ▶ these atoms are not convex or concave, but they are smooth

Outline

Nonlinear programming

Disciplined nonlinear programming

DNLP examples

Conclusions

Example: Circle packing

$$\begin{array}{ll}\text{minimize} & \max_{i=1,\dots,n} (\|c_i\|_\infty + r_i) \\ \text{subject to} & \|c_i - c_j\|_2^2 \geq (r_i + r_j)^2, \quad 1 \leq i < j \leq n,\end{array}$$

- ▶ goal is to pack n circles with given radii r_i into smallest square
- ▶ variables are the circle centers $c_i \in \mathbf{R}^2$
- ▶ becomes DCP when smooth atoms are linearized, so DNLP-compliant

Example: Circle packing

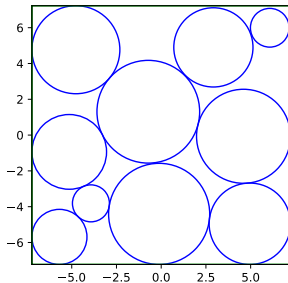
$$\begin{array}{ll}\text{minimize} & \max_{i=1,\dots,n} (\|c_i\|_\infty + r_i) \\ \text{subject to} & \|c_i - c_j\|_2^2 \geq (r_i + r_j)^2, \quad 1 \leq i < j \leq n,\end{array}$$

- ▶ goal is to pack n circles with given radii r_i into smallest square
- ▶ variables are the circle centers $c_i \in \mathbf{R}^2$
- ▶ becomes DCP when smooth atoms are linearized, so DNLP-compliant

```
1 c, constr = Variable((n, 2)), []
2 for i in range(n - 1):
3     constr += [sum((c[i, :] - c[i+1:, :]) ** 2, axis=1) >= (r[i] + r[i+1:]) ** 2]
4 cost = max(norm_inf(c, axis=1) + r)
5 prob = Problem(Minimize(cost), constr)
6 c.value = uniform(-5.0, 5.0, (n, 2))           # random initial point
7 prob.solve(nlp=True)                          # or prob.solve(nlp=True, best_of=500)
```

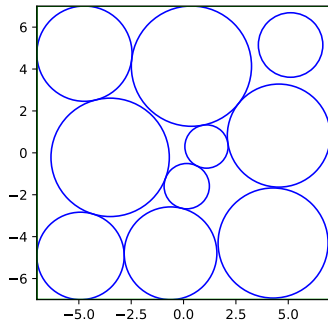
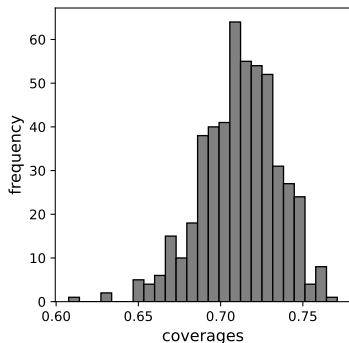
Example: Circle packing

- ▶ result for $n = 10$ circles with random radii, with random initial point
- ▶ the fraction of the square covered by the circles is 0.72



Example: Circle packing

- ▶ result for same problem instance over 500 random initializations using `best_of=500`
- ▶ best coverage found is 0.77



Example: Nonlinear optimal control

$$\begin{aligned} & \text{minimize} && g(u_0, \dots, u_{T-1}) \\ & \text{subject to} && x_{t+1} = f(x_t, u_t), \quad t = 0, \dots, T-1 \\ & && x_0 = x^{\text{init}}, \quad x_T = x^{\text{des}} \\ & && (u_0, \dots, u_{T-1}) \in \mathcal{U} \end{aligned}$$

- ▶ simple car model: state x_t is position and orientation, input u_t is speed and steering angle
- ▶ goal is to drive the car from initial state x^{init} to desired state x^{des}
- ▶ $\mathcal{U}$ is the set of input constraints, e.g., speed and acceleration limits
- ▶ nonlinear dynamics, with time step h and wheelbase L :

$$f(x, u) = x + hu_1 \begin{bmatrix} \cos x_3 \\ \sin x_3 \\ \tan(u_2)/L \end{bmatrix}$$

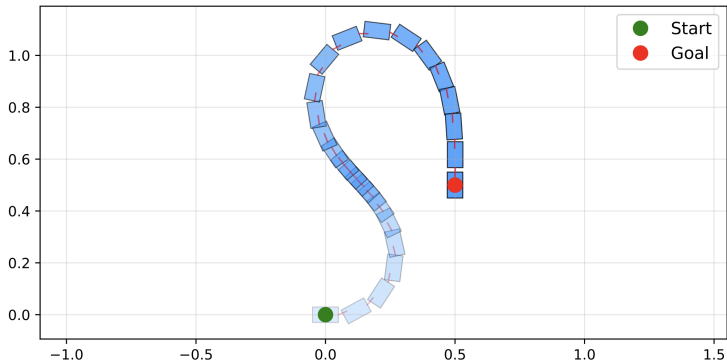
Example: Nonlinear optimal control

- ▶ example with $g(u) = \sum_t \|u_t\|_2^2 + \gamma \sum_t \|u_{t+1} - u_t\|_2^2$ and $\mathcal{U}$ given by limits on speed, steering angle, acceleration, and steering rate
- ▶ variables $x \in \mathbf{R}^{(T+1) \times 3}$ and $u \in \mathbf{R}^{T \times 2}$, with rows indexed by time step

```
1 x, u = Variable((T+1, 3)), Variable((T, 2))
2 cost = sum_squares(u) + gamma * sum_squares(u[1:, :] - u[:-1, :])
3 constr = [x[0, :] == x_init, x[T, :] == x_des]
4 constr += [x[1:, 0] == x[:-1, 0] + h * multiply(u[:, 0], cos(x[:-1, 2])),
5           x[1:, 1] == x[:-1, 1] + h * multiply(u[:, 0], sin(x[:-1, 2])),
6           x[1:, 2] == x[:-1, 2] + (h / L) * multiply(u[:, 0], tan(u[:, 1]))]
7 constr += [abs(u[1:, 0] - u[:-1, 0]) <= a_max * h,
8           abs(u[1:, 1] - u[:-1, 1]) <= omega_max * h]
9 constr += [s_min <= u[:, 0], u[:, 0] <= s_max,
10          phi_min <= u[:, 1], u[:, 1] <= phi_max]
11 prob = Problem(Minimize(cost), constr)
12 prob.solve(nlp=True)
```

Example: Nonlinear optimal control

- car starts at the origin facing right, must end at $(0.5, 0.5)$ facing down



Example: Sparse signal recovery

$$\begin{array}{ll}\text{minimize} & \sum_{i=1}^n \sqrt{|x_i|} \\ \text{subject to} & Ax = y,\end{array}$$

- ▶ goal is to recover a sparse signal $x_0 \in \mathbf{R}^n$ from m given measurements $y = Ax_0$
- ▶ variable is the signal $x \in \mathbf{R}^n$
- ▶ the objective is nonconvex but promotes sparsity better than the ℓ_1 norm
- ▶ becomes DCP when smooth atoms are linearized, so DNLP-compliant

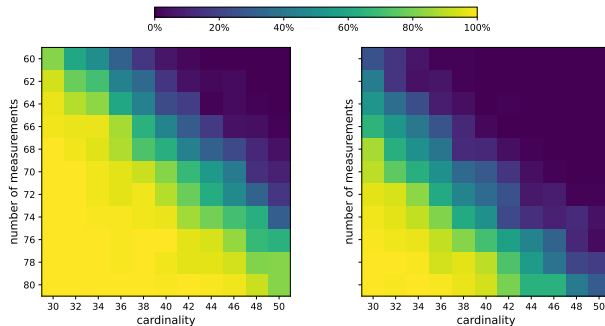
Example: Sparse signal recovery

$$\begin{array}{ll}\text{minimize} & \sum_{i=1}^n \sqrt{|x_i|} \\ \text{subject to} & Ax = y,\end{array}$$

- ▶ goal is to recover a sparse signal $x_0 \in \mathbf{R}^n$ from m given measurements $y = Ax_0$
- ▶ variable is the signal $x \in \mathbf{R}^n$
- ▶ the objective is nonconvex but promotes sparsity better than the ℓ_1 norm
- ▶ becomes DCP when smooth atoms are linearized, so DNLP-compliant

```
1 x = Variable(n)
2 cost, constr = sum(sqrt(abs(x))), [A @ x == y]
3 prob = Problem(Minimize(cost), constr)
4 prob.solve(nlp=True, solver='knitro_ipm')
```

Example: Sparse signal recovery



- probability of successful recovery for signal dimension $n = 100$, nonconvex approach (left) and convex ℓ_1 approach (right)

Outline

Nonlinear programming

Disciplined nonlinear programming

DNLP examples

Conclusions

Conclusions

- ▶ NLP solvers come with weak guarantees
 - may return a local minimizer, a saddle point, or an infeasible point
 - but in practice they often work well, especially with a good initialization
- ▶ disciplined nonlinear programming is a grammar for NLP, analogous to DCP
 - a problem is DNLP if it is DCP when smooth atoms are treated as linear
- ▶ more on DNLP
 - talk: https://www.youtube.com/watch?v=DEgWhUBp__I
 - paper: <https://arxiv.org/abs/2606.02896>