

Constrained Optimization

Basic topics and examples using Julia
Julia v1.5.3

Ricardo A. Fernandes

ricardoaf@lccv.ufal.br

Advisor: Adeildo S. Ramos Jr.

February, 2021



- Introduction
- Constraint types
- Transformations (remove constraints)
- Lagrange multipliers
- Inequality constraints
- Penalty methods
- Augmented Lagrange method
- Linear constrained optimization

General optimization problem equation

$$\begin{array}{ll}\underset{\mathbf{x}}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \mathcal{X}\end{array}$$

Unconstrained problems:

- Feasible set \mathcal{X} is \mathbb{R}^n

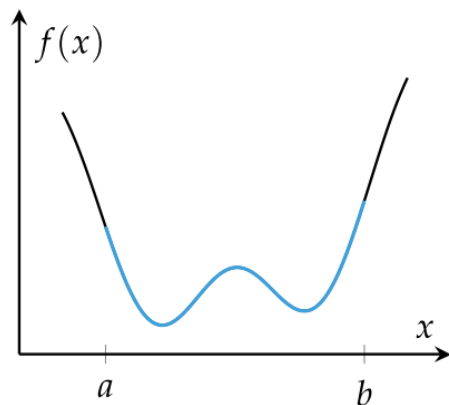
Constrained problems:

- Feasible set: subset of \mathbb{R}^n
- Design points must satisfy certain conditions

Bracketing constraints

Some constraints are simply upper or lower bounds on design variables

Univariate problem

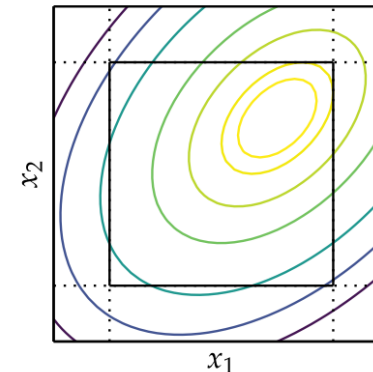


$$\begin{array}{ll}\underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & x \in [a, b]\end{array}$$

Inequality constraints

- $x \geq a$
- $x \leq b$

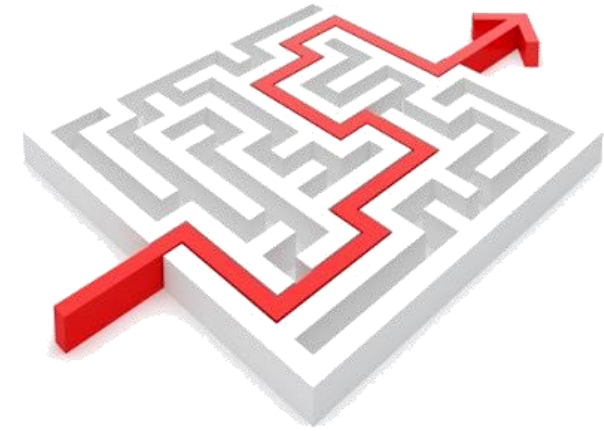
Multivariate problems



Solution must lie within a hyperrectangle

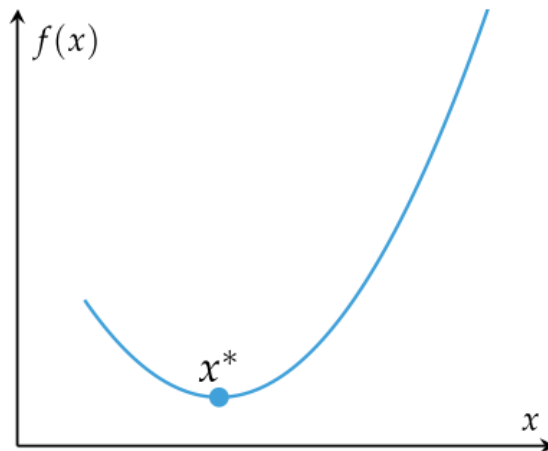
Constrained problems

- Constraints arise naturally when formulating real problems
 - A structure dimension can't exceed a certain design limit
 - A fund manager can't sell more stock than they have
 - Number of hours spend per day on your job can't exceed X
 - In Sudoku, one can't repeat the same number within a row, column or block
- Constraints are included to prevent infeasible solutions

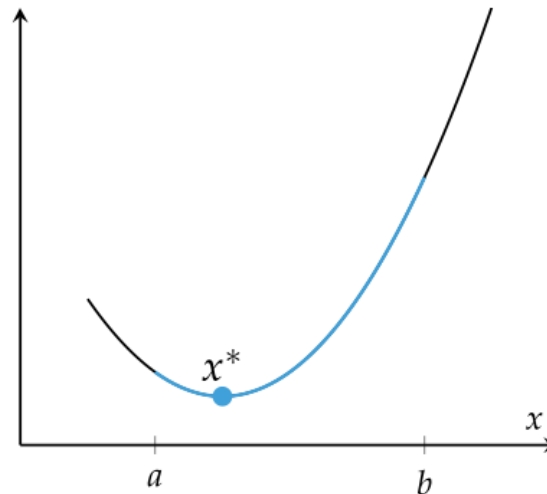


Applying constraints can affect the solution, or not

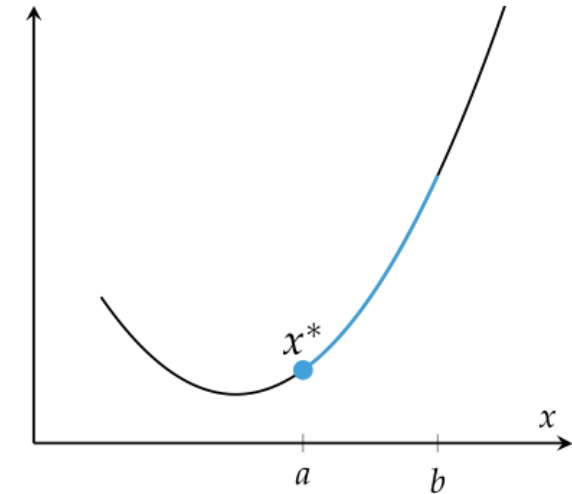
Unconstrained



Constrained, Same Solution



Constrained, New Solution



The two constraint types

- The feasible set is typically formed from:
 - Equality constraints, $h(\mathbf{x}) = 0$
 - Inequality constraints, $g(\mathbf{x}) \leq 0$

Greater-than inequalities $G(x) \geq 0$
can be translated as $-G(x) \leq 0$

Using set membership (not typical)

- $h(x) = (x \notin \chi)$

Using functions

- $h(x) = 0, g(x) \leq 0$
- Functions are often used because they can provide information about how far a given point is from being feasible, helping drive solution methods

Any optimization problem can be rewritten like:

$$\begin{aligned} & \underset{\mathbf{x}}{\text{minimize}} && f(\mathbf{x}) \\ & \text{subject to} && h_i(\mathbf{x}) = 0 \text{ for all } i \text{ in } \{1, \dots, \ell\} \\ & && g_j(\mathbf{x}) \leq 0 \text{ for all } j \text{ in } \{1, \dots, m\} \end{aligned}$$

Equality constraint as two inequality constraints

$$h(\mathbf{x}) = 0 \quad \Longleftrightarrow \quad \begin{cases} h(\mathbf{x}) \leq 0 \\ h(\mathbf{x}) \geq 0 \end{cases}$$

Problem transformation

- Removing constraints
- Such transformations may be possible in some cases

Eliminating design variables

- Consider the equality constraint

$$h(\mathbf{x}) = \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n - \beta = 0$$

- x_n can be solved using the first $n - 1$ variables

$$x_n = 1/\alpha_n [\beta - \alpha_1 x_1 - \alpha_2 x_2 - \cdots - \alpha_{n-1} x_{n-1}]$$

- So, one can transform the problem

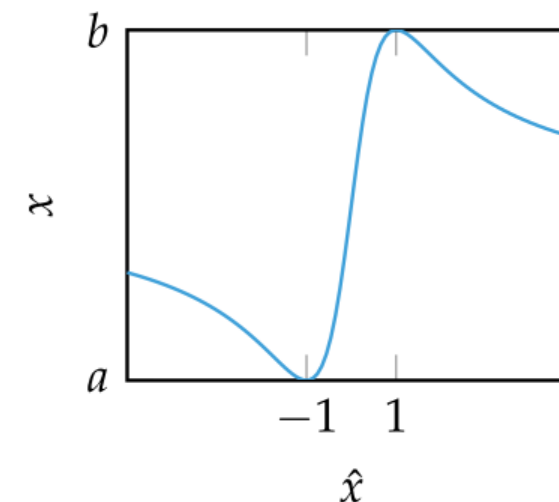
$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{s.t.} & h(x) = 0 \end{array} \quad \text{into}$$

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x_1, \dots, x_{n-1}) \\ \text{with} & x_n = 1/\alpha_n [\beta - \alpha_1 x_1 - \alpha_2 x_2 - \cdots - \alpha_{n-1} x_{n-1}] \end{array}$$

Removing bound constraints

- Bound constraints $a \leq x \leq b$ can be removed

$$x = t_{a,b}(\hat{x}) = \frac{b+a}{2} + \frac{b-a}{2} \left(\frac{2\hat{x}}{1+\hat{x}^2} \right)$$



This transformation ensures that $a \leq x \leq b$

Example: Removing bound constraints

Kochenderfer & Wheeler (2019) Algorithms for Optimization,
MIT Press: Example 10.1 (page 170)

- Consider the following optimization problem

$$\begin{array}{ll}\underset{x}{\text{minimize}} & x \sin(x) \\ \text{subject to} & 2 \leq x \leq 6\end{array}$$

Transformation

$$x = t_{a,b}(\hat{x}) = \frac{b+a}{2} + \frac{b-a}{2} \left(\frac{2\hat{x}}{1+\hat{x}^2} \right)$$

- Problem can be transformed into

$$\underset{\hat{x}}{\text{minimize}} \quad t_{2,6}(\hat{x}) \sin(t_{2,6}(\hat{x}))$$

$$\underset{\hat{x}}{\text{minimize}} \quad \left(4 + 2 \left(\frac{2\hat{x}}{1+\hat{x}^2} \right) \right) \sin \left(4 + 2 \left(\frac{2\hat{x}}{1+\hat{x}^2} \right) \right)$$

- One can solve the unconstrained problem
 - two minimum values can be found
 - $\hat{x} \approx 0.242, \hat{x} \approx 4.139$
 - both values of \hat{x} produce
 - $x = t_{2,6}(\hat{x}) = 4.914$
 - an objective function value of ≈ -4.814

Optimization problem with bound constraints

In [1]: `using Optim, Plots`

```
In [2]: # Objective function
f(x) = x * sin(x)

# Design variable bounds
xmin, xmax = 2., 6.;
```

Constrained Optimization

```
In [3]: # Constrained optimization
function constrainedOptimization(f, xmin, xmax)
```

```
    res = optimize(f, xmin, xmax)
```

```
    conv = Optim.converged(res); println("converged? ", conv)
    xopt = Optim.minimizer(res); println(" xOpt: ", xopt)
    fmin = Optim.minimum(res);   println("  fMin: ", fmin)
```

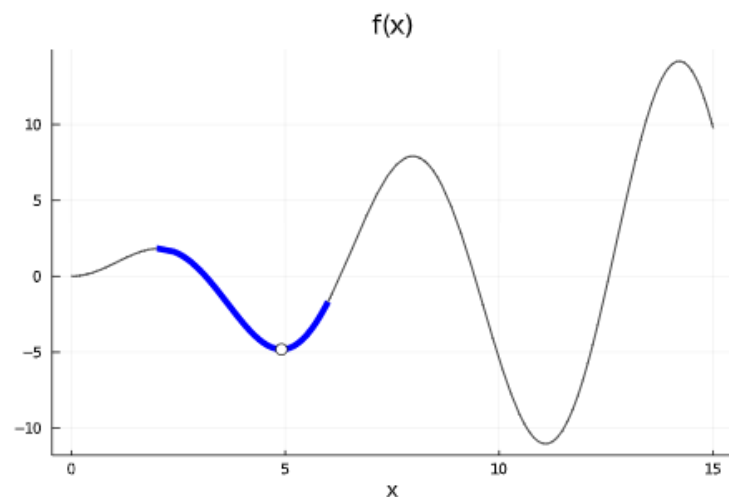
```
    xopt, fmin
end;
```

```
In [4]: # Perform constrained optimization
xopt, fmin = constrainedOptimization(f, xmin, xmax)
```

```
# plot
plot(f, 0, xmin, color=:black, title="f(x)", xlabel="x", legend=false)
plot!(f, xmin, xmax, color=:blue, linewidth=5)
plot!(f, xmax, 15, color=:black)
scatter!([xopt], [fmin], color=:white, markersize=5)
```

```
converged? true
# iter: 10
xOpt: 4.913180438706312
fMin: -4.814469889712268
```

Out[4]:



Unconstrained Optimization (transformed problem)

```
In [5]: # Transformation
x(y) = @. (xmin + xmax) / 2 + (xmax - xmin) * y / (1 + y^2)
fy(y) = f(x(y))
```

```
# Unconstrained optimization: transformed problem
function unconstrainedOptimization(fy, y0, method=LBFGRS())
```

```
    println("\ninitial guess: ", y0)
    res = optimize(y->fy(first(y)), [y0], method)
```

```
    conv = Optim.converged(res); println("converged? ", conv)
    yopt = Optim.minimizer(res); println(" yOpt: ", yopt[1])
    println(" xOpt: ", x(yopt[1]))
    fmin = Optim.minimum(res);   println("  fMin: ", fmin)
```

```
    yopt, fmin
end;
```

```
In [6]: # Perform constrained optimization
```

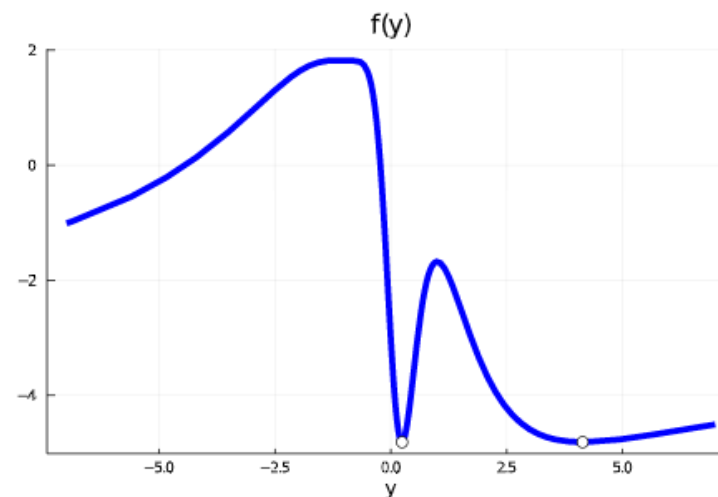
```
# different initial guesses
yopt_a, fmin_a = unconstrainedOptimization(fy, 0.)
yopt_b, fmin_b = unconstrainedOptimization(fy, 0.5)

# plot
plot(fy, -7, 7, color=:blue, linewidth=5, title="f(y)", xlabel="y", legend=false)
scatter!([yopt_a], [fmin_a], color=:white, markersize=5)
scatter!([yopt_b], [fmin_b], color=:white, markersize=5)
```

```
initial guess: 0.0
converged? true
yOpt: 4.138671948497414
xOpt: 4.913180431692558
fMin: -4.814469889712268
```

```
initial guess: 0.5
converged? true
yOpt: 0.24162340533417573
xOpt: 4.913180439619471
fMin: -4.814469889712269
```

Out[6]:



Optim

<https://juliansolvers.github.io/Optim.jl/stable/>



See 01_alg4opt170.ipynb

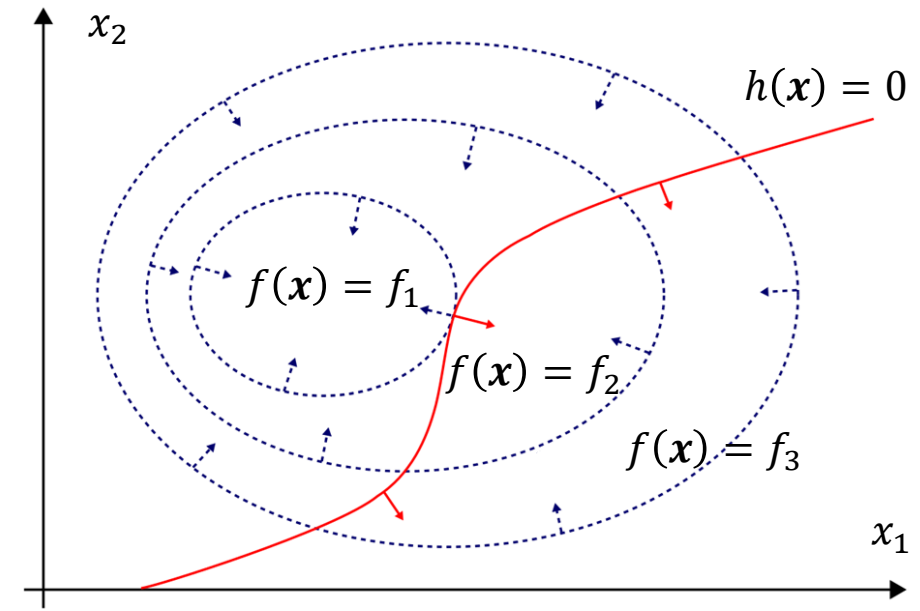
Method of Lagrange Multipliers

Used to optimize a function subjected to equality constraints

Consider an optimization problem where f and h have continuous partial derivatives

$$\begin{array}{ll} \underset{\mathbf{x}}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & h(\mathbf{x}) = 0 \end{array}$$

- Method of Lagrange multipliers is used to compute where a contour line of f is aligned with a contour line of $h = 0$
- Hence, we need to find where the gradient of f and the gradient of h are aligned



- Contour lines of f are lines of constant f
- The gradient of a function at a point is perpendicular to the contour line of that function through that point
- The optimum solution \mathbf{x}^* lies where a contour line of f is aligned with the contour line $h = 0$

Example: Aligned gradients

Kochenderfer & Wheeler (2019) Algorithms for Optimization,
MIT Press: Example 10.3 (page 172)

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & -\exp\left(-\left(x_1x_2 - \frac{3}{2}\right)^2 - \left(x_2 - \frac{3}{2}\right)^2\right) \\ \text{subject to} \quad & x_1 - x_2^2 = 0 \end{aligned}$$

Lagrange Multiplier Motivation

Optimization problem with equality constraint

```
In [1]: using Optim, Calculus, LinearAlgebra, Plots
```

```
In [2]: # Objective function
f(x) = -exp(-(x[1] * x[2] - 3/2)^2 - (x[2] - 3/2)^2)

# Equality constraint
h(x) = x[1] - x[2]^2;
```

Find optimal solution

```
In [3]: # Substitute the constraint into the objective function
x1(x2) = x2.^2
fu(x2) = f([x1(x2); x2])

# solve unconstrained objective
res = optimize(x2 -> fu(first(x2)), [0.], LBFGS())
opt_x2 = first(Optim.minimizer(res))
opt_x = [x1(opt_x2), opt_x2]; min_f = f(opt_x)

println("Optimal x1, x2: ", opt_x)
println("Min f: ", min_f)
```

```
Optimal x1, x2: [1.3578043154345563, 1.1652486067078374]
Min f: -0.887974742266445
```

Alignment of gradients

```
In [4]: # Objective and constraint gradients
∇f = Calculus.gradient(f)
∇h = Calculus.gradient(h);
```

```
In [5]: # eval gradients at optimal point
nf = normalize!(∇f(opt_x))
nh = normalize!(∇h(opt_x));

println("nf: ", nf)
println("nh: ", nh)

nf: [0.3943241793241729, -0.918971404125459]
nh: [0.39432418041471096, -0.9189714036575167]
```

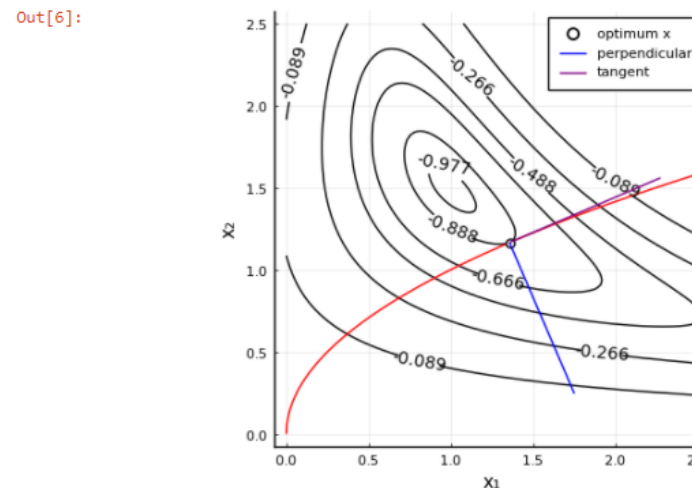
```
In [6]: # plots
pyplot(xlabel="x1", ylabel="x2", colorbar=false, aspect_ratio=:equal)
x = y = LinRange(0, 2.5, 100)

contour(x, y, (x, y)->f([x, y]), levels=[1.1, 1, 0.75, 0.55, 0.3, 0.1]*min_f, c=:black, contour_labels=true)
contour!(x, y, (x, y)->h([x, y]), levels=[0.], c=:red)

scatter!([x1(opt_x2)], [opt_x2], markersize=5, c=:white, label="optimum x")

perpend_x = opt_x + nh
tangent_x = opt_x + [-nh[2], nh[1]]

plot!([opt_x[1], perpend_x[1]], [opt_x[2], perpend_x[2]], color=:blue, label="perpendicular")
plot!([opt_x[1], tangent_x[1]], [opt_x[2], tangent_x[2]], color=:purple, label="tangent")
```



Method of Lagrange Multipliers

So, we seek the best \mathbf{x} such that

- $h(\mathbf{x}) = 0$
- $\nabla f(\mathbf{x}) = \lambda \nabla h(\mathbf{x})$

Lagrange multiplier, λ

- We need the scalar λ because the magnitudes of the gradients may not be the same

We can formulate the *Lagrangian* as

- $\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda h(\mathbf{x})$

Solving $\nabla \mathcal{L}(\mathbf{x}, \lambda) = \mathbf{0}$

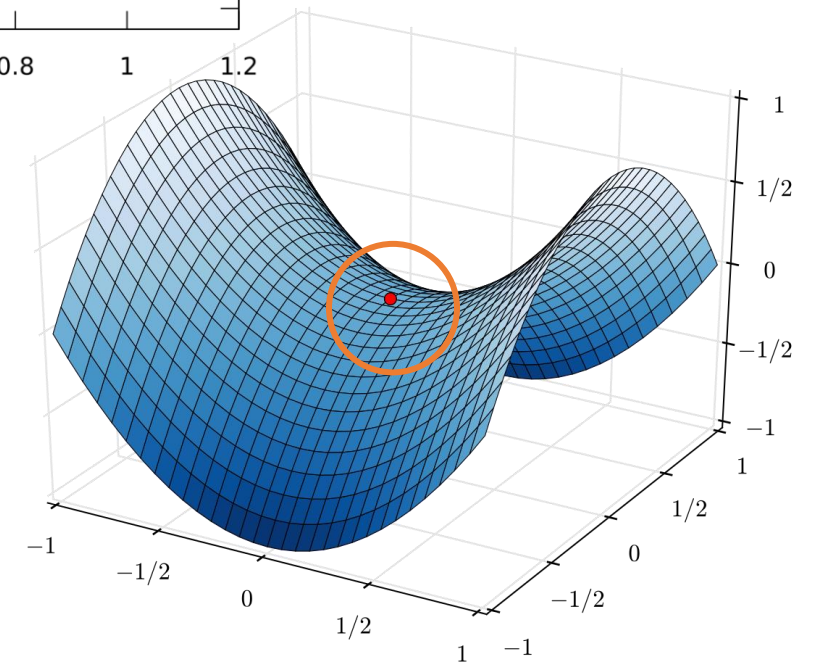
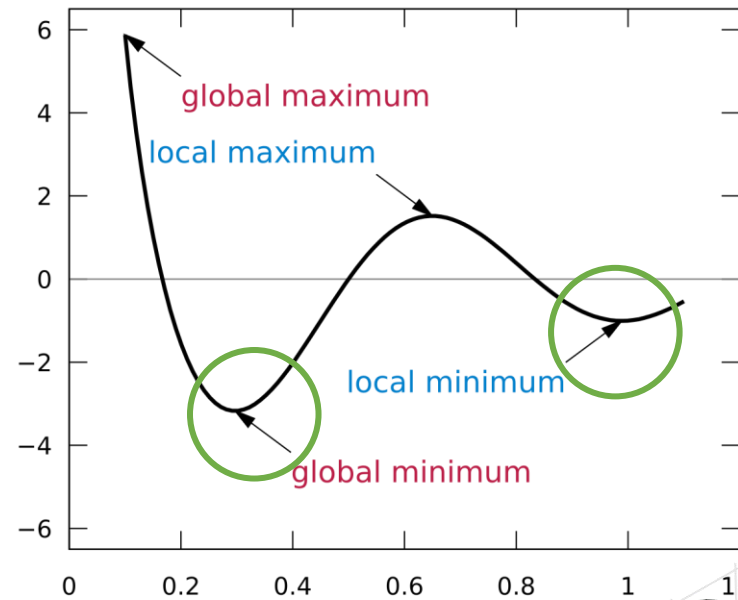
- $\nabla_{\mathbf{x}} \mathcal{L} = \mathbf{0} \Rightarrow \nabla f(\mathbf{x}) = \lambda \nabla h(\mathbf{x})$
- $\nabla_{\lambda} \mathcal{L} = 0 \Rightarrow h(\mathbf{x}) = 0$

Necessary optimality conditions
(not sufficient)

Any solution is considered a *critical point*

- Local/global minimum
- Saddle points

For convex problems,
Necessary conditions = sufficient conditions



Consider the same last example

$$\begin{array}{ll}\underset{\mathbf{x}}{\text{minimize}} & -\exp\left(-\left(x_1x_2 - \frac{3}{2}\right)^2 - \left(x_2 - \frac{3}{2}\right)^2\right) \\ \text{subject to} & x_1 - x_2^2 = 0\end{array}$$

- Form the Lagrangian

$$\mathcal{L}(x_1, x_2, \lambda) = -\exp\left(-\left(x_1x_2 - \frac{3}{2}\right)^2 - \left(x_2 - \frac{3}{2}\right)^2\right) - \lambda(x_1 - x_2^2)$$

- Compute the gradient and set the derivatives to zero

$$\frac{\partial \mathcal{L}}{\partial x_1} = 2x_2 f(\mathbf{x}) \left(\frac{3}{2} - x_1x_2\right) - \lambda$$

$$\frac{\partial \mathcal{L}}{\partial x_2} = 2\lambda x_2 + f(\mathbf{x}) \left(-2x_1(x_1x_2 - \frac{3}{2}) - 2(x_2 - \frac{3}{2})\right)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = x_2^2 - x_1$$

- Solve equations and the same solution is obtained:
 - $x_1 \approx 1.358, x_2 \approx 1.165$
 - with Lagrange multiplier, $\lambda \approx 0.170$

Lagrange Multipliers for multiple equality constraints

Consider the following optimization problem

$$\begin{array}{ll}\underset{\mathbf{x}}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & h_1(\mathbf{x}) = 0 \\ & h_2(\mathbf{x}) = 0 \\ & \vdots\end{array}$$

- Lagrangian can be defined as

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) &= f(\mathbf{x}) - \sum_i \lambda_i h_i(\mathbf{x}) \\ &= f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x})\end{aligned}$$

Example: Minimization of Complementary Energy

- Optimization problem

$$\begin{aligned} &\underset{\mathbf{r}}{\text{minimize}} && \Pi_c(\mathbf{r}) \\ &\text{s.t.} && \mathbf{B}^T \mathbf{r} = \mathbf{f} \end{aligned}$$

where:

\mathbf{r} : Internal forces of the bars

\mathbf{f} : Nodal external forces

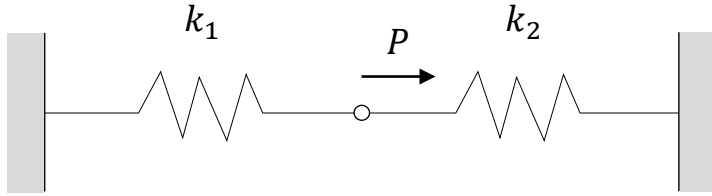
\mathbf{B}^T : Equilibrium matrix

\mathbf{C}^{-1} : Internal Flexibility matrix

- Complementary Energy

$$\Pi_c(\mathbf{r}) = \frac{1}{2} \mathbf{r}^T \mathbf{C}^{-1} \mathbf{r}$$

Example: Minimization of Complementary Energy



- Equilibrium condition (internal x external forces)

$$\begin{array}{c}
 r_1 \quad \xrightarrow{P} \quad r_2 \\
 \longleftarrow \quad \longrightarrow \quad r_1 - r_2 = P
 \end{array}$$

- Complementary Energy

$$\Pi_c(r_1, r_2) = \frac{1}{2} \frac{r_1^2}{k_1} + \frac{1}{2} \frac{r_2^2}{k_2}$$

- Optimization Problem

$$\begin{array}{ll}
 \underset{r_1, r_2}{\text{minimize}} & \Pi_c(r_1, r_2) \\
 \text{s.t.} & r_1 - r_2 = P
 \end{array}$$

Complementary Energy Example

Complementary Energy and Equilibrium Condition

$$\Pi_c := (r_1, r_2) \rightarrow \frac{1}{2} \cdot \frac{1}{k_1} \cdot r_1^2 + \frac{1}{2} \cdot \frac{1}{k_2} \cdot r_2^2 :$$

$$h := (r_1, r_2) \rightarrow r_2 - r_1 + P :$$

using Force method

$$r_2 := \text{solve}(h(r_1, r_2) = 0, r_2) = r_1 - P$$

$$\Pi_c(r_1, r_2) = \frac{r_1^2}{2 k_1} + \frac{(r_1 - P)^2}{2 k_2}$$

$$r_1 := \text{solve}(\text{diff}(\Pi_c(r_1, r_2), r_1) = 0, r_1) = \frac{P k_1}{k_2 + k_1}$$

$$r_2 := \text{solve}(h(r_1, r_2) = 0, r_2) = -\frac{P k_2}{k_2 + k_1}$$

using Lagrange multiplier

$$L := (r_1, r_2, \lambda) \rightarrow \Pi_c(r_1, r_2) + \lambda \cdot h(r_1, r_2) :$$

$$eq_1 := \text{diff}(L(r_1, r_2, \lambda), r_1) = 0 = \frac{r_1}{k_1} - \lambda = 0$$

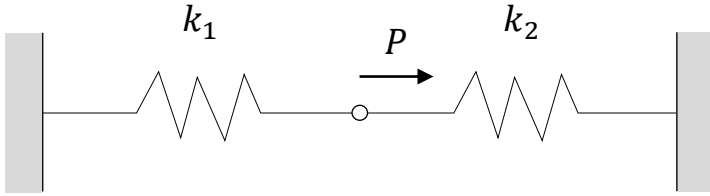
$$eq_2 := \text{diff}(L(r_1, r_2, \lambda), r_2) = 0 = \frac{r_2}{k_2} + \lambda = 0$$

$$eq_3 := \text{diff}(L(r_1, r_2, \lambda), \lambda) = 0 = r_2 - r_1 + P = 0$$

$$\text{solve}(\{eq_1, eq_2, eq_3\}, \{r_1, r_2, \lambda\}) = \left\{ \lambda = \frac{P}{k_2 + k_1}, r_1 = \frac{P k_1}{k_2 + k_1}, r_2 = -\frac{P k_2}{k_2 + k_1} \right\}$$

Lagrange Multipliers

Example: Minimization of Complementary Energy



- Equilibrium condition (internal x external forces)

$$\begin{array}{c} r_1 \quad \xrightarrow{P} \quad r_2 \\ \leftarrow \quad \circ \quad \rightarrow \end{array} \quad r_1 - r_2 = P$$

- Complementary Energy

$$\Pi_c(r_1, r_2) = \frac{1}{2} \frac{r_1^2}{k_1} + \frac{1}{2} \frac{r_2^2}{k_2}$$

- Optimization Problem

$$\begin{array}{ll} \underset{r_1, r_2}{\text{minimize}} & \Pi_c(r_1, r_2) \\ \text{s.t.} & r_1 - r_2 = P \end{array}$$

Complementary Energy Example

```
In [1]: using JuMP, Ipopt
```

```
In [2]: # Problem data
P, k1, k2 = 10., 100., 200.

# Complementary energy and equilibrium condition
Πc(r1, r2) = 1/2 * r1^2 / k1 + 1/2 * r2^2 / k2
h(r1, r2) = r2 - r1 + P;

# reference results
r1ref(P, k1, k2) = P * k1 / (k1 + k2)
r2ref(P, k1, k2) = -P * k2 / (k1 + k2)

println("r1ref : ", r1ref(P, k1, k2), ", r2ref : ", r2ref(P, k1, k2))

r1ref : 3.3333333333333335, r2ref : -6.666666666666667
```

```
In [3]: m = Model(Ipopt.Optimizer)
set_optimizer_attribute(m, "print_level", 0)

@variable(m, r1)
@variable(m, r2)

@objective(m, Min, Πc(r1, r2))

@constraint(m, h(r1, r2) == 0)

println(m)
optimize!(m)

println("Termination status: ", termination_status(m))
println("Primal status: ", primal_status(m))

println(" f* : ", objective_value(m))
println("r1* : ", value(r1), ", r2* : ", value(r2))

Min 0.005 r1^2 + 0.0025 r2^2
Subject to
-r1 + r2 == -10.0
```

```
*****
This program contains Ipopt, a library for large-scale nonlinear optimization.
Ipopt is released as open source code under the Eclipse Public License (EPL).
For more information visit http://projects.coin-or.org/Ipopt
*****
```

```
Termination status: LOCALLY_SOLVED
Primal status: FEASIBLE_POINT
f* : 0.16666666666666669
r1* : 3.3333333333333335, r2* : -6.666666666666667
```



See 03_complementaryenergy.ipynb

JuMP depends on solvers to solve optimization problems

<https://jump.dev/JuMP.jl/v0.21.1/installation/#Getting-Solvers-1>

| Solver | Julia Package | License | Supports |
|----------------|---------------|---------|------------------------------------|
| Artelys Knitro | KNITRO.jl | Comm. | LP, MILP, SOCP, MISOCP, NLP, MINLP |
| Cbc | Cbc.jl | EPL | MILP |
| CDCS | CDCS.jl | GPL | LP, SOCP, SDP |
| CDD | CDDLib.jl | GPL | LP |
| Clp | Clp.jl | EPL | LP |
| COSMO | COSMO.jl | Apache | LP, QP, SOCP, SDP |
| CPLEX | CPLEX.jl | Comm. | LP, MILP, SOCP, MISOCP |
| CSDP | CSDP.jl | EPL | LP, SDP |
| ECOS | ECOS.jl | GPL | LP, SOCP |
| FICO Xpress | Xpress.jl | Comm. | LP, MILP, SOCP, MISOCP |
| GLPK | GLPK.jl | GPL | LP, MILP |
| Gurobi | Gurobi.jl | Comm. | LP, MILP, SOCP, MISOCP |
| Ipopt | Ipopt.jl | EPL | LP, QP, NLP |

| | | | |
|---------|------------------------|----------|-----------------------------|
| Juniper | Juniper.jl | MIT | MISOCP, MINLP |
| MOSEK | MosekTools.jl | Comm. | LP, MILP, SOCP, MISOCP, SDP |
| OSQP | OSQP.jl | Apache | LP, QP |
| ProxSDP | ProxSDP.jl | MIT | LP, SOCP, SDP |
| SCIP | SCIP.jl | ZIB | MILP, MINLP |
| SCS | SCS.jl | MIT | LP, SOCP, SDP |
| SDPA | SDPA.jl, SDPAFamily.jl | GPL | LP, SDP |
| SDPNAL | SDPNAL.jl | CC BY-SA | LP, SDP |
| SDPT3 | SDPT3.jl | GPL | LP, SOCP, SDP |
| SeDuMi | SeDuMi.jl | GPL | LP, SOCP, SDP |
| Tulip | Tulip.jl | MPL-2 | LP |

Where:

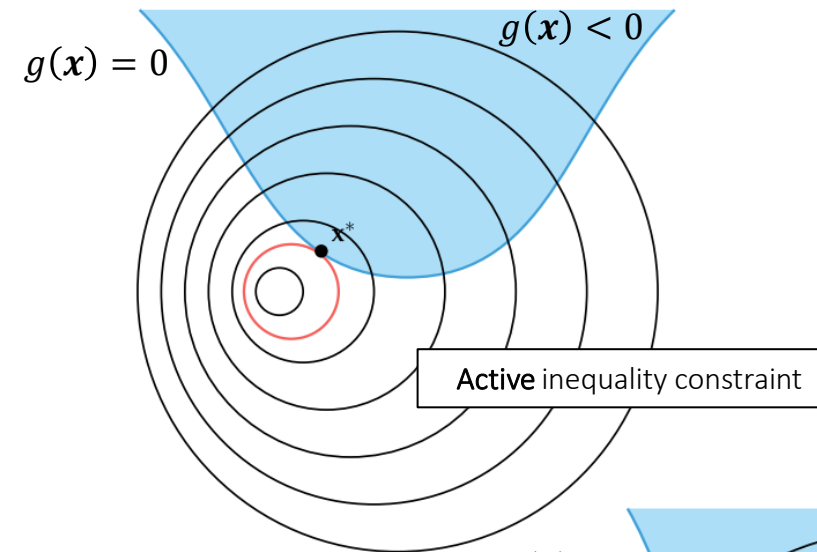
- LP = Linear programming
- QP = Quadratic programming
- SOCP = Second-order conic programming (including problems with convex quadratic constraints and/or objective)
- MILP = Mixed-integer linear programming
- NLP = Nonlinear programming
- MINLP = Mixed-integer nonlinear programming
- SDP = Semidefinite programming
- MISOCP = Mixed-integer semidefinite programming

Inequality constraints

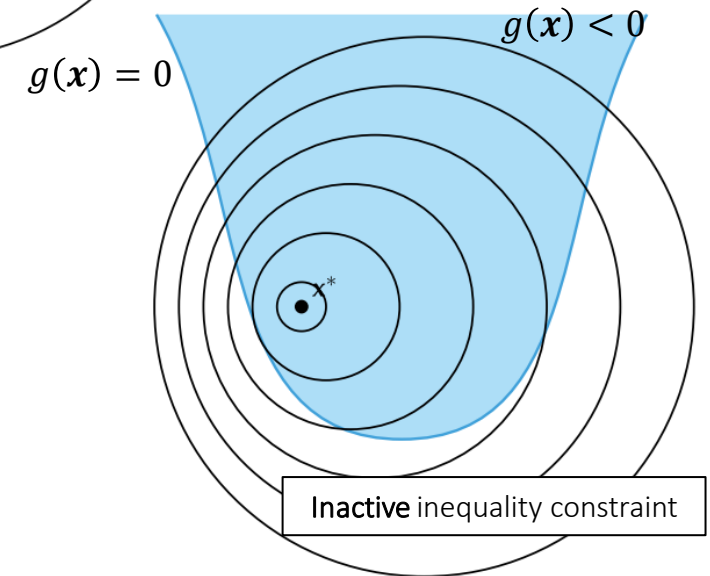
Consider a problem with a single inequality constraint

$$\begin{array}{ll}\underset{\mathbf{x}}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & g(\mathbf{x}) \leq 0\end{array}$$

- If the solution lies at the constraint boundary,
 - the Lagrange condition holds for some constant μ
 - the constraint is considered **active**
- If the solution does not lie at the constraint boundary,
 - the constraint is considered **inactive**
 - solutions will lie where $\nabla f = 0$, as with unconstrained optimization
 - the Lagrange condition holds by setting $\mu = 0$
- The Lagrangian of the problem is $\mathcal{L}(\mathbf{x}, \mu) = f(\mathbf{x}) + \mu g(\mathbf{x})$
- In order to penalize the objective, for inequalities $\mu \geq 0$



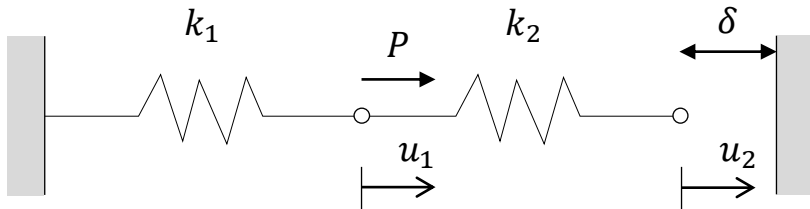
$$\nabla f + \mu \nabla g = \mathbf{0}$$



- Total Potential Energy

$$\Pi(\mathbf{u}) = \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u} - \mathbf{f}^T \mathbf{u}$$

Example: Contact Problem



- Contact condition

$$u_2 \leq \delta$$

- Optimization Problem

$$\begin{aligned} \underset{u_1, u_2}{\text{minimize}} \quad & \Pi(u_1, u_2) = \frac{1}{2} k_1 u_1^2 + \frac{1}{2} k_2 (u_2 - u_1)^2 - P u_1 \\ \text{s.t.} \quad & u_2 - \delta \leq 0 \end{aligned}$$

Contact Problem

restart: with(LinearAlgebra) :

Set stiffness matrix

$$K := \langle k_1 + k_2, -k_2; -k_2, k_2 \rangle :$$

Approach #1: Solving linear system, $Ku = f$

Unconstrained problem

$$F := \langle P, 0 \rangle :$$

$$u := \text{LinearSolve}(K, F) :$$

$$u^{\%T} = \begin{bmatrix} \frac{P}{k_1} & \frac{P}{k_1} \end{bmatrix}$$

Constrained problem

$$u := \langle 0, \delta \rangle : \text{free} := 1 : \text{fix} := 2 :$$

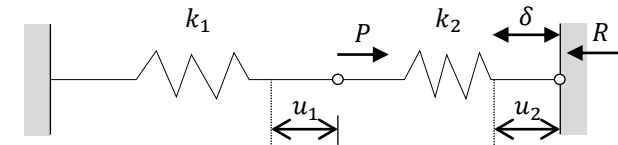
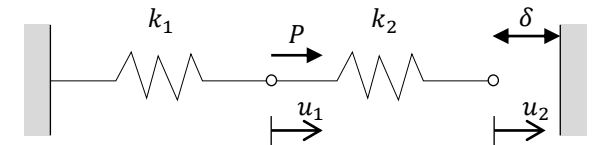
Apply partition method for solving the unknowns

$$u[\text{free}] := \frac{F[\text{free}] - K[\text{free}, \text{fix}] \cdot u[\text{fix}]}{K[\text{free}, \text{free}]} :$$

$$F[\text{fix}] := K[\text{fix}, \text{free}] \cdot u[\text{free}] + K[\text{fix}, \text{fix}] \cdot u[\text{fix}] :$$

$$u^{\%T} = \begin{bmatrix} \frac{k_2 \delta + P}{k_1 + k_2} & \delta \end{bmatrix}$$

$$F^{\%T} = \begin{bmatrix} P & k_2 \delta - \frac{k_2 (k_2 \delta + P)}{k_1 + k_2} \end{bmatrix}$$

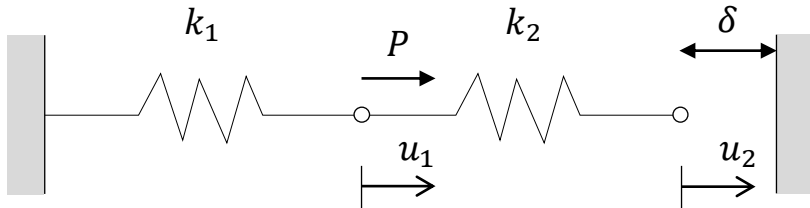


Inequality Constraints

- Total Potential Energy

$$\Pi(\mathbf{u}) = \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u} - \mathbf{f}^T \mathbf{u}$$

Example: Contact Problem



- Contact condition

$$u_2 \leq \delta$$

- Optimization Problem

$$\begin{aligned} \underset{u_1, u_2}{\text{minimize}} \quad & \Pi(u_1, u_2) = \frac{1}{2} k_1 u_1^2 + \frac{1}{2} k_2 (u_2 - u_1)^2 - P u_1 \\ \text{s.t.} \quad & u_2 - \delta \leq 0 \end{aligned}$$

Approach #2: Solving a optimization problem

Total Potential Energy and Inequality constraint

$$\Pi := (u_1, u_2) \rightarrow \frac{1}{2} \cdot k_1 \cdot u_1^2 + \frac{1}{2} \cdot k_2 \cdot (u_2 - u_1)^2 - P \cdot u_1$$

$$g := (u_1, u_2) \rightarrow u_2 - \delta$$

Lagrangian

$$L := \text{unapply}(\Pi(u_1, u_2) + \mu \cdot g(u_1, u_2), u_1, u_2, \mu)$$

$$L(u_1, u_2, \mu) = \frac{k_1 u_1^2}{2} + \frac{k_2 (u_2 - u_1)^2}{2} - P u_1 + \mu (u_2 - \delta)$$

Number of cases are 2^n , where n is the number of inequality constraints

Here, $n=1$, so the two cases are the unconstrained and the constrained cases

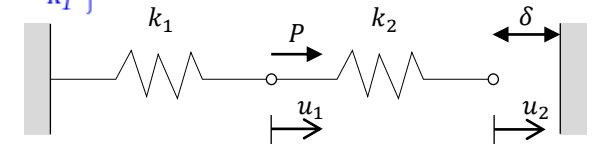
Unconstrained problem

Set Lagrange multiplier to zero (remove constraint from Lagrangian)

$$eq_1 := D[1](L)(u_1, u_2, 0) = k_1 u_1 - k_2 (u_2 - u_1) - P$$

$$eq_2 := D[2](L)(u_1, u_2, 0) = k_2 (u_2 - u_1)$$

$$\text{solve}(\{eq_1 = 0, eq_2 = 0\}, \{u_1, u_2\}) = \left\{ u_1 = \frac{P}{k_1}, u_2 = \frac{P}{k_1} \right\}$$



Constrained problem

Constraint g must be zero

$$u_2 := \text{solve}(g(u_1, u_2) = 0, u_2) = \delta$$

$$eq_1 := D[1](L)(u_1, u_2, \mu) = k_1 u_1 - k_2 (\delta - u_1) - P$$

$$eq_2 := D[2](L)(u_1, u_2, \mu) = k_2 (\delta - u_1) + \mu$$

$$\text{solve}(\{eq_1 = 0, eq_2 = 0\}, \{u_1, \mu\}) = \left\{ \mu = \frac{k_2 (-\delta k_1 + P)}{k_1 + k_2}, u_1 = \frac{k_2 \delta + P}{k_1 + k_2} \right\}$$

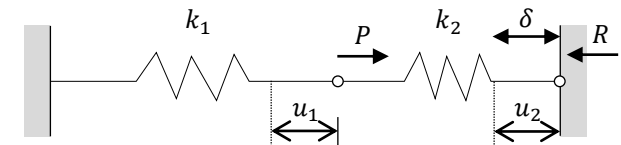
$$\mu = R$$

assign(%) :

Lagrange multiplier and reaction force

$$R := -F[2]:$$

$$\text{simplify}(R - \mu) = 0$$



For inequality constraints, since $P - \delta \cdot k_1 \geq 0$, or reaction force due to contact is always non-negative, lagrange multiplier must always be non-negative too!

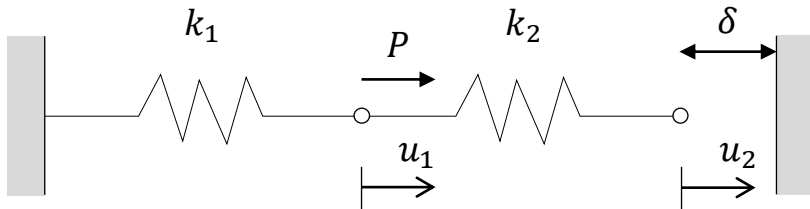
- Total Potential Energy

$$\Pi(\mathbf{u}) = \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u} - \mathbf{f}^T \mathbf{u}$$

Given this optimization problem

Homework

Example: Contact Problem



- Contact condition

$$u_2 \leq \delta$$

- Optimization Problem

$$\begin{aligned} &\underset{u_1, u_2}{\text{minimize}} && \Pi(u_1, u_2) = \frac{1}{2} k_1 u_1^2 + \frac{1}{2} k_2 (u_2 - u_1)^2 - P u_1 \\ &\text{s.t.} && u_2 - \delta \leq 0 \end{aligned}$$

- A) Consider:

- $P = 100$ kN
- $k_1 = k_2 = 1000$ kN/m
- $\delta = 0.15$ m

- Develop a computational routine to solve the problem
- Verify the results with the analytical solution

- B) Repeat A) but now with $P = 200$ kN

Optimizing the problem

Requires finding critical points \mathbf{x}^* such that

- The point is feasible $g(\mathbf{x}^*) \leq 0$
- The penalty must point in the right direction $\mu \geq 0$
- Active constraint, $g(x) = 0$
- Inactive constraint $\mu = 0$ $\mu g(\mathbf{x}^*) = 0$
- Active constraint, Lagrange condition holds
- Inactive constraint, optimum will have
 - $\nabla f(\mathbf{x}^*) = \mathbf{0}$
 - $\mu = 0$ $\nabla f(\mathbf{x}^*) + \mu \nabla g(\mathbf{x}^*) = \mathbf{0}$

KKT Conditions

Generalizing for multiple equality and inequality constraints

- **Feasibility** $g(\mathbf{x}^*) \leq 0$
All constraints satisfied $\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$
- **Dual feasibility** $\mu \geq 0$
Penalization is toward feasibility
- **Complementary slackness** $\mu_i g_i(\mathbf{x}^*) = 0$
Either $\mu_i = 0$ or $g_i(\mathbf{x}^*) = 0$
- **Stationarity**
 f contour is tangent to each active constraint
$$\nabla f(\mathbf{x}^*) + \sum_i \mu_i \nabla g_i(\mathbf{x}^*) + \sum_j \lambda_j \nabla h_j(\mathbf{x}^*) = \mathbf{0}$$

- First-order necessary conditions for optimality
- Identified critical points still should be tested for local minima
- For convex problems, these conditions are already sufficient conditions for optimality

Example: Solution to the KKT necessary conditions

Arora (2012) Introduction to Optimum Design, Elsevier: Example 4.32 (page 150)

Minimize

$$f(x_1, x_2) = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 2$$

subject to

$$g_1 = -2x_1 - x_2 + 4 \leq 0, \quad g_2 = -x_1 - 2x_2 + 4 \leq 0$$

Objective function and inequality constraints

$$f := (x_1, x_2) \rightarrow x_1^2 + x_2^2 - 2 \cdot x_1 - 2 \cdot x_2 + 2 :$$

$$g_1 := (x_1, x_2) \rightarrow -2 \cdot x_1 - x_2 + 4 :$$

$$g_2 := (x_1, x_2) \rightarrow -x_1 - 2 \cdot x_2 + 4 :$$

Generalized Lagrangian

$$l := f(x_1, x_2) + \mu_1 \cdot g_1(x_1, x_2) + \mu_2 \cdot g_2(x_1, x_2) :$$

$$L := \text{unapply}(l, x_1, x_2, \mu_1, \mu_2) :$$

KKT conditions (Stationarity and feasibility)

$$eq_1 := D[1](L)(x_1, x_2, \mu_1, \mu_2) = 0 :$$

$$eq_2 := D[2](L)(x_1, x_2, \mu_1, \mu_2) = 0 :$$

$$eq_3 := g_1(x_1, x_2) = G_1 :$$

$$eq_4 := g_2(x_1, x_2) = G_2 :$$

Equations and unknowns

$$eq := \text{unapply}(\{eq_1, eq_2, eq_3, eq_4\}, x_1, x_2, \mu_1, \mu_2, G_1, G_2) :$$

KKT conditions (Complementary slackness and dual feasibility)

Check 2^n μ_1, μ_2, G_1, G_2 combinations for $\mu_1, \mu_2 \geq 0$ and $G_1, G_2 \leq 0$

$$case_1 := 0, 0, G_1, G_2 : vars_1 := G_1, G_2 :$$

$$case_2 := 0, \mu_2, G_1, 0 : vars_2 := \mu_2, G_1 :$$

$$case_3 := \mu_1, 0, 0, G_2 : vars_3 := \mu_1, G_2 :$$

$$case_4 := \mu_1, \mu_2, 0, 0 : vars_4 := \mu_1, \mu_2 :$$

$$\text{solve}(eq(x_1, x_2, case_1), \{x_1, x_2, vars_1\}) = \{G_1 = 1, G_2 = 1, x_1 = 1, x_2 = 1\}$$

$$\text{solve}(eq(x_1, x_2, case_2), \{x_1, x_2, vars_2\}) = \left\{G_1 = \frac{1}{5}, \mu_2 = \frac{2}{5}, x_1 = \frac{6}{5}, x_2 = \frac{7}{5}\right\}$$

$$\text{solve}(eq(x_1, x_2, case_3), \{x_1, x_2, vars_3\}) = \left\{G_2 = \frac{1}{5}, \mu_1 = \frac{2}{5}, x_1 = \frac{7}{5}, x_2 = \frac{6}{5}\right\}$$

$$\text{solve}(eq(x_1, x_2, case_4), \{x_1, x_2, vars_4\}) = \left\{\mu_1 = \frac{2}{9}, \mu_2 = \frac{2}{9}, x_1 = \frac{4}{3}, x_2 = \frac{4}{3}\right\}$$

Feasible solution: Case 4

$$\text{assign}(\text{solve}(eq(x_1, x_2, case_4), \{x_1, x_2, vars_4\}))$$

$$x_1, x_2 = \frac{4}{3}, \frac{4}{3}$$

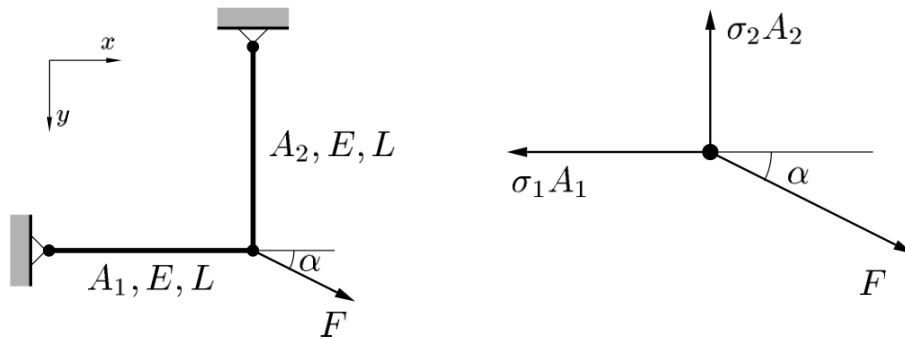
$$\mu_1, \mu_2 = \frac{2}{9}, \frac{2}{9}$$

$$f(x_1, x_2) = \frac{2}{9}$$

$$g_1(x_1, x_2), g_2(x_1, x_2) = 0, 0$$

Example: Weight minimization with stress constraints

Christensen & Klarbring (2009) An Introduction to Structural Optimization, Springer: Section 2.1 (page 10)



$$\begin{aligned} \min_{A_1, A_2} \quad & A_1 + A_2 \\ \text{s.t.} \quad & \begin{cases} A_1 \geq \frac{F \cos \alpha}{\sigma_0} \\ A_2 \geq \frac{F \sin \alpha}{\sigma_0} \end{cases} \end{aligned}$$

Objective function and inequality constraints

$$f := (A_1, A_2) \rightarrow A_1 + A_2 :$$

$$g_1 := (A_1, A_2) \rightarrow \frac{F \cos(\alpha)}{\sigma_0} - A_1 :$$

$$g_2 := (A_1, A_2) \rightarrow \frac{F \sin(\alpha)}{\sigma_0} - A_2 :$$

Generalized Lagrangian

$$l := f(A_1, A_2) + \mu_1 g_1(A_1, A_2) + \mu_2 g_2(A_1, A_2) :$$

$$L := \text{unapply}(l, A_1, A_2, \mu_1, \mu_2) :$$

KKT conditions (Stationarity and feasibility)

$$eq_1 := D[1](L)(A_1, A_2, \mu_1, \mu_2) = 0 :$$

$$eq_2 := D[2](L)(A_1, A_2, \mu_1, \mu_2) = 0 :$$

$$eq_3 := g_1(A_1, A_2) = G_1 :$$

$$eq_4 := g_2(A_1, A_2) = G_2 :$$

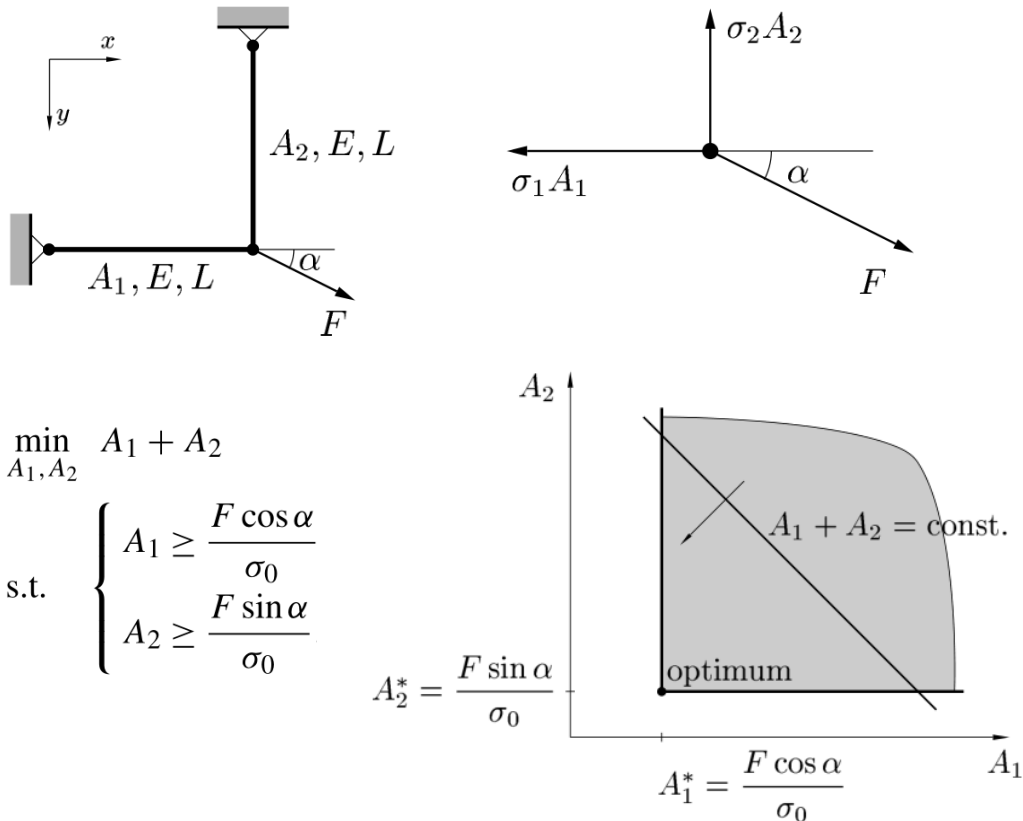
Equations and unknowns

$$eq := \text{unapply}(\{eq_1, eq_2, eq_3, eq_4\}, A_1, A_2, \mu_1, \mu_2, G_1, G_2) :$$

Inequality Constraints

Example: Weight minimization with stress constraints

Christensen & Klarbring (2009) An Introduction to Structural Optimization, Springer: Section 2.1 (page 10)



$$\min_{A_1, A_2} A_1 + A_2$$

$$\text{s.t.} \quad \begin{cases} A_1 \geq \frac{F \cos \alpha}{\sigma_0} \\ A_2 \geq \frac{F \sin \alpha}{\sigma_0} \end{cases}$$

$$A_2^* = \frac{F \sin \alpha}{\sigma_0}$$

$$A_1^* = \frac{F \cos \alpha}{\sigma_0}$$

KKT conditions (Complementary slackness and dual feasibility)

Check 2^n μ_1, μ_2, G_1, G_2 combinations for $\mu_1, \mu_2 \geq 0$ and $G_1, G_2 \leq 0$

$$\text{case}_1 := 0, 0, G_1, G_2: \text{vars}_1 := G_1, G_2:$$

$$\text{case}_2 := 0, \mu_2, G_1, 0: \text{vars}_2 := \mu_2, G_1:$$

$$\text{case}_3 := \mu_1, 0, 0, G_2: \text{vars}_3 := \mu_1, G_2:$$

$$\text{case}_4 := \mu_1, \mu_2, 0, 0: \text{vars}_4 := \mu_1, \mu_2:$$

$$\text{eq}(A_1, A_2, \text{case}_1) = \left\{ 1 = 0, \frac{F \cos(\alpha)}{\sigma_0} - A_1 = G_1, \frac{F \sin(\alpha)}{\sigma_0} - A_2 = G_2 \right\}$$

$$\text{eq}(A_1, A_2, \text{case}_2) = \left\{ 1 = 0, 1 - \mu_2 = 0, \frac{F \cos(\alpha)}{\sigma_0} - A_1 = G_1, \frac{F \sin(\alpha)}{\sigma_0} - A_2 = 0 \right\}$$

$$\text{eq}(A_1, A_2, \text{case}_3) = \left\{ 1 = 0, 1 - \mu_1 = 0, \frac{F \cos(\alpha)}{\sigma_0} - A_1 = 0, \frac{F \sin(\alpha)}{\sigma_0} - A_2 = G_2 \right\}$$

$$\text{eq}(A_1, A_2, \text{case}_4) = \left\{ 1 - \mu_1 = 0, 1 - \mu_2 = 0, \frac{F \cos(\alpha)}{\sigma_0} - A_1 = 0, \frac{F \sin(\alpha)}{\sigma_0} - A_2 = 0 \right\}$$

$$\text{solve}(\text{eq}(A_1, A_2, \text{case}_4), \{A_1, A_2, \text{vars}_4\}) = \left\{ A_1 = \frac{F \cos(\alpha)}{\sigma_0}, A_2 = \frac{F \sin(\alpha)}{\sigma_0}, \mu_1 = 1, \mu_2 = 1 \right\}$$

Feasible solution: Case 4

$$\text{assign}(\text{solve}(\text{eq}(A_1, A_2, \text{case}_4), \{A_1, A_2, \text{vars}_4\}))$$

$$A_1, A_2 = \frac{F \cos(\alpha)}{\sigma_0}, \frac{F \sin(\alpha)}{\sigma_0}$$

$$\mu_1, \mu_2 = 1, 1$$

$$f(A_1, A_2) = \frac{F \cos(\alpha)}{\sigma_0} + \frac{F \sin(\alpha)}{\sigma_0}$$

$$g_1(A_1, A_2), g_2(A_1, A_2) = 0, 0$$

Transforming inequalities into equalities

Arora (2012) Introduction to Optimum Design,
Elsevier: Theorem 4.6 (page 142)

Karush-Kuhn-Tucker Optimality Conditions

Let \mathbf{x}^* be a regular point of the feasible set that is a local minimum for $f(\mathbf{x})$, subject to $h_i(\mathbf{x}) = 0$; $i = 1$ to p ; $g_j(\mathbf{x}) \leq 0$; $j = 1$ to m . Then there exist Lagrange multipliers \mathbf{v}^* (a p -vector) and \mathbf{u}^* (an m -vector) such that the Lagrangian function is stationary with respect to x_j , v_i , u_j , and s_j at the point \mathbf{x}^* .

1. Lagrangian Function for the Problem

Written in the Standard Form:

$$\begin{aligned} L(\mathbf{x}, \mathbf{v}, \mathbf{u}, \mathbf{s}) &= f(\mathbf{x}) + \sum_{i=1}^p v_i h_i(\mathbf{x}) \\ &\quad + \sum_{j=1}^m u_j (g_j(\mathbf{x}) + s_j^2) \\ &= f(\mathbf{x}) + \mathbf{v}^T \mathbf{h}(\mathbf{x}) + \mathbf{u}^T (\mathbf{g}(\mathbf{x}) + \mathbf{s}^2) \end{aligned} \quad (4.46)$$

2. Gradient Conditions:

$$\begin{aligned} \frac{\partial L}{\partial x_k} &= \frac{\partial f}{\partial x_k} + \sum_{i=1}^p v_i^* \frac{\partial h_i}{\partial x_k} \\ &\quad + \sum_{j=1}^m u_j^* \frac{\partial g_j}{\partial x_k} = 0; \quad k = 1 \text{ to } n \end{aligned} \quad (4.47)$$

$$\frac{\partial L}{\partial v_i} = 0 \Rightarrow h_i(\mathbf{x}^*) = 0; \quad i = 1 \text{ to } p \quad (4.48)$$

$$\frac{\partial L}{\partial u_j} = 0 \Rightarrow (g_j(\mathbf{x}^*) + s_j^2) = 0; \quad j = 1 \text{ to } m \quad (4.49)$$

3. Feasibility Check for Inequalities:

$$s_j^2 \geq 0; \text{ or equivalently } g_j \leq 0; \quad j = 1 \text{ to } m \quad (4.50)$$

4. Switching Conditions:

$$\frac{\partial L}{\partial s_j} = 0 \Rightarrow 2u_j^* s_j = 0; \quad j = 1 \text{ to } m \quad (4.51)$$

5. Non-negativity of Lagrange Multipliers for Inequalities:

$$u_j^* \geq 0; \quad j = 1 \text{ to } m \quad (4.52)$$

6. Regularity Check: Gradients of the active constraints must be linearly independent. In such a case the Lagrange multipliers for the constraints are unique.

Penalty Methods

- Reformulate a constrained optimization problem as an unconstrained problem
- Penalization of the objective function value when constraints are violated

- Simple example

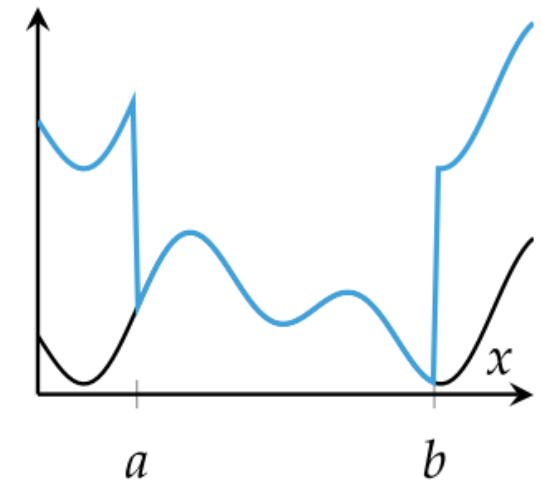
$$\begin{array}{ll}\underset{\mathbf{x}}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0}\end{array}$$



$$\underset{\mathbf{x}}{\text{minimize}} \quad f(\mathbf{x}) + \rho \cdot p_{\text{count}}(\mathbf{x})$$

Where $\rho > 0$ adjusts the penalty magnitude

$$p_{\text{count}}(\mathbf{x}) = \sum_i (g_i(\mathbf{x}) > 0) + \sum_j (h_j(\mathbf{x}) \neq 0)$$



— $f(x)$
— $f(x) + \rho p_{\text{count}}(x)$

Algorithm for Penalty Methods

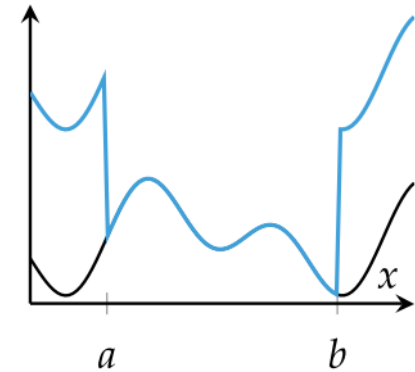
- Start with an initial point \mathbf{x} and a small value for ρ
- Solve the unconstrained optimization problem
- The resulting design point is then used as the starting point for another optimization with an increased penalty
- We continue with this procedure until the resulting point is feasible or a max number of iterations has been reached

```
function penalty_method(f, p, x, k_max; ρ=1, γ=2)
    for k in 1 : k_max
        x = minimize(x -> f(x) + ρ*p(x), x)
        ρ *= γ
        if p(x) == 0
            return x
        end
    end
    return x
end
```

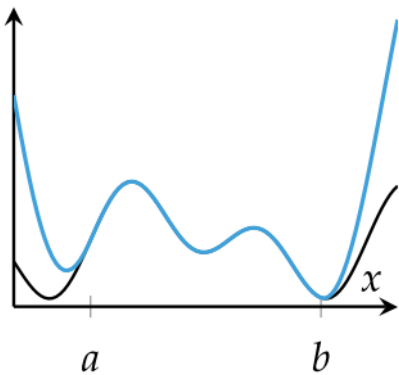
Julia code for penalty method

minimize can be any suitable
unconstrained minimization
method

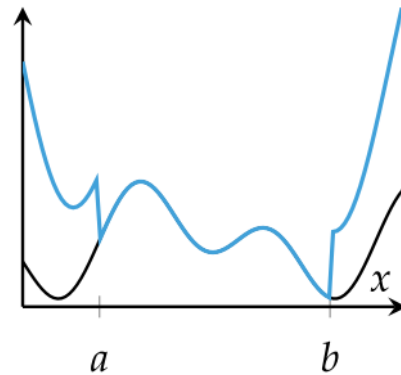
Penalty functions



— $f(x)$
— $f(x) + \rho p_{\text{count}}(x)$



— $f(x)$
— $f(x) + \rho p_{\text{quadratic}}(x)$



— $f(x)$
— $f(x) + p_{\text{mixed}}(x)$

Important notes

$$p_{\text{count}}(\mathbf{x}) = \sum_i (g_i(\mathbf{x}) > 0) + \sum_j (h_j(\mathbf{x}) \neq 0)$$

p_{count} preserves problem solution for large ρ values

- But introduces a sharp discontinuity
- Points not inside the feasible set lack gradient info

$$p_{\text{quadratic}}(\mathbf{x}) = \sum_i \max(g_i(\mathbf{x}), 0)^2 + \sum_j h_j(\mathbf{x})^2$$

$p_{\text{quadratic}}$ very small close to the constrain boundary

- May require ρ to approach infinity before cease violation

$$p_{\text{mixed}}(\mathbf{x}) = \rho_1 p_{\text{count}}(\mathbf{x}) + \rho_2 p_{\text{quadratic}}(\mathbf{x})$$

p_{mixed} clear boundary between the feasible/unfeasible regions

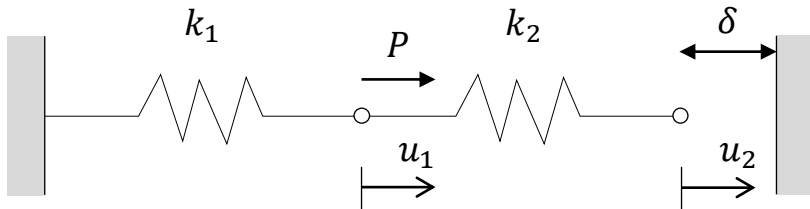
- Also provides gradient info to the solver

Penalty Methods

- Total Potential Energy

$$\Pi(\mathbf{u}) = \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u} - \mathbf{f}^T \mathbf{u}$$

Example: Contact Problem



- Contact condition

$$u_2 \leq \delta$$

- Optimization Problem

$$\begin{aligned} \underset{u_1, u_2}{\text{minimize}} \quad & \Pi(u_1, u_2) = \frac{1}{2} k_1 u_1^2 + \frac{1}{2} k_2 (u_2 - u_1)^2 - P u_1 \\ \text{s.t.} \quad & u_2 - \delta \leq 0 \end{aligned}$$

Contact Problem using Penalty Method

```
In [1]: using Optim, LinearAlgebra
```

```
In [2]: # Problem data
P = 200.
k1, k2, delta = 1000., 1000., 0.15

# Stiffness matrix and external force vector
K = [k1+k2 -k2; -k2 k2]
F = [P, 0.]

# Total Potential Energy and contact condition
Pi(u) = 1/2 * u * (K * u) - (F * u)
g(u) = u[2] - delta

# Penalty function (quadratic)
p(u) = max(g(u), 0)^2;
```

```
In [3]: # minimize: calling Optim unconstrained optimize function
function minimize(f, x0)

    res = optimize(f, x0)
    return Optim.minimizer(res)

end;
```

```
In [7]: # Penalty method code
function penalty_method(f, p, x, k_max; p=1, gamma=2)
    for k in 1 : k_max
        x = minimize(x -> f(x) + p * p(x), x)
        p *= gamma
        if p(x) == 0
            return x, k, p
        end
    end
    return x, k, p
end;
```

```
In [8]: # Eval contact problem using penalty method
uopt, k, p = penalty_method(Pi, p, [0., 0], 100)
println("u* : ", uopt, ", iterations: ", k, ", p : ", p)
```

```
u* : [0.17500468911768227, 0.14999999945312512], iterations: 46, p : 70368744177664
```



See 04_contactproblem-penalty.ipynb

Augmented Lagrange Method

- Adaption of penalty method for equality constraints
- Unlike the penalty method, works with smaller values of ρ

$$p_{\text{Lagrange}}(\mathbf{x}) = \frac{1}{2}\rho \sum_i (h_i(\mathbf{x}))^2 - \sum_i \lambda_i h_i(\mathbf{x})$$

where λ converges towards the Lagrange multiplier

- ρ still increases with each iteration
- and also, the linear penalty vector is updated according to

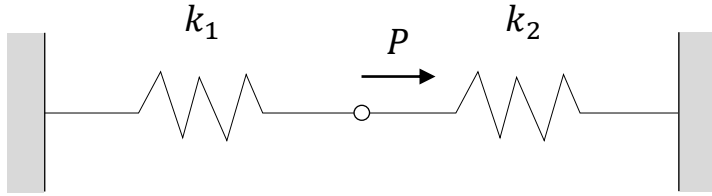
$$\lambda^{(k+1)} = \lambda^{(k)} - \rho \mathbf{h}(\mathbf{x})$$

Algorithm for Augmented Lagrange Method

```
function augmented_lagrange_method(f, h, x, k_max; p=1, γ=2)
    λ = zeros(length(h(x)))
    for k in 1 : k_max
        p = x -> f(x) + p/2*sum(h(x).^2) - λ·h(x)
        x = minimize(x -> f(x) + p(x), x)
        ρ *= γ
        λ -= ρ*h(x)
    end
    return x
end
```

Julia code for Augmented
Lagrange Method

Example: Complementary Energy with Augmented Lagrange Method



- Equilibrium condition (internal x external forces)

$$r_1 - r_2 = P$$

- Complementary Energy

$$\Pi_c(r_1, r_2) = \frac{1}{2} \frac{r_1^2}{k_1} + \frac{1}{2} \frac{r_2^2}{k_2}$$

- Optimization Problem

$$\begin{aligned} &\underset{r_1, r_2}{\text{minimize}} && \Pi_c(r_1, r_2) \\ &\text{s.t.} && r_1 - r_2 = P \end{aligned}$$

Given this optimization problem

Homework

- Consider:
 - $P = 10 \text{ kN}$
 - $k_1 = 100 \text{ kN/m}$
 - $k_2 = 200 \text{ kN/m}$
- Develop an Augmented Lagrange Method routine to solve the problem
 - Consider a break condition if $\Delta\lambda$ is smaller than a given tolerance
- Verify the results with the prementioned analytical solution
- Display the number of iterations and ρ value obtained

Interior Point Methods

- Also called Barrier Methods
- Ensure that each step is feasible
- Allows premature termination
 - to return a nearly optimal, feasible point

Barrier Functions, $p_{\text{barrier}}(\mathbf{x})$

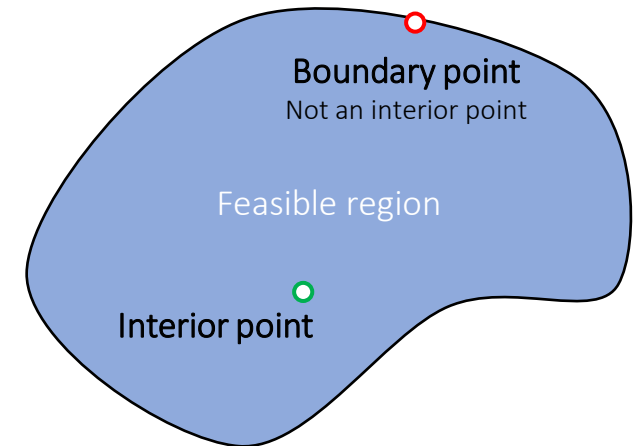
- Similar to penalties but must meet the following conditions:
 - Continuous
 - Non-negative
 - Approach ∞ as \mathbf{x} approaches any constraint boundary

Special care with line searches

- Line searches $f(\mathbf{x} + \alpha \mathbf{d})$ with $0 < \alpha < \alpha_u$
- α_u is the step to the nearest boundary

Initial guess

- Interior Point method requires a feasible point as an initial guess



Examples of barrier functions

- Inverse Barrier

$$p_{\text{barrier}}(\mathbf{x}) = -\sum_i \frac{1}{g_i(\mathbf{x})}$$

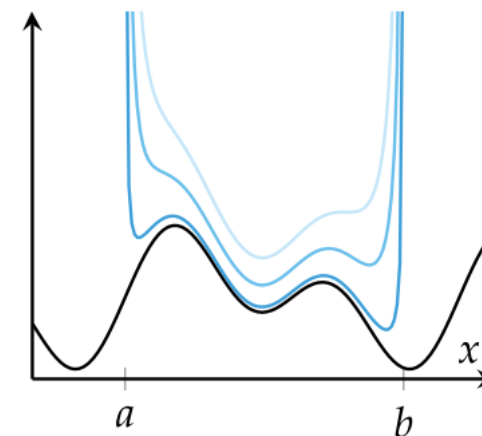
- Log Barrier

$$p_{\text{barrier}}(\mathbf{x}) = -\sum_i \begin{cases} \log(-g_i(\mathbf{x})) & \text{if } g_i(\mathbf{x}) \geq -1 \\ 0 & \text{otherwise} \end{cases}$$

- A problem with inequality constraints can be transformed into an unconstrained optimization problem

$$\underset{\mathbf{x}}{\text{minimize}} \ f(\mathbf{x}) + \frac{1}{\rho} p_{\text{barrier}}(\mathbf{x})$$

- Applying interior point method with an inverse barrier for minimizing f s.t. $a \leq x \leq b$



- $f(x)$
- $f(x) + p_{\text{barrier}}(x)$
- $f(x) + \frac{1}{2} p_{\text{barrier}}(x)$
- $f(x) + \frac{1}{10} p_{\text{barrier}}(x)$

Problem Formulation

- Optimization problem with a linear objective and constraints
- Called a **linear program** and its general form is

$$\begin{aligned} & \underset{\mathbf{x}}{\text{minimize}} && \mathbf{c}^\top \mathbf{x} \\ & \text{subject to} && \mathbf{A}_{\text{LE}} \mathbf{x} \leq \mathbf{b}_{\text{LE}} \\ & && \mathbf{A}_{\text{GE}} \mathbf{x} \geq \mathbf{b}_{\text{GE}} \\ & && \mathbf{A}_{\text{EQ}} \mathbf{x} = \mathbf{b}_{\text{EQ}} \end{aligned}$$

- Every general form linear program can be rewritten more compactly in standard form

$$\begin{aligned} & \underset{\mathbf{x}}{\text{minimize}} && \mathbf{c}^\top \mathbf{x} \\ & \text{subject to} && \mathbf{A} \mathbf{x} \leq \mathbf{b} \\ & && \mathbf{x} \geq \mathbf{0} \end{aligned}$$

$$\mathbf{A}_{\text{GE}} \mathbf{x} \geq \mathbf{b}_{\text{GE}} \rightarrow -\mathbf{A}_{\text{GE}} \mathbf{x} \leq -\mathbf{b}_{\text{GE}}$$

$$\mathbf{A}_{\text{EQ}} \mathbf{x} = \mathbf{b}_{\text{EQ}} \rightarrow \begin{cases} \mathbf{A}_{\text{EQ}} \mathbf{x} \leq \mathbf{b}_{\text{EQ}} \\ -\mathbf{A}_{\text{EQ}} \mathbf{x} \leq -\mathbf{b}_{\text{EQ}} \end{cases}$$

$$\mathbf{x} = \mathbf{x}^+ - \mathbf{x}^- \rightarrow \begin{cases} \mathbf{x}^+ \geq \mathbf{0} \\ \mathbf{x}^- \geq \mathbf{0} \end{cases}$$

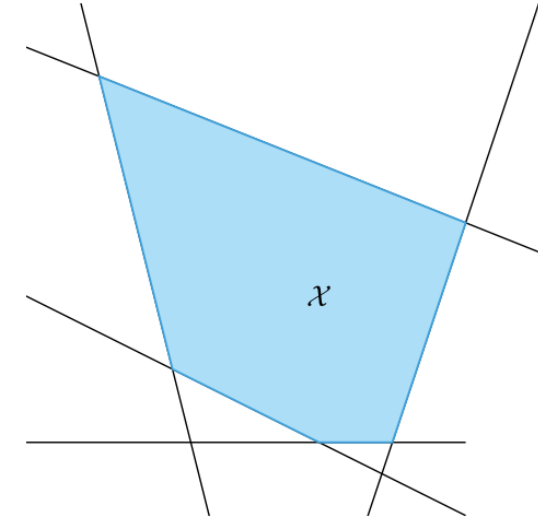


$$\begin{aligned} & \underset{\mathbf{x}^+, \mathbf{x}^-}{\text{minimize}} && \begin{bmatrix} \mathbf{c}^\top & -\mathbf{c}^\top \end{bmatrix} \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \end{bmatrix} \\ & \text{subject to} && \begin{bmatrix} \mathbf{A} & -\mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \end{bmatrix} \leq \mathbf{b} \\ & && \begin{bmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \end{bmatrix} \geq \mathbf{0} \end{aligned}$$

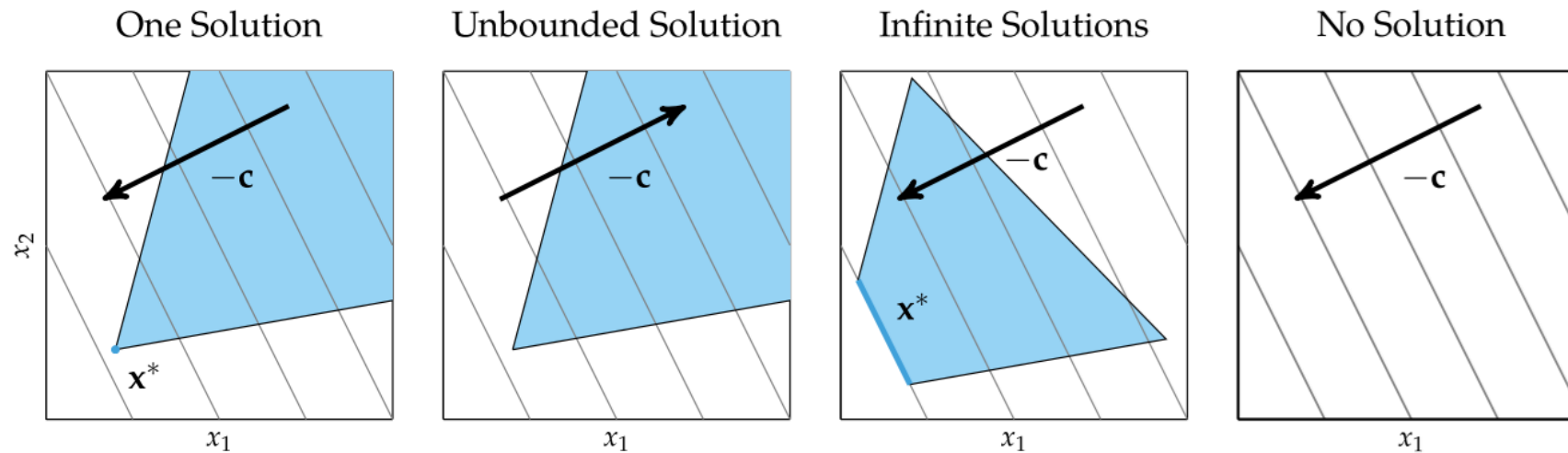
Problem Formulation

- Each inequality constraint defines a planar boundary of the feasible set (half-space)
- The set of inequality constraints define the intersection of multiple half-spaces (convex set)

$$\begin{aligned} & \underset{\mathbf{x}}{\text{minimize}} && \mathbf{c}^\top \mathbf{x} \\ & \text{subject to} && \mathbf{Ax} \leq \mathbf{b} \\ & && \mathbf{x} \geq \mathbf{0} \end{aligned}$$



- No unique solution exists if the problem is unbounded
- There are infinite solutions, or it is over-constrained



Problem Formulation

- Linear programs are often solved in equality form
- Feasible sets lie on hyperplanes

$$\begin{array}{ll}\underset{\mathbf{x}}{\text{minimize}} & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}\end{array}$$

- Any linear program in standard form can be transformed to equality form using slack variables \mathbf{s}

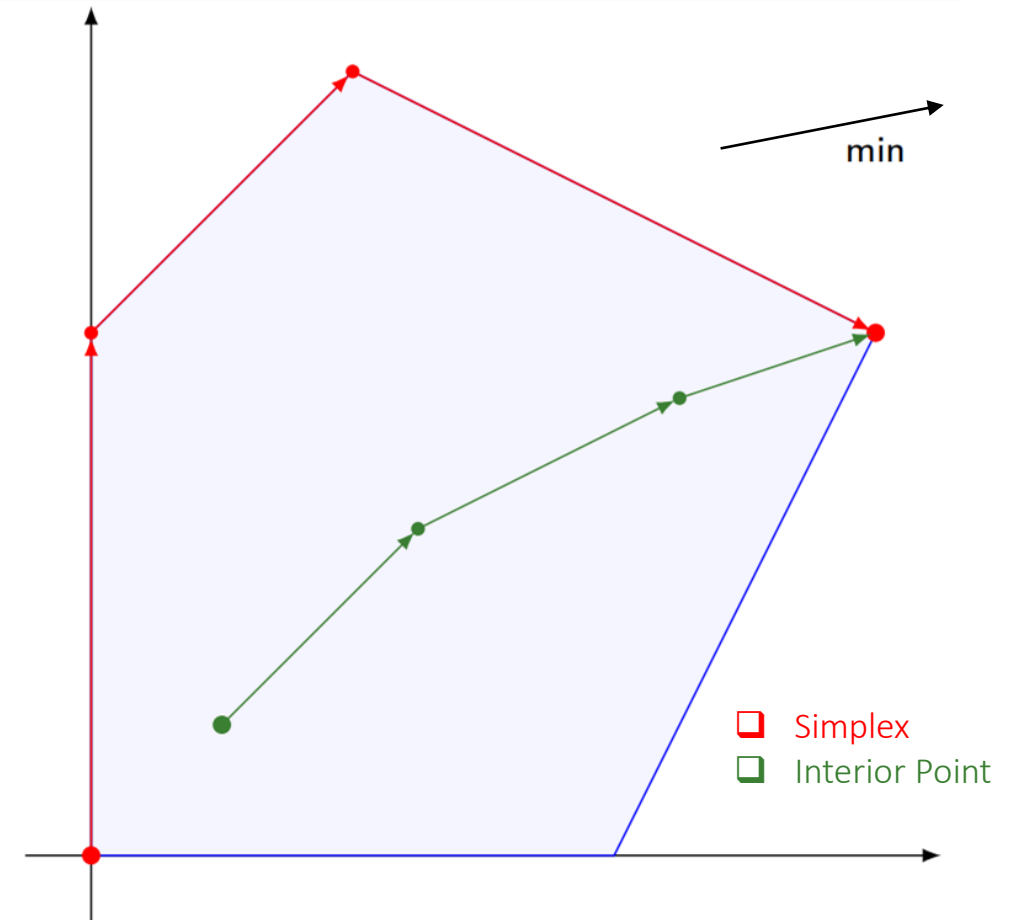
$$\mathbf{Ax} \leq \mathbf{b} \rightarrow \mathbf{Ax} + \mathbf{s} = \mathbf{b}, \quad \mathbf{s} \geq \mathbf{0}$$



$$\begin{array}{ll}\underset{\mathbf{x}, \mathbf{s}}{\text{minimize}} & \begin{bmatrix} \mathbf{c}^\top & \mathbf{0}^\top \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} \\ \text{subject to} & \begin{bmatrix} \mathbf{A} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} = \mathbf{b} \\ & \begin{bmatrix} \mathbf{x} \\ \mathbf{s} \end{bmatrix} \geq \mathbf{0}\end{array}$$

Linear Programming algorithms

- Solve any feasible and bounded linear program
- The feasible set of a linear program forms a polytope
- **Simplex**
 - Proposed by George B. **Dantzig** (1946-1947)
 - Moves between extreme vertices (basic points) until it finds an optimal vertex
 - Many cheap iterations
- **Interior Point**
 - Proposed by Narendra K. **Karmarkar** (1984)
 - A polynomial-time algorithm for linear programming that use interior point method
 - Few expensive iterations



Linear Constrained Optimization

Example

$$\begin{aligned} &\text{minimize} && -2x_1 - 3x_2 - 4x_3 \\ & && x \geq 0 \\ &\text{s. t.} && 3x_1 + 2x_2 + x_3 \leq 10 \\ & && 2x_1 + 5x_2 + 3x_3 \leq 15 \\ & && x_1, x_2, x_3 \geq 0 \end{aligned}$$

- GLPK Optimizer (Simplex-based)
 - Obtained the exact solution
 - $x = [0 \quad 0 \quad 5]$
- Tulip Optimizer (Interior Point-based)
 - Obtained approximated solution
 - $x = [9.50 \cdot 10^{-10} \quad 3.116 \cdot 10^{-11} \quad 4.999]$

Linear Program Example

```
In [1]: using JuMP, GLPK, Tulip
```

Build the model

```
In [2]: model = Model()

@variable(model, x[1:3] >= 0)

c = [-2, -3, -4]
@objective(model, Min, c' * x)

A = [3 2 1; 2 5 3]; b = [10, 15]
@constraint(model, A * x .<= b)

print(model)

Min -2 x[1] - 3 x[2] - 4 x[3]
Subject to
 3 x[1] + 2 x[2] + x[3] <= 10.0
 2 x[1] + 5 x[2] + 3 x[3] <= 15.0
 x[1] >= 0.0
 x[2] >= 0.0
 x[3] >= 0.0
```



See 05_LPexample.ipynb

Define optimize function

```
In [3]: function optimize(model, optimizer)

    # set optimizer and optimize
    println(optimizer)
    set_optimizer(model, optimizer)
    @time optimize!(model)

    # check the termination and primal status to see if we have a solution
    println("Termination status : ", termination_status(model))
    println("Primal status      : ", primal_status(model))

    # print the solution
    println("f* : ", objective_value(model))
    println("x* : ", [value(x[i]) for i=1:3])

end;
```

Perform optimizations

```
In [4]: optimize(model, GLPK.Optimizer) # simplex-based
```

```
GLPK.Optimizer
7.029700 seconds (28.18 M allocations: 1.416 GiB, 7.91% gc time)
Termination status : OPTIMAL
Primal status      : FEASIBLE_POINT
f* : -20.0
x* : [0.0, 0.0, 5.0]
```

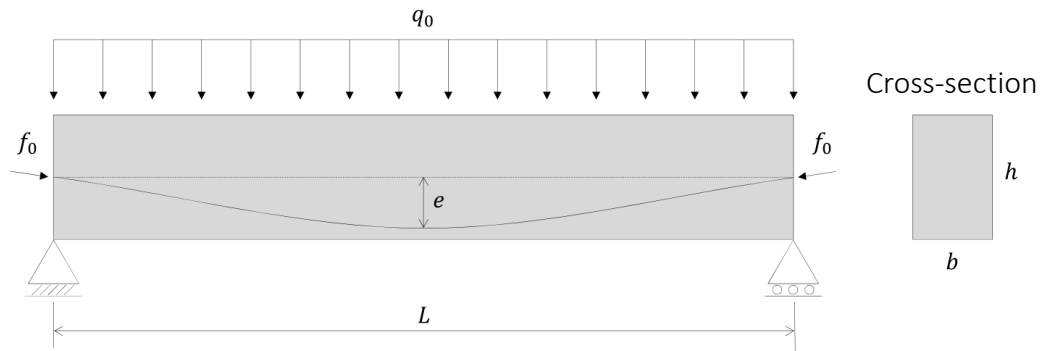
```
In [5]: optimize(model, Tulip.Optimizer) # interior point-based
```

```
Tulip.Optimizer
12.081357 seconds (33.00 M allocations: 1.572 GiB, 4.82% gc time)
Termination status : OPTIMAL
Primal status      : FEASIBLE_POINT
f* : -19.999999997829246
x* : [9.849747875647242e-10, 3.115766236747574e-11, 4.999999998941455]
```

Example: Prestressed beam problem

Kirsch (1993) Structural Optimization, Springer-Verlag: Adapted from Example 4.6 (page 203)

Minimize the prestress force



- Solving as a conventional optimization problem

$$\begin{aligned} \min_{f_0, e} \quad & f_0 \\ \text{s.t.} \quad & \sigma_L \leq -\frac{f_0}{bh} \pm \frac{6}{bh^2} \left[\frac{q_0 L^2}{8} - f_0 e \right] \leq \sigma_U, \\ & \delta_L \leq \frac{5L^4}{32Ebh^3} \left[q_0 - \frac{8}{L^2} f_0 e \right] \leq \delta_U, \\ & e_L \leq e \leq e_U, \\ & f_0 \geq 0. \end{aligned}$$



See 06_prestressedbeam.ipynb

Prestressed Beam Problem ¶

Input data

```
In [1]: using JuMP, Ipopt, GLPK
```

```
In [2]: # Geometry
L, b, h = 12., 0.1, 1. # m

# Load and material
q0 = 20. # kN/m
E = 30e+6 # kPa

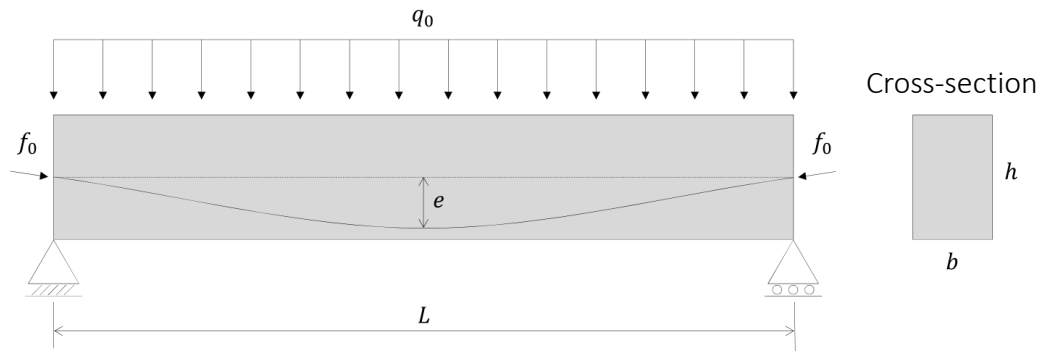
# Bounds
σmin, σmax = -15e+3, 1e+3 # kPa
δmin, δmax = -L/300, L/300 # m
f0max = 5e+3 # kN
cover = 0.05 # m
emax = (h-2*cover)/2;

# Beam maximum stress (top/bottom) and deflection
σt(f0, e) = -f0 / (b * h) - 6 / (b * h^2) * (q0 / 8 * L^2 - f0 * e)
σb(f0, e) = -f0 / (b * h) + 6 / (b * h^2) * (q0 / 8 * L^2 - f0 * e)
δ(f0, e) = 5 / (32E * b * h^3) * L^4 * (q0 - 8 / L * L / L * f0 * e);
```

Example: Prestressed beam problem

Kirsch (1993) Structural Optimization, Springer-Verlag: Adapted from Example 4.6 (page 203)

Minimize the prestress force



■ Solving as a conventional optimization problem

$$\begin{aligned}
 \min_{f_0, e} \quad & f_0 \\
 \text{s.t.} \quad & \sigma_L \leq -\frac{f_0}{bh} \pm \frac{6}{bh^2} \left[\frac{q_0 L^2}{8} - f_0 e \right] \leq \sigma_U, \\
 & \delta_L \leq \frac{5L^4}{32Ebh^3} \left[q_0 - \frac{8}{L^2} f_0 e \right] \leq \delta_U, \\
 & e_L \leq e \leq e_U, \\
 & f_0 \geq 0.
 \end{aligned}$$

Solving as a conventional problem

```

In [3]: model = Model(Ipopt.Optimizer)
        set_optimizer_attribute(model, "print_level", 0)

        @variable(model, f0 >= 0)
        @variable(model, 0 <= e <= emax)

        @objective(model, Min, f0)

        @constraint(model, sigma_min <= ob(f0, e) <= sigma_max)
        @constraint(model, sigma_min <= ot(f0, e) <= sigma_max)
        @constraint(model, delta_min <= delta(f0, e) <= delta_max)

        println(model)
        optimize!(model)

        println("Termination status: ", termination_status(model))
        println("Primal status: ", primal_status(model))

        println("f* : ", objective_value(model))
        println("f0* : ", value(f0), ", e* : ", value(e))
    
```

```

Min f0
Subject to
 -60 f0*e - 10 f0 in [-36600.0, -20600.0]
 60 f0*e - 10 f0 in [6600.0, 22600.0]
-5.9999999999999995e-5 f0*e in [-0.0616, 0.0184]
 f0 >= 0.0
 e >= 0.0
 e <= 0.45
    
```

```

*****
This program contains Ipopt, a library for large-scale nonlinear optimization.
Ipopt is released as open source code under the Eclipse Public License (EPL).
For more information visit http://projects.coin-or.org/Ipopt
*****
    
```

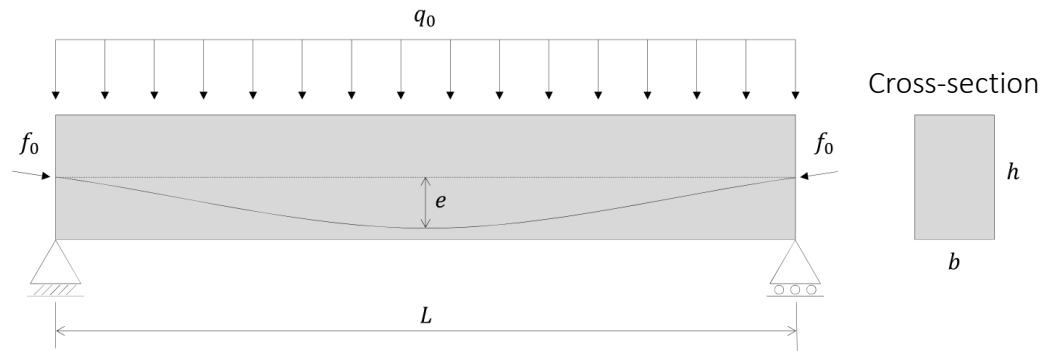
```

Termination status: LOCALLY_SOLVED
Primal status: FEASIBLE_POINT
f* : 556.7567421657133
f0* : 556.7567421657133, e* : 0.45
    
```


Example: Prestressed beam problem

Kirsch (1993) Structural Optimization, Springer-Verlag: Adapted from Example 4.6 (page 203)

Minimize the prestress force



■ Solving as a linear program

Transformation
 $m = f_0 e$

$$\begin{aligned} \min_{f_0, e} \quad & f_0 \\ \text{s.t.} \quad & \sigma_L \leq -\frac{f_0}{bh} \pm \frac{6}{bh^2} \left[\frac{q_0 L^2}{8} - m \right] \leq \sigma_U, \\ & \delta_L \leq \frac{5L^4}{32Ebh^3} \left[q_0 - \frac{8}{L^2} m \right] \leq \delta_U, \\ & f_0 e_L \leq m \leq f_0 e_U \\ & f_0 \geq 0. \end{aligned}$$

Solving as a linear program

using $m = f_0 \cdot e$

```
In [4]: model = Model(GLPK.Optimizer)

@variable(model, f0 >= 0)
@variable(model, m >= 0) # m = f0 * e

@objective(model, Min, f0)

# Beam maximum stress (top/bottom) and deflection
ot(f0, m) = -f0 / (b * h) - 6 / (b * h^2) * (q0 / 8 * L^2 - m)
ob(f0, m) = -f0 / (b * h) + 6 / (b * h^2) * (q0 / 8 * L^2 - m)
δ(f0, m) = 5 / (32E * b * h^3) * L^4 * (q0 - 8 / L * L * m);

@constraint(model, σmin <= ob(f0, m) <= σmax)
@constraint(model, σmin <= ot(f0, m) <= σmax)
@constraint(model, δmin <= δ(f0, m) <= δmax)

# Additional constraint
@constraint(model, m <= f0 * emax)

println(model)
optimize!(model)

println("Termination status: ", termination_status(model))
println("Primal status: ", primal_status(model))

println(" f* : ", objective_value(model))
println("f0* : ", value(f0), ", m* : ", value(m))
println(" e* : ", value(m)/value(f0))
```

```
Min f0
Subject to
-0.45 f0 + m <= 0.0
-10 f0 - 60 m in [-36600.0, -20600.0]
-10 f0 + 60 m in [6600.0, 22600.0]
-5.9999999999999995e-5 m in [-0.0616, 0.0184]
f0 >= 0.0
m >= 0.0
```

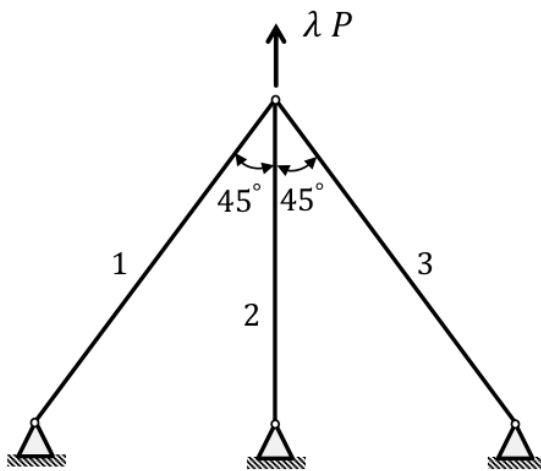
```
Termination status: OPTIMAL
Primal status: FEASIBLE_POINT
f* : 556.7567567567568
f0* : 556.7567567567568, m* : 250.54054054054055
e* : 0.45
```

Example: Truss with collapse load

Homework

Given the following optimization problem

- Reduce design variables to 2 and format as a linear program
- Solve it using a linear program routine
- Verify the results using a graphical optimization procedure



Optimization problem

$$\begin{aligned} &\underset{\lambda, r_1, r_2, r_3}{\text{maximize}} && \lambda \\ &\text{s.t.} && \mathbf{B}^T \mathbf{r} = \lambda \mathbf{f} \\ & && \mathbf{r} \leq \bar{\mathbf{r}} \\ & && \lambda \geq 0 \end{aligned}$$

where:

\mathbf{r} : Internal forces
 \mathbf{f} : Reference external forces
 \mathbf{B}^T : Equilibrium matrix
 λ : Load factor

$\bar{\mathbf{r}}$: Tensile strength
 $\bar{\mathbf{r}} = r_{\max} [\alpha \quad \beta \quad \gamma]^T$

Questions? Comments?

Constrained Optimization
Ricardo A. Fernandes

This presentation and its complementary files

<https://github.com/ricardoaf/conopt>

New to Julia?

<https://github.com/ricardoaf/juliafirststeps>

