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

Contents

Introduction

General optimization problem equation

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

Unconstrained problems:

- Feasible set $\boldsymbol{\chi}$ is $\mathbb{R}^n$

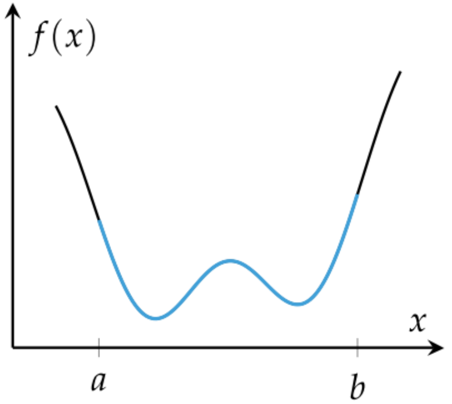
Constrained problems:

- 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

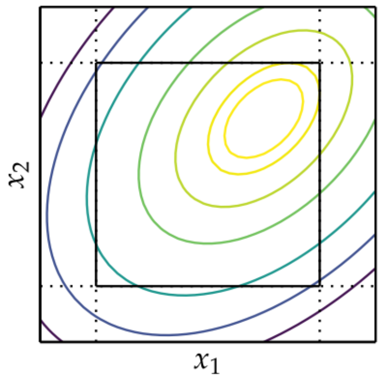


$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & x \in [a, b] \end{aligned}$$

Inequality constraints

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

Multivariate problems



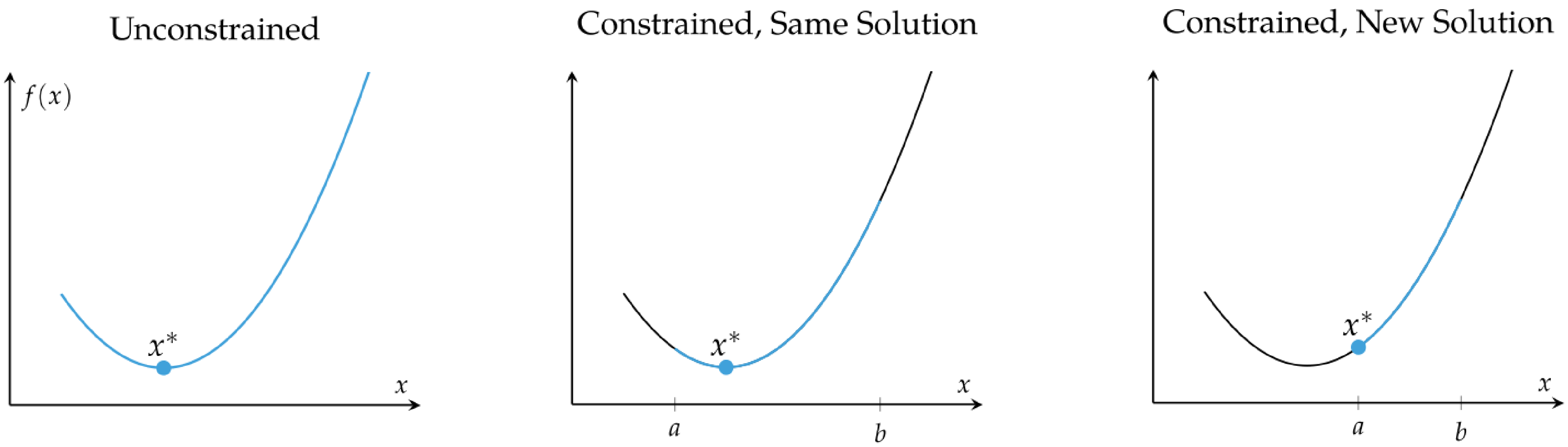
Solution must lie within a hyperrectangle

Introduction

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 constrains can affect the solution, or not



Constraint Types

The two constraint types

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

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:

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

Equality constraint as two inequality constraints

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

Transformations: Removing Constraints

Problem transformation

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

Eliminating design variables

- Consider the equality constraint

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

- x_n can 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

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

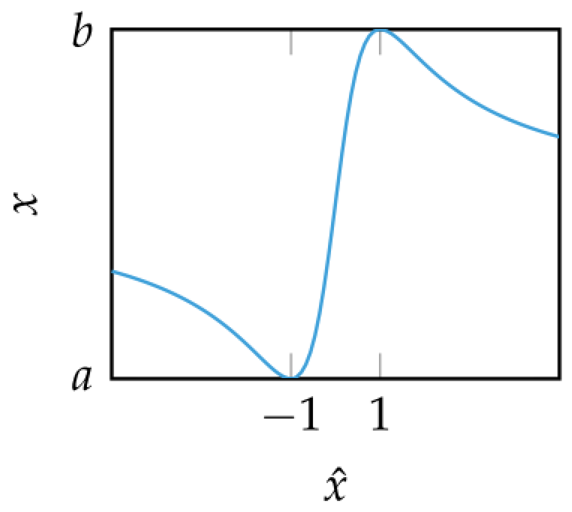
into

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

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 $a \leq x \leq b$

Transformations: Removing Constraints

Example: Removing bound constraints

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

- Consider the following optimization problem

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

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.;
```

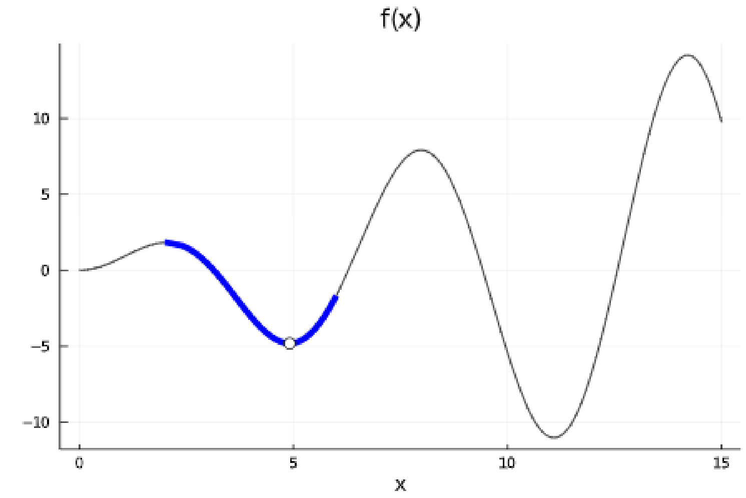
Constrained Optimization

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]:



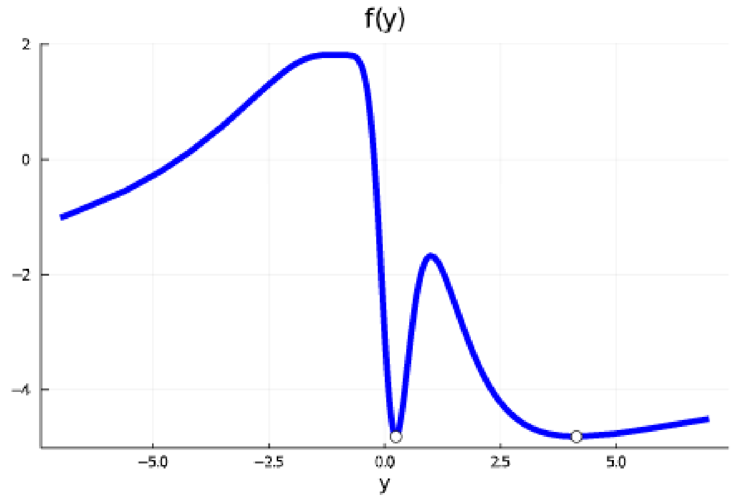
Unconstrained Optimization (transformed problem)

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=LBFGS())

    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://julianlsolvers.github.io/Optim.jl/stable/

See 01_alg4opt170.ipynb

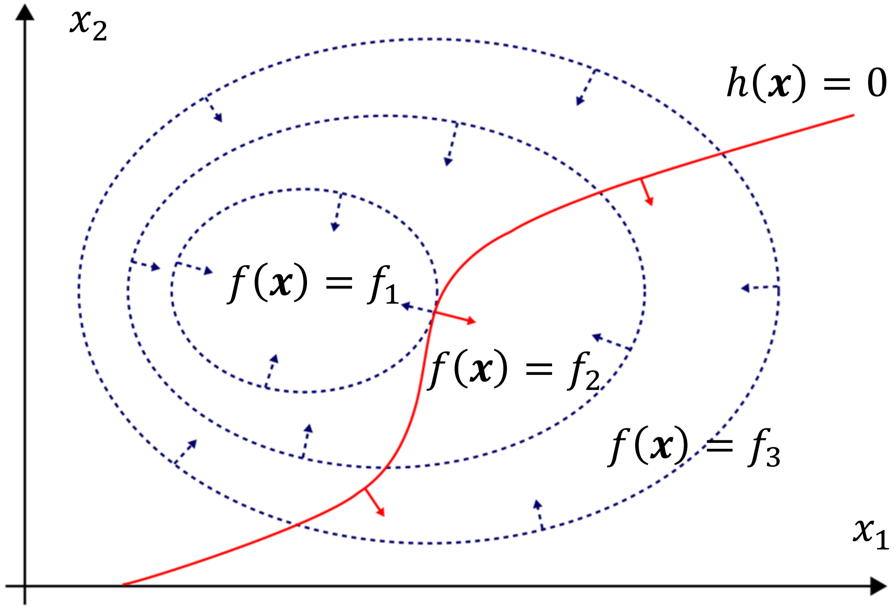
Lagrange Multipliers

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

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



- 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 $\boldsymbol{x}^*$ lies where a contour line of f is align with the contour line $h = 0$

Lagrange Multipliers

Example: Aligned gradients

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

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

Lagrange Multiplier Motivation

Otimization 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

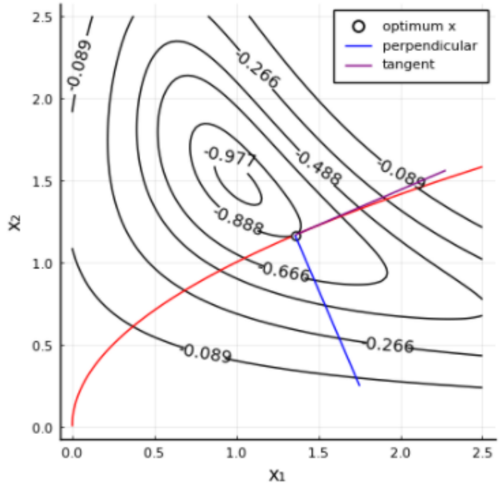
In [4]:
```
# Objective and constraint gradients
∇f = Calculus.gradient(f)
∇h = Calculus.gradient(h);
```

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")
```

Out[6]:



See 02_lagmul.ipynb

Lagrange Multipliers

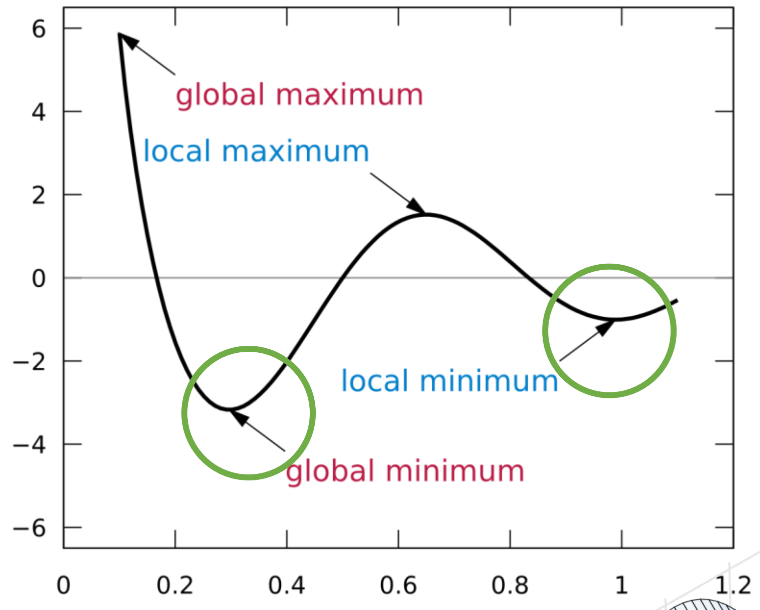
Method of Lagrange Multipliers

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

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

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

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

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

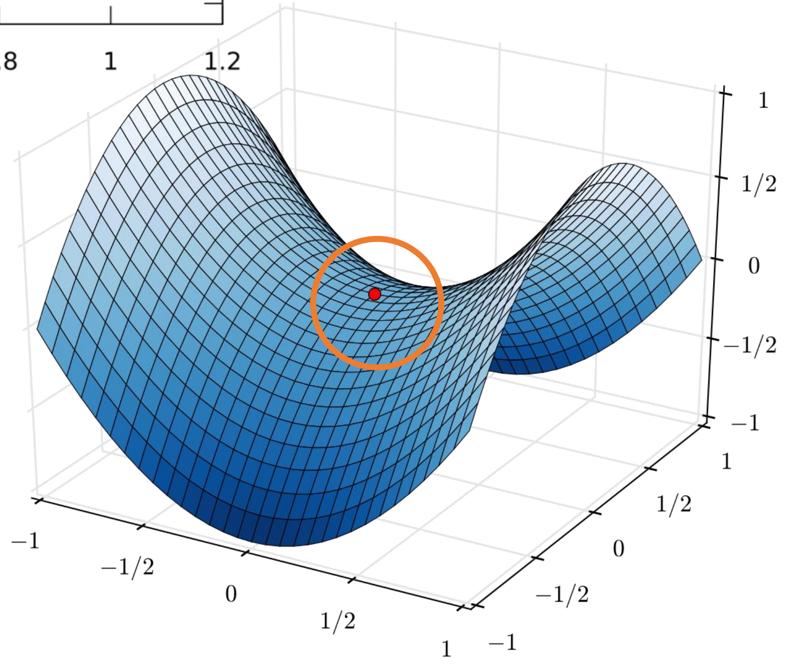
- $\nabla_x \mathcal{L} = \boldsymbol{0} \implies \nabla f(\boldsymbol{x}) = \lambda \nabla h(\boldsymbol{x})$
- $\nabla_\lambda \mathcal{L} = 0 \implies h(\boldsymbol{x}) = 0$

Necessary optimally conditions (not sufficient)

For convex problems,
Necessary conditions = sufficient conditions

Any solution is considered a *critical point*

- Local/global minimum
- Saddle points



Lagrange Multipliers

Consider the same last example

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

- 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

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_i \lambda_i\, h_i(\boldsymbol{x})$$

$$= f(\boldsymbol{x})\ - \boldsymbol{\lambda}^T\, \boldsymbol{h}(\boldsymbol{x})$$

Lagrange Multipliers

Example: Minimization of Complementary Energy

- **Optimization problem**

$$\underset{\boldsymbol{r}}{\text{minimize}} \quad \Pi_c(\boldsymbol{r})$$
$$\text{s.t.} \quad \boldsymbol{B}^T\,\boldsymbol{r} = \boldsymbol{f}$$

  - Complementary Energy

$$\Pi_c(\boldsymbol{r}) = \frac{1}{2}\,\boldsymbol{r}^T\,\boldsymbol{F}\,\boldsymbol{r}$$

where:

$\boldsymbol{r}$: Internal forces of the bars

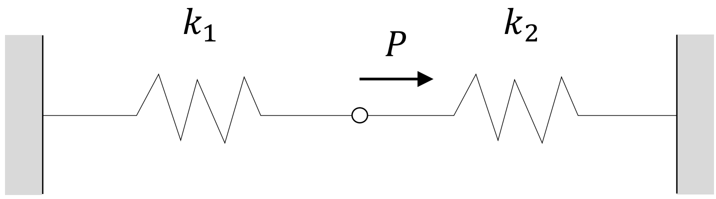
$\boldsymbol{f}$: Nodal external forces

$\boldsymbol{B}^{\boldsymbol{T}}$: Equilibrium matrix

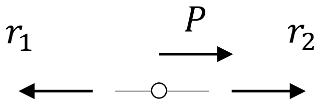
$\boldsymbol{F}$: Flexibility matrix

Lagrange Multipliers

Example: Minimization of Complementary Energy



- Equilibrium condition (internal x external forces)



$$r_2 - r_1 = P$$

- 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

$$\underset{r_1, r_2}{\text{minimize}} \quad \Pi_c(r_1, r_2)$$
$$\text{s.t.} \quad r_2 - r_1 = P$$

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

$$r2 := solve(h(r_1, r_2) = 0, r_2) = r_1 - P$$

$$\Pi_c(r_1, r2) = \frac{r_1^{\,2}}{2\,k_1} + \frac{(r_1 - P)^2}{2\,k_2}$$

$$r1 := solve(diff(\Pi_c(r_1, r2), r_1) = 0, r_1) = \frac{P\,k_1}{k_2 + k_1}$$

$$r2 := solve(h(r1, r_2) = 0, r_2) = -\frac{P\,k_2}{k_2 + k_1}$$

using Lagrange multiplier

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

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

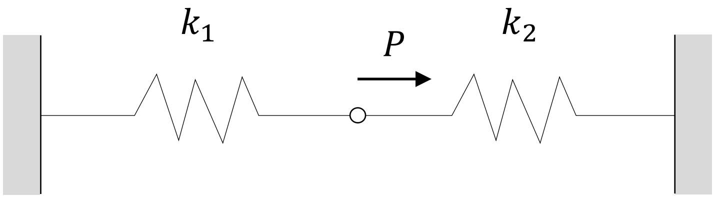
$$eq_2 := diff(L(r_1, r_2, \lambda), r_2) = 0 = \frac{r_2}{k_2} + \lambda = 0$$

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

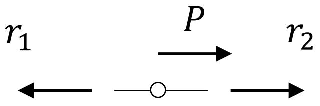
$$solve(\{eq_1, eq_2, eq_3\}, \{r_1, r_2, \lambda\}) = \left\{\lambda = \frac{P}{k_2 + k_1}, r_1 = \frac{P\,k_1}{k_2 + k_1}, r_2 = -\frac{P\,k_2}{k_2 + k_1}\right\}$$

Lagrange Multipliers

Example: Minimization of Complementary Energy



- Equilibrium condition (internal x external forces)



$$r_2 - r_1 = P$$

- 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

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

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))
```

See 03_complementaryenergy.ipynb

```
Min 0.005 r₁² + 0.0025 r₂²
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
```

About JuMP solvers

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:

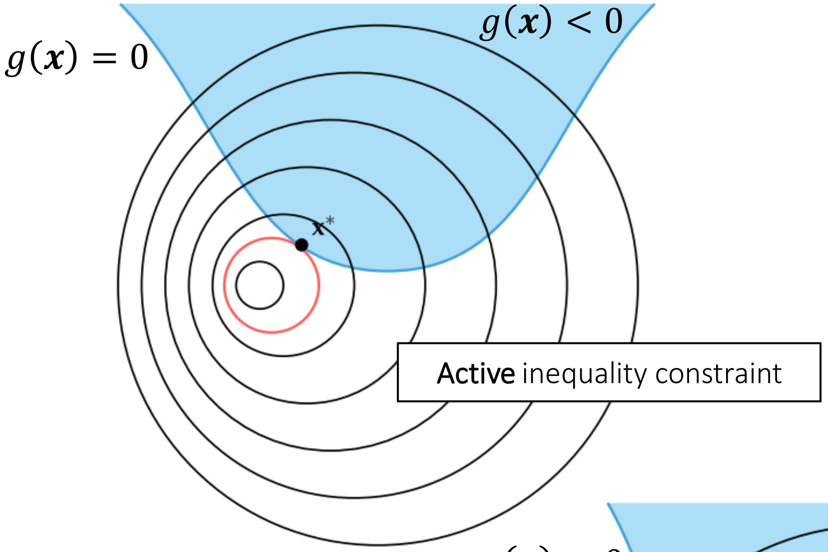
- LP = Linear programming
- QP = Quadratic programming
- SOCP = Second-order conic programming (including problems with convex quadratic constraints and/or objective)
- MILP = Mixed-integer linear programming
- NLP = Nonlinear programming
- MINLP = Mixed-integer nonlinear programming
- SDP = Semidefinite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

Inequality Constraints

Inequality constraints

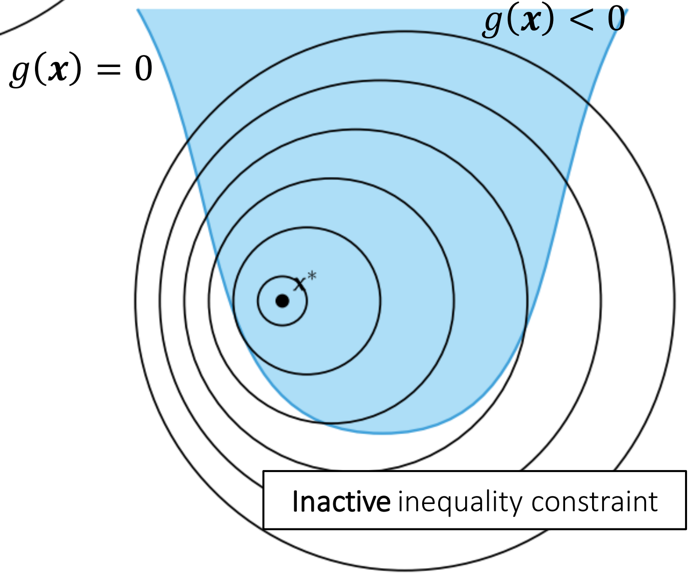
Consider a problem with a single inequality constraint

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



- If the solution <u>lies</u> at the constraint boundary,
  - the Lagrange condition holds for some constant μ
  - the constraint is considered **active**

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



- If the solution <u>does not lie</u> 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 $\mu = 0$

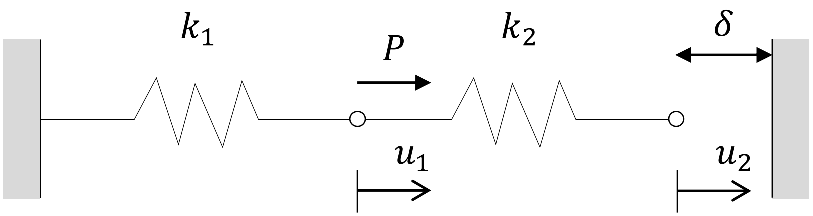
- The Lagrangian of the problem is $\mathcal{L}(\boldsymbol{x}, \mu) = f(\boldsymbol{x}) + \mu g(\boldsymbol{x})$
- In order to penalize the objective, for inequalities $\mu \geq 0$

Inequality Constraints

- Total Potential Energy

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

Example: Contact Problem



- Contact condition

$$u_2 \leq \delta$$

- Optimization Problem

$$\underset{u_1, u_2}{\text{minimize}} \quad \Pi(u_1, u_2) = \frac{1}{2}k_1 u_1^2 + \frac{1}{2}k_2(u_2 - u_1)^2 - P\,u_1$$

$$\text{s.t.} \quad u_2 - \delta \leq 0$$

Contact Problem

$restart : with(LinearAlgebra) :$

Set stiffness matrix

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

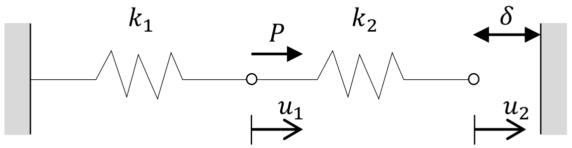
Approach #1: Solving linear system, $K\,u = f$

Unconstrained problem

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

$u := LinearSolve(K, F) :$

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



Constrained problem

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

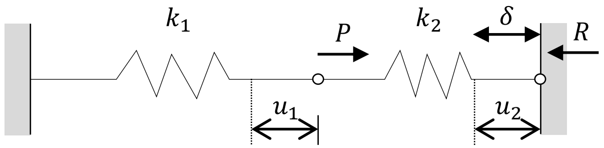
Apply partition method for solving the unknowns

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

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

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

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

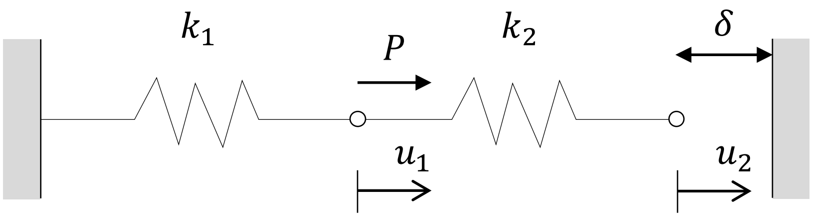


Inequality Constraints

- Total Potential Energy

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

Example: Contact Problem



- Contact condition

$$u_2 \le \delta$$

- Optimization Problem

$$\underset{u_1, u_2}{\text{minimize}} \quad \Pi(u_1, u_2) = \frac{1}{2}k_1 u_1^2 + \frac{1}{2}k_2(u_2 - u_1)^2 - P\,u_1$$

$$\text{s.t.} \quad u_2 - \delta \le 0$$

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

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

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

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

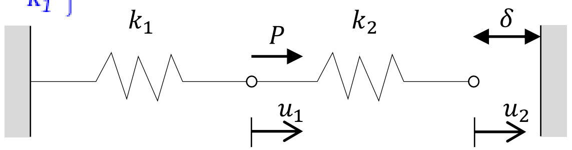
Unconstrained problem

Set Lagrange multiplier to zero (remove constraint from Lagrangian)

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

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

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



Constrained problem

Constraint g must be zero

$$u2 := solve(g(u_1, u_2) = 0, u_2) = \delta$$

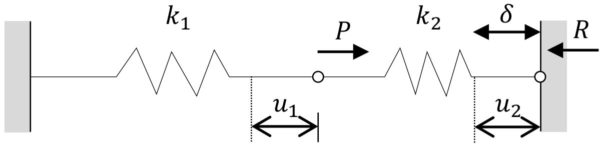
$$eq_1 := \mathrm{D}[1](L)(u_1, u2, \mu) = k_1 u_1 - k_2(\delta - u_1) - P$$

$$eq_2 := \mathrm{D}[2](L)(u_1, u2, \mu) = k_2(\delta - u_1) + \mu$$

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

$$\mu = R$$

$$assign(\%) :$$



Lagrange multiplier and reaction force

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

$$simplify(R - \mu) = 0$$

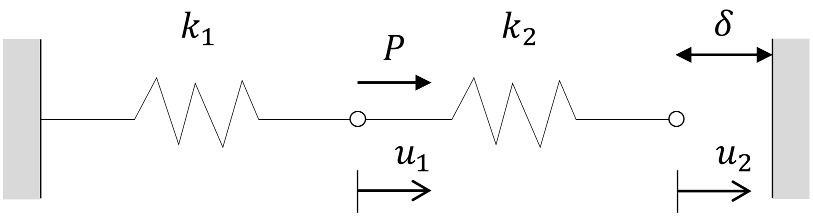
For inequality constraints, since $P - \delta \cdot k_1 \ge 0$, or reaction force due to contact is always non-negative, **lagrange multiplier must always be non-negative** too!

Inequality Constraints

- Total Potential Energy

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

Example: Contact Problem



- Contact condition

$$u_2 \leq \delta$$

- Optimization Problem

$$\underset{u_1,u_2}{\text{minimize}} \quad \Pi(u_1,u_2) = \frac{1}{2}k_1u_1^2 + \frac{1}{2}k_2(u_2-u_1)^2 - P\,u_1$$

$$\text{s.t.} \quad u_2 - \delta \leq 0$$

Given this optimization problem **Homework**

- A) Consider:
  - $P = 100\ \text{kN}$
  - $k_1 = k_2 = 1000\ \text{kN/m}$
  - $\delta = 0.15\ \text{m}$

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

- B) Repeat A) but now with $P = 200\ \text{kN}$

Inequality Constraints

Optimizing the problem

Requires finding critical points $\boldsymbol{x}^*$ 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
  - $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$
  - $\mu = 0$

$$\nabla f(\mathbf{x}^*) + \mu \nabla g(\mathbf{x}^*) = \mathbf{0}$$

KKT Conditions

Generalizing for multiple equality and inequality constraints

- **Feasibility**
  All constraints satisfied
  $$\mathbf{g}(\mathbf{x}^*) \leq \mathbf{0}$$
  $$\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$$
- **Dual feasibility**
  Penalization is toward feasibility
  $$\boldsymbol{\mu} \geq 0$$
- **Complementary slackness**
  Either $\mu_i = 0$ or $g_i(\boldsymbol{x}) = 0$
  $$\mu_i \, g_i(\boldsymbol{x}^*) = 0$$
- **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

Inequality Constraints

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

Objective function and inequality constraints

$f := (x_1, x_2) \to x_1{}^2 + x_2{}^2 - 2 \cdot x_1 - 2 \cdot x_2 + 2 :$

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

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

Generalized Lagrangian

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

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

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

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

KKT conditions (Complementary slackness and dual feasibility)

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

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

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

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

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

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

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

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

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

Feasible solution: Case 4

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

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

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

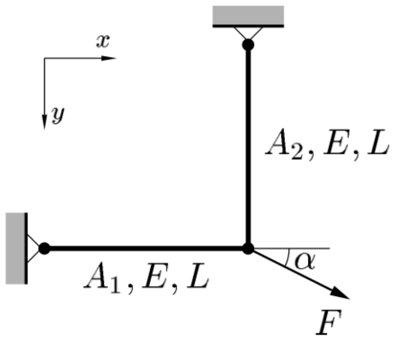
$f(x_1, x_2) = \frac{2}{9}$

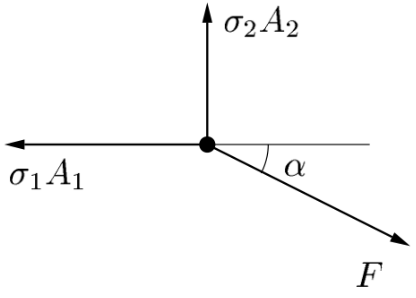
$g_1(x_1, x_2), g_2(x_1, x_2) = 0, 0$

Inequality Constraints

Example: Weight minimization with stress constraints

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





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

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

Objective function and inequality constraints

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

$g_1 := (A_1, A_2) \to \dfrac{F \cdot \cos(\alpha)}{\sigma 0} - A_1 :$

$g_2 := (A_1, A_2) \to \dfrac{F \cdot \sin(\alpha)}{\sigma 0} - A_2 :$

Generalized Lagrangian

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

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

KKT conditions (Stationarity and feasibility)

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

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

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

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

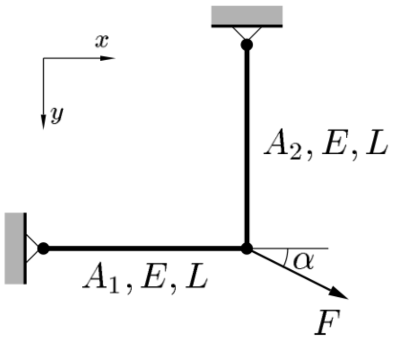
Equations and unknowns

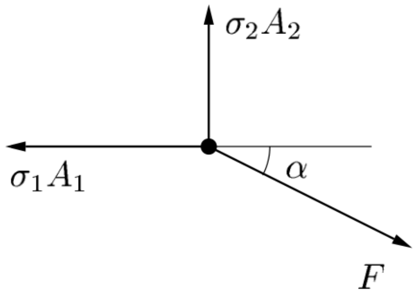
$eq := unapply(\{eq_1, eq_2, eq_3, eq_4\}, A_1, A_2, \mu_1, \mu_2, G_1, G_2) :$

Inequality Constraints

Example: Weight minimization with stress constraints

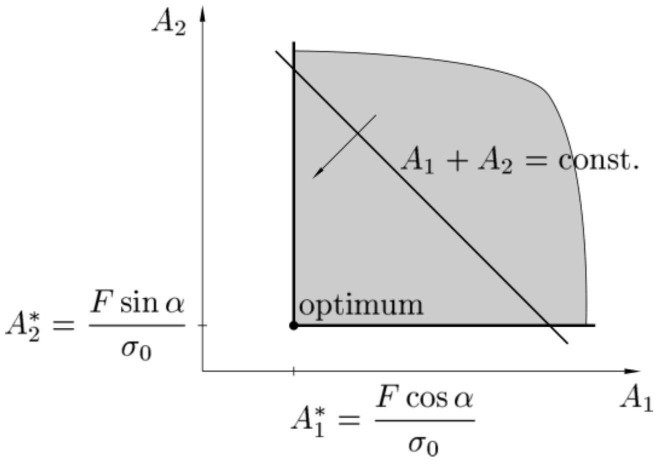
Christensen & Klarbring (2009) An Introduction to Structural Optimization, Springer: Section 2.1 (page 10)





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

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



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 $G_1, G_2 \leq 0$

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

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

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

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

$$eq(A_1, A_2, case_1) = \left\{1 = 0, \frac{F\cos(\alpha)}{\sigma 0} - A_1 = G_1, \frac{F\sin(\alpha)}{\sigma 0} - A_2 = G_2\right\}$$

$$eq(A_1, A_2, case_2) = \left\{1 = 0, 1 - \mu_2 = 0, \frac{F\cos(\alpha)}{\sigma 0} - A_1 = G_1, \frac{F\sin(\alpha)}{\sigma 0} - A_2 = 0\right\}$$

$$eq(A_1, A_2, case_3) = \left\{1 = 0, 1 - \mu_1 = 0, \frac{F\cos(\alpha)}{\sigma 0} - A_1 = 0, \frac{F\sin(\alpha)}{\sigma 0} - A_2 = G_2\right\}$$

$$eq(A_1, A_2, case_4) = \left\{1 - \mu_1 = 0, 1 - \mu_2 = 0, \frac{F\cos(\alpha)}{\sigma 0} - A_1 = 0, \frac{F\sin(\alpha)}{\sigma 0} - A_2 = 0\right\}$$

$$solve(eq(A_1, A_2, case_4), \{A_1, A_2, vars_4\}) = \left\{A_1 = \frac{F\cos(\alpha)}{\sigma 0}, A_2 = \frac{F\sin(\alpha)}{\sigma 0}, \mu_1 = 1, \mu_2 = 1\right\}$$

Feasible solution: Case 4

$assign(solve(eq(A_1, A_2, case_4), \{A_1, A_2, vars_4\}))$

$$A_1, A_2 = \frac{F\cos(\alpha)}{\sigma 0}, \frac{F\sin(\alpha)}{\sigma 0}$$

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

$$f(A_1, A_2) = \frac{F\cos(\alpha)}{\sigma 0} + \frac{F\sin(\alpha)}{\sigma 0}$$

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

Inequality Constraints

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; $g_j(\mathbf{x}) \le 0;\ j = 1$ 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:*

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

2. *Gradient Conditions:*

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

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

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

3. *Feasibility Check for Inequalities:*

$$s_j^2 \ge 0; \text{ or equivalently } g_j \le 0; \quad j = 1 \text{ to } m \tag{4.50}$$

4. *Switching Conditions:*

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

5. *Non-negativity of Lagrange Multipliers for Inequalities:*

$$u_j^* \ge 0; \quad j = 1 \text{ to } m \tag{4.52}$$

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

Penalty Methods

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

- Simple example

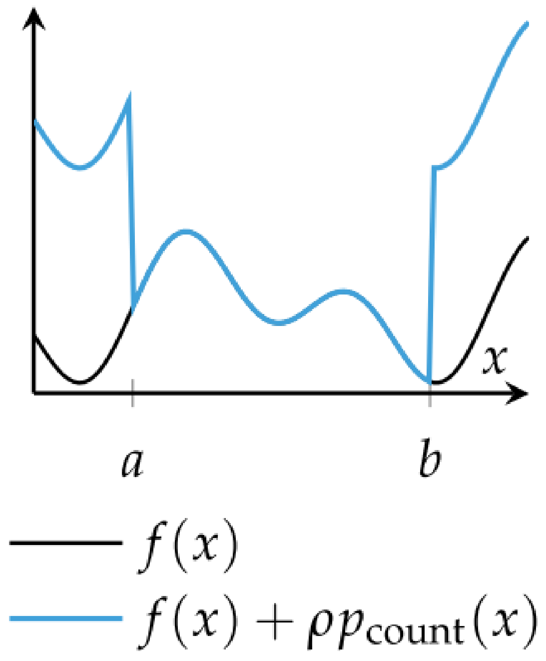
$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}) = \mathbf{0} \end{aligned}$$

$$\downarrow$$

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

Where $\rho > 0$ adjusts the penalty magnitude

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



Penalty Methods

Algorithm for Penalty Methods

- Start with an initial point $\boldsymbol{x}$ 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

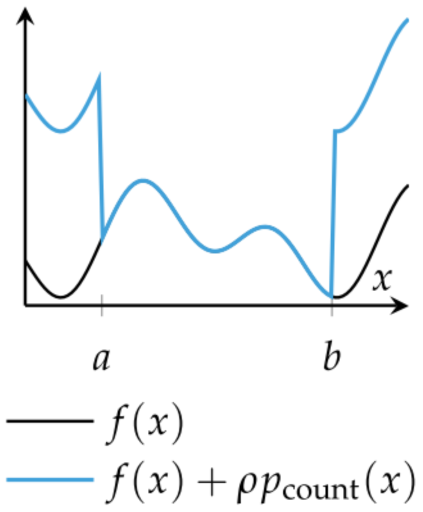
```
function penalty_method(f, p, x, k_max; ρ=1, γ=2)
    for k in 1 : k_max
        x = minimize(x -> f(x) + ρ*p(x), x)
        ρ *= γ
        if p(x) == 0
            return x
        end
    end
    return x
end
```

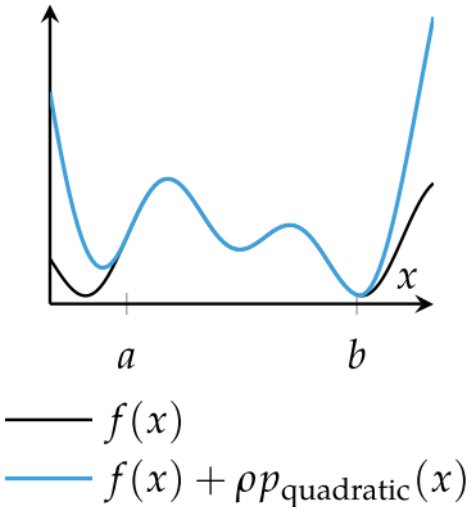
Julia code for penalty method

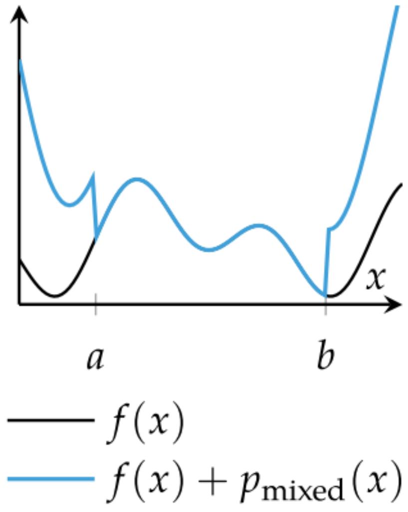
minimize can be any suitable unconstrained minimization method

Penalty Methods

Penalty functions







Important notes

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

p_{count} preserves problem solution for large ρ values

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

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

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

- May require ρ to approach infinity before cease violation

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

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

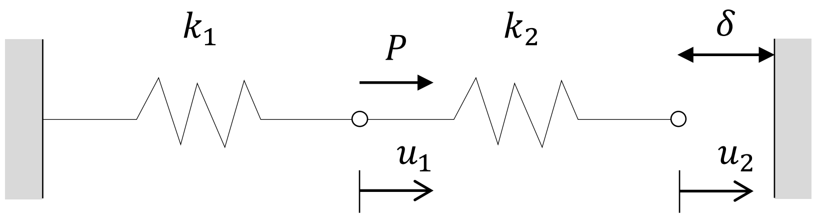
- Also provides gradient info to the solver

Penalty Methods

- Total Potential Energy

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

Example: Contact Problem



- Contact condition

$$u_2 \leq \delta$$

- Optimization Problem

$$\underset{u_1, u_2}{\text{minimize}} \quad \Pi(u_1, u_2) = \frac{1}{2} k_1 u_1^2 + \frac{1}{2} k_2 (u_2 - u_1)^2 - P\, u_1$$

$$\text{s.t.} \quad u_2 - \delta \leq 0$$

Contact Problem using Penalty Method

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

In [2]:
```
# Problem data
P = 200.
k₁, k₂, δ = 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
Π(u) = 1/2 * u ⋅ (K * u) - (F ⋅ u)
g(u) = u[2] - δ

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

In [3]:
```
# minimize: calling Optim unconstained 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; 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, k, ρ
        end
    end
    return x, k, ρ
end;
```

See 04_contactproblem-penalty.ipynb

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

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

Augmented Lagrange Method

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

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

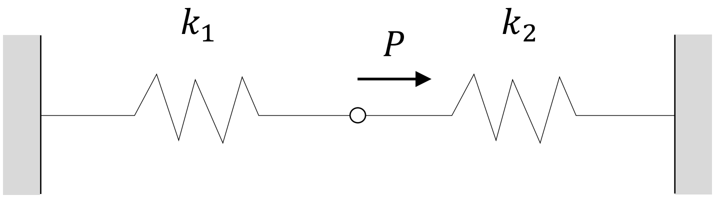
Algorithm for Augmented Lagrange Method

```
function augmented_lagrange_method(f, h, x, k_max; ρ=1, γ=2)
    λ = zeros(length(h(x)))
    for k in 1 : k_max
        p = x -> f(x) + ρ/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
```

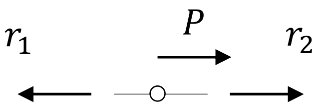
Julia code for Augmented Lagrange Method

Augmented Lagrange Method

Example: Complementary Energy with Augmented Lagrange Method



- Equilibrium condition (internal x external forces)



$$r_2 - r_1 = P$$

- 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

$$\underset{r_1, r_2}{\text{minimize}} \quad \Pi_c(r_1, r_2)$$
$$\text{s.t.} \quad r_2 - r_1 = P$$

Given this optimization problem

Homework

- Consider:
  - $P = 10\ \mathrm{kN}$
  - $k_1 = 100\ \mathrm{kN/m}$
  - $k_2 = 200\ \mathrm{kN/m}$

- 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

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, $p_{\text{barrier}}(\boldsymbol{x})$

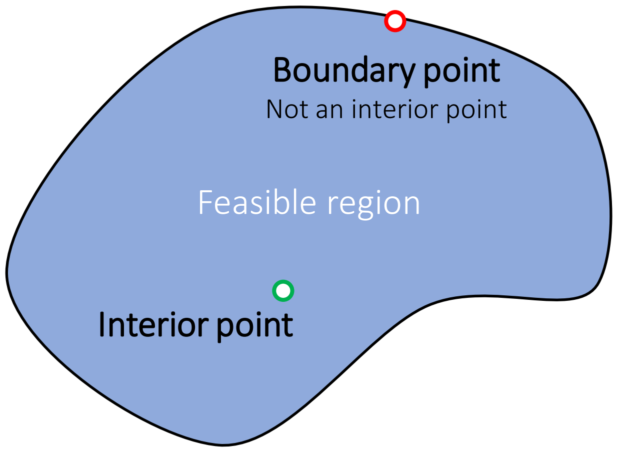
- Similar to penalties but must meet the following conditions:
  - Continuous
  - Non-negative
  - Approach ∞ as $\boldsymbol{x}$ approaches any constraint boundary

Special care with line searches

- Line searches $f(\boldsymbol{x} + \alpha \boldsymbol{d})$ 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



Interior Point Methods

Examples of barrier functions

- Inverse Barrier

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

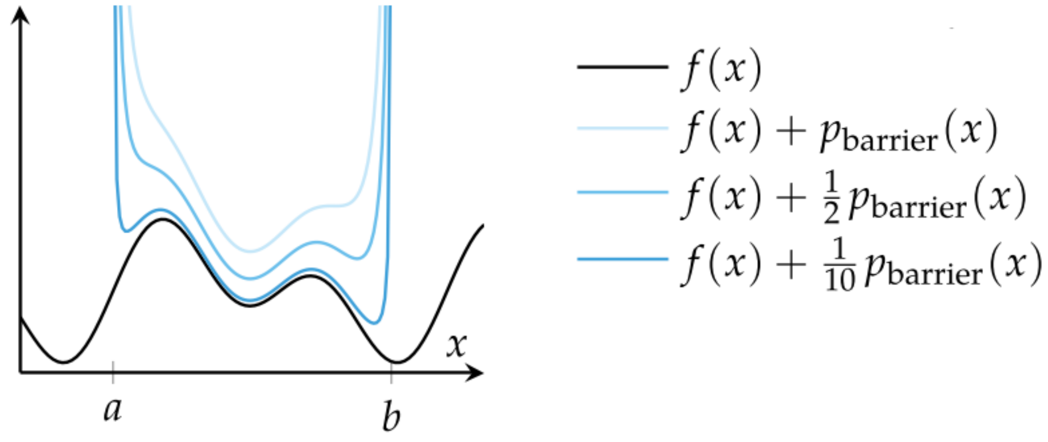
- Log Barrier

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

- 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. $a \leq x \leq b$



Questions? Comments?

This presentation and its complementary files

https://github.com/ricardoaf/conopt

New to Julia?

https://github.com/ricardoaf/juliafirststeps