Smooth **unconstrained** minimization

The unconstrained problem: minimize $f(x)$

Assumptions:

- f convex, twice continuously differentiable (hence $\mathbf{dom}\, f$ open)
- we assume optimal value $p^\star = \inf_x f(x)$ is attained (and finite)

Unconstrained minimization methods:

- produce sequence of points $x^{(k)} \in \mathbf{dom}\, f$, $k = 0, 1, \ldots$ with

$$f(x^{(k)}) \to p^\star$$

- can be interpreted as iterative methods for solving optimality condition

$$\nabla f(x^\star) = 0$$

Why do we need this?

$\nabla f(x^\star) = 0$ is in general a set of nonlinear equations. Usually no analytical solution!

Initial point and sublevel set

The unconstrained methods we consider require a starting point $x^{(0)}$ such that:

- $x^{(0)} \in \mathbf{dom}\, f$
- sublevel set $S = \{x \mid f(x) \leq f(x^{(0)})\}$ is closed

2nd condition is hard to verify,

- true if $\mathbf{dom}\, f = \mathbf{R}^n$
- true if $f(x) \to \infty$ as $x \to \mathbf{bd}\,\mathbf{dom}\, f$

Descent methods

Produce a sequence of points with decreasing cost value

$$x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)} \quad \text{with } f(x^{(k+1)}) < f(x^{(k)})$$

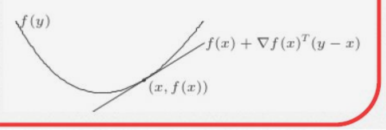
- other notations: $x^+ = x + t\Delta x$, $x := x + t\Delta x$
- Δx is the *step*, or *search direction*; t is the *step size*, or *step length*
- from convexity, $f(x^+) < f(x)$ implies $\nabla f(x)^T \Delta x < 0$

(*i.e.*, Δx is a *descent direction*) (t was assumed 1 here)

First order condition: for f differentiable (i.e. its gradient exists at each point of **dom** f, which is open) f is convex if and only if **dom** f is convex and

$$f(y) \geq f(x) + \nabla f(x)^\mathsf{T}(y - x)$$

holds for all $x, y \in$ **dom** f.



General descent method.

given a starting point $x \in$ **dom** f.
repeat
1. Determine a descent direction Δx.
2. *Line search*. Choose a step size $t > 0$.
3. *Update*. $x := x + t\Delta x$.

until stopping criterion is satisfied.

Line search types

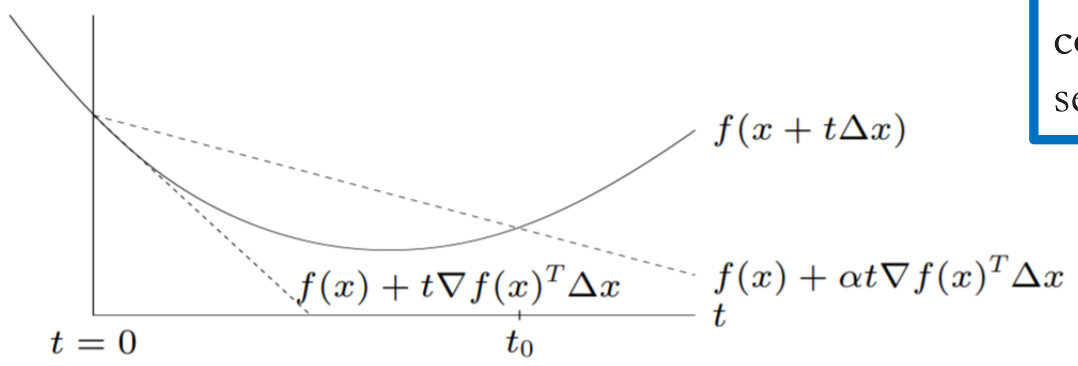
exact line search: $t = \operatorname{argmin}_{t>0} f(x + t\Delta x)$

backtracking line search (with parameters $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$)

- starting at $t = 1$, repeat $t := \beta t$ until

$$f(x + t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$$

- graphical interpretation: backtrack until $t \leq t_0$



Exact or backtracking?

Usually undesirable to devote substantial resources to finding the optimal value of t.

The resources to find a more precise minimum for one particular direction could be used to identify a better search direction.

Gradient descent method

general descent method with $\Delta x = -\nabla f(x)$ ⇒ $\nabla f(x)^T \Delta x < 0$ always satisfied!

---

given a starting point $x \in \mathbf{dom}\, f$.
repeat
1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

- very simple, but often very slow; rarely used in practice
- stopping criterion usually of the form $\|\nabla f(x)\|_2 \leq \epsilon$

Example

- Quadratic problem in two dimensions

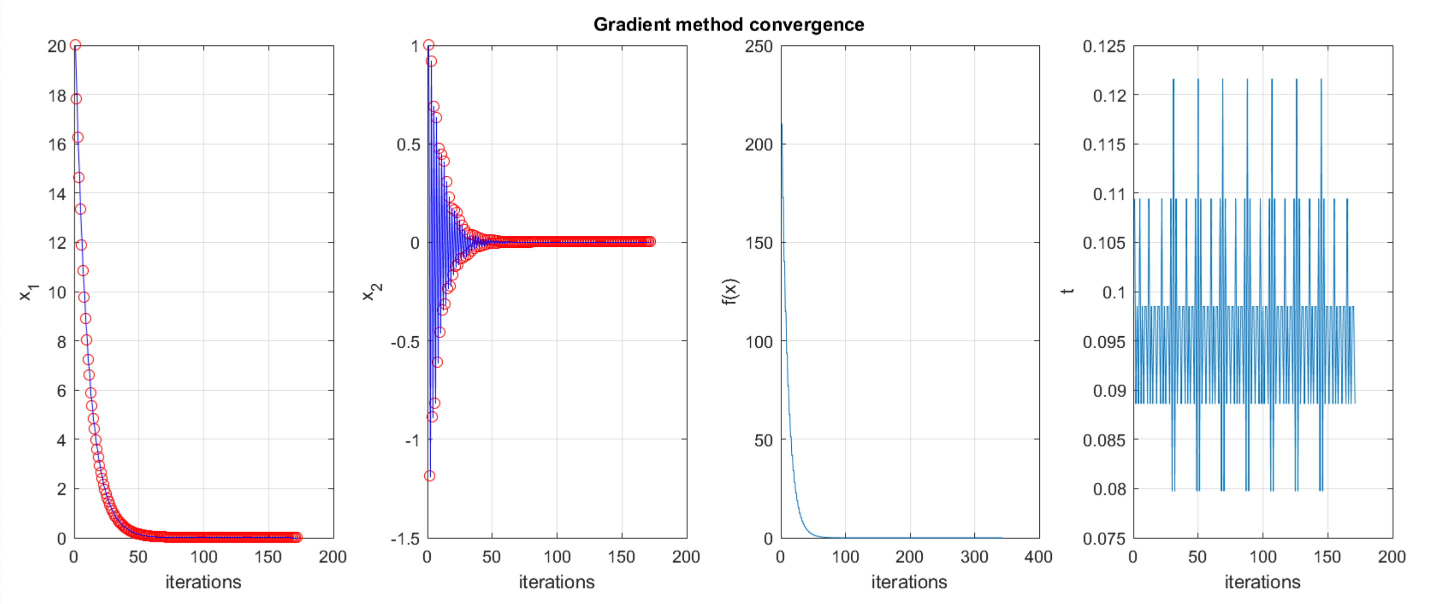
$$f(x) = \frac{1}{2}(x_1^2 + 20x_2^2)$$

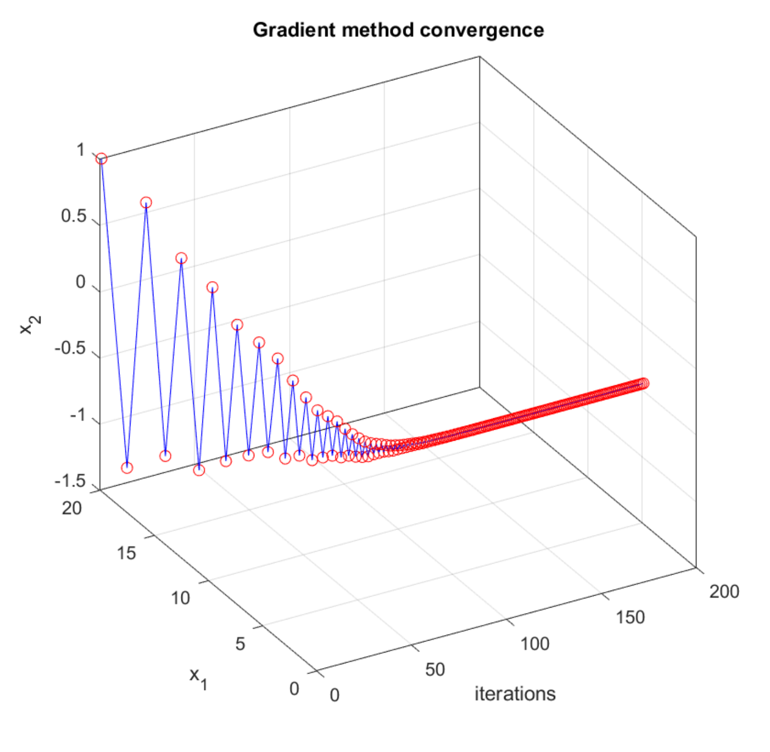
- Let do solve few steps of the gradient method on paper.
- Let implement the method in Matlab.

Example – Matlab code

```
clear all;close all;
alpha=0.4;beta=0.9;
eps=1e-6;
syms x1 x2
f=0.5*(x1^2+20*x2^2);
Jac=jacobian(f)';
x=[20;1];
xtot=x;
fxtot=[];
tchosen=[];
flag1=0;
while flag1==0
    x1=x(1);
    x2=x(2);
    fx=eval(f);
    fxtot=[fxtot fx];
     Jac_x=eval(Jac);
     if norm(Jac_x,2)<=eps
        flag1=1;
    end
    if flag1==1
        break;
    end
     Dx=-eval(Jac);
     t=1;
     flag=0;
     while flag==0
         xtest=x+t*Dx;
         x1=xtest(1);
         x2=xtest(2);
         fxtDx=eval(f);
         if fxtDx<=fx+alpha*t*Jac_x'*Dx
             flag=1;
         else
             t=beta*t;
           end
      end
      tchosen=[tchosen t];
      x=x+t*Dx;
      xtot=[xtot x];
      fxtot=[fxtot fx];
end
figure
subplot(1,4,1)
plot(xtot(1,:),'ro')
hold on
plot(xtot(1,:),'b')
xlabel('iterations')
ylabel('x_1');grid on;box on;
subplot(1,4,2)
plot(xtot(2,:),'ro')
hold on
plot(xtot(2,:),'b')
xlabel('iterations')
ylabel('x_2')
grid on;box on;
subplot(1,4,3)
plot(fxtot)
xlabel('iterations')
ylabel('f(x)')
grid on;box on
subplot(1,4,4)
plot(tchosen)
xlabel('iterations')
grid on;box on
```

Example – Results





Other methods

Other descent methods exist:

- steepest descent method
- the Newton's method
- …

Not part of this course!

What do we do in the presence of **constraints**?

We will discuss methods for solving constrained LP programs.