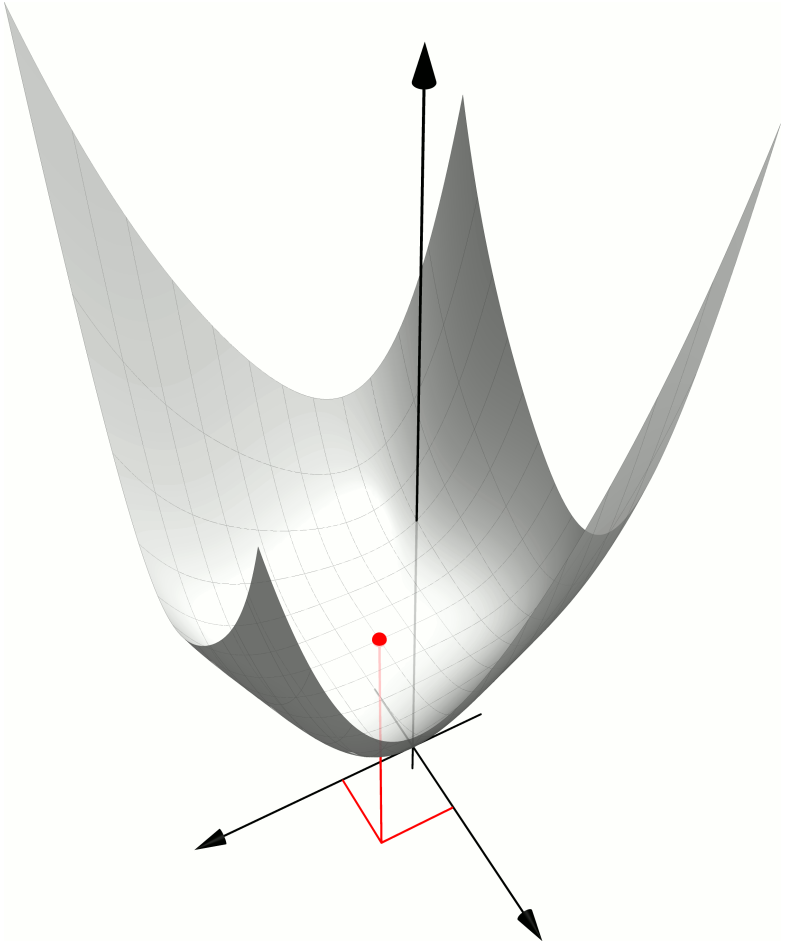
Modern Methods of Optimization

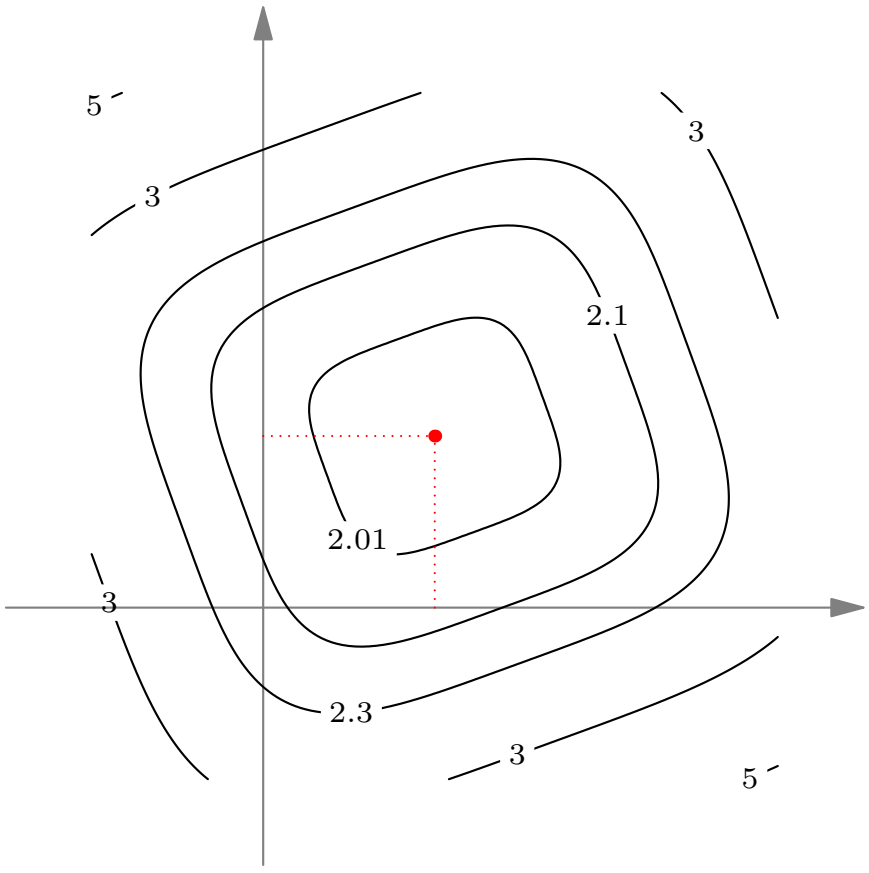
Lecture 4: Unconstrained optimization in nD ($n = 2$)

Faculty of Civil Engineering / CTU in Prague

Visualizing (unconstrained) minimization in nD

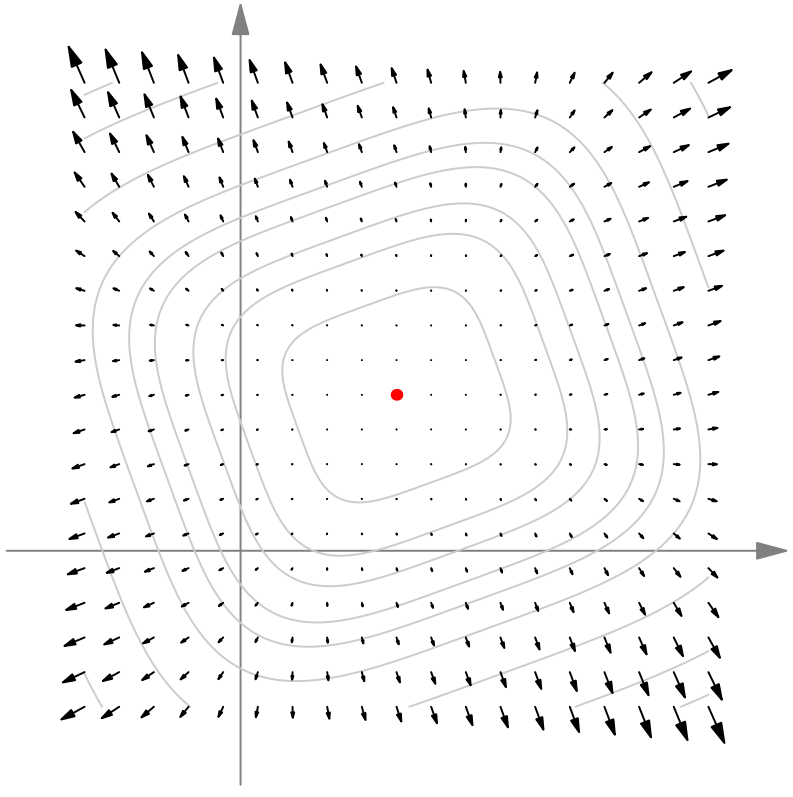
$$\min\ f(x_1, x_2)$$





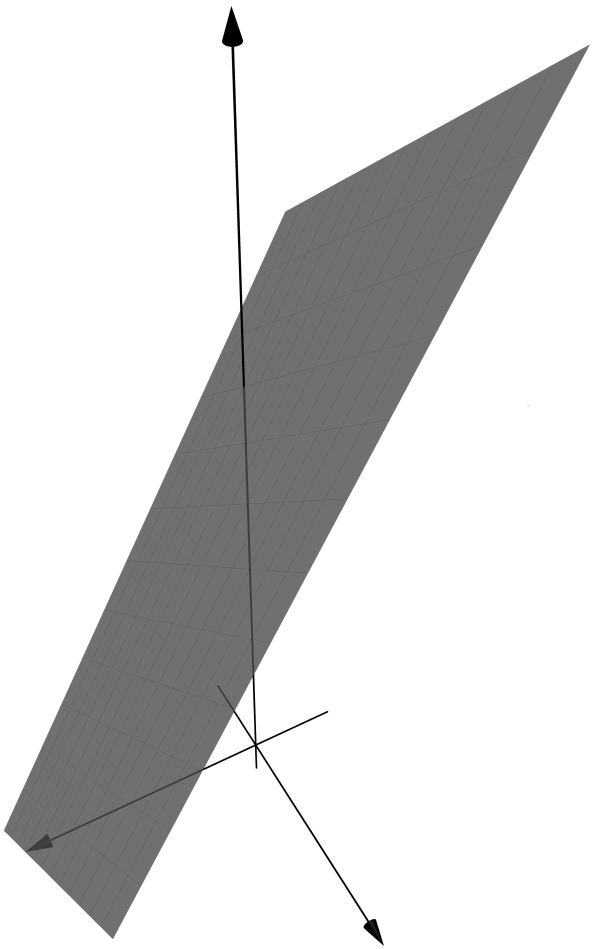
- f is smooth and strictly convex

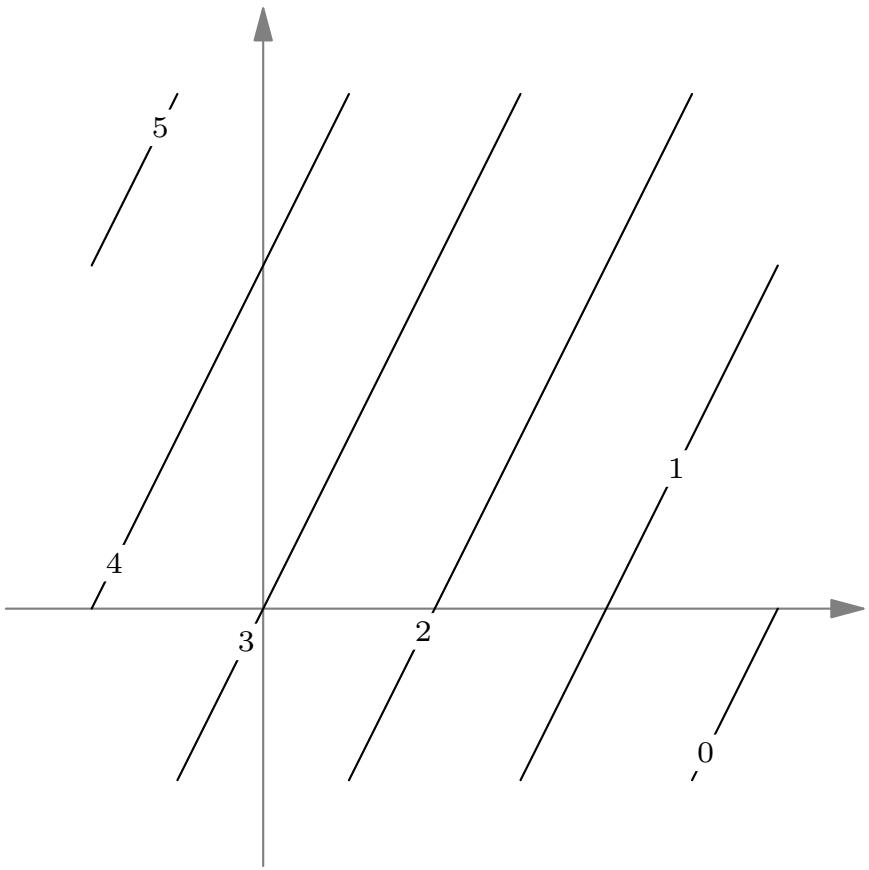
"f is smooth and strictly convex" via calculus



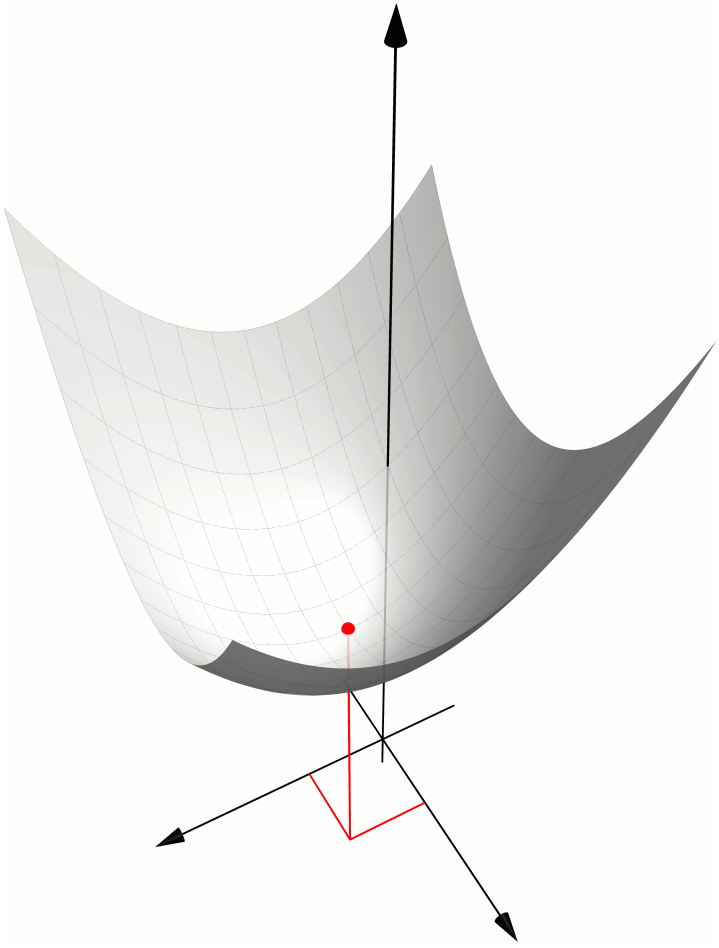
Minimizing linear functions

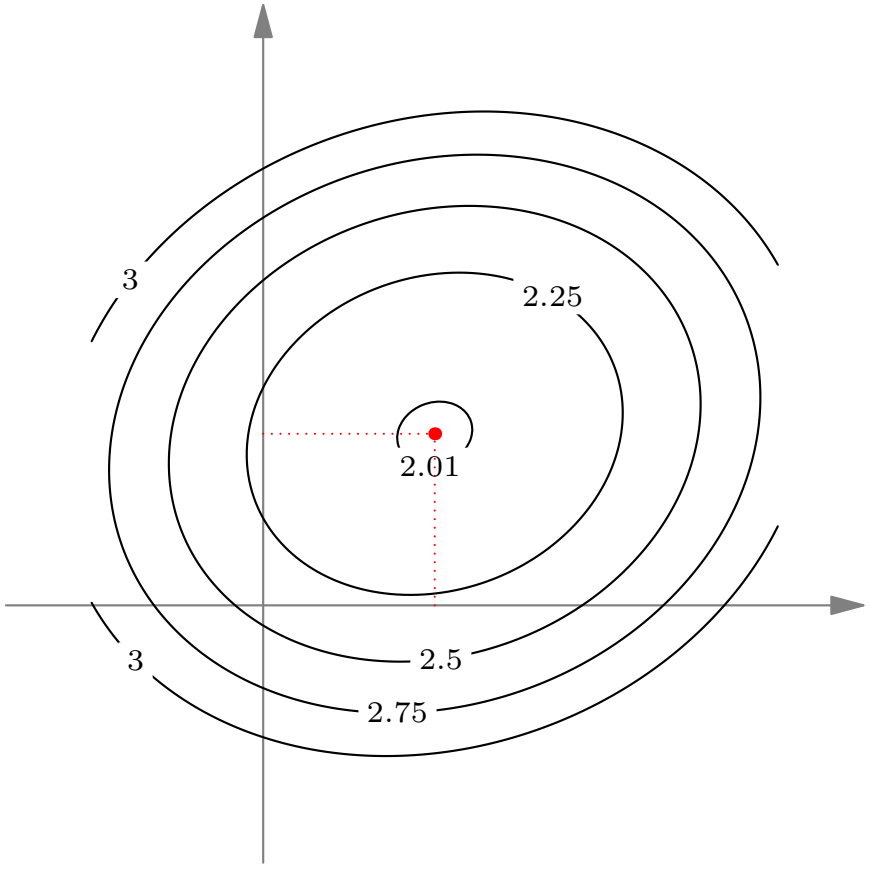
$$\min\ f(x_1, x_2) =$$



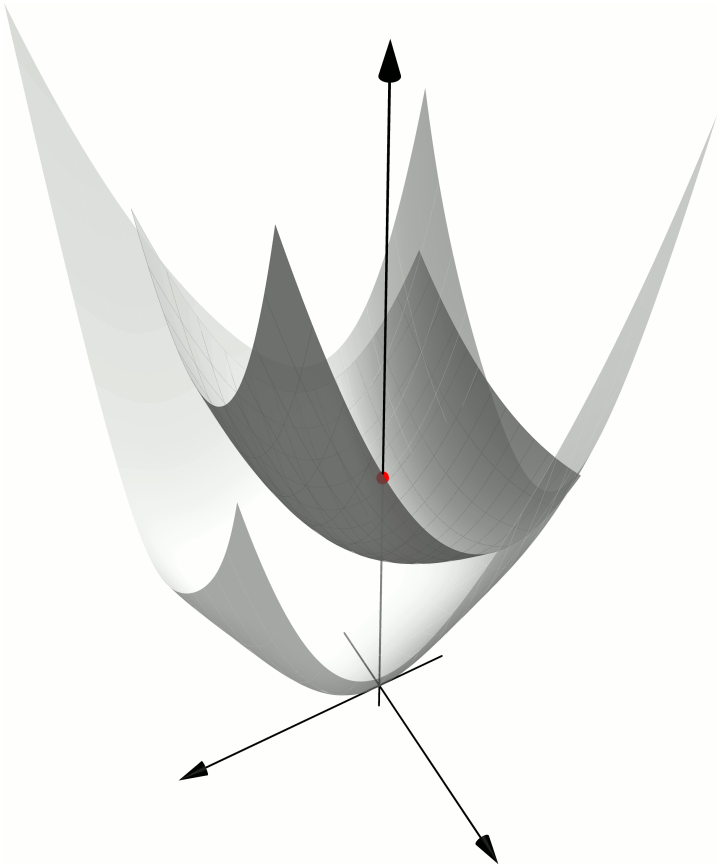


Minimizing quadratic functions





General f



- Surrogate function: Taylor expansion

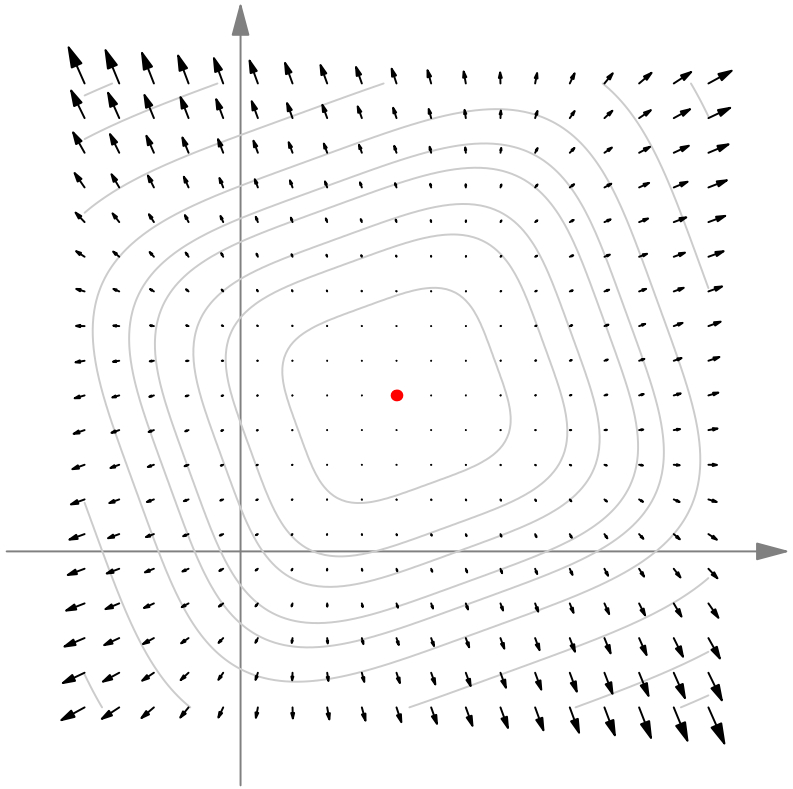
Newton method

- 1D

$$x_{k+1} = x_k + d_k, \qquad d_k = -(f''(x_k))^{-1} f'(x_k)$$

- nD

Stabilizing Newton method



- Descent direction
- Line search

Algorithm

1. **Input**
   - $f : \underline{x} \to f(\underline{x})$, $\underline{g} : \underline{x} \mapsto \underline{\nabla} f(\underline{x})$ $\underline{H} : \underline{x} \mapsto \underline{\nabla}^2 f(\underline{x})$
   - initial guess $\underline{x}_0$
   - tolerance ε
2. **Initiation:** $0 \leftarrow k$
3. $\underline{g}_k \leftarrow g(\underline{x}_k)$
4. if $\|\underline{g}_k\| \leq \varepsilon$: $\underline{x}_{\text{opt}} \leftarrow x_k$ and terminate
5. $\underline{H}_k \leftarrow \underline{h}(\underline{x}_k)$
6. solve $\underline{H}_k \underline{d}_k = -\underline{g}_k$
7. if $\underline{d}_k^T \underline{g}_k > 0$
   $d_k = -g_k$
8. $\alpha_k = \arg\min f(\underline{x}_k + \alpha \underline{d}_k)$ for $\alpha > 0$
9. $\underline{x}_{k+1} \leftarrow \underline{x}_k + \alpha_k \underline{d}_k$
10. $k \leftarrow k+1$
11. go to 3

Algorithm

- Example

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

- `https://gitlab.com/jan.zeman4/132mmo` > `codes`

Input (Line 1)

```
function newton
    function val = f(x)
       val = (x(1,:)-2).^2 + (x(1,:)-2*x(2,:).^2).^2;
    end

    function val = g(x)
        val = zeros(2,1);
        val(1) = -4*x(2)^2 + 4*x(1) - 4;
        val(2) = 8*x(2)*(2*x(2)^2-x(1));
    end

    function val = H(x)
        val = zeros(2,2);
        val(1,1) = 4;
        val(1,2) = -8*x(2);
        val(2,1) = val(1,2);
        val(2,2) = 8*(6*x(2)^2-x(1));
    end
```

Input (Line 1)

```
    function val = fdir(alpha)
        val = f(xk+dk*alpha);
    end

xk = [2;2];
tolerance = 1e-6;
```

```
clf;
h = ezcontour(@(x,y)f([x';y']), [-1,5,-3,3]);
h.LevelList = [0:0.5:10];
colorbar;
colormap gray;
hold on;
```

Initiation, convergence test, Hessian update, Newton step, test of descent direction (Lines 2–7)

```
k = 0;
gk = g(xk);

while(norm(gk) > tolerance)
    disp(['iteration = ' num2str(k) ' || gradient || = ' ...
        num2str(norm(gk)) ' f = ' num2str(f(xk))]);
    plot(xk(1),xk(2),'or');
    pause;
```

```
    Hk = H(xk);
    dk = -Hk\gk;
```

```
    if( dk'*gk >= 0 || max(isnan(dk))==1 )
        disp( 'correcting' );
        dk = -gk;
    end
```

Line search, solution update, counter increment (Lines 8–11)

```
    alphak = fminbnd(@fdir,0,10);
    disp(['linesearch: alpha = ' num2str(alphak)]);
```

```
    xk = xk + alphak*dk;
    gk = g(xk);
    k = k+1;
end
```

```
disp(['Convergence: iteration = ' num2str(k) ...
    ' ||gradient|| = ' num2str(norm(gk))]);
xk
f(xk)
end
```

- Try: $\varepsilon = 10^{-10}$, $\underline{x}_0 = [-2;0]$, $[0;0]$, $[-10;30]$

Summary

- Advantages
- Disadvantages
  -
  -
  -
- More lightweight versions of Newton method
  -
  -
- Interested in learning the full story? Try [1] or
  - AE4B33OPT: Optimization @ CTU [link]
  - AE4M35KO: Combinatorial Optimization @CTU [link]
  - EE364a: Convex Optimization I @ Stanford [link]
  - NMNV534: Numerical Optimization Methods @ CUNI [link]


J.-F. Bonnans, J. C. Gilbert, C. Lemarechal, and C.A. Sagastizábal, *Numerical optimization: Theoretical and practical aspects*, Springer, 2006.