Practical Course: Modeling, Simulation, Optimization

Week 10

Daniël Veldman

Chair in Dynamics, Control, and Numerics, Friedrich-Alexander-University Erlangen-Nürnberg

Contents

10.A Improved gradient descent algorithms

Constrained optimization

Consider the optimization problem

$$\min_{u \in U_{\text{ad}}} J(\mathbf{x}, \mathbf{u})$$

with $\mathbf{u} \in U_{\text{ad}} \subset \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^N$ subject to

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{0}.$$

Assume that $\mathbf{A}$ is invertible such that we can consider $J(\mathbf{x}(\mathbf{u}), \mathbf{u}) =: \tilde{J}(\mathbf{u})$.

Question: How to compute the Jacobian?

ANSWER 1: By finite differences.
Choose a step size h (typically 10^{-5}) and approximate for every $m \in \{1, 2, \ldots, M\}$

$$\left(\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\mathbf{u}}(\mathbf{u})\right)_m = \frac{\mathrm{d}\tilde{J}}{\mathrm{d}u_m}(\mathbf{u}) \approx \frac{\tilde{J}(\mathbf{u} + h\mathbf{e}_m) - J(\mathbf{u})}{h} = \frac{J(\mathbf{x} + \delta\mathbf{x}_m, \mathbf{u} + h\mathbf{e}_m) - J(\mathbf{x}, \mathbf{u})}{h},$$

where $\delta\mathbf{x}_m$ satisfies

$$\mathbf{A}\delta\mathbf{x}_m + h\mathbf{B}\mathbf{e}_m = \mathbf{0}.$$

Note: we need to solve M linear systems in N unknowns.
This is very time-consuming when M and N are large.

Constrained optimization

Consider the optimization problem

$$\min_{u \in U_{\mathrm{ad}}} J(\mathbf{x}, \mathbf{u})$$

with $\mathbf{u} \in U_{\mathrm{ad}} \subset \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^N$ subject to

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{0}.$$

Assume that $\mathbf{A}$ is invertible such that we can consider $J(\mathbf{x}(\mathbf{u}), \mathbf{u}) =: \tilde{J}(\mathbf{u})$.

Question: How to compute the Jacobian?

ANSWER 2: Analytically.
Similarly, as in the exercise we can use the chain rule to find

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\mathbf{u}} = \frac{\partial J}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{u}} + \frac{\partial J}{\partial \mathbf{u}} = -\frac{\partial J}{\partial \mathbf{x}}\mathbf{A}^{-1}\mathbf{B} + \frac{\partial J}{\partial \mathbf{u}}.$$

Constrained optimization

Consider the optimization problem

$$\min_{u \in U_{\mathrm{ad}}} J(\mathbf{x}, \mathbf{u})$$

with $\mathbf{u} \in U_{\mathrm{ad}} \subset \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^N$ subject to

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{0}.$$

Assume that $\mathbf{A}$ is invertible such that we can consider $J(\mathbf{x}(\mathbf{u}), \mathbf{u}) =: \tilde{J}(\mathbf{u})$.

Question: How to compute the Jacobian?

ANSWER 2: Analytically.
Similarly, as in the exercise we can use the chain rule to find

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\mathbf{u}} = \frac{\partial J}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{u}} + \frac{\partial J}{\partial \mathbf{u}} = -\frac{\partial J}{\partial \mathbf{x}}\mathbf{A}^{-1}\mathbf{B} + \frac{\partial J}{\partial \mathbf{u}}.$$

The computational cost depends on where you put the brackets:

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\mathbf{u}} = -\frac{\partial J}{\partial \mathbf{x}}\left(\mathbf{A}^{-1}\mathbf{B}\right) + \frac{\partial J}{\partial \mathbf{u}} = -\left(\frac{\partial J}{\partial \mathbf{x}}\mathbf{A}^{-1}\right)\mathbf{B} + \frac{\partial J}{\partial \mathbf{u}}.$$

Note: the first expression requires the solution of M linear system in N unknowns, whereas the second requires requires the solution of 1 linear system in N unknowns.

Constrained optimization

Consider the optimization problem

$$\min_{u \in U_{\mathrm{ad}}} J(\mathbf{x}, \mathbf{u})$$

with $\mathbf{u} \in U_{\mathrm{ad}} \subset \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^N$ subject to

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{0}.$$

Assume that $\mathbf{A}$ is invertible such that we can consider $J(\mathbf{x}(\mathbf{u}), \mathbf{u}) =: \tilde{J}(\mathbf{u})$.

Question: How to compute the Jacobian?

ANSWER 3: Using the Lagrangian.
Introduce the vector of Lagrange multipliers $\boldsymbol{\lambda}$ and form the Lagrangian

$$\mathcal{L}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) = J(\mathbf{x}, \mathbf{u}) + \boldsymbol{\lambda}^\top (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u})$$

Take the partial derivative w.r.t. $\mathbf{u}$ to find the Jacobian

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\mathbf{u}} = \frac{\partial \mathcal{L}}{\partial \mathbf{u}} = \frac{\partial J}{\partial \mathbf{u}} + \boldsymbol{\lambda}^\top \mathbf{B}\mathbf{u}.$$

Set the partial derivative w.r.t. $\mathbf{x}$ to zero to determine $\boldsymbol{\lambda}$:

$$\mathbf{0} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \frac{\partial J}{\partial \mathbf{x}} + \boldsymbol{\lambda}^\top \mathbf{A}, \qquad -\boldsymbol{\lambda}^\top = \frac{\partial J}{\partial \mathbf{x}} \mathbf{A}^{-1}, \qquad \boldsymbol{\lambda} = -\left(\mathbf{A}^\top\right)^{-1} \left(\frac{\partial J}{\partial \mathbf{x}}\right)^\top.$$

The result is the same as for answer 2 (with well-placed brackets).

Step size selection

For a convex C^2-functional $J(\mathbf{u})$,
we can estimate the stepsize based on a quadratic approximation:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k), \qquad \beta_k > 0,$$

$$J(\mathbf{u}_{k+1}) \approx J(\mathbf{u}_k) - \beta_k G + \frac{H}{2}\beta_k^2 + O(\beta_k^3)$$

with

$$G = \langle \nabla J(\mathbf{u}_k), \nabla J(\mathbf{u}_k) \rangle$$

$$H = \left[\frac{\mathrm{d}^2}{\mathrm{d}\theta^2} J(\mathbf{u}_k + \theta \nabla J(\mathbf{u}_k)) \right]_{\theta=0}.$$

Note: G is positive because we update in a descent direction.
H is positive because J is convex.

Step size selection

For a convex C^2-functional $J(\mathbf{u})$,
we can estimate the stepsize based on a quadratic approximation:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k), \qquad \beta_k > 0,$$

$$J(\mathbf{u}_{k+1}) \approx J(\mathbf{u}_k) - \beta_k G + \frac{H}{2}\beta_k^2 + O(\beta_k^3)$$

with

$$G = \langle \nabla J(\mathbf{u}_k), \nabla J(\mathbf{u}_k) \rangle$$

$$H = \left[\frac{\mathrm{d}^2}{\mathrm{d}\theta^2} J(\mathbf{u}_k + \theta \nabla J(\mathbf{u}_k)) \right]_{\theta=0}.$$

Note: G is positive because we update in a descent direction.
H is positive because J is convex.
Set derivative of the quadratic approximation to zero:

$$-G + H\beta_{k,\mathrm{opt}} = 0, \qquad \beta_{k,\mathrm{opt}} = \frac{G}{H}.$$

Step size selection

For a convex C^2-functional $J(\mathbf{u})$,
we can estimate the stepsize based on a quadratic approximation:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k), \qquad \beta_k > 0,$$

$$J(\mathbf{u}_{k+1}) \approx J(\mathbf{u}_k) - \beta_k G + \frac{H}{2}\beta_k^2 + O(\beta_k^3)$$

with

$$G = \langle \nabla J(\mathbf{u}_k), \nabla J(\mathbf{u}_k) \rangle$$

$$H = \left[\frac{\mathrm{d}^2}{\mathrm{d}\theta^2} J(\mathbf{u}_k + \theta \nabla J(\mathbf{u}_k)) \right]_{\theta=0}.$$

Note: G is positive because we update in a descent direction.
H is positive because J is convex.
Set derivative of the quadratic approximation to zero:

$$-G + H\beta_{k,\text{opt}} = 0, \qquad \beta_{k,\text{opt}} = \frac{G}{H}.$$

When J is quadratic, $J(\mathbf{u}_k + \beta_{k,\text{opt}} \nabla J(\mathbf{u}_k)) = J(\mathbf{u}_k) - \beta_{k,\text{opt}} G + \frac{H}{2}\beta_{k,\text{opt}}^2 = J(\mathbf{u}_k) - \frac{G^2}{2H}$
When J is not quadratic, there are higher order terms and we cannot guarantee that $J(\mathbf{u}_k + \beta_{k,\text{opt}} \nabla J(\mathbf{u}_k)) \leq J(\mathbf{u}_k)$. We still need to do a line search (starting from $\beta_{k,\text{opt}}$)

Computation of H (example)

Consider the optimization problem

$$\min_{u \in U_{\mathrm{ad}}} \tfrac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \tfrac{1}{2}\mathbf{u}^\top \mathbf{R}\mathbf{u}$$

with $\mathbf{Q} = \mathbf{Q}^\top$, $\mathbf{R} = \mathbf{R}^\top$, $\mathbf{u} \in U_{\mathrm{ad}} \subset \mathbb{R}^M$, and $\mathbf{x} \in \mathbb{R}^N$ subject to

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{0}.$$

As explained before, we can compute the gradient $\nabla J(\mathbf{u}_k)$ at the current iterate $\mathbf{u}_k$. We want to compute

$$H = \left[\frac{\mathrm{d}^2}{\mathrm{d}\theta^2} J(\mathbf{u_k} + \theta \nabla J(\mathbf{u}_k))\right]_{\theta=0}.$$

Observe that

$$\begin{aligned} J(\mathbf{u}_k + \theta\nabla J) &= \tfrac{1}{2}(\mathbf{x}_k + \theta \mathbf{x}_k^\nabla)^\top \mathbf{Q}(\mathbf{x}_k + \theta\mathbf{x}_k^\nabla) + \tfrac{1}{2}(\mathbf{u}_k + \theta\nabla J(\mathbf{u}_k))^\top \mathbf{R}(\mathbf{u}_k + \theta \nabla J(\mathbf{u}_k)) \\ &= \tfrac{1}{2}\mathbf{x}_k^\top \mathbf{Q}\mathbf{x}_k + \tfrac{1}{2}\mathbf{u}_k^\top \mathbf{R}\mathbf{u}_k + \theta\left(\mathbf{x}_k^\top \mathbf{Q}\mathbf{x}_k^\nabla + \mathbf{u}_k^\top \mathbf{R}\nabla J(\mathbf{u}_k)\right) \\ &\quad \theta^2 \left(\tfrac{1}{2}\left(\mathbf{x}_k^\nabla\right)^\top \mathbf{Q}\mathbf{x}_k^\nabla + \tfrac{1}{2}\left(\nabla J(\mathbf{u}_k)\right)^\top \mathbf{R}\nabla J(\mathbf{u}_k)\right), \end{aligned}$$

where $\mathbf{x}_k = \mathbf{A}^{-1}\mathbf{B}\mathbf{u}_k$ and $\mathbf{x}_k^\nabla = \mathbf{A}^{-1}\mathbf{B}\nabla J(\mathbf{u}_k)$. Differentiating twice to θ, we obtain

$$H = \left(\mathbf{x}_k^\nabla\right)^\top \mathbf{Q}\mathbf{x}_k^\nabla + (\nabla J(\mathbf{u}_k))^\top \mathbf{R}\nabla J(\mathbf{u}_k).$$

Inequality constraints

Consider the optimization problem

$$\min_{u \in U_{\text{ad}}} J(\mathbf{u}) = J(\mathbf{x}(\mathbf{u}), \mathbf{u})$$

with $\mathbf{u} \in U_{\text{ad}} \subset \mathbb{R}^M$ and $\mathbf{x} \in \mathbb{R}^N$ subject to

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{0}.$$

We distinguish between two types of constraints:

- Constraints on $\mathbf{u}$ ('input constraints'), $g(\mathbf{u}) \geq \mathbf{0}$
- Constraints on $\mathbf{x}(\mathbf{u})$ ('state constraints') $h(\mathbf{x}(\mathbf{u})) \geq \mathbf{0}$.

Input constraints can be easily incorporated with the projected gradient method.

Projected gradient method

Suppose we want to solve an optimization problem with the constraints:

$$a \leq u_m \leq b, \qquad m \in \{1, 2, \ldots, M\}.$$

(This thus defines the admissible set U_{ad})

Projected gradient method

Suppose we want to solve an optimization problem with the constraints:

$$a \leq u_m \leq b, \qquad m \in \{1, 2, \ldots, M\}.$$

(This thus defines the admissible set U_{ad})

Problem: We do not know whether $\mathbf{u}_{k+1} = \mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)$ is in U_{ad}.
(Even when $\mathbf{u}_k \in U_{\text{ad}}$)

Solution: Project $\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)$ onto the U_{ad}, i.e. do the update as

$$\mathbf{u}_{k+1} = \Pi_{U_{\text{ad}}} \left(\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k) \right) \in U_{\text{ad}}$$

Projected gradient method

Suppose we want to solve an optimization problem with the constraints:

$$a \leq u_m \leq b, \qquad m \in \{1, 2, \dots, M\}.$$

(This thus defines the admissible set U_{ad})

Problem: We do not know whether $\mathbf{u}_{k+1} = \mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)$ is in U_{ad}.
(Even when $\mathbf{u}_k \in U_{\mathrm{ad}}$)

Solution: Project $\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)$ onto the U_{ad}, i.e. do the update as

$$\mathbf{u}_{k+1} = \Pi_{U_{\mathrm{ad}}}(\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)) \in U_{\mathrm{ad}}$$

In general, the projection onto the admissible set is difficult to compute
(it requires the solution of another optimization problem).

However, for the considered admissible set, the computation is straightforward:

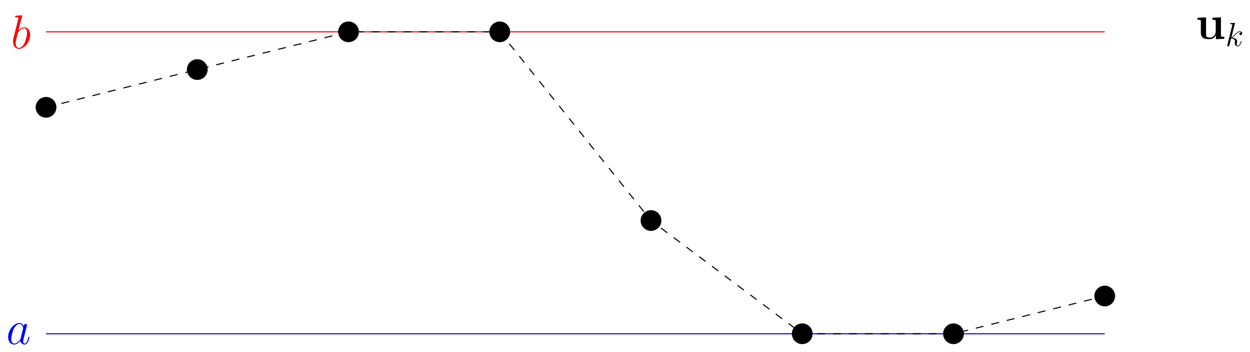
$$(\Pi_{U_{\mathrm{ad}}}(\mathbf{u}))_m = \begin{cases} a & u_m \leq a \\ u_m & a < u_m < b \\ b & u_m \geq b \end{cases}$$

Projected gradient method (graphical illustration)

$$a \leq u_m \leq b, \qquad m \in \{1, 2, \ldots, M\}.$$

$$\mathbf{u}_{k+1} = \Pi_{U_{\text{ad}}} \left(\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)\right) \in U_{\text{ad}}$$

$$\left(\Pi_{U_{\text{ad}}}(\mathbf{u})\right)_m = \begin{cases} a & u_m \leq a \\ u_m & a < u_m < b \\ b & u_m \geq b \end{cases}$$

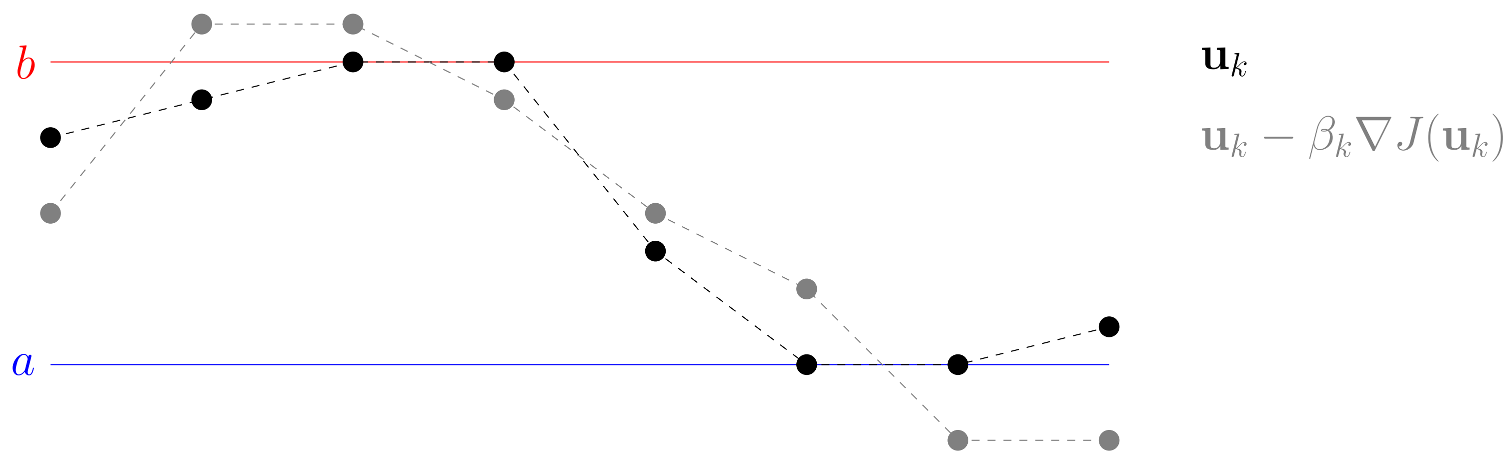


Projected gradient method (graphical illustration)

$$a \leq u_m \leq b, \qquad m \in \{1, 2, \dots, M\}.$$

$$\mathbf{u}_{k+1} = \Pi_{U_{\text{ad}}}\left(\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)\right) \in U_{\text{ad}}$$

$$\left(\Pi_{U_{\text{ad}}}(\mathbf{u})\right)_m = \begin{cases} a & u_m \leq a \\ u_m & a < u_m < b \\ b & u_m \geq b \end{cases}$$

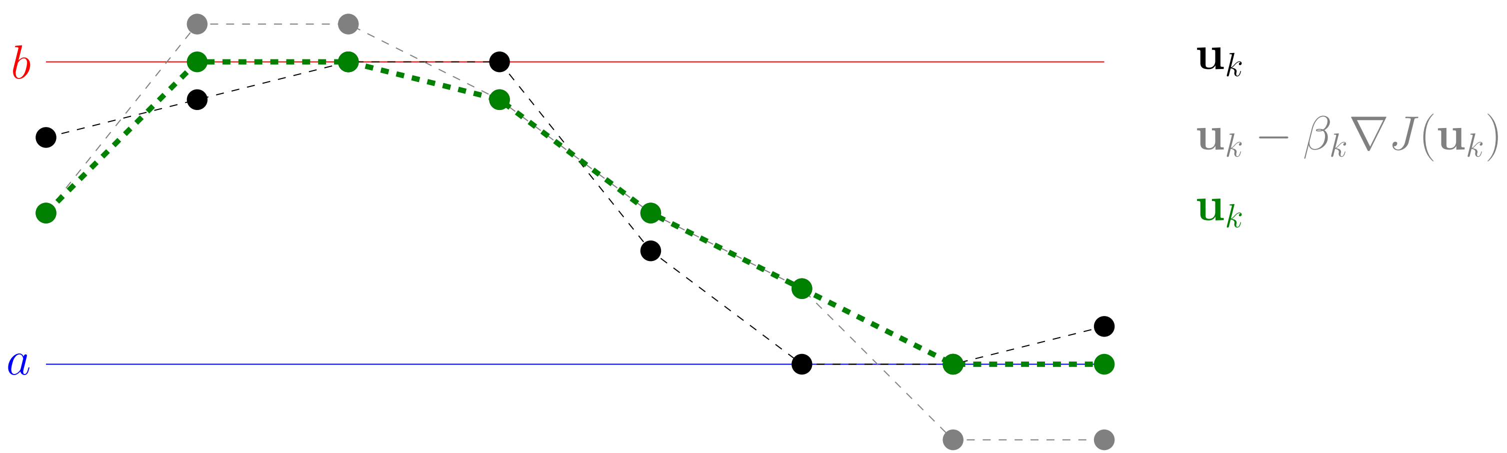


Projected gradient method (graphical illustration)

$$a \leq u_m \leq b, \qquad m \in \{1, 2, \dots, M\}.$$

$$\mathbf{u}_{k+1} = \Pi_{U_{\text{ad}}} \left(\mathbf{u}_k - \beta_k \nabla J(\mathbf{u}_k)\right) \in U_{\text{ad}}$$

$$\left(\Pi_{U_{\text{ad}}}(\mathbf{u})\right)_m = \begin{cases} a & u_m \leq a \\ u_m & a < u_m < b \\ b & u_m \geq b \end{cases}$$



Quadratic approximation for the projected gradient

We replace $\nabla J(\mathbf{u}_k)$ by

$$\nabla \Pi J(\mathbf{u}_k) = -\lim_{h \downarrow 0} \frac{\Pi(\mathbf{u}_k - h\nabla J(\mathbf{u}_k)) - \mathbf{u}_k}{h}$$

$\nabla \Pi J(\mathbf{u}_k)$ is equal to $\nabla J(\mathbf{u}_k)$ except for entries where the $-\nabla J(\mathbf{u}_k)$ is pointing out of the admissible set.

Explicitly,

$$(\nabla \Pi J(\mathbf{u}_k))_m = \begin{cases} 0 & (\mathbf{u}_k)_m = a \text{ and } (\nabla J(\mathbf{u}_k))_m \geq 0 \\ & \quad \text{or } (\mathbf{u}_k)_m = b \text{ and } (\nabla J(\mathbf{u}_k))_m \leq 0 \\ (\nabla J(\mathbf{u}_k))_m & \text{otherwise.} \end{cases}$$

Quadratic approximation for the projected gradient

We replace $\nabla J(\mathbf{u}_k)$ by

$$\nabla \Pi J(\mathbf{u}_k) = -\lim_{h \downarrow 0} \frac{\Pi(\mathbf{u}_k - h\nabla J(\mathbf{u}_k)) - \mathbf{u}_k}{h}$$

$\nabla \Pi J(\mathbf{u}_k)$ is equal to $\nabla J(\mathbf{u}_k)$ except for entries where the $-\nabla J(\mathbf{u}_k)$ is pointing out of the admissible set.

Explicitly,

$$(\nabla \Pi J(\mathbf{u}_k))_m = \begin{cases} 0 & (\mathbf{u}_k)_m = a \text{ and } (\nabla J(\mathbf{u}_k))_m \geq 0 \\ & \quad \text{or } (\mathbf{u}_k)_m = b \text{ and } (\nabla J(\mathbf{u}_k))_m \leq 0 \\ (\nabla J(\mathbf{u}_k))_m & \text{otherwise.} \end{cases}$$

We then can use the quadratic approximation:

$$J(\mathbf{u}_{k+1}) \approx J(\mathbf{u}_k) - \beta_k G + \frac{H}{2}\beta_k^2 + O(\beta_k^3)$$

with

$$G = \langle \nabla J(\mathbf{u}_k), \nabla \Pi J(\mathbf{u}_k) \rangle$$
$$H = \left[\frac{\mathrm{d}^2}{\mathrm{d}\theta^2} J(\mathbf{u_k} + \theta \nabla \Pi J(\mathbf{u}_k)) \right]_{\theta=0}.$$

Computation of H with projected gradient (example)

Consider the optimization problem

$$\min_{u \in U_{\mathrm{ad}}} \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \frac{1}{2}\mathbf{u}^\top \mathbf{R}\mathbf{u}$$

with $\mathbf{Q} = \mathbf{Q}^\top$, $\mathbf{R} = \mathbf{R}^\top$, $\mathbf{u} \in U_{\mathrm{ad}} \subset \mathbb{R}^M$, and $\mathbf{x} \in \mathbb{R}^N$ subject to

$$\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \mathbf{0}.$$

We have the 'projected gradient' (which is a bad name) $\nabla \Pi J(\mathbf{u}_k)$.

Compute the state resulting from the projected gradient

$$\mathbf{x}_k^{\nabla \Pi} = -\mathbf{A}^{-1} \left(\mathbf{B} \nabla \Pi J(\mathbf{u}_k) \right).$$

We can then compute

$$H = \left(\mathbf{x}_k^{\nabla \Pi} \right)^\top \mathbf{Q}\mathbf{x}_k^{\nabla \Pi} + \left(\nabla \Pi J(\mathbf{u}_k) \right)^\top \mathbf{R} \nabla \Pi J(\mathbf{u}_k).$$

State constraints

For state constraints (i.e. constraints on $\mathbf{x}(\mathbf{u})$),
it is not so straightforward to determine the projection on the admissible set.

State constraints can for example be included using a penalty function method, but we will not discuss this further in this course.

10.B Convergence analysis for gradient descent

Main result

We return to the more abstract optimization problem:

$$\min_{u \in \mathbb{R}^M} J(u).$$

Denote the minimizer by u^*.

For simplicity, we consider a gradient descent algorithm with a fixed step size β

$$u_{k+1} = u_k - \beta \nabla J(u_k).$$

Main result

We return to the more abstract optimization problem:

$$\min_{u \in \mathbb{R}^M} J(u).$$

Denote the minimizer by u^*.

For simplicity, we consider a gradient descent algorithm with a fixed step size β

$$u_{k+1} = u_k - \beta \nabla J(u_k).$$

Two assumptions:

- The functional J is α-convex, i.e.

$$J(\theta u + (1-\theta)v) \leq \theta J(u) + (1-\theta)J(v) - \frac{\alpha\theta(1-\theta)}{2}|u-v|^2, \qquad \theta \in [0,1].$$

- The gradient $\nabla J(u)$ is Lipschitz, i.e. there is an $L > 0$ such that for all u and v

$$|\nabla J(u) - \nabla J(v)| \leq L|u-v|.$$

Theorem

$$|u_k - u^*|^2 \leq \left(1 - 2\alpha\beta + \beta^2 L^2\right)^k |u_0 - u^*|^2$$

Observation 1

The functional J is α-convex:

$$J(\theta u + (1-\theta)v) \leq \theta J(u) + (1-\theta)J(v) - \frac{\alpha\theta(1-\theta)}{2}|u-v|^2.$$

Subtract expand the brackets on the LHS and subtract $J(v)$ on both sides:

$$J(v+\theta(u-v)) - J(v) \leq \theta J(u) - \theta J(v) - \frac{\alpha\theta(1-\theta)}{2}|u-v|^2.$$

Divide by θ and take the limit $\theta \to 0$:

$$\langle \nabla J(v), u-v\rangle = \lim_{\theta\to 0} \frac{J(v+\theta(u-v)) - J(v)}{\theta} \leq J(u) - J(v) - \frac{\alpha}{2}|u-v|^2.$$

We conclude

$$\langle \nabla J(v), u-v\rangle \leq J(u) - J(v) - \frac{\alpha}{2}|u-v|^2.$$

Observation 2

From the previous slide:

$$\langle \nabla J(v), u - v \rangle \leq J(u) - J(v) - \frac{\alpha}{2}|u - v|^2.$$

Because this holds for all u and v, we may interchange u and v to obtain:

$$\langle \nabla J(u), v - u \rangle \leq J(v) - J(u) - \frac{\alpha}{2}|v - u|^2.$$

Adding these two equations, we find

$$\langle \nabla J(v) - \nabla J(u), u - v \rangle \leq -\alpha |u - v|^2.$$

Multiply by -1, to find

$$\langle \nabla J(u) - \nabla J(v), u - v \rangle \geq \alpha |u - v|^2.$$

Proof

Theorem

$$|u_k - u^*|^2 \leq \left(1 - 2\alpha\beta + \beta^2 L^2\right)^k |u_0 - u^*|^2$$

$$\begin{aligned}|u_{k+1} - u^*|^2 &= \langle u_{k+1} - u^*, u_{k+1} - u^* \rangle \\ &= \langle u_k - \beta \nabla J(u_k) - u^*, u_k - \beta \nabla J(u_k) - u^* \rangle \\ &= \langle u_k - u^*, u_k - u^* \rangle - 2\beta \langle \nabla J(u_k), u_k - u^* \rangle + \beta^2 \langle \nabla J(u_k), \nabla J(u_k) \rangle\end{aligned}$$

Proof

Theorem

$$|u_k - u^*|^2 \leq \left(1 - 2\alpha\beta + \beta^2 L^2\right)^k |u_0 - u^*|^2$$

$$\begin{aligned}|u_{k+1} - u^*|^2 &= \langle u_{k+1} - u^*, u_{k+1} - u^* \rangle \\ &= \langle u_k - \beta \nabla J(u_k) - u^*, u_k - \beta \nabla J(u_k) - u^* \rangle \\ &= \langle u_k - u^*, u_k - u^* \rangle - 2\beta \langle \nabla J(u_k), u_k - u^* \rangle + \beta^2 \langle \nabla J(u_k), \nabla J(u_k) \rangle\end{aligned}$$

Using that $\nabla J(u^*) = 0$ and Observation 2, we find

$$\langle \nabla J(u_k), u_k - u^* \rangle = \langle \nabla J(u_k) - \nabla J(u^*), u_k - u^* \rangle \geq \alpha |u_k - u^*|^2.$$

Again using that $\nabla J(u^*) = 0$ and the Lipschitz continuity of $\nabla J(u)$, we also have that

$$\langle \nabla J(u_k), \nabla J(u_k) \rangle = |\nabla J(u_k) - \nabla J(u^*)|^2 \leq L^2 |u_k - u^*|^2.$$

Proof

Theorem

$$|u_k - u^*|^2 \leq \left(1 - 2\alpha\beta + \beta^2 L^2\right)^k |u_0 - u^*|^2$$

$$\begin{aligned} |u_{k+1} - u^*|^2 &= \langle u_{k+1} - u^*, u_{k+1} - u^* \rangle \\ &= \langle u_k - \beta \nabla J(u_k) - u^*, u_k - \beta \nabla J(u_k) - u^* \rangle \\ &= \langle u_k - u^*, u_k - u^* \rangle - 2\beta \langle \nabla J(u_k), u_k - u^* \rangle + \beta^2 \langle \nabla J(u_k), \nabla J(u_k) \rangle \end{aligned}$$

Using that $\nabla J(u^*) = 0$ and Observation 2, we find

$$\langle \nabla J(u_k), u_k - u^* \rangle = \langle \nabla J(u_k) - \nabla J(u^*), u_k - u^* \rangle \geq \alpha |u_k - u^*|^2.$$

Again using that $\nabla J(u^*) = 0$ and the Lipschitz continuity of $\nabla J(u)$, we also have that

$$\langle \nabla J(u_k), \nabla J(u_k) \rangle = |\nabla J(u_k) - \nabla J(u^*)|^2 \leq L^2 |u_k - u^*|^2.$$

Inserting these two results back into the original expression, we conclude

$$|u_{k+1} - u^*|^2 \leq \left(1 - 2\alpha\beta + \beta^2 L^2\right) |u_k - u^*|^2$$

The result now follows by induction over k.

Other algorithms

There are many more gradient-based algorithms.
Gradient-descent/steepest descent is the simplest one.
For quadratic problems, the Conjugate Gradient (CG) method is the best method.
When optimizing $u \in \mathbb{R}^M$, it converges in at most M iterations to the minimizer.
For nonquadratic problems, other algorithms can be more effective.

see e.g. Ascher, The chaotic nature of faster gradient descent methods