Institut für Numerische Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn

universitätbonn

Computer lab Numerical Algorithms
Winter term 2012/2013
Prof. Dr. M. Rumpf – B. Geihe, B. Heeren

Problem sheet 6 **January 15th, 2012**

Problem 8 (Gradient descent)

To find a local minimum of a function $E : \mathbb{R}^d \to \mathbb{R}^+$ we want to apply a gradient descent method. Starting at some initial value $x_0 \in \mathbb{R}^d$ we compute iteratively a sequence $(x_k)_k$ with $E(x_k) > E(x_{k+1})$. At a fixed iteration step $k \geq 0$ we are thus looking for a direction d_k and some stepsize $\tau_k > 0$, such that $x_{k+1} := x_k + \tau_k d_k$ fulfills the above property.

As the negative gradient is known to be the *direction of steepest descent* one chooses $d_k = -\nabla E(x_k)$. To achieve convergence it is crucial that $E(x_k)_k$ decreases fast enough. This is ensured by choosing a so called *efficient stepsize*, e.g. by applying Armijo's rule.

Algorithm 1: Gradient descent with stepsize control

Input: Initial value $x_0 \in \mathbb{R}^d$, tolerance $\epsilon \in \mathbb{R}$, max. number of iterations $k_{\max} \in \mathbb{N}$
notTerminated = true;
$k = 0$
while *notTerminated* **do**
  compute descent direction $d_k = -\nabla E(x_k)$
  find an admissible stepsize τ_k
  set $x_{k+1} = x_k + \tau_k\, d_k$
  $k = k + 1$
  if $\|d_k\| \leq \epsilon$ *or* $k = k_{\max}$ **then**
    notTerminated = false;

Algorithm 2: Armijo stepsize control

Input: current location and direction $x, d \in \mathbb{R}^n$, parameters $\beta \in (0,1)$, $0 < \tau_{\min} < \tau_{\max}$.
Output: maximal stepsize $\tau^\star = 2^i\,\tau \in [\tau_{\min}, \tau_{\max}]$, $i \in \mathbb{Z}$, such that

$$E(x + \tau^\star\, d) \leq E(x) + \beta\, \tau^\star d \cdot \nabla E(x).$$

Choose initial stepsize τ
Compute the expected slope $s_e = d \cdot \nabla E(x)$
and the actually realized slope $s_r = (E(x + \tau\, d) - E(x))/\tau$
if $s_r > \beta\, s_e$ **then**
    while $s_r > \beta\, s_e$ *and* $\tau > \tau_{\min}$ **do**
        $\tau = \tau/2$
        update s_r
else
    while $s_r \leq \beta\, s_e$ *and* $\tau \leq \tau_{\max}$ **do**
        $\tau = 2\,\tau$
        update s_r
    $\tau = \tau/2$
if $\tau \leq \tau_{\min}$ **then**
    return *No stepsize found*.

To choose a reasonable initial stepsize τ one could either use the final stepsize τ_{k-1} of the previous iteration step or make use of a local quadratic approximation of the function $f(\lambda) := E(x + \lambda d)$. Note that $f'(\lambda) = \nabla E(x + \lambda d) \cdot d$, i.e. $f'(0) = -|d|^2$. We now set $\tau = \arg\min p(\lambda)$ where p is the unique quadratic function with $p(0) = f(0)$, $p'(0) = f'(0)$ and $p(\bar{\lambda}) = f(\bar{\lambda})$ for some suitable $\bar{\lambda} > 0$, e.g. $\bar{\lambda} = \tau_{k-1}$.

Note that evaluations of the objective functional E and especially its derivative ∇E are in general very time consuming. Hence it is important to reduce the number of evaluations as far as possible and to store corresponding terms to avoid repetitive computations!

Tasks:

(i) Complete GradientDescent::performSingleStep().

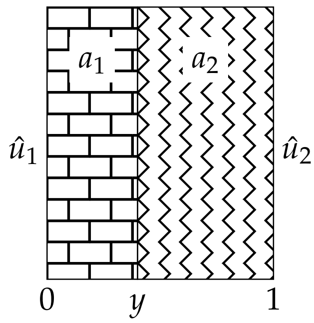
(ii) Implement GradientDescent::findStepsizeWithArmijo().

(iii) Write a simple test energy and a corresponding gradient and test the gradient descent.

Note: Your energy functional as well as the corresponding gradient should be derived from a suitable Operator<DomType,RangeType>. You can check your derivative by means of DerivativeChecker.

Problem 9 (1D Shape optimization: Wall example)

Consider a wall made up of concrete with a high thermal conductivity $a_1 > 0$ and low material costs c_1 on one side and an insulating material with a low thermal conductivity $0 < a_2 < a_1$ but high costs $c_2 > c_1$ on the other side. Given inner and outer temperatures $\hat{u}_1$ and $\hat{u}_1$ the heat loss shall be minimized while keeping the overall material costs low. This can be phrased as a minimization problem under the constraint that u minimizes the heat functional, i.e.



$$
\begin{aligned}
\mathbf{J}[y, u[y]] &= (c_1 y + c_2(1-y)) - (a_2 u'(1)) \\
\text{subject to} \qquad u[y] &= v[y] + g \\
v[y] &= \operatorname*{arg\,min}_{v(0)=v(1)=0} \; E[y, v]
\end{aligned}
$$

with some function $g : [0,1] \to \mathbb{R}$, $g(0) = \hat{u}_1$ and $g(1) = \hat{u}_2$, and

$$
E[y, v] := \frac{1}{2} \int_0^1 a_y(x)|v'(x)|^2 + a_y(x)v'(x) \cdot g'(x) \,\mathrm{d}x \,, \qquad a_y(x) = \begin{cases} a_1, & 0 \le x \le y \\ a_2, & y < x \le 1 \end{cases} .
$$

Now we discretize $\Omega = [0,1]$ and $u : \Omega \to \mathbb{R}$ by means of linear Finite Elements. Therefore we choose $N+1$ nodes $x_i = i\,h$, $i = 0, \ldots, N$, with $h = N^{-1}$, and consider linear basis functions ϕ_i uniquely defined by $\phi_i(x_j) = \delta_{ij}$. If we now write $u_h(x) = \sum u_i \phi_i(x)$, $\bar{u} := (u_i)_i \in \mathbb{R}^{N+1}$, and define a weighted stiffness matrix $A_y \in \mathbb{R}^{N+1,N+1}$ by

$$
(A_y)_{ij} := \int_0^1 a_y(x) \phi_i'(x) \cdot \phi_j'(x) \,\mathrm{d}x \,,
$$

we can rewrite $E[y, v_h] = E[y, \bar{v}] = \frac{1}{2} A_y \bar{v} \cdot \bar{v} + A_y \bar{v} \cdot \bar{g}$ and hence $\mathbf{J}[y, u_h] = \mathbf{J}[y, \bar{u}]$ as

$$
\begin{aligned}
\mathbf{J}[y, \bar{u}] &= (c_1 y + c_2(1-y)) - A_y \bar{u} \cdot e_{N+1} \\
\text{subject to} \qquad \bar{u} &= \bar{v} + \bar{g} \\
A_y \bar{v} &= -A_y \bar{g}
\end{aligned}
$$

where $g_i = 0$, $0 < i \le N$, and $g_0 = \hat{u}_1$, $g_{N+1} = \hat{u}_2$. Note, that we have to account for the zero boundary conditions by applying a suitable boundary mask to A_y.

To minimize $\mathbf{J}[y] = \mathbf{J}[y, \bar{u}]$ we want to apply a gradient descent method. Hence we need to compute

$$
\mathbf{J}_{,y}[y, \bar{u}] = (\mathbf{J}_{,y})[y, \bar{u}] - E_{,uy}[y, \bar{u}]\bar{p} \;=\; (c_1 - c_2) - A_y' \bar{u} \cdot \bar{p} \,,
$$

where $\bar{p}$ solves the dual problem $E_{,uu}[y, \bar{u}]\bar{p} = (\mathbf{J}_{,u})[y, \bar{u}]$ and $\bar{u}$ satisfies the constraint. Note, that $E_{,uu}[y, \bar{u}] = A_y$ and $(\mathbf{J}_{,u})[y, \bar{u}] = A_y' e_{N+1}$.

Tasks:

(i) Implement a function to "manually" assemble A_y and account for the jump in $a(x)$.

(ii) Derive the matrix A_y' and compute $E_{,uy}[y, \bar{u}]\bar{p} = A_y' \bar{u} \cdot \bar{p}$.

(iii) Implement $\mathbf{J}$ and $\mathbf{J}_{,y}$ and apply the gradient descent method to obtain an optimal y.

Note: You can check your numerical results by comparing them to the analytical values!