Optimierung 1, SS 2017

Implementationsaufgabe Newton-Verfahren

Implementieren Sie in MATLAB (oder Octave, wenn MATLAB nicht möglich) das im Folgenden beschriebene globalisierte Newton-Verfahren, das auch dann eingesetzt werden kann, wenn $\nabla^2 f(x_k)$ nicht invertierbar ist.

Seien $f \in \mathcal{C}^2(\mathbb{R}^n, \mathbb{R})$ und **die Einstellungsparameter** $\delta > 0$, $\mu > 0$, λ, ρ und η mit $0 < \lambda < 1$, $0 < \rho < 0.5$, $\eta > 1$, sowie $\varepsilon > 0$, $M > 0$, $maxit, maxit\_delta, maxit\_alpha \in \mathbb{N}$ und ein Startpunkt $x_0 \in \mathbb{R}^n$ gegeben.

Das Verfahren generiert eine Folge $\{x_k\}$ von Vektoren im $\mathbb{R}^n$, mit $x_{k+1} = x_k + \alpha_k d_k$, wobei die Schrittweite α_k den Minimierer der Funktion $f(x_k + \alpha d_k)$ für $\alpha \geq 0$, approximiert.
Für die Richtung d_k gilt $d_k = -(\delta_k E + \nabla^2 f(x_k))^{-1} \nabla f(x_k)$, wobei E die Einheitsmatrix in $\mathbb{R}^{n \times n}$ ist und δ_k die kleinste positive Zahl ist, für die die Eigenwerte von $(\delta_k E + \nabla^2 f(x_k))$ größer als der Einstellungsparameter δ sind. Passende Werte des Parameters δ sind problemspezifisch und können durch experimentieren bestimmt werden. Ein kleiner Wert von δ bedeutet, dass eine beinahe singuläre Matrix invertiert werden muss; ein großer Wert von δ würde die Konverzgenzordnung schwächen.

In jeder Iteration soll δ_k mit Hilfe der folgenden Prozedur approximiert werden: Starte mit $\delta_k = 0$ und überprüfe ob alle Eigenwerte der Matrix $(\delta_k E + \nabla^2 f(x_k))$ mindestens δ sind. Wenn das nicht der Fall ist erhöhe δ_k: $\delta_k := \delta_k + \mu$, wobei μ ebenfalls ein vorsichtig zu wählender Steuerungsparameter ist.

Die Schrittweite α_k soll durch das unten beschriebene Verfahren auf Basis der Armijo bzw. Wolfe Bedingung (das gleiche wie bei der dritten Implementierungsaufgabe) wie folgt ermittelt werden:
Eine Schrittweite $\alpha > 0$ erfüllt die Armijo-Bedingung (A) im k-ten Schritt, wenn

$$f(x_k + \alpha d_k) \leq f(x_k) - \lambda \alpha ||d_k||^2$$

und die Wolfe Bedingung (W), wenn

$$\nabla f(x_k + \alpha d_k)^t d_k \geq -(1 - \rho)||d_k||^2.$$

Typische Einstellungen für die Steuerungsparameter λ und ρ sind z.B.: $\lambda = 0,2$; $\lambda = 0,3$; $\rho = 0,1$; $\rho = 0,2$.

Die Schrittweite α_k wird durch das folgende Bisektionsverfahren bestimmt: Gestartet wird mit dem Suchintervall $[0, \bar{\alpha}_k]$, mit

$$\bar{\alpha}_k = \min_{i \in \mathbb{N}} \{\eta^i \mid f(x_k + \eta^i d_k) > f(x^k) - \lambda \eta^i ||d^k||^2\}.$$

Sei $[a_0, a_1]$ das aktuelle Suchintervall für α_k. Betrachte nun $a := (a_0 + a_1)/2$.

- Wenn a die Bedingungen (A) und (W) erfüllt, dann breche ab und setze $\alpha_k := a$.
- Wenn a die Bedingung (A) erfüllt, aber nicht die Bedingung (W), dann setze mit dem Suchintervall $[a, a_1]$ fort.
- Wenn a die Bedingung (A) nicht erfüllt, setze mit $[a_0, a]$ fort.

Typische Einstellungen für den Steuerungsparameter η sind z.B.: $\eta = 1,1$; $\eta = 1,2$.

Abbruchkriterien. Abgebrochen werden sollte wenn eine der folgenden Bedingungen erfüllt ist:

- $||\nabla f(x_k)|| \leq \varepsilon$. Dann soll $x = x_k$ gesetzt werden und der Fallparameter `Fall` auf 0 gesetzt werden.
- wenn die Iterationszahl k über $maxit$ steigt, soll mit $x = x_k$ abgebrochen werden und der Fallparameter `Fall` auf 1 (zuviele Iterationen) gesetzt werden.

- $||x_k|| \geq M$, dann soll mit $x = x_k$ und `Fall=2` abgebrochen werden (möglicherweise unbeschränktes Problem).
- wenn die Anzahl der Iterationen l_k, die für die Bestimmung von δ_k in Iteration k über *maxit_delta* steigt, soll mit $x = x_k$ abgebrochen werden und der Fallparameter `Fall` auf 3 gesetzt werden (zuviele Iterationen zur Bestimmung von δ_k).
- wenn die Armijo-Suche zur Bestimmung von α_k nach *maxit_alpha* Iterationen nicht konvergiert, dann soll mit $x = x_k$ abgebrochen werden und der Fallparameter `Fall` auf 4 gesetzt werden (Armijo-Suche konvergiert nicht).

In jedem Fall soll auch die Anzahl der durchgeführten Iterationen *nr_it* , also der Wert von k bei Abbruch, ausgegeben werden. Als Norm soll die Euklidische Norm verwendet werden.

Das Ergebnis soll eine Funktion sein der Form
`[x Fall nr_it] = myGenNewton(xinit, delta, mu, lambda, rho, eta, epsilon, M, maxit, maxit_delta, maxit_alpha, funct, gradient, HesseMat, smallestEV)`.

`xinit` soll ein Spaltenvektor sein, der x_0 entspricht. `maxit` steht für die maximale Anzahl von Iterationen. `funct` ist ein Funktions-Handle einer Funktion, die bei Aufruf von `funct(x)` den Wert $f(x)$ zurückgibt. `gradient` ist ein Funktions-Handle einer Funktion, die bei Aufruf von `gradient(x)` den Wert $\nabla f(x)$ zurückgibt. `HesseMat` ist ein Funktions-Handle einer Funktion, die bei Aufruf von `HesseMat(x)` den Wert $\nabla^2 f(x)$ zurückgibt. `smallestEV` ist ein Funktions-Handle einer Funktion, die bei Aufruf von `smallestEV(x,`δ_k`)` den kleinsten Eigenwert von $(\delta_k E + \nabla^2 f(x))$ zurückgibt.

Falls Sie eine Funktion `bspf(x)` in einem eigenen Funktionsfile `bspf.m` definieren, erhält man das Funktions-Handle mittels `@bspf`. Auf diese Art kann die Funktion unserer Funktion `myGenNewton` übergeben werden. Einfachere Funktionen können in einem Matlab Script auch anonym definiert werden (z.B. `bspf = @(x) 2*x+1`). In diesem Fall ist in der Variable `bspf` bereits ein Funktions-Handle auf die definierte anonyme Funktion gespeichert, und `bspf` kann direkt an `myGenNewton` übergeben werden. Damit lässt sich der Code modular anwenden.

Testen Sie Ihr Programm für verschiedene Funktionen und unterschiedliche Einstellungen der Steuerungsparameter δ, μ, λ, ρ, η, ε, M, *maxit*, *maxit_delta*, *maxit_alpha*, sowie unterschiedliche Startpunkte x_0. Vergewissern Sie sich zB., dass Ihr Programm auf jeden Fall für jede der folgenden Funktionen (dieselben wie bei der dritten Implementierungsuafgabe) das Minimum korrekt ermittelt, falls es existiert, und andernfalls die Unbeschränktheit erkennt.

- Rosenbrock Funktion (siehe 10. Übungsblatt)
- quadratische Funktionen der Form $\frac{1}{2}x^tQx - b^tx$ mit

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 25 \end{pmatrix} \text{ und } b = (-1,-1,-1)^t$$

  oder

$$Q = \begin{pmatrix} 14 & 9 & -1 \\ 9 & 18 & 6 \\ -1 & 6 & 5 \end{pmatrix} \text{ und } b = (3,9,6)^t.$$

- $f(x) = -\exp(-||x||^2)$ mit $f\colon \mathbb{R}^2 \to \mathbb{R}$ (verschiedene x_0 testen).

Darüberhinaus sollen Sie Ihren Code für (mindestens) 3 weitere, von Ihnen erzeugte Minimierungsprobleme wie folgt testen: ein strikt konvexes quadratisches Problem, ein nicht quadratisches Problem mit endlichem Minimum und ein unbeschränktes Problem. Erstellen Sie ein m-File `testNewton.m`, das 7

Test-Instanzen erzeugt (die vier vorgegeben und die drei von Ihnen definierten) und durch den Aufruf von `myGenNewton` (approximativ) löst bzw. die Unbeschränktheit feststellt. `testNewton.m` sollte mitabgegeben werden.

Achtung: Bitte achten Sie darauf, dass die Funktion `myGenNewton` in der abgegebenen Version keine Debug-Ausgaben auf der Konsole macht.

Abgabe: Die Abgabe hat via TeachCenter in Form einer ZIP-Datei zu erfolgen. Die ZIP-Datei sollte alle notwendigen Files enthalten, also im Speziellen auch ein File `myGenNewton.m` mit der obengenannten Funktion, sowie ein File `testNewton.m` (siehe oben). Die Abgabefrist ist Dienstag, der 2.7.2017, 23:55. Benennen Sie Ihre Zip-Datei in der Art `steep+Nachname+1.Buchstabe des Vornamen.zip` (lauter Kleinbuchstaben) - also z.B. bei Lena Musterfrau würde der Dateiname `steepmusterfraul.zip` lauten.