تکلیف سری چهارم
درس روشهای عددی بهینه‌سازی

مسعود بابایی‌زاده

۱ هدف

در این تکلیف هدف آن است که روش‌های Steepest Descent و نیوتن همراه با Line Search ایده‌آل را پیاده‌سازی نماییم. مرحله Line Search در این تکلیف با استفاده از آلگوریتم Golden Section Search و تابع GSS.m که در تکلیف اول درس پیاده‌سازی کرده‌اید انجام می‌گیرد.

مشابه تکلیف اول، این آلگوریتمها را باید به صورت کلی بنویسید، یعنی باید دو تابع SD.m و Newton.m بنویسید به طوریکه نام تابعی که باید حداقل شود (و نیز گرادیان و هسین آن) به صورت یک پارامتر (با استفاده از handle) به این توابع داده شود. به این ترتیب می‌توان از این mفایل‌ها برای حداقل کردن هر تابعی استفاده کرد.

بنابراین آرگومانهای ورودی و خروجی این توابع به صورت زیر هستند:

```
[x_min, f_min, iter] = SD_GSS(f, gf, x0, Stop_tol, GSS_tol, varargin)
[x_min, f_min, iter] = Newton_GSS(f, gf, Hf, x0, Stop_tol, GSS_tol, varargin)
```

که در آن:

- `f`، `gf` و `Hf` به ترتیب handle به تابعی که باید حداقل شود، گرادیان آن و هسین آن.
- `x0` نقطه شروع آلگوریتم است.
- `Stop_tol` شرط توقف آلگوریتم است، یعنی آلگوریتم وقتی متوقف می‌شود که $\|\mathbf{x}_{k+1} - \mathbf{x}_k\| \leq$ `Stop_tol` شود. البته شرطهای دیگری هم می‌توان گذاشت (که احتمالاً بهتر هم هستند) که در درس بیان شده است. ولی در این تکلیف از این شرط استفاده می‌کنیم. اما بهتر است شرطهای دیگر را نیز برای خودتان بررسی کنید.
- `GSS_tol` شرط توقف آلگوریتم Golden Section Search در مرحله Line Search است (همان تلرانس در تکلیف سری اول).
- `varargin` بقیه پارامترهای احتمالی‌ای که تابع f به آنها نیاز دارد (مشابه تکلیف سری اول).
- خروجی‌های تابع: `x_min` حداقل‌کننده تابع f و `f_min` مقدار تابع f در نقطه حداقل‌کننده است. `iter` نیز تعداد iterationهایی است که آلگوریتم برای یافتن نقطه حداقل‌کننده انجام داده است.

۲ ارزیابی آلگوریتم نوشته‌شده

پس از اینکه آلگوریتمهای SD و نیوتن را پیاده‌سازی کردید، از آنها برای حداقل‌کردن توابع زیر استفاده کنید. بدیهی است که نقطه حداقل‌کننده تئوریک هر دو تابع زیر $\mathbf{x}^* = \mathbf{0}$ (مبدا مختصات) است.

- تابع Rosenbrock (بخصوص به عملکرد روش SD در حداقل‌کردن این تابع دقت کنید):

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

- تابع Powel (توجه کنید که توان در دو جمله اول ۲ و در دو جمله آخر ۴ است):

$$f(x_۱, x_۲, x_۳, x_۴) = (x_۱ + ۱۰x_۲)^۲ + ۵(x_۳ - x_۴)^۲ + (x_۲ - ۲x_۳)^۴ + ۱۰(x_۱ - x_۴)^۴$$

۳ پارامترها

برای اینکه همه جوابهای مشابهی بگیرند و کیفیت آلگوریتم‌های نوشته‌شده توسط دانشجویان مختلف قابل مقایسه باشد، پارامترهای مختلف مربوط به این آلگوریتم‌ها را به شرح زیر تعیین می‌شوند:

- تلرانس روش Golden Section Search برای مرحله Line Search:

$$\texttt{GSS\_tol} = ۰/۰۰۰۰۱ = ۱۰^{-۵}$$

(مقادیر دیگری نیز برای خودتان امتحان کنید، بخصوص برای SD در تابع Rosenbrock).

- نقطه شروع: برای تابع روزنبروک $\mathbf{x}_۰ = (۱, ۲)^T$ و برای تابع پاول $\mathbf{x}_۰ = (۱, ۲, ۲, ۲)^T$.
- شرط توقف آلگوریتم نهایی: $\|\mathbf{x}_{k+۱} - \mathbf{x}_k\| \leq ۰/۰۰۱ = ۱۰^{-۳}$، یعنی $\texttt{Stop\_tol} = ۱۰^{-۳}$.

۴ خروجی‌هایی که باید تحویل داده شود

علاوه بر توابع SD_GSS.m و Newton_GSS.m باید دو جدول (یکی برای آلگوریتم SD و دیگری برای آلگوریتم نیوتن) به صورت زیر تهیه کرده و آنها را نیز به فرمت Word یا pdf تحویل دهید (در صورت تمایل می‌توانید جداول مشابهی نیز برای Stop_tolهای متفاوت تهیه کنید):

	$\mathbf{x}$ نهایی	f نهایی	تعداد iteration	تعداد function evaluation	تعداد gradient evaluation	تعداد Hessian evaluation
تابع پاول						
تابع روزنبروک						

توجه ۱. برای شمردن تعداد function evaluation (و نیز gradient evaluation و Hessian evaluation)، روش درست آن است که در هر کدام از توابع مربوطه (f، gf و Hf) متغیری از نوع static (مثلاً به نامهای f_eval، gf_eval و Hf_eval) تعریف کنید و در ابتدا مقدار آنها را صفر قرار دهید. سپس اولین سطر تابع هم اضافه‌کردن یک واحد به مقدار این متغیر باشد. احتمالاً باید مکانیزمی نیز در تابع قرار دهید که بتوانید هروقت خواستید مقدار این متغیر را از تابع بخوانید[۱]. به این روش همواره می‌توانید تعداد دقیق function evaluationها را داشته باشید. توجه کنید که اندازه‌گیری تعداد function evaluation از روی تعداد iterationهای آلگوریتم حداقل‌سازی و ... روش مناسبی نیست، چون به سادگی احتمال اشتباه وجود دارد. با روش گفته‌شده، شمارش تعداد دفعات را به عهده خود تابع گذاشته‌اید و عدد مطمئنی به دست خواهید آورد. توجه شود که هر سه فانکشنی که خود تابع، گرادیان آن و هسین آن را حساب می‌کنند، باید بتوانند تعداد دفعات فراخوانی خود را بشمارند.

توجه ۲. چون هیچ چیز تصادفی وجود ندارد و نقطه شروع آلگوریتم و تلرانس‌ها برای همه دانشجویان یکسان است، پس قاعدتاً اعداد جدول فوق باید برای همه یکسان باشد. بنابراین نتایجی را که بدست آورده‌اید با دوستانتان مقایسه کنید. در صورتی که مثلاً تعداد function evaluationهای شما خیلی زیاد شده، به این معنی است که برنامه‌تان را خوب ننوشته‌اید (مثلاً عددی را که قبلاً محاسبه کرده بودید، دوباره در جای دیگری محاسبه کرده‌اید).

[۱] البته راه دیگر استفاده از متغیرهایی global برای شمردن function evaluation ها است، که اطمینان کمتری نسبت به شمارش با استفاده از متغیرهای static در درون تابع دارد، چون باز هم ممکن است در جایی خارج از تابع مقدار آن در اثر اشتباه تغییر کند. ولی از نظر برنامه‌نویسی آسان‌تر است.