AM 205: lecture 17

- Last time: introduction to optimization
- Today: scalar and vector optimization
- Note: last year’s midterm is now available on the website for practice
Fixed-Point Iteration

Suppose \( \alpha \) is such that \( g(\alpha) = \alpha \), then we call \( \alpha \) a fixed point of \( g \).

For example, we see that \( \sqrt{a} \) is a fixed point of \( g_{\text{heron}} \) since

\[
g_{\text{heron}}(\sqrt{a}) = \frac{1}{2} \left( \sqrt{a} + \frac{a}{\sqrt{a}} \right) = \sqrt{a}
\]

A fixed-point iteration terminates once a fixed point is reached, since if \( g(x_k) = x_k \) then we get \( x_{k+1} = x_k \).

Also, if \( x_{k+1} = g(x_k) \) converges as \( k \to \infty \), it must converge to a fixed point: Let \( \alpha \equiv \lim_{k \to \infty} x_k \), then\(^1\)

\[
\alpha = \lim_{k \to \infty} x_{k+1} = \lim_{k \to \infty} g(x_k) = g \left( \lim_{k \to \infty} x_k \right) = g(\alpha)
\]

\(^1\)Third equality requires \( g \) to be continuous
Hence, for example, we know if Heron’s method converges, it will converge to $\sqrt{a}$.

It would be very helpful to know when we can guarantee that a fixed-point iteration will converge.

Recall that $g$ satisfies a Lipschitz condition in an interval $[a, b]$ if $\exists L \in \mathbb{R}_{>0}$ such that

$$|g(x) - g(y)| \leq L|x - y|, \quad \forall x, y \in [a, b]$$

$g$ is called a contraction if $L < 1$. 
Fixed-Point Iteration

**Theorem:** Suppose that \( g(\alpha) = \alpha \) and that \( g \) is a contraction on \([\alpha - A, \alpha + A]\). Suppose also that \( |x_0 - \alpha| \leq A \). Then the fixed point iteration converges to \( \alpha \).

**Proof:**

\[
|x_k - \alpha| = |g(x_{k-1}) - g(\alpha)| \leq L|x_{k-1} - \alpha|,
\]

which implies

\[
|x_k - \alpha| \leq L^k|x_0 - \alpha|
\]

and, since \( L < 1 \), \( |x_k - \alpha| \to 0 \) as \( k \to \infty \). (Note that \( |x_0 - \alpha| \leq A \) implies that all iterates are in \([\alpha - A, \alpha + A]\).) □

(This proof also shows that error decreases by factor of \( L \) each iteration)
Recall that if $g \in C^1[a, b]$, we can obtain a Lipschitz constant based on $g'$:

$$L = \max_{\theta \in (a, b)} |g'(\theta)|$$

We now use this result to show that if $|g'(\alpha)| < 1$, then there is a neighborhood of $\alpha$ on which $g$ is a contraction.

This tells us that we can verify convergence of a fixed point iteration by checking the gradient of $g$. 
By continuity of $g'$ (and hence continuity of $|g'|$), for any $\epsilon > 0$ \exists \delta > 0 such that for $x \in (\alpha - \delta, \alpha + \delta)$:

$$| |g'(x)| - |g'(\alpha)| | \leq \epsilon \implies \max_{x \in (\alpha - \delta, \alpha + \delta)} |g'(x)| \leq |g'(\alpha)| + \epsilon$$

Suppose $|g'(\alpha)| < 1$ and set $\epsilon = \frac{1}{2}(1 - |g'(\alpha)|)$, then there is a neighborhood on which $g$ is Lipschitz with $L = \frac{1}{2}(1 + |g'(\alpha)|)$

Then $L < 1$ and hence $g$ is a contraction in a neighborhood of $\alpha$
Furthermore, as $k \to \infty$, 

$$\frac{|x_{k+1} - \alpha|}{|x_k - \alpha|} = \frac{|g(x_k) - g(\alpha)|}{|x_k - \alpha|} \to |g'(\alpha)|,$$

Hence, asymptotically, error decreases by a factor of $|g'(\alpha)|$ each iteration.
We say that an iteration converges \textit{linearly} if, for some $\mu \in (0, 1)$,

$$\lim_{k \to \infty} \frac{|x_{k+1} - \alpha|}{|x_k - \alpha|} = \mu$$

An iteration converges \textit{superlinearly} if

$$\lim_{k \to \infty} \frac{|x_{k+1} - \alpha|}{|x_k - \alpha|} = 0$$
Fixed-Point Iteration

We can use these ideas to construct practical fixed-point iterations for solving $f(x) = 0$

e.g. suppose $f(x) = e^x - x - 2$

From the plot, it looks like there’s a root at $x \approx 1.15$
Fixed-Point Iteration

\( f(x) = 0 \) is equivalent to \( x = \log(x + 2) \), hence we seek a fixed point of the iteration

\[
x_{k+1} = \log(x_k + 2), \quad k = 0, 1, 2, \ldots
\]

Here \( g(x) \equiv \log(x + 2) \), and \( g'(x) = 1/(x + 2) < 1 \) for all \( x > -1 \), hence fixed point iteration will converge for \( x_0 > -1 \)

Hence we should get linear convergence with factor approx. \( g'(1.15) = 1/(1.15 + 2) \approx 0.32 \)
An alternative fixed-point iteration is to set

\[ x_{k+1} = e^{x_k} - 2, \quad k = 0, 1, 2, \ldots \]

Therefore \( g(x) \equiv e^x - 2 \), and \( g'(x) = e^x \)

Hence \( |g'(x)| > 1 \), so we can’t guarantee convergence

(And, in fact, the iteration diverges...)
Python demo: Comparison of the two iterations
Newton’s Method

Constructing fixed-point iterations can require some ingenuity

Need to rewrite \( f(x) = 0 \) in a form \( x = g(x) \), with appropriate properties on \( g \)

To obtain a more generally applicable iterative method, let us consider the following fixed-point iteration

\[
x_{k+1} = x_k - \lambda(x_k)f(x_k), \quad k = 0, 1, 2, \ldots
\]

corresponding to \( g(x) = x - \lambda(x)f(x) \), for some function \( \lambda \)

A fixed point \( \alpha \) of \( g \) yields a solution to \( f(\alpha) = 0 \) (except possibly when \( \lambda(\alpha) = 0 \)), which is what we’re trying to achieve!
Newton’s Method

Recall that the asymptotic convergence rate is dictated by $|g'(\alpha)|$, so we’d like to have $|g'(\alpha)| = 0$ to get superlinear convergence.

Suppose (as stated above) that $f(\alpha) = 0$, then

$$g'(\alpha) = 1 - \lambda'(\alpha)f(\alpha) - \lambda(\alpha)f'(\alpha) = 1 - \lambda(\alpha)f'(\alpha)$$

Hence to satisfy $g'(\alpha) = 0$ we choose $\lambda(x) \equiv 1/f'(x)$ to get Newton’s method:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \ldots$$
Newton’s Method

Based on fixed-point iteration theory, Newton’s method is convergent since $|g'(\alpha)| = 0 < 1$

However, we need a different argument to understand the superlinear convergence rate properly

To do this, we use a Taylor expansion for $f(\alpha)$ about $f(x_k)$:

$$0 = f(\alpha) = f(x_k) + (\alpha - x_k)f'(x_k) + \frac{(\alpha - x_k)^2}{2}f''(\theta_k)$$

for some $\theta_k \in (\alpha, x_k)$
Newton’s Method

Dividing through by $f'(x_k)$ gives

$$
\left( x_k - \frac{f(x_k)}{f'(x_k)} \right) - \alpha = \frac{f''(\theta_k)}{2f'(x_k)} (x_k - \alpha)^2,
$$

or

$$
x_{k+1} - \alpha = \frac{f''(\theta_k)}{2f'(x_k)} (x_k - \alpha)^2,
$$

Hence, roughly speaking, the error at iteration $k + 1$ is the square of the error at each iteration $k$

This is referred to as quadratic convergence, which is very rapid!

Key point: Once again we need to be sufficiently close to $\alpha$ to get quadratic convergence (result relied on Taylor expansion near $\alpha$)
An alternative to Newton’s method is to approximate $f'(x_k)$ using the finite difference

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

Substituting this into the iteration leads to the secant method

$$x_{k+1} = x_k - f(x_k) \left( \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \right), \quad k = 1, 2, 3, \ldots$$

The main advantages of secant are:

- does not require us to determine $f'(x)$ analytically
- requires only one extra function evaluation, $f(x_k)$, per iteration (Newton’s method also requires $f'(x_k)$)
As one may expect, secant converges faster than a fixed-point iteration, but slower than Newton’s method.

In fact, it can be shown that for the secant method, we have

$$\lim_{k \to \infty} \frac{|x_{k+1} - \alpha|}{|x_k - \alpha|^q} = \mu$$

where $\mu$ is a positive constant and $q \approx 1.6$

Python demo: Newton’s method versus secant method for $f(x) = e^x - x - 2 = 0$
Multivariate Case
We now consider fixed-point iterations and Newton’s method for systems of nonlinear equations

We suppose that $F : \mathbb{R}^n \to \mathbb{R}^n$, $n > 1$, and we seek a root $\alpha \in \mathbb{R}^n$ such that $F(\alpha) = 0$

In component form, this is equivalent to

\[
\begin{align*}
F_1(\alpha) &= 0 \\
F_2(\alpha) &= 0 \\
&\quad \vdots \\
F_n(\alpha) &= 0
\end{align*}
\]
For a fixed-point iteration, we again seek to rewrite $F(x) = 0$ as $x = G(x)$ to obtain:

$$x_{k+1} = G(x_k)$$

The convergence proof is the same as in the scalar case, if we replace $|·|$ with $∥·∥$

i.e. if $∥G(x) − G(y)∥ \leq L∥x − y∥$, then $∥x_k − α∥ \leq L^k∥x_0 − α∥$

Hence, as before, if $G$ is a contraction it will converge to a fixed point $α$
Recall that we define the Jacobian matrix, $J_G \in \mathbb{R}^{n \times n}$, to be

$$(J_G)_{ij} = \frac{\partial G_i}{\partial x_j}, \quad i, j = 1, \ldots, n$$

If $\|J_g(\alpha)\|_{\infty} < 1$, then there is some neighborhood of $\alpha$ for which the fixed-point iteration converges to $\alpha$

The proof of this is a natural extension of the corresponding scalar result
Fixed-Point Iteration

Once again, we can employ a fixed point iteration to solve $F(x) = 0$

e.g. consider

$$x_1^2 + x_2^2 - 1 = 0$$
$$5x_1^2 + 21x_2^2 - 9 = 0$$

This can be rearranged to $x_1 = \sqrt{1 - x_2^2}$, $x_2 = \sqrt{(9 - 5x_1^2)/21}$
Hence, we define

\[ G_1(x_1, x_2) \equiv \sqrt{1 - x_2^2}, \quad G_2(x_1, x_2) \equiv \sqrt{(9 - 5x_1^2)/21} \]

**Python Example:** This yields a convergent iterative method
Newton’s Method

As in the one-dimensional case, Newton’s method is generally more useful than a standard fixed-point iteration.

The natural generalization of Newton’s method is

\[ x_{k+1} = x_k - J_F(x_k)^{-1} F(x_k), \quad k = 0, 1, 2, \ldots \]

Note that to put Newton’s method in the standard form for a linear system, we write

\[ J_F(x_k) \Delta x_k = -F(x_k), \quad k = 0, 1, 2, \ldots, \]

where \( \Delta x_k \equiv x_{k+1} - x_k \)
Newton’s Method

Once again, if \( x_0 \) is sufficiently close to \( \alpha \), then Newton’s method converges quadratically — we sketch the proof below.

This result again relies on Taylor’s Theorem.

Hence we first consider how to generalize the familiar one-dimensional Taylor’s Theorem to \( \mathbb{R}^n \).

First, we consider the case for \( F : \mathbb{R}^n \to \mathbb{R} \).
Multivariate Taylor Theorem

Let \( \phi(s) \equiv F(x + s\delta) \), then one-dimensional Taylor Theorem yields

\[
\phi(1) = \phi(0) + \sum_{\ell=1}^{k} \frac{\phi^{(\ell)}(0)}{\ell!} + \phi^{(k+1)}(\eta), \quad \eta \in (0, 1),
\]

Also, we have

\[
\begin{align*}
\phi(0) &= F(x) \\
\phi(1) &= F(x + \delta) \\
\phi'(s) &= \frac{\partial F(x + s\delta)}{\partial x_1} \delta_1 + \frac{\partial F(x + s\delta)}{\partial x_2} \delta_2 + \cdots + \frac{\partial F(x + s\delta)}{\partial x_n} \delta_n \\
\phi''(s) &= \frac{\partial^2 F(x + s\delta)}{\partial x_1^2} \delta_1^2 + \cdots + \frac{\partial^2 F(x + s\delta)}{\partial x_1 x_n} \delta_1 \delta_n + \cdots + \\
&\quad \frac{\partial^2 F(x + s\delta)}{\partial x_1 \partial x_n} \delta_1 \delta_n + \cdots + \frac{\partial^2 F(x + s\delta)}{\partial x_n^2} \delta_n^2 \\
&\quad \vdots
\end{align*}
\]
Hence, we have

\[ F(x + \delta) = F(x) + \sum_{\ell=1}^{k} \frac{U_{\ell}(\delta)}{\ell!} + E_k, \]

where

\[ U_{\ell}(x) \equiv \left[ \left( \frac{\partial}{\partial x_1} \delta_1 + \cdots + \frac{\partial}{\partial x_n} \delta_n \right)^\ell F \right](x), \quad \ell = 1, 2, \ldots, k, \]

and

\[ E_k \equiv U_{k+1}(x + \eta \delta), \quad \eta \in (0, 1) \]
Multivariate Taylor Theorem

Let $A$ be an upper bound on the abs. values of all derivatives of order $k + 1$, then

$$|E_k| \leq \frac{1}{(k + 1)!} \left|(A, \ldots, A)^T (\|\delta\|_\infty^{k+1}, \ldots, \|\delta\|_\infty^{k+1})\right|$$

$$= \frac{1}{(k + 1)!} A\|\delta\|_\infty^{k+1} \left|(1, \ldots, 1)^T (1, \ldots, 1)\right|$$

$$= \frac{n^{k+1}}{(k + 1)!} A\|\delta\|_\infty^{k+1}$$

where the last line follows from the fact that there are $n^{k+1}$ terms in the inner product (i.e. there are $n^{k+1}$ derivatives of order $k + 1$)
Multivariate Taylor Theorem

We shall only need an expansion up to first order terms for analysis of Newton’s method

From our expression above, we can write first order Taylor expansion succinctly as:

\[ F(x + \delta) = F(x) + \nabla F(x)^T \delta + E_1 \]
Multivariate Taylor Theorem

For $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$, Taylor expansion follows by developing a Taylor expansion for each $F_i$, hence

$$F_i(x + \delta) = F_i(x) + \nabla F_i(x)^T \delta + E_{i,1}$$

so that for $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ we have

$$F(x + \delta) = F(x) + J_F(x)\delta + E_F$$

where $\|E_F\|_\infty \leq \max_{1 \leq i \leq n} |E_{i,1}| \leq \frac{1}{2} n^2 \left( \max_{1 \leq i,j,\ell \leq n} \left| \frac{\partial^2 F_i}{\partial x_j \partial x_\ell} \right| \right) \|\delta\|_\infty^2$
Newton’s Method

We now return to Newton’s method

We have

$$0 = F(\alpha) = F(x_k) + J_F(x_k) [\alpha - x_k] + E_F$$

so that

$$x_k - \alpha = [J_F(x_k)]^{-1} F(x_k) + [J_F(x_k)]^{-1} E_F$$
Newton’s Method

Also, the Newton iteration itself can be rewritten as

\[ J_F(x_k) [x_{k+1} - \alpha] = J_F(x_k) [x_k - \alpha] - F(x_k) \]

Hence, we obtain:

\[ x_{k+1} - \alpha = [J_F(x_k)]^{-1} E_F, \]

so that \( \|x_{k+1} - \alpha\|_\infty \leq \text{const.} \|x_k - \alpha\|_\infty^2 \), i.e. quadratic convergence!
Newton’s Method

Example: Newton’s method for the two-point Gauss quadrature rule

Recall the system of equations

\[
\begin{align*}
F_1(x_1, x_2, w_1, w_2) &= w_1 + w_2 - 2 = 0 \\
F_2(x_1, x_2, w_1, w_2) &= w_1x_1 + w_2x_2 = 0 \\
F_3(x_1, x_2, w_1, w_2) &= w_1x_1^2 + w_2x_2^2 - 2/3 = 0 \\
F_4(x_1, x_2, w_1, w_2) &= w_1x_1^3 + w_2x_2^3 = 0
\end{align*}
\]
Newton’s Method

We can solve this in Python using our own implementation of Newton’s method.

To do this, we require the Jacobian of this system:

\[
J_F(x_1, x_2, w_1, w_2) = \begin{bmatrix}
0 & 0 & 1 & 1 \\
w_1 & w_2 & x_1 & x_2 \\
2w_1x_1 & 2w_2x_2 & x_1^2 & x_2^2 \\
3w_1x_1^2 & 3w_2x_2^2 & x_1^3 & x_2^3 \\
\end{bmatrix}
\]
Newton’s Method

Alternatively, we can use Python’s built-in \texttt{fsolve} function

\textbf{Note that} \texttt{fsolve} computes a finite difference approximation to the Jacobian by default

\textit{(Or we can pass in an analytical Jacobian if we want)}

\textbf{Matlab} has an equivalent \texttt{fsolve} function.
Newton’s Method

**Python example:** With either approach and with starting guess $x_0 = [-1, 1, 1, 1]$, we get

```
x_k =
-0.577350269189626
  0.577350269189626
  1.000000000000000
  1.000000000000000
```