Math 311

Numerical Methods

2.5: Accelerating Convergence

Solutions of Equations of One Variable

S. K. Hyde

Burden and Faires, any ed.

Winter 2024

1 Introduction

- It is rare to have quadratic convergence (but they do exist). Search online.
- We will illustrate two methods to speed convergence:
  - Aitkin's Δ^2 method
  - Steffensen's Method.

1.1 Aitken's Method

- Suppose we have a linearly converging sequence $p_n \to p$ as $n \to \infty$.
- It stands to reason that as n gets large,

$$\frac{|\, p_{n+1} - p \,|}{|\, p_n - p \,|} \approx \frac{|\, p_{n+2} - p \,|}{|\, p_{n+1} - p \,|} \tag{1}$$

- We can solve (1) for p! To make the algebra easier, we will let $n = 0$ in the equation.

- Let's go:

$$\frac{|p_1 - p|}{|p_0 - p|} \approx \frac{|p_2 - p|}{|p_1 - p|} \qquad \text{(Cross multiply)}$$

$$(p_1 - p)^2 \approx (p_2 - p)(p_0 - p)$$

$$p_1^2 - 2p_1 p + \cancel{p^2} \approx p_2 p_0 - p_2 p - p_0 p + \cancel{p^2} \qquad \text{(Now isolate } p\text{)}$$

$$p_2 p + p_0 p - 2p_1 p \approx p_2 p_0 - p_1^2$$

$$(p_2 - 2p_1 + p_0)p \approx p_2 p_0 - p_1^2$$

$$p \approx \frac{p_2 p_0 - p_1^2}{p_2 - 2p_1 + p_0}$$

- To make it easier to use, we're going to do an add zero trick:

$$p \approx \frac{p_2 p_0 - p_1^2 \overbrace{-2p_0 p_1 + 2p_0 p_1 + p_0^2 - p_0^2}^{\text{zero}}}{p_2 - 2p_1 + p_0}$$

- Now rearrange

$$p \approx \frac{p_2 p_0 - 2p_0 p_1 + p_0^2 - (p_1^2 - 2p_0 p_1 + p_0^2)}{p_2 - 2p_1 + p_0}$$

$$p \approx \frac{(p_2 - 2p_1 + p_0)p_0 - (p_1 - p_0)^2}{p_2 - 2p_1 + p_0}$$

$$p \approx p_0 - \frac{(p_1 - p_0)^2}{p_2 - 2p_1 + p_0}$$

- You should be able to replicate this on an exam.
- So, let $\hat{p}_n = p$. Aitken's Method defined as

Aitken's Δ^2 Method

Definition.

$$\hat{p}_n = p_n - \frac{(p_{n+1} - p_n)^2}{p_{n+2} - 2p_{n+1} + p_n}$$

- The sequence defined above converges faster than the original p_n sequence in the following sense:

Convergence speed of $\hat{p}_n$

Theorem. *Let $\{\boldsymbol{p_n}\}_{\boldsymbol{n=0}}^{\boldsymbol{\infty}}$ be a sequence that converges linearly to the limit p with asymptotic constant less than 1 and $p_n - p \neq 0$ for all $n \geqslant 0$. Then the sequence $\{\hat{p}_n\}$ converges to p faster than $\{\boldsymbol{p_n}\}_{\boldsymbol{n=0}}^{\boldsymbol{\infty}}$ in the sense that*

$$\lim_{n\to\infty} \frac{\hat{p}_n - p}{p_n - p} = 0$$

Forward Difference Δp_n

Definition. *Given the sequence $\{\boldsymbol{p_n}\}_{\boldsymbol{n=0}}^{\boldsymbol{\infty}}$, define the **forward difference** Δp_n by*

$$\Delta p_n = p_{n+1} - p_n, \ \textit{for } n \geqslant 0$$

Higher powers $\Delta^k p_n$ are defined recursively as

$$\Delta^k p_n = \Delta(\Delta^{k-1} p_n), \ \textit{for } k \geqslant 2$$

- As a result of the definition, we have

$$\begin{aligned}\Delta^2 p_n = \Delta(p_{n+1} - p_n) = \Delta p_{n+1} - \Delta p_n &= (p_{n+2} - p_{n+1}) - (p_{n+1} - p_n) \\ &= p_{n+2} - 2p_{n+1} - p_n\end{aligned}$$

- We can redefine Aitken's $\Delta^2 p_n$ using this operator:

Aitken's Δ^2 Method

Definition.

$$\hat{p}_n = p_n - \frac{(\Delta p_n)^2}{\Delta^2 p_n}, \text{ for all } n \geqslant 0$$

- By applying Aitken's Δ^2 method, we can accelerate the original p_n.
- Note that the way Aitkens works is that it takes in a vector (a sequence of numbers) and outputs a new sequence of numbers (with length reduced by 2).
- For example, let's speed up the original fixed point function we tried ($f(x) = \cos x$)
- The R function for implementing Aitkens is

```
aitkens = function(p) {
  ## Inputs:    p = sequence of numbers (a vector)
  ## Outputs:   phat = sequence of numbers accelerated by Aitken's method
  delta = function(p) { p[1]-(p[2]-p[1])^2/(p[3]-2*p[2]+p[1]) }
  phat = 0
  for (i in 1:(length(p)-2)) phat[i] = delta(p[i:(i+2)])
  return(phat)
}
```

- `g = function(x) { cos(x) }`

- Start with initial guess of $x = 1$ and $n = 30$ for iterations.

```
iter = 30 #number of iterations
fixpt = 1 #fixpt=initial guess; values after will be added
```

- Now a simple fixed pt loop (Look to code `root_fixedpt.r` for prettier code.
- To perform fixed point on g do:

```
for (i in 2:iter) fixpt[i] = g(fixpt[i-1]) #fast-no check-fixed point method
```

- The variable `fixedpt` contains the sequence p_n. Let's accelerate it using `aitkens`.

```
phat = aitkens(fixpt)
```

- Compare the fixpt method with Aitkens (NA is added to end of phat so they have the same size)

```
p=1; for (i in 2:300) p=g(p)  # find the fixed pt to analyze the absolute error
compare = cbind(fixpt,abs(fixpt-p),c(phat,NA,NA),abs(c(phat,NA,NA)-p))
colnames(compare)=c("fixed pt", "abs err", "Aitkens", "abs error")
options(width=100)
noquote(formatC(compare,digits=15,format="f"))
```

- Here is the result: (Note that Aitkens has 5 correct digits when fixed pt has only 2!)

```
     fixed pt          abs err           Aitkens           abs error
[1,] 1.000000000000000 0.260914866784839 0.728010361467617 0.011074771747544
```

```
 [2,] 0.540302305868140 0.198782827347021 0.733665164585231 0.005419968629929
 [3,] 0.857553215846393 0.118468082631233 0.736906294340474 0.002178838874687
 [4,] 0.654289790497779 0.084795342717382 0.738050421371664 0.001034711843497
 [5,] 0.793480358742566 0.054395225527405 0.738636096881655 0.000449036333505
 [6,] 0.701368773622757 0.037716359592404 0.738876582817136 0.000208550398025
 [7,] 0.763959682900654 0.024874549685494 0.738992243027034 0.000092890188127
 [8,] 0.722102425026708 0.016982708188453 0.739042511328159 0.000042621887002
 [9,] 0.750417761763761 0.011332628548600 0.739065949599941 0.000019183615220
[10,] 0.731404042422510 0.007681090792651 0.739076383318956 0.000008749896205
[11,] 0.744237354900557 0.005152221685396                NA                NA
[12,] 0.735604740436347 0.003480392778813                NA                NA
```

2 Steffensen's Method

- Steffensen's Method is a modification of the fixed point method.
- It combined Fixed Point and Aitken's, but in a different order than above.
- Note that Aitken's requires three points to generate a new value.
- In a nutshell, Steffensen's has two steps:
  1. Start with initial guess p_0, then run fixed point twice: $p_1 = g(p_0)$ and $p_2 = g(p_1)$.
  2. Then take the points (p_0, p_1, p_2) and run Aitken's method once to get a new p_0.
- However, each round has 3 numbers (0,1,2). We need to keep track of the round number, so we will add a "current round" number k (starts at 0).

Steffensen's Method Algorithm

Definition. *Starting with round $k = 0$ and initial guess $p_0^{(0)}$, then do repeatedly the following two steps for $k = 1$ to end:*

1. *Let $p_1^{(k)} = g(p_0^{(k)})$ and Let $p_2^{(k)} = g(p_1^{(k)})$,*
2. *$p_0^{(k+1)} = \Delta^2(p_0^{(k)}, p_1^{(k)}, p_2^{(k)})$. Increment k, goto 1.*

- Think "Fixed Point – Fixed Point – Aitken's" is "Steffensen's".
- Usually $p_0^{(k)}$ is the sequence that we say Steffensen's generates.
- But the more complete Steffensen's also keeps track of $p_1^{(k)}$ and $p_2^{(k)}$.

Convergence rate of Steffensen's

Theorem. *Suppose that $x = g(x)$ has the solution p with $g'(p) \neq 1$. If there exists a $\delta > 0$ such that $g \in C^3[p-\delta, p+\delta]$, then Steffensen's method gives **quadratic** convergence for any $p_0 \in [p-\delta, p+\delta]$.*

- Let's check out Steffensen's on $f(x) = \cos x$

```
> g = function(x) { cos(x) }
> bb=steff(1,g)
```

```
> bb               ##this shows iterations and zero
$iterations
                p_n                g(p)          abs(g(p)-p)
0 1.000000000000000 0.540302305868140 4.59697694131860e-01
1 0.728010361467617 0.746499756045220 1.84893945776032e-02
2 0.739066966908674 0.739097370135781 3.04032271070120e-05
3 0.739085133166075 0.739085133248225 8.21495094172064e-11

$zero
[1] 0.739085133166075
```

- It only took 3 iterations!
- The `steff` procedure has the option to show a complete record of all the iterations, including the fixed point ones.
- Here is a complete record of the iterations including the fixed point.

```
> steff(1,g,1e-15,100,complete=TRUE)
            p_0^(n)            p_1^(n)            p_2^(n)
0 1.000000000000000 0.540302305868140 0.857553215846393
1 0.728010361467617 0.746499756045220 0.734070283736530
2 0.739066966908674 0.739097370135781 0.739076890222895
3 0.739085133166075 0.739085133248225 0.739085133192888
4 0.739085133215161                NA                NA
```