arXiv:2303.03314v1 [math.NA] 6 Mar 2023

多分法的效率*

J. S. C. Prentice†

2023年2月20日

摘要

我们研究了多分法求解单变量非线性方程的效率，并将其与众所周知的二分法进行比较。我们证明，存在一种最小工作量算法，其使用的分段数多于二分法，尽管这种最优算法取决于具体问题。达到最优所需的分段数可通过朗伯 W 函数确定。

1 引言

用计算方法求解非线性方程是应用数学的重要组成部分。二分法 [1] 是一种简单而稳健的求解技术，但由于其线性收敛（就近似误差的上界而言；见附录），与牛顿法等更快的方法相比，它通常不受重视。多分法是二分法的一种扩展（事实上，二分法只是多分法的一个特例），旨在减少迭代次数，尽管这会增加每次迭代的函数求值次数。很自然地会想知道，一般而言，多分法是否在某种意义上比二分法更有效率。本文通过建立一个合适的理论模型，并开展相应的数值实验，探讨这一问题。

*数学主题分类：65D99, 65Y20。

†高级研究官，Mathsophical Ltd.，南非约翰内斯堡

2 相关概念、术语与记号

二分法是一种用于求解以下问题的迭代方法

$$f(x) = 0 \tag{1}$$

其中 $f(x)$ 为实数且连续。如果能够找到一个区间 $[a, b]$，使得 $f(x)$ 在该区间内变号，那么其思路就是用节点 x_1 将 $[a, b]$ 二分（我们将这样的区间称为相关区间）。取该节点作为 1 的根的近似值；如果 $|f(x_1)|$ 足够小，则将 x_1 作为问题的解。如果 $|f(x_1)|$ 不够小，那么根据区间上三个节点处 $f(x)$ 的符号，就可以在下一次迭代中将一个新的、更小的区间（ $[a, x_1]$ 或 $[x_1, b]$）作为相关区间。该过程持续进行，直到 $|f(x_i)|$ 足够小，或相关区间的长度缩小到机器精度。后一条件尤其有用，因为它使我们能够预测最多需要进行多少次迭代。请注意，每次迭代得到的相关区间都比上一次小一半。

多分法（也称为 N 分法）与二分法类似，不同之处在于，每次迭代时，都会在相关区间内引入 $N-1$ 个等距节点（$N \in \mathbb{Z}$、 $N \geqslant 2$），从而将区间划分为 N 个等长子区间（例如，参见 [2]„，其中讨论了 $N = 4$ 和 $N = 6$ 的情况）。如果某个节点处的 $|f(x)|$ 足够小，则停止这一过程，并将该节点作为解。否则，确定一个新的相关区间，并继续这一过程，直到相关区间缩小至机器精度。每个连续的相关区间都比前一次迭代中的区间小 N 倍。与二分法一样，我们可以预测最多需要多少次迭代。请注意，二分法只是 $N = 2$ 的多分法。

在下一节中，为了确定迭代次数的最小值如何随 N 变化，需要求解以下形式的方程：

$$ye^y = x,\ x > 0. \tag{2}$$

该方程的解为

$$y = W(x)$$

其中 $W(x)$ 是朗伯 W 函数的主分支（有时

记作 $W\left(0,x\right)$) [3][4]。此外， $W\left(x\right)$ 满足恒等式

$$e^{W(x)}=\frac{x}{W\left(x\right)} \tag{3}$$

这将在后文派上用场。

3 理论

对于二分法，将相关区间缩小至机器精度所需的最少迭代次数 M 为

$$\frac{b-a}{2^{M}}\leqslant\mu\Rightarrow M\geqslant\frac{\ln\left(\frac{b-a}{\mu}\right)}{\ln 2}$$

其中，我们选择满足这一不等式的最小整数值 M ，而 μ 表示机器精度（通常为 2^{-52}）。多分法的相应表达式为

$$\frac{b-a}{N^{M}}\leqslant\mu\Rightarrow M\geqslant\frac{\ln\left(\frac{b-a}{\mu}\right)}{\ln N}.$$

由于多分法是迭代式的，典型的计算实现会采用 `for` 或 `while` 循环。显然，所需的此类循环次数为 M。如果我们选择用 f（x）的总求值次数来衡量效率，那么每次循环需要对 f（x）求值 $N-1$ 次，因此函数求值总次数 T_{f} 为

$$T_{f}\left(N\right)=\frac{\left(N-1\right)\ln\left(\frac{b-a}{\mu}\right)}{\ln N}.$$

然而，这个量并未体现该算法的细微差别。函数（$N-1$）$/\ln N$ 单调递增，因此 $T_{f}(2)$ 是 $T_{f}.$ 可能达到的最小值。这进而意味着，二分法比任何其他可能的多分法都更有效率。然而，正如我们将在下一节看到的，实验结果并不支持这一结论，因此 T_{f} 并不是衡量多分法效率的良好模型。

我们更倾向于用总实际时间来衡量效率 $T_{t}.$ 我们假设完成一次循环所需的时间具有 $mN+c$ 的形式，因此完成 M 迭代所需的总时间为

$$T_{t}\left(N\right)=\frac{\left(mN+c\right)\ln\left(\frac{b-a}{\mu}\right)}{\ln N}. \tag{4}$$

尽管该表达式与 T_f 的形式相同，但我们将看到，常数 m 和 c 对于确定是否存在最小值至关重要。我们将 m 解释为将 N 增加一所需的成本（时间），并将c 解释为循环中所有不依赖于N的部分所需的成本（时间）。

要在 T_t中找到最小值，我们对其求导并令其等于零:

$$\frac{dT_t}{dN} = \ln\left(\frac{b-a}{\mu}\right)\left(\frac{m}{\ln N} - \frac{(mN+c)}{N(\ln N)^2}\right)$$
$$\frac{dT_t}{dN} = 0 \Rightarrow N\ln N = N + \frac{c}{m}.$$

要解这个方程，还需要进一步变形。采用记号 $L \equiv \ln\ N$，并利用 $N = e^{\ln N}$这一事实，可得

$$\begin{aligned} N\ln N &= N + \frac{c}{m} \\ \Rightarrow Le^L &= e^L + \frac{c}{m} \\ \Rightarrow e^L(L-1) &= \frac{c}{m} \\ \Rightarrow e^{L-1}(L-1) &= \frac{c}{m}e^{-1}. \end{aligned}$$

现在，方程具有式（2）的形式，其中 $y = L-1$ 和 $x = c/me$.因此，我们有

$$L - 1 = W\left(\frac{c}{m}e^{-1}\right)$$

因此

$$\begin{aligned} L = \ln N &= W\left(\frac{c}{m}e^{-1}\right) + 1 \\ \Rightarrow N &= e^{W\left(\frac{c}{m}e^{-1}\right)+1} = ee^{W\left(\frac{c}{m}e^{-1}\right)} \\ &= \frac{e\frac{c}{m}e^{-1}}{W\left(\frac{c}{m}e^{-1}\right)} \\ \Rightarrow N(R) &= \frac{R}{W\left(Re^{-1}\right)} \equiv N_{min} \end{aligned} \tag{5}$$

其中我们使用了（3）式以及 $R = c/m$。这是 $T_t(N)$ 取得最小值时 N 的值，我们特别强调了这一数值取决于比值 R。

现在我们有

$$T_t\left(N_{min}\right)=\frac{\left(mN_{min}+c\right)\ln\left(\frac{b-a}{\mu}\right)}{\ln N_{min}}=c\ln\left(\frac{b-a}{\mu}\right)\left(\frac{\frac{N_{min}}{R}+1}{\ln N_{min}}\right)$$
$$=c\ln\left(\frac{b-a}{\mu}\right)\left(\frac{N_{min}+R}{R\ln N_{min}}\right)$$

这清楚地表明， $T_t\left(N_{min}\right)$不仅取决于 R，还取决于 a、 b、 c 和 μ. 。后面这四个参数体现了算法与具体问题和平台相关的特性。

比值 $T_t\left(N_{min}\right)\ /T_t\left(2\right)$衡量了 N_{min}处相对于二分法的效率，我们将其称为 $RelEff$，其表达式为

$$RelEff\equiv\frac{T_t\left(N_{min}\right)}{T_t\left(2\right)}=\frac{c\ln\left(\frac{b-a}{\mu}\right)\left(\frac{N_{min}+R}{R\ln N_{min}}\right)}{c\ln\left(\frac{b-a}{\mu}\right)\left(\frac{2+R}{R\ln 2}\right)}$$
$$=\left(\frac{R\ln 2}{2+R}\right)\left(\frac{N_{min}+R}{R\ln N_{min}}\right).\qquad(6)$$

我们在图1和图2中展示了 N_{min} 和 $RelEff$ 随 R 变化的函数关系。 N_{min} 的最小值为2.73。它并非恰好为2（对应于二分法），这表明从我们的分析角度来看，二分法并非最优方法。我们看到 N_{min} 单调递增，同时，对于此处考虑的 R 取值， $RelEff$ 达到最小值 ~ 0.1；这表明相对于二分法，效率有可能得到显著提升。

4 数值计算

本节报告与前文所述理论相关的数值实验。对于一组选定的非线性方程，我们在 $N\in[2,250]$. 范围内针对 N确定了相关量，例如R、 N_{min}和 $RelEff$。对于每个 N，我们测量了每次循环时间，并对 1000 次循环取平均。结果汇总于表 1。表 1 中的 r^2 系数衡量了以下假设的拟合优度：每次循环时间可视为线性函数（即 $mN+c$）。

$f(x)$	$[a,b]$	$R\ (=c/m)$	N_{min}	r^2	$RelEff$
$\sin\,x-\cos x$	$[0,\pi/2]$	2.73×10^2	81	0.963	0.203
e^x-2-x	$[1,4]$	2.17×10^2	67	0.980	0.214
x^2-8	$[-5,-2]$	5.51×10^2	140	0.930	0.175
$\frac{1}{10-x}-\frac{1}{4}$	$[-2,7]$	3.60×10^2	100	0.894	0.191
$e^{-x}\sin x$	$[-8,-5]$	1.18×10^2	43	0.996	0.247
$\ln(x^2)-7$	$[20,40]$	1.44×10^2	49	0.991	0.236

表 1：数值实验结果。

对于每个 $f(x)$，我们通过最小二乘拟合确定了 m 和 c。系数 r^2 表明这些拟合效果很好。随后，我们使用式（5）和式（6）计算了 N_{min} 和 $RelEff$。显然，在所有情况下，N_{min} 都显著大于 2，而且 $RelEff$ 的效率比二分法高出约 80%。

我们在图 3 和图 4 中提供了一些细节，其中展示了 $f(x)=\sin x-\cos x$. 的实验数据和理论曲线。其余五种情况类似。我们注意到，R 在每种情况下的数量级都相同；尽管 N_{min} 变化很大，$RelEff$ 却相当一致。

5 结论

我们建立了一个模型，用于描述多分法的效率。我们以求解给定问题所需的实际时间来衡量效率。模型假设每次迭代所需的时间与 N 呈线性关系，数值实验似乎支持这一假设。通过朗伯 W 函数，可以根据这一线性关系中参数的比值确定最佳分段数。显然，在数值研究中考察的所有情况下，最佳多分法都与二分法（即 $N_{min}\gg2$）有显著差异，效率也因此得到显著提升（约 80%）。当然，我们的结果是针对计算平台 [5]，得出的，但我们认为，合理推测在其他系统上也会得到类似结果。

参考文献

[1] R.L. Burden 与 J.D. Faires，《数值分析》第 9 版，Brooks/Cole，波士顿，2011 年。

[2] Mohd Ali 等，《第 n 分段法：二分法的一种改进》，《马来西亚基础与应用科学杂志》，第 13 卷，第 4 期（2017 年），728–731 页。[3] Robert M. Corless、G. H. Gonnet、D. E. G. Hare、D. J. Jeffrey 和 D. E. Knuth，《论朗伯 W 函数》，《计算数学进展》，第 5 卷（1996 年），329–359 页。[4] Thomas P. Dence，《浅谈朗伯 W 函数》，《应用数学》，第 4 卷（2013 年），887–892 页。[5] 英特尔 i7 9700K 处理器、Strix Z390-F 主板、48GB DDR4 内存、Windows 10 64 位、Matlab R2022b。[6] S. S. Cheng 和 T. T. Lu，《二分算法并非线性收敛》，《大学数学杂志》，第 16 卷（1985 年），56–57 页。

6 附录

Cheng 和 Lu [6] 通过一个反例证明，严格来说，不应将二分法视为线性收敛。虽然他们的论证在分析上是正确的，但这似乎削弱了该方法已知的一项性质：经过有限次迭代，它就能得到精度可接受的解。这一性质源于该方法的近似误差上界的变化。事实上，如果 p 是我们要求的根，而 p_i 是第 i次迭代后的近似值，那么必有

$$0 \leqslant |p_i - p| \leqslant \frac{b-a}{2^i}.$$

该表达式右侧是 $|p_i - p|$的一个上界，记作 B_i，显然有

$$B_{i+1} \leqslant \frac{1}{2} B_i$$

对所有 $i \geqslant 0$ （其中 $B_0 = b - a$）均成立。因此，该上界是线性收敛的。

此外，总存在一个 i 的值，使得

$$B_i \leqslant \varepsilon$$

对于任意 $0 < \varepsilon \leqslant b - a$，由于 $|p_i - p| \leqslant B_i$，我们可以使 $|p_i - p|$任意小。此外，在有限精度计算环境中，存在一个 z，使得

$$B_z = \frac{b-a}{2^z} = 0,$$

因此，在这样的环境中，z、$|p_z - p| = 0$ 和 p_z 可以视为精确值。在我们的平台上，我们发现 $z = 1024$.

在 [6], 中给出了一个例子，其中不存在整数 n 使得该比值

$$\frac{|p_{i+1} - p|}{|p_i - p|}$$

对所有 $i \geqslant n$都单调递减，因此不满足线性收敛的条件。尽管如此，误差界确实会（线性地）收敛，并且该问题可以通过有限次迭代求解（在我们的平台上需要 1024 次迭代）。事实上，我们发现 $p = 0.564468413605939$。

从业者往往会非正式地称二分法线性收敛，仅仅是因为误差界具有这一性质。更准确地说，我们应该说，二分法的近似误差上界是线性收敛的。尽管如此，二分法能够在有限次迭代内达到任意精度；在有限精度环境中，也能得到事实上精确的结果。

很自然会推测，也许应当对有限精度环境中的收敛概念进行形式化定义，但这是另一个话题。

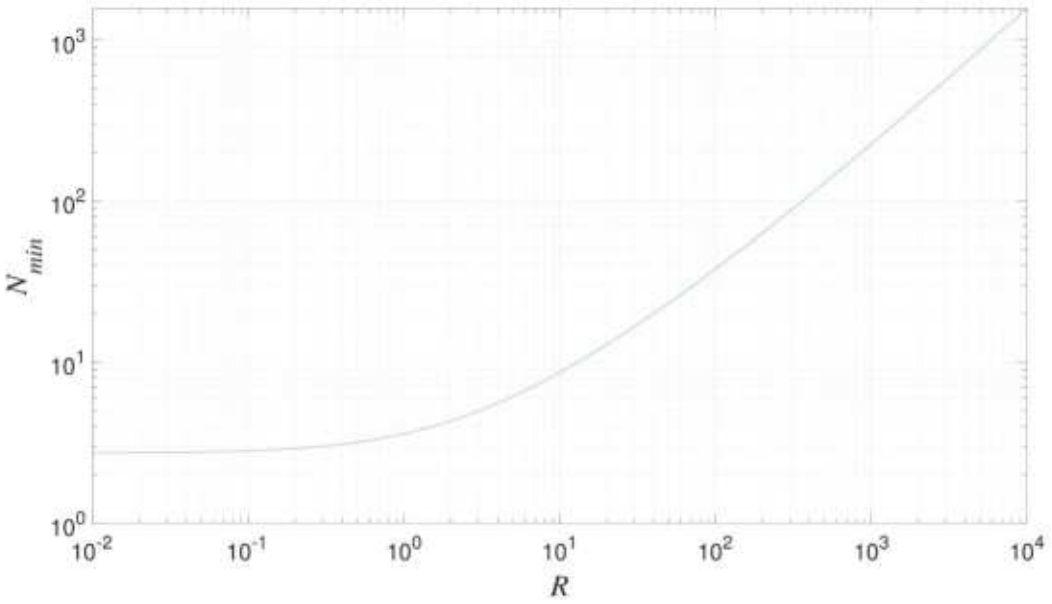


Figure 1: N_{min} vs R

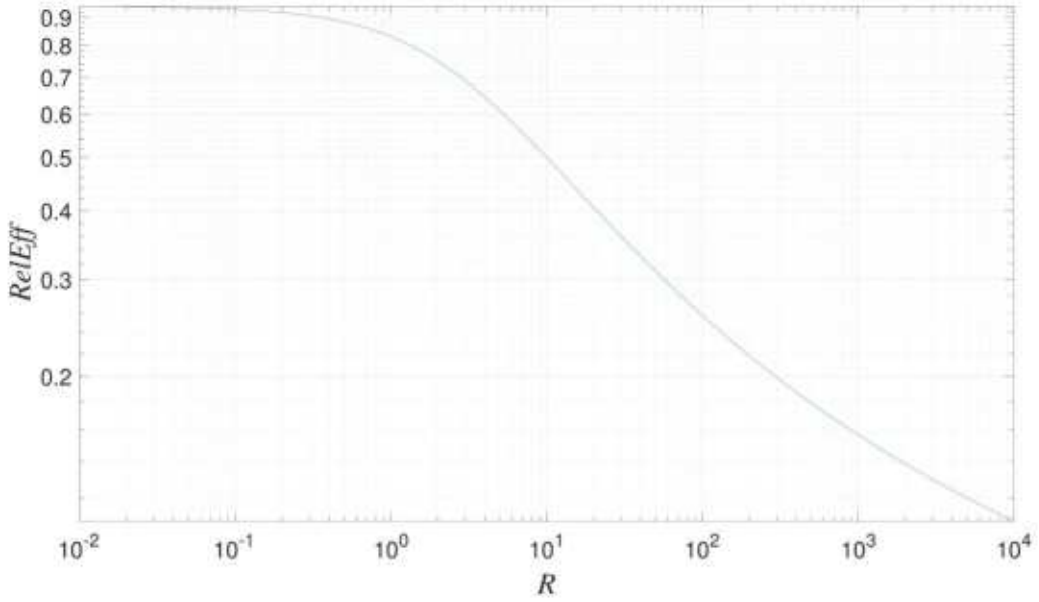


Figure 2: $RelEff$ vs R

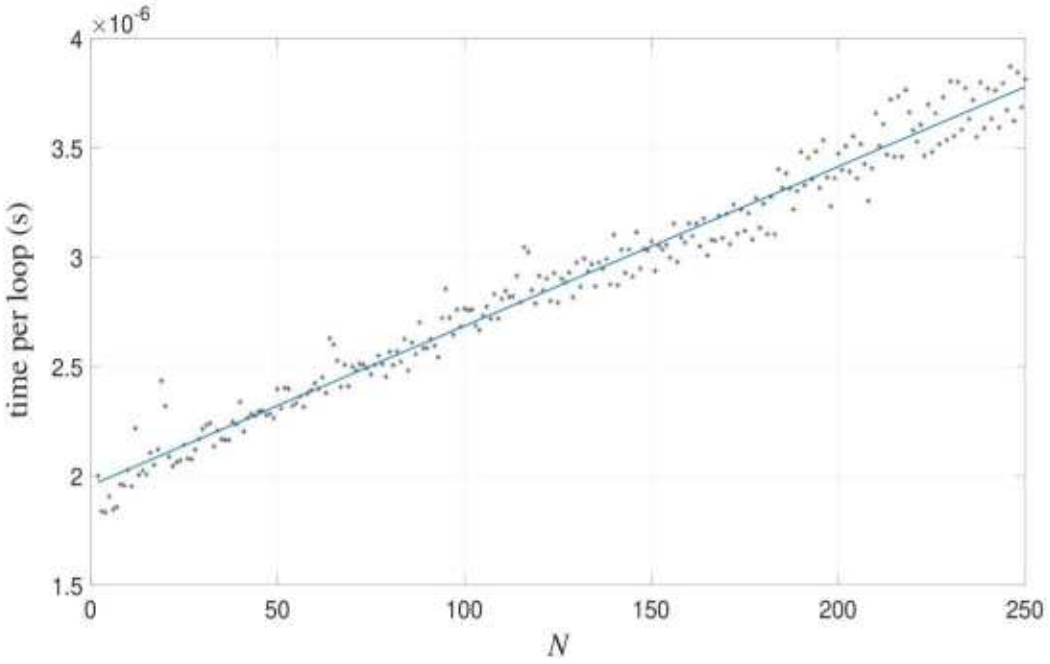


Figure 3: Time per loop (s) vs N for $f(x) = sin\, x - cos\, x$.

Scatter in experimental data is evident; solid line is linear fit.

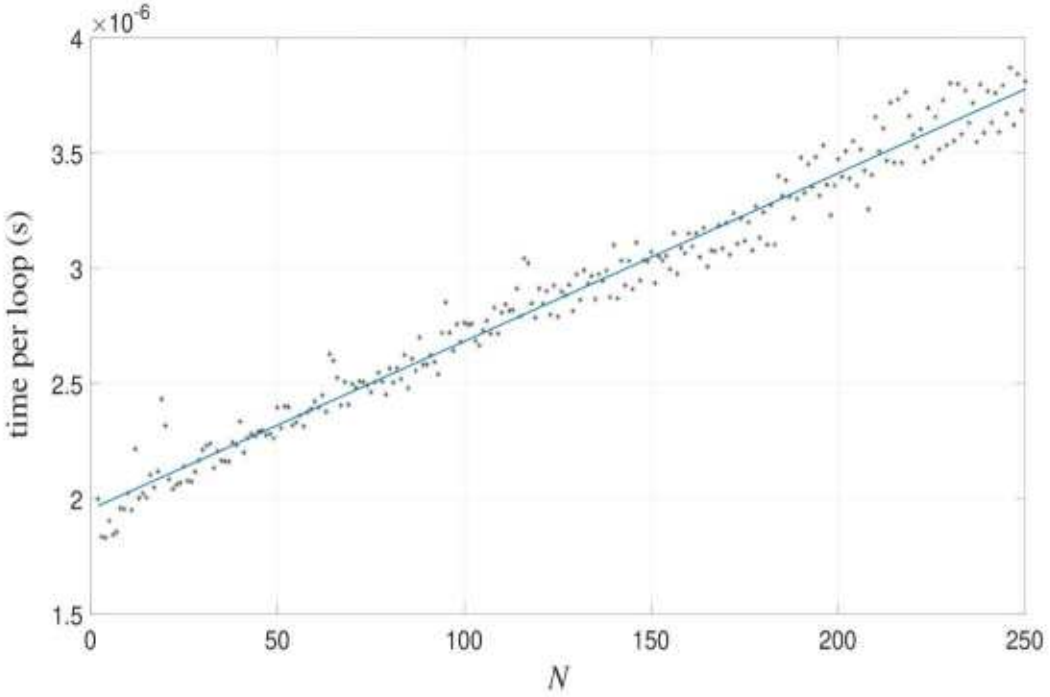


Figure 4: T_t vs N for $f(x) = sin\, x - cos\, x$. Solid line is (4).