E08a 编程作业解答

姓名：范舟　学号：516030910574

注意：

1. 程序在文档中也要粘贴，同时把代码和该文档放在同一个文件夹中打包发给我.（建议多个同学或整个班级一起打包；邮箱：terenceyuyue@sjtu.edu.cn）
2. 该文档不需打印，只收电子版.

1　问题 1 的解答

求下列方程的实根

(1)　$x^2-3x+2-e^x=0$

(2)　$x^3+2x^2+10x-20=0$

1.1　设计不动点迭代

对方程 (1) 和 (2)，分别设计不动点迭代，即给出相应的 $\varphi(x)$，并说明其收敛性.

方程 (1)　首先分析方程 (1) 的根所在的大致区间，设 $f(x)=x^2-3x+2-e^x$，分析可知 $f'(x)=2x-3-e^x<0$ 在 R 上恒成立，因此 $f(x)$ 在 R 上单调递减，$f(x)=0$ 至多有一个实根，又有

$$f(0)=1>0,\quad f(1)=-e<0$$

因此 $f(x)=0$ 存在唯一的实根 x_*，且 $x_*\in(0,1)$.
令迭代公式为

$$x=\varphi(x)=\frac{1}{3}(x^2-e^x+2)$$

则 $\varphi'(x)=\frac{1}{3}(2x-e^x)$，在 $(0,1)$ 上，$\varphi'(x)$ 连续，且有

$$|\varphi'(x)|<\max(|\varphi'(0),\varphi'(1)|)=\frac{1}{3}<1$$

因此此迭代公式局部收敛. 且 $\varphi'(x)\neq 0$ 在 $(0,1)$ 上成立，因此此迭代公式为线性收敛.

方程 (2) 首先分析方程 (2) 的根所在的大致区间，设 $f(x) = x^3 + 2x^2 + 10x - 20$，分析可知 $f'(x) = 3x^2 + 4x + 10 > 0$ 在 R 上恒成立，因此 $f(x)$ 在 R 上单调递增，$f(x) = 0$ 至多有一个实根，又有

$$f(1) = -7 < 0, \quad f(2) = 16 > 0$$

因此 $f(x) = 0$ 存在唯一的实根 x_*，且 $x_* \in (1, 2)$.

令迭代公式为

$$x = \varphi(x) = x - \frac{1}{100}(x^3 + 2x^2 + 10x - 20)$$

则 $\varphi'(x) = 1 - \frac{1}{100}(3x^2 + 4x + 10)$，在 $(1, 2)$ 上，$\varphi'(x)$ 连续，且有

$$|\varphi'(x)| < \max(|\varphi'(1), \varphi'(2)|) = \frac{83}{100} < 1$$

因此此迭代公式局部收敛. 且 $\varphi'(x) \neq 0$ 在 $(1, 2)$ 上成立，因此此迭代公式为线性收敛.

1.2 Steffensen 加速迭代法

1) 对之前设计的不动点迭代格式，编写直接迭代的程序，命名为 direct_iteration.m，迭代停止准则为 $|x_k - x_{k-1}| < 10^{-8}$.

为了方便，以下将各类迭代方式都实现为接口统一的函数，函数接受 3 个参数 x0，phi 和 eps，其中 x0 为迭代初值，phi 为在 Matlab 脚本中实现的迭代函数 $\varphi(x)$ 的函数句柄，eps 为迭代停止准则中设定的误差界；并在 test_room.m 中定义了题目中使用的迭代函数 $\varphi(x)$，并调用实现好的迭代方法函数进行方程根的求解.

下面是在 direct_iteration.m 中实现的直接迭代的函数.

```matlab
function ret = direct_iteration(x0, phi, eps)
    prev_x = x0;
    step_count = 0;
    while true
        step_count = step_count + 1;
        x = phi(prev_x);
        if abs(x - prev_x) < eps
            break;
        end
        prev_x = x;
    end
    fprintf("step number using direct_iteration: %d\n", step_count);
    ret = x;
end
```

下面是在 test_room.m 中定义的两个迭代函数以及调用 direct_iteration 函数求根的代码.

```matlab
x0 = 1;
eps = 1e-8;
ans1 = direct_iteration(x0, @phi1,eps);
ans2 = direct_iteration(x0, @phi2, eps);
fprintf("%.8f %.8f\n", ans1, ans2);
```

```
function y = phi1(x)
    y = (x^2 - exp(x) + 2) / 3;
end

function y = phi2(x)
    y = x - (x^3 + 2 * x^2 + 10 * x - 20) / 100;
end
```

运行此脚本，得到方程 (1) 的根为 0.25753028，方程 (2) 的根为 1.36880807.

2) 编写 Steffensen 加速程序，命名为 Steffensen.m, 在相同条件下重复方程 (1) 的计算：给出不同初值下，加速前和加速后的迭代步数（列出表格）.

下面是在 Steffensen.m 中实现的 Steffensen 加速迭代函数.

```
function ret = Steffensen(x0, phi, eps)
    prev_x = x0;
    step_count = 0;
    while true
        step_count = step_count + 1;
        x = prev_x - (phi(prev_x) - prev_x)^2 / ...
            (phi(phi(prev_x)) - 2 * phi(prev_x) + prev_x);
        if abs(x - prev_x) < eps
            break;
        end
        prev_x = x;
    end
    fprintf("step number using Steffensen: %d\n", step_count);
    ret = x;
end
```

在 test_room.m 中调用 Steffensen 函数求方程 (1) 的根.

```
ans1_steffensen = Steffensen(x0, @phi1,eps);
fprintf("ans1_steffensen = %.8f\n", ans1_steffensen);
```

运行脚本得到方程 (1) 的根为 0.25753029.

设定一组不同的初值，分别调用 direct_iteration 函数和 Steffensen 函数求方程 (1) 的根并记录迭代步数，得到下表的结果. 可以明显看出 Steffensen 加速有效减少了迭代步数.

迭代初值	迭代步数	
	加速前	加速后
0	14	4
0.2	13	3
0.3	13	3
0.5	14	4
1	15	4
2	16	4

1.3 Newton 迭代

1) 针对方程 (2)，讨论其实根的范围, 并判断其是单根还是重根.

设 $f(x) = x^3+2x^2+10x-20$，由 b^2-4ac 判别式小于 0 可知 $f'(x) = 3x^2+4x+10 > 0$ 在 R 上恒成立，因此 $f(x)$ 在 R 上单调递增，$f(x) = 0$ 至多有一个实根，且此实根为单根，又有

$$f(1) = -7 < 0, \quad f(2) = 16 > 0$$

因此方程 (2) 存在唯一的实根 x_*，$x_* \in (1, 2)$，且 x_* 为单根.

2) 编写基于课本 P227 式 (4.14) 的 Newton 迭代程序，命名为 Newton_iteration.m, 迭代停止准则同前.

课本 P227 式 (4.14) 的迭代函数为

$$\varphi(x) = x - \frac{f(x)f'(x)}{[f'(x)]^2 - f(x)f''(x)}$$

下面是在 Newton_iteration.m 中实现的基于上式的 Newton 迭代过程，依照上式求出 $\varphi(x)$ 后，可利用上面实现的 direct_iteration 函数求出根. 这段代码求解了方程 (1) 和方程 (2) 的根.

```
x0 = 1;
eps = 1e-8;
ans1 = direct_iteration(x0, @(x) Newton_phi(@f1, @d_f1, @d2_f1, x), eps);
ans2 = direct_iteration(x0, @(x) Newton_phi(@f2, @d_f2, @d2_f2, x), eps);
fprintf("ans1 = %.8f, ans2 = %.8f\n", ans1, ans2);

function y = Newton_phi(f, d_f, d2_f, x)
    y = x - (f(x) * d_f(x)) / (d_f(x)^2 - f(x) * d2_f(x));
end
```

上述代码使用了方程 (1) 与方程 (2) 的相关函数及导函数，如下

```
function y = f1(x)
    y = x^2 - 3 * x + 2 - exp(x);
end

function y = d_f1(x)
    y = 2 * x - 3 - exp(x);
end

function y = d2_f1(x)
    y = 2 - exp(x);
end

function y = f2(x)
    y = x^3 + 2 * x^2 + 10 * x - 20;
end

function y = d_f2(x)
    y = 3 * x^2 + 4 * x + 10;
```

```
end

function y = d2_f2(x)
    y = 6 * x + 4;
end
```

运行此脚本得到方程 (1) 的根为 0.25753029，方程 (2) 的根为 1.36880811.

3) 在相同条件下，比较 Steffensen 加速法与 Newton 迭代法的迭代步数.

方程 (1) 使用不同的初值，分别使用两种方法求解方程 (1) 的根，得到迭代步数如下表.

迭代初值	迭代步数	
	Steffensen 加速法	Newton 迭代法
0	4	4
0.01	3	4
0.2	3	4
0.3	3	3
0.5	4	4
1	4	5
2	4	8
5	5	8

方程 (2) 使用不同的初值，分别使用两种方法求解方程 (2) 的根，得到迭代步数如下表.

迭代初值	迭代步数	
	Steffensen 加速法	Newton 迭代法
0.01	6	5
0.5	5	5
0.7	5	5
1.5	4	4
2	5	5
5	6	6
7	6	8

4) 选做：若方程有复根，应如何求解？可能的话，求出方程 (2) 的一个复根（如果有的话）.

答 使用 Newton 迭代法等求解非线性方程根的不动点迭代法，只要将迭代初值设定为在复根附近的复数，迭代将收敛到复根，与求解实根的方法几乎完全相同.

使用上面实现的 Newton_iteration.m，将迭代初值设定为 $x_0 = -5 + 5i$，其余代码无需改动，即可直接求得方程 (2) 的一个复根 $-1.6844 + 3.4313i$; 若将迭代初值设定为 $x_0 = -5 - 5i$，则求得方程 (2) 的另一个复根 $-1.6844 - 3.4313i$.

事实上，除一个实根外，方程 (2) 仅有这两个复根.

2 问题 2 的解答

多项式求根是一个病态问题，考虑多项式

$$p(x) = (x-1)(x-2)\dots(x-10) = a_0 + a_1 x + \cdots + a_9 x^9 + x^{10}$$

求解扰动方程 $p(x) + \varepsilon x^9 = 0$.

2.1 病态性分析

按照展开式给定系数，对 $\varepsilon = 10^{-6}, 10^{-8}, 10^{-10}$，用 MATLAB 求根函数计算扰动方程的根，分析 ε 对根的影响（注意 MATLAB 多项式的存储方式）.

在 Matlab 中使用下面的代码调用 expand 函数展开多项式 $p(x)$.

```
syms x
p = (x - 1) * (x - 2) * (x - 3) * (x - 4) * (x - 5) * ...
    (x - 6) * (x - 7) * (x - 8) * (x - 9) * (x - 10);
expand_p = expand(p)
```

得 $p(x)$ 的系数如下

$$\begin{aligned} p(x) &= (x-1)(x-2)\dots(x-10) \\ &= x^{10} - 55x^9 + 1320x^8 - 18150x^7 + \cdots + 12753576x^2 - 10628640x + 3628800 \end{aligned}$$

下面是在 poly_root.m 中计算扰动方程的根的代码.

```
eps = 1e-10;
p = [1 (-55+eps) 1320 -18150 157773 -902055 3416930 -8409500 12753576 -10628640 3628800];
ans = roots(p);
for i = 1:9
    fprintf("%.8f, ", ans(i));
end
fprintf("%.8f\n", ans(10));
```

得到的计算结果如下：

```
# eps = 1e-6:
roots = 9.99722949, 9.00954409, 7.98668735, 7.00940116, 5.99651655, 5.00067909, 3.99993933,
    3.00000195, 1.99999999, 1.00000000

# eps = 1e-8:
```

```
roots = 9.99997244, 9.00009608, 7.99986685, 7.00009342, 5.99996501, 5.00000678, 3.99999939,
    3.00000002, 2.00000000, 1.00000000

# eps = 1e-10
roots = 9.99999972, 9.00000096, 7.99999867, 7.00000093, 5.99999965, 5.00000007, 3.99999999,
    3.00000000, 2.00000000, 1.00000000
```

可以看出，ε 较小时，若扰动量是多项式次数较高项的系数，也能对方程的根产生相当的影响，因此求解多项式的根是病态问题.

2.2 解决方案思考

选做：可能的话提出一个稳定化求解的策略（可以查阅文献）.

答 对于解决这一问题本身的病态性，我并没有很好的想法... 或许可以在求解实际问题时，尽量避免对多项式高次项系数的直接测量，或考虑使用提高测量精度的方法减小系数的扰动.