线性方程组的迭代解法-上机作业

张晨，2017011307

【实验内容】

对比Jacobi, G-S, SOR方法求解线性方程组的迭代次数及误差

【实验结果】

ϵ	Jacobi		G-S		SOR		
	迭代	误差	迭代	误差	ω	迭代	误差
1	8538	0.010182	4657	0.007171	1.95	217	0.000401
0.1	4314	0.012268	2328	0.011090	1.9	170	0.008795
0.01	489	0.066343	297	0.066260	1.5	101	0.066060
0.0001	114	0.004966	108	0.004951	1.01	101	0.004951

ω取在一定精度内使得SOR迭代次数最少的ω，误差采用1-范数误差

【结果分析】

三种方法的求解误差较为接近。

对于不同的线性方程组，使得迭代次数最少的ω的取值一般不同。具体到此问题，随着ϵ的增大，ω的最佳取值不断减小。

【关键代码】

```
solve(1, 0.5, 100, 1.95)
solve(0.1, 0.5, 100, 1.9)
solve(0.01, 0.5, 100, 1.5)
solve(0.0001, 0.5, 100, 1.01)

function [] = solve(epsilon, a, n, w)
    fprintf("Now solving %f, choose w = %f\n", a, w);
    [A,b] = getMatrix(epsilon, a, n);
    y = precise(epsilon, a, n);
    fprintf("%f %f %f %f %f %f %f %f %f %f %f\n", y');
    y1 = jacobi(A, b);
    y2 = G_S(A, b);
    y3 = SOR(A, b, w);
    fprintf("1-norm: %f %f %f\n\n", max(abs(y-y1)), ...
        max(abs(y-y2)), max(abs(y-y3)))
end

function [A, b] = getMatrix(epsilon, a, n)
    A = diag(repmat(epsilon, 1, n-2), -1) + ...
        diag(repmat(-(2*epsilon+1/n), 1, n-1), 0) ...
        + diag(repmat(epsilon+1/n, 1, n-2), 1);
    b = repmat(a/n/n, n-1, 1);
    b(n-1) = b(n-1)-epsilon-1/n;
end

function y = precise(epsilon, a, n)
```

```
    x = (1:n-1)/n;
    y = (1-a)/(1-exp(-1/epsilon))*(1-exp(-x/epsilon))+a*x;
    y = y';
end

function x = jacobi(A, b)
    D = diag(A);
    a1 = [0; diag(A, -1)];
    a2 = [diag(A, 1); 0];
    n= length(b);
    x = zeros(n, 1);
    r = 0;
    while (1)
        r = r+1;
        x_new = (b - a1 .* [0; x(1:n-1)] - a2 .* [x(2:n); 0]) ./ D;
        st1 = num2str(x_new, '%.4g');
        st2 = num2str(x, '%.4g');
        if (all(size(st1)==size(st2)) && all(all(st1 == st2))), ...
            break; end
        x = x_new;
    end
    fprintf("Jacobi: run %d iterations\n", r)
end

function x = G_S(A, b)
    n = length(b);
    x = zeros(n, 1);
    r = 0;
    while (1)
        x_pre = x;
        r = r+1;
        x(1) = (b(1) - A(1,2) * x(2)) / A(1,1);
        for i = 2:n-1
            x(i) = (b(i) - A(i,i-1)*x(i-1) - A(i,i+1)*x(i+1)) / ...
                A(i,i);
        end
        x(n) = (b(n) - A(n,n-1)*x(n-1)) / A(n,n);
        st1 = num2str(x_pre, '%.4g');
        st2 = num2str(x, '%.4g');
        if (all(size(st1)==size(st2)) && all(all(st1 == st2))), ...
            break; end
    end
    fprintf("G_S: run %d iterations\n", r)
end

function x = SOR(A, b, w)
    n = length(b);
    x = zeros(n, 1);
    r = 0;
    while (1)
        x_pre = x;
        r = r+1;
        x(1) = (1-w)*x(1) + w * (b(1) - A(1,2) * x(2)) / A(1,1);
        for i = 2:n-1
            x(i) = (1-w)*x(i) + w * (b(i) - A(i,i-1)*x(i-1) - ...
                A(i,i+1)*x(i+1)) / A(i,i);
        end
        x(n) = (1-w)*x(n) + w * (b(n) - A(n,n-1)*x(n-1)) / A(n,n);
        st1 = num2str(x_pre, '%.4g');
        st2 = num2str(x, '%.4g');
        if (all(size(st1)==size(st2)) && all(all(st1 == st2))), ...
            break; end
    end
    fprintf("SOR: run %d iterations\n", r)
end
```

【程序输出】

```
Now solving 0.500000, choose w = 1.950000
0.012870 0.025663 0.038377 0.051015 0.063577 0.076064
0.088476 0.100814 0.113079 0.125272 0.137394 0.149445
0.161425 0.173336 0.185178 0.196953 0.208659 0.220299
0.231873 0.243382 0.254826 0.266205 0.277522 0.288775
0.299966 0.311096 0.322164 0.333172 0.344121 0.355010
0.365840 0.376613 0.387328 0.397986 0.408588 0.419135
0.429626 0.440062 0.450444 0.460773 0.471049 0.481272
0.491443 0.501563 0.511632 0.521650 0.531619 0.541538
0.551408 0.561230 0.571003 0.580730 0.590409 0.600041
0.609628 0.619169 0.628664 0.638115 0.647522 0.656885
0.666204 0.675481 0.684714 0.693906 0.703056 0.712165
0.721233 0.730260 0.739248 0.748195 0.757104 0.765973
0.774804 0.783597 0.792352 0.801070 0.809750 0.818395
0.827002 0.835574 0.844111 0.852612 0.861078 0.869510
0.877908 0.886272 0.894603 0.902900 0.911164 0.919396
0.927596 0.935764 0.943901 0.952006 0.960081 0.968125
0.976138 0.984122 0.992076
Jacobi: run 8538 iterations
G_S: run 4657 iterations
SOR: run 217 iterations
1-norm: 0.010182 0.007171 0.000401

Now solving 0.500000, choose w = 1.900000
0.052583 0.100639 0.144597 0.184847 0.221744 0.255604
0.286719 0.315348 0.341729 0.366075 0.388580 0.409419
0.428751 0.446719 0.463453 0.479070 0.493677 0.507370
0.520235 0.532352 0.543792 0.554619 0.564891 0.574662
0.583978 0.592884 0.601418 0.609616 0.617510 0.625128
0.632497 0.639641 0.646580 0.653335 0.659923 0.666360
0.672660 0.678837 0.684901 0.690864 0.696736 0.702525
0.708238 0.713884 0.719468 0.724997 0.730475 0.735908
0.741299 0.746654 0.751974 0.757264 0.762527 0.767764
0.772979 0.778174 0.783350 0.788509 0.793653 0.798783
0.803901 0.809008 0.814105 0.819192 0.824271 0.829342
0.834407 0.839466 0.844519 0.849567 0.854610 0.859649
0.864685 0.869717 0.874746 0.879772 0.884796 0.889818
0.894837 0.899855 0.904871 0.909885 0.914898 0.919910
0.924921 0.929931 0.934939 0.939947 0.944955 0.949961
0.954967 0.959972 0.964977 0.969981 0.974985 0.979989
0.984992 0.989995 0.994998
Jacobi: run 4314 iterations
G_S: run 2328 iterations
SOR: run 170 iterations
1-norm: 0.012268 0.011090 0.008795
```

```
Now solving 0.500000, choose w = 1.500000
0.321060 0.442332 0.490106 0.510842 0.521631 0.528761
0.534544 0.539832 0.544938 0.549977 0.554992 0.559997
0.564999 0.570000 0.575000 0.580000 0.585000 0.590000
0.595000 0.600000 0.605000 0.610000 0.615000 0.620000
0.625000 0.630000 0.635000 0.640000 0.645000 0.650000
0.655000 0.660000 0.665000 0.670000 0.675000 0.680000
0.685000 0.690000 0.695000 0.700000 0.705000 0.710000
0.715000 0.720000 0.725000 0.730000 0.735000 0.740000
0.745000 0.750000 0.755000 0.760000 0.765000 0.770000
0.775000 0.780000 0.785000 0.790000 0.795000 0.800000
0.805000 0.810000 0.815000 0.820000 0.825000 0.830000
0.835000 0.840000 0.845000 0.850000 0.855000 0.860000
0.865000 0.870000 0.875000 0.880000 0.885000 0.890000
0.895000 0.900000 0.905000 0.910000 0.915000 0.920000
0.925000 0.930000 0.935000 0.940000 0.945000 0.950000
0.955000 0.960000 0.965000 0.970000 0.975000 0.980000
0.985000 0.990000 0.995000
Jacobi: run 489 iterations
G_S: run 297 iterations
SOR: run 101 iterations
1-norm: 0.066343 0.066260 0.066060

Now solving 0.500000, choose w = 1.010000
0.505000 0.510000 0.515000 0.520000 0.525000 0.530000
0.535000 0.540000 0.545000 0.550000 0.555000 0.560000
0.565000 0.570000 0.575000 0.580000 0.585000 0.590000
0.595000 0.600000 0.605000 0.610000 0.615000 0.620000
0.625000 0.630000 0.635000 0.640000 0.645000 0.650000
0.655000 0.660000 0.665000 0.670000 0.675000 0.680000
0.685000 0.690000 0.695000 0.700000 0.705000 0.710000
0.715000 0.720000 0.725000 0.730000 0.735000 0.740000
0.745000 0.750000 0.755000 0.760000 0.765000 0.770000
0.775000 0.780000 0.785000 0.790000 0.795000 0.800000
0.805000 0.810000 0.815000 0.820000 0.825000 0.830000
0.835000 0.840000 0.845000 0.850000 0.855000 0.860000
0.865000 0.870000 0.875000 0.880000 0.885000 0.890000
0.895000 0.900000 0.905000 0.910000 0.915000 0.920000
0.925000 0.930000 0.935000 0.940000 0.945000 0.950000
0.955000 0.960000 0.965000 0.970000 0.975000 0.980000
0.985000 0.990000 0.995000
Jacobi: run 114 iterations
G_S: run 108 iterations
SOR: run 101 iterations
1-norm: 0.004966 0.004951 0.004951
```