Note: In this problem set, expressions in green cells match corresponding expressions in the text answers.

```
Clear["Global`*"]
```

4 - 10 Gauss-Seidel iteration
Do 5 steps, starting from $x_0 = [1\ 1\ 1]^T$ and using 6S in the computation. Hint. Make sure that you solve each equation for the variable that has the largest coefficient.

5. $10\,x_1 + x_2 + x_3 = 6$; $x_1 + 10\,x_2 + x_3 = 6$; $x_1 + x_2 + 10\,x_3 = 6$

For less deceitful acquisition of matrix iteration, focusing on actual Gauss-Seidel operations, see problem 11.

```
m = {{10, 1, 1}, {1, 10, 1}, {1, 1, 10}};
n = {{6}, {6}, {6}};
```

```
LinearSolve[m, n]
```

$$\left\{\left\{\frac{1}{2}\right\}, \left\{\frac{1}{2}\right\}, \left\{\frac{1}{2}\right\}\right\}$$

7. $5\,x_1 - 2\,x_2 + 0 = 18$; $-2\,x_1 + 10\,x_2 - 2\,x_3 = -60$; $0 - 2\,x_2 + 15\,x_3 = 128$

```
m = {{5, -2, 0}, {-2, 10, -2}, {0, -2, 15}};
n = {{18}, {-60}, {128}};
```

```
LinearSolve[m, n]
```

```
{{2}, {-4}, {8}}
```

9. $5\,x_1 + x_2 + 2\,x_3 = 19$; $x_1 + 4\,x_2 - 2\,x_3 = -2$; $2\,x_1 + 3\,x_2 + 8\,x_3 = 39$

```
Clear["Global`*"]
```

```
m = {{5, 1, 2}, {1, 4, -2}, {2, 3, 8}};
n = {{19}, {-2}, {39}};
```

```
LinearSolve[m, n]
```

```
{{2}, {1}, {4}}
```

11. Apply the Gauss-Seidel iteration (3 steps) to the system in problem 5, starting from (a) 0,0,0 (b) 10,10,10. Compare and comment.

While searching for code dealing with Gauss-Seidel I ran into a recent (Oct '18) question on SEMma that talks about different larger scaled matrix factorization schemes for Mathematica, which I found interesting. It's at

https://mathematica.stackexchange.com/questions/173616/a-geometric-multigrid-solver-for-mathemati-

ca/173617, authored by Henrik Schumacher.

The following Gauss-Seidel code I found at *https://www.youtube.com/watch?v=n1n3pfRbBvE,* the video by miles hill. It's based on the equation

$x_i^{(k+1)} = \mathbb{L}^{-1}\left(\vec{b} - \mathbb{U}.\vec{x}^{(k)}\right)$, the standard Gauss – Seidel iteration equation.

```
Clear["Global`*"]
```

```
GaussSeidelMat[a_?MatrixQ, b_?MatrixQ, x0_?MatrixQ,
  error_Real, steps_Integer] := Block[{l, u, x, abs}, x[0] = x0;
  l = a SparseArray[{i_, j_} /; j ≤ i → 1, {3, 3}];
  u = a SparseArray[{i_, j_} /; j > i → 1, {3, 3}];
  Reap[Do[ x[i] = Inverse[l].(b - u.x[i - 1]);
     abs = Norm[x[i] - x[i - 1]] / Norm[x[i]];
     If[abs < error, Sow@x[i];
      Break[]];
     If[i == steps, Sow@x[steps]], {i, steps}]][[-1, -1, 1]]]
```

Matrix aa is the matrix of coefficients, bb is the rhs sol'n set, pp is an initial guess (set to zero by the problem and also by convention). The first run is with the test data in the video. The video adjusted the tolerance variable greatly downward during its demonstration, but somehow the relatively high 0.5 doesn't affect the matrices in the present problem.

```
aa = {{10, 2, -1}, {-3, -6, 2}, {1, 1, 5}}; (* sample data from video *)
bb = {{27}, {-61.5}, {-21.5}};
pp = {{0}, {0}, {0}};
args = {N@aa, N@bb, N@pp, 0.05, 50};
```

```
GaussSeidelMat @@ args
```

```
{{0.523687}, {8.01}, {-6.00674}}
```

The data from problem 5 is inserted, which is part (a) of the current problem.

```
a2 = {{10, 1, 1}, {1, 10, 1}, {1, 1, 10}};
b2 = {{6}, {6}, {6}};
p2 = {{0}, {0}, {0}};
arg3 = {N@a2, N@b2, N@p2, 0.05, 3};
```

The results, I think, meet the text answer within 5S standards.

```
GaussSeidelMat @@ arg2
```

```
{{0.499825}, {0.500008}, {0.500017}}
```

Now for part (b). Again, the results are within 5S of the text.

```
a2 = {{10, 1, 1}, {1, 10, 1}, {1, 1, 10}};
b2 = {{6}, {6}, {6}};
p2 = {{10}, {10}, {10}};
arg3 = {N@a2, N@b2, N@p2, 0.05, 3};
```

```
GaussSeidelMat @@ arg3
```

```
{{0.503333}, {0.499845}, {0.499682}}
```

14 - 17 Jacobi iteration

Do 5 steps, starting from x_0=[1 1 1]. Compare with the Gauss-Seidel iteration. Which of the two seems to converge faster?

15. The system in problem 9.

I got the Jacobi iteration code from *https://community.wolfram.com/groups/-/m/t/831921*. And I haven't analyzed it to make sure it is genuinely Jacobi algorithmic. But it seems to work. It has the slightly primitive quality of necessitating the setting of the variable k, in the **While** loop, to fit each circumstance.

```
Clear["Global`*"]
```

```
jac[A_, b_, xstart_] := Module[{n, k, i, j, xneu, xalt}, n = Length[b];
   xalt = xstart;
   k = 0;
   While[k < 9, xneu = b;
    For[i = 1, i ≤ n, i++, For[j = 1, j ≤ n, j++,
      If[i ≠ j, xneu[[i]] = xneu[[i]] - A[[i, j]] * xalt[[j]]]];
     xneu[[i]] = xneu[[i]] / A[[i, i]]];
    xalt = xneu;
    k++];
   xalt];
```

```
a2 = {{5, 1, 2}, {1, 4, -2}, {2, 3, 8}};
b2 = {{19}, {-2}, {39}};
p2 = {{1}, {1}, {1}};
```

```
N[jac[a2, b2, p2]]
```

```
{{1.99997}, {1.00003}, {3.99999}}
```

The text answer shows {{1.99934},{1.00043},{3.99684}} after five steps, whereas the above takes eight steps to get to (about) the same place. Anyway, GS seems to converge faster to me.

18 - 20 Norms

Compute the norms (9),(10),(11) for the following (square) matrices. Comment on the reasons for greater or smaller differences among the three numbers.

19. The matrix in problem 5.

```
Clear["Global`*"]
```

```
m = {{10, 1, 1}, {1, 10, 1}, {1, 1, 10}};
n = {{6}, {6}, {6}};
```

```
Norm[m]
```

```
12
```

The text reports three numbers, associated with three phases of intermediate calculation; Mathematica has only one result, which matches the text's final two.