Solutions to Homework 3

1. Some of the values below may differ from those that you calculated. This will depend on the hardware and version of MATLAB used.

<table>
<thead>
<tr>
<th>n</th>
<th>(\frac{|x - x_c|<em>\infty}{|x|</em>\infty})</th>
<th>(\frac{|x - x_s|<em>\infty}{|x|</em>\infty})</th>
<th>(|r|_\infty)</th>
<th>(\text{cond}(A, \infty))</th>
<th>(\epsilon \text{ cond}(A, \infty))</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1.4375e-12</td>
<td>2.9104e-11</td>
<td>2.220e-16</td>
<td>9.4366e+05</td>
<td>2.0953e-10</td>
</tr>
<tr>
<td>15</td>
<td>6.7323</td>
<td>16</td>
<td>8.8818e-16</td>
<td>8.0290e+17</td>
<td>178.28</td>
</tr>
<tr>
<td>20</td>
<td>44.841</td>
<td>21</td>
<td>6.613e-16</td>
<td>5.5963e+18</td>
<td>1242.6</td>
</tr>
</tbody>
</table>

The MATLAB code is:

```matlab
function z = hw3(n)
    A = hilb(n);
    x = ones(n, 1);
    b = A * x;
    format short e;
    xc = hilb(n)x;
    xs = inv(A) * b;
    error1 = norm(x - xc, inf)/norm(x, inf)
    error2 = norm(x - xs, inf)/norm(x, inf)
    residual = norm(b - A * xc, inf)
    condition = cond(A, inf)
    eps * condition
```

(a) No. As seen in the table the residual is of the order of machine \(\epsilon\) (2.2204e-16) for each value of \(n\). At the same time the error increases with \(n\) as does the condition number \(\kappa\). In fact, the larger \(\kappa\) the worse the error. This is in agreement with the expectation that systems with large \(\kappa\) are much more sensitive to round-off error.

(b) No. They are approximately the same.
(c) It does not predict the relative error but it does provide a reasonable upper bound for it. Note that if we follow the rule of thumb (i.e. if $\epsilon = 10^{-\alpha}$ and $\kappa = 10^{\beta}$ then you can expect no more than $\beta - \alpha$ digits of accuracy) then we would need

$$\epsilon \times 5.6 \times 10^{18} \approx 10^{-12}$$

Hence $\epsilon \approx 1/5.6 \times 10^{-30} \approx 10^{-31}$.

2(a, b) MATLAB code for these two parts are as follows:

```
n=300;         % Assign value to n

% Set up column vectors for diagonal entries
e=ones(n,1);  d=-4*e;  c=6*e;

% Sparse storage of matrix
T=spdiags([e,d,c,d,e],[-2,-1,0,1,2],n,n);
T(n,n)=1; T(n,n-1)=-2;
T(n-1,n)=-2; T(n-1,n-1)=5; T(1,1)=9;

% Full storage of matrix
A=full(T);

% Sparse and full storage of right hand side
b=ones(n,1); s=sparse(b);

% Solve using sparse, then full, storage
tic; y=T\s; toc
tic; x=A\b; toc
```

(c) Tic-toc times for each value of $n$:

<table>
<thead>
<tr>
<th>$n$</th>
<th>Full storage</th>
<th>Sparse storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>0.11</td>
<td>0</td>
</tr>
<tr>
<td>500</td>
<td>0.55</td>
<td>0.05</td>
</tr>
<tr>
<td>1000</td>
<td>2.50</td>
<td>0.11</td>
</tr>
</tbody>
</table>
As well as providing storage benefits, this table clearly shows gains in processing time when sparse storage is used in this case. The gains increase as \( n \) increases.

(d) Show that \( RR^T = A \) for some value of \( n \).

(e) MATLAB code for these two parts are as follows:

\[
\begin{align*}
n &= 1000; & \quad & \text{\% Assign value to } n \\
\text{e} &= \text{ones}(n,1); & \quad & \text{\% Set up column vectors for diagonal entries} \\
\text{d} &= -2 \times \text{e}; & \quad & \text{c} &= \text{zeros}(n,1); \\
\text{R} &= \text{spdiags}([\text{c}, \text{c}, \text{e}, \text{d}, \text{e}], [-2, -1, 0, 1, 2], n, n); & \quad & \text{\% Sparse storage of matrix and its transpose} \\
\text{R}(1,1) &= 2; & \quad & \text{T} &= \text{R}'; \\
\text{A} &= \text{full} \,(\text{R}); & \quad & \text{B} &= \text{full} \,(\text{T}); \\
\text{b} &= \text{ones}(n,1); & \quad & \text{s} &= \text{sparse} \,(\text{b}); \\
\text{tic}; & \quad & \text{y} &= \text{R} \backslash \text{s}; & \quad & \text{x} &= \text{T} \backslash \text{y}; & \quad & \text{toc}; \\
\text{tic}; & \quad & \text{y} &= \text{A} \backslash \text{x}; & \quad & \text{B} \backslash \text{y}; & \quad & \text{toc};
\end{align*}
\]

(f) Tic-toc times are:

<table>
<thead>
<tr>
<th>( n )</th>
<th>Full storage</th>
<th>Sparse storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.50</td>
<td>0</td>
</tr>
</tbody>
</table>

The times are smaller because the factorization has already been done. All that is required is the back substitution. The advantage of sparse storage is demonstrated again.