LDPC code construction

Problem 1 (Tanner graph of a linear code):
Let $C$ be a linear code described by the following parity-check matrix

$$H = \begin{pmatrix}
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 0
\end{pmatrix}.$$ 

Draw a Tanner graph corresponding to the parity-check matrix $H$.

Problem 2:
Let $(L(x), R(x))$ be a normalized degree distribution (from node perspective) defined as

$$L(x) = \sum_{i=1}^{l_{max}} L_i x^i \quad \text{and} \quad R(x) = \sum_{i=1}^{r_{max}} R_i x^i,$$

where $L_i$ denotes the relative number of variable nodes of degree $i$, and $R_i$ denotes the relative number of parity-check nodes of degree $i$. The design rate $r = 1 - L'(1)/R'(1)$, where $L'(1)$ is the derivative of $L(x)$ at $1$. Write an algorithm that, for given polynomials $L(x), R(x)$, and code length $n$, creates random parity-check matrix $H$ of approximate size $(1 - r)n \times n$ with the relative number of columns (rows) having $i$ ones being close to $L_i$ ($R_i$). I expect the Matlab function in the following form:

```matlab
function [H rate] = ldpc_create_matrix(L, R, n)
% L and R describe the normalized degree distribution
% (3,6) regular LDPC code will be described by L=[0 0 1] and R=[0 0 0 0 0 1]
% return sparse parity-check matrix H, rate of the code
end
```

and all graphs and output you obtain from the following test script (see test_ldpc_create.m on the web):

```matlab
clc; clear;
n = 1000;

% (3,6) regular LDPC code
L = [0 0 1];
R = [0 0 0 0 0 1];
[H rate] = ldpc_create_matrix(L, R, n);
figure; spy(H);
figure; subplot(2,1,1); hist(full(sum(H,1))); subplot(2,1,2); hist(full(sum(H,2))); disp(rate);

% irregular LDPC code 1
L = [0 0.4994 0.3658 0 0 0 0.0581 0 0 0 0 0.0767];
R = [0 0 0 0 0 1];
[H rate] = ldpc_create_matrix(L, R, n);
figure; spy(H);
figure; subplot(2,1,1); hist(full(sum(H,1))); subplot(2,1,2); hist(full(sum(H,2))); disp(rate);

% irregular LDPC code 2
L = [0 0 0.76 0 0 0 0 0.24];
R = [0 0 0 0 0 1];
[H rate] = ldpc_create_matrix(L, R, n);
```
The algorithm for creating random matrix (or a Tanner graph) with specified number of ones in rows and columns can be described as follows. Given the normalized degree distribution pair \((L(x), R(x))\), create the (non-normalized) degree distribution polynomials \((\Lambda(x), P(x))\), as \(\Lambda(x) = nL(x)\) and \(P(x) = (1 - r)nR(x)\). Round the coefficients of \(\Lambda(x)\) and \(P(x)\) to integers. Make sure that the number of edges going from the variable nodes, \(\Lambda'(1)\), is the same as the number of edges going from parity-check nodes, \(P'(1)\). If not, then adjust the degree of sufficient number of parity-check nodes by one starting from the parity-check nodes with the highest degree and update \(P(x)\). Make sure that \(P'(1) = \Lambda'(1)\). Now imagine that every variable node of degree \(i\) contain \(i\) sockets from which \(i\) edges will be connected to some parity-check nodes. The same holds for the parity-check nodes thus there are \(P'(1)\) sockets at each side. Label each sockets by numbers from 1 to \(P'(1)\) and let \(\sigma\) be a random permutation over the set \(\{1, ..., P'(1)\}\). The Tanner graph is constructed by connecting \(i\)-th socket on the variable side with \(\sigma(i)\)-th socket on the parity-check side. This approach may create a small number of double edges which have to be resolved as shown in the following example. The result is a random Tanner graph (or parity-check matrix).

Hints, warnings and suggestions:
- Try to work with a \((3,6)\) regular code of length 10 first.
- Try to work with sparse matrices. See ‘doc sparse’ to know how to create sparse matrix. Try and study the matrix produced by the following code:

```
H = full(sparse([1 1 2 1],[1 3 5 1],ones(1,4)))
```

**Optional problem (degree distribution from edge perspective):** (will be graded softly)
Let \((L(x), R(x))\) be a normalized degree distribution (from node perspective) as defined above. Suppose that the Tanner graph is obtained randomly with the variable and parity-check degrees following \(L(x), R(x)\). Define \(\lambda_i\) as the fraction of edges connected to variable nodes of degree \(i\). Similarly, \(\rho_i\) as the fraction of edges connected to parity-check nodes of degree \(i\). Define degree distribution from edge perspective as the following polynomials (the term \(x^{i-1}\) is intentional)

\[
\lambda(x) = \sum_{i=1}^{l_{\text{max}}} \lambda_i x^{i-1} \quad \text{and} \quad \rho(x) = \sum_{i=1}^{r_{\text{max}}} \rho_i x^{i-1}.
\]

Hint: What is \(\int_0^x \lambda(z)dz\)?
- Give a simple formula for calculating \(\lambda(x)\) and \(\rho(x)\) from \(L(x)\) and \(R(x)\).
- Give a simple formula for calculating \(L(x)\) and \(R(x)\) from \(\lambda(x)\) and \(\rho(x)\).
- Express the design rate in the terms of \(\lambda(x)\) and \(\rho(x)\).
- Express the average variable node degree and average parity-check node degree in terms of \(\lambda(x)\) and \(\rho(x)\).