1  [Matlab]

(a) Use Matlab to generate the array $X$ of size $5 \times 3$ with entries $x_{ij} = i + j - 1$. Then generate a tridiagonal matrix $T$ whose sub-diagonal is $X(1 : n - 1, 1)$, diagonal is $X(1 : n, 2)$, and whose super-diagonal is $X(2 : n, 3)$.

(b) Convert $T$ to sparse format by using

   (i) $S = \text{sparse}(T)$;

   (ii) The command $\text{spdiags}$;

   (iii) The command $\text{spconvert}$ or $\text{sparse}$ but using

arrays obtained from $X$..
This exercise is about computational graphs and ‘back-propagation’. Consider the simple expression:

\[ c(x, y, z) = z \cdot (x + y) + 2 \cdot y + z \]

[a] Show a computational graph that computes \( c(x, y, z) \) where each node performs an atomic operation comprising an add a multiply (at most) \([a = x + y, b = 2 \cdot y + z\) and finally (at top) \(c = z \cdot a + b\), and \(x, y, z\) are ‘leafs’]. Is the resulting graph a tree? Represent the dependencies by directed edges. Build the graph from left (‘leaves’) to right.

[b] Forward loop: Show how calculation proceeds for case \(x = y = 1, z = 2\).

[c] Show how to calculate \(\partial c / \partial y\) in the same forward manner.

[d] Now start from \(c\) at the top, and see how you can calculate \(\nabla c\) with a chain-rule in situation where \(a, b, c\) have already been computed.