1. (20 points) Recall that in the previous assignment we considered a problem in which we were given two positive numbers $n$ and $d$ and a function $f : \{0,1,\ldots,n\} \to \mathbb{R}$. Our goal was to find the best approximation of $f$ with a polynomial of degree at most $d$. More precisely, we want to find a polynomial of degree $d$ that minimizes $\max_{x \in \{0,\ldots,n\}} |f(x) - p(x)|$. Show that this minimum is equal to

$$\max_{g} \langle f, g \rangle := \max \sum_{i=0}^{n} f(i)g(i)$$

where the maximum is over all functions $g : \{0,1,\ldots,n\} \to \mathbb{R}$ that satisfy $\sum_{j=0}^{n} |g(j)| \leq 1$, and are orthogonal to all the polynomials of degree at most $d$ (i.e. $\langle g, p \rangle := \sum_{i=0}^{n} g(i)p(i) = 0$ for all polynomials $p$ of degree at most $d$).

**Solution:** We start by showing that all functions $g$ satisfying the two given constraints also satisfy the property

$$\sum_{i=0}^{n} f(i)g(i) \leq \max_{x \in \{0,\ldots,n\}} |f(x) - p(x)|$$

for any polynomial $p$ of degree at most $d$. For any function $g$ and any polynomial $p$ satisfying the given constraints:

$$\sum_{i=0}^{n} f(i)g(i) = \sum_{i=0}^{n} f(i)g(i) - \sum_{i=0}^{n} g(i)p(i) \leq \sum_{i=0}^{n} |f(i) - p(i)| \times |g(i)| \leq \max_{x \in \{0,\ldots,n\}} |f(x) - p(x)| \times \sum_{i=0}^{n} |g(i)| \leq \max_{x \in \{0,\ldots,n\}} |f(i) - p(i)|$$

The second line holds because the new sum is 0, by assumption. The third line results from grouping both sums, factoring $g(i)$ and taking the absolute value. The fourth line simply maximizes the value of $f(i) - p(i)$, and extracts it from the sum when it no longer depends on $i$, and the final line is a consequence of the constraint $\sum_{i=0}^{n} |g(i)| \leq 1$. 


Hence, we have proven that $\max_g \langle f, g \rangle \leq \max_{x \in \{0, \ldots, n\}} |f(x) - p(x)|$. What is left to show is that this maximum exists, and that we have the equality. We do so by finding a function $g$ for which the equality holds. We define this function as $g(x) = b_x - c_x$, using the optimal solution of the dual form of the linear program in Question 5 of the previous assignment (see on-line solution).

The norm constraint ($\sum_{i=0}^{n} |g(i)| \leq 1$) follows from the constraint $\sum_{i=0}^{n} b_i + c_i = 1$ from the dual LP, by taking the absolute value over all terms (they are all nonnegative, so the values are not affected), then flipping the sign (again, because of the nonnegativity, this can only decrease the overall sum). The orthogonality constraint is obtained from the other constraint of the dual LP, $\forall k \in \{0, \ldots, d\}, \sum_{i=0}^{n} (b_i - c_i) d^k = 0$, which implies that, for any set of coefficients $a_0, \ldots, a_d$, $\sum_{k=0}^{d} \sum_{i=0}^{n} g(i)a_kx^k = 0$. Reversing the sums and factoring $g(i)$ gives the orthogonality.

Finally, recall that the function to maximize in the dual form was $\sum_{i=0}^{n} f(i)(b_i - c_i) = \sum_{i=0}^{n} f(i)g(i)$, and the function to minimize in the primal form was the maximum error between the polynomial and the function $f$. Hence, we have $\max_g \langle f, g \rangle = \max_{x \in \{0, \ldots, n\}} |f(x) - p(x)|$ due to the strong duality theorem (the primal form of the LP is trivially feasible and bounded).

2. (20 points) Either prove that the following problem is NP-complete or show that it belongs to P by giving a polynomial time algorithm:

- Input: A graph $G$ on $n$ vertices.
- Question: Does $G$ have a proper 3-colouring, where each colour is used exactly $n/3$ times?

Solution: Let us call this problem a balanced 3-colouring. This problem is NP-complete. First, we show that it is in $\mathcal{NP}$: we can certify that a graph has a balanced 3-colouring if we are given (as certificate) the colours to attribute to each vertex, and verify that each colour appears exactly $n/3$ times (which takes $O(n)$ steps), and that no edge joins two vertices of the same colour (which takes $O(n^2)$ steps).

Next, we show that the balanced 3-colouring problem is NP-complete by reducing the usual 3-colouring problem to it. Given a graph $G$ with $n$ vertices as an instance of the usual 3-colouring problem, we create a new extended graph $G'$ by adding to each original vertex 2 new vertices, linked only to their corresponding original vertex (this creates a somewhat hairy graph as shown below). This takes $O(n)$ steps, and results in a graph with $3n$ vertices.

---

1 One could also just add $2n$ isolated nodes to the graph, or just take three disconnected copy of the original graph.
Next, we solve the balanced 3-colouring problem on $G'$ (reading the solution is done in $O(n)$ steps). Finally, we use the colours of the original $n$ vertices of $G$ as the solution to the usual 3-colouring problem over $G$. We are left to show that this is a valid algorithm: $G$ is 3-colourable if and only if $G'$ is 3-colourable using each colour exactly $n$ times (recall that $G'$ has $3n$ nodes). If $G'$ has a balanced 3-colouring, any of its subgraph trivially has a proper 3-colouring. Therefore, $G$ has a proper 3-colouring. The reverse implication also holds. If $G$ has a proper 3-colouring, $G'$ will have a balanced 3-colouring: for each triplet of vertices (the original one from $G$ and the two new ones joined to it), we assign to the two new vertices the two colours not used for the original vertex. Because the new vertices are linked only to one original vertex, this does not add new constraints over the vertices of $G$. This concludes the proof.

3. (20 points) Prove that the following problem is NP-complete:

- Input: A number $k$, and a formula $\phi$ in conjunctive normal form.
- Output: Is there a truth assignment that satisfies $\phi$ and assigns False to exactly $k$ variables?

What happens if in the above problem we replace $k$ with the fixed number 100?

**Solution:** First, the given problem, call it SAT-$k$, is in NP because we can verify in polynomial time whether the given assignment satisfies $\phi$, and if it assigns False to the $k$ variables, by checking all $n$ variables. To prove that SAT-$k$ is NP-complete we can show that the SAT problem can be reduced to SAT-$k$ in polynomial time. Given an oracle for SAT-$k$, which can output YES or NO, for a given number $k$ and CNF $\phi$, we can call it, for $k = 0, 1, ... n$, with the same CNF $\phi$. If any of the outputs from these call results in a YES, then it would be a YES instance of SAT problem, otherwise it would be a NO instance of SAT problem. Since we made polynomial number of call to the oracle, SAT $\leq_P$ SAT-$k$. Now, if we replace $k$ with a fixed number 100, the problem belongs to $P$ because there are at most $n^{100}$ different truth assignments where there are 100 False variables. Since $n^{100}$ is polynomial in $n$, the problem can be solved in polynomial time by checking all the truth assignments for satisfiability.

4. (20 points) Either prove that the following problem is NP-complete or show that it belongs to $P$ by giving a polynomial time algorithm:

- Input: A CNF $\phi$ where every variable appears in at most 3 clauses.
- Question: Is $\phi$ satisfiable?

**Solution:** We are given a CNF formula $\phi = C_1 \land C_2 \land \ldots \land C_m$ with restriction that each variable appears in at most 3 clauses. We show that we can construct $\phi'$ from the unrestricted 3-SAT CNF formula $\phi' = C'_1 \land C'_2 \land \ldots \land C'_k$ in polynomial time and that if $\phi$ is satisfiable then $\phi'$ is satisfiable and vice versa. For each variable $x_i$ that appears in the clause $C'_j$, let us have a corresponding clause $C_j$ with $x_{i,j}$. (e.g. if $C'_j = \neg x_1 \lor x_2 \lor x_3$ then $C_j = \neg x_{1,j} \lor x_{2,j} \lor x_{3,j}$) Let us denote the set of these clauses $C_j$ as set $A$. Now we add a set of clauses, call it set $B$, to $\phi$:

- Let $j_1 \ldots j_d$ be the indices of the clauses $C'_j$ in which $x_i$ appears.
- We add the $d$ clauses to $\phi$: $\neg x_{i,j_1} \lor x_{i,j_2} \lor \neg x_{i,j_2} \lor x_{i,j_2} \lor \ldots \lor \neg x_{i,j_d} \lor x_{i,j_1}$

Thus, each variable $x_{i,j}$ appears at most 3 clauses: once in a clause $C_j$ from set $A$ and at most twice in the clauses from set $B$.

Let an assignment $x_i = a_i$ satisfy $\phi'$. Then, in case of $\phi$, we can assign $x_{i,j} = a_i$ so that all the clauses in set $A$ preserve their truth values (i.e. are satisfiable) and conjunction of all the clauses in the set $B$, is true (by the definition of clauses in $B$). Therefore $\phi$ is also satisfiable under the same assignment.
Now, let an assignment $x_{i,j} = b_{i,j}$ satisfy $\phi$. Since this assignment satisfies all clauses in the set $B$, it must be that for each $i$, all $b_{i,j}$ are equal. In other words, $b_{i,j} = b_i$. If $C_j$ in set $A$ is satisfied by assignment $x_{i,j} = b_i$, then $C'_j$ is also satisfied by $x_i = b_{i,j} = b_i$. Therefore, $\phi'$ is also satisfiable.

Since any 3-SAT formula can be reduced to an equivalent restricted formula $\phi$ in polynomial time, the given problem is also NP-complete.

5. (20 points) Show that if in the decision version of linear programming we allow constraints of the form $|\sum_{i=1}^n a_i x_i| \geq b$ for integers $b$ and $a_i$, then the problem becomes NP-complete.

Solution: Note that $|x_i| \geq 1$, $x_i \geq -1$ and $x_i \leq 1$ together imply that $x_i \in \{-1, 1\}$. Then $\frac{x_i + 1}{2} \in \{0, 1\}$ and we can use these to solve NP-complete problems. For example we can reduce vertex cover to this problem. For an input $\langle G = (V,E), k \rangle$ to vertex cover, we can write

$$\min \sum_{u \in V} y_u$$

s.t. $y_u + y_v \geq 1$ \quad $\forall uv \in E$

$y_u = \frac{x_u + 1}{2}$ \quad $\forall u \in V$

$|x_u| \geq 1$ \quad $\forall u \in V$

$x_u \geq -1$ \quad $\forall u \in V$

$x_u \leq 1$ \quad $\forall u \in V$

Hence $G$ has a vertex cover of size at most $k$ if and only if the solution to this program is at most $k$. 

4