Question 5:
Delta hedging Using NS Model

Any product that whose payoffs depends upon the interest rates at a particular time (tenor) will have a **risk** to the yield curve movements at that particular tenor.

E.g. A bond with time to maturity of 3.5 years will have yield curve risk to zero rates at 0.5,1.5,2.5 and 3.5 years. This product can be any interest rate or debt instrument.

Let's assume a bank has offered some exotic fixed income product to a client. A rates trader does not want usually want to have directional risk to the first order yield curve movements and eliminates it using a set of hedging instruments. The hedging used to eliminate first order risk to underlying (in this case interest rates) is known as delta-hedging and the risk is known as deltas. These hedging instruments are usually vanilla products as opposed to the products actually being traded which can be quite complex.
For example, take a bond **B** we want to calculate its delta risk at pre-specified benchmark yield curve tenors $\{t_1, t_2, ..., t_j, ...\}$.(Please note the bond can have risk on tenors outside of this set, but we are only interested in risk on these benchmark tenors). Interest rate deltas for a particular tenor t_j can be calculated by increasing the interest rate at $t = t_j$ i.e., $r(t_j)$ by **0.0001** (1e-4 or 1 basis point or **1bp**) and **keeping the others constant** and then calculating the change in price of the bond i.e. -

$$\Delta B_j = B[r(t_j) + 1bp\,;\, r(t_1)\, r(t_2) \ldots] - B[r(t_j); r(t_1), r(t_2) \ldots]$$

As you can see, we can form an array of deltas by increasing one of the interest rates and keeping others constant. And if we have multiple such bonds we can create a matrix of deltas where each element represents a change in one interest rate tenor for one of the bonds. i.e.

$$\Delta B_{ij} = B^i[r(t_j) + 1bp\,;\; r(t_1)\, r(t_2) \ldots] - B^i[r(t_j)\,;\, r(t_1)\, r(t_2) \ldots]$$

Optional Read: {Practically speaking if we tweak the zero rate at t_i by itself it can create arbitrage because it could be possible that discount factor at time $t_i+\epsilon$ is more than discount factor at t_i (Can you think why that could potentially create an arbitrage?). That is why a forward rate approach is sometimes used where instantaneous forward rate curve is tweaked instead of zero rates themselves. But to maintain simplicity for our quant challenge audience we assume that zero rates are tweaked. }
Now, if we want to hedge any instrument **V** with risk to interest rates at times (tenors) $\{r(t_1) \ldots r(t_n)\}$ we can use the interest rates deltas for the bonds at these particular (benchmark) tenors to match the risk and hedge. We will take the m number of bonds to hedge.
Then we can solve the following set of following linear equations to calculate the weight we should allocate for each of the bond:

$$\Delta V = W^T \Delta B$$

Where, **ΔV** is deltas of size (1 X n) for the instrument V. W is the weights of size (m X 1) and **ΔB** is the delta matrix for the bonds of size (m X n). i.e.

$$\Delta V_j = V[r(j) + 1bp\,; r(t_1) \ldots r(t_n)] - V[r(j)\,; r(t_1) \ldots r(t_n)]$$

$$\Delta B_{ij} = B^i[r(t_j) + 1bp\,;\; r(t_1) \ldots r(t_n)] - B^i[r(t_j)\,;\; r(t_1) \ldots r(t_n)]$$

For this question we will take that n = m. i.e. #Bonds = #Interest rates to be perturbated. So that ΔB is invertible.

This method of creating a hedge portfolio using perturbations to the underlying curve is known as the Jacobian method because **ΔB** is a Jacobian matrix of the bond hedge set w.r.t interest rate curve perturbations.

Optional Reference: In case of n != m, hedging form can be extended by giving relative importance to certain hedging instruments using an 'importance matrix'. Also, to stabilize solutions a technique known as Tikhonov regularization can be used.
https://en.wikipedia.org/wiki/Tikhonov_regularization

In this question, you are given a set of deltas for the instrument to be hedged i.e. you are given the array **ΔV**. You are given the yield curve tenors $\{t_1,t_2\ldots,t_j,\ldots t_n\}$ for which we want to hedge **V** for. You are also given a set of n bonds to be used to hedge with their respective coupons and time to maturities. You are also given the nelson siegel parameters in the order $[\beta_0, \beta_1, \beta_2, \tau]$. Same as last question, assume annual payment frequency, continuously compounded discount factors and FV = 100, tweak size for both deltas of **V** and **B** are 1bp.
You need to find the weights to be used for each bond used as a hedge i.e., the array **W**.

STEPS:

1. Create the cashflow matrix of given bonds, determine all the payment times
2. Calculate the zero rates at all the payment times using the Nelson Seigel equation from last question and the input parameters.
3. Calculate the base bond prices using these rates, coupons and payment times.
4. Increase the rates at the benchmark tenors $\{t_1,t_2\ldots,t_j,\ldots t_n\}$ **one-by-one** by 1bp (0.0001) and recalculate the bond prices using the increased rates and the unchanged coupons and payment times. Because the rates are increased one-by-one, this will create a matrix of new bond prices.
5. Calculate the deltas by subtracting the new bond prices with the base ones.
6. Solve the system of linear equations to get **W**.

Input Specification:

input1: A double type array of tenors $\{t_1,t_2\ldots,t_j,\ldots t_n\}$ to be hedged in increasing order
input2: A double type array of deltas for the instrument to be hedged in order of the benchmark tenors.
input3: A double type array of Nelson Siegel parameters in the order $[\beta_0, \beta_1, \beta_2, \tau]$
input4: A double type array of time to maturity for the bonds in increasing order. Can have duplicate values, can take floating decimal values.
input5: A double type array of coupons for the bonds. Bonds are in increasing order of their time to maturity.
input6: An integer n representing the size of tenors

Output Specification:

Return an array/list of hedging weights for the bonds in order and round the values to 3 decimal places. Needs to be a 1-D array in C++ and 1-D list in Python.