***Practicals – 4***

*-BS19B032*

*-R. Vasantha Kumar*

*1) The human hemoglobin beta chain is:*

*>sp|P68871|HBB\_HUMAN Hemoglobin subunit beta OS=Homo sapiens OX=9606 GN=HBB PE=1 SV=2*

*MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPK*

*VKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFG*

*KEFTPPVQAAYQKVVAGVANALAHKYH*

*The chicken hemoglobin beta chain is:*

*>sp|P02112|HBB\_CHICK Hemoglobin subunit beta OS=Gallus gallus OX=9031 GN=HBB PE=1 SV=2*

*MVHWTAEEKQLITGLWGKVNVAECGAEALARLLIVYPWTQRFFASFGNLSSPTAILGNPM*

*VRAHGKKVLTSFGDAVKNLDNIKNTFSQLSELHCDKLHVDPENFRLLGDILIIVLAAHFS*

*KDFTPECQAAWQKLVRVVAHALARKYH*

*a) The code for constructing dot plot for alignment of human and chicken hemoglobin beta chain is:*

*#importing required libraries*

*import numpy as np*

*import matplotlib as plt*

*import matplotlib.pyplot as pyplt*

*#given sequences*

*human\_seq = "MVHLTPEEKSAVTALWGKVN"*

*chick\_seq = "MVHWTAEEKQLITGLWGKVN"*

*#function to check matches between given sequences*

*def seq\_check(seq1,seq2):*

*mat=np.zeros((len(seq1),len(seq2)))*

*for i in range(len(seq1)):*

*for j in range(len(seq2)):*

*if(seq1[i]==seq2[j]):*

*if(i==j):*

*mat[i][j]=2*

*else:*

*mat[i][j]=1*

*return mat*

*#function to print matched parts of seuence*

*def align\_seq(seq1,seq2):*

*seq = ""*

*for i in range(len(seq1)):*

*if(seq1[i]==seq2[i]):*

*seq = seq + seq1[i]*

*else:*

*seq = seq + "-"*

*return seq*

*aligned\_seq = align\_seq(human\_seq,chick\_seq)*

*print(aligned\_seq)*

*#plotting the graph*

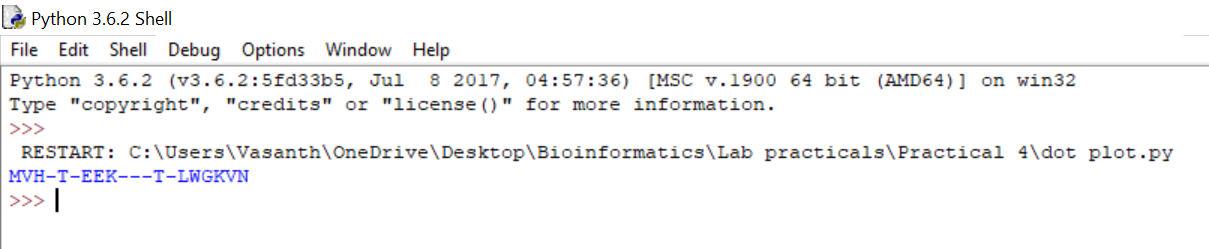
*mat = seq\_check(human\_seq,chick\_seq)*

*pyplt.imshow(mat)*

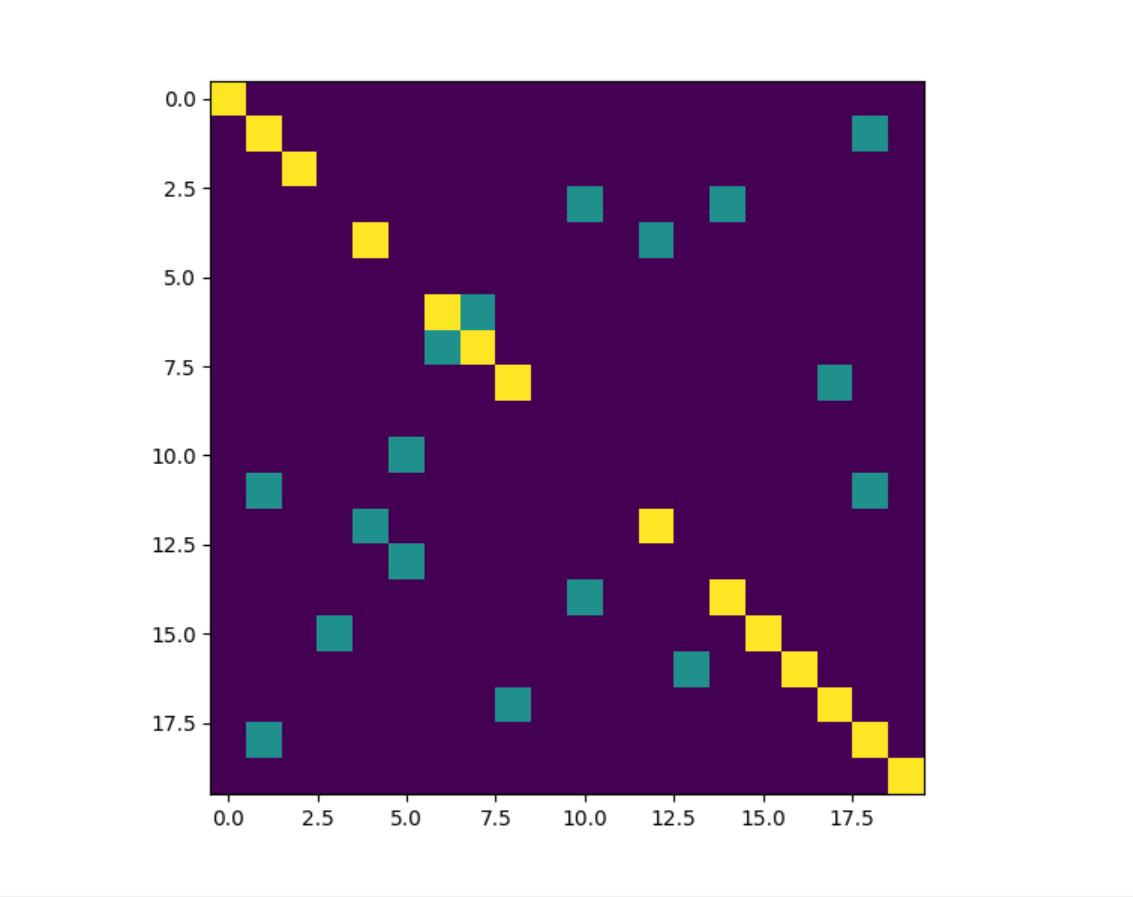
*pyplt.show()*

*The aligned parts of the given sequences for length of 20 residues is:*

***MVH-T-EEK---T-LWGKVN***

******

*The dot plot obtained is:*

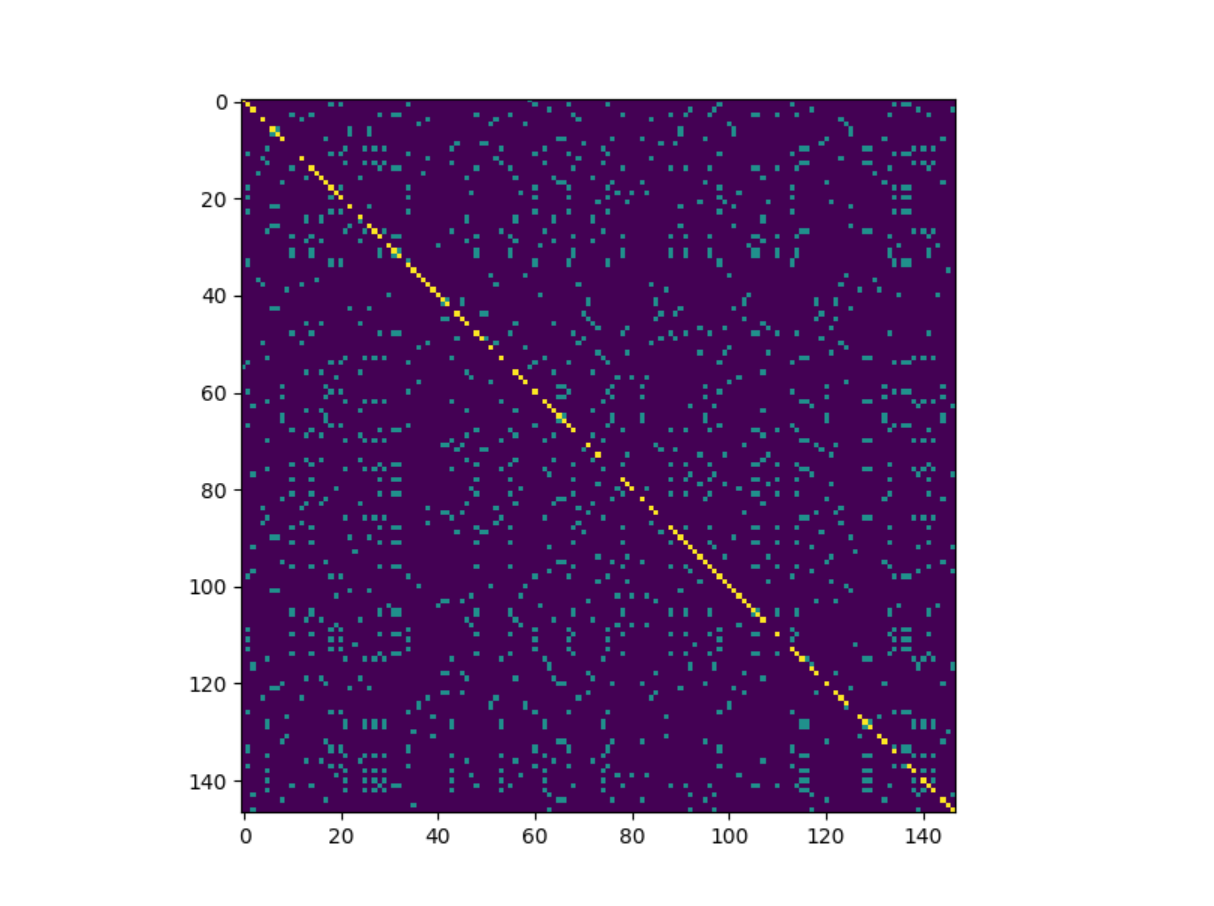
**

*Yellow dots = Same segments in both sequences*

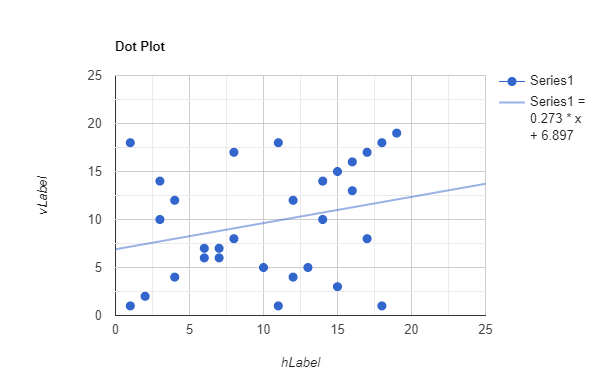
*Green dots = Matched pairs in sequences*

*Violet dots = Unmatched pairs in sequences*

*Dot plot for complete sequences is:*

**

*b) The dot plot for first 20 residues of human and chicken sequences is:*

**

*This dot plot is identical to the dot plot constructed form the code. But only the y-axis is reversed.*

*Hence, verified.*

*2) The code for calculating the score is:*

*#getting sequence input from user*

*seq1 = input("Enter your sequence 1:")*

*seq2 = input("Enter your sequence 2:")*

*match\_score = 1*

*mismatch\_score = 0*

*origination\_penalty = -2*

*length\_penalty = -1*

*#function to find number of gaps*

*def gap\_find(seq):*

*gap\_count=0*

*flag=0*

*for i in range(len(seq)):*

*if(seq[i]=='-'):*

*if(flag==0):*

*gap\_count=gap\_count+1*

*flag=1*

*else:*

*flag=0*

*return gap\_count*

*#function to count total gaps(including consecutive gaps)*

*def gap\_num(seq):*

*gaps = 0*

*for i in range(len(seq)):*

*if(seq[i]=='-'):*

*gaps=gaps+1*

*return gaps*

*#initialising total match and mismatch counts to 0*

*match=0*

*mismatch=0*

*#calculating length penalty(including the gaps)*

*if(len(seq1)==len(seq2)):*

*length\_diff=abs(gap\_num(seq1)-gap\_num(seq2))*

*else:*

*length\_diff=abs(abs(len(seq1)-len(seq2)) - abs(gap\_num(seq1)-gap\_num(seq2)))*

*for i in range(len(seq1)):*

*if(seq1[i]==seq2[i]):*

*match=match+1*

*else:*

*mismatch=mismatch+1*

*#calculating gap penalty*

*gap\_count = gap\_find(seq1) + gap\_find(seq2)*

*#calculating total score for alignment with given match scores*

*score = match\*match\_score + mismatch\*mismatch\_score + gap\_count\*origination\_penalty + length\_diff\*length\_penalty*

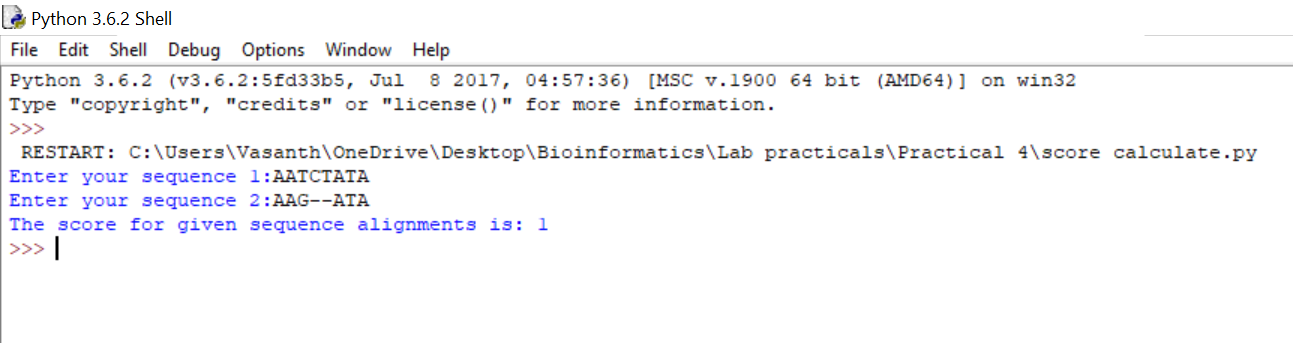
*print("The score for given sequence alignments is:",score)*

*Using the above code, I calculated the sequence alignment score for given sequences:*

*Seq1: AATCTATA*

*Seq2: AAG—ATA*

*The result score is* ***1****.*

**

*3) Given sequences,*

*Seq1: AATCTATA*

*Seq2: AAG—ATA*

*Total matches = 5*

*position 1: A=A*

*position 2: A=A*

*position 6: A=A*

*position 7: T=T*

*position 8: A=A*

*Total mismatches = 1*

*position 3: T ~= G*

*Total Gaps = 1*

*At position 4,5.*

*Length penalty = length of seq1 – length of seq2*

*= 8-6*

*=2*

*Match score = 1; Mismatch score = 0;*

*Origination penalty = -2; Length penalty = -1*

*Therefore,*

*Total score = (5\*1) + (1\*0) + (1\*-2) + (2\*-1)*

*=1*

*Hence, the final score is 1, and it is also the same score we got from code.*

*Hence, verified.*

*4) The code for calculating partial alignment score table using Needleman and Wunsch dynamic programming method is:*

*import numpy as np*

*#given sequences*

*seq1 = "ACAGTCGAACG"*

*seq2 = "ACCGTCCG"*

*#given scores*

*match\_score = 2*

*mismatch\_score = -1*

*gap\_penalty = -2*

*#initialisng a matrix of 0 of size-length of seq1+1 and length of seq2+1*

*mat = np.zeros((len(seq2)+1,len(seq1)+1))*

*#filling gap penalties in first row and column*

*for i in range(1,len(seq1)+1):*

*mat[0][i]=mat[0][i-1] + gap\_penalty*

*for j in range(1,len(seq2)+1):*

*mat[j][0]=mat[j-1][0] + gap\_penalty*

*#function to calculate left score*

*def left\_score(matrix,n,m):*

*a = matrix[n][m-1] + gap\_penalty*

*return a*

*#function to calculate top score*

*def top\_score(matrix,n,m):*

*b = matrix[n-1][m] + gap\_penalty*

*return b*

*#function to calculate diaagonal score*

*def diag\_score(matrix,n,m):*

*if seq2[n-1]==seq1[m-1]:*

*c = matrix[n-1][m-1] + match\_score*

*else:*

*c = matrix[n-1][m-1] + mismatch\_score*

*return c*

*#filling the matrix using formula: maximum of top,left,diagonal score*

*for i in range(1,len(seq2)+1):*

*for j in range(1,len(seq1)+1):*

*a=left\_score(mat,i,j)*

*b=top\_score(mat,i,j)*

*c=diag\_score(mat,i,j)*

*mat[i][j] = max(a,b,c)*

*x=len(seq2)*

*y=len(seq1)*

*#initialising traceback matrix*

*traceback\_mat=[]*

*#tracing back*

*while(x>=0):*

*while(y>=0):*

*if(seq2[x-1]==seq1[y-1]):*

*traceback\_mat.append(1)*

*x=x-1*

*y=y-1*

*else:*

*if(mat[x-1][y-1]>=mat[x-1][y] and mat[x-1][y-1]>=mat[x][y-1]):*

*traceback\_mat.append(0)*

*x=x-1*

*y=y-1*

*elif(mat[x][y-1]>=mat[x-1][y] and mat[x][y-1]>mat[x-1][y-1]):*

*traceback\_mat.append(-1)*

*y=y-1*

*elif(mat[x-1][y]>mat[x][y-1] and mat[x-1][y]>mat[x-1][y-1]):*

*traceback\_mat.append(-2)*

*x=x-1*

*#reversing the traceback matrix*

*traceback\_mat = traceback\_mat[::-1]*

*#initialsing new akignment sequence*

*align\_seq1=""*

*align\_seq2=""*

*#creating aligned sequence using traceback matrix*

*for i in range(1,len(traceback\_mat)):*

*if(traceback\_mat[i]==1):*

*align\_seq1=align\_seq1+seq1[i-1]*

*align\_seq2=align\_seq2+seq1[i-1]*

*elif(traceback\_mat[i]==0):*

*align\_seq1=align\_seq1+seq1[i-1]*

*align\_seq2=align\_seq2+seq2[i-1]*

*elif(traceback\_mat[i]==-1):*

*align\_seq1=align\_seq1+seq1[i-1]*

*align\_seq2=align\_seq2+"-"*

*else:*

*align\_seq1=align\_seq1+"-"*

*align\_seq2=align\_seq2+seq2[i-1]*

*#printing the values*

*print("The aligned sequence 1 is:")*

*print(align\_seq1)*

*print("The aligned sequence 2 is:")*

*print(align\_seq2)*

*print("The partial alignment matrix is:")*

*print(mat)*

*The aligned sequences are:*

*ACAGTCGAACG*

*ACCGTC---CG*

*The alignment table is:*

*[[ 0. -2. -4. -6. -8. -10. -12. -14. -16. -18. -20. -22.]*

*[ -2. 2. 0. -2. -4. -6. -8. -10. -12. -14. -16. -18.]*

*[ -4. 0. 4. 2. 0. -2. -4. -6. -8. -10. -12. -14.]*

*[ -6. -2. 2. 3. 1. -1. 0. -2. -4. -6. -8. -10.]*

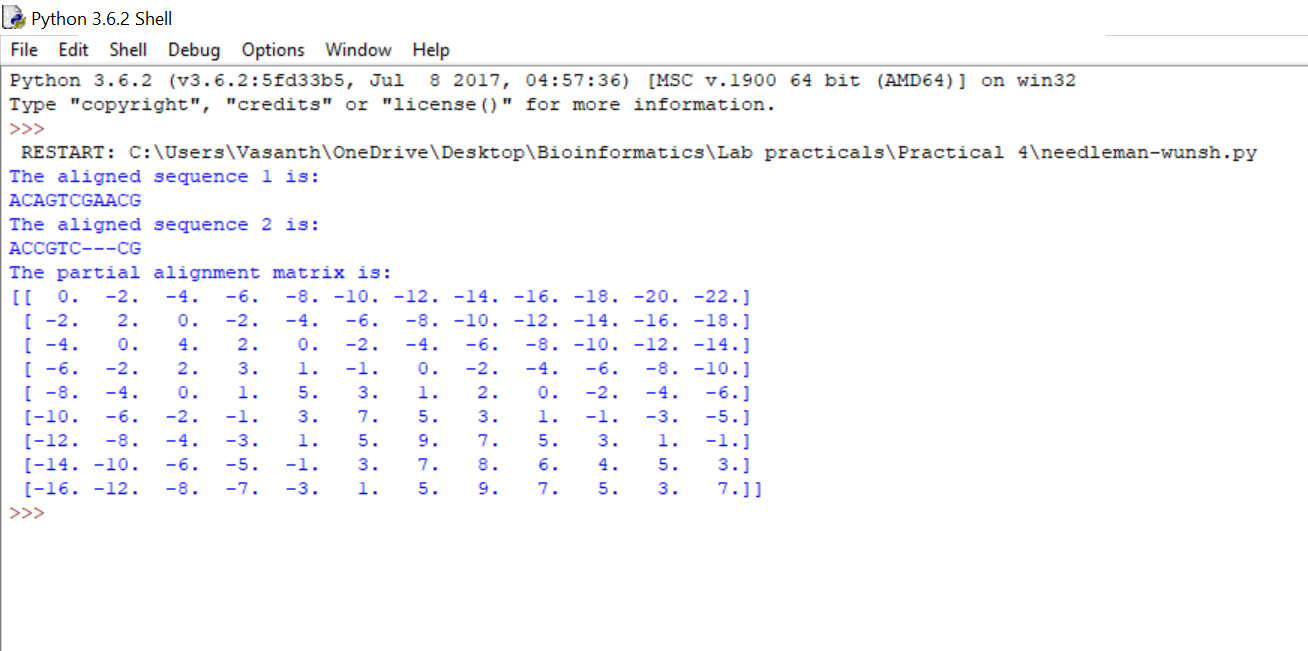
*[ -8. -4. 0. 1. 5. 3. 1. 2. 0. -2. -4. -6.]*

*[-10. -6. -2. -1. 3. 7. 5. 3. 1. -1. -3. -5.]*

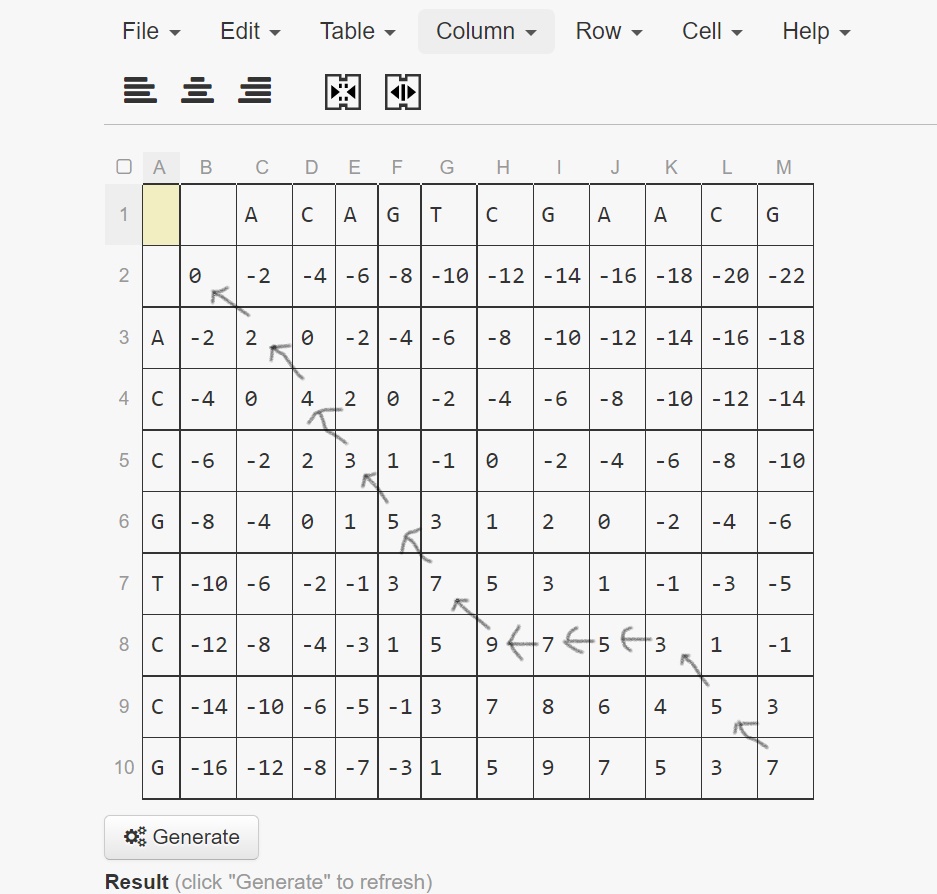
*[-12. -8. -4. -3. 1. 5. 9. 7. 5. 3. 1. -1.]*

*[-14. -10. -6. -5. -1. 3. 7. 8. 6. 4. 5. 3.]*

*[-16. -12. -8. -7. -3. 1. 5. 9. 7. 5. 3. 7.]]*

**

*5) The partial alignment table for given sequences is:*

**

*So, the aligned sequences are:*

*ACAGTCGAACG*

*ACCGTC---CG*

*The score for these sequences are:*

*Matches = 7*

*Mismatches = 1*

*Gaps = 3*

*Score = matches\*(2) + mismatches\*(-1) + gaps\*(-2)*

*= 7\*2 + 1\*(-1) + 3\*(-2)*

*= 7*

*Total score is 7.*

*Hence, verified.*