```
//-----------------------------
public Complex[] acplxConvertSignalToDft(double[] adSignalYValues)
{
   int iNumOfValues = adSignalYValues.Length;
   Complex[] acplxFrequencies = new Complex[iNumOfValues];
   for (int ii = 0; ii < iNumOfValues; ii++) {
      acplxFrequencies[ii] = 0;
      for (int iii = 0; iii < iNumOfValues; iii++) {
         acplxFrequencies[ii] += adSignalYValues[iii] * Complex.Exp(-Complex.ImaginaryOne *
                                                        2 * Math.PI * (ii * iii) /
                                                        Convert.ToDouble(iNumOfValues));
      }
      acplxFrequencies[ii] = acplxFrequencies[ii] / iNumOfValues;
   }
   return acplxFrequencies;
}//acplxConvertSignalToDft
//-----------------------------
//-----------------------------
public double[] adConvertDftToSignal(Complex[] acplxFrequencies)
{
   int iNumOfValues = acplxFrequencies.Length; // Number of spectrum elements
   double[] adInverseDftSignalYValues = new double[iNumOfValues];

   for (int ii = 0; ii < iNumOfValues; ii++) {
      Complex cplxSum = 0;
      for (int iii = 0; iii < iNumOfValues; iii++) {
         cplxSum += acplxFrequencies[iii] * Complex.Exp(Complex.ImaginaryOne *
                                                2 * Math.PI * (iii * ii) /
                                                Convert.ToDouble(iNumOfValues));
      }
      // As a result we expect only real values (if our calculations are correct imaginary
      //  values should be equal or close to zero)
      adInverseDftSignalYValues[ii] = cplxSum.Real;
   }
   return adInverseDftSignalYValues;
}//adConvertDftToSignal
 //-----------------------------
```