AMPLITUDE MODULATION AND DEMODULATION

Course code and Name:

2EC403 Communication System

Submitted by:

21BEC075 Dhruvit Nimavat

21BEC084 Krinal Parmar

AIM: To study the function of Amplitude Modulation & Demodulation

OBJECTIVE:

- Learn about modulation and demodulation process.
- Understand the amplitude modulation theory.
- See how modulation index affects AM signals.

SOFTWARE:

MATLAB

Toolbox used:

- Simulink
- DSP System Toolbox
- Simulink Coder

CODE:

```
/*
 * AM_MODN.c
 *
 * Academic License - for use in teaching, academic research, and meeting
 * course requirements at degree granting institutions only.  Not for
 * government, commercial, or other organizational use.
 *
 * Code generation for model "AM_MODN".
 *
 * Model version              : 1.8
 * Simulink Coder version : 9.9 (R2023a) 19-Nov-2022
 * C source code generated on : Fri Apr 28 12:19:11 2023
```

```
 *
 * Target selection: grt.tlc
 * Note: GRT includes extra infrastructure and instrumentation for
prototyping
 * Embedded hardware selection: Intel->x86-64 (Windows64)
 * Code generation objective: Execution efficiency
 * Validation result: Not run
 */

#include "AM_MODN.h"
#include <string.h>
#include <math.h>
#include "rtwtypes.h"
#include "AM_MODN_private.h"

/* Block signals (default storage) */
B_AM_MODN_T AM_MODN_B;

/* Continuous states */
X_AM_MODN_T AM_MODN_X;

/* Real-time model */
static RT_MODEL_AM_MODN_T AM_MODN_M_;
RT_MODEL_AM_MODN_T *const AM_MODN_M = &AM_MODN_M_;
```

```
/*
 * This function updates continuous states using the ODE3 fixed-step
 * solver algorithm
 */
static void rt_ertODEUpdateContinuousStates(RTWSolverInfo *si )
{
  /* Solver Matrices */
  static const real_T rt_ODE3_A[3] = {
    1.0/2.0, 3.0/4.0, 1.0
  };

  static const real_T rt_ODE3_B[3][3] = {
    { 1.0/2.0, 0.0, 0.0 },

    { 0.0, 3.0/4.0, 0.0 },

    { 2.0/9.0, 1.0/3.0, 4.0/9.0 }
  };

  time_T t = rtsiGetT(si);
  time_T tnew = rtsiGetSolverStopTime(si);
  time_T h = rtsiGetStepSize(si);
```

```
real_T *x = rtsiGetContStates(si);
ODE3_IntgData *id = (ODE3_IntgData *)rtsiGetSolverData(si);
real_T *y = id->y;
real_T *f0 = id->f[0];
real_T *f1 = id->f[1];
real_T *f2 = id->f[2];
real_T hB[3];
int_T i;
int_T nXc = 8;
rtsiSetSimTimeStep(si,MINOR_TIME_STEP);

/* Save the state values at time t in y, we'll use x as ynew. */
(void) memcpy(y, x,
              (uint_T)nXc*sizeof(real_T));

/* Assumes that rtsiSetT and ModelOutputs are up-to-date */
/* f0 = f(t,y) */
rtsiSetdX(si, f0);
AM_MODN_derivatives();

/* f(:,2) = feval(odefile, t + hA(1), y + f*hB(:,1), args(:)(*)); */
hB[0] = h * rt_ODE3_B[0][0];
for (i = 0; i < nXc; i++) {
```

```
  x[i] = y[i] + (f0[i]*hB[0]);
}

rtsiSetT(si, t + h*rt_ODE3_A[0]);
rtsiSetdX(si, f1);
AM_MODN_step();
AM_MODN_derivatives();

/* f(:,3) = feval(odefile, t + hA(2), y + f*hB(:,2), args(:)(*)); */
for (i = 0; i <= 1; i++) {
  hB[i] = h * rt_ODE3_B[1][i];
}

for (i = 0; i < nXc; i++) {
  x[i] = y[i] + (f0[i]*hB[0] + f1[i]*hB[1]);
}

rtsiSetT(si, t + h*rt_ODE3_A[1]);
rtsiSetdX(si, f2);
AM_MODN_step();
AM_MODN_derivatives();

/* tnew = t + hA(3);
```

```
    ynew = y + f*hB(:,3); */
  for (i = 0; i <= 2; i++) {
    hB[i] = h * rt_ODE3_B[2][i];
  }

  for (i = 0; i < nXc; i++) {
    x[i] = y[i] + (f0[i]*hB[0] + f1[i]*hB[1] + f2[i]*hB[2]);
  }

  rtsiSetT(si, tnew);
  rtsiSetSimTimeStep(si,MAJOR_TIME_STEP);
}

/* Model step function */
void AM_MODN_step(void)
{
  real_T Product_tmp;
  real_T Product_tmp_0;
  if (rtmIsMajorTimeStep(AM_MODN_M)) {
    /* set solver stop time */
    rtsiSetSolverStopTime(&AM_MODN_M->solverInfo,((AM_MODN_M-
>Timing.clockTick0+
      1)*AM_MODN_M->Timing.stepSize0));
  }                                    /* end MajorTimeStep */
```

```
/* Update absolute time of base rate at minor time step */
if (rtmIsMinorTimeStep(AM_MODN_M)) {
  AM_MODN_M->Timing.t[0] = rtsiGetT(&AM_MODN_M->solverInfo);
}

/* SignalGenerator: '<Root>/Message Signal m(t)' incorporates:
 *  SignalGenerator: '<Root>/Carrier wave'
 */
Product_tmp = AM_MODN_M->Timing.t[0];

/* SignalGenerator: '<Root>/Carrier wave' incorporates:
 *  SignalGenerator: '<Root>/Local oscillator'
 */
Product_tmp_0 = sin(2000.0 * Product_tmp);

/* Product: '<Root>/Product' incorporates:
 *  Constant: '<Root>/DC offset'
 *  Gain: '<Root>/Modulation Index'
 *  Product: '<Root>/Product Modulator'
 *  SignalGenerator: '<Root>/Carrier wave'
 *  SignalGenerator: '<Root>/Local oscillator'
 *  SignalGenerator: '<Root>/Message Signal m(t)'
```

```
   *  Sum: '<Root>/Adder'
   */
  AM_MODN_B.Product = (sin(100.0 * Product_tmp) * 0.7 + 1.0) * (10.0 *
    Product_tmp_0) * (15.0 * Product_tmp_0);
  if (rtmIsMajorTimeStep(AM_MODN_M)) {
    rt_ertODEUpdateContinuousStates(&AM_MODN_M->solverInfo);

    /* Update absolute time for base rate */
    /* The "clockTick0" counts the number of times the code of this task has
     * been executed. The absolute time is the multiplication of "clockTick0"
     * and "Timing.stepSize0". Size of "clockTick0" ensures timer will not
     * overflow during the application lifespan selected.
     */
    ++AM_MODN_M->Timing.clockTick0;
    AM_MODN_M->Timing.t[0] = rtsiGetSolverStopTime(&AM_MODN_M-
>solverInfo);

    {
      /* Update absolute timer for sample time: [0.0010472300764477955s,
0.0s] */
      /* The "clockTick1" counts the number of times the code of this task
has
       * been executed. The resolution of this integer timer is
0.0010472300764477955, which is the step size
       * of the task. Size of "clockTick1" ensures timer will not overflow
during the
```

```
       * application lifespan selected.
       */
      AM_MODN_M->Timing.clockTick1++;
    }
  }                                    /* end MajorTimeStep */
}

/* Derivatives for root system: '<Root>' */
void AM_MODN_derivatives(void)
{
  XDot_AM_MODN_T *_rtXdot;
  int_T is;
  uint32_T ri;
  _rtXdot = ((XDot_AM_MODN_T *) AM_MODN_M->derivs);

  /* Derivatives for StateSpace: '<Root>/LPF' */
  memset(&_rtXdot->LPF_CSTATE[0], 0, sizeof(real_T) << 3U);
  for (is = 0; is < 8; is++) {
    for (ri = AM_MODN_ConstP.LPF_A_jc[(uint32_T)is]; ri <
         AM_MODN_ConstP.LPF_A_jc[(uint32_T)is + 1U]; ri++) {
      _rtXdot->LPF_CSTATE[AM_MODN_ConstP.LPF_A_ir[ri]] +=
        AM_MODN_ConstP.LPF_A_pr[ri] *
AM_MODN_X.LPF_CSTATE[(uint32_T)is];
    }
```

```
  }

  _rtXdot->LPF_CSTATE[0U] += 100.0 * AM_MODN_B.Product;

  /* End of Derivatives for StateSpace: '<Root>/LPF' */
}

/* Model initialize function */
void AM_MODN_initialize(void)
{
  /* Registration code */

  /* initialize real-time model */
  (void) memset((void *)AM_MODN_M, 0,
                sizeof(RT_MODEL_AM_MODN_T));

  {
    /* Setup solver object */
    rtsiSetSimTimeStepPtr(&AM_MODN_M->solverInfo, &AM_MODN_M-
>Timing.simTimeStep);
    rtsiSetTPtr(&AM_MODN_M->solverInfo, &rtmGetTPtr(AM_MODN_M));
    rtsiSetStepSizePtr(&AM_MODN_M->solverInfo, &AM_MODN_M-
>Timing.stepSize0);
    rtsiSetdXPtr(&AM_MODN_M->solverInfo, &AM_MODN_M->derivs);
```

```
    rtsiSetContStatesPtr(&AM_MODN_M->solverInfo, (real_T **)
                         &AM_MODN_M->contStates);
    rtsiSetNumContStatesPtr(&AM_MODN_M->solverInfo,
      &AM_MODN_M->Sizes.numContStates);
    rtsiSetNumPeriodicContStatesPtr(&AM_MODN_M->solverInfo,
      &AM_MODN_M->Sizes.numPeriodicContStates);
    rtsiSetPeriodicContStateIndicesPtr(&AM_MODN_M->solverInfo,
      &AM_MODN_M->periodicContStateIndices);
    rtsiSetPeriodicContStateRangesPtr(&AM_MODN_M->solverInfo,
      &AM_MODN_M->periodicContStateRanges);
    rtsiSetErrorStatusPtr(&AM_MODN_M->solverInfo,
(&rtmGetErrorStatus(AM_MODN_M)));
    rtsiSetRTModelPtr(&AM_MODN_M->solverInfo, AM_MODN_M);
  }


  rtsiSetSimTimeStep(&AM_MODN_M->solverInfo, MAJOR_TIME_STEP);
  AM_MODN_M->intgData.y = AM_MODN_M->odeY;
  AM_MODN_M->intgData.f[0] = AM_MODN_M->odeF[0];
  AM_MODN_M->intgData.f[1] = AM_MODN_M->odeF[1];
  AM_MODN_M->intgData.f[2] = AM_MODN_M->odeF[2];
  AM_MODN_M->contStates = ((X_AM_MODN_T *) &AM_MODN_X);
  rtsiSetSolverData(&AM_MODN_M->solverInfo, (void *)&AM_MODN_M-
>intgData);
  rtsiSetIsMinorTimeStepWithModeChange(&AM_MODN_M->solverInfo,
false);
```

```
  rtsiSetSolverName(&AM_MODN_M->solverInfo,"ode3");
  rtmSetTPtr(AM_MODN_M, &AM_MODN_M->Timing.tArray[0]);
  AM_MODN_M->Timing.stepSize0 = 0.0010472300764477955;

  /* block I/O */
  (void) memset(((void *) &AM_MODN_B), 0,
                sizeof(B_AM_MODN_T));

  /* states (continuous) */
  {
    (void) memset((void *)&AM_MODN_X, 0,
                  sizeof(X_AM_MODN_T));
  }

  /* InitializeConditions for StateSpace: '<Root>/LPF' */
  memset(&AM_MODN_X.LPF_CSTATE[0], 0, sizeof(real_T) << 3U);
}

/* Model terminate function */
void AM_MODN_terminate(void)
{
  /* (no terminate code required) */
}
```

THEORY:

Modulation: It is defined as the process of superimposing a low-frequency signal on a high-frequency carrier signal.

There are mainly three types of modulation:

(1) Amplitude Modulation, (2) Frequency Modulation, (3) Phase Modulation.

Here, we are going to learn about Amplitude Modulation.

Amplitude Modulation: The modulation process in which the peak amplitude of the carrier wave is varied in proportion to the message signal.

Here, the phase and frequency are kept constant.

m(t) = message signal

c(t) = carrier signal

s(t) = amplitude modulated signal

We can write the above signals as,

$m(t) = A_m \cos(2\pi f_m t + \phi)$

$c(t) = A_c \sin(2\pi f_c t)$

Therefore, we can put the values of m(t) and c(t) in s(t) as given below

$s(t) = [1 + k_a m(t)]\, c(t)$; where k_a = amplitude sensitivity ; $0<k_a<1$

$s(t) = [1 + k_a m(t)]\, A_c \sin(2\pi f_c t)$

$s(t) = A_c \sin(2\pi f_c t) + A_c k_a m(t) \sin(2\pi f_c t)$

Taking the Fourier transform of the above equation,

$s(f) = \int_{-\infty}^{\infty} s(t) e^{-j\omega t}\, dt$

$|s(f)| = \frac{A_c}{2} [\delta(f - f_c) - \delta(f + f_c) + \frac{A_c k_a}{2} [M(f - f_c) - M(f + f_c)]$

In the equation of s(t), max $|k_a m(t)| = \mu$ is called the modulation index.

If,

$\mu<1$ → under modulation

$\mu=1$ → perfect modulation

$\mu>1$ → over modulation

$$\mu = \frac{A_m}{A_c} = \frac{A_{max} - A_{min}}{A_{max} + A_{min}}$$

where,

A_m = peak amplitude of message signal

A_c = peak amplitude of carrier wave

Demodulation: Demodulation is defined as extracting the original information-carrying signal from a modulated carrier wave. A demodulator is an electronic circuit that is mainly used to recover the information content from the modulated carrier wave.

The circuit here used to demodulate the signal is known as the Coherent Detector.

Advantages Of Amplitude Modulation:

- It is simple to implement.
- Demodulation of AM signals can be done using simple circuits consisting of diodes.
- AM transmitters are less complex.
- AM receivers are very cheap as no specialized components are needed.
- AM waves can travel a longer distance.
- AM waves have low bandwidth

Disadvantages of Amplitude Modulation:

- An amplitude modulation signal is not efficient in terms of its power usage.
- It is not efficient in terms of its use of bandwidth. It requires a bandwidth equal to twice that of the highest audio frequency. In amplitude modulation sidebands contain the signal. The power in sidebands is the only useful power. For 100 % modulation, the power carried by AM waves is 33.3 %. The power carried by the AM wave decreases with the decrease in the extent of modulation.
- AM detectors are sensitive to noise hence an amplitude modulation signal is prone to high levels of noise.

As amplitude modulation doesn't provide 100% of efficiency in modulation and usage of large amount of power, DSB-SC modulation technique is used.

DSB-SC (Double Sideband Suppressed Carrier) modulation:

DSB-SC is an amplitude modulated wave transmission scheme in which only sidebands are transmitted and the carrier is not transmitted as it gets suppressed.

The carrier does not contain any information and its transmission results in loss of power. Thus, only sidebands are transmitted that contains information. This results in saving of power used in transmission.

Advantages of DSB-SC modulation:

- It provides 100% modulation efficiency.
- Due to suppression of carrier, it consumes less power.
- It provides a larger bandwidth.

Disadvantages of DSB-SC modulation:

- It involves a complex detection process.
- Using this technique, it is sometimes difficult to recover the signal at the receiver.
- It is an expensive technique when it comes to demodulation of the signal.

To get better modulated signal, we use SSB-SC modulation technique. As it has less bandwidth requirements, and lots of power saving. At 100% modulation, the percent power saving is 83.33%.

SSB-SC (Single Sideband Suppressed Carrier) modulation:

The process of suppressing one of the sidebands along with the carrier and transmitting a single sideband is called as Single Sideband Suppressed Carrier system or simply SSBSC.

This SSBSC system, which transmits a single sideband has high power, as the power allotted for both the carrier and the other sideband is utilized in transmitting this Single Sideband.

Advantages of DSB-SC modulation:

- Bandwidth or spectrum space occupied is lesser than AM and DSB-SC waves.
- Transmission of more number of signals is allowed.
- Power is saved.
- High power signal can be transmitted.
- Less amount of noise is present.
- Signal fading is less likely to occur.

Disadvantages of DSB-SC modulation:

- The generation and detection of SSB-SC wave is a complex process.
- The quality of the signal gets affected unless the SSB transmitter and receiver have an excellent frequency stability.

Clearly, SSB-SC conveys the information in the message signal while halving the bandwidth compared with DSB. The reason for leaving a part of one sideband in VSB is to allow for easier sideband filter realization. And therefore, we use VSB-SC modulation.

VSB-SC (Vestigial Sideband Suppressed Carrier) modulation:

Vestigial Sideband (VSB) modulation is a modulation technique which allows transmission of one sideband in addition with a part or vestige of the other.

As SSB modulation requires accurate frequency response of the filter to transmit only one sideband completely. Thus by using VSB modulation one can simplify the design of the filter to a great extent.

Advantages of VSB-SC modulation:

- It is a highly efficient modulation technique used for wave transmission.
- It reduces the bandwidth utilization.
- The filter characteristics do not need to be highly accurate thus making its design simple.
- It easily transmits low-frequency components and possesses good phase characteristics.

Disadvantages of VSB-SC modulation:

- Its bandwidth requirement is somewhat higher than that of SSB modulation, due to the presence of vestige.
- Vestigial sideband modulation leads to a complex demodulation process at the receiver end.

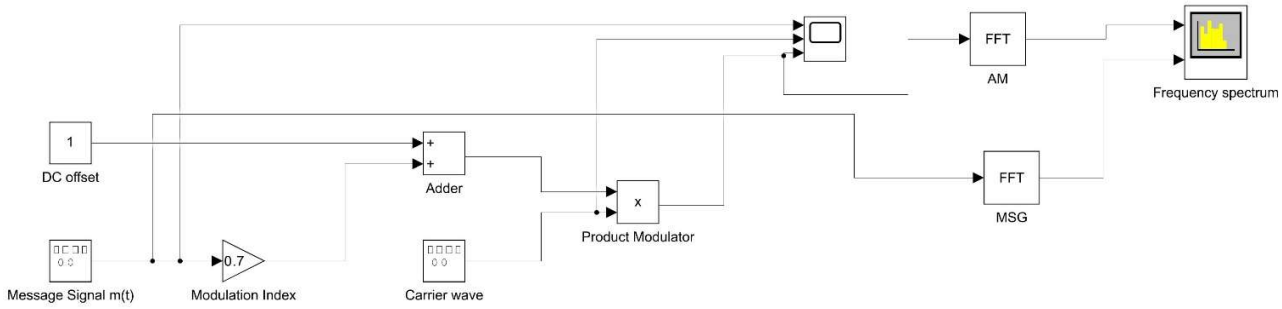
Application of Amplitude Modulation:

- Communication between Aircraft: AM is more commonly utilized for pilot-to-ground control communications than other approaches like FM because it can receive numerous signals on the same channel.

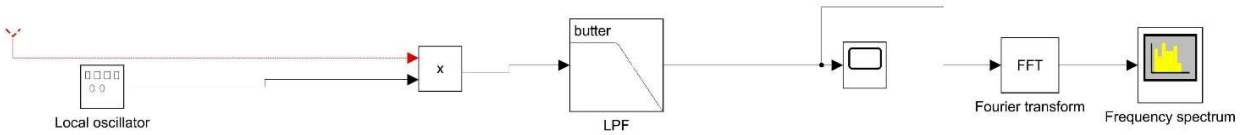
- Broadcast Transmission: Long, medium, and short-wave bands, are still served by AM. Equipment is less expensive even though it is simple to demodulate and has a low production cost.
- Quadrature Amplitude Modulation: Two carriers are combined throughout this approach and are 90° out of phase. They are commonly employed in relatively brief wireless communication networks such as WIFI and cellular networks.
- Single sideband: Regarding high-frequency radio connections, a singular sideband of amplitude modulation waves has always been employed. Lower bandwidth is also employed, allowing for more effective use of transmitted power. This is still utilized for many high-frequency point-to-point links.

BLOCK DIAGRAM:

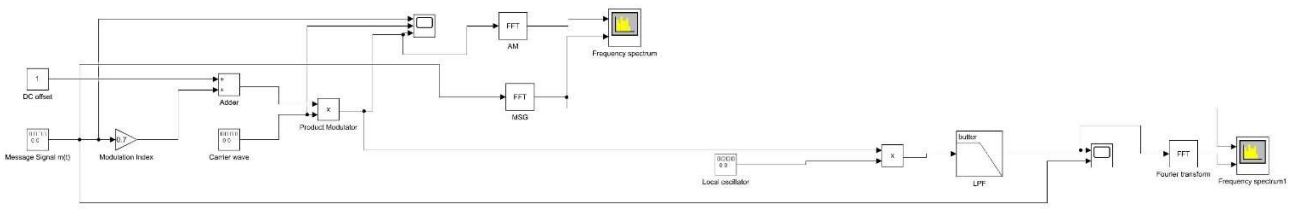
For Amplitude Modulation:



For Amplitude Demodulation:

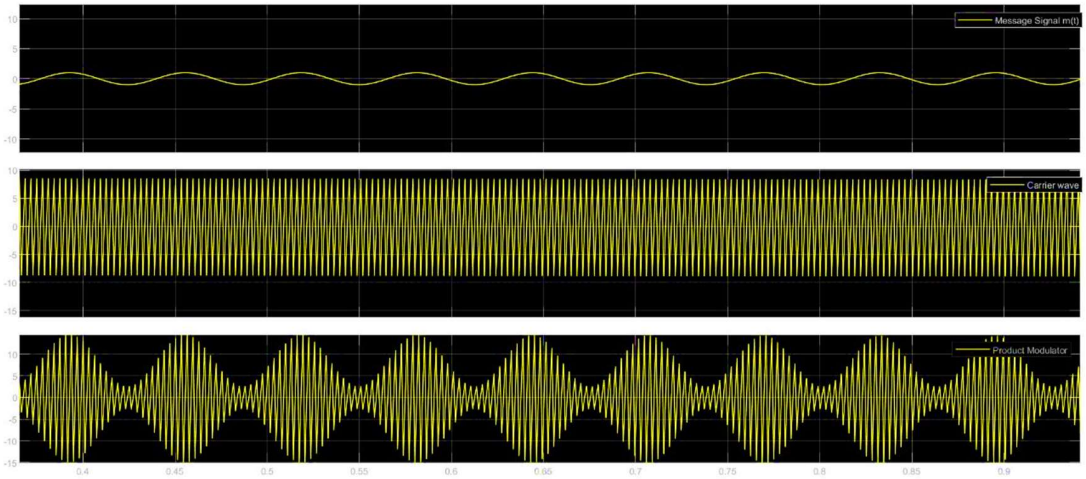


CIRCUIT:

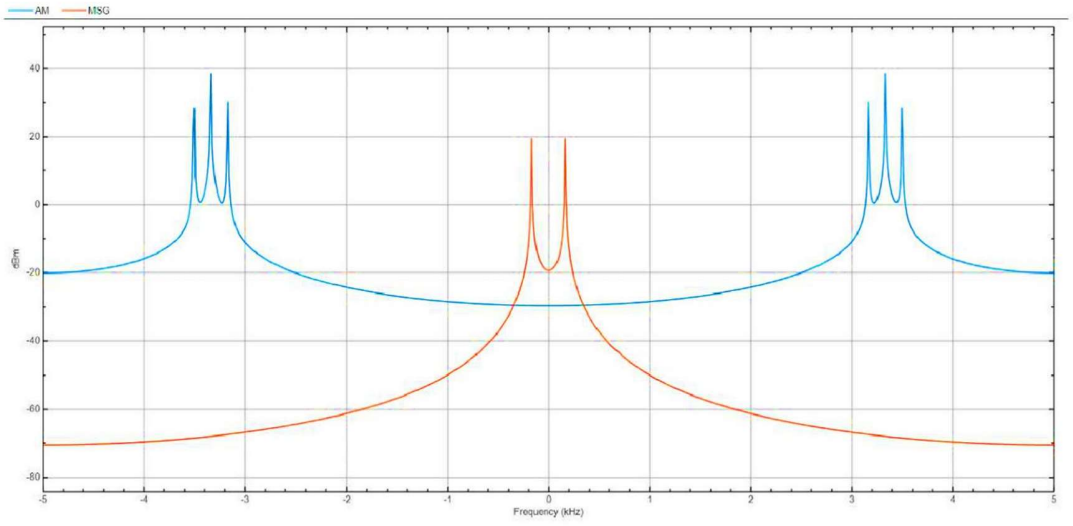


OBSERVATION:

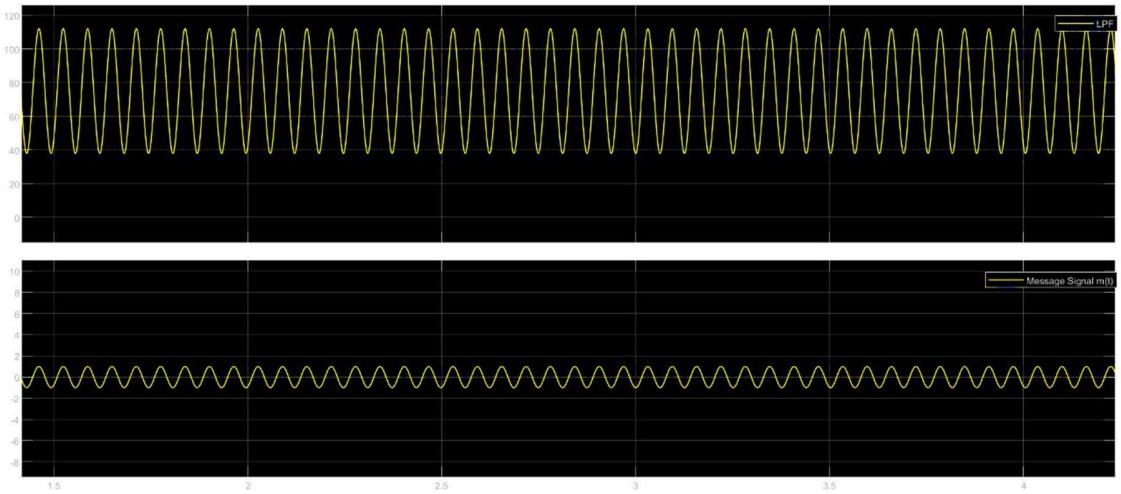
Amplitude Modulated Waveform (in time domain):



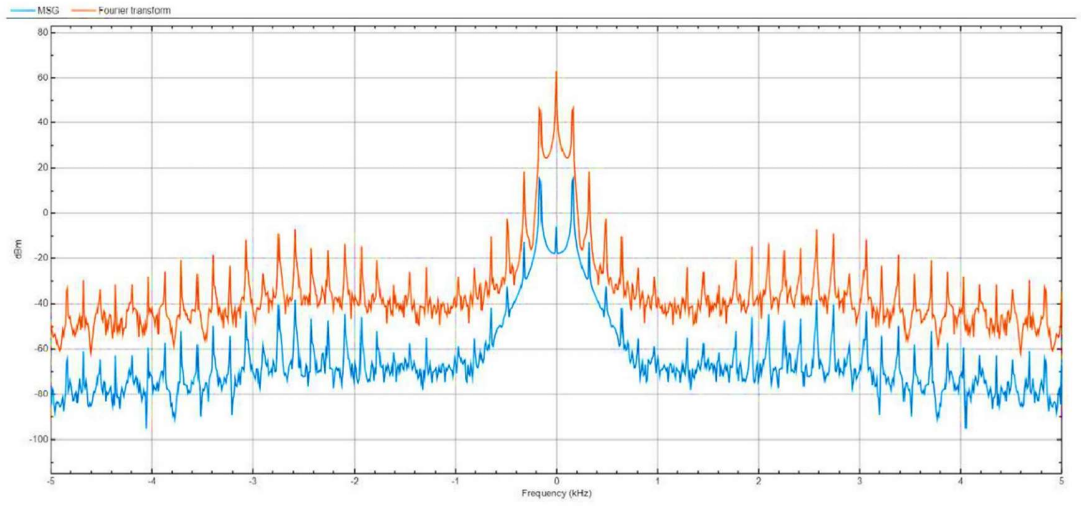
Amplitude Modulated Waveform (in frequency domain):



Amplitude Demodulated Waveform (in time domain):



Amplitude Demodulated Waveform (in frequency domain):



CONCLUSION:

From the above, we conclude that amplitude modulation is achieved by multiplying carrier and message signals. Demodulation is achieved by sampling the AM signal at carrier frequency. This technique is used in many areas of communication, such as in portable two-way radios, citizens band radios, VHF aircraft radios and in modems for computers.