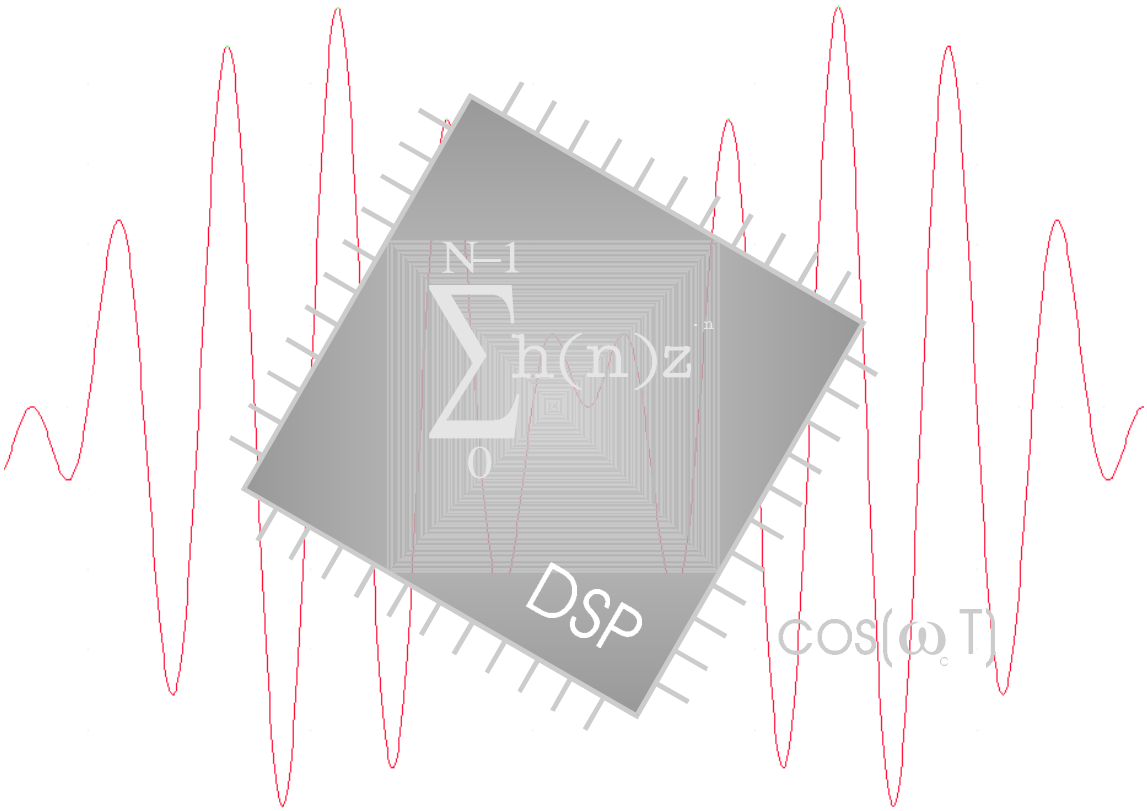
# Lab-Report
# Digital Signal Processing

## *Amplitude Modulation*

using DSP methods





| | |
|---|---|
| Name: | Dirk Becker |
| Course: | BEng 2 |
| Group: | A |
| Student No.: | 9801351 |
| Date: | 28/01/1999 |

UNIVERSITY of EAST LONDON

# 1. Contents

## 2. Introduction

Over very many years Amplitude Modulation was the standard modulation scheme because of it's easy and cheap realisation. For Amplitude modulation and demodulation multipliers are used, which are realised by semiconductors or in earlier times by vacuum tubes.
A more modern way of amplitude modulation is the use of Digital Signal Processing methods. The carrier generation and also the multiplication are done by a DSP Software.
Amplitude Modulation is today still used because of the small consumption of bandwidth in the broadcast bands.

## 3. Amplitude Modulation Systems

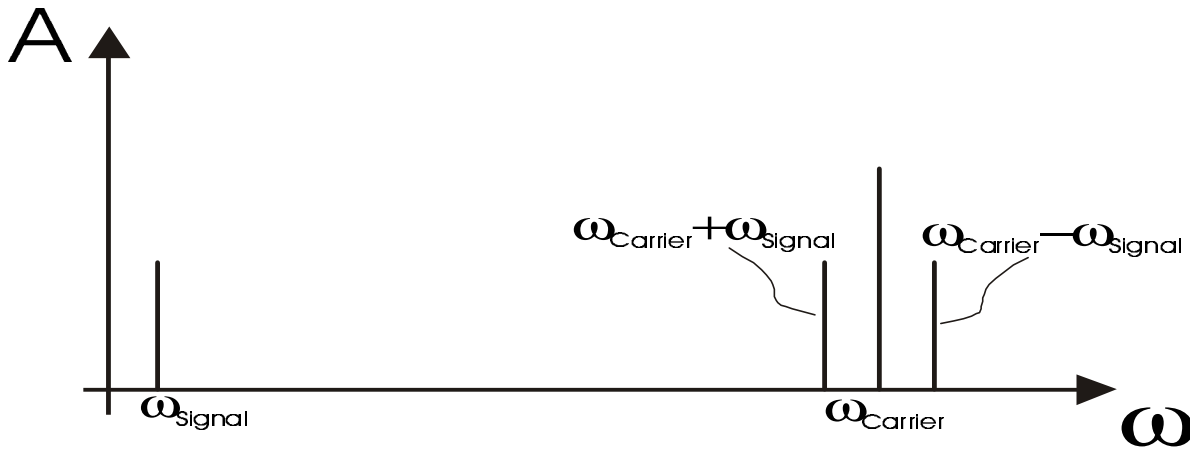



**figure 1 - Standard AM - System**

Figure 1 shows the block diagram of an AM modulation and demodulation system. The major blocks are the two multipliers and the low pass filter to remove the high frequency parts of the down-mixed signal.
AM modulation simply means the shifting of a signal frequency to another (usually higher) frequency. The information, or better the content of the original (modulating) signal is transferred to another frequency, the carrier.
Frequency shifting is done by multiplication of two signal in the time domain. Multiplication in the time-domain correspondents with frequency shifting in the frequency (ω) domain.

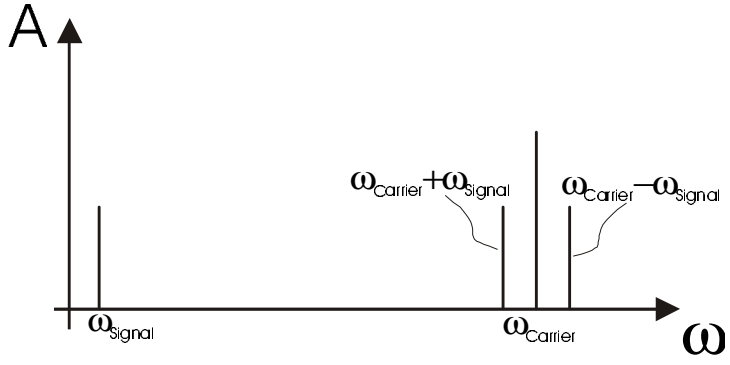



**figure 2 - Standard Amplitude modulation**

## a) Mathematical justification for the frequency shifting

The Frequency shifting can be proofed by applying the Fourier Transform to a function f(t) multiplied with an cosine function.

$$f(t)\cos\omega_0 t = \frac{1}{2}\left(f(t)e^{j\omega_0 t} + f(t)e^{-j\omega_0 t}\right) \quad \text{where f(t) = modulating signal and } \omega_0 = \text{carrier signal}$$

$$f(t)\cos\omega_0 t \Leftrightarrow \frac{1}{2}\left(F(\omega-\omega_0) + F(\omega-\omega_0)\right)$$

This shows that multiplication of a signal f(t) by a sinusoid frequency $\omega_0$ shifts the spectrum $F(\omega)$ by $\pm\omega_0$. Multiplication of a sinusoid $\cos\omega_0 t$ by f(t) amounts to modulating the sinusoid amplitude. This kind of modulation is called (balanced) amplitude modulation.

## b) Standard amplitude modulation

Balance amplitude modulation, like shown above results in loss of the carrier signal, which carries only redundant information. But for different reasons the carrier is transmitted at standard amplitude modulation. Therefore an offset is added to the carrier and the carrier is transmitted as well.

**Mathematical justification for standard amplitude modulation:**

$$V_m = A\sin(\omega_c t)(1 + m\cos(\omega_{mod} t))$$

$$V_m = A\left(-\frac{1}{2j}\right)\left(e^{j\omega_c t} - e^{-j\omega_c t}\right)\left(1 + m\frac{1}{2}\left(e^{j\omega_{mod} t} + e^{-j\omega_{mod} t}\right)\right)$$

$$V_m = A\left(-\frac{1}{2j}\right)\left(e^{j\omega_c t} - e^{-j\omega_c t}\right)\left(1 + m\frac{1}{2}\left(e^{j\omega_{mod} t} + e^{-j\omega_{mod} t}\right)\right)$$

$$V_m = -\frac{A}{2j}\left(e^{j\omega_c t} + e^{-j\omega_c t}\right) - \frac{Am}{4j}\left(e^{j\omega_{mod} t} + e^{-j\omega_{mod} t}\right)\left(e^{j\omega_c t} - e^{-j\omega_c t}\right)$$

$$V_m = -\frac{A}{2j}\left(e^{j\omega_c t} + e^{-j\omega_c t}\right) - \frac{Am}{4j}\left(e^{j(\omega_{mod}+\omega_c)t} - e^{-j(\omega_{mod}+\omega_c)t} + e^{j(\omega_{mod}-\omega_c)t} - e^{-j(\omega_{mod}-\omega_c)t}\right)$$

$$V_m = A\sin(\omega_c t) - \frac{Am}{4j}\left[-2j\sin((\omega_{mod} + \omega_c)t) - 2j\sin((\omega_{mod} - \omega_c)t)\right]$$

$$V_m = A\sin(\omega_c t) + \frac{Am}{2}\left[\sin((\omega_{mod} + \omega_c)t) + \sin((\omega_{mod} - \omega_c)t)\right]$$

where m is called the modulation index (ratio of peak modulating signal to peak carrier signal), and A is the amplitude of the carrier signal.

Standard amplitude modulated signals can be demodulated by means of simple. diodes.

## c) Balanced Amplitude Modulation

The opportunity of balanced amplitude modulation is the suppression of the carrier signal, which contains no useful information and consumes a lot of energy, when transmitting. Hence balanced Amplitude Modulation can be described as:

$V_m=Af(t)\cos(\omega_{carrier}t)$ where f(t) is the modulating signal and $\omega_{carrier}$ the carrier frequency.

The main difference in generating balanced amplitude modulation to standard amplitude modulation is the missing offset of the (1) offset in the modulating signal.

**Mathematical justification for balanced amplitude modulation:**
(see also a) Mathematical justification for the frequency shifting)

$$V_m = A\sin(\omega_c t)m\cos(\omega_{mod} t)$$

$$V_m = A\left(-\frac{1}{2j}\right)\left(e^{jw_c t} - e^{jw_{mod} t}\right)\frac{1}{2}\left(e^{jw_{mod} t} + e^{-jw_{mod} t}\right)$$

$$V_m = -\frac{Am}{4j}\left(e^{j(w_c+\omega_{mod})t} - e^{-j(w_c+\omega_{mod})t} + e^{j(w_c-\omega_{mod})t}e^{-j(w_c-\omega_{mod})t}\right)$$

$$V_m = -\frac{Am}{4j}\left(-\frac{1}{2j}\sin(\omega_c+\omega_{mod}) - \frac{1}{2j}\sin(\omega_c-\omega_{mod})\right)$$

$$V_m = -\frac{Am}{2j}\left(-j\sin(\omega_c+\omega_{mod}) - j\sin(\omega_c-\omega_{mod})\right)$$

$$V_m = \frac{A\times m}{2}\left(\sin(\omega_c+\omega_{mod}) + \sin(\omega_c-\omega_{mod})\right)$$

where m is the modulation index and A the amplitude of the carrier signal.

# 4. Amplitude Modulation using DSP-methods

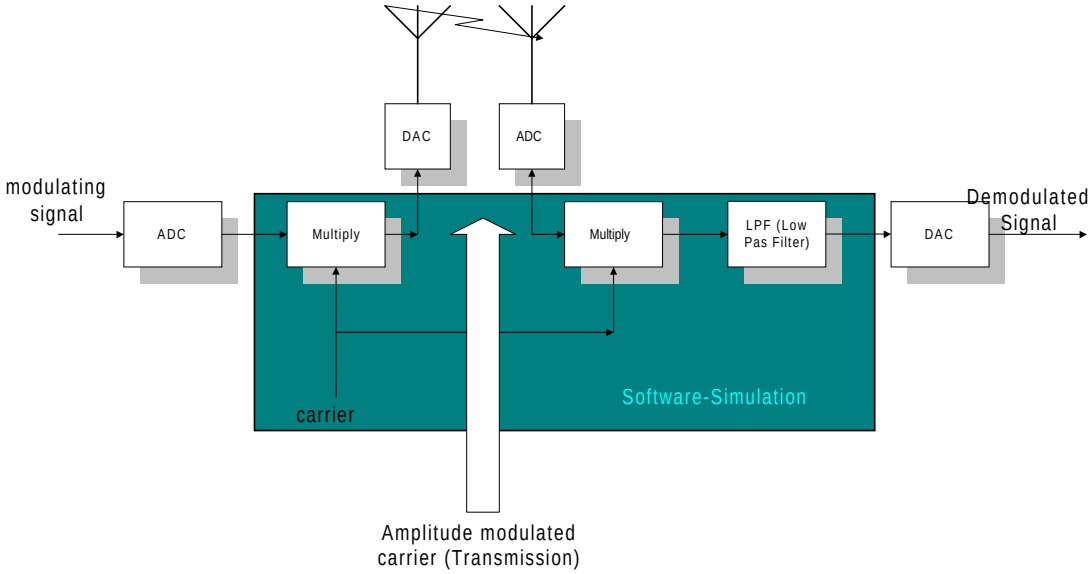



**figure 3 - Amplitude modulation using DSP methods with analogue transmission**

In Amplitude Modulation Systems using Digital Signal Processing methods the generation of the carrier and the arithmetic (multiplication and carrier generation) are done via a Digital Signal Processor. The modulating signal is fed via an ADC (Analogue to Digital Converter) to the system. The transmission (the channel) can be either digital or analogue. For an analogue transmission the digital amplitude modulated carrier must be converted to an analogue one and back to a digital at the receiver.
Objectives of the Lab were to design an Amplitude Modulation and Demodulation System simulated in C. The modulating signal was to implement via a sinus function and the carrier signal was to implement via a $2^{nd}$ order IIR filter.
The low pass filter to remove the remaining was to realise by means of a FIR filter, a so called convolver.

## a) The Digital Oscillator

The oscillator for the carrier signal was to implement by means of a 2nd order recursive Function.
The 2nd Order differential equation for an analogue oscillator was given by:

$$\frac{d^2y(t)}{dt^2} + \omega_0{}^2 y(t) = \omega_0{}^2 x(t)$$

Applying backwards difference approximation leads to:

$$\frac{d^2y(t)}{dt^2} + \omega_0{}^2 y(t) = \omega_0{}^2 x(t)$$

$$\frac{[y(nT) - y(nT-T)]/T - [y(nT-T) - y(nT-2T)]/T}{T} + \omega_0{}^2 y(nT) = \omega_0{}^2 x(nT)$$

$$\frac{y(n) - 2y(n-1) + y(n-2)}{T^2} + \omega_0{}^2 y(n) - \omega_0{}^2 x(n) = 0$$

$$y(n) - 2y(n-1) + y(n-2) + \omega_0{}^2 T^2 y(n) - \omega_0{}^2 T^2 x(n) = 0$$

$$y(n) - \frac{2y(n-1)}{1+\omega_o{}^2 T^2} + \frac{y(n-2)}{1+\omega_o{}^2 T^2} - \frac{\omega_0{}^2 T^2 x(n)}{1+\omega_o{}^2 T^2} = 0$$

$$y(n) = \frac{1}{1+\omega_o{}^2 T^2} x(n) + \frac{2}{1+\omega_o{}^2 T^2} y(n-1) - \frac{1}{1+\omega_o{}^2 T^2} y(n-2)$$

.... where x(n) is usually an impulse to start the system

The problem of the backward difference approximation is, that it changes the frequency of the oscillator and the capability to oscillate without signal loss because of the change of the coefficients (Transforms in Signals and Systems, Peter Kraniauskas, Addison-Wesley, Page 321).
A better calculation of the coefficients results in the following equation (C Algorithms for Real-Time DSP, Paul M. Embree, Prentice Hall, :1995, Page 178)

$$y(n) = c_1 y(n-1) - c_2 y(n-2) + x(n)$$

where $c_1 = 2e^{-dT}\cos(\omega T)$ and $c_2 = e^{-2dT}$

$$T = \frac{1}{f_{sample}}, \quad d = \text{damping}(=0), \quad \omega = 2\pi f_{Oscillate}$$

For the capability of oscillating the poles of the Z-Transfer function have to be on the unit circle, which is not given by the backward difference approximation.
The frequency of the carrier signal can be easily checked by comparing with the modulation frequency. The ratio should be 1/10.

## b) Modulation and demodulation

Hence the modulation and the demodulation is only based on simple multiplication, they can be realised very simple only using standard multiplication provided by the DSP circuit.

## c) Low Pass Filter

The transfer function of a FIR filter H(z) is given by:

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n}$$

The multiplication by $z^{-1}$ means a delay of one unit of time in the time domain.
The lowpass filter can be described by the equation:

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \quad \text{where } h(k) \text{ are the filter coefficents and } x(n-k) \text{ the sampled input values}$$

Coming from the global definition of a low pass filter realised via a convolver we can say:

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) = h(k) * x(n) \text{ where } * \text{ means convolution}$$

$$y(n) = h(0)x(n) + h(1)x(n-1) + h(2)x(n-2) + \cdots + h(N-1)n(n-N+1)$$

The output is just a linear weighted sum of present and past inputs. So the FIR filter is called a "Running average filter".

The filter function can be described by the following block diagram, where $z^{-1}$ represents the unit delay:

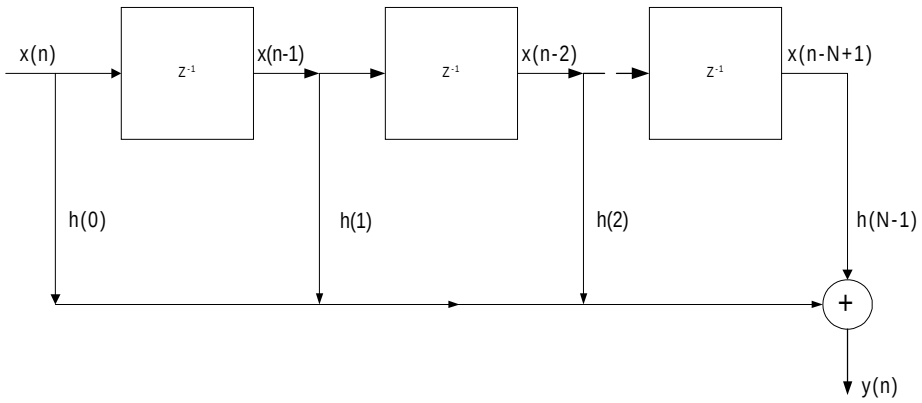



**figure 4**

The input x(n) is multiplied by the coefficient h(n). Then x(n) is delayed by one step and multiplied with the next coefficient h(n+1). After that x(n) is delayed again and multiplied with the next filter coefficient h(n+2). All these single multiplication are added together and one run of an x(n) value through all the filter coefficients results in one output y(n).
So it takes at least N steps, till the filter correctly works. It's the time required for the 1st value to arrive at the filter-output.
This filter method is realised by means of a convolver, which stores and shifts the x-values and multiplies them by the correct filter coefficients.
Figure 4 shows the standard block diagram of a transversal filter.

The coefficients of the filter were calculated using the Fourier design method
The requirements were:
$f_{cutoff}$=1kHz,, $f_{sample}$=50kHz, no given transition band, no given stop band attenuation

Hence a rectangular window function was used:

Corner frequency of ideal filter :

$$\Delta\omega_c = \frac{1000}{50000}2\pi = 0.04\pi$$

thus the ideal frequency response :

$$H_D(\omega) = \begin{Bmatrix} e^{-j\alpha\omega}, |\omega| \le 0.04\pi \\ 0, 0.04\pi < |\omega| \le \pi \end{Bmatrix} \quad \text{where } \alpha = \frac{N-1}{2}$$

IFT of the frequency response :

$$h_D(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\omega)e^{j\omega n}d\omega$$

$$h_D(n) = \frac{1}{2\pi}\int_{0}^{0.04\pi} e^{-j\alpha\omega}e^{j\omega n}d\omega + \frac{1}{2\pi}\int_{0}^{0.04\pi} 0\times e^{j\omega n}d\omega$$

$$h_D(n) = \frac{1}{2\pi}\int_{0}^{0.04\pi} e^{-j\alpha\omega}e^{j\omega n}d\omega = \frac{1}{2\pi}\int_{0}^{0.04\pi} e^{j\omega(n-\alpha)}d\omega$$

$$h_D(n) = \frac{1}{2\pi}\left[\frac{e^{j\omega(n-\alpha)}}{j(n-\alpha)}\right]_0^{0.04\pi} = \frac{1}{2\pi j(n-\alpha)}\left(e^{j0.04\pi(n-\alpha)} - 1\right)$$

$$h_D(n) = \frac{1}{\pi(n-\alpha)}\sin[0.04\pi(n-\alpha)]$$ from where the coefficients can be directly calculated since no window function is used

For the middle filter-coefficient L'Hospital is used to calculate it (0/0)!:

$$h(n) = \frac{\sin[A\pi(n-\alpha)]}{\pi(n-\alpha)} = \frac{0}{0} \quad \text{when } n = \alpha \quad \rightarrow \text{differntiate Numerator and Denominator seperatly}$$

$$h(n) = \frac{A\pi[\cos(nA\pi - A\pi\alpha)]}{\pi}$$

$$h(n) = A = 0.04$$

## 5. Realisation using C

### a) The modulation signal

In usual amplitude modulation signal, where speech or equivalent signals are modulated, the modulation signal is fed to the DSP via an analogue to digital converter.
In the lab-simulation the modulation signal was to generate using a sinus function with a frequency of 1kHz.
The following line shows the realisation using C:

y[0]=sin(n*wn1), where ωn1 is the normalised signal frequency of 1kHz

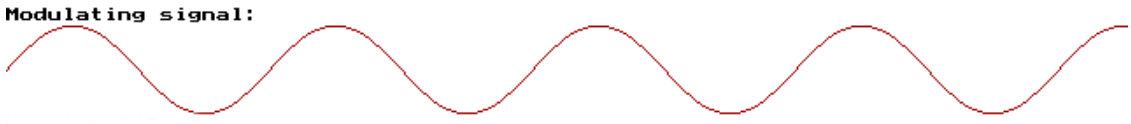



**figure 4 - Modulating signal**

Figure 4 shows the modulating signal output of the program.

### b) The digital Oscillator

The oscillator was designed based on the following difference equation derived via the backwards difference approximation.
Only the coefficients were to determine new, in order to full-fill the requirements, which said the oscillator frequency to be 10kHz at a sampling frequency of 50kHz.

$$y(n) = c_1 y(n-1) - c_2 y(n-2) + x(n)$$

$$\text{where } c_1 = 2e^{-dT}\cos(\omega T) \text{ and } c_2 = e^{-2dT}$$

$$T = \frac{1}{f_{sample}}, \quad d = \text{damping}(=0), \quad \omega = 2\pi f_{Oscillate}$$

$$\text{Hence}: c_1 = 2\times 1\times \cos\left(\frac{2\times\pi\times 10\text{kHz}}{50\text{kHz}}\right) = 0.6180339$$

$$c_2 = 1$$

The resulting code is the following:

```
x[0]=0.6180339*x[1]-x[2]+i;
x[2]=x[1];   // shifting oscillator array
x[1]=x[0];
```

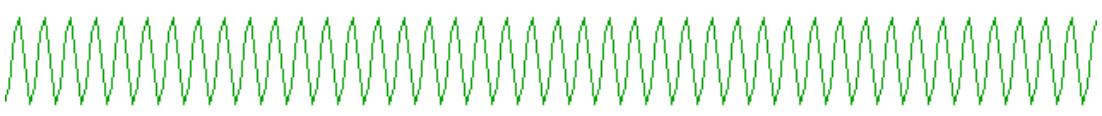



**figure 5 - Digital carrier signal**

## c) Modulation and Demodulation

The modulation and the demodulation is based on simple multiplication. Hence the code for it was very simple to implement:

```
// Modulating
mod=(ampmod+(2-ampmod*1.5)*signal)*carrier;
// Demodulating
demod=-mod*carrier;
```

The variable ampmod is used for changing between balanced and standard amplitude modulation. It's value is previous set 0 for balanced modulation and 1 for standard modulation. The factor 1 and 0.5 (=2-1.5) are chosen to get appropriate modulating results, because the ratio of the carrier and the modulation signal should not be to high (m⇒1), or to low (m⇒0), to get best visible results for Standard amplitude modulation.
The values for balanced modulation are A=2 and m=1.

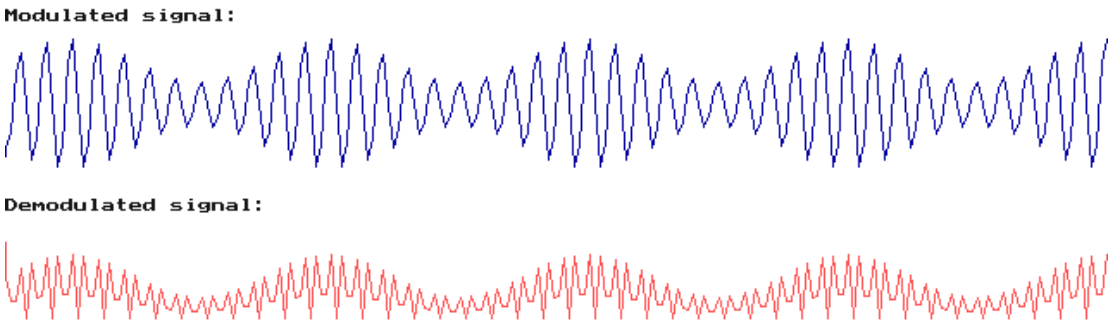



**figure 6 - Modulated and demodulated signal**

**figure 7 - Balance Amplitude Modulated and demodulated signal**

Figure 5 and figure 6 show the modulated and demodulated but not low pass filtered outputs of the software.

## d) Low Pass Filtering

The low pass filter was realised by means of a convolver. The order of the convolver was increased, until a interference free low pass signal available was. The 51 coefficients of the convolver are calculated with every new run of the modulator modulation section of the program.
The filter coefficients are calculated by the following lines using the algorithm from 4c) Low Pass Filter.
The resulting program code is as follows:

```
// Calculating Convolver Filter coefficients
    for(n=0; n<=25 ; n++)
     {
     h[n]=sin((0.04)*pi*(n-25))/(pi*(n-25));
     h[(50)-n]=h[n];
     // Remove comments for printing filter coefficients
     // printf("%d=>%f | %d=>%f  \n",n,h[n],(50-n),h[50-n]);
     }
     h[25]=0.04;   // Set mid filter coefficient (from
L'Hospital)
```

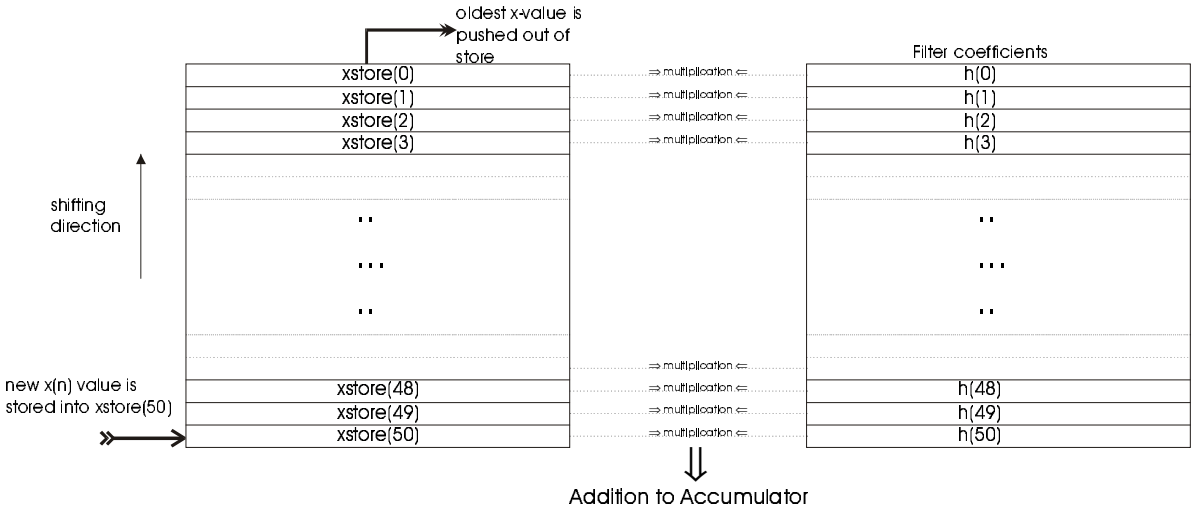



**figure 8 - Working model of a convolver**

Figure 8 shows how the convolver implemented in the software works. Every new demodulated value is written into the last storage of the convolver. Then all present stores are multiplied with their appropriate coefficient.

The following lines show the program code of the convolver:

```
// Filtering via convolver ------------------------------
filter_out=0;
for(k=0; k<=50; k++)
    {
     filter_out+=xstore[k]*h[k];    // convoling
     xstore[k]=xstore[k+1];        // shifting array
    }
xstore[50]=demod;    /* writing new xstore(n) value */
// END OF CONVOLVER -------------------------------------
```

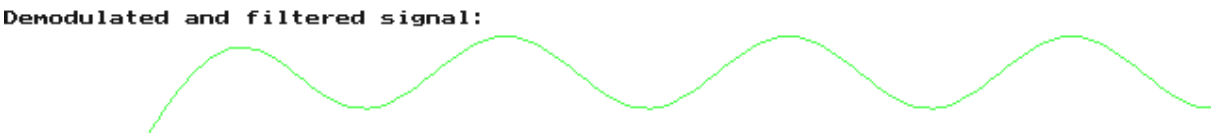



**figure 9 - Low pass filtered signal**

Figure 9 shows the output of the low pass filter realised via convolver. There are no visible interference in the sine signal.

## e) Output Plots

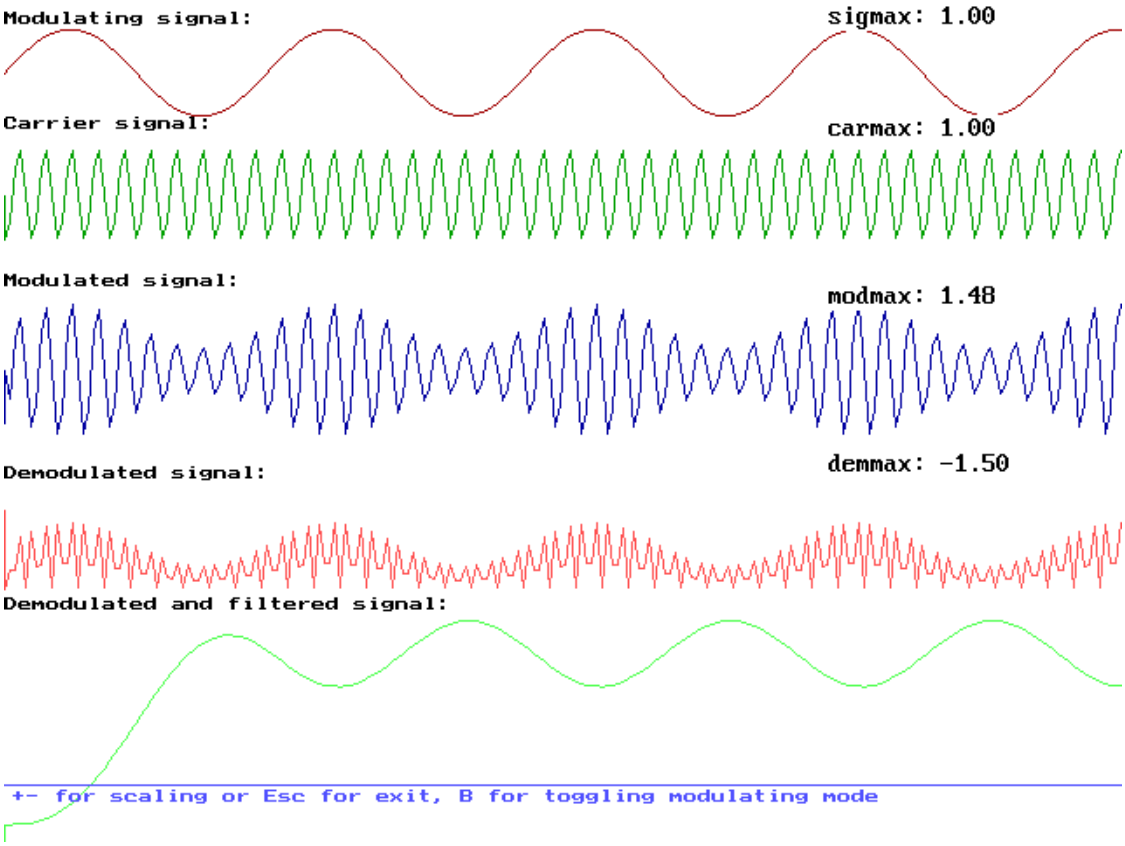



**figure 10 - Output plot for standard amplitude modulation**

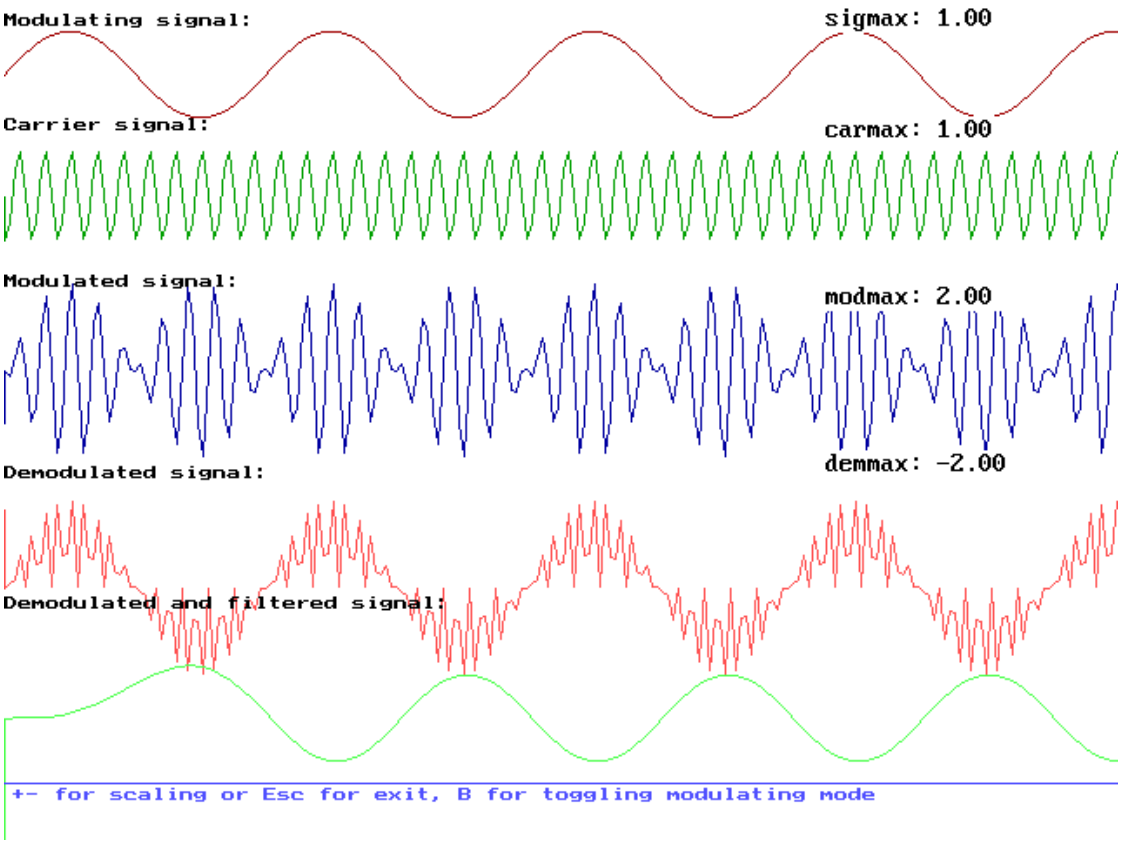



**figure 11 - Balanced amplitude modulation**

Figure 10 and figure 11 show the output plots of the programme in both modulation modes. The min/max function provides the peak levels of the different signals and is realised via simple comparing function.

## 6. Amplitudes

The following peak amplitudes can be read out from the output plots (figure 10 and 11):

| | signal | carrier | modulated | demodulated |
|---|---|---|---|---|
| Standard | **1** | **1** | **1.5** | **1.5** |
| **Balanced** | **1** | **1** | **2** | **2** |

**Standard Amplitude Modulation**

$$V_m = A\sin(\omega_c t)(1 + m\cos(\omega_{mod} t))$$

$$V_m = A\sin(\omega_c t) + \frac{Am}{2}[\sin((\omega_{mod} + \omega_c)t) + \sin((\omega_{mod} - \omega_c)t)]$$

From the equation above it can bee seen, that the maximum peak–amplitude is 1.5, when the carrier is A=1 and the modulation index m=0.5 (values from programme).

**Balanced Amplitude Modulation**

$$V_m = A\sin(\omega_c t)m\cos(\omega_{mod} t)$$
$$V_m = \frac{A \times m}{2}(\sin(\omega_c + \omega_{mod}) + \sin(\omega_c - \omega_{mod}))$$

The maximum peak value of the modulated signal is equal 2, when A=2 and m=1, which are the used values in the programme.

**Demodulation**

$$V_{dem} = \left\{A\sin(\omega_c t) + \left[\frac{Am}{2}(\sin(\omega_c + \omega_{mod})t) + \sin((\omega_c + \omega_{mod})t)\right]\right\} \times \sin(\omega_c t)$$
$$V_{dem} = -\frac{A}{2}(\cos(\omega_c t) - 1) - \frac{Am}{4}\begin{bmatrix}\cos((2\omega_c + \omega_{mod})t) - 2\cos(\omega_{mod} t) + \\ + \cos((2\omega_c - \omega_{mod})t)\end{bmatrix}$$

The demodulated signal is negative signed. It's amplitude is at standard amplitude modulation 1.5 (=0-.5A-0.5Am-2*0.25Am) with A=1 and m=0.5.

The amplitude of the filtered signal was not measured, because of the attenuation of the low pass filter.

## 7. Conclusion

Amplitude modulation is very easy realisable with digital systems, when the systems are capable to calculate at the speed of the desired frequencies.
Analogue amplitude modulation systems can be directly converted from the time to the discrete time domain. The conversion has to be done very carefully, in order to prevent frequency shifting like the backward difference approximation does.
Although amplitude modulation is used since the first days of the 20$^{th}$ century, it is still very popular. The advantages of AM are the easy and cheap way of realisation and the little consumption of bandwidth. The disadvantages are the poor signal to noise ratio and the proneness to amplitude distortions. These disadvantages can't be reduced only with the change from analogue to digital, because the transmission channel is the same analogue one than before.
The effort of the future is to replace today's analogue based amplitude modulation systems with digital systems, like the change from RTTY and Morse-code to digital transmission systems like the amateur's packet-radio.

# 8. Appendix

## a) Complete source code of the modulation software

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>

#define pi 3.14159
#define end 640

//
// Amplitude modulation - DSP simulation using C
// Dirk Becker, BENG 2/3, 9801351
//
// Balanced and Standard Amplitude Modulation
//

float x(int n);
void difference(int ypos, int scale, int ampmod);

int main(void)
{
   /* request autodetection */
   int gdriver = DETECT, gmode, errorcode, scale=3;
   char key;
   int modtype=1;
   /* initialize graphics and local variables */
   initgraph(&gdriver, &gmode, "");
   setbkcolor(BLACK);
   do {
      /*  start  */
      difference(40,scale,modtype);
      setcolor(LIGHTBLUE);
      moveto(0,447);lineto(639,447);
      outtextxy(5,450,"+- for scaling or Esc for exit, B for toggling
modulating mode");
      key=getch();
      if (key=='+') scale++;
      if (key=='-') scale--;
      if ((key=='b')||(key=='B')) modtype=abs(modtype-1);   // toggle
      if (scale<=0) {scale=1;};
      cleardevice();
      clrscr;
      } while (key!=27);

   /* closegraph (clean up) */
   closegraph();
   return 0;
}
```

```
/* ---------- END OF MAIN FUNCTION --------------------------------- */

/* ----------------------------------------------------------------- */
/* ---------- Oscillators, modulator, demodulator and filter ------- */
void difference(int ypos, int scale, int ampmod)
{
    int n, k, y_old, x_old, y_old2, xplot, yplot, yplot2, z_old, zplot,
      f_old=500, fplot; // Vars for line drawing
                    // (storing old points)
    int m_old=290, mplot; // (       -"-        )
    float xvalue, y[1]={0}, x[5]={0}, mod, demod, i=1;
    float fsignal1=1000, fsignal2=10000;     // signal and oscillator
    float fsample=50000;                  // and sample frequencies
    float wn1=2*pi*fsignal1/fsample;         // normalisation of them and
    float wn2=2*cos(2*pi*fsignal2/fsample);  // coefficients of difference
                                  // equation
    float carrier, signal;               // outputs vars
    float filter_out, xstore[52]={0}, h[52]={0}; // filter vars
    setcolor(RED);
    int scaley=25; // scaling in y direction
    float carriermax=0, signalmax=0, modmax=0, demodmax=0 ; // max-min
calcs

    // Calculating Convolver Filter coefficients
    for(n=0; n<=25 ; n++)
     {
      h[n]=sin((0.04)*pi*(n-25))/(pi*(n-25));
      h[(50)-n]=h[n];
      // Remove comments for printing filter coefficients
      // printf("%d=>%f | %d=>%f  \n",n,h[n],(50-n),h[50-n]);
     }
     h[25]=0.04;   // Set mid filter coefficient (from L'Hospital)
    moveto(0,ypos);
    i=1;
    y_old=ypos;y_old2=ypos+70;z_old=ypos+200; // preset storings
                                    // for line function
    x_old=0;
    for(n=0; n<=end; n++)                     // main loop
     {
      // Calculating frequency 1 (signal)
      y[0]=sin(n*wn1);
      signal=y[0];           // for normalisation to 1 (already is)
      if (signal>=signalmax) signalmax=signal;

      // Calculating frequency 2 (carrier)
      x[0]=wn2*x[1]-x[2]+i;
      carrier=x[0];            // for normalisation to 1 ( - "" - )
      if (carrier>=carriermax) carriermax=carrier;

      // Modulating  Ymod=A*f1*(1+m*f2) and change between
      //                 Balanced and Standard modulation
      // Standard Modulation: A=1(ampmod) , m=1.5
      // Balanced Modulation: A=2         , m=1
      //
      mod=(ampmod+(2-ampmod*1.5)*signal)*carrier;
      if (mod>=modmax) modmax=mod;

      // Demodulating
      demod=-mod*carrier;
      if (demod<=demodmax) demodmax=demod;


      // Filtering via convolver --------------------------------
      filter_out=0;
      for(k=0; k<=50; k++)
          {
            filter_out+=xstore[k]*h[k];    // convoling
            xstore[k]=xstore[k+1];        // shifting array
          }
      xstore[50]=demod;    /* writing new xstore(n) value */
      // END OF CONVOLVER ---------------------------------------
```

```
x[2]=x[1];   // shifting oscillator array
x[1]=x[0];
```

```
        // Drawing the waveforms
        // the old points are always stored in the _old vars

        setcolor(RED);      // Raw signal
        xplot=n*scale;
        yplot=-signal*scaley+ypos;
        line(x_old, y_old, xplot, yplot);
        y_old=yplot;

        setcolor(GREEN);    // Carrier frequency
        yplot2=carrier*scaley+ypos+70;
        line(x_old ,y_old2, xplot, yplot2);
        y_old2=yplot2;

        setcolor(BLUE);     // Modulated signal
        zplot=mod*scaley+ypos+170;
        line(x_old ,z_old, xplot, zplot);
        z_old=zplot;

        setcolor(LIGHTRED); // Demodulated (multiplied)
        mplot=demod*scaley+ypos+295;
        line(x_old ,m_old, xplot, mplot);
        m_old=mplot;

        setcolor(LIGHTGREEN); // Filtered via convolver
        if ((filter_out>400)|(filter_out<-400)) filter_out=0;
        fplot=filter_out*(50+100*ampmod)+ypos+(370+60*ampmod);
        line(x_old ,f_old, xplot, fplot);
        f_old=fplot;
        x_old=xplot;


        i=0; /* End of impulse */

      }

      // Adding text to curves
      //
      setcolor(WHITE);
      outtextxy(0,5,"Modulating signal:");
      outtextxy(0,65,"Carrier signal:");
      outtextxy(0,155,"Modulated signal:");
      outtextxy(0,265,"Demodulated signal:");
      outtextxy(0,340,"Demodulated and filtered signal:");

      // Printing max values
      gotoxy(60,1);
      printf("sigmax: %1.2f \n",signalmax);
      gotoxy(60,5);
      printf("carmax: %1.2f \n",carriermax);
      gotoxy(60,11);
      printf("modmax: %1.2f \n",modmax);
      gotoxy(60,17);
      printf("demmax: %1.2f \n",demodmax);




}
```