Knowledge Implementations, Inc.
Communication Systems Engineering/Marketing

Submisssion to IEEE 802.4.L
Atlanta Meeting, May 1990

Description of a set of codes suitable for use with CDMA operations as described in IEEE 802.4L/90-09

Prepared By
Larry Van Der Jagt

Knowledge Implementations, Inc.
32 Conklin Road, Warwick, NY 10990
Voice 914-986-3492, FAX 914-986-6441

At this time it looks unlikely that I will be attending the meeting of IEEE 802.4.L to be held in Atlanta so I have forwarded this information to you for your review. Between meetings some work concerning codes suitable for use in a CDMA or CSK environments was done. Specifically, the Kasami codes discussed briefly in our previous submission were generated and the auto/cross and periodic correlation characteristics of those codes were examined. The attached represents some of the information obtained in that analysis.

What you will find is first the m-sequence of length 64 which was chosen totally at random as the basis for the kasami codes. I actually generated codes from all of the phases of this m-sequence and found that codes from different phases of the generating sequence also have good properties when correlated against each other. The first page however shows the sequence and the sequence correlated in a periodic fashion against the seven sequences which were generated from it. This code is called "sally".

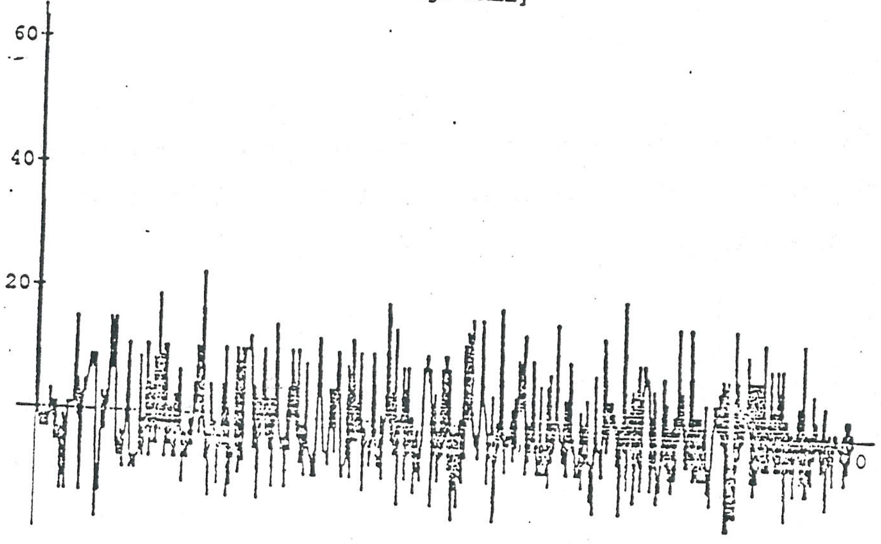
On the next page you will find the seven sequences generated from "sally". The next set of charts illustrates the autocorrelation and cross correlation characteristics of the seven codes listed on the previous page. As you can see they exhibit the predicted the level side lobes with the set of sidelobe levels being {m/2-1, -(m/2+1), -1}. For the next set of diagrams, I generated a test signal with each of the codes in succession and correlated that test sequence against each code in turn. These diagarams illustrate the performance of the code in periodic correlation situations. Finally, there are a series of diagrams showing the individual fourier transforms of each of the individual codes.

Knowledge Implementations, Inc.
Communication Systems Engineering/Marketing

Although not illustrated in the attached diagrams it appears that the codes which are derived from other phases the same m-sequence are also well behaved with respect to the codes listed. This allows the possibility that more than the m/2 codes which are available from a specific generating sequence can be used in a system. In fact if I had to take an educated guess based on the analysis so far I would guess that the number of useful codes approaches or exceeds m, making significant increases in data rate for a given bandwidth expansion factor available at the expense of receiver complexity.

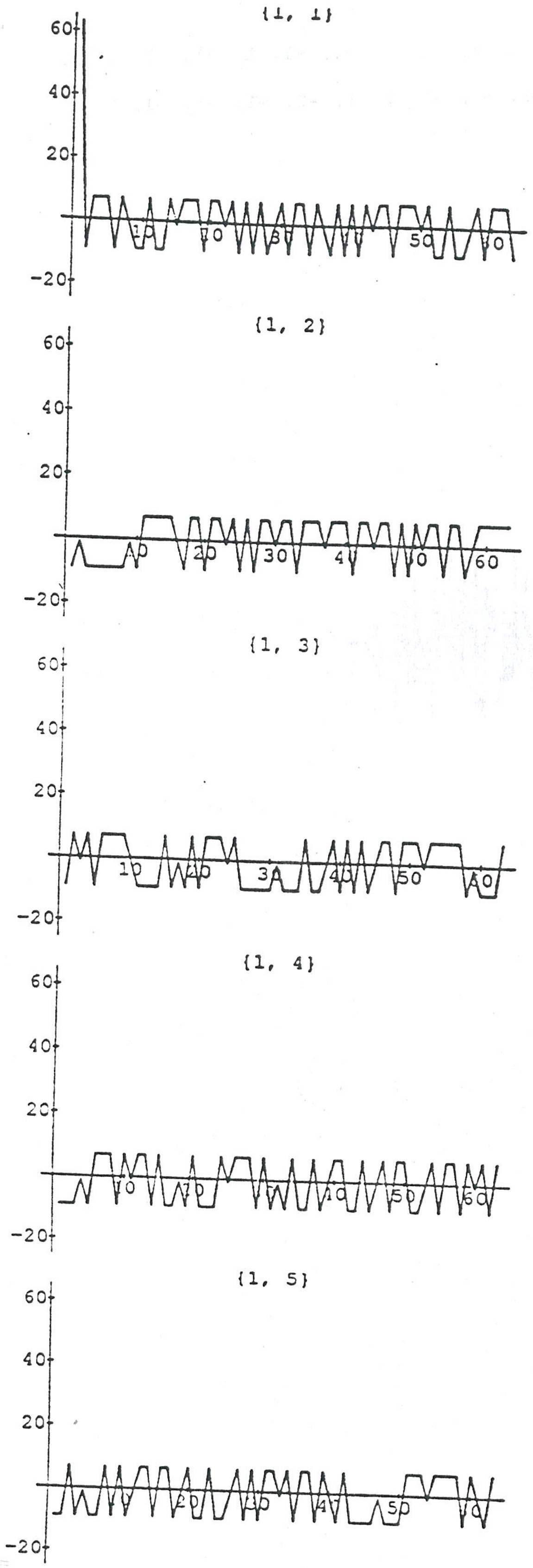
If I were to generate a specific proposal at this time I would propose that we use Code Shift Keying with BPSK modulation rather than deal with the potential objections associated with transmitting multiple codes simultaneously. In a scenario in which the eight codes listed in this document were the codes transmitted, and if we use the entire band at 900 MHz we would have a chip time of about 40 ns, a symbol time of about 2500 ns and a symbol rate of 400 ksymbols/second. With the eight possible codes in a CSK environment three bits per symbol could be transmitted. If the error correcting code was a rate 2/3 code we could transmit 2 information bits per symbol leaving us with an information rate of 800 Kbps in the 900 MHz band. A similar analysis in the 2.5 GHz band yields an information rate of about 2 Mbps.

```
Out[37]=  SALLY
  {-1, 1, -1, 1, -1, 1, 1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, 1, -1, 1, -1, -1, 1, -1, -1, 1, 1,
   1, -1, -1, -1, 1, -1, 1, 1, 1, 1, -1, -1, 1, -1, 1, -1, -1, -1, 1, 1, -1, -1, -1, -1, 1, -1,
   -1, -1, -1, -1, 1, 1, 1, 1, 1, 1}
In[43]:=
  a=crosscorr[Length[test],test,sally];
  ListPlot[a,PlotJoined ->True,PlotRange->All]
```



Kasami code correlations

```
In[29]:=
  kasamitable[[1,1]]
Out[29]=
  {1, -1, -1, 1, -1, -1, 1, -1, -1, 1, 1, -1, -1, -1, 1, -1, 1, 1, -1, 1, -1, 1, -1, 1, 1, 1, -1,
    1, -1, 1, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, -1, 1, -1, -1, -1, -1, -1, -1, -1, -1,
    -1, 1, -1, -1, 1, 1, 1, -1, 1, -1, 1}
In[30]:=
  kasamitable[[1,2]]
Out[30]=
  {-1, -1, 1, -1, 1, -1, 1, 1, -1, -1, -1, 1, -1, -1, -1, -1, -1, -1, 1, 1, -1, -1, -1, -1, -1,
    -1, -1, 1, 1, 1, 1, -1, 1, -1, -1, 1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, -1, -1, 1, -1, 1,
    1, -1, -1, -1, -1, 1, -1, 1, -1, -1, 1}
In[31]:=
  kasamitable[[1,3]]
Out[31]=
  {-1, 1, -1, 1, 1, -1, -1, 1, 1, 1, 1, 1, -1, 1, -1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, -1, -1, -1,
    1, -1, -1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, -1, 1, -1, -1, -1, 1, -1, 1, 1, 1, -1, -1, -1, -1,
    1, -1, -1, 1, -1, -1, -1, -1}
In[32]:=
  kasamitable[[1,4]]
Out[32]=
  {1, -1, 1, 1, 1, 1, -1, -1, -1, -1, 1, 1, 1, 1, 1, -1, -1, 1, 1, -1, 1, 1, -1, -1, 1, -1, 1,
    -1, -1, 1, 1, 1, 1, 1, 1, -1, -1, -1, 1, 1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, -1, 1, -1, -1,
    1, 1, 1, 1, -1, -1, -1, 1, -1}
In[33]:=
  kasamitable[[1,5]]
Out[33]=
  {-1, 1, 1, 1, -1, 1, 1, 1, 1, -1, 1, -1, 1, -1, -1, 1, -1, 1, -1, -1, -1, -1, 1, -1, 1, 1, 1,
    1, 1, -1, 1, 1, -1, 1, -1, 1, 1, -1, 1, -1, -1, 1, 1, -1, 1, -1, -1, 1, -1, 1, 1, 1, -1, 1,
    1, -1, -1, -1, -1, -1, 1, 1, 1}
In[34]:=
  kasamitable[[1,6]]
Out[34]=
  {1, 1, 1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, 1, 1, 1, -1, -1, -1, 1, 1, 1, 1, -1, -1, 1,
    -1, -1, -1, -1, 1, -1, -1, -1, 1, -1, 1, -1, -1, -1, 1, -1, -1, -1, 1, 1, -1, -1, 1, -1, 1,
    1, 1, 1, -1, 1, 1, -1, -1, 1, 1, -1, -1}
In[36]:=
  kasamitable[[1,7]]
Out[36]=
  {1, 1, -1, -1, 1, 1, 1, -1, 1, 1, -1, 1, 1, -1, 1, 1, 1, -1, 1, -1, -1, 1, 1, 1, -1, -1, 1, 1,
    -1, -1, -1, -1, 1, 1, -1, -1, 1, 1, -1, 1, -1, 1, -1, -1, -1, 1, 1, 1, -1, -1, 1, -1, 1, -1,
    1, -1, 1, -1, 1, 1, -1, 1, 1}
```



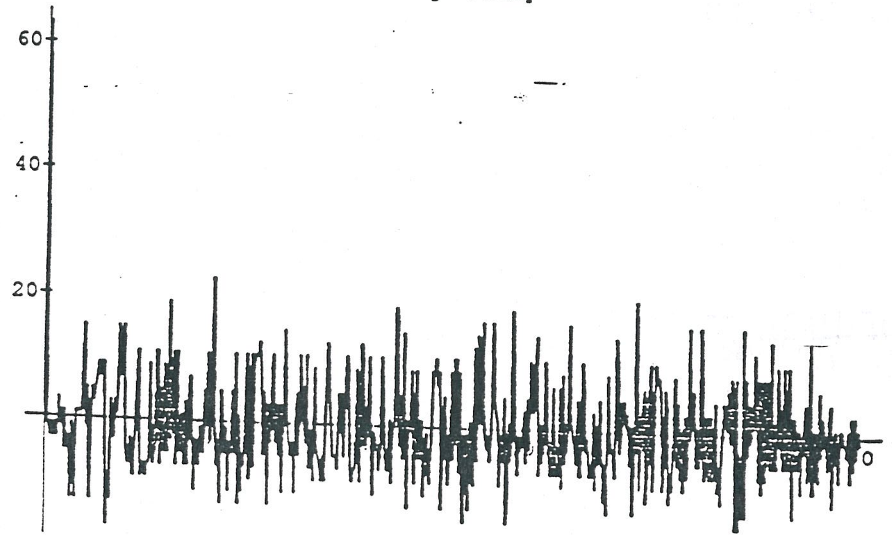
Kasami code correlations

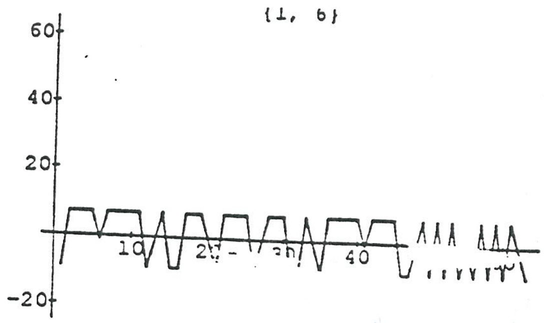
Out[37]= SALLY

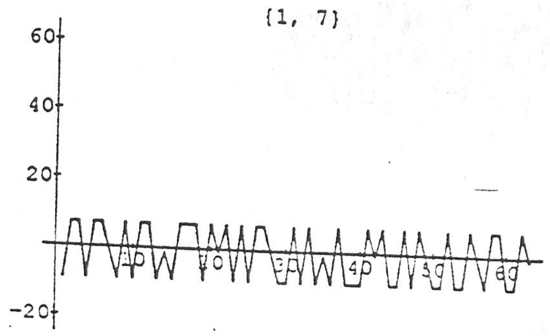
```
{-1, 1, -1, 1, -1, 1, 1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, 1, -1, 1, -1, -1, 1, -1, -1, 1, 1,
  1, -1, -1, -1, 1, -1, 1, 1, 1, 1, -1, -1, 1, -1, 1, -1, -1, -1, 1, 1, -1, -1, -1, -1, 1, -1,
  -1, -1, -1, -1, 1, 1, 1, 1, 1, 1}
```

In[43]:=

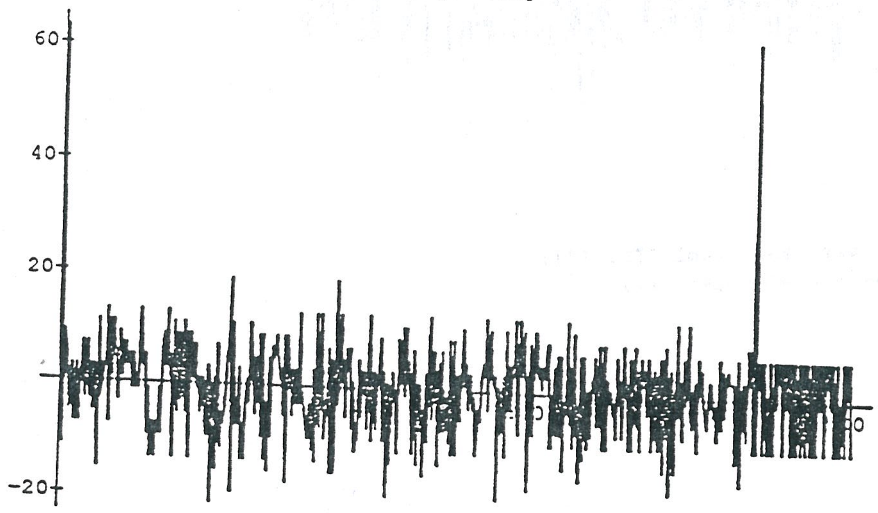
```
a=crosscorr[Length[test],test,sally];
ListPlot[a,PlotJoined ->True,PlotRange->All]
```



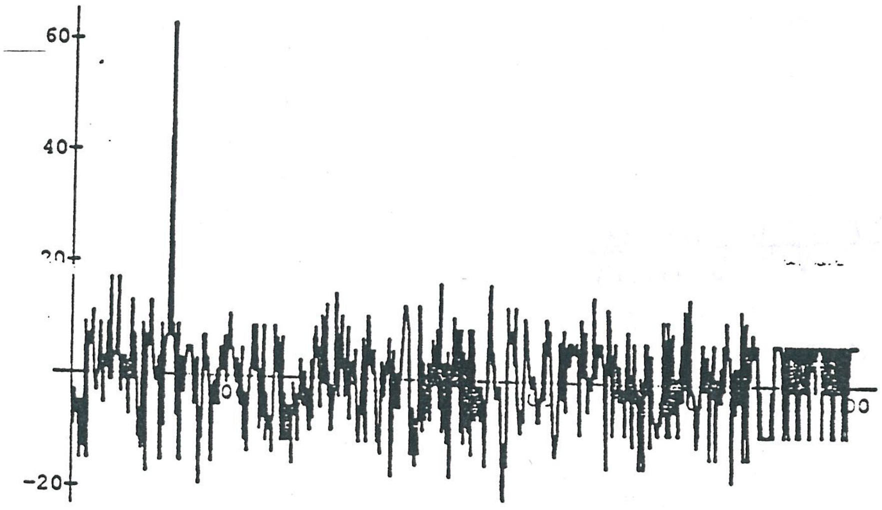




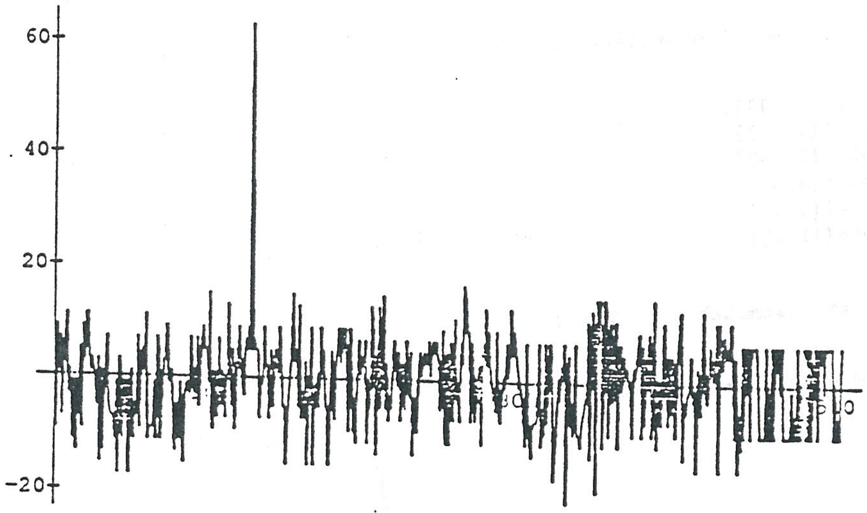
In[6]:=
```
test=Join[kasamitable[[1,1]],kasamitable[[1,2]]];
```
In[7]:=
```
test=Join[test,kasamitable[[1,3]]];
test=Join[test,kasamitable[[1,4]]];
test=Join[test,kasamitable[[1,5]]];
test=Join[test,kasamitable[[1,6]]];
test=Join[test,kasamitable[[1,7]]];
test=Join[test,kasamitable[[1,1]]];
```
In[9]:=
```
crosscorr[Length[test],test,kasamitable[[1,1]]];
```
In[10]:=
```
ListPlot[%,PlotJoined ->True,PlotRange->All]
```



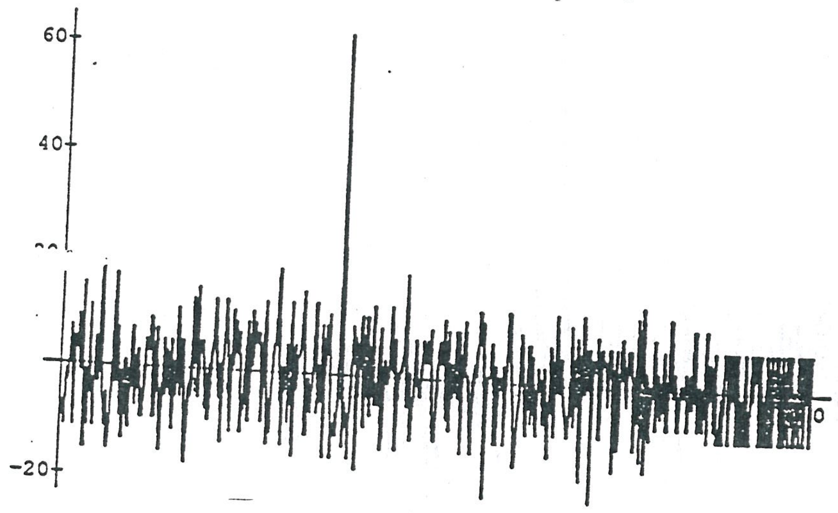
In[11]:=
```
crosscorr[Length[test],test,kasamitable[[1,2]]];
```
In[12]:=
```
ListPlot[%,PlotJoined ->True,PlotRange->All]
```



```
Out[12]=
  -Graphics-
In[15]:=
  a=crosscorr[Length[test],test,kasamitable[[1,3]]];
  ListPlot[a,PlotJoined ->True,PlotRange->All]
```

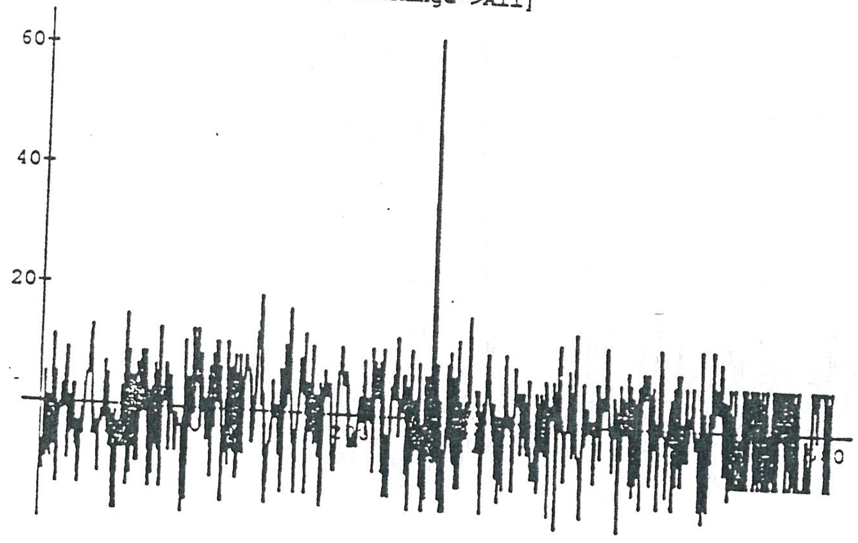


```
Out[15]=
  -Graphics-
In[16]:=
  a=crosscorr[Length[test],test,kasamitable[[1,4]]];
  ListPlot[a,PlotJoined ->True,PlotRange->All]
```



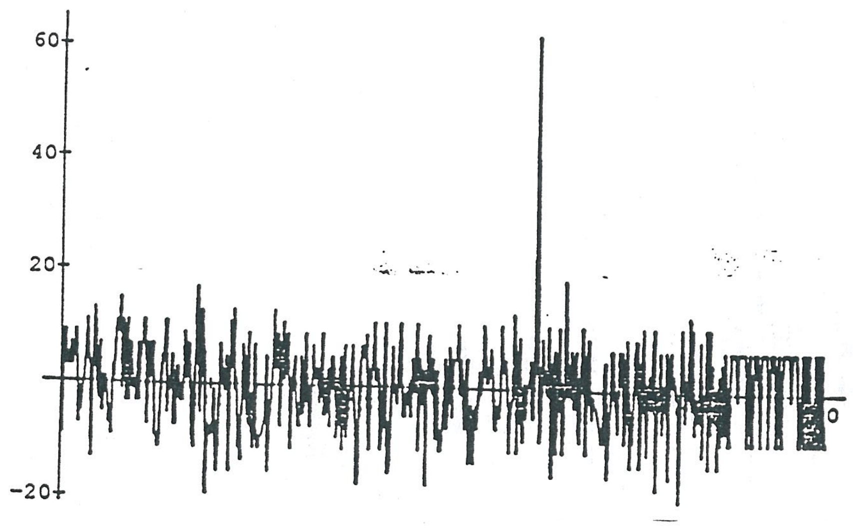
```
Out[16]=
  -Graphics-

In[17]:=
  a=crosscorr[Length[test],test,kasamitable[[1,5]]];
  ListPlot[a,PlotJoined ->True,PlotRange->All]
```



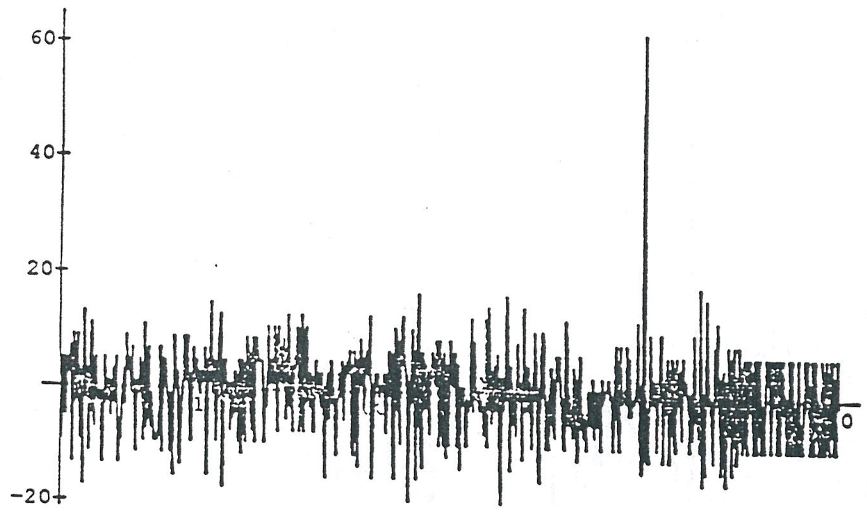
```
Out[17]=
  -Graphics-

In[18]:=
  a=crosscorr[Length[test],test,kasamitable[[1,6]]];
  ListPlot[a,PlotJoined ->True,PlotRange->All]
```



```
Out[18]=
  -Graphics-

In[19]:=
  a=crosscorr[Length[test],test,kasamitable[[1,7]]];
  ListPlot[a,PlotJoined ->True,PlotRange->All]
```



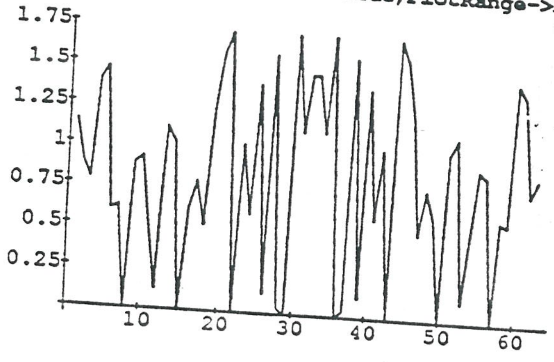
```
Out[19]=
  -Graphics-

In[26]:=
  test=Join[kasamitable[[1,1]],RotateRight[kasamitable[[1,2]],1]];
In[27]:=
  test=Join[test,RotateRight[kasamitable[[1,2]],2]];
  test=Join[test,RotateRight[kasamitable[[1,2]],3]];
  test=Join[test,RotateRight[kasamitable[[1,2]],4]];
  test=Join[test,RotateRight[kasamitable[[1,2]],5]];
  test=Join[test,RotateRight[kasamitable[[1,2]],6]];
  test=Join[test,RotateRight[kasamitable[[1,2]],7]];
In[28]:=
  a=crosscorr[Length[test],test,kasamitable[[1,1]]];
  ListPlot[a,PlotJoined ->True,PlotRange->All]
```

```
a=Fourier[kasamitable[[1,1]]];
ListPlot[Abs[a],PlotJoined ->True,PlotRange->All]
```

In[47]:=

```
ListPlot[Abs[a],PlotJoined ->True,PlotRange->All]
```

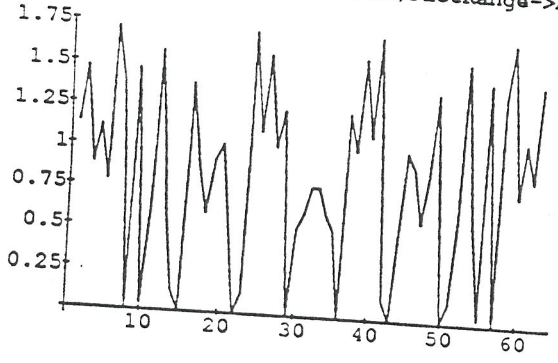


Out[47]=

```
-Graphics-
```

In[48]:=

```
a=Fourier[kasamitable[[1,2]]];
ListPlot[Abs[a],PlotJoined ->True,PlotRange->All]
```

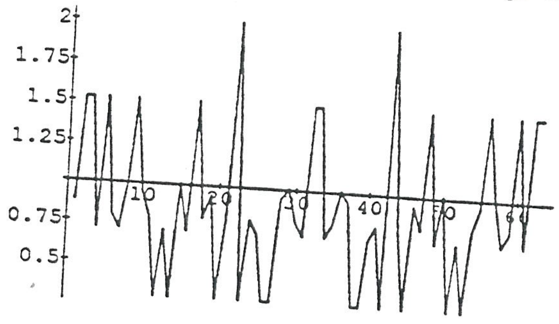


Out[48]=

```
-Graphics-
```

In[49]:=

```
a=Fourier[kasamitable[[1,3]]];
ListPlot[Abs[a],PlotJoined ->True,PlotRange->All]
```

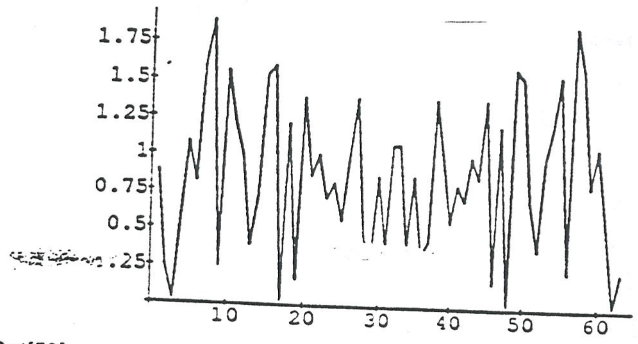


Out[49]=

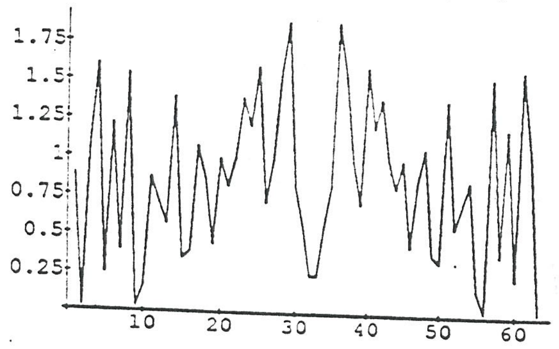
```
-Graphics-
```

In[50]:=

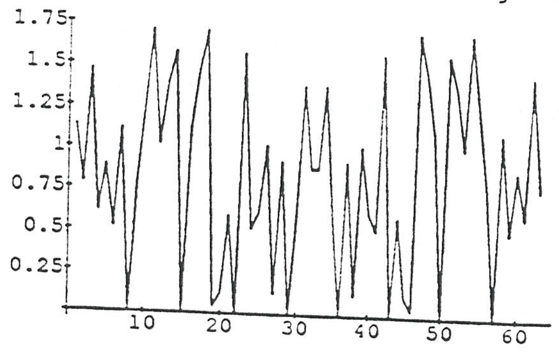
```
a=Fourier[kasamitable[[1,4]]];
ListPlot[Abs[a],PlotJoined ->True,PlotRange->All]
```



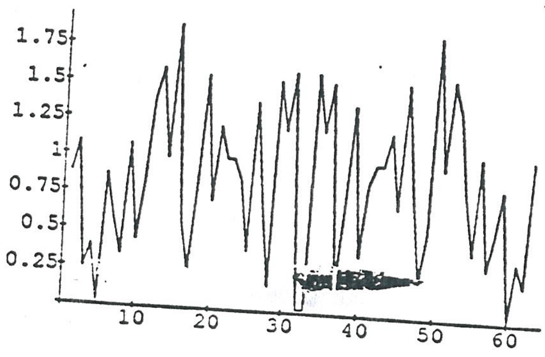
Out[50]=
```
-Graphics-
```
In[51]:=
```
a=Fourier[kasamitable[[1,5]]];
ListPlot[Abs[a],PlotJoined ->True,PlotRange->All]
```



Out[51]=
```
-Graphics-
```
In[52]:=
```
a=Fourier[kasamitable[[1,6]]];
ListPlot[Abs[a],PlotJoined ->True,PlotRange->All]
```



Out[52]=
```
-Graphics-
```
In[53]:=
```
a=Fourier[kasamitable[[1,7]]];
ListPlot[Abs[a],PlotJoined ->True,PlotRange->All]
```



```
Out[53]=
  -Graphics-
```