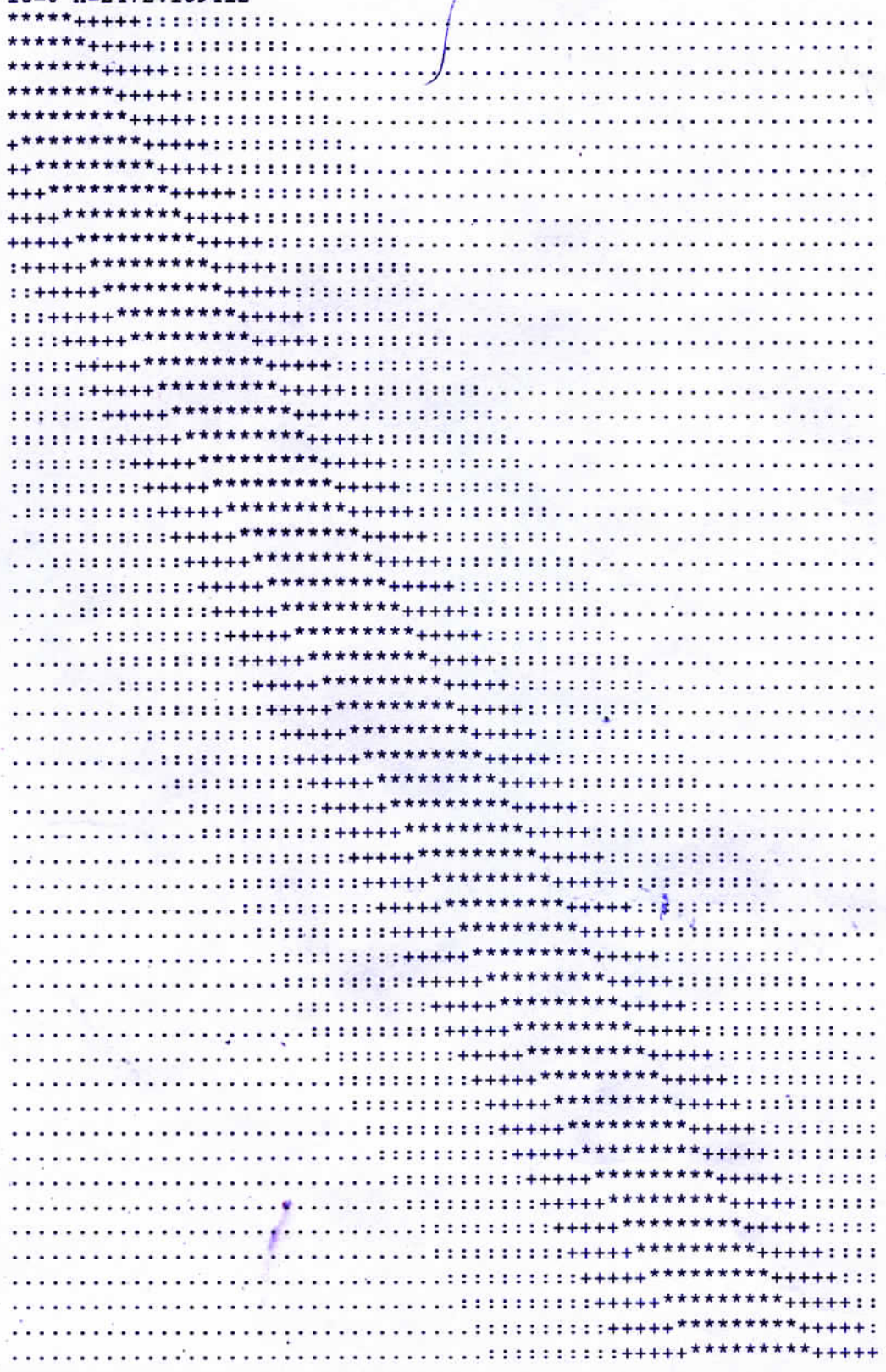
11-4-98

MR0060

Here is my latest 64x64 algorithm, that uses a norm that is a combination of

- distance from (0,0)
- distance from main diagonal
- vertical first differences
- horizontal first differences

I start with a diagonal matrix. Note the norm is about 2472.

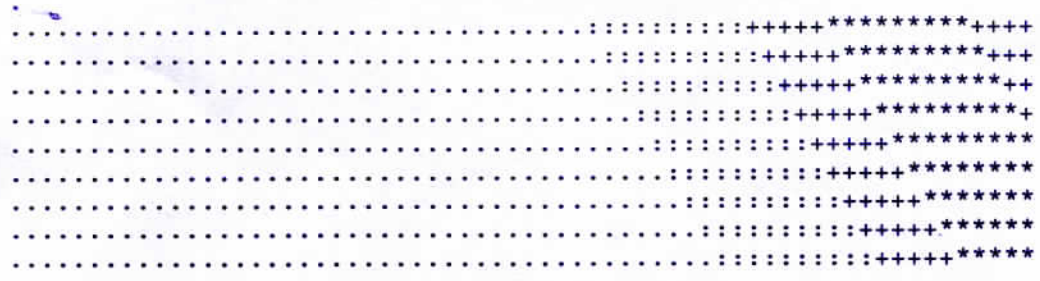


problem:

re-order a set of N elements, e_i so that their correlation matrix, c_{ij} is as diagonal as possible.

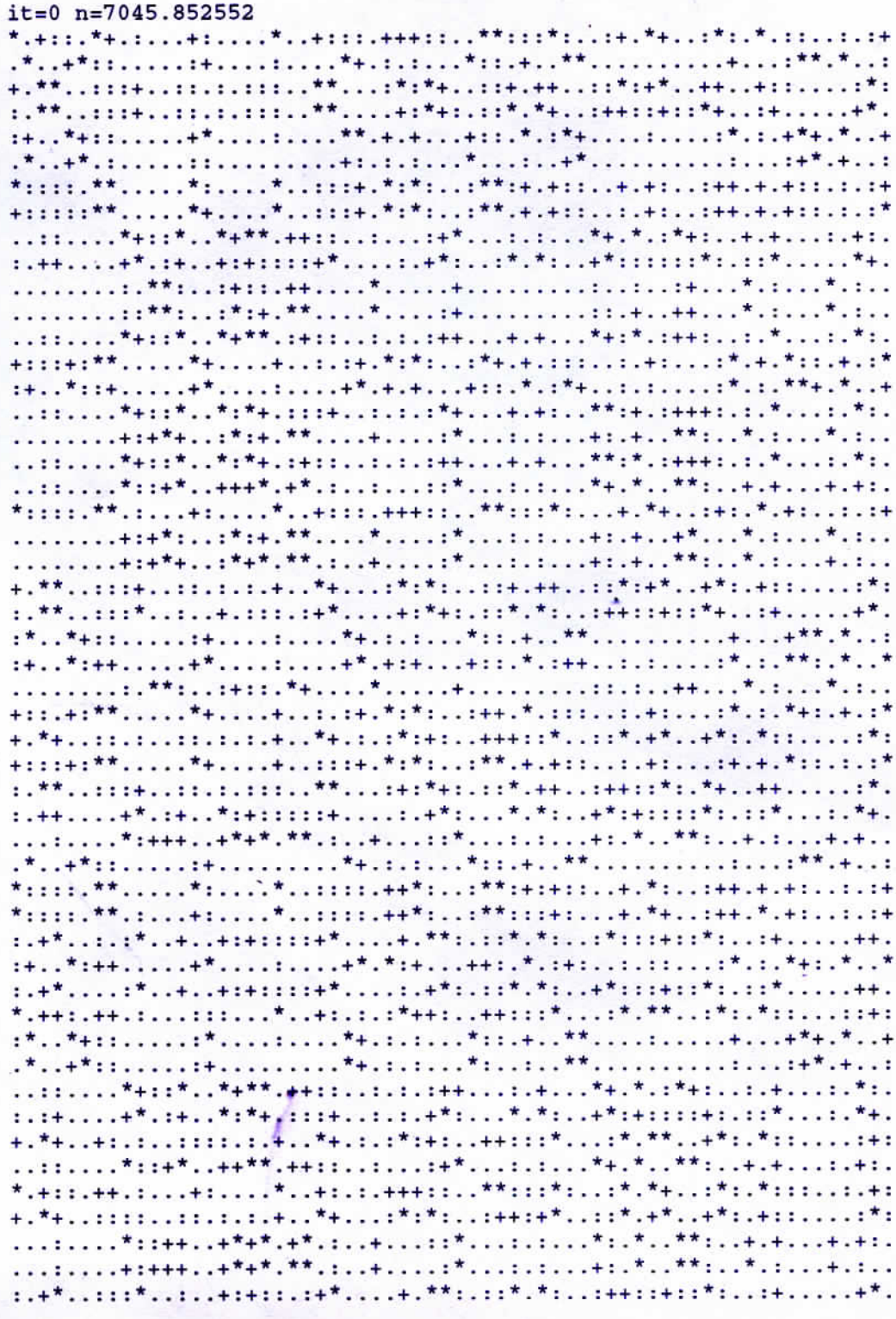
note. reordering requires a "distance from UL corner", condition too, to stabalize process.

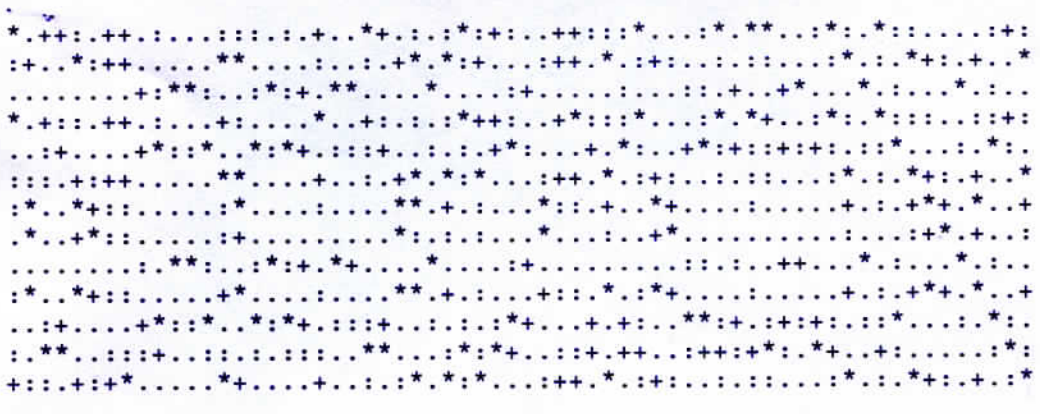
method: monte carlo search



Now I randomize it, with an algorithm that guarantees that each of the 64 factorial combinations have the same probability. Note the norm is about 7045, or about three times bigger.

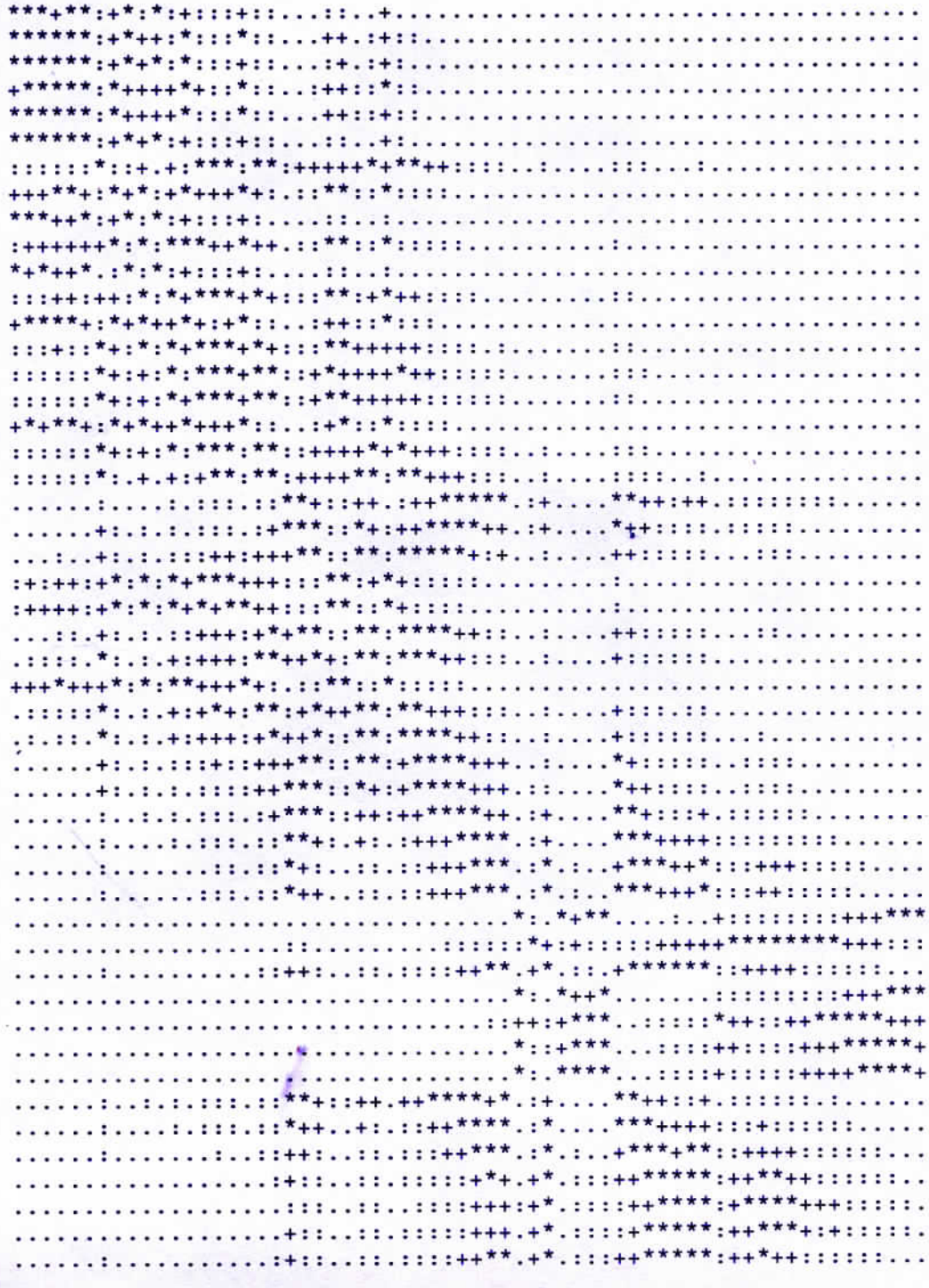
it=0 n=7045.852552

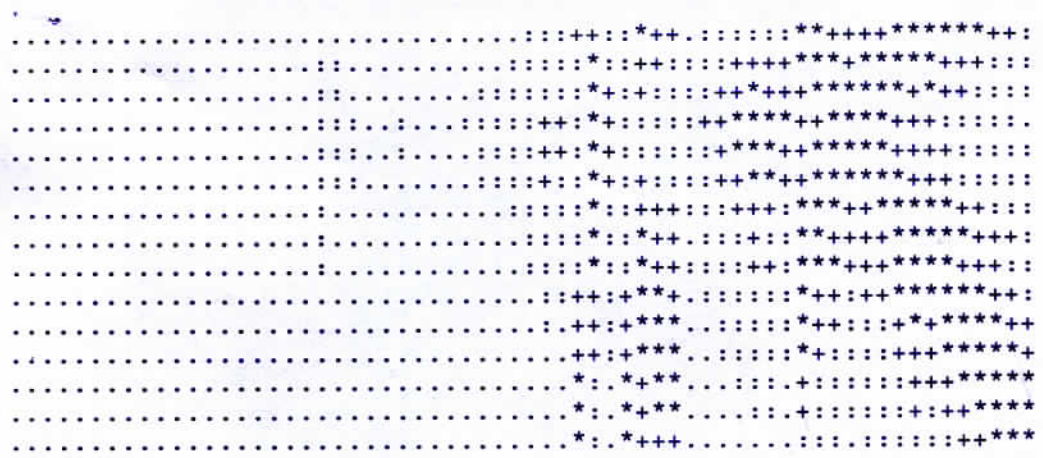




Now observe the results after a thousand iterations ...

it=1000 n=3588.530328

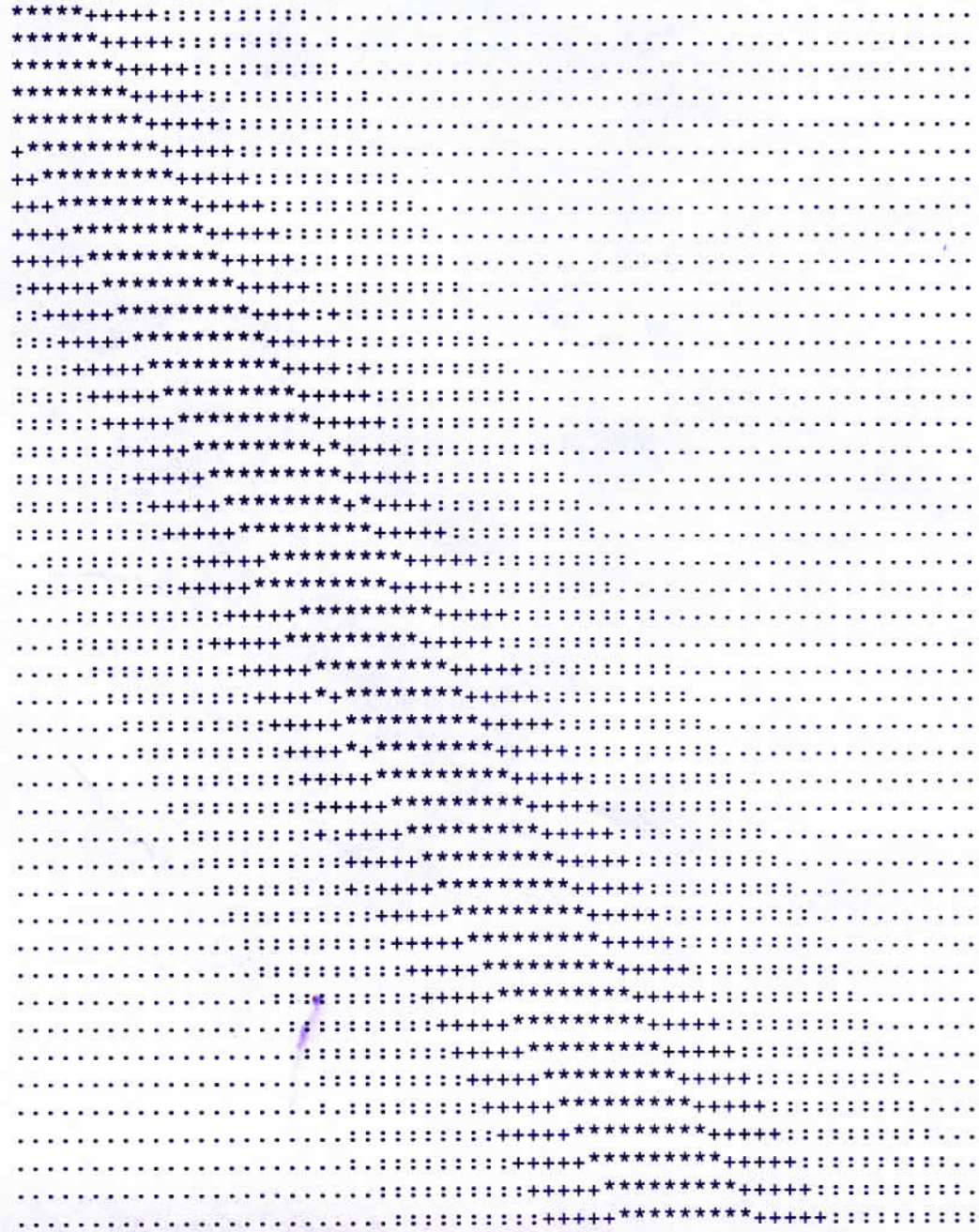


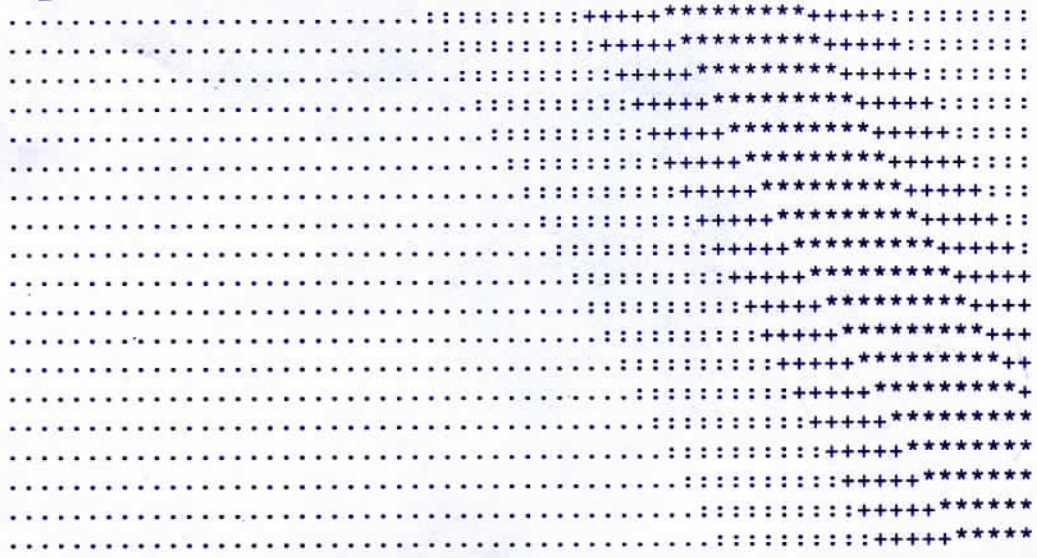


And eleven thousand iterations ...

And eleven thousand iterations ...

it=11000 n=2465.566103





Pretty good, eh! The result is pretty much what we wanted. Indeed, the norm is smaller than the original's (which had some random noise superimosed on the band structure, so it did not necessarily minimize the norm).

Here's the code:

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

#define TRUE (1)
#define FALSE (0)

long random();
double white();
long r64();
double norm();

main(argc,argv)
int argc;
char *argv[];
{
        double c1[64][64], c2[64][64];
        long i, j, k, it, a, b, c, d;
        long e1[64], e2[64];
        double n1, n2;

        srandom(time(0));

/* create the band matrix, with some additive noise */

        for(i=0; i<64; i++ ) for(j=0; j<i; j++ ) {
                k =  i-j;
                if( k<5 ) c1[i][j]=0.8+white(0.1);
                else if( k<10 ) c1[i][j]=0.5+white(0.1);
                else if ( k<20 ) c1[i][j]=0.3+white(0.1);
                else c1[i][j]=white(0.1);
                c1[j][i]=c1[i][j];
                }
        for(i=0; i<64; i++ ) {
                c1[i][i]=1.0;
                e1[i]=i;
                }
        n1=norm( c1 );
        printc( 0, c1, n1, e1 );
```

```
/* randomize it */
        newe(e1);
        for(i=0; i<64; i++ ) for(j=0; j<64; j++ ) {
                c2[i][j]= c1[e1[i]][e1[j]];
                }
        for(i=0; i<64; i++ ) for(j=0; j<64; j++ ) {
                c1[i][j]= c2[i][j];
                }
        n1=norm( c1 );

/* iterate a large number of times */

        for( it=0; it<1000000; it++ ) {

                if(!(it%1000)) printc( it, c1, n1, e1 );

/* randomly choose between one of three type of exchange operations */

                d = r64()/16;
/* exchange a randomly chosen pair of elements */
                if(  d==0 ) {
                        a = r64();
                        do { b=r64(); } while (a==b);
                        for( i=0; i<64; i++ ) e2[i]=e1[i];
                        j=e2[a]; e2[a]=e2[b]; e2[b]=j;
                        }
/* exchange two pairs of adjoining of elements */
                else if ( d==1 ) {
                        a = r64();
                        do { b=r64(); } while (a==b);
                        for( i=0; i<64; i++ ) e2[i]=e1[i];
                        j=e2[a]; e2[a]=e2[b]; e2[b]=j;
                        if( a<(64-1) ) a++; else a--;
                        if( b<(64-1) ) b++; else b--;
                        j=e2[a]; e2[a]=e2[b]; e2[b]=j;
                        }
/* exchange two triplets of adjoining  elements */
                else if ( d==2 ) {
                        a = r64();
                        do { b=r64(); } while (a==b);
                        for( i=0; i<64; i++ ) e2[i]=e1[i];
                        j=e2[a]; e2[a]=e2[b]; e2[b]=j;
                        if( a<(64-2) ) a++; else a--;
                        if( b<(64-2) ) b++; else b--;
                        j=e2[a]; e2[a]=e2[b]; e2[b]=j;
                        if( a<(64-1) ) a++; else a--;
                        if( b<(64-1) ) b++; else b--;
                        j=e2[a]; e2[a]=e2[b]; e2[b]=j;
                        }
/* exchange an random number of randomly positioned pairs of elements */
                else {
                        c=r64()/8;
                        for( i=0; i<64; i++ ) e2[i]=e1[i];
                        for( k=0; k<c; k++ ) {
                                a = r64();
                                do { b=r64(); } while (a==b);
                                j=e2[a]; e2[a]=e2[b]; e2[b]=j;
                                }
                        }
                checke(e2);
                for(i=0; i<64; i++ ) for(j=0; j<64; j++ ) {
                        c2[i][j]= c1[e2[i]][e2[j]];
                        }
                if( (n2=norm(c2)) < n1 ) {
                        for(i=0; i<64; i++ ) {
                                e1[i]=e2[i];
```

```
                                }
                        n1=n2;
                        }
                }

        exit(0);
        }

double norm( c )
double c[64][64];
{
        double n;
        long i, j;

        n=0.0;
        for( i=0; i<64; i++ ) for( j=0; j<64; j++ ) {
                n += 0.1*c[i][i]*((double)i+(double)j)/(2.0*64.0);
                n += 10.0*c[i][j]*fabs((double)i-(double)j)/64.0;
                }
        for( i=0; i<64; i++ ) for( j=0; j<(64-1); j++ ) {
                n += fabs( c[i][j] - c[i][j+1] );
                n += fabs( c[j][i] - c[j+1][i] );
                }
        return(n);
        }

printc( it, c, n, e )
long it;
double c[64][64];
double n;
long e[64];
{
        char s[20];
        int i, j, k;

        strcat(s,".:+*456789ABCDEF");
        printf("it=%d n=%f\n", it, n );
        for( i=0; i<64; i++ ) {
                for(j=0; j<64; j++ ) {
                        k = (long)(c[e[i]][e[j]]*3.999999);
                        if( k<0 ) k=0; else if (k>4) k=3;
                        printf("%c",s[k]);
                        }
                printf("\n");
                }
        printf("\n");
        fflush(stdout);

        }


double white(a)
double a;
/* returns a random number between 0 amd a */
{
        int mask= 000000177777;
        double b;

        b = (double)(random()&mask)/65536.0;

        return( a*b );
}

long r64()
/* returns a random number between 0 amd 63 */
{
```

```
        long mask= 000000000077;
        return( random() & mask );
}

newe( e )
long e[64];
{
        int i, j, k, f;

        for( i=0; i<64; i++ ) {
                f=TRUE;
                while( f ) {
                        j = r64();
                        f=FALSE;
                        for( k=0; k<i; k++ ) {
                                if( j==e[k] ) f=TRUE;
                                }
                        }
                e[i]=j;
                }
        checke(e);
        }

checke(e)
long e[64];
{
        long i, j, f;

        for( i=0; i<64; i++ ) {
                f=FALSE;
                for( j=0; j<64; j++ ) {
                        if( e[j] == i ) {
                                f=TRUE;
                                break;
                                }
                        }
                if( !f ) { printf("checke failed\n"); exit(-1); };
                }
        }
```