

```

/*****
/*          IRAsimu.cpp          */
/*          Hui Jin             */
/*****
/* simulation file of Irregular Repeat Accumulative code.
   The code structure is actually more like an LDGM code
   + an accumulative inner code.

   First verstion; 3/20/2000
   Copyright @ Hui Jin, Caltech 2000

   Modification: initial verison 3/20/2000
                   We should use the same .prm code as in GetInter.cpp,
which has degree decreasing.
                   second versioin Use checknode length instead, so
the edge number = k * check_len;
                   And now we don't need the restriction in first
version. The interleaver has been changed.
*/

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <math.h>

#include "IRA.h"
#include "vector.h"
#include "ldpc.h"
#include "paramfio.h"
#include "capacity.h"
#include "datatype.hh"
#include "random.hh"
#include "node.h"

void printusage()
{
    cout << "Usage" << endl;
    cout << "IRAsimu paramfile 0 snr infonodelength
Interfile" << endl;
    cout << "IRAsimu paramfile 1 sigma infonodelength
Interfile" << endl;
    cout << "IRAsimu paramfile 2 p infonodelength
Interfile" << endl;
    exit(1);
}

void getLRdegree(degree_sequence& L, degree_sequence &R, char *
filename)

```

```

{
    ivector deg;
    vector fe;
    paramfio P;
    P.set_filename(filename);
    P.read("variable degree", deg);
    P.read("variable fraction", fe);
    L.init(deg.getsize());
    L.d = deg;
    L.fe = fe;
    P.read("check degree", deg);
    P.read("check fraction", fe);
    R.init(deg.getsize());
    R.d = deg;
    R.fe = fe;
    L.edge_to_node();
    R.edge_to_node();
}

void rate_info(const double rate)
{
    cout << "Rate=" << rate << endl;
    capacity C;
    double capacity = C.sn_binary_AWGN(rate);
    cout << "capacity = " << capacity << ", i.e. " \
        << 10*log10(1/(2*rate*capacity*capacity)) << "dB" << endl;
}

void getInterleaver(int * _interleaver, char *filename, int Edge_num)
{
    ifstream inFile(filename, ios::in);

    if(!inFile) {
        cerr<< "cannot open file" << endl;
        exit(-1);
    }

    for(int i=0; i<Edge_num; i++)
        inFile >> _interleaver[i];
    inFile.close();
}

int main(int argc, char* argv[])
{

```

```

if (argc !=6)
{
    printusage();
}

degree_sequence LV;
degree_sequence LC;
int argcount = 1;

getLRdegree(LV, LC, argv[argcount++]);

double rate = LV.av_inv_degree_edge();
rate/= LC.av_inv_degree_edge()+ LV.av_inv_degree_edge();

rate_info(rate);

int mode=atoi(argv[argcount++]);

double sigma;
double snr;
if(mode==0)
    {snr = atof(argv[argcount++]);
    sigma= sqrt(pow(10, -snr/10.0)/(2*rate));
    }
else
    {
    sigma=atof(argv[argcount++]);
    snr=10*log10(1/(2*rate*sigma*sigma));
    }

int _info_len=atoi(argv[argcount++]);
int LEdge_num= 0;
int nv[LV.n];
{
    double s=0;
    for(int i=0; i< LV.n-1; i++)
    {
        nv[i]= int(floor((s+LV.fn[i])*_info_len) -
floor(s*_info_len));
        s+=LV.fn[i];
        LEdge_num += nv[i] * LV.d[i];
    }
    nv[LV.n-1]= _info_len -(int)floor(s*_info_len);
    LEdge_num += nv[LV.n-1] * LV.d[LV.n-1];

    for(int i=0; i< LV.n; i++)
    {

```

```

        cout << nv[i] << " degree " << LV.d[i] << endl;
    }
}

int _check_len= (int) floor(LEdge_num/LC.av_degree_node());

cout <<"simulation: sigma=" << sigma <<" , SNR=" << snr << "dB"
<< endl;
cout << "check_len" << _check_len << "   Info length" <<
_info_len
<< endl;

cout << "total edge is " << LEdge_num << "(using check node is
"
<<(int) floor(LC.av_degree_node()*_check_len)<< ")" << endl;

int _Interleaver[LEdge_num];

getInterleaver(_Interleaver, argv[argcount], LEdge_num);

RandomGenerator rand;

int simu_num=0;
int WBT=0;          //wrong bit counter
int WBL=0;          //wrong blocks counter
/*
  IRAcode *ira =new IRAcode(_info_len, _check_len, LEdge_num, nv, &LV,
&LC, sigma, &rand, _Interleaver);

  int k= ira->smartIteration();
  cout <<"stop at " <<k << "wrong bits " << ira->wrong_bits()<< endl;
*/
while(WBL < 20)
{
  simu_num++;
  IRAcode ira(_info_len, _check_len, LEdge_num, nv, &LV,
&LC, sigma, &rand, _Interleaver);
  int k= ira.smartIteration();
  int wb=ira.wrong_bits();
  if(wb>0) //wrong block
  {
    WBT+=wb;
    WBL++;
    cout << endl <<"wrong block " << simu_num << " with
"<<k << "ite, " \
<< wb << "wrong bits" << endl;

```

```
        cout << "\t \t current total wrong block " << WBL <<
endl;
    }
    else //correct block
        cout << "success block " << simu_num << " with " << k <<
"ite." << endl;
    }

    cout << "BER" << double(WBT)/double(_info_len*simu_num) <<
endl;
    cout << "WER" << double(WBL)/double(simu_num) << endl;

}
```