*//  main.cpp*

#include "BojayLED.hpp"

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

    Mat CieImage;

    DrawBojayCIE(1,60,1,CieImage);

}

*//  BojayLED.hpp*

#ifndef BojayLED\_hpp

#define BojayLED\_hpp

#include <stdio.h>

#include "CieImageClass.hpp"

void DrawBojayCIE(float r,float g,float b,Mat &OutPutImage);

#endif */\* BojayLED\_hpp \*/*

*//  BojayLED.cpp*

#include "BojayLED.hpp"

CieImageClass myCieImageClass;

void DrawBojayCIE(float r,float g,float b,Mat &OutPutImage)

{

*//获取CIE图片*

    myCieImageClass.DrawCIE();

    Mat img = myCieImageClass.CieOriginalImage;

    Mat dst;

    float x,y,z;

*//获取坐标*

    myCieImageClass.RGB2XYZ(r, g, b, x, y, z);

    myCieImageClass.DrawCoordinates(r, g, b, x, y, img);

    boxFilter(img, dst, IPL\_DEPTH\_8U, Size(1, 1), Point(-1, -1), true, BORDER\_DEFAULT);

    Mat BojayCIE =Mat(dst.rows+180, dst.cols+200, CV\_8UC3, Scalar(255,255,255));

*//添加坐标轴*

    myCieImageClass.InputFigure(dst, BojayCIE);

    string yylabel = "Y";

    string xxlabel = "X";

    string ttitle\_name = "      CIE 1931";

    myCieImageClass.yLabel(yylabel, Scalar(0, 0, 0),1,0,10);

    myCieImageClass.xLabel(xxlabel, Scalar(0, 0, 0), 1, 0, 10);

    myCieImageClass.Title(ttitle\_name);

    boxFilter(BojayCIE, BojayCIE, IPL\_DEPTH\_8U, Size(1.5, 1.5), Point(-1, -1), true, BORDER\_DEFAULT);

    imshow("view", BojayCIE);

    waitKey();

    imwrite("/Users/huyujian/BojayCIE.bmp", BojayCIE);

}

*//  CIE1931ApiClass.hpp*

#ifndef CIE1931ApiClass\_hpp

#define CIE1931ApiClass\_hpp

#include <stdio.h>

#include <opencv2/imgproc/imgproc\_c.h>

#include <opencv2/highgui/highgui\_c.h>

#include <opencv2/opencv.hpp>

using namespace cv;

using namespace std;

class CieImageClass

{

    typedef struct \_myPoint

    {

        double x;

        double y;

    }myPoint;

    vector<myPoint>mySmall;

public:

    int CieR,CieG,CieB;

    Mat ImgInside;  *//传入的中间的伪彩图*

    Mat ImgOutside;*//整个外框*

    Mat CieOriginalImage;

    int InStartCols;*//经计算得的内图像宽开始处*

    int InStartRows;*//经计算得的内图像高开始处*

    void InputFigure(Mat InsideFigure, Mat OuterFrame);*//把图片及外框大小加载进入大图片，并在大框中放置图片*

    void xLabel(string xLabelName, CvScalar LabelColor, double xMax, double xMin, const int xNumber);*//添加x轴*

    void yLabel(string yLabelName, CvScalar LabelColor, double yMax, double yMin,  const int yNumber);*//添加y轴*

    void Title(string TitleName, CvScalar TitleColor = Scalar(0, 0, 0));*//添加标题*

    void DrawCoordinates(float r,float g,float b,float x, float y,Mat &img);*//标定坐标*

    void XYZ2RGB(float x, float y, float z);

    void RGB2XYZ(int intputR, int intputG, int intputB,float &outputX, float &outputY, float &outputZ);

    void DrawCIE(void);

    void XYZtoRGB(float intputx, float intputy, float intputz,float &outputR, float &outputG, float &outputB);

};

#endif */\* CIE1931ApiClass\_hpp \*/*

*//  CIE1931ApiClass.cpp*

#include "CieImageClass.hpp"

#include <stdio.h>

#include <vector>

#include <string>

#include <fstream>

using namespace std;

using namespace cv;

void CieImageClass::XYZ2RGB(float x, float y, float z)

{

    double dr,dg,db;

    dr =  0.4185  \* x - 0.1587 \* y - 0.0828 \* z;

    dg = -0.0912 \* x + 0.2524 \* y + 0.0157\* z;

    db =  0.0009 \* x - 0.0025\* y + 0.1786 \* z;

    double max = 0;

    max = dr > dg ? dr : dg;

    max = max > db ? max : db;

    dr=dr/max\*255;

    dg=dg/max\*255;

    db=db/max\*255;

    dr = dr > 0 ? dr : 0;

    dg = dg > 0 ? dg : 0;

    db = db > 0 ? db : 0;

    if(dr > 255)

    {

        dr = 255;

    }

    if(dg > 255)

    {

        dg = 255;

    }

    if(db > 255)

    {

        db = 255;

    }

    CieR=int(dr+0.5);

    CieG=int(dg+0.5);

    CieB=int(db+0.5);

}

void CieImageClass::RGB2XYZ(int intputR, int intputG, int intputB,float &outputX, float &outputY, float &outputZ)

{

    double dr,dg,db;

    double dx,dy,dz;

    dr=intputR/255.0;

    dg=intputG/255.0;

    db=intputB/255.0;

    dr = dr > 0 ? dr : 0.000001;

    dg = dg > 0 ? dg : 0.000001;

    db = db > 0 ? db : 0.000001;

    dx = 2.7689\*dr + 1.7517\*dg + 1.1302\*db;

    dy = 1.0002\*dr + 4.5907\*dg + 0.0601\*db;

    dz =             0.0565\*dg + 5.5943\*db;

    outputX =dx/(dx+dy+dz);

    outputY =dy/(dx+dy+dz);

    outputZ =dz/(dx+dy+dz);

}

void CieImageClass::XYZtoRGB(float intputx, float intputy, float intputz,float &outputR, float &outputG, float &outputB)

{

    double dr,dg,db;

    dr =  0.4185  \* intputx - 0.1587 \* intputy - 0.0828 \* intputz;

    dg = -0.0912 \* intputx + 0.2524 \* intputy + 0.0157\* intputz;

    db =  0.0009 \* intputx - 0.0025\* intputy + 0.1786 \* intputz;

    double max = 0;

    max = dr > dg ? dr : dg;

    max = max > db ? max : db;

    dr=dr/max\*255;

    dg=dg/max\*255;

    db=db/max\*255;

    dr = dr > 0 ? dr : 0;

    dg = dg > 0 ? dg : 0;

    db = db > 0 ? db : 0;

    if(dr > 255)

    {

        dr = 255;

    }

    if(dg > 255)

    {

        dg = 255;

    }

    if(db > 255)

    {

        db = 255;

    }

    outputR=int(dr+0.5);

    outputG=int(dg+0.5);

    outputB=int(db+0.5);

}

void CieImageClass::DrawCIE()

{

    mySmall.clear();

    vector<double>mySmallx;

    vector<double>mySmally;

    vector<double>mySmallz;

    string temp;

    fstream Data("/Users/huyujian/Data.txt",ios::in);

    if(Data.is\_open())

    {

        while (getline(Data, temp)) {

            vector<double> nums;

            char \*s\_input = (char \*)temp.c\_str();

            const char \* split = " ";

            char \*p = strtok(s\_input, split);

            double a;

            while (p != NULL) {

                a = atof(p);

                nums.push\_back(a);

                p = strtok(NULL, split);

            }

            myPoint tempP;

            tempP.x =nums[1];

            tempP.y =nums[2];

            mySmall.push\_back(tempP);

            char debug[256] = {};

            sprintf(debug,"%f,%f",nums[1],nums[2]);

        }

    }

    struct \_node

    {

        double x;

        double y;

    };

*//1:选择3个顶点*

    struct \_node p1; *// 代表最下面的点*

    struct \_node p2; *// 代表最上面的点*

    struct \_node p3; *// 代表最右边的点*

*//代表最下面的点*

    p1.x = 0.173101;

    p1.y = 0.004774;

*//代表最上面的点*

    p2.x = 0.082053;

    p2.y = 0.83409;

*//代表最右边的点*

    p3.x = 0.73469;

    p3.y = 0.26531;

    double k3 = 0,b3= 0;

    k3 = (p3.y - p1.y) / (p3.x - p1.x);

    b3 = p3.y - p3.x \* k3;

*//左边数组*

    vector<struct \_node>leftArray;

    vector<struct \_node>rightArray;

    struct \_node tempPonit;

    for(int i=0; i<(int)mySmall.size(); i++)

    {

        tempPonit.x = mySmall[i].x;

        tempPonit.y = mySmall[i].y;

        if(i < 161)

        {

            leftArray.push\_back(tempPonit);

            if(i == 160)

            {

                rightArray.push\_back(tempPonit);

            }

        }

        else if(i>= 161 && i< 339)

        {

            rightArray.push\_back(tempPonit);

        }

    }

    vector<struct \_node>downArray;

    for(double y=0.001; y<0.265; y+=0.001)

    {

        if(y > p1.y && y <p3.y)

        {

            tempPonit.x = (y - b3)/k3;

            tempPonit.y = y;

            downArray.push\_back(tempPonit);

        }

    }

    IplImage \*cie = cvCreateImage(cvSize(1000, 1000), IPL\_DEPTH\_8U,3);

    memset(cie->imageData, 255, 1000\*1000\*3);

    double tempx= 0,tempy=0;

    double miny = 0,maxy=0;

    double minx = 0,maxx=0;

    double middlex=0;

    double minvalue1=999999;

    double minvalue2=999999;

    int maxindex = 0,minindex=0;

    cout << rightArray.size() << endl;

    for(int j=0; j<1000; j++)

    {

        tempy = (double)j / 1000.0;

        if(tempy < 0.004774  || tempy > 0.83409)

        {

            continue;

        }

        for(int i=0; i<1000; i++)

        {

            tempx = (double)i / 1000.0;

            if(tempx < 0.003636  || tempx > 0.73469)

            {

                continue;

            }

*//1:首先判断左边的数组*

            maxindex = 0;

            minindex=0;

            minvalue2 = 999999;

            minvalue1=999999;

            int k = 0;

            for(k=0; k<(int)leftArray.size(); k++)

            {

                double value1 = tempy - leftArray.at(k).y;

                double value2 = leftArray.at(k).y - tempy;

                if(value1 < minvalue1 && value1 > 0)

                {

                    minvalue1 = value1;

                    minindex = k;

                }

                if(value2 < minvalue2 && value2 > 0)

                {

                    minvalue2 = value2;

                    maxindex = k;

                }

            }

            miny = leftArray.at(minindex).y;

            minx = leftArray.at(minindex).x;

            maxy = leftArray.at(maxindex).y;

            maxx = leftArray.at(maxindex).x;

            middlex = minx - (tempy - miny)/(maxy - miny)\*(minx - maxx);

            if(tempx < middlex)

            {

*//cout << "Left out" << endl;*

                continue;

            }

*//#if 1*

*//1:首先判断右边的数组*

            maxindex = 0;

            minindex=0;

            minvalue2 = 999999;

            minvalue1=999999;

            for(int k=0; k<(int)rightArray.size(); k++)

            {

                double value1 = tempy - rightArray.at(k).y;

                double value2 = rightArray.at(k).y - tempy;

                if(value1 < minvalue1 && value1 > 0)

                {

                    minvalue1 = value1;

                    minindex = k;

                }

                if(value2 < minvalue2 && value2 > 0)

                {

                    minvalue2 = value2;

                    maxindex = k;

                }

            }

            miny = rightArray.at(minindex).y;

            minx = rightArray.at(minindex).x;

            maxy = rightArray.at(maxindex).y;

            maxx = rightArray.at(maxindex).x;

            middlex = minx - (tempy - miny)/(maxy - miny)\*(minx - maxx);

            if(tempx > middlex)

            {

                continue;

            }

*//1:首先判断下边的数组*

            maxindex = 0;

            minindex=0;

            minvalue2 = 999999;

            minvalue1=999999;

            for(int k=0; k<(int)downArray.size(); k++)

            {

                double value1 = tempy - downArray.at(k).y;

                double value2 = downArray.at(k).y - tempy;

                if(value1 < minvalue1 && value1 > 0)

                {

                    minvalue1 = value1;

                    minindex = k;

                }

                if(value2 < minvalue2 && value2 > 0)

                {

                    minvalue2 = value2;

                    maxindex = k;

                }

            }

            miny = downArray.at(minindex).y;

            minx = downArray.at(minindex).x;

            maxy = downArray.at(maxindex).y;

            maxx = downArray.at(maxindex).x;

            middlex = minx - (tempy - miny)/(maxy - miny)\*(minx - maxx);

            if(tempx > middlex)

            {

                continue;

            }

*//#endif*

            double x=0,y=0,z=0;

            x = tempx;

            y = tempy;

            z = 1- x - y;

            XYZ2RGB(x,y,z);

            \*(cie->imageData + (1000-j)\*cie->widthStep + 3\*i) = CieB;

            \*(cie->imageData + (1000-j)\*cie->widthStep + 3\*i + 1) = CieG;

            \*(cie->imageData + (1000-j)\*cie->widthStep + 3\*i + 2) = CieR;

        }

    }

    Mat image = cvarrToMat(cie);

    CieOriginalImage = cvarrToMat(cie);

    imwrite("/Users/huyujian/CIE.bmp", image);

    cvReleaseImage(&cie);

}

void CieImageClass::InputFigure(Mat InsideFigure, Mat OuterFrame)

{

    ImgInside = InsideFigure;

    ImgOutside = OuterFrame;

    InStartCols = OuterFrame.cols - InsideFigure.cols - 120;

    InStartRows = OuterFrame.rows - InsideFigure.rows - 60;

    for (int i = 0; i < InsideFigure.rows; i++)

    {

        for (int j = 0; j < InsideFigure.cols; j++)

        {

            ImgOutside.at<Vec3b>(i+ InStartRows,j+InStartCols)= ImgInside.at<Vec3b>(i,j);

        }

    }

}

void CieImageClass::yLabel(string yLabelName, CvScalar LabelColor, double yMax, double yMin, const int yNumber)

{

    line(ImgOutside, Point(InStartCols - 1,InStartRows - 1 ), Point( InStartCols - 1,InStartRows + ImgInside.rows ), LabelColor, 2, 8, 0);

*//line(ImgOutside, Point(InStartCols + ImgInside.cols ,InStartRows - 1),  Point(InStartCols + ImgInside.cols , InStartRows + ImgInside.rows ), Scalar(0, 0, 0), 2, 8, 0);*

    int ylabel\_step = ImgInside.rows / yNumber;

    for (int y = 0; y <= yNumber; y++)

    {

        double y\_jhj = double(y\*ylabel\_step) / double(ImgInside.rows)\*(yMax - yMin) + yMin;

        char str[20];

        sprintf(str,"%0.2f", y\_jhj);

        string result = str;

        line(ImgOutside, Point(InStartCols - 10, InStartRows + ImgInside.rows - y\*ylabel\_step), Point(InStartCols - 1, InStartRows + ImgInside.rows - y\*ylabel\_step), LabelColor, 2, 8, 0);

        putText(ImgOutside, result, Point(InStartCols - 55, InStartRows + ImgInside.rows - y\*ylabel\_step +3), CV\_FONT\_HERSHEY\_SIMPLEX, 0.5, LabelColor, 2, 8);

    }

    putText(ImgOutside, yLabelName, Point(100, 130), CV\_FONT\_HERSHEY\_SIMPLEX, 0.7, LabelColor, 2, 8);

}

void CieImageClass::xLabel(string xLabelName, CvScalar LabelColor, double xMax, double xMin, const int xNumber)

{

*//line(ImgOutside, Point(InStartCols - 1, InStartRows - 1), Point(InStartCols + ImgInside.cols , InStartRows - 1), Scalar(0, 0, 0), 2, 8, 0);*

    line(ImgOutside, Point(InStartCols - 1, InStartRows + ImgInside.rows ), Point(InStartCols + ImgInside.cols , InStartRows + ImgInside.rows ), LabelColor, 2, 8, 0);

    int xlabel\_step = ImgInside.cols / xNumber;

    for (int x = 0; x <= xNumber; x++)

    {

        double x\_jhj = double(x\*xlabel\_step) / double(ImgInside.cols)\*(xMax - xMin) + xMin;

        char str[20];

        sprintf(str, "%0.2f", x\_jhj);

        string result = str;

        line(ImgOutside, Point(InStartCols + x\*xlabel\_step, InStartRows + ImgInside.rows), Point(InStartCols + x\*xlabel\_step, InStartRows + ImgInside.rows +10), LabelColor, 2, 8, 0);

        putText(ImgOutside, result, Point(InStartCols + x\*xlabel\_step-15, InStartRows + ImgInside.rows +22), CV\_FONT\_HERSHEY\_SIMPLEX, 0.5, LabelColor, 2, 8);

    }

    putText(ImgOutside, xLabelName, Point(InStartCols + ImgInside.cols, InStartRows + ImgInside.rows-20), CV\_FONT\_HERSHEY\_SIMPLEX, 0.7, LabelColor, 2, 8);

}

void CieImageClass::Title(string TitleName, CvScalar TitleColor)

{

    putText(ImgOutside, TitleName, Point(InStartCols + ImgInside.cols / 2 -200, InStartRows - 20), CV\_FONT\_HERSHEY\_SIMPLEX, 1, TitleColor, 2, 8);

}

void CieImageClass::DrawCoordinates(float r,float g,float b,float x, float y,Mat &img)

{

*//画X轴虚线*

    for(int LineCount = 0;LineCount < x\*1000/10;LineCount++)

    {

        int LastCount = x\*1000/10;

        if(LineCount == LastCount)

        {

            line(img, Point(LineCount\*10+5,1000-y\*1000), Point(LineCount\*10,1000-y\*1000), Scalar(0,0,0),2,8);

            continue;

        }

        if(LineCount%2==0)

        {

            line(img, Point(LineCount\*10+10,1000-y\*1000), Point(LineCount\*10,1000-y\*1000), Scalar(0,0,0),2,8);

        }

    }

*//画Y轴虚线*

    for(int LineCount = 0;LineCount < y\*1000/10;LineCount++)

    {

        int LastCount = y\*1000/10;

        if(LineCount == LastCount)

        {

            line(img, Point(x\*1000,1000-LineCount\*10+15), Point(x\*1000,1000-LineCount\*10), Scalar(0,0,0),2,8);

            continue;

        }

        if(LineCount%2==0)

        {

            line(img, Point(x\*1000,1000-LineCount\*10+10), Point(x\*1000,1000-LineCount\*10), Scalar(0,0,0),2,8);

        }

    }

*//坐标标点*

    circle(img,Point(x\*1000,1000-y\*1000),5,Scalar(0,0,0));

*//画文字RGB与颜色*

    char strR[256] = {};

    sprintf(strR, "R: %0.f",r);

    char strG[256] = {};

    sprintf(strG, "G: %0.f",g);

    char strB[256] = {};

    sprintf(strB, "B: %0.f",b);

    putText(img, "Color", Point(560,150), 0, 1, Scalar(0,0,0),2,8);

    putText(img, strR, Point(560,270), 0, 1, Scalar(0,0,0),2,8);

    putText(img, strG, Point(560,300), 0, 1, Scalar(0,0,0),2,8);

    putText(img, strB, Point(560,330), 0, 1, Scalar(0,0,0),2,8);

    circle(img,Point(600,200),20,Scalar(b,g,r),-1);

    circle(img,Point(600,200),20,Scalar(0,0,0));

*//画标定坐标*

    char pos[256] = {};

    sprintf(pos, "[%0.6f,%0.6f]",x,y);

    putText(img, pos, Point(x\*1000+5,1000-y\*1000), CV\_FONT\_HERSHEY\_SIMPLEX, 0.7, Scalar(0,0,0),2,8);

}

//Data.txt

360 0.17556 0.005294

361 0.175483 0.005286

362 0.1754 0.005279

363 0.175317 0.005271

364 0.175237 0.005263

365 0.175161 0.005256

366 0.175088 0.005247

367 0.175015 0.005236

368 0.174945 0.005226

369 0.17488 0.005221

370 0.174821 0.005221

371 0.17477 0.005229

372 0.174722 0.005238

373 0.174665 0.005236

374 0.174595 0.005218

375 0.17451 0.005182

376 0.174409 0.005127

377 0.174308 0.005068

378 0.174222 0.005017

379 0.174156 0.004981

380 0.174112 0.004964

381 0.174088 0.004964

382 0.174073 0.004973

383 0.174057 0.004982

384 0.174036 0.004986

385 0.174008 0.004981

386 0.173972 0.004964

387 0.173932 0.004943

388 0.173889 0.004926

389 0.173845 0.004916

390 0.173801 0.004915

391 0.173754 0.004925

392 0.173705 0.004937

393 0.173655 0.004944

394 0.173606 0.00494

395 0.17356 0.004923

396 0.173514 0.004895

397 0.173468 0.004865

398 0.173424 0.004836

399 0.17338 0.004813

400 0.173337 0.004797

401 0.173291 0.004786

402 0.173238 0.004779

403 0.173174 0.004775

404 0.173101 0.004774

405 0.173021 0.004775

406 0.172934 0.004781

407 0.172843 0.004791

408 0.172751 0.004799

409 0.172662 0.004802

410 0.172577 0.004799

411 0.172489 0.004795

412 0.172396 0.004796

413 0.172296 0.004803

414 0.172192 0.004815

415 0.172087 0.004833

416 0.171982 0.004855

417 0.171871 0.004889

418 0.171741 0.004939

419 0.171587 0.00501

420 0.171407 0.005102

421 0.171206 0.005211

422 0.170993 0.005334

423 0.170771 0.00547

424 0.170541 0.005621

425 0.170301 0.005789

426 0.17005 0.005974

427 0.169786 0.006177

428 0.169505 0.006398

429 0.169203 0.006639

430 0.168878 0.0069

431 0.168525 0.007184

432 0.168146 0.007491

433 0.167746 0.007821

434 0.167328 0.008175

435 0.166895 0.008556

436 0.166446 0.008964

437 0.165977 0.009402

438 0.165483 0.009865

439 0.164963 0.010351

440 0.164412 0.010858

441 0.163828 0.011385

442 0.16321 0.011937

443 0.162552 0.01252

444 0.161851 0.013137

445 0.161105 0.013793

446 0.16031 0.014491

447 0.159466 0.015232

448 0.158573 0.016015

449 0.157631 0.01684

450 0.156641 0.017705

451 0.155605 0.018609

452 0.154525 0.019556

453 0.153397 0.020554

454 0.152219 0.021612

455 0.150985 0.02274

456 0.149691 0.02395

457 0.148337 0.025247

458 0.146928 0.026635

459 0.145468 0.028118

460 0.14396 0.029703

461 0.142405 0.031394

462 0.140796 0.033213

463 0.139121 0.035201

464 0.137364 0.037403

465 0.135503 0.039879

466 0.133509 0.042692

467 0.131371 0.045876

468 0.129086 0.04945

469 0.126662 0.053426

470 0.124118 0.057803

471 0.121469 0.062588

472 0.118701 0.06783

473 0.115807 0.073581

474 0.112776 0.079896

475 0.109594 0.086843

476 0.106261 0.094486

477 0.102776 0.102864

478 0.099128 0.112007

479 0.095304 0.121945

480 0.091294 0.132702

481 0.087082 0.144317

482 0.08268 0.156866

483 0.078116 0.17042

484 0.073437 0.185032

485 0.068706 0.200723

486 0.063993 0.217468

487 0.059316 0.235254

488 0.054667 0.254096

489 0.050031 0.274002

490 0.045391 0.294976

491 0.040757 0.316981

492 0.036195 0.3399

493 0.031756 0.363598

494 0.027494 0.387921

495 0.02346 0.412703

496 0.019705 0.437756

497 0.016268 0.462955

498 0.013183 0.488207

499 0.010476 0.513404

500 0.008168 0.538423

501 0.006285 0.563068

502 0.004875 0.587116

503 0.003982 0.610447

504 0.003636 0.633011

505 0.003859 0.654823

506 0.004646 0.675898

507 0.006011 0.69612

508 0.007988 0.715342

509 0.010603 0.733413

510 0.01387 0.750186

511 0.017766 0.765612

512 0.022244 0.77963

513 0.027273 0.792104

514 0.03282 0.802926

515 0.038852 0.812016

516 0.045328 0.819391

517 0.052177 0.825164

518 0.059326 0.829426

519 0.066716 0.832274

520 0.074302 0.833803

521 0.082053 0.83409

522 0.089942 0.833289

523 0.09794 0.831593

524 0.106021 0.829178

525 0.114161 0.826207

526 0.122347 0.82277

527 0.130546 0.818928

528 0.138702 0.814774

529 0.146773 0.810395

530 0.154722 0.805864

531 0.162535 0.801238

532 0.170237 0.796519

533 0.17785 0.791687

534 0.185391 0.786728

535 0.192876 0.781629

536 0.200309 0.776399

537 0.20769 0.771055

538 0.21503 0.765595

539 0.222337 0.76002

540 0.22962 0.754329

541 0.236885 0.748524

542 0.244133 0.742614

543 0.251363 0.736606

544 0.258578 0.730507

545 0.265775 0.724324

546 0.272958 0.718062

547 0.280129 0.711725

548 0.287292 0.705316

549 0.29445 0.698842

550 0.301604 0.692308

551 0.30876 0.685712

552 0.315914 0.679063

553 0.323066 0.672367

554 0.330216 0.665628

555 0.337363 0.658848

556 0.344513 0.652028

557 0.351664 0.645172

558 0.358814 0.638287

559 0.365959 0.631379

560 0.373102 0.624451

561 0.380244 0.617502

562 0.387379 0.610542

563 0.394507 0.603571

564 0.401626 0.596592

565 0.408736 0.589607

566 0.415836 0.582618

567 0.422921 0.575631

568 0.429989 0.568649

569 0.437036 0.561676

570 0.444062 0.554714

571 0.451065 0.547766

572 0.458041 0.540837

573 0.464986 0.53393

574 0.471899 0.527051

575 0.478775 0.520202

576 0.485612 0.513389

577 0.492405 0.506615

578 0.499151 0.499887

579 0.505845 0.493211

580 0.512486 0.486591

581 0.519073 0.480029

582 0.5256 0.473527

583 0.532066 0.467091

584 0.538463 0.460725

585 0.544787 0.454434

586 0.551031 0.448225

587 0.557193 0.442099

588 0.563269 0.436058

589 0.569257 0.430102

590 0.575151 0.424232

591 0.580953 0.418447

592 0.58665 0.412758

593 0.592225 0.40719

594 0.597658 0.401762

595 0.602933 0.396497

596 0.608035 0.391409

597 0.612977 0.386486

598 0.617779 0.381706

599 0.622459 0.377047

600 0.627037 0.372491

601 0.631521 0.368026

602 0.6359 0.363665

603 0.640156 0.359428

604 0.644273 0.355331

605 0.648233 0.351395

606 0.652028 0.347628

607 0.655669 0.344018

608 0.659166 0.340553

609 0.662528 0.337221

610 0.665764 0.334011

611 0.668874 0.330919

612 0.671859 0.327947

613 0.67472 0.325095

614 0.677459 0.322362

615 0.680079 0.319747

616 0.682582 0.317249

617 0.684971 0.314863

618 0.68725 0.312586

619 0.689426 0.310414

620 0.691504 0.308342

621 0.69349 0.306366

622 0.695389 0.304479

623 0.697206 0.302675

624 0.698944 0.30095

625 0.700606 0.299301

626 0.702193 0.297725

627 0.703709 0.296217

628 0.705163 0.29477

629 0.706563 0.293376

630 0.707918 0.292027

631 0.709231 0.290719

632 0.7105 0.289453

633 0.711724 0.288232

634 0.712901 0.287057

635 0.714032 0.285929

636 0.715117 0.284845

637 0.716159 0.283804

638 0.717159 0.282806

639 0.718116 0.28185

640 0.719033 0.280935

641 0.719912 0.280058

642 0.720753 0.279219

643 0.721555 0.27842

644 0.722315 0.277662

645 0.723032 0.276948

646 0.723702 0.276282

647 0.724328 0.27566

648 0.724914 0.275078

649 0.725467 0.27453

650 0.725992 0.274008

651 0.726495 0.273505

652 0.726975 0.273025

653 0.727432 0.272568

654 0.727864 0.272136

655 0.728272 0.271728

656 0.728656 0.271344

657 0.72902 0.27098

658 0.729361 0.270639

659 0.729678 0.270322

660 0.729969 0.270031

661 0.730234 0.269766

662 0.730474 0.269526

663 0.730693 0.269307

664 0.730896 0.269104

665 0.731089 0.268911

666 0.73128 0.26872

667 0.731467 0.268533

668 0.73165 0.26835

669 0.731826 0.268174

670 0.731993 0.268007

671 0.73215 0.26785

672 0.7323 0.2677

673 0.732443 0.267557

674 0.732581 0.267419

675 0.732719 0.267281

676 0.732859 0.267141

677 0.733 0.267

678 0.733142 0.266858

679 0.733281 0.266719

680 0.733417 0.266583

681 0.733551 0.266449

682 0.733683 0.266317

683 0.733813 0.266187

684 0.733936 0.266064

685 0.734047 0.265953

686 0.734143 0.265857

687 0.734221 0.265779

688 0.734286 0.265714

689 0.734341 0.265659

690 0.73439 0.26561

691 0.734438 0.265562

692 0.734482 0.265518

693 0.734523 0.265477

694 0.73456 0.26544

695 0.734592 0.265408

696 0.734621 0.265379

697 0.734649 0.265351

698 0.734673 0.265327

699 0.73469 0.26531

700 0.73469 0.26531

701 0.73469 0.26531

702 0.73469 0.26531

703 0.73469 0.26531

704 0.73469 0.26531

705 0.73469 0.26531

706 0.73469 0.26531

707 0.73469 0.26531

708 0.73469 0.26531

709 0.73469 0.26531

710 0.73469 0.26531

711 0.73469 0.26531

712 0.73469 0.26531

713 0.73469 0.26531

714 0.73469 0.26531

715 0.73469 0.26531

716 0.73469 0.26531

717 0.73469 0.26531

718 0.73469 0.26531

719 0.73469 0.26531

720 0.73469 0.26531

721 0.73469 0.26531

722 0.73469 0.26531

723 0.73469 0.26531

724 0.73469 0.26531

725 0.73469 0.26531

726 0.73469 0.26531

727 0.73469 0.26531

728 0.73469 0.26531

729 0.73469 0.26531

730 0.73469 0.26531

731 0.73469 0.26531

732 0.73469 0.26531

733 0.73469 0.26531

734 0.73469 0.26531

735 0.73469 0.26531

736 0.73469 0.26531

737 0.73469 0.26531

738 0.73469 0.26531

739 0.73469 0.26531

740 0.73469 0.26531

741 0.73469 0.26531

742 0.73469 0.26531

743 0.73469 0.26531

744 0.73469 0.26531

745 0.73469 0.26531

746 0.73469 0.26531

747 0.73469 0.26531

748 0.73469 0.26531

749 0.73469 0.26531

750 0.73469 0.26531

751 0.73469 0.26531

752 0.73469 0.26531

753 0.73469 0.26531

754 0.73469 0.26531

755 0.73469 0.26531

756 0.73469 0.26531

757 0.73469 0.26531

758 0.73469 0.26531

759 0.73469 0.26531

760 0.73469 0.26531

761 0.73469 0.26531

762 0.73469 0.26531

763 0.73469 0.26531

764 0.73469 0.26531

765 0.73469 0.26531

766 0.73469 0.26531

767 0.73469 0.26531

768 0.73469 0.26531

769 0.73469 0.26531

770 0.73469 0.26531

771 0.73469 0.26531

772 0.73469 0.26531

773 0.73469 0.26531

774 0.73469 0.26531

775 0.73469 0.26531

776 0.73469 0.26531

777 0.73469 0.26531

778 0.73469 0.26531

779 0.73469 0.26531

780 0.73469 0.26531

781 0.73469 0.26531

782 0.73469 0.26531

783 0.73469 0.26531

784 0.73469 0.26531

785 0.73469 0.26531

786 0.73469 0.26531

787 0.73469 0.26531

788 0.73469 0.26531

789 0.73469 0.26531

790 0.73469 0.26531

791 0.73469 0.26531

792 0.73469 0.26531

793 0.73469 0.26531

794 0.73469 0.26531

795 0.73469 0.26531

796 0.73469 0.26531

797 0.73469 0.26531

798 0.73469 0.26531

799 0.73469 0.26531

800 0.73469 0.26531

801 0.73469 0.26531

802 0.73469 0.26531

803 0.73469 0.26531

804 0.73469 0.26531

805 0.73469 0.26531

806 0.73469 0.26531

807 0.73469 0.26531

808 0.73469 0.26531

809 0.73469 0.26531

810 0.73469 0.26531

811 0.73469 0.26531

812 0.73469 0.26531

813 0.73469 0.26531

814 0.73469 0.26531

815 0.73469 0.26531

816 0.73469 0.26531

817 0.73469 0.26531

818 0.73469 0.26531

819 0.73469 0.26531

820 0.73469 0.26531

821 0.73469 0.26531

822 0.73469 0.26531

823 0.73469 0.26531

824 0.73469 0.26531

825 0.73469 0.26531

826 0.73469 0.26531

827 0.73469 0.26531

828 0.73469 0.26531

829 0.73469 0.26531

830 0.73469 0.26531