2.1 Task 1: Improved Printing of the Grid

```
if(p==0)
 {
            printf("\nsend from processor %d\n",p);
            k = n;
            if (k > 122)
                        k = 122;
             for (i = 1; i < m-1; i++)
                        { for (j = 1; j < k-1; j++)
                                    printf("%c", grayscale10[(int)(hp[n*i+j]+0.5)]);
                        printf("\n");
                        }
              fflush(stdout);
              usleep(100);
              MPI_Send(&message,1,MPI_INT,1,0,MPI_COMM_WORLD);
 }else if(p>0 && p<P-1){
            int src,dest;
             src=p-1;
             dest=p+1;
            MPI_Recv(&message,1,MPI_INT,src,0,MPI_COMM_WORLD,&status);
            printf("\n from processor %d\n",p);
            k = n;
            if (k > 122)
            k = 122;
            for (i = 1; i < m-1; i++)
                       { for (j = 1; j < k-1; j++)
                                    printf("%c", grayscale10[(int)(hp[n*i+j]+0.5)]);
                       printf("\n");
                        }
            fflush(stdout);
            usleep(100);
            MPI_Send(&message,1,MPI_INT,dest,0,MPI_COMM_WORLD);
 }else if(p=P-1){
            int src,dest;
            src=p-1;
            MPI_Recv(&message,1,MPI_INT,src,0,MPI_COMM_WORLD,&status);
            printf("\n from processor %d\n",p);
            k = n;
            if (k > 122)
            k = 122;
            for (i = 1; i < m-1; i++)
                       { for (j = 1; j < k-1; j++)
                                    printf("%c", grayscale10[(int)(hp[n*i+j]+0.5)]);
                       printf("\n");
                       }
            fflush(stdout);
            usleep(100);
 }
```

2.2 Task 2: Adding a Pacheco Stopping Test

n	time	iteration	relative speedup	the ratio of t_{comp} to t_{comm}
1	0.539601s	4300	1	Nan
2	0.259221s	4300	2.0816	5:1
4	0.376133s	4300	1.4346	2:1
8	0.728641s	4300	0.7406	2:3

(The tolerance in this code is 0.05, it is different from the material (that is tolerance=0.01). I don't know which one do I need to use, so I just use the data in the original code.)

```
 float ep,sum;
  ep=0;sum=0;

  if(t%100==0){
  for (i = 1; i < m-1; i++)
    for (j = 1; j < n-1; j++)
        ep=ep+(hp[n*i+j]-hnew[n*i+j])*(hp[n*i+j]-hnew[n*i+j]);
  MPI_Allreduce(&ep,&sum,1,MPI_FLOAT,MPI_SUM,MPI_COMM_WORLD);
  if( sum<=TOLERANCE*TOLERANCE){
   return 1;}
  }
```

2.3 Task 3: MPI Asynchronous Non-blocking Implementation

Explain why we need to wait for the MPI Irecv and MPI Isend to complete at the two specific points in the code? In this code we use nonblocking method to impelment the communication. So inside each interior rows,i=2...m-2,do not need to wait. But for ghost cell 0 and m-1, we need to receive the data from other processor, due to the non blocking communication Isend(), we need to wait the data, then update the row 1 and m-2.

n	time	iteration	relative speedup	the ratio of t_{comp} to t_{comm}
1	0.234487s	4300	1	Nan
2	0.286044s	4300	0.8198	14:1
4	0.116864s	4300	2.0065	7:2
8	0.143353s	4300	1.6357	5:2