# Finite Difference Algorithm Jacobi's Method

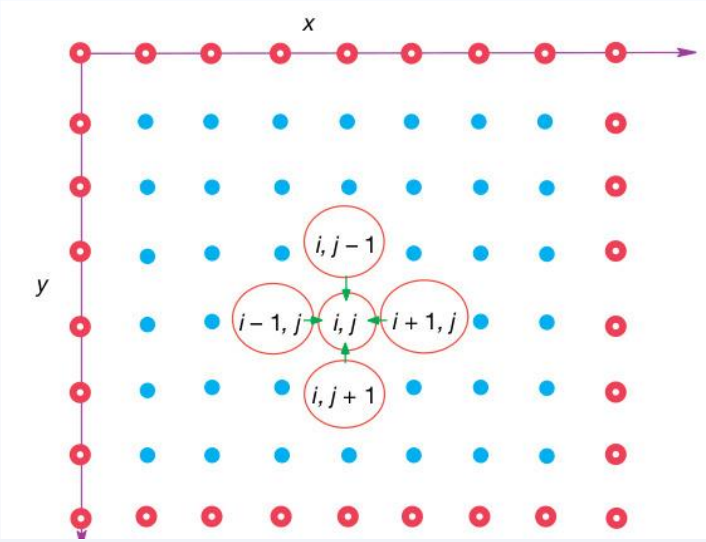

Arnold Alonso Alvarez

Universidad Distrital Francisco Jose de Caldas

Nutty.blood@gmail.com


# Introduction

To solve our 2D PDE numerically, we divide space up into a lattice and solve for U at each site on the lattice. Because we will express derivatives in terms of the finite differences in the values of U at the lattice sites, this is called a finite-difference method.




# 2D Laplace's equation

$$U(x+\Delta x, y) = U(x, y) + \frac{\partial U}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 U}{\partial x^2}(\Delta x)^2 + \cdots \quad ,$$

$$U(x-\Delta x, y) = U(x, y) - \frac{\partial U}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 U}{\partial x^2}(\Delta x)^2 - \cdots \quad ,$$

$$U(x, y+\Delta y) = U(x, y) + \frac{\partial U}{\partial y}\Delta y + \frac{1}{2}\frac{\partial^2 U}{\partial y^2}(\Delta y)^2 + \cdots \quad ,$$

$$U(x, y-\Delta y) = U(x, y) - \frac{\partial U}{\partial y}\Delta y + \frac{1}{2}\frac{\partial^2 U}{\partial y^2}(\Delta y)^2 - \cdots \quad .$$

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

# Jacobi's Method

$$\frac{\partial^2 U(x,y)}{\partial x^2} \simeq \frac{U(x+\Delta x, y) + U(x-\Delta x, y) - 2U(x,y)}{(\Delta x)^2},$$

$$\frac{\partial^2 U(x,y)}{\partial y^2} \simeq \frac{U(x, y+\Delta y) + U(x, y-\Delta y) - 2U(x,y)}{(\Delta y)^2}.$$

$$\frac{U(x+\Delta x, y) + U(x-\Delta x, y) - 2U(x,y)}{(\Delta x)^2} + \frac{U(x, y+\Delta y) + U(x, y-\Delta y) - 2U(x,y)}{(\Delta y)^2} = 0$$

# Jacobi's Method

$$u_{i,j} = \frac{1}{4}\left[u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}\right]$$

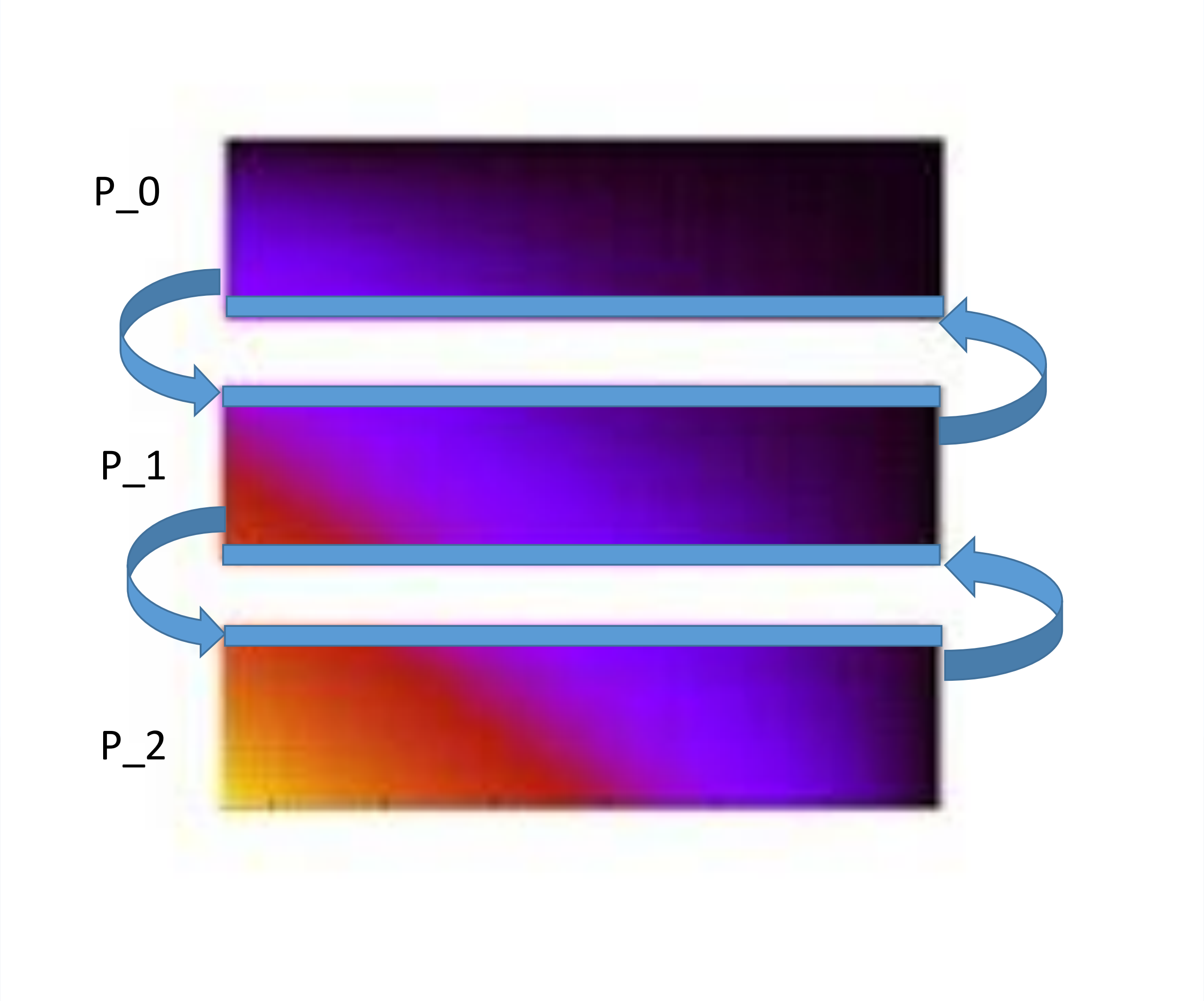
```
void evolve( double *matrix, double *new_matrix, size_t Loc_dimention, size_t dimention){

  size_t i , j;
  //This will be a row dominant program.
  for( i = 1 ; i <= Loc_dimention; ++i )
    for( j = 1; j <= dimention; ++j )
      new_matrix[ ( i * ( dimention + 2 ) ) + j ] = ( 0.25 ) *
        ( matrix[ ( ( i - 1 ) * ( dimention + 2 ) ) + j ] +
          matrix[ ( i * ( dimention + 2 ) ) + ( j + 1 ) ] +
          matrix[ ( ( i + 1 ) * ( dimention + 2 ) ) + j ] +
          matrix[ ( i * ( dimention + 2 ) ) + ( j - 1 ) ] );
}
```

# Initialization

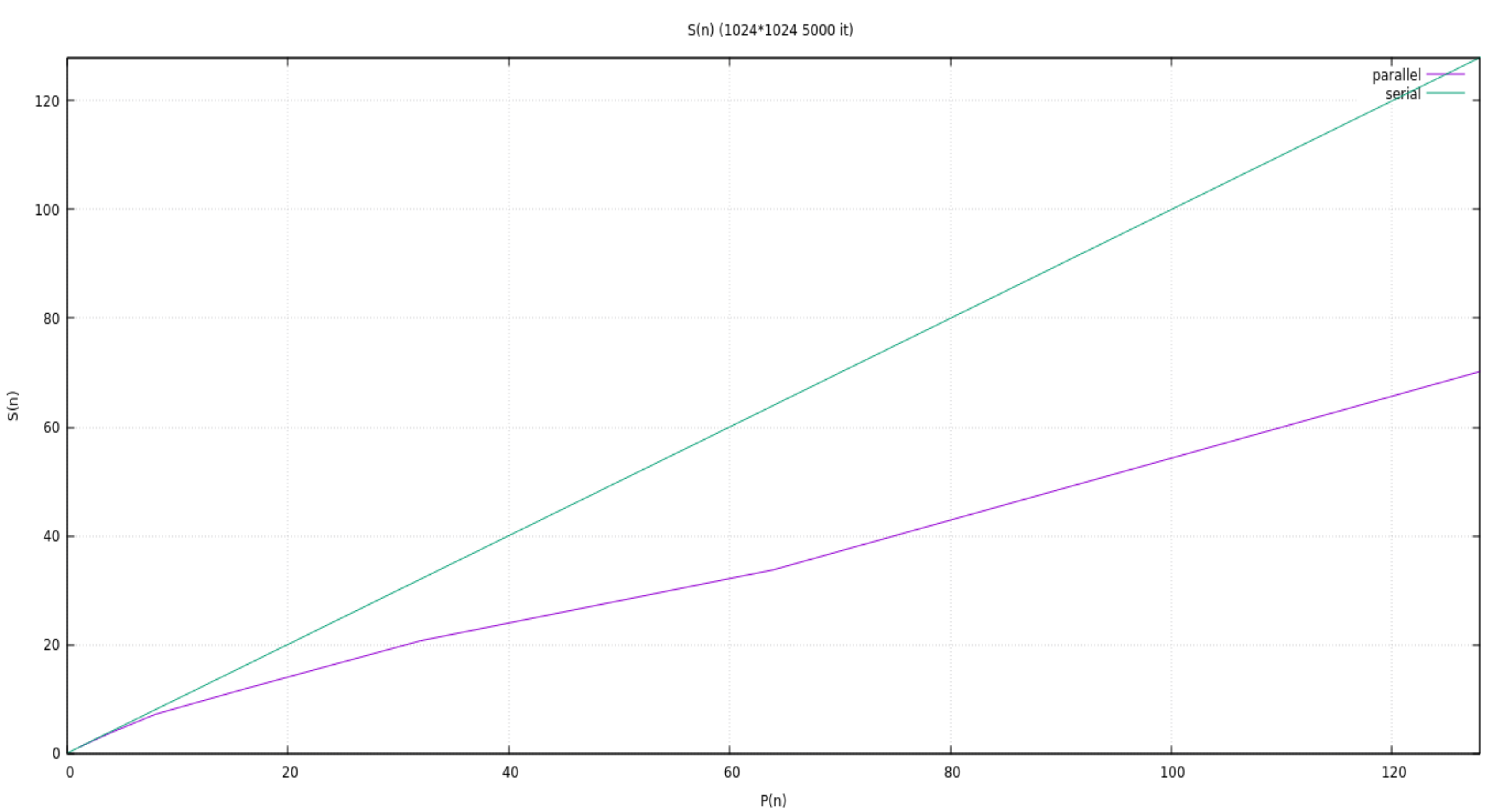
```
for(i = 1; i <= Loc_dimention; i ++){
  matrix[i*(dimention + 2)] = i*increment + rank*(Loc_dimention*increment);
  new_matrix[i*(dimention + 2)] = i*increment + rank*(Loc_dimention*increment);
  for(j = 1; j <= dimention; j ++){
    matrix[ (i*(dimention + 2)) + j] = 0.5;
  }
}

if(rank == (npe - 1)){
  for( j = 0; j <= dimention + 1; j ++){
    matrix[((Loc_dimention + 1)*(dimention + 2)) + (dimention + 1 - j)] = j*increment;
    new_matrix[((Loc_dimention + 1)*(dimention + 2)) + (dimention + 1 - j)] = j*increment;
  }
}
```

# Communication between threads

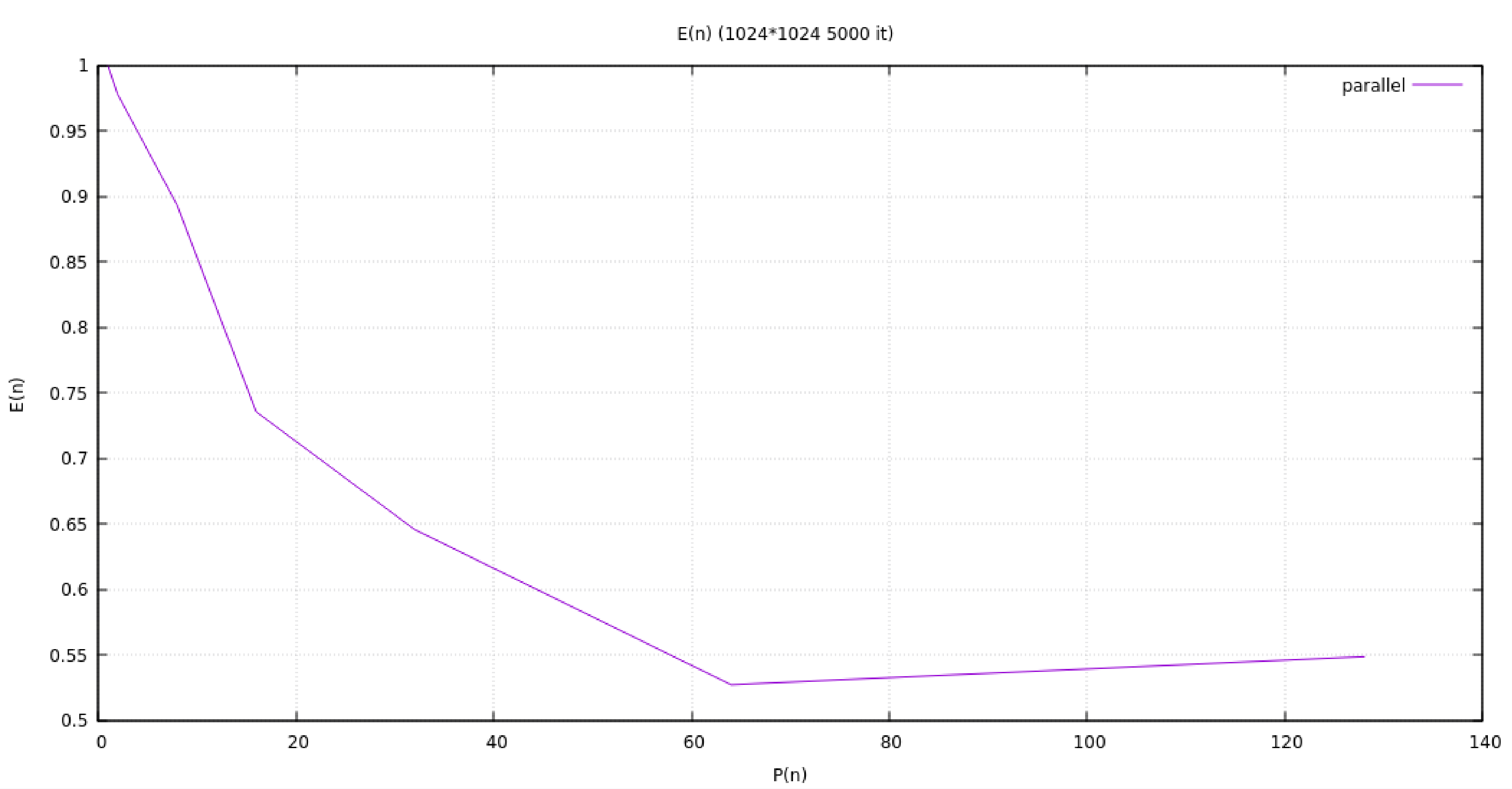
```
if(rank == 0){
  MPI_Sendrecv(&matrix[Loc_dimention*(dimention + 2)], (dimention + 2), MPI_DOUBLE, 1, 10,
               &matrix[(Loc_dimention + 1)*(dimention + 2)], (dimention + 2), MPI_DOUBLE , 1, 10, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
}
else if(rank == (npe-1)){
  MPI_Sendrecv(&matrix[dimention + 2], (dimention + 2), MPI_DOUBLE, (npe - 2), 10,
               &matrix[0], (dimention + 2), MPI_DOUBLE, (npe - 2), 10, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
}
else{
  MPI_Sendrecv(&matrix[dimention + 2], (dimention + 2), MPI_DOUBLE, (rank - 1), 10,
               &matrix[0], (dimention + 2), MPI_DOUBLE, (rank - 1), 10, MPI_COMM_WORLD, MPI_STATUS_IGNORE );

  MPI_Sendrecv(&matrix[Loc_dimention*(dimention + 2)], (dimention + 2), MPI_DOUBLE, (rank + 1), 10,
               &matrix[(Loc_dimention + 1)*(dimention + 2)], (dimention + 2),
               MPI_DOUBLE, (rank + 1), 10, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
}
```


P_0
P_1
P_2

# Speed-Up

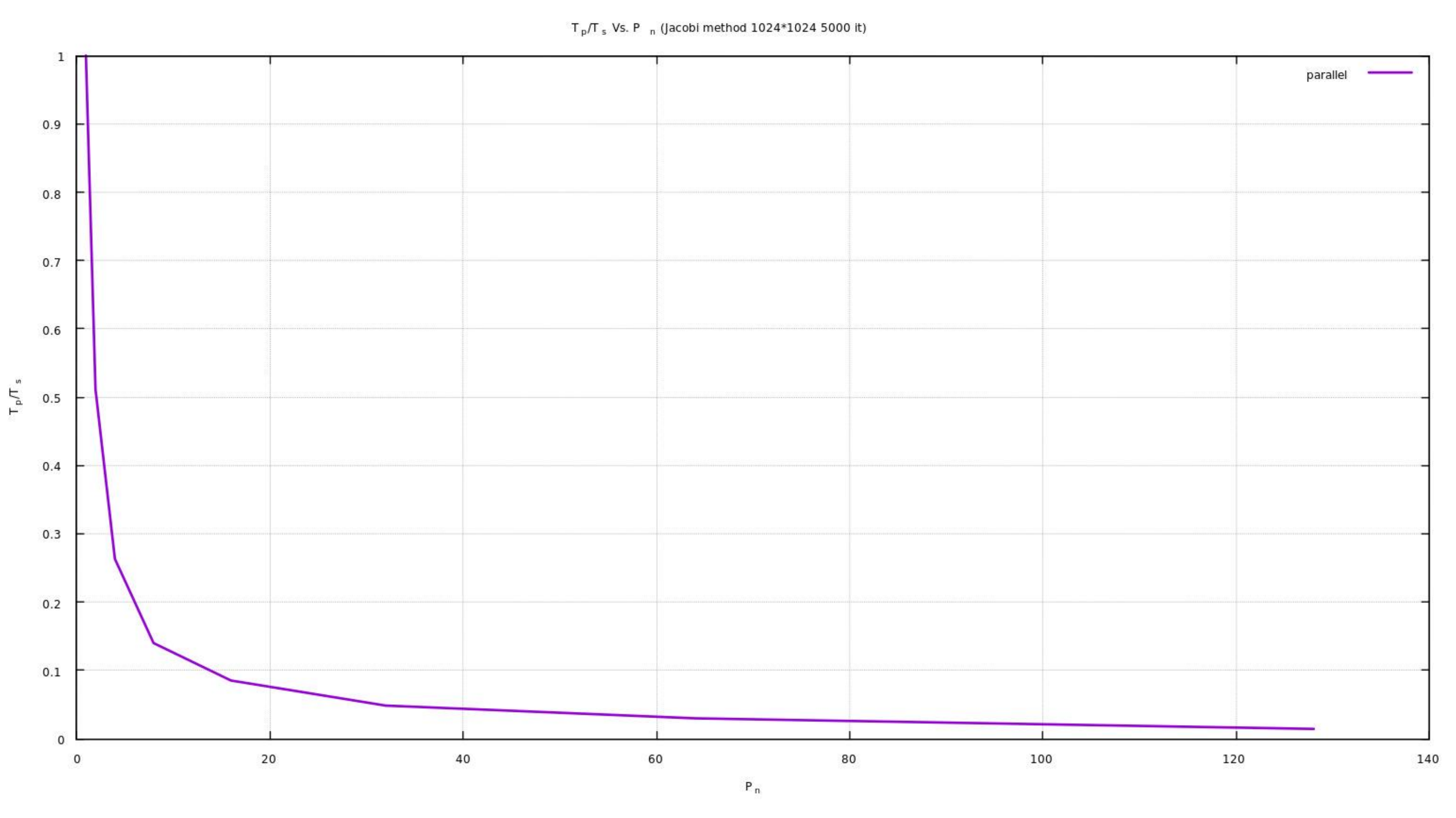


# Efficiency



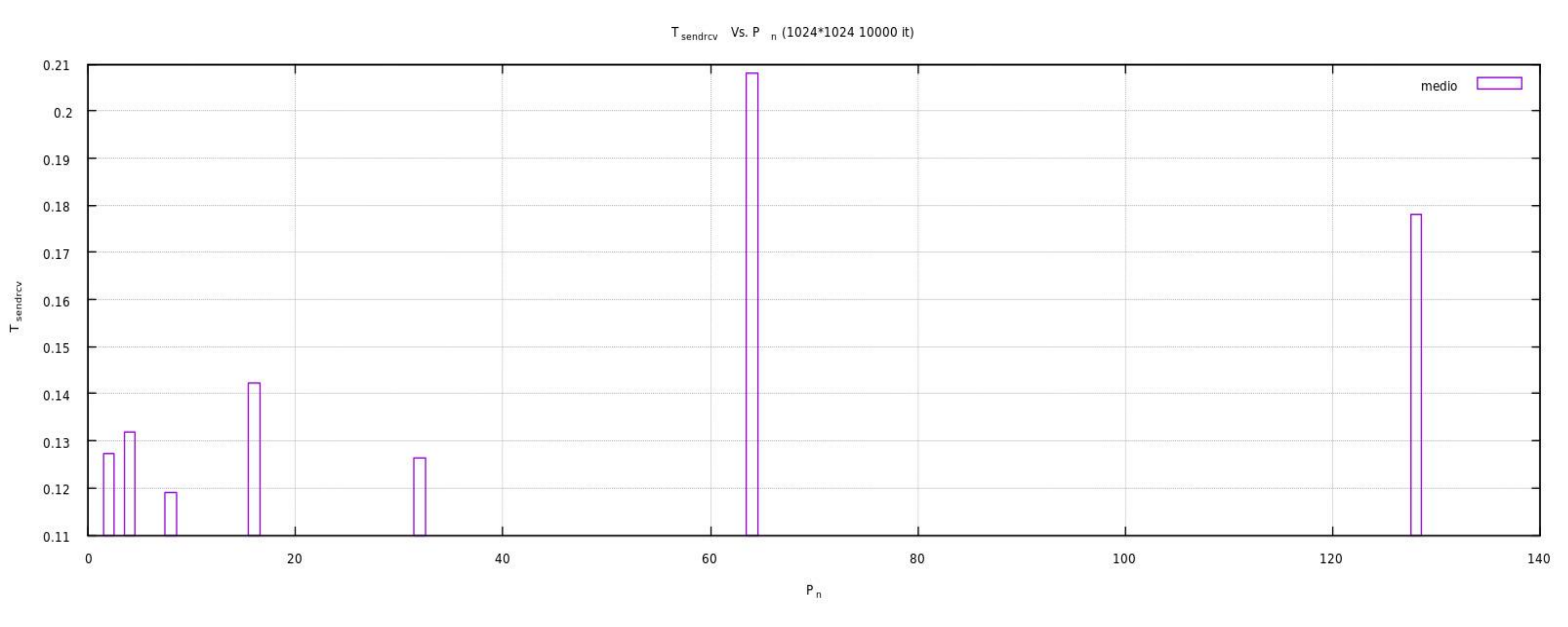
E(n) (1024*1024 5000 it)
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
E(n)
parallel
0
20
40
60
80
100
120
140
P(n)

# T Vs. P(n)



T p/T s Vs. P n (Jacobi method 1024*1024 5000 it)
parallel
T p/T s
P n
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
20
40
60
80
100
120
140

# Shipping and receiving time



# Final graphics