3D FFT Poisson solver for calculation thermal resistance.

(Черновик статьи в журнал 20,05,2021)

Исправлено *,*,*

Фрагменты математической постановки задачи.

Сетка в плоскости.

```
lengthx = 2050.0e-6;
lengthy = 1625.0e-6;

M = 2049;//511;
N = 129;// 511;
```

Подложка.

```
const int ilayer_count = 8;
//const int idiv_layer = 4;
STACK_LAYERS* stack_layer = new STACK_LAYERS[ilayer_count];
stack_layer[0].chikness = 250.0e-6;// 1.6e-3;
stack_layer[0].lambda = 210.0; // МД40
stack_layer[0].idiv_layer = 7;
stack_layer[1].chikness = 25.0e-6;
stack_layer[1].lambda = 57.0; // AuSn
stack_layer[1].idiv_layer = 4;
stack_layer[2].chikness = 250.0e-6;
stack_layer[2].lambda = 390.0; // Cu
stack_layer[2].idiv_layer = 7;
stack_layer[3].chikness = 25.0e-6;
stack_layer[3].lambda = 57.0; // AuSn
stack_layer[3].idiv_layer = 4;
stack_layer[4].chikness = 100.0e-6;
stack_layer[4].lambda = 370.0; // SiC
stack_layer[4].idiv_layer = 7;
stack_layer[5].chikness = 5.0e-6;
stack_layer[5].lambda = 130.0;// 130.0; // GaN
stack_layer[5].idiv_layer = 1;
stack_layer[6].chikness = 5.0e-6;
stack_layer[6].lambda = 317.0;// 317.0; // gold
stack_layer[6].idiv_layer = 1;
stack_layer[7].chikness = 250.0e-6;
stack_layer[7].lambda = 1.0;// 0.026; // air
stack_layer[7].idiv_layer = 7;
```

Алгоритм решения задачи с помощью fft.

1. Алгоритм fast Fourier transform (fft)

```
#include <iostream>
#include <iomanip>
#include <complex>
#include <cmath>
typedef std::complex<double> base;

void fft(base* &a, int n, bool invert) {
```

```
	if (n == 1) return;

	base* a0 = new base[n / 2];
	base* a1 = new base[n / 2];

	for (int i = 0, j = 0; i < n; i += 2, ++j) {
		a0[j] = a[i];
		a1[j] = a[i + 1];
	}
	fft(a0, n/2, invert);
	fft(a1, n/2, invert);

	base w  (1,0);
	double ang = 2.0*M_PI / n * (invert ? -1 : 1);
	base wn(cos(ang), sin(ang));
	for (int i = 0; i < n / 2; ++i) {
		a[i] = a0[i] + w * a1[i];
		a[i + n / 2] = a0[i] - w * a1[i];

		if (invert) {
			a[i] /= 2;
			a[i + n / 2] /= 2;
		}

		w *= wn;
	}

	delete[] a0;
	delete[] a1;
}

void DFTx(Real***& u, Real***& a, int m, int n, int l) {

	int Nx = m + 1; // < M+2
	int Ny = n + 1; // < N+2
	int Nz = l + 1;

	// Находим искомую функцию.

#pragma omp parallel for
	for (int k = 1; k < Nz - 1; k++) {

		base* a1 = new base[Nx];

		for (int j = 0; j <= Ny - 1; j++) {

			a1[0]= (0.0, 0.0);
			for (int i = 1; i < Nx - 1; i++) {
				a1[i-1] = (0.0,a[i][j][k]);
			}
			fft(a1, Nx-2,false);
			for (int i = 1; i < Nx - 1; i++) {
				u[i][j][k] = -a1[i-1]._Val[0];
			}
		}

		delete[] a1;
	}

}

void DFTy(Real***& u, Real***& a, int m, int n, int l) {

	int Nx = m + 1; // < M+2
```

```
	int Ny = n + 1; // < N+2
	int Nz = l + 1;

	// Находим искомую функцию.
#pragma omp parallel for
	for (int k = 1; k < Nz - 1; k++) {

		base* a1 = new base[Ny];

		for (int i = 0; i <= Nx - 1; i++) {

			for (int j = 1; j < Ny - 1; j++) {
				a1[j-1] = (0.0,a[i][j][k]);
			}
			fft(a1, Ny-2, false);
			for (int j = 1; j < Ny - 1; j++) {
				u[i][j][k] = -a1[j-1]._Val[0];
			}
		}

		delete[] a1;
	}
}


void IDFTx(Real***& u, Real***& f, int m, int n, int l) {

	int Nx = m + 1; // < M+2
	int Ny = n + 1; // < N+2
	int Nz = l + 1;

	// Преобразование правой части.
#pragma omp parallel for
	for (int k = 1; k < Nz - 1; k++) {


		base* a1 = new base[Nx];

		for (int j = 0; j <= Ny - 1; j++) {

			for (int i = 1; i < Nx - 1; i++) {

				a1[i-1] = (0.0,f[i][j][k]);
			}

			fft(a1, Nx-2,true);
			for (int i = 1; i < Nx - 1; i++) {

				u[i][j][k] = a1[i-1]._Val[0];
			}
		}

		delete[] a1;
	}
}

void IDFTy(Real***& u, Real***& f, int m, int n, int l) {

	int Nx = m + 1; // < M+2
	int Ny = n + 1; // < N+2
	int Nz = l + 1;

	// Преобразование правой части.
#pragma omp parallel for
```

```
	for (int k = 1; k < Nz - 1; k++) {


		base* a1 = new base[Ny];

		for (int i = 0; i <= Nx - 1; i++) {

			for (int j = 1; j < Ny - 1; j++) {
				a1[j-1] = (0.0,f[i][j][k]);
			}
			fft(a1, Ny-2, true);
			for (int j = 1; j < Ny - 1; j++) {
				u[i][j][k] = a1[j-1]._Val[0];
			}
		}

		delete[] a1;
	}
}
```

2. Код реализации Poisson solver.

```
1.  	alloc_u3D(source, M, N, L);
2.  	alloc_u3D(b, M, N, L);
3.  
4.  	// инициализация.
5.  	init_u3D(b, M, N, L, 0.0);
6.  	init_u3D(source, M, N, L, 0.0);
7.  
8.  
9.  
10. 	// Задаём тепловую мощность тепловыделения.
11. 	//for (int i = M / 2 - 117; i < M / 2 + 124; i += 26) {// ПТБШ 1,25мм 2s
    204Mb
12. 	//for (int i = M / 2 - 234; i < M / 2 + 240; i += 26) {// ПТБШ 2,5мм 2s
    204Mb
13. 	//for (int i = M / 2 - 468; i < M / 2 + 480; i += 26) { // ПТБШ 5мм 2s
    204Mb
14. 		//for (int i = M / 2 - 936; i < M / 2 + 946; i += 26) { // ПТБШ 10мм
    4s 420Mb
15. 	for (int i = M / 2 - 1872; i < M / 2 + 1888; i += 26) { // ПТБШ 20мм 4s
    420Mb
16. 		//for (int j = N / 2 - 2; j < N / 2 + 3; j++) {
17. 		for (int j = N / 2 - 5; j < N / 2 + 5; j++) {
18. 			//if ((i - (M / 2 - 117)) % 26 == 0)
19. 			{
20. 
21. 
22. 				//source[i][j][izstop] = 0.675 / (5*h1*h2*hz[izstop-
    1]);
23. 				double Ssource = 2.0 * 125e-12;
24. 				double Vol_source = Ssource * hz[izstop - 2];
25. 				source[i + 1][j][izstop - 1] = source[i][j][izstop - 1]
    = 0.675 / (Vol_source);
26. 			}
27. 		}
28. 	}
29. 
30. 	IDFTx(b, source, M, N, L);
31. 
32. #pragma omp parallel for
33. 	for (int i = 0; i < M + 2; i++) {
34. 		for (int j = 0; j < N + 2; j++) {
35. 			for (int k = 0; k < L + 2; k++) {
36. 
```

```
                        source[i][j][k] = b[i][j][k];
                    }
                }
        }

        IDFTy(b, source, M, N, L);

        std::cout << "1" << std::endl;

        free_u3D(source, M, N, L);
        alloc_u3D(a, M, N, L);
        init_u3D(a, M, N, L, 0.0);

        // Ядро солвера.
#pragma omp parallel for
        for (int i = 1; i < Nx - 1; i++) {
                for (int j = 1; j < Ny - 1; j++) {

                        double* P = new double[Nz + 1];
                        double* Q = new double[Nz + 1];

                        double Rj1 = (2.0 * lambda[0] * lambda[1] / (lambda[0] + lambda[1])) * (2.0 / (hz[1] * (hz[1] + hz[0])));
                        double Rjm1 = (2.0 * lambda[0] * lambda[1] / (lambda[0] + lambda[1])) * (2.0 / (hz[0] * (hz[1] + hz[0])));

                        // Для фиксированного k решение системы  с трёх диагональной матрицей:
                        double b1 = Rj1;
                        double a1 = 2.0 * lambda[1] / (h1 * h1) + 2.0 * lambda[1] / (h2 * h2) + (Rj1 + Rjm1) - 2.0 * cos(2.0*M_PI * i / Nx) * lambda[1] / (h1 * h1) - 2.0 * cos(2.0*M_PI * j / Ny) * lambda[1] / (h2 * h2);
                        P[1] = b1 / a1;
                        double d1 = -b[i][j][1];
                        Q[1] = d1 / a1;

                        for (int k = 2; k <= Nz; k++) {

                                Real lmax = lambda[k];
                                if (k != Nz) {
                                        lmax = lambda[k + 1];
                                }
                                double Rj = (2.0 * lambda[k] * lmax / (lambda[k] + lmax)) * (2.0 / (hz[k] * (hz[k] + hz[k - 1])));
                                double Rjm1 = (2.0 * lambda[k - 1] * lambda[k] / (lambda[k - 1] + lambda[k])) * (2.0 / (hz[k - 1] * (hz[k] + hz[k - 1])));

                                // bk -> k+1
                                // ck -> k-1
                                double bk = Rj;
                                double ck = Rjm1;
                                double ak = (2.0 / (h1 * h1) + 2.0 / (h2 * h2)) * lambda[k] + (Rj + Rjm1) - 2.0 * cos(2.0*M_PI * i / Nx) * lambda[k] / (h1 * h1) - 2.0 * cos(2.0*M_PI * j / Ny) * lambda[k] / (h2 * h2);
                                double dk = -b[i][j][k];

                                P[k] = bk / (ak - ck * P[k - 1]);
                                Q[k] = (dk + ck * Q[k - 1]) / (ak - ck * P[k - 1]);
                        }
                        a[i][j][Nz] = Q[Nz];

                        for (int k = Nz - 1; k >= 1; k--) {
                                a[i][j][k] = P[k] * a[i][j][k + 1] + Q[k];
```

```
                }

                delete[] P;
                delete[] Q;
            }
        }

        std::cout << "2" << std::endl;

            free_u3D(b, M, N, L);
            alloc_u3D(u, M, N, L);
            init_u3D(u, M, N, L, 0.0);

            DFTx(u, a, M, N, L);

        #pragma omp parallel for
            for (int i = 0; i < M + 2; i++) {
                for (int j = 0; j < N + 2; j++) {
                    for (int k = 0; k < L + 2; k++) {
                        a[i][j][k] = -u[i][j][k];
                    }
                }
            }

            DFTy(u, a, M, N, L);

        #pragma omp parallel for
            for (int i = 0; i < M + 2; i++) {
                for (int j = 0; j < N + 2; j++) {
                    for (int k = 0; k < L + 2; k++) {

                        u[i][j][k] += Tamb;
                    }
                }
            }

            std::cout << "3" << std::endl;

            free_u3D(a, M, N, L);

            double tmax = -1.0e30;

            for (int i = 0; i < M + 2; i++) {
                for (int j = 0; j < N + 2; j++) {
                    if (u[i][j][izstop - 1] > tmax) {
                        tmax = u[i][j][izstop - 1];
                    }
                }
            }

            printf("Maximum temperature %e\n", tmax);


            // экспорт 3D полевой величины u в программу tecplot 360.
            //exporttecplot3D(u, xf, yf, zf, M, N, izstop - 1);
            //exporttecplot3D_fft(u, xf, yf, zf, M, N, izstop - 1);
            exporttecplot3D22D(u, xf, yf, zf, M, N, izstop - 1);


            free_u3D(u, M, N, L);

            delete[] xf;
            delete[] yf;
            delete[] zf;
```

Результаты.

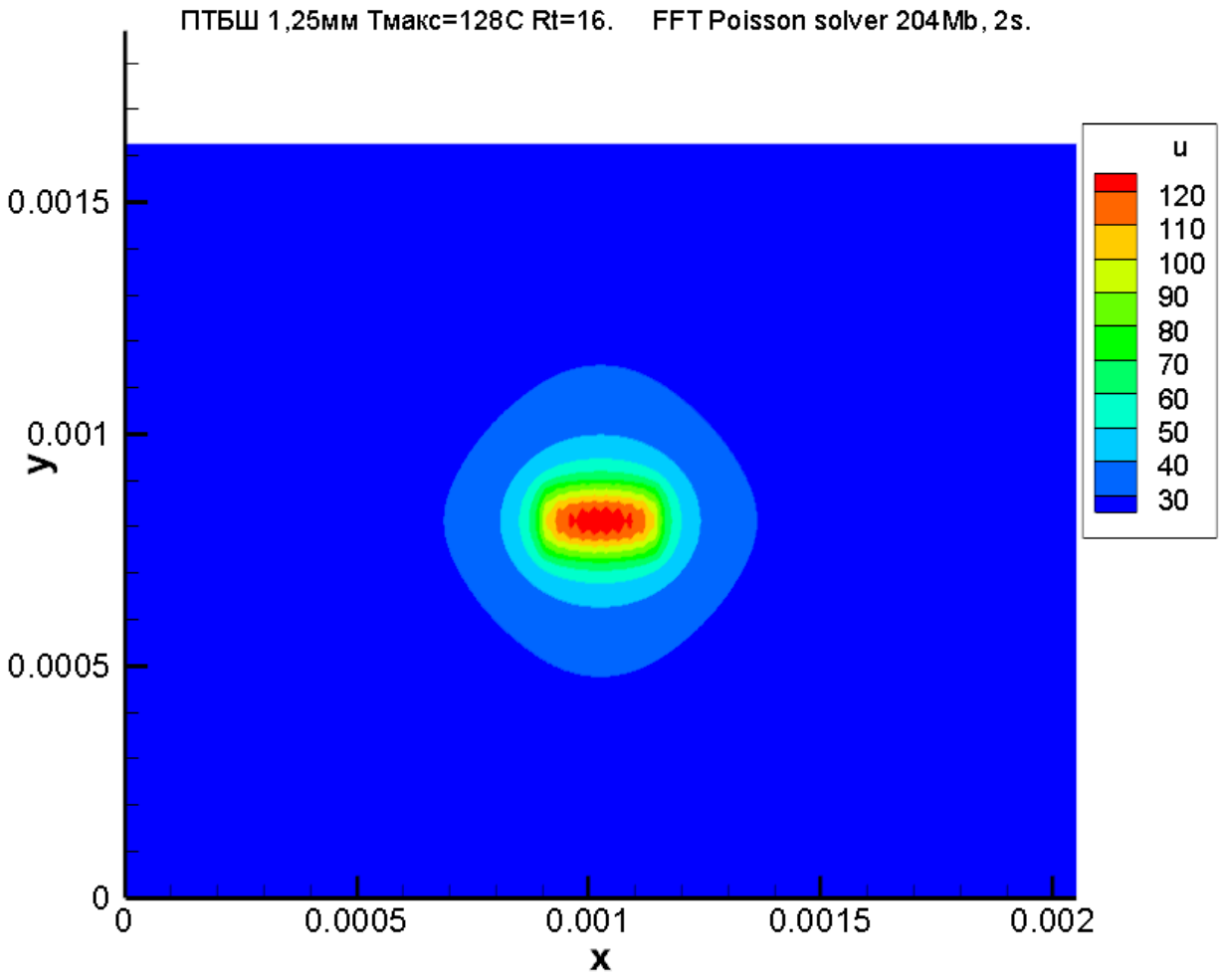


Рисунок 1 – Температурное поле на поверхности ПТБШ с Шз=1,25мм при тепловой мощности 6.75Вт.

Время расчета теплового сопротивления 2s – 2*intel xeon 2630 v4. (40потоков 2.2ГГц).

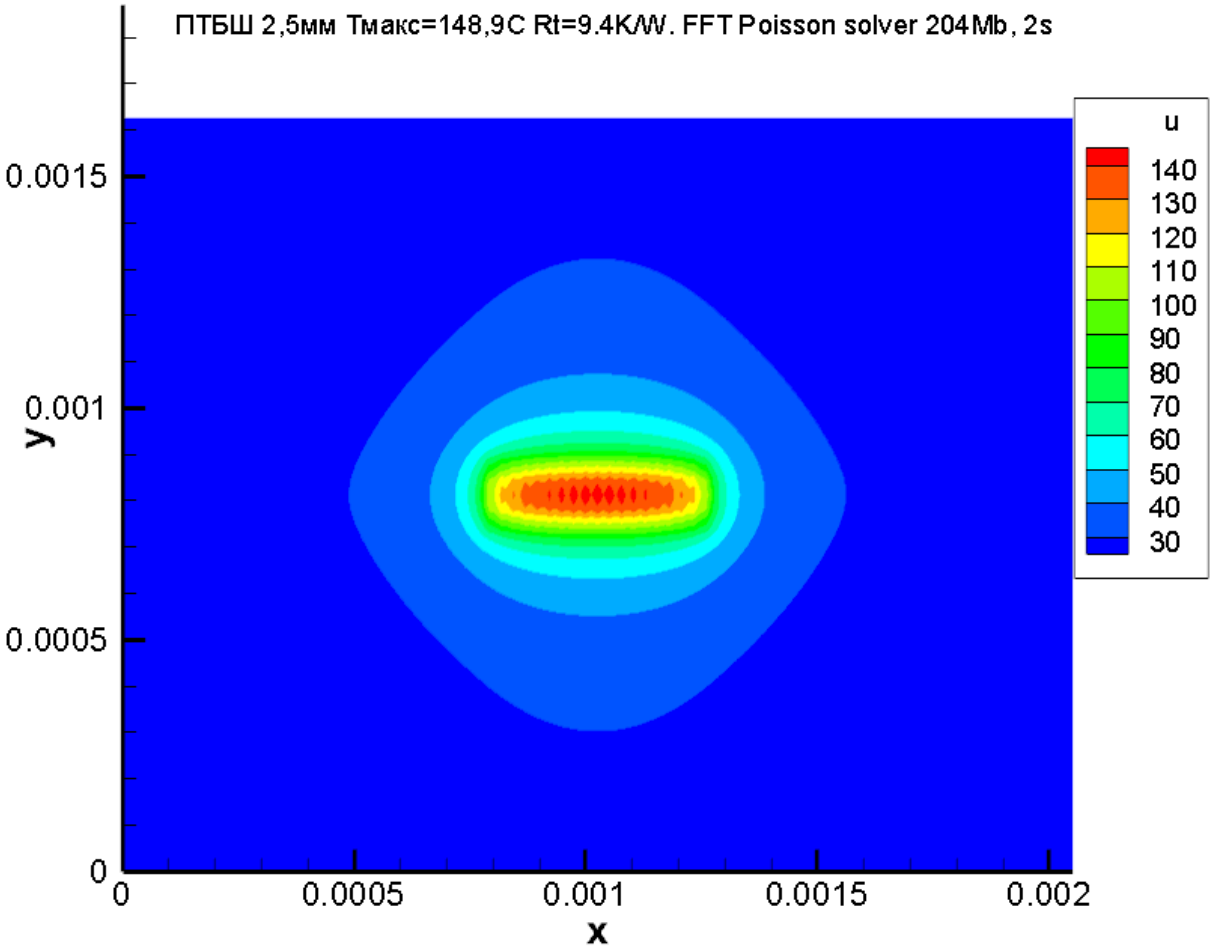


Рисунок 2 – Температурное поле на поверхности ПТБШ с Шз=2,5мм при тепловой мощности 13,5Вт.

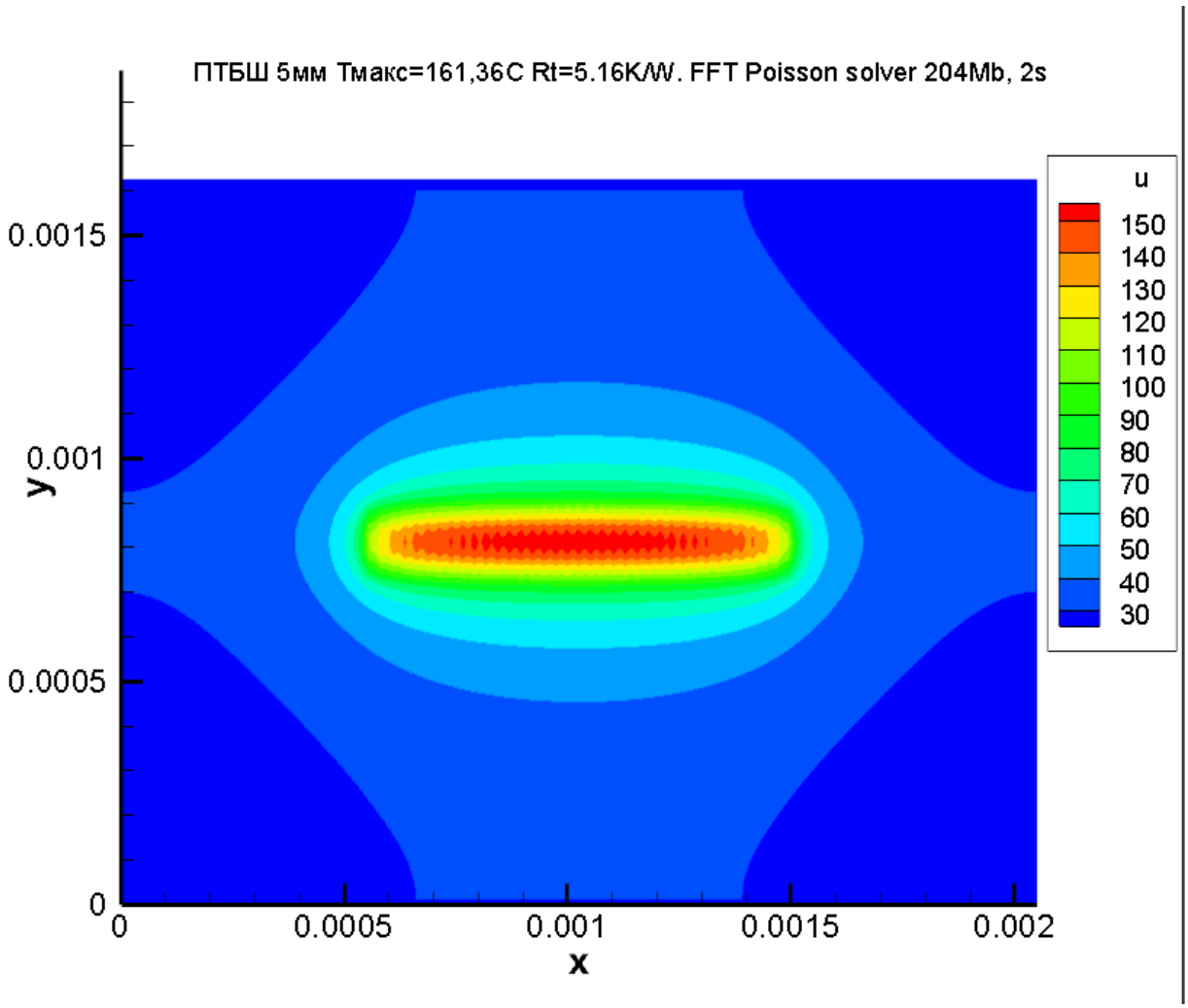


Рисунок 3 – Температурное поле на поверхности ПТБШ с Шз=5мм при тепловой мощности 27Вт.

```
lengthx = 4098.0e-6;// 2050.0e-6;
lengthy = 1625.0e-6;

M = 4097; // 2049;//511;
N = 129;// 511;
```

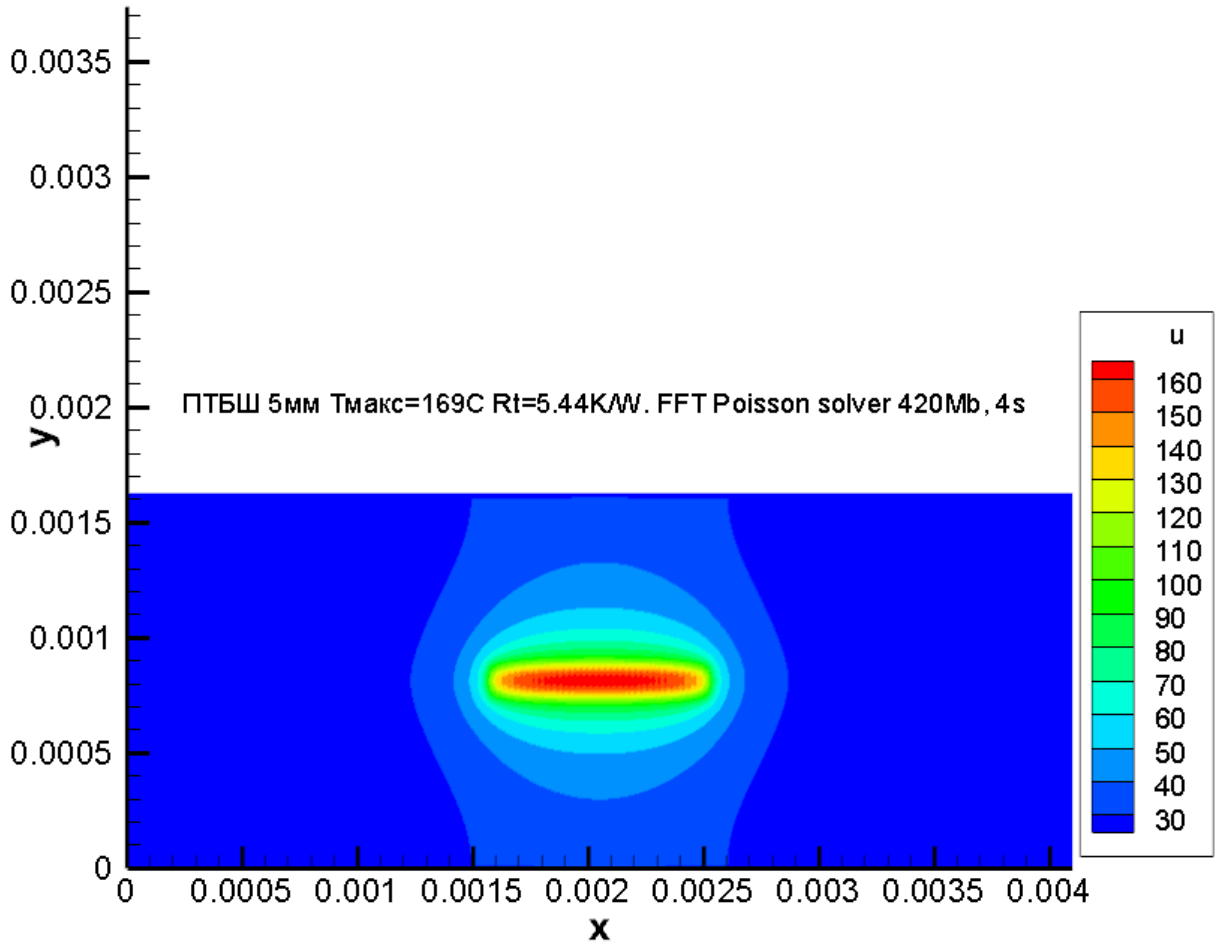


Рисунок 4 – Температурное поле на поверхности ПТБШ с Шз=5мм при тепловой мощности 27Вт.

Время расчета теплового сопротивления 4s – 2*intel xeon 2630 v4. (40потоков 2.2ГГц).

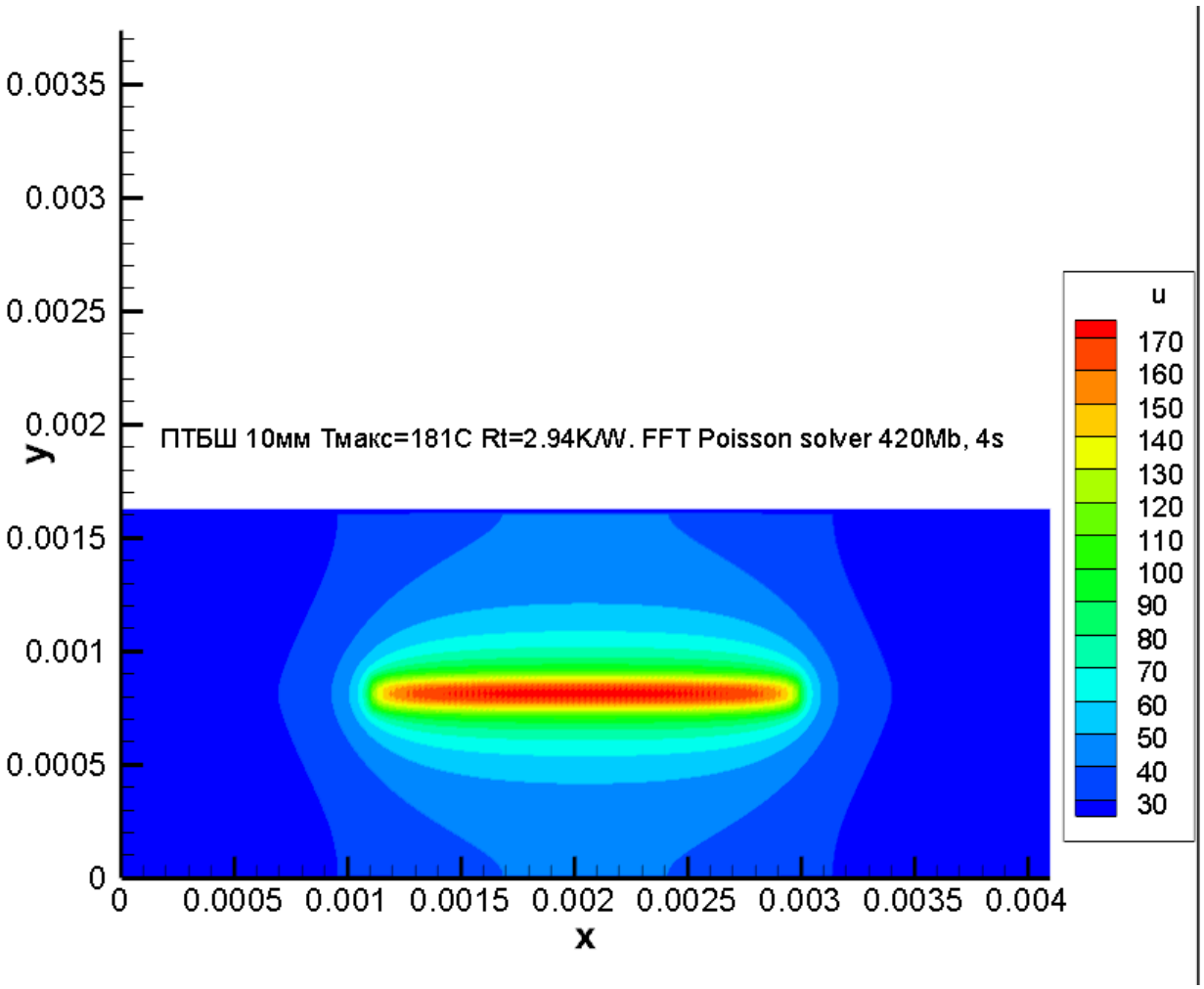


Рисунок 5 – Температурное поле на поверхности ПТБШ с Шз=10мм при тепловой мощности 54Вт.

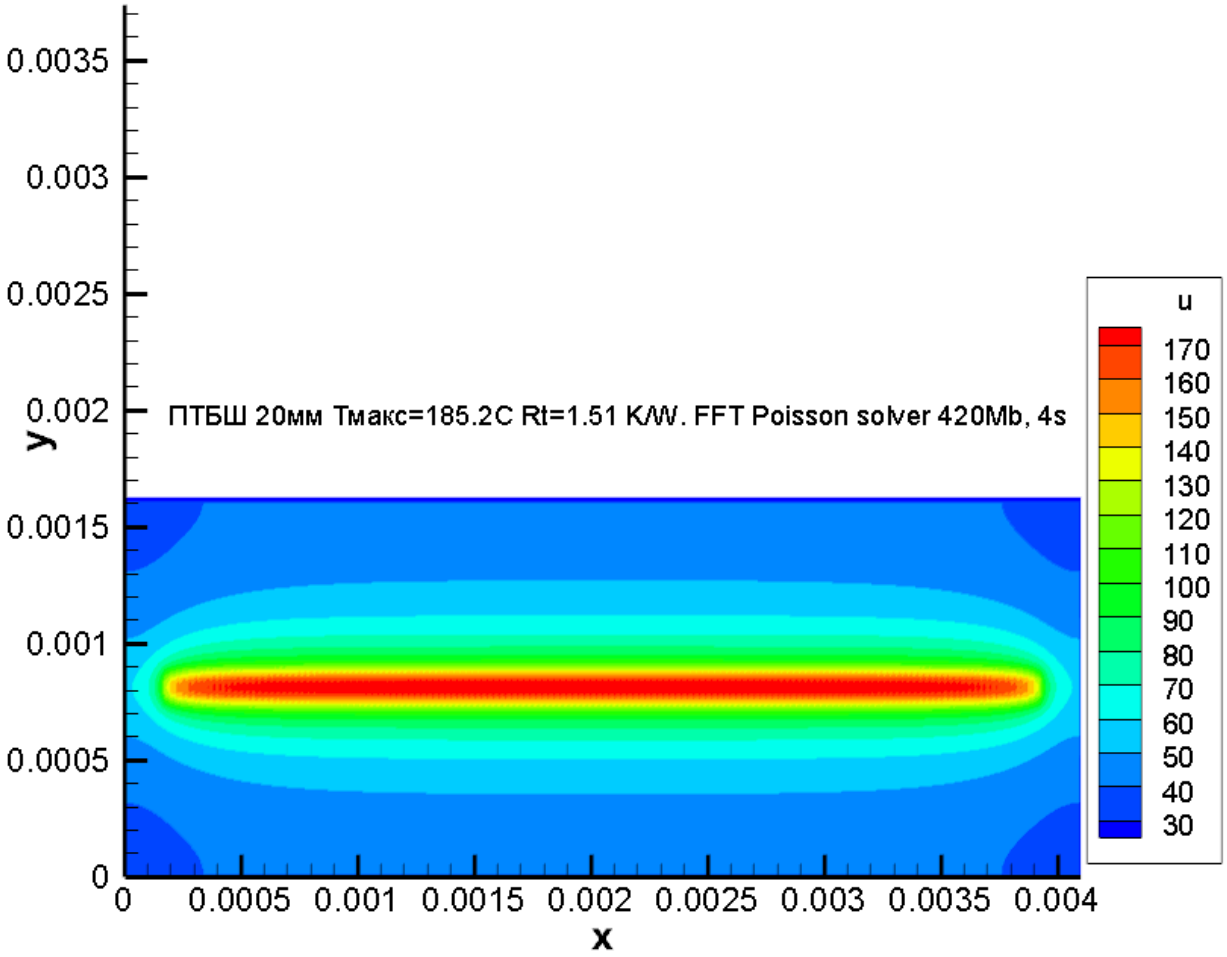
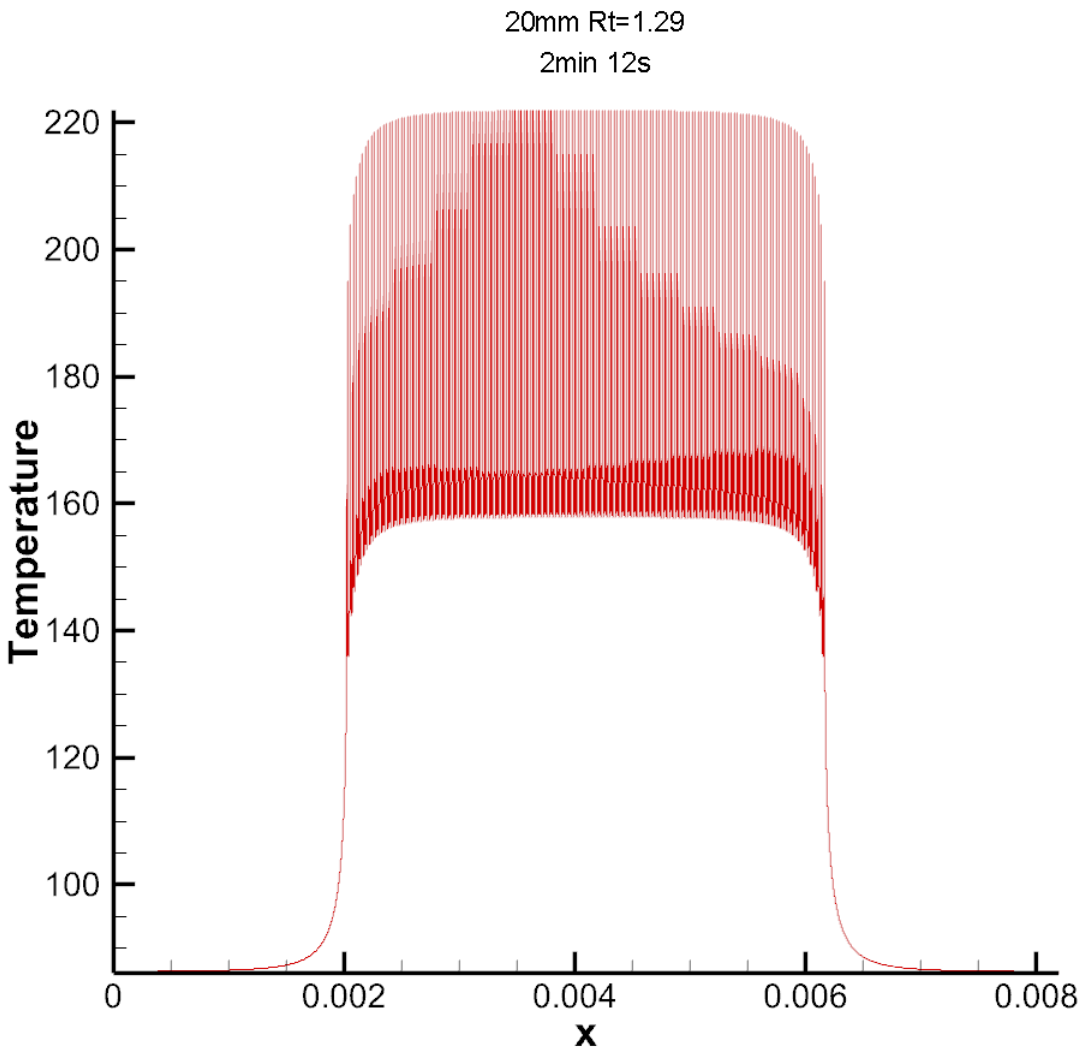
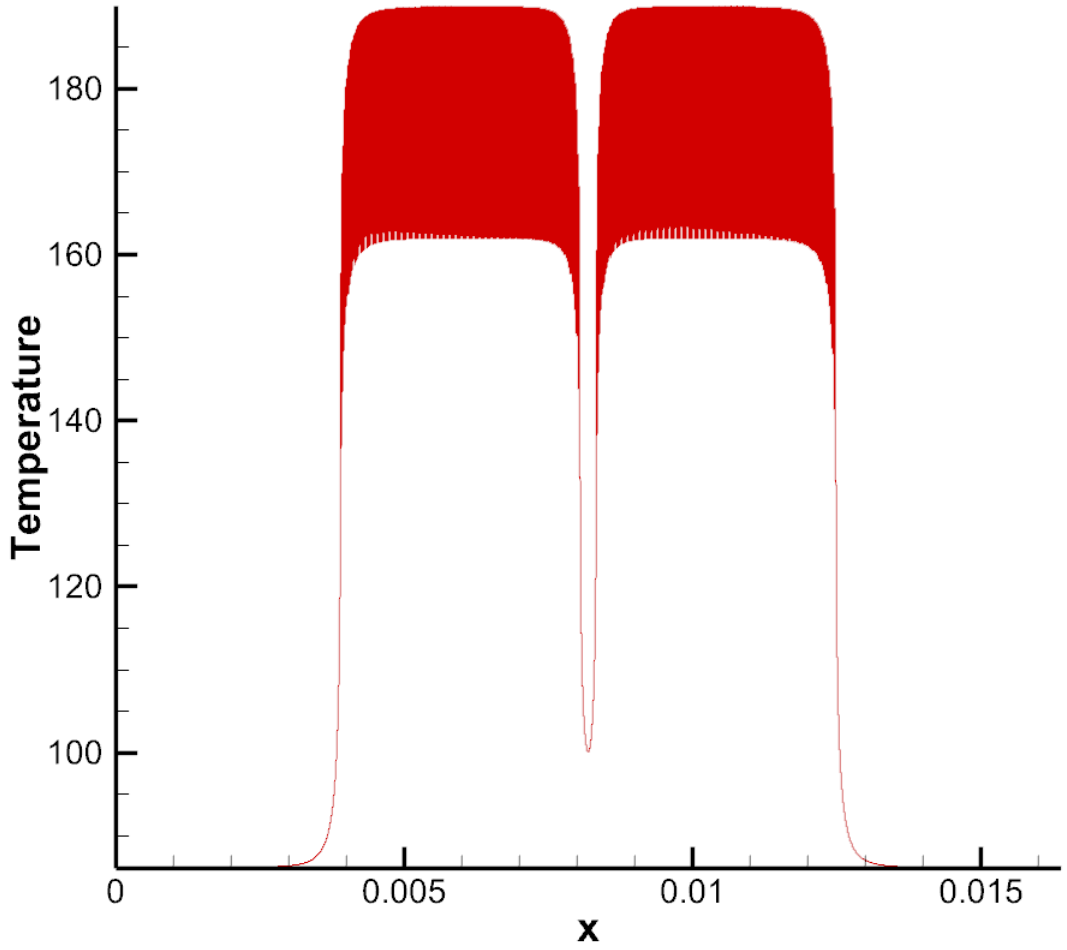
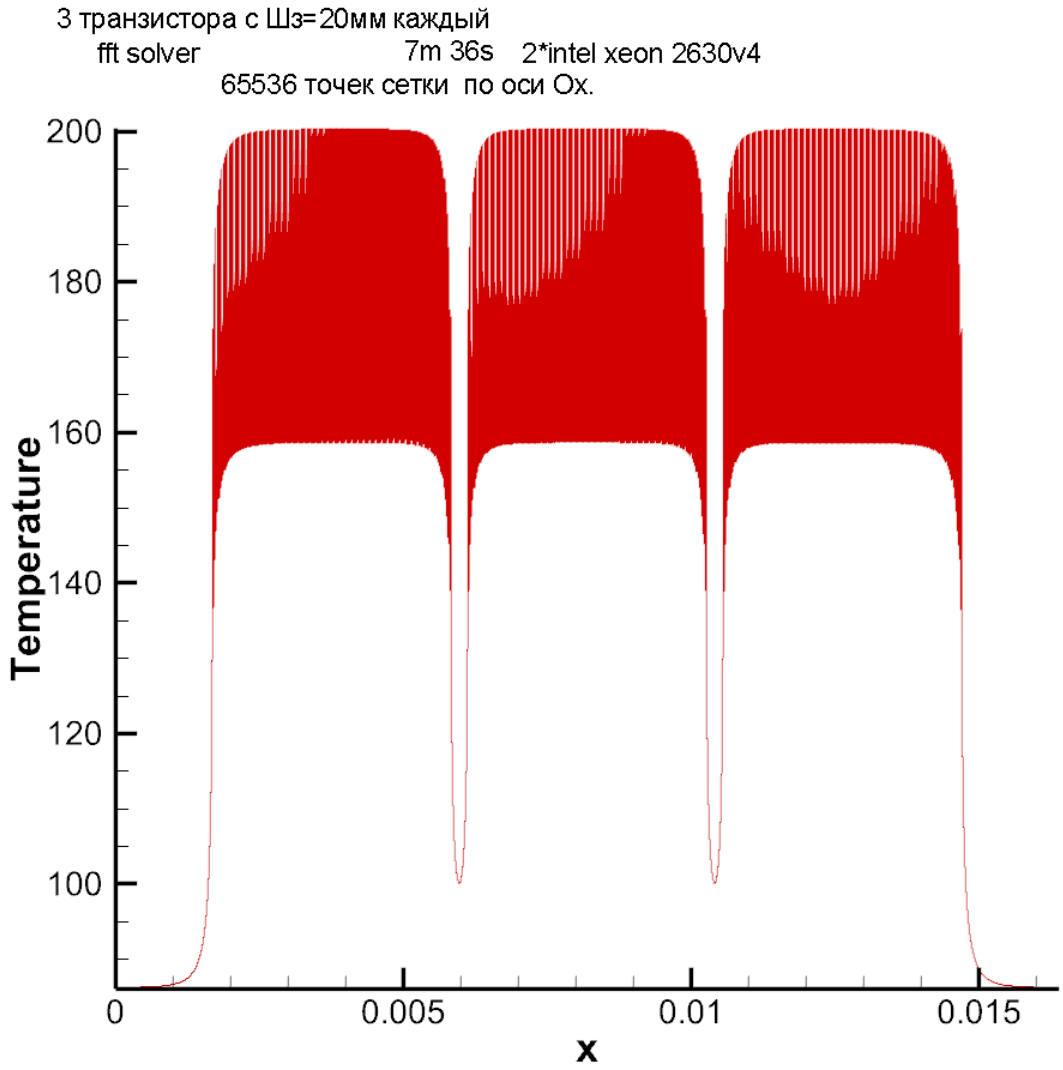


Рисунок 6 – Температурное поле на поверхности ПТБШ с Шз=20мм при тепловой мощности 108Вт.


20mm Rt=1.29
2min 12s
Temperature
220
200
180
160
140
120
100
0
0.002
0.004
0.006
0.008
x

5min 38s





Выводы

Быстрая штука получилась и оперативной памяти совсем мало ест.