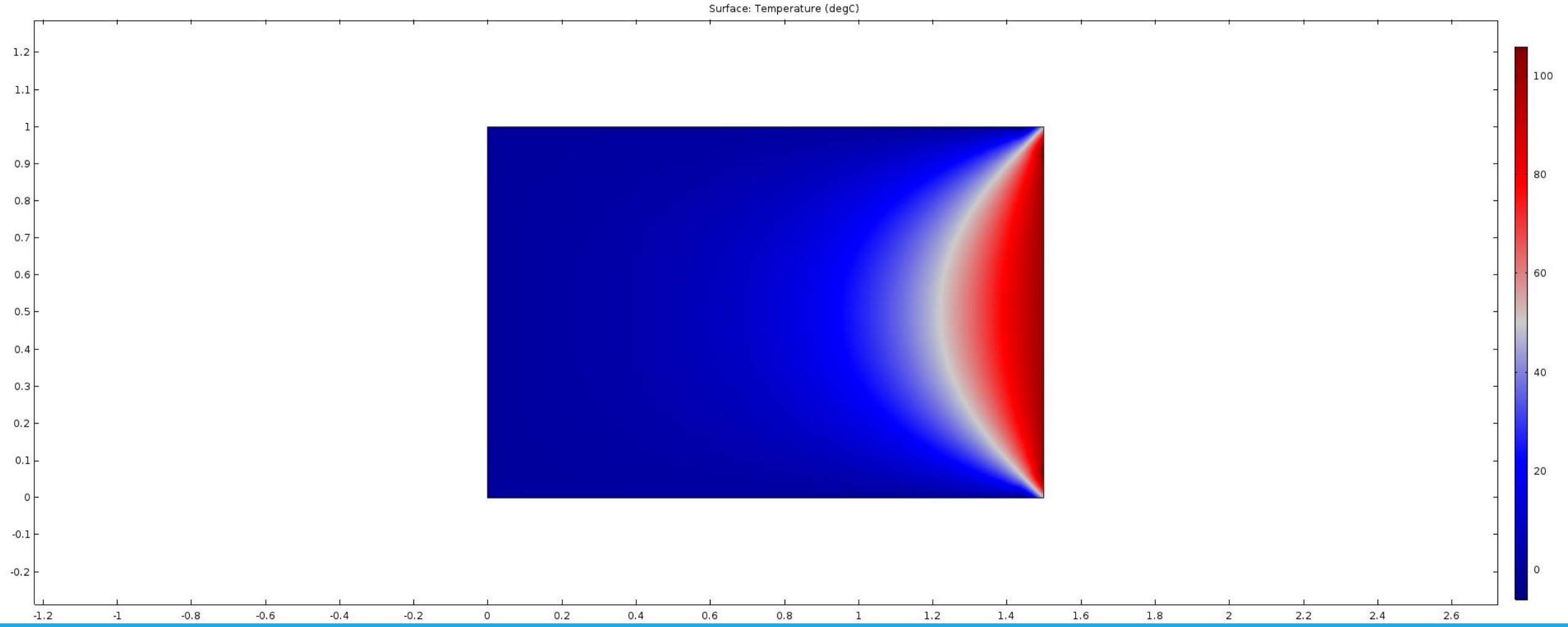


Jacobi and Gauss Seidel 1D Example Codes

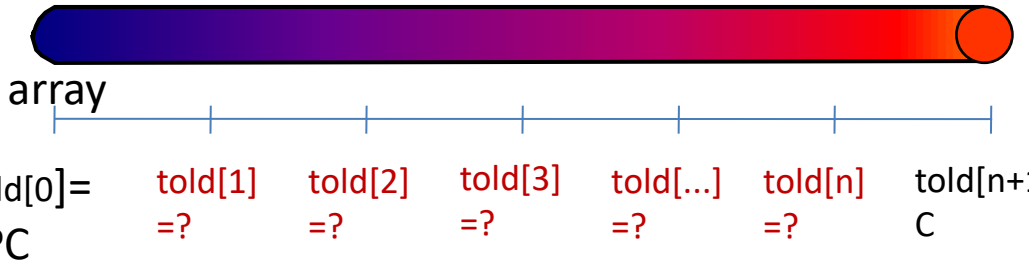
Jacobi Example: 1D hot-cold rod SERIAL

```
#include <stdio.h>
#include <math.h>

int max_size=1000001;

int main(int argc, char *argv[])
{
double told[max_size], t[max_size], tol, diff, difmax;
int i, iter, n;
printf("enter the problem size (n) and the convergence
      tolerance\n");
scanf("%d %lf", &n, &tol);

// initialise temperature array
for (i=1; i <= n; i++) {
  told[i] = 20.0;
  }
// fix end points as cold and hot
told[0] = 0.0;
told[n+1] = 100.0;
```

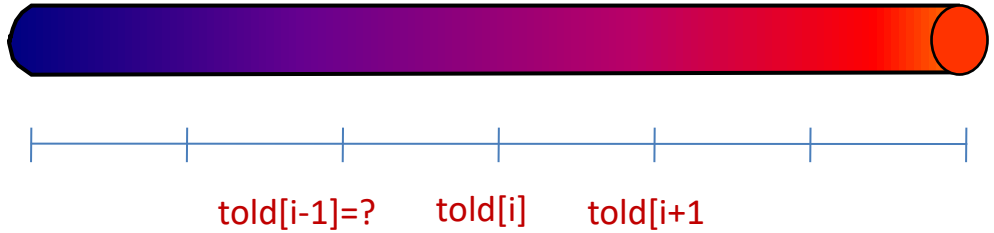


Jacobi Example: 1D hot-cold rod SERIAL

```
iter = 0;

difmax = 1000000.0;
        while (difmax > tol) {
iter=iter+1;
// update temperature for next iteration
for (i=1; i <= n; i++) {
  t[i] = (told[i-1]+told[i+1])/2.0;
  }
// work out maximum difference between old and new temperatures
difmax=0.0;
for (i=1; i <= n; i++) {
  diff = fabs(t[i]-told[i]);
  if (diff > difmax) {
    difmax = diff;
    }
  told[i] = t[i];
  }
       }//while (difmax>tol)
 for (i=1; i <= n; i++) {
    printf("told[%d] = %-5.7lf  \n", i, t[i]);
    }
  printf("iterations = %d  maximum difference = %-5.7lf  \n", iter, difmax)
}
```

using a simple average of two temperatures from neighbouring points.



redefine temperatures from new (t) to old (told) in preparation for the next iteration of the solution (JACOBI algorithm)

Jacobi Example: 1D hot-cold rod OMP v1

```
#include <omp.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

int max_size=400000;

int main(int argc, char *argv[]) {
   double told[max_size], t[max_size], tol, diff, difmax, priv_difmax;
   double tstart, tstop;
   int i, iter, n, nthreads;
   printf("enter the problem size (n) and the convergence tolerance\n");
   scanf("%d %lf", &n, &tol);
   printf("Enter the number of threads (max 10) ");
   scanf("%d",&nthreads);

   /* define the number of threads to be used */
   omp_set_num_threads(nthreads);
```

Jacobi Example: 1D hot-cold rod OMP v1

```
tstart = omp_get_wtime ();
    // initialise temperature array
   #pragma omp parallel for schedule(static) \
      default(shared) private(i)
   for (i=1; i <= n; i++) {
      told[i] = 20.0;
   }
  // fix end points as cold and hot
   told[0] = 0.0;
   told[n+1] = 100.0;
   iter = 0;
   difmax = 1000000.0;
   while (difmax > tol) {
      iter=iter+1;
      // update temperature for next iteration
      #pragma omp parallel for schedule(static) \
         default(shared) private(i)
      for (i=1; i <= n; i++) {
         t[i] = (told[i-1]+told[i+1])/2.0;
      }
```

loop iterations distributed amongst available threads

loop iterations distributed amongst available threads

Jacobi Example: 1D hot-cold rod OMP v1

```
// work out maximum difference between old and new temperatures
      difmax = 0.0;
     #pragma omp parallel default(shared) private(i, diff, priv_difmax)
     {
        priv_difmax = 0.0;
        #pragma omp for schedule(static)
        for (i=1; i <= n; i++) {
           diff = fabs(t[i]-told[i]);
           if (diff > priv_difmax) {
              priv_difmax = diff;
           }
           told[i] = t[i];
        }
        #pragma omp critical
        if (priv_difmax > difmax) {
           difmax = priv_difmax;
        }
     }
    }//while (difmax>tol)
```

create parallel region with more than one directive being used here. NOTE priv_difmax variable

not a standard reduction operation so need to use less elegant (but practical) solution using local private difmax for each thread

critical region ensures only one local priv_difmax is compared at a time in order to create the global difmax value

Jacobi Example: 1D hot-cold rod OMP v1

```
    tstop = omp_get_wtime ();

    for (i=1; i <= n; i++) {
        printf("told[%d] = %-5.7lf  \n", i, t[i]);
    }
    printf("iterations = %d  maximum difference = %-5.7lf  \n",
        iter, difmax);


    printf("time taken is %4.3lf\n", (tstop-tstart));
}
```

Example – 1D hot-cold rod OMP v1

maximum difference tolerance = 0.001
Machine : cms-grid-03
Compile: -fopenmp -O3

Grid points (n) = 100000 serial time :151.27		
# of threads	Wall clock time (s)	Speedup
1	147.62	-
2	106.33	1.388
4	87.09	1.695
8	96.83	1.525

Grid points (n) = 1000000 serial time :5399.53		
# of threads	Wall clock time (s)	Speedup
1	5256.73	-
2	2997.04	1.754
4	1425.78	3.687
8	893.36	5.884

Gauss Seidel Example: 1D hot-cold rod SERIAL

```
#include <stdio.h>
#include <math.h>

int max_size=100000;

int main(int argc, char *argv[])
{
double t[max_size], told[max_size], tol, diff, difmax;
int i, iter, n;
printf("enter the problem size (n) and the convergence
       tolerance\n");
scanf("%d %lf", &n, &tol);

// initialise temperature array
for (i=1; i <= n; i++) {
  t[i] = 20.0;
  }
// fix end points as cold and hot
t[0] = 0.0;
t[n+1] = 100.0;
```

Gauss Seidel Example: 1D hot-cold rod SERIAL

```
iter = 0;
difmax = 1000000.0;
while (difmax > tol) {
iter=iter+1;
//populate tnew array to keep old values for calculating residual
for (i=1;i <=n ; i++ )
{
        told[i]= t[i];
}
// update temperature for next iteration
for (i=1; i <= n; i++) {
  t[i] = (t[i-1]+t[i+1])/2.0;
  }
// work out maximum difference between old and new temperatures
difmax=0.0;
for (i=1; i <= n; i++) {
  diff = fabs(t[i]-told[i]);
  if (diff > difmax) {
    difmax = diff;
    }
  }
}//while (difmax > tol)
  for (i=1; i <= n; i++) {
    printf("t[%d] = %-5.7lf  \n", i, told[i]);
    }
  printf("iterations = %d  maximum difference = %-5.7lf  \n", iter, difmax);
}
```

redefine temperatures from new (t) to old (told) in preparation for calculating the residual

using a simple average of two temperatures from neighbouring points.