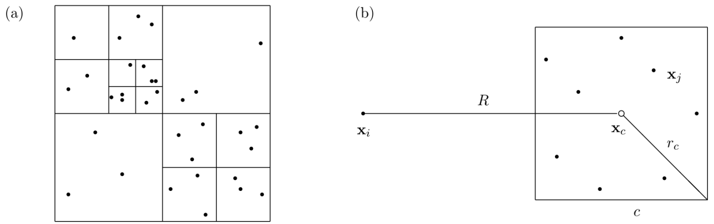
Parallel Implementation of a Kernel-Independent Barycentric Lagrange Treecode Algorithm

Md Abu Talha

Southern Methodist University

May 1, 2025

Electrostatic Potential and Kernel



- The **electrostatic potential** due to a set of charged particles,

$$\phi(x_i) = \sum_{j=1}^{N} K(x_i, y_j) q_j, \quad i = 1, ..., N$$

  Where $K(x_i, y_j)$ is the interaction between a target particle x_i and a source particle y_j with charge q_j.
- The kernel $K(x, y)$ is the **screened Coulomb** (Yukawa) potential:

$$K(x, y) = \frac{e^{-\kappa|x-y|}}{|x-y|}$$

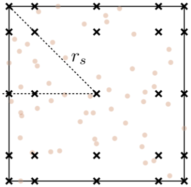
Barycentric Lagrange Interpolation

Polynomial Interpolation for N-Body Acceleration:

$$p_n(x) = \sum_{k=0}^{n} f(s_k) L_k(x), \quad L_k(x) = \left(\frac{\omega_k}{x - s_k}\right) \Bigg/ \left(\sum_{k'=0}^{n} \frac{\omega_{k'}}{x - s_{k'}}\right)$$

where weights are given by:

$$\omega_k = \frac{1}{\sum\limits_{\substack{j=0 \\ j \neq k}}^{n} (s_k - s_j)}$$



Chebyshev Interpolation Points: For the interval $[-1, 1]$,

$$s_k = \cos\theta_k, \quad \theta_k = \frac{\pi k}{n}, \quad k = 0, ..., n$$

with weights:

$$\omega_k = (-1)^k \delta_k, \quad \delta_k = \begin{cases} 1/2, & k = 0 \text{ or } n \\ 1, & \text{otherwise} \end{cases}$$

Kernel-Independent Treecode

- The kernel $K(x, y)$ is approximated using Chebyshev points $s_k = (s_{k1}, s_{k2}, s_{k3})$ then,

$$K(x, y) \approx \sum_k K(x, s_k) L_{k1}(y_1) L_{k2}(y_2) L_{k3}(y_3), \quad k_l = 0, \dots, n$$

- In a treecode, particles $\{y_j\}$ are clustered hierarchically. The potential at x_i due to a **particle-cluster** interaction is:

$$\phi(x_i, C) = \sum_{y_j \in C} K(x_i, y_j) q_j$$

Using kernel interpolation, this simplifies to:

$$\phi(x_i, C) \approx \sum_k K(x_i, s_k) \hat{q}_k$$

where the modified weights are:

$$\hat{q}_k = \sum_{y_j \in C} L_{k1}(y_1) L_{k2}(y_2) L_{k3}(y_3) q_j$$

Algorithm

Algorithm 1 computation of modified weights in (4.4)

```
1: input: source particles and weights for a given cluster, $\mathbf{y}_j, f_j$
2: input: Chebyshev points mapped to the cluster, $s_{\mathbf{k}}$
3: output: modified weights, $\widehat{f}_{\mathbf{k}}$
4: initialize all $\widehat{f}_{\mathbf{k}} = \widehat{f}(k_1,k_2,k_3) = 0$
5: % loop over source particles
6: for $j = 1:N_c$
7:     initialize flag(1:3) = -1, sum(1:3) = 0
8: % loop over coordinate indices and Chebyshev points to compute $a_{j,\ell,k_\ell}$
9:     for $\ell = 1:3$ and $k_\ell = 0:n$
10:        if $|y_{j\ell} - s_{k_\ell}| \le$ DBL_MIN, $\text{flag}(\ell) = k_\ell$
11:            else $a(j,\ell,k_\ell) = w_{k_\ell}/(y_{j\ell} - s_{k_\ell})$, $\text{sum}(\ell)$ += $a(j,\ell,k_\ell)$
12:        end if
13:    end for
14: % if a flag was set, adjust $\text{sum}(\ell)$ and $a(j,\ell,k_\ell)$ to handle removable singularity
15:    for $\ell = 1:3$
16:        if $\text{flag}(\ell) > -1$, $\text{sum}(\ell) = 1$, $a(j,\ell,0:n) = 0$, $a(j,\ell,\text{flag}(\ell)) = 1$, end if
17:    end for
18:    denom = sum(1) $\cdot$ sum(2) $\cdot$ sum(3)
19: % loop over tensor product of Chebyshev point indices as in (4.4)
20:    for $(k_1,k_2,k_3) = (0:n, 0:n, 0:n)$
21:        $\widehat{f}(k_1,k_2,k_3)$ += $(a(j,1,k_1) \cdot a(j,2,k_2) \cdot a(j,3,k_3)/\text{denom}) \cdot f_j$
22:    end for
23: end for
```

Algorithm

Algorithm 2 kernel-independent treecode

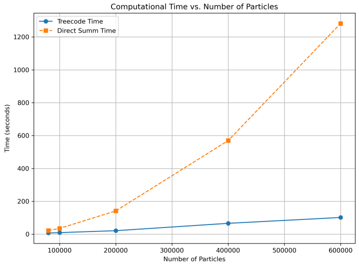
```
input: particle coordinates and weights $\mathbf{x}_i, f_i, i=1,\cdots,N$
input: treecode MAC parameter $\theta$, polynomial degree $n$, maximum leaf size $N_0$
output: particle velocities $u_i, i=1,\cdots,N$
program **main**
    build tree of particle clusters
    compute modified weights $\widehat{f}_{\mathbf{k}}$ in (4.4) for each cluster
    for $i=1,\cdots,N$, **compute_velocity**($\mathbf{x}_i$, root), end for
end program
subroutine **compute_velocity**($\mathbf{x}$, $C$)
    if MAC is satisfied
        compute particle-cluster interaction by approximation (4.3)
    else
        if $C$ is a leaf, compute particle-cluster interaction by direct sum (4.2)
    else
        for each child $C'$ of $C$, **compute_velocity**($\mathbf{x}$, $C'$), end for
end subroutine
```

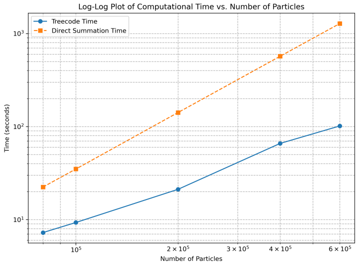
Performance Comparison: Treecode vs. Direct Summation

Table: Treecode vs Direct Summation Performance

Number of Particles	Treecode Time	Direct Sum Time	L_inf Error	L2 Error
80 K	7.26	22.40	4.42E-06	1.04E-05
100 K	9.31	34.98	3.59E-06	9.9E-06
200 K	21.16	141.52	3.79E-06	1.03E-05
400 K	65.89	569.58	3.09E-06	1.17E-05
600 K	101.59	1281.86	2.99E-06	1.27E-05

- **Computational Cost:**
  - Direct Summation: $\mathcal{O}(N^2)$
  - Treecode: $\mathcal{O}(N \log N)$

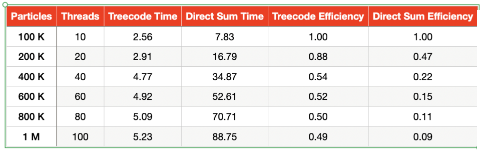




Parallel Performance Using OpenMP

Weak Scaling

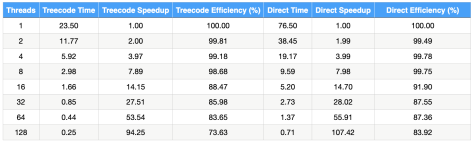
Table: Parallel Performance (Weak Scaling)



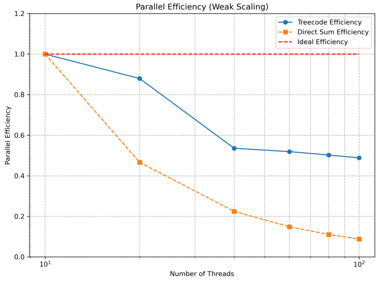
Particles	Threads	Treecode Time	Direct Sum Time	Treecode Efficiency	Direct Sum Efficiency
100 K	10	2.56	7.83	1.00	1.00
200 K	20	2.91	16.79	0.88	0.47
400 K	40	4.77	34.87	0.54	0.22
600 K	60	4.92	52.61	0.52	0.15
800 K	80	5.09	70.71	0.50	0.11
1 M	100	5.23	88.75	0.49	0.09

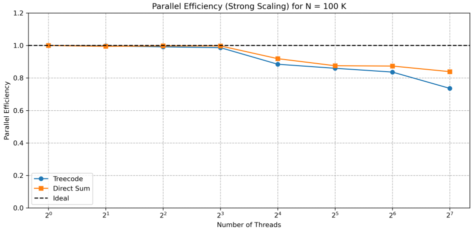
Strong Scaling

Table: Parallel Performance (Strong Scaling) for N = 100,000



Threads	Treecode Time	Treecode Speedup	Treecode Efficiency (%)	Direct Time	Direct Speedup	Direct Efficiency (%)
1	23.50	1.00	100.00	76.50	1.00	100.00
2	11.77	2.00	99.81	38.45	1.99	99.49
4	5.92	3.97	99.18	19.17	3.99	99.78
8	2.98	7.89	98.68	9.59	7.98	99.75
16	1.66	14.15	88.47	5.20	14.70	91.90
32	0.85	27.51	85.98	2.73	28.02	87.55
64	0.44	53.54	83.65	1.37	55.91	87.36
128	0.25	94.25	73.63	0.71	107.42	83.92

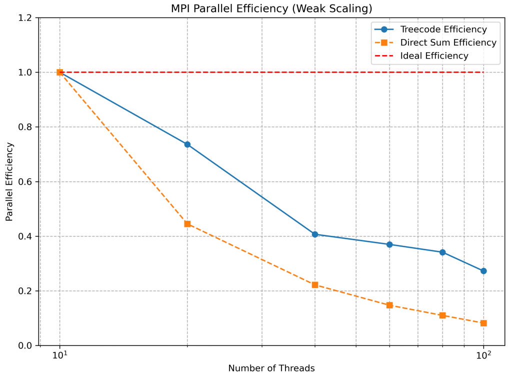




Parallel Performance Using MPI (Weak Scaling)

Table: MPI Parallel Performance (Weak Scaling)

Particles	Threads	Treecode Time (s)	Direct Sum Time (s)	Treecode Efficiency	Direct Sum Efficiency
100000	10	2.61	9.02	1.00	1.00
200000	20	3.55	20.25	0.74	0.45
400000	40	6.42	40.61	0.41	0.22
600000	60	7.06	61.19	0.37	0.15
800000	80	7.65	81.78	0.34	0.11
1000000	100	9.58	109.65	0.27	0.08



Parallel Performance Using MPI (Sequential)

100K Particles

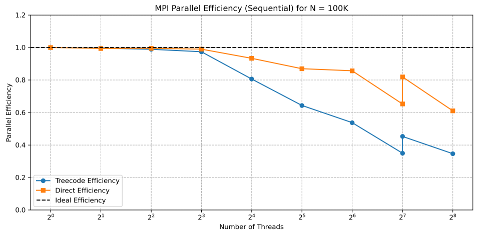
Table: MPI Parallel Performance (Sequential) for N = 100,000

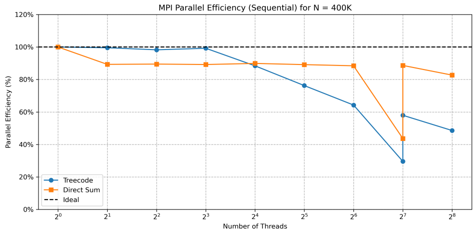
Number of Threads	Treecode Time (s)	Direct Time (s)	Treecode Speedup	Direct Speedup	Treecode Efficiency	Direct Efficiency
1	23.57	88.36	1.00	1.00	1.00	1.00
2	11.83	44.44	1.99	1.99	1.00	0.99
4	5.96	22.20	3.96	3.98	0.99	1.00
8	3.03	11.16	7.79	7.92	0.97	0.99
16	1.83	5.92	12.90	14.94	0.81	0.93
32	1.14	3.18	20.59	27.82	0.64	0.87
64	0.69	1.61	34.41	54.84	0.54	0.86
128	0.53	1.06	44.77	83.53	0.35	0.65
128	0.41	0.84	57.97	104.89	0.45	0.82
256	0.27	0.57	88.72	156.36	0.35	0.61

400K Particles

Table: MPI Parallel Performance (Sequential) for N = 400 K

Number of Processes	Treecode Time (s)	Direct Time (s)	Treecode Speedup	Direct Speedup	Treecode Efficiency	Direct Efficiency
1	186.58	1443.50	1.00	1.00	1.00	1.00
2	93.71	808.16	1.99	1.79	1.00	0.89
4	47.46	403.26	3.93	3.58	0.98	0.89
8	23.50	202.25	7.94	7.14	0.99	0.89
16	13.19	100.35	14.14	14.38	0.88	0.90
32	7.65	50.60	24.40	28.53	0.76	0.89
64	4.54	25.50	41.13	56.60	0.64	0.88
128	4.92	25.82	37.89	55.91	0.30	0.44
128 (2n)	2.51	12.72	74.20	113.48	0.58	0.89
256 (4n)	1.50	6.82	124.41	211.71	0.49	0.83

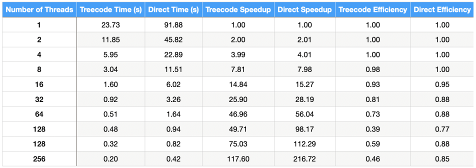




Parallel Performance Using MPI (Cyclic)

100K Particles

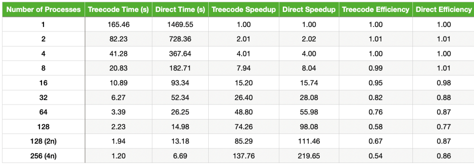
Table: MPI Parallel Performance (Cyclic) for N = 100,000



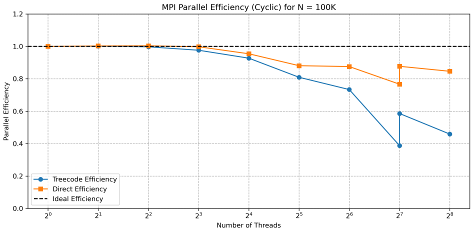
Number of Threads	Treecode Time (s)	Direct Time (s)	Treecode Speedup	Direct Speedup	Treecode Efficiency	Direct Efficiency
1	23.73	91.88	1.00	1.00	1.00	1.00
2	11.85	45.82	2.00	2.01	1.00	1.00
4	5.95	22.89	3.99	4.01	1.00	1.00
8	3.04	11.51	7.81	7.98	0.98	1.00
16	1.60	6.02	14.84	15.27	0.93	0.95
32	0.92	3.26	25.90	28.19	0.81	0.88
64	0.51	1.64	46.96	56.04	0.73	0.88
128	0.48	0.94	49.71	98.17	0.39	0.77
128	0.32	0.82	75.03	112.29	0.59	0.88
256	0.20	0.42	117.60	216.72	0.46	0.85

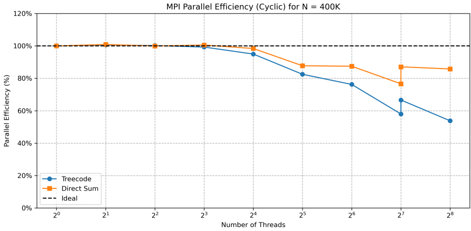
400K Particles

Table: MPI Parallel Performance (Cyclic) for N = 400 K



Number of Processes	Treecode Time (s)	Direct Time (s)	Treecode Speedup	Direct Speedup	Treecode Efficiency	Direct Efficiency
1	165.46	1469.55	1.00	1.00	1.00	1.00
2	82.23	728.36	2.01	2.02	1.01	1.01
4	41.28	367.64	4.01	4.00	1.00	1.00
8	20.83	182.71	7.94	8.04	0.99	1.01
16	10.89	93.34	15.20	15.74	0.95	0.98
32	6.27	52.34	26.40	28.08	0.82	0.88
64	3.39	26.25	48.80	55.98	0.76	0.87
128	2.23	14.98	74.26	98.08	0.58	0.77
128 (2n)	1.94	13.18	85.29	111.46	0.67	0.87
256 (4n)	1.20	6.69	137.76	219.65	0.54	0.86





Acknowledgement

- Wang, Lei. (2020). *"A Kernel-Independent Treecode Based on Barycentric Lagrange Interpolation."* Communications in Comp. Physics. 28. 1415-1436.

THANK YOU

Any Question

Do you have any questions or comments for me before we conclude?