Solar System Simulator

Marius Ion
Tudora Monica

Contents

- Mathematical Model
- Solar System Input Data
- Graphical Interface
- Serial Algorithm and Test Results
- Parallel Algorithm and Test Results
- Conclusions
- Further Work

Mathematical Model

- Short History
  - 1605 – Kepler's *Laws of Planetary Motion*
  - 1687 – Isaac Newton's *Principia*
  - 1915 – Albert Einstein's *General Relativity*

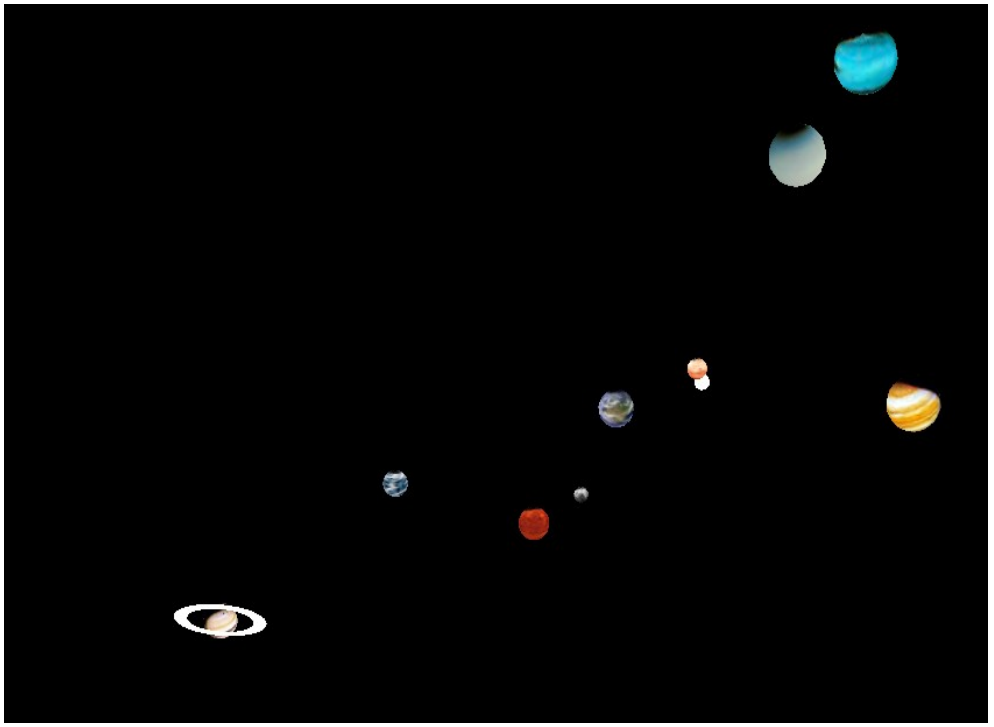
$$F_1 = F_2 = \frac{G \cdot m_1 \cdot m_2}{r^2}$$

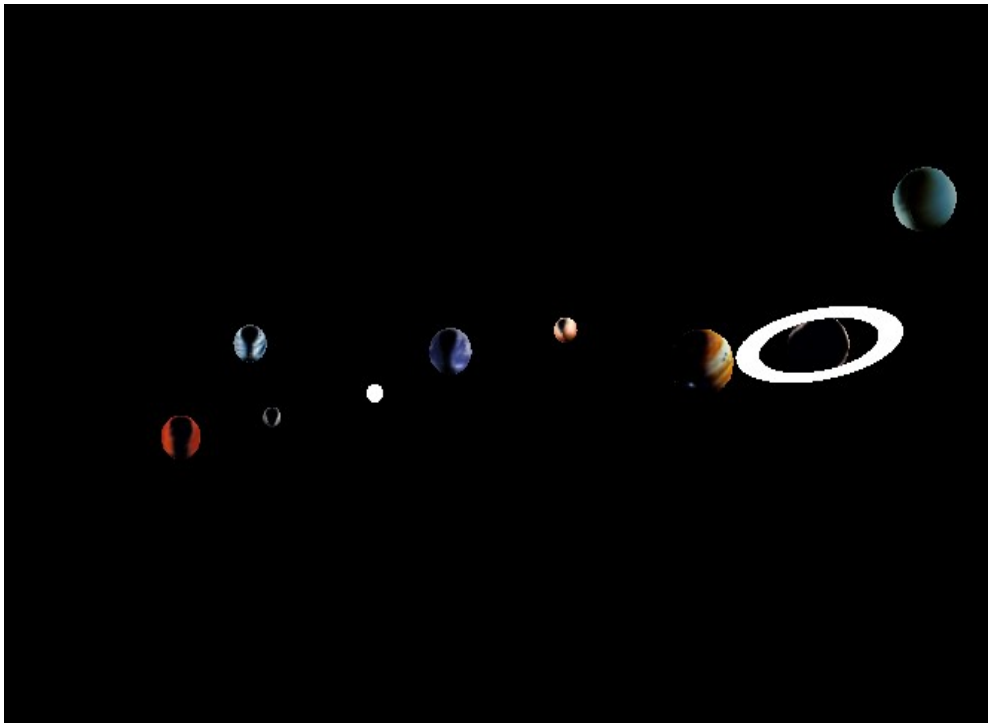
$$r = r_0 + v_0 \cdot t + a_0 \cdot \frac{t^2}{2}$$

Graphical Interface

- JOGL (java library for open gl) -> used to generate planets
- 8 planets (without Pluto) and the Sun
- Real time movement
- Data from the algorithm received through a socket

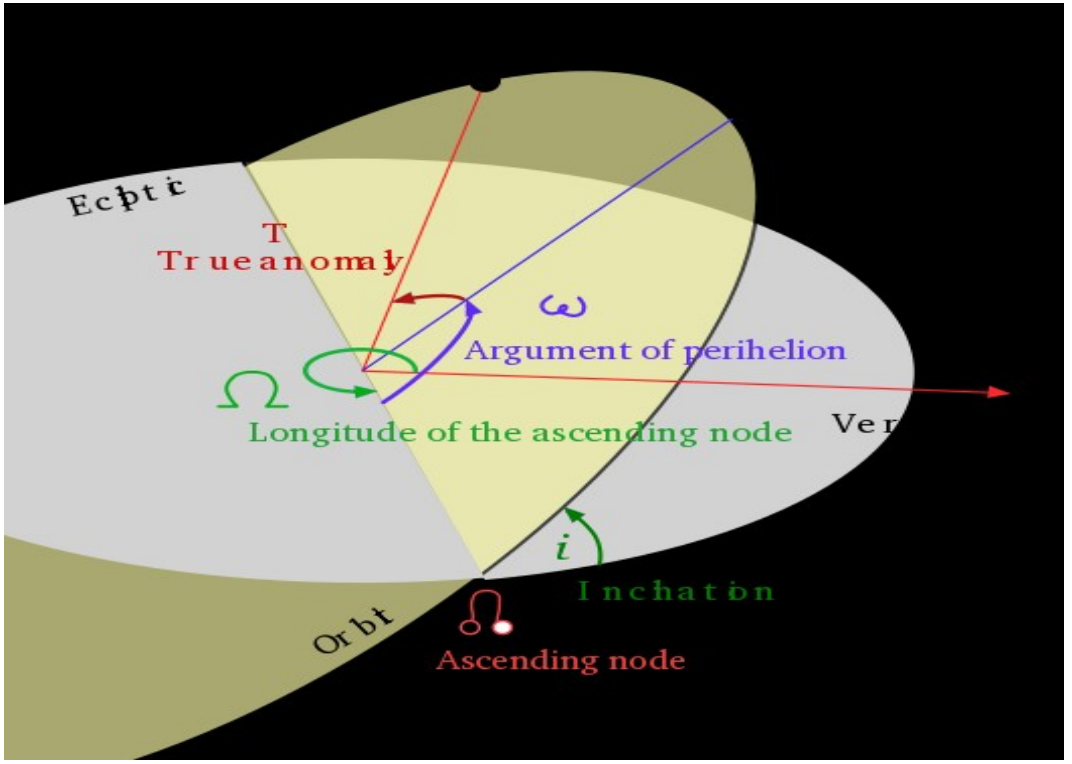
Graphical Interface images





Input Data

- J2000 epoch
- Data from US Naval Observatory
- Keplerian Orbital Elements



Input Data

$$M = \varepsilon - e \cdot \sin(\varepsilon)$$

$$\tan\left(\frac{\nu}{2}\right) = \sqrt{\frac{1+e}{1-e}} \cdot \tan\left(\frac{\varepsilon}{2}\right)$$

$$r = a(1 - e\cos\varepsilon)$$

$$rx = r(\cos(\Omega)\cos(\omega+\nu) - \sin(\Omega)\cos(i)\sin(\omega+\nu))$$

$$ry = r(\sin(\Omega)\cos(\omega+\nu) + \cos(\Omega)\cos(i)\sin(\omega+\nu))$$

$$rz = r\sin(i)\sin(\omega+\nu)$$

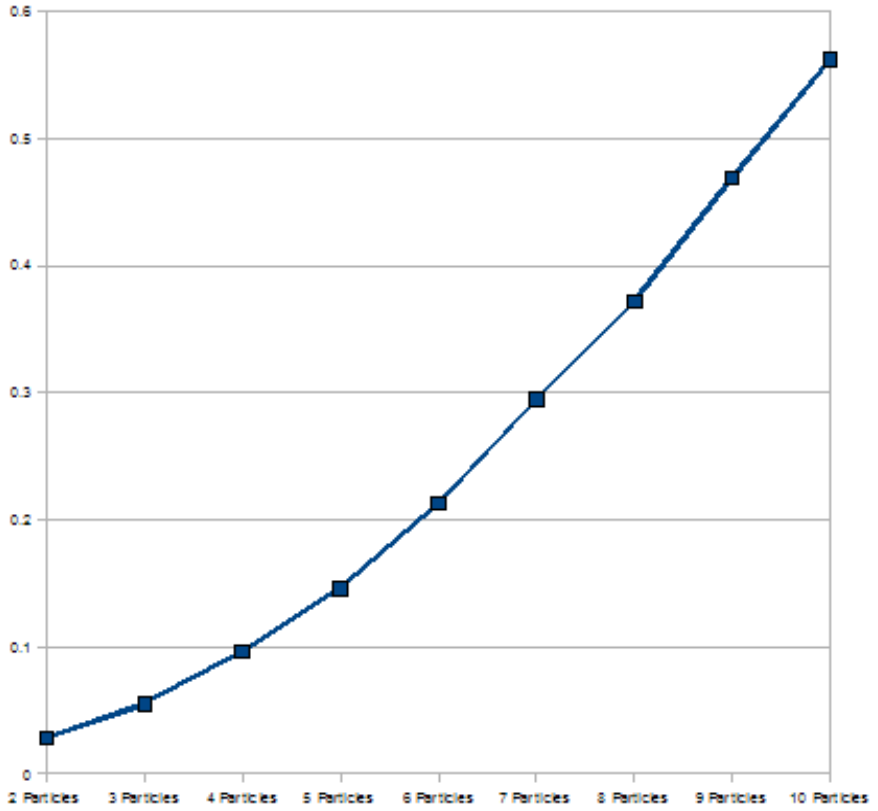
Serial Algorithm

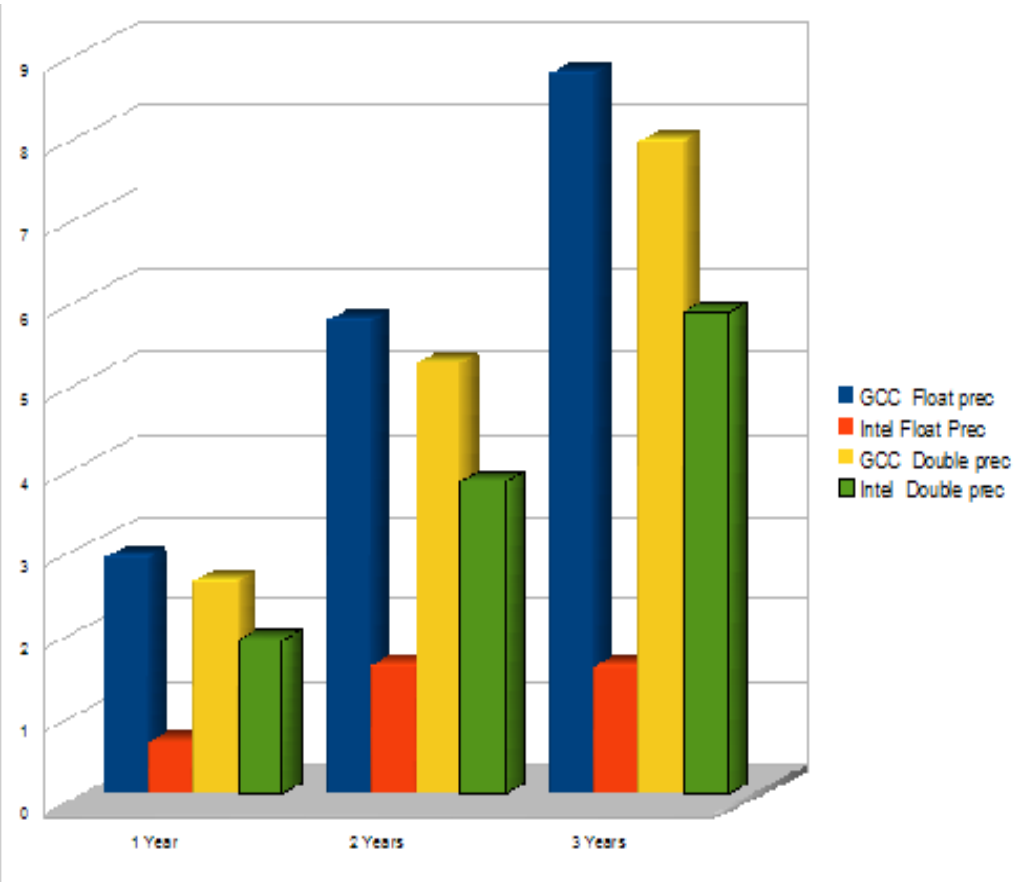
```
while( sim_step < SIM_STEPS ) {
        // compute accelerations
        memset( a, 0, n * sizeof(vector) );
        for( i = 0; i < n-1; i++ ) {
                for( j = i+1; j < n; j++ ) {
                        compute_acc(r+i, r+j, v+i, v+j, a+i, a+j, m[i], m[j]);
                }
        }

        // update positions and speeds
        for( i = 0; i < n; i++ ) {
                update_pos_vel(r+i, v+i, a+i, DT);
        }

        sim_step ++;
}
```

Serial Algorithm Test Results





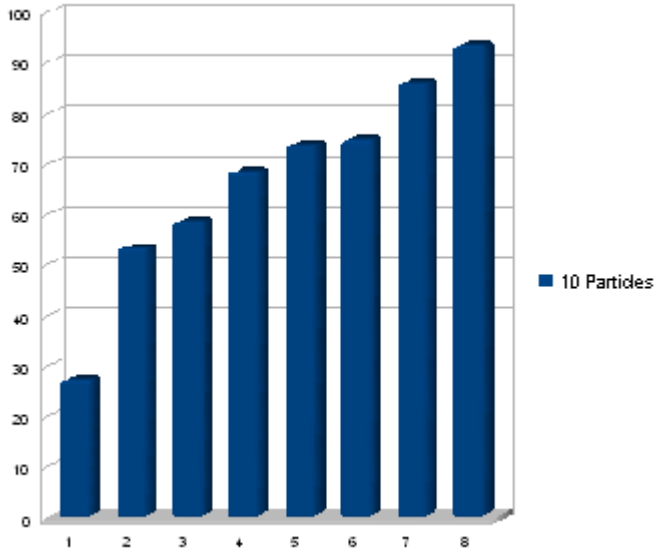
Parallel Algorithm 1

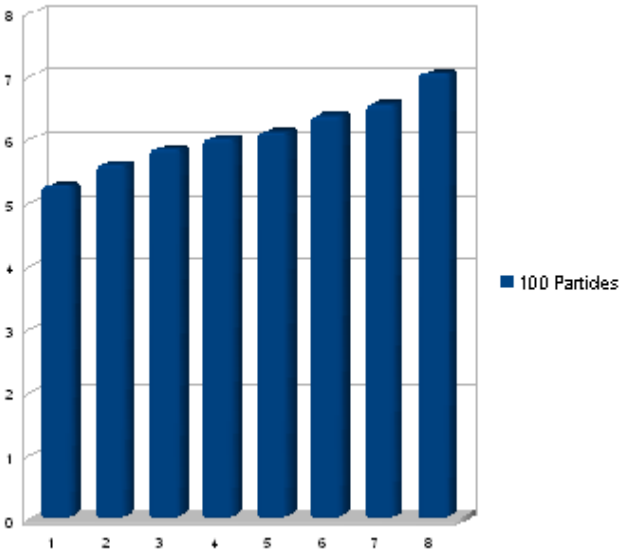
```
while( sim_step < SIM_STEPS ) {
    #pragma omp parallel for schedule(runtime) shared(ac,n,nthreads) private(i)
    for( i = 0; i < n*nthreads; i++ )
        v_init(ac+i, 0, 0, 0);
    #pragma omp parallel for schedule(runtime) shared(tasks, ntasks, r, v, ac, m, n)
private(i,j,t,tid)
    for( t = 0; t < ntasks; t++ ) {
        i = tasks[t].i;j = tasks[t].j;
        compute_acc(r+i, r+j, v+i, v+j, ac+tid*n+i, ac+tid*n+j, m[i], m[j]);
    }
    #pragma omp parallel for schedule(runtime) shared(a,ac,n,nthreads) private(i,j)
    for( i = 0; i < n; i++ ) {
        v_init(a+i,0,0,0);
        for( j = 0; j < nthreads; j++ )
            v_add(a+i, ac+j*n+i);
    }
    #pragma omp parallel for schedule(runtime) shared(r,v,a,n,DT) private(i)
    for( i = 0; i < n; i++ ) {
        update_pos_vel(r+i, v+i, a+i, DT);
    }
    sim_step ++;
}
```

Parallel Algorithm 2

```
while( sim_step < SIM_STEPS ) {
#pragma omp for schedule(static,n)
      for( i = 0; i < n*nthreads; i++ )
                  v_init(ac+i, 0,0,0);
      #pragma omp for schedule(static)
      for( t = 0; t < ntasks; t++ ) {
                  i = tasks[t].i;
                  j = tasks[t].j;
                  tid = omp_get_thread_num();
                  compute_acc(r+i, r+j, v+i, v+j, ac+tid*n+i, ac+tid*n+j, m[i], m[j]);
      }
      #pragma omp parallel for schedule(static,1)
      for( i = 0; i < n; i++ ) {
            v_init(a+i,0,0,0);
            for( j = 0; j < nthreads; j++ )
                  v_add(a+i, ac+j*n+i);
      }
      #pragma omp parallel for schedule(static,1)
      for( i = 0; i < n; i++ )
            update_pos_vel(r+i, v+i, a+i, DT);
      #pragma omp master
      sim_step ++;
      #pragma omp barrier
}
```

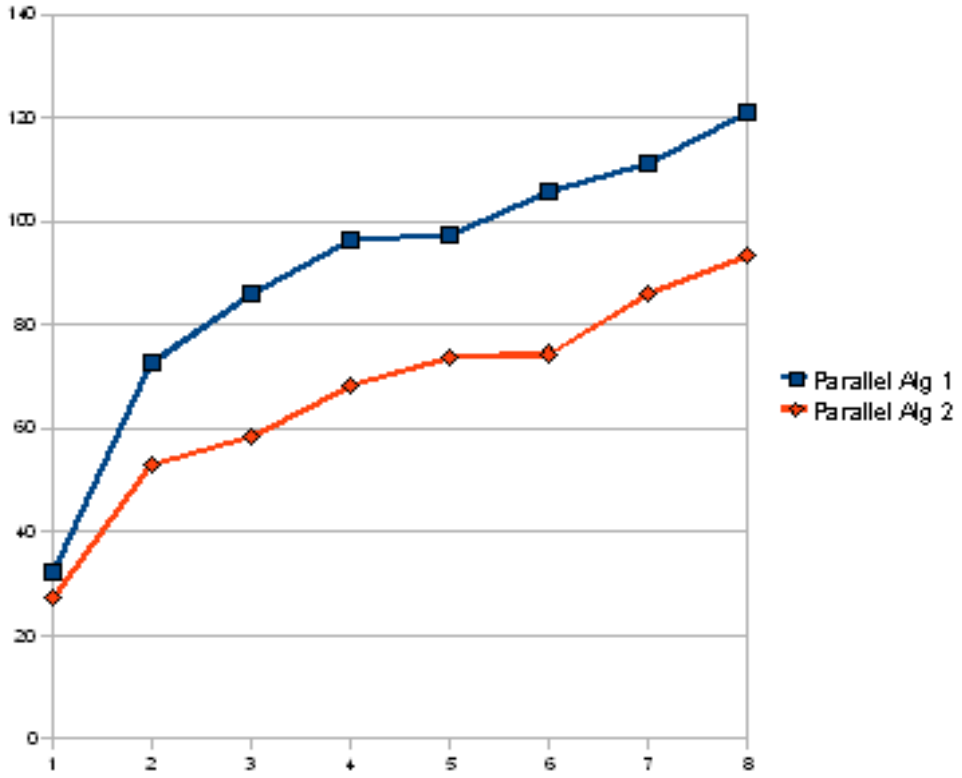
Parallel Algorithm 2 results





- Simulation time : 10 years
- Number of particles : 10/100
- Number of threads: 1-8

Alghoritms: Parallel1 vs Parallel2



- Simulation time: 10 years
- Number of particles: 10
- Number of threads: 1-8

Conclusions

- Precision ok
- Algorithm suitable for SIMD
- Algorithm not suitable for OpenMP

Further Work

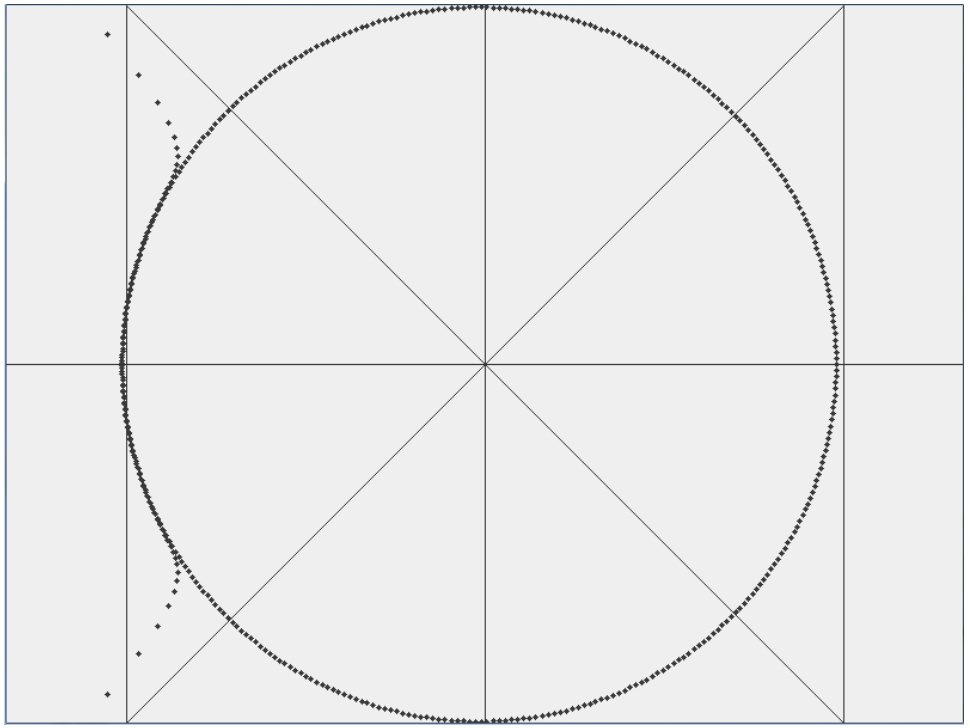
- Sochacki Method

$$\frac{d\,x_{ij}}{d\,t} = v_{ij} \qquad s_{jk} = \frac{1}{u_{jk}} \qquad s_{jk} = \sqrt{\sum_{i=1}^{3} (x_{ik} - x_{ij})^2}$$

$$\frac{d\,v_{ij}}{d\,t} = \sum_{k=1}^{N_p} \left(G \cdot mk \frac{x_{ik} - x_{ij}}{s_{jk}^{\,3}} \right)$$

$$\frac{d\,s_{jk}}{d\,t} = u_{jk} \sum_{i=1}^{3} (x_{ik} - x_{ij})(v_{ik} - v_{ij})$$

$$\frac{d\,u_{jk}}{d\,t} = -u_{jk}^{3} \sum_{i=0}^{2} (x_{ik} - x_{ij})(v_{ik} - v_{ij})$$



Questions

Thank you for listening