1 Introduction

- congruential random number generator
- transformation method
- χ^2 test of random number generator

2 Algorithm Description

2.1 congruential random number generator

The algorithm of congruential random number generator is

$$x_0 \neq 0$$
$$x_i = cx_{i-1} \bmod p$$

where c,p,x_0 are positive integers and p is prime number. The square test is created by plotting the points (x_{i-1}, x_i) in the $x - y$ plane.

2.2 transformation method

The transformation method is

$$z = \int_0^z P(z')dz' = \int_0^y P(y')dy'$$

where $P(z) = 1 \text{ if } z \in [0,1] \, 0 \text{ otherwise}$. For example, to generate a homogeneous distribution of random points inside a circle using polar coordinates ϕ and r, two uniformly distributed random number ϕ' and r' map to ϕ and r by

$$r = R\sqrt{r'}$$
$$\phi = 2\pi\phi'$$

The relation is obtained by assuming the density of points are the same in the regions $[0, 2\pi] \times [0, R]$ and $\{(x, y)|x^2 + y^2 \leq R^2\}$. Without the loss of generality, pick the density $\rho = 1$,

$$\frac{1}{2\pi R}\int_0^{r'} \mathrm{d}r \int_0^{\phi'} \mathrm{d}\phi = \frac{1}{\pi R^2}\int_0^{r} r\mathrm{d}r \int_0^{\phi} \mathrm{d}\phi$$
$$r'\phi' = \frac{1}{R}r^2\phi$$

If we shrink the range of r' and ϕ' in $[0, 1]$, it becomes

$$Rr'2\pi\phi' = \frac{1}{R}r^2\phi$$

Therefore, the map is obtained by let $\phi = 2\pi\phi'$ and $r = R\sqrt{r'}$.

2.3 χ^2 test

Generate n random numbers and count the appearance of random number N_i in k intervals with equal width. Since the range of random number is $p - 1$ for congruential random number generator, the interval width is $l = \frac{p-1}{k}$. Then, χ^2 is calculated by

$$\chi^2 = \sum_{i=1}^{k} \frac{(N_i - np_i)^2}{np_i}$$

where $p_i = \frac{1}{k}$ is the probability of a random number in the ith interval.

3 Results

3.1 Task 1

The square test of congruential random number generator with $c = 3$, $p = 101$, $x_0 = 1$ is plotted in Fig.(1) below. The random number x is in the range $[0, 1]$ by dividing the period p. $p - 1$ random numbers are generated in this case. Besides, the square test of rand() function in the standard C

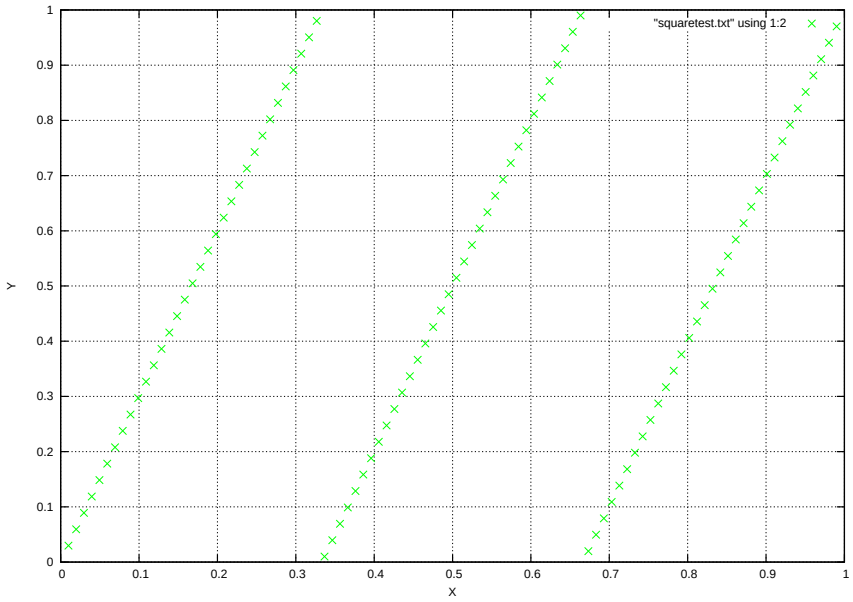


Figure 1: Square test of congruential random number generator($c = 3$, $p = 101$, $x_0 = 1$)

library(stdlib.h) is plot in Fig.(2) The congruential random number generator has a linear relation of

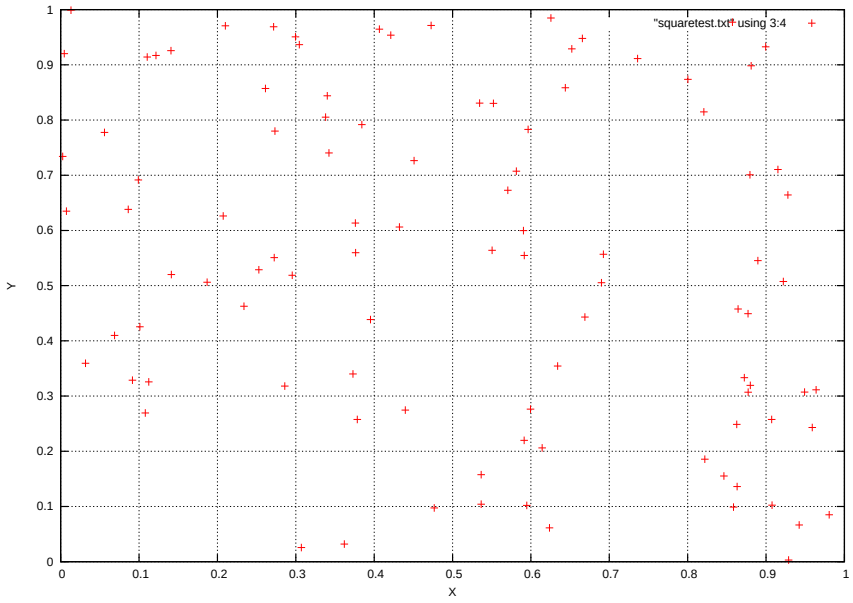


Figure 2: Square test of rand() function in stdlib.h

$x_{i-1} - x_i$ but the rand() doesn't have such property.

3.2 Task 2

The homogeneous distribution of random points inside a circle is plot in Fig.(3).

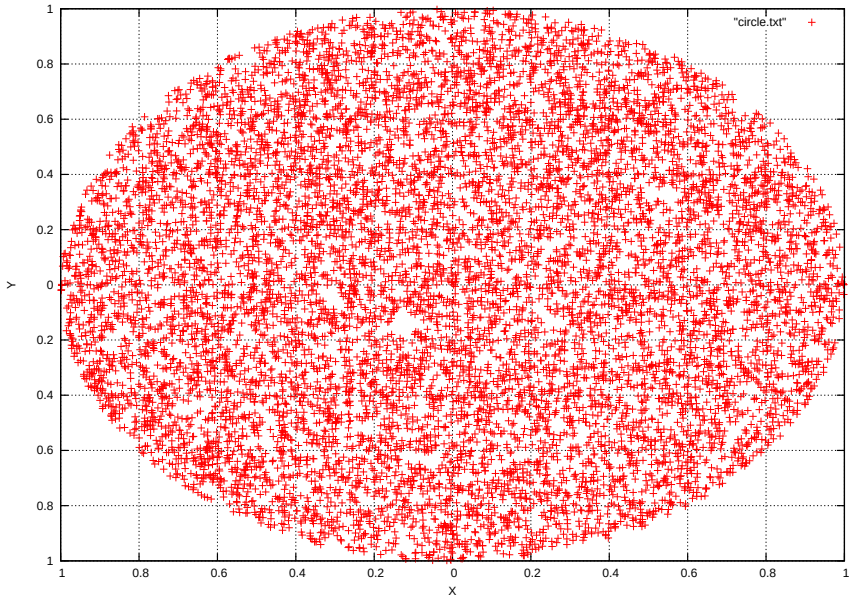


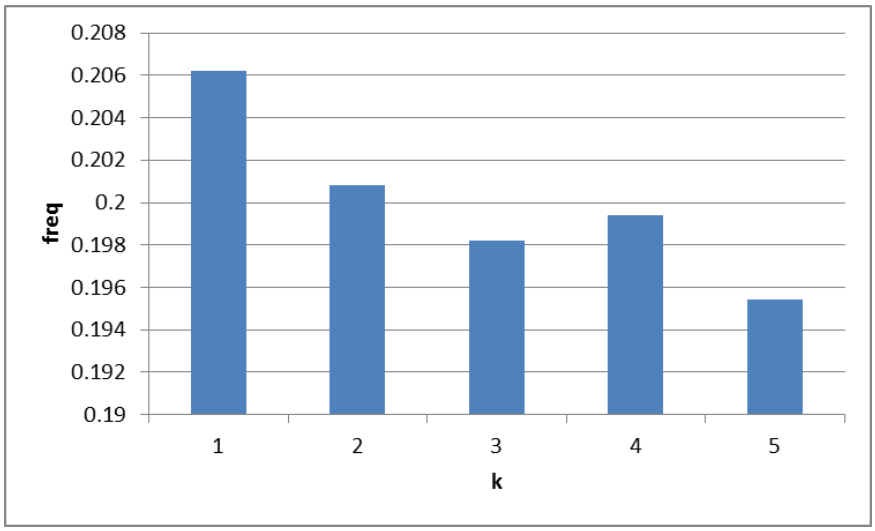
Figure 3: Homogeneous distribution of random points inside a circle($R = 1$)

3.3 Task 3

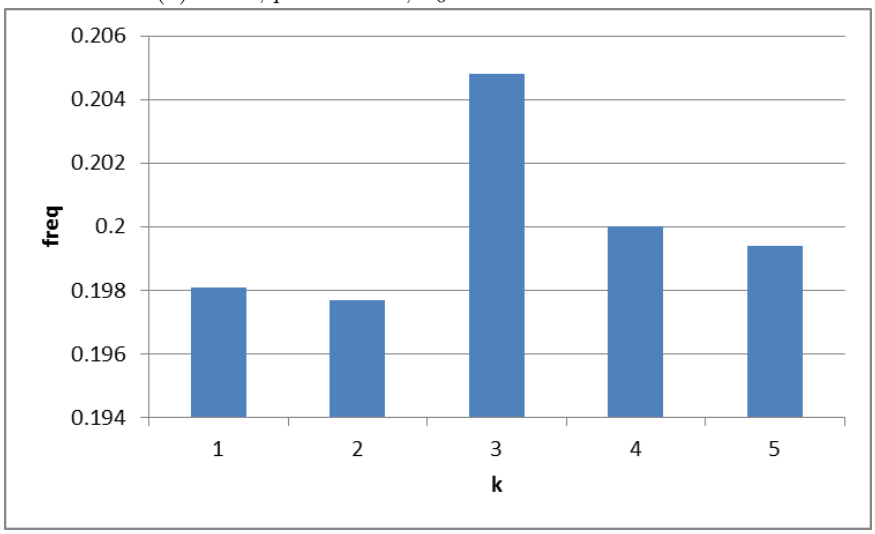
The χ^2 test of congruential random number generator($c = 7$, $p = 2^{19} - 1$, $x_0 = 1$) is plot in Fig.(4a) where $n =$ 1e4 and $k = 5$. $\chi^2 = 3.192$ and $P = 0.5$ according to the χ^2 table in [1]. By changing $c = 3$, $p = 2^{17} - 1$, the χ^2 test is plot in Fig.(4b). $\chi^2 = 1.615$ and $P = 0.25$.

References

[1] Knuth, Ervin D., *The art of computer programming*, Addison Wesley, Massachusetts, 3rd edition, 1997.



(a) $c = 7$, $p = 2^{19} - 1$, $x_0 = 1$



(b) $c = 3$, $p = 2^{17} - 1$, $x_0 = 1$

Figure 4: χ^2 test of congruential random number generator

1 Introduction

- percolation on a square lattice
- forest fires
- statistics

2 Algorithm Description

2.1 Percolation

Generate random numbers on a $N \times N$ lattice. For each site of the lattice, if the random number on this site is smaller than the occupying probability p, the site is occupied(assigned to 1), otherwise, it is empty(assigned to 0).

2.2 Forest fires

1. pick parameters N and p, create the lattice as mentioned before
2. set the first row of forests on fire.(change the sites occupied by 1 to 2)
3. for each burned site($B > 1$), check the surrounding place whether the forest(1) exists, if it does, burn it($B = B + 1$)
4. keep burning until there is no forest available for burning
5. calculate the minimum B at the end row, which is the shortest path and decide whether the cluster is spanning or not
6. calculate the maximum B of the whole lattice, which is the life time of fire

2.3 Statistics

Do several tests by varying N and p to get the average value of the shortest path and the life time.

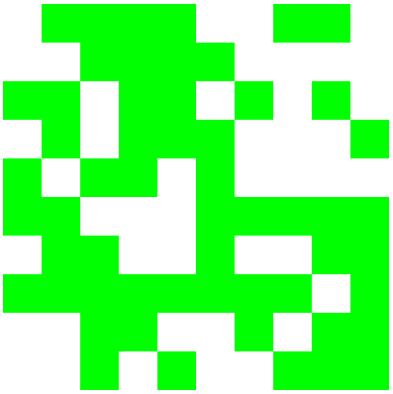
3 Results

3.1 Task 1,2

The percolation on a square lattice 10×10 is shown in Fig.(1a) while the fore fires generate the Fig.(1b).

3.2 Task 3

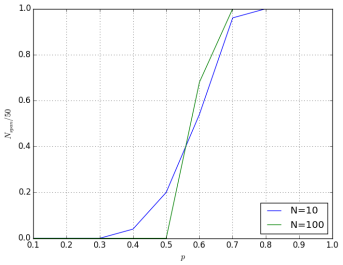
Set the number of tests to be 50, the average shortest path and lifetime are 9.98 and 13.3 for $N = 10$, $p = 0.6$. Pick p in the range $[0, 1]$ with step 0.1 and N to be 10 or 100. The frequency of span cluster versus p is plot Fig.(2a) while the average shortest path and the average life time are plot in Fig.(2b) and Fig.(2c) respectively. The threshold probability $p_c \approx 0.55$.



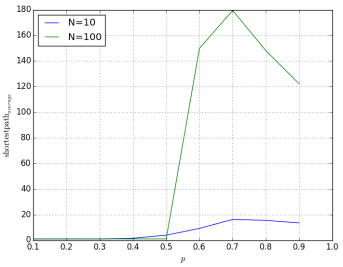
(a) $N = 10$, $p = 0.6$

(b) $N = 10$, $p = 0.6$; the lifetime is depicted in blue, the first row on fire is depicted in red and the burned sites are in black, the cluster is spanning(the shortest path=16)

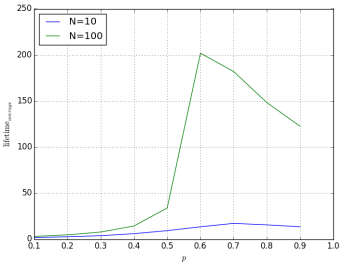
Figure 1: Percolation and Forest fire



(a) The frequency of span cluster versus p



(b) The average shortest path versus p



(c) The average life time versus p

Figure 2: The statistical values of 50 tests for $p = 0.1, 0.2, ..., 0.9$ and $N = 10, 100$

Exercise 3

Hui Zhou

October 16, 2016

1 Introduction

- Cluster size distribution using Hoshen-Kopelman algorithm(J.Hoshen and R.Kopelman, Phys. Rev. B 14, 3428, 1976)

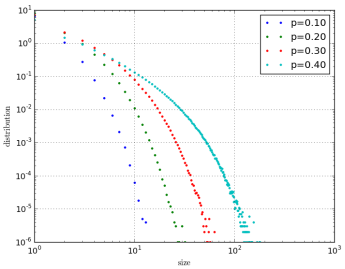
2 Algorithm Description

The site of $N \times N$ lattice is occupied by $N_{ij} \in \{0, 1\}$, $i, j \in \{1, 2, ..., N\}$. Use $k = 2, ..., k_{max}$ to label the cluster. The possible maximum number of clusters are $k_{max} = 1 + \sum_{i,j} N_{ij}$. Count the size of cluster k and store it in the array M_k

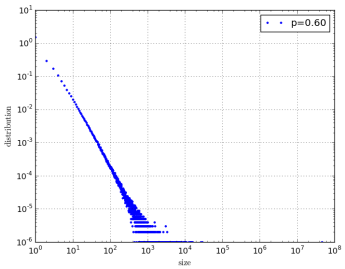
1. $k = 1$
2. for all i, j of N_{ij}, N_{ij} is occupied$(= 1)$
   (a) if top and left are empty$(= 0)$, then $k = k + 1$, $N_{ij} = k$, $M_k = 1$.
   (b) if one is occupied(> 0) with k_0 or both are occupied with the same $k_1 = k_2 = k_0$, then $N_{ij} = k_0$, $M_{k_0} = M_{k_0} + 1$.
   (c) if both are occupied with different cluster k_1 and $k_2(k_1 \neq k_2)$, then two clusters meet together and are united to one cluster. So pick a smaller one $k_1 = \min(k_1, k_2)$, $k_2 = \max(k_1, k_2)$, $N_{ij} = k_1$, merge the size $M_{k_1} = M_{k_1} + M_{k_2} + 1$, and denote the cluster k_2 belonging to cluster k_1 by $M_{k_2} = -k_1$.
   (d) for each case, k_0, k_1, k_2 should be refreshed to the label of true cluster before any operation. Use the recursive loop: while$(M_k < 0), k = -M_k$.
3. for $k = 2, ..., k_{max}$, count the cluster size distribution n
   (a) if $M_k > 0$ then $n(M_k) = n(M_k) + 1$.

3 Results

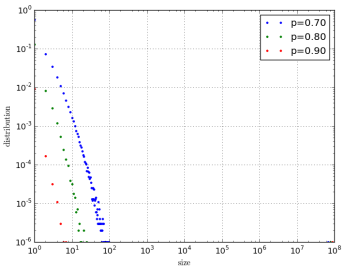
Take $N = 1e4$, $p = 0.1, 0.2, ..., 0.9$, the cluster size distribution versus size is plot in log-log scale as shown in Fig.(1)



(a) $p = 0.1 \sim 0.4$



(b) $p = 0.6$



(c) $p = 0.7 \sim 0.9$

Figure 1: $N = 1e4$, $p = 0.1, ..., 0.9$

1 Introduction

Fractal dimension of the percolating cluster

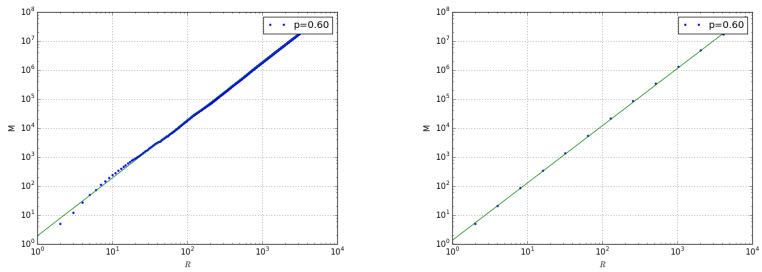
2 Algorithm Description

- percolate a $N \times N$ lattice by the probability p
- find the largest cluster by Hoshen-Kopelman algorithm
- Sandbox: compute the numbers of sites M of the largest cluster within the box $2R \times 2R$, the center of the box should be occupied by one site of the largest cluster.
- Boxcount: compute the numbers of sites M of the largest cluster within the grid $R \times R$.
- plot $R - M$ and get the fractal dimension(the slop in the log-log plot)

3 Results

3.1 Task 1, 2

Pick $N = 1e4$ and $p = 0.6$, the $M(R)$ and its fitting is plot in Fig.(1a), its fractal dimension $d \approx 1.99$. The boxcount method also yields the fractal dimension $d \approx 1.89$ as Fig.(1b) shows. The boxcount method is closer to the literature $\frac{91}{48}$ than the sandbox method.



(a) Sandbox method: $N = 1e4$, $p = 0.6$, $R = 1, 2, ..., N/2$, $\frac{\mathrm{d}\ln M}{\mathrm{d}\ln R} \approx 2$

(b) Boxcount method: $N = 1e4$, $p = 0.6$, $R = 1, 2, 4, 8, ...$, $\frac{\mathrm{d}\ln M}{\mathrm{d}\ln R} \approx 2$

Figure 1: fractal dimension of the percolating cluster

1 Introduction

Hard spheres with radius R in a 3d box $L \times L \times L$

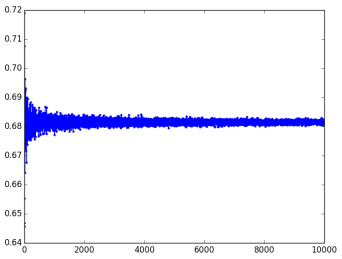
2 Algorithm Description

- populate n spheres inside $L \times L \times L$ box, the position (x, y, z) are generated by random number rand() in the range $[0, L)$ separately, the distance between two sphere $d_{ij} \geq 2R$ so that they are not overlapping. For a sphere i, if $d_{ij} < 2R$ for some sphere j, reject this position by assigning new random position, if many new random positions still cannot fulfill the volume exclusion condition, reject the present configuration (i.e. spheres $1, 2, ..., i$) and restart a new configuration until all n spheres are well placed. We denote such configuration by k
- For each configuration k, calculate average distance $d^k = \frac{2}{n(n-1)} \sum_{i<j} d_{ij}$
- For M configurations, calculate the mean average distance $\bar{d} = \frac{1}{M} \sum_{k=1}^{M} d^k$
- Vary M and plot $\bar{d}$ v.s. M

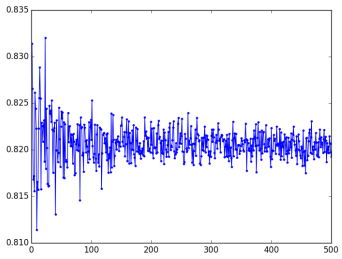
3 Results

3.1 Task

Choose $n = 16$, $R = 0.1$, $L = 1$, $M =$ 1e4, we get the volume fraction $\nu = n * \frac{4\pi R^3}{3L^3} \approx 0.065$, the $\bar{d}$ v.s. M for the dilute spheres is plot in Fig.(1a). By only changing $R = 0.22$, the volume fraction $\nu \approx 0.702$ which is close to the cubic close packing(face centered cubic structure) $\nu = \frac{\pi}{3\sqrt{2}} \approx 0.74$.



(a) $n = 16$, $R = 0.1$, $L = 1$, $M =$ 1e4, $\nu \approx 0.065$, mean average distance(M) for dilute spheres



(b) $n = 16$, $R = 0.22$, $L = 1$, $M = 500$, $\nu \approx 0.702$, mean average distance(M) for dense spheres

Figure 1: n hard spheres with radius R in 3d box $L \times L \times L$

1 Introduction

2D Ising model Monte Carlo simulation using single spin flip Metropolis algorithm

2 Algorithm Description

- Populate spin $S_{kl} \in \{-1, 1\}$ in a $L \times L$ lattice, here, $S_{kl} = 1$ at the beginning, which is the ground state (lowest energy).
- For each temperature $T = 0, 0.5, 1, ..., 5$, the configuration S_{kl}^T is relaxed to its equilibrium state first and then its average magnetization $\langle M \rangle^T$ and energy $\langle E \rangle^T$ is calculated by

$$\langle M \rangle^T = \frac{1}{N} \sum_{i=1}^{N} M_i^T, \quad \langle E \rangle^T = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} E_i^T$$

  where N is the number of system sweeps, M_i^T is the sum of spins in each lattice site and E_i^T is the sum of coupling energy in each lattice site. In detail,

$$M_i^T = \sum_{k=1}^{L} \sum_{l=1}^{L} S_{kl}$$

$$E_i^T = \sum_{k=1}^{L} \sum_{l=1}^{L} -J S_{kl} \left[S_{(k-1)l} + S_{(k+1)l} + S_{k(l-1)} + S_{k(l+1)} \right]$$

  Note that periodic boundary condition: $(k \pm 1) = k \pm 1 \bmod L$, $(l \pm 1) = l \pm 1 \bmod L$ is used here for the energy calculation and the factor $\frac{1}{2}$ for the average energy is because we calculate the coupling energy twice in each site.

Furthermore, a system sweep i for the magnetization M_i^T and the energy E_i^T is done by the single spin flip Metropolis algorithm,

1. For each site k, l in the $L \times L$ lattice,
2. calculate the coupling energy E_{kl}^0 for $S_{kl}^0 = S_{kl}$ by

$$E_{kl}^0 = -J S_{kl}^0 \left[S_{(k-1)l} + S_{(k+1)l} + S_{k(l-1)} + S_{k(l+1)} \right]$$

3. flip the spin $S_{kl}^1 = -S_{kl}^0$ and calculate its new coupling energy E_{kl}^1,

$$E_{kl}^1 = -J S_{kl}^1 \left[S_{(k-1)l} + S_{(k+1)l} + S_{k(l-1)} + S_{k(l+1)} \right]$$

4. the energy difference $\Delta E_{kl} = E_{kl}^1 - E_{kl}^0$ is

$$\Delta E_{kl} = 2J S_{kl}^0 \left[S_{(k-1)l} + S_{(k+1)l} + S_{k(l-1)} + S_{k(l+1)} \right]$$

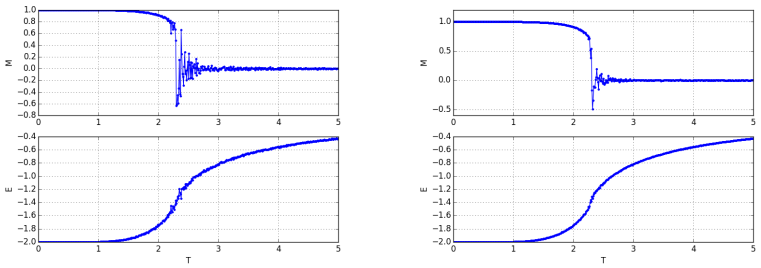
5. pick a random number $r \in [0, 1)$ by rand()
6. if $r < \min\{1, \exp(-\frac{\Delta E}{k_B T})\}$, accept the flip $S_{kl} = S_{kl}^1$, else reject the flip $S_{kl} = S_{kl}^0$
7. sum S_{kl} and E_{kl} to get M_i^T and E_i^T

Besides, the relaxation of the configuration S_{kl}^T is also done by several systems sweeps $j = 1, 2, ..., N_{relax}$ in order to facilitate the N samplings afterwards.

3 Results

3.1 Task

Choose $L = 32$, $J = 1$, $k_B = 1$ and $T = 0, 0.01, 0.02, ..., 5$, the relaxation sweeps $N_{relax} = 100$ and sampling sweeps $N = 100$ yield the average magnetization $\langle M \rangle^T$ and energy $\langle E \rangle^T$ in Fig.(1a). By only changing $L = 100$, $\langle M \rangle^T$ and $\langle E \rangle^T$ are plot in Fig.(1b). The critical slowing down happens near the critical temperature $T_c = \frac{2}{\log(1+\sqrt{2})} \approx 2.27$.



(a) $L = 32$, $J = 1$, $k_B = 1$ (b) $L = 100$, $J = 1$, $k_B = 1$

Figure 1: 2D Ising Model Monte Carlo simulation using single flip Metropolis algorithm

Exercise X

Hui Zhou

November 13, 2016

1 Introduction

Diffusion Limited Aggregation (DLA)

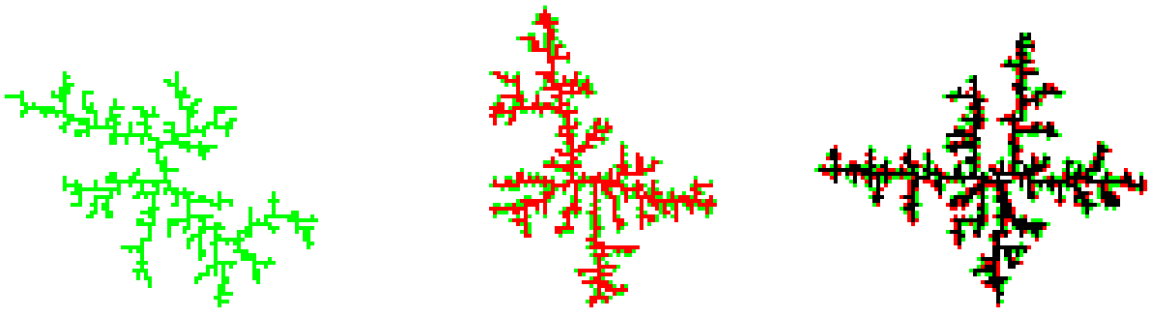
2 Algorithm Description

- Populate $N \times N$ lattice with integer 0. Here, the value of each site represents the visiting times m of random walker.
- Pick the center of lattice $(x_c, y_c) = (N/2, N/2)$ as a seed and set the value at this cite to be a integer $m > 0$. Initialize a circle $\partial B_c(R)$ centered at (x_c, y_c) with radius $R = 0 + \mathrm{d}R$, we choose $\mathrm{d}R = 3$ here. Set the maximum radius $R_{\max} = \min\{2R, N/2\}$.
- The random walker starts from a random position (x_r, y_r) on the circle $\partial B_c(R)$ (i.e. $x_r = \lfloor x_c + R\cos\theta \rfloor, y_r = \lfloor y_c + R\sin\theta \rfloor$, $\theta = \mathrm{rand}()$).
- As long as the random walker does not go outside the maximum radius $R_{\max}$, the random walking continues by $(x_r \pm 1, y_r)$ or $(x_r, y_r \pm 1)$ where the direction is also determined by a random number rand() mod 4. Otherwise, release a new random walker.
- If the random walker touches the cluster (i.e. the value at $(x_r \pm 1, y_r)$ or $(x_r, y_r \pm 1)$ is m), increase the value by 1 at this cite (x_r, y_r) and calculate the distance to the center d. If $d + \mathrm{d}R > R$, update the radius $R = R + \mathrm{d}R$ and the maximum radius $R_{\max} = \min\{2R, N/2\}$ so that the circle $\partial B_c(R)$ always covers the DLA cluster meanwhile is inside the lattice. Then release a new random walker.
- The releasing of random walker stops when $R \geq R_{max}$.

3 Results

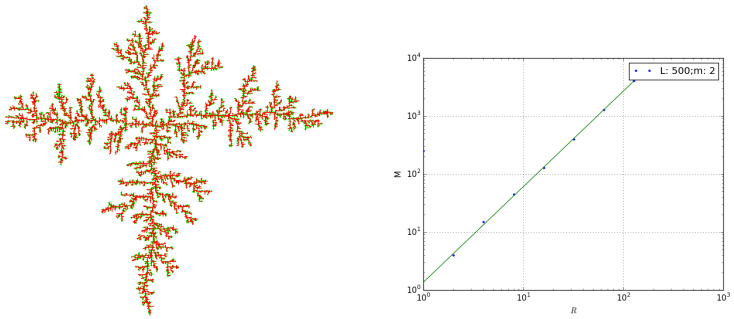
3.1 Task

Choose $N = 100$ and $m = 1, 2, 3$ separately, the DLA clusters are plot in 1. Choose $N = 500$ and $m = 2$, the DLA cluster is plot in 2 and the fractal dimension calculated by box counting method gives $d_f \approx 1.6478$.



(a) green($m = 1$) (b) red($m = 2$) (c) black($m = 3$)

Figure 1: DLA model: 100×100 lattice, the visited times $m = 1, 2, 3$ for a perimeter cite



(a) red($m = 2$) (b) fractal dimension $d_f \approx 1.6478$

Figure 2: DLA model: 500×500 lattice, the visited times $m = 2$ for a perimeter cite and the boxcouting method for fractal dimension calculation

1 Introduction

Random Walk

2 Algorithm Description

- A 2D random walk starts from initial position (x_0, y_0), by the random angle $\theta \in [0, 2\pi]$ and constant step size $\mathrm{d}R = 1$, the position (x_n, y_n) is updated by $(x_{n-1} + \mathrm{d}R\cos\theta, y_{n-1} + \mathrm{d}R\sin\theta)$. The self-avoiding random walk requires a special θ that the new position doesn't overlap the previous positions. For a 2D sphere with radius $\frac{\mathrm{d}R}{2}$, $(x_i - x_j)^2 + (y_i - y_j)^2 \geq (\mathrm{d}R)^2$
- The square of distance between the initial position (x_0, y_0) and the final position (x_N, y_N) is calculated and denoted by R^2
- Repeat the random walk M times to obtain the average value of R^2 and the estimated error δ, that is

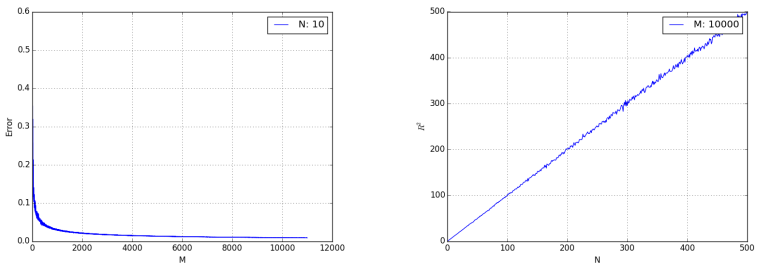
$$\langle R^2\rangle = \frac{1}{M}\sum_{k=1}^{M} R_k^2, \quad \delta = \sqrt{\frac{1}{M}\left[\langle (R^2)^2\rangle - \langle R^2\rangle^2\right]}, \quad \langle (R^2)^2\rangle = \frac{1}{M}\sum_{k=1}^{M}(R_k^2)^2$$

- Choose $M = M^*$ such that $\frac{\delta}{\langle R^2\rangle} < 1\%$

3 Results

3.1 Task 1

Choose $N = 10$, the estimated error δ v.s. the sampling times M is plot in Fig.(1a). Find the $M^* =$ 1e4 such that $\delta < 1\%$, the square of distance $\langle R^2\rangle$ v.s. the number of steps N is plot in Fig.(1b), which indicates $\langle R^2\rangle \propto N$.

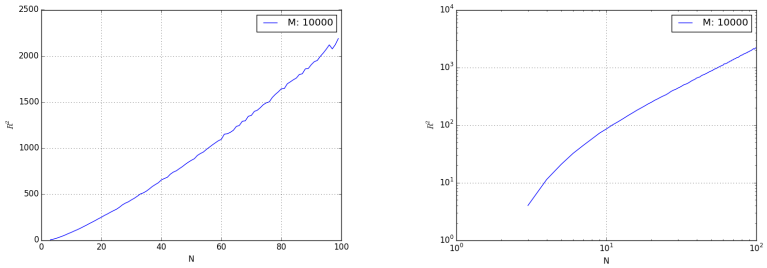


(a) N=10 (b) $M =$ 1e4

Figure 1: Simple Random Walk

3.2 Task 2

Choose $M =$ 1e4 and $N \in [3, 100]$, the square of distance $\langle R^2\rangle$ v.s. the number of particles N is plot in Fig.(2a) and its log-log plot in the large N indicates $\log\langle R^2\rangle \propto \log N$.



(a) $\langle R^2 \rangle$-N (b) $\log\langle R^2 \rangle$-$\log N$

Figure 2: Self-avoiding Random Walk

1 Introduction

Newton/Secant Method

2 Algorithm Description

The optimization problem is

$$\max_{\boldsymbol{x}} f(\boldsymbol{x}) = e^{-(x_1-\mu_1)^2-(x_2-\mu_2)^2}$$

Newton/Secant method is used here for solving this problem, which is to find the root of $\nabla f = 0$. The iteration converges to the maximum if the initial guess is near the maximum. In summary, the root finding of ∇f is

- Set the initial guess $\boldsymbol{x}_0$ near the maximum.
- The root is searched iteratively,

  $$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} - \left[\nabla^2 f(\boldsymbol{x}_{k-1})\right]^{-1} \cdot \nabla f(\boldsymbol{x}_{k-1})$$

  where $\left[\nabla^2 f\right]^{-1}$ is the inverse of the Hessian matrix of f. Obviously, $\det(\nabla^2 f) \neq 0$. For Newton method, the Hessian matrix is analytical. For Secant method, the Hessian matrix is numerical. Namely,

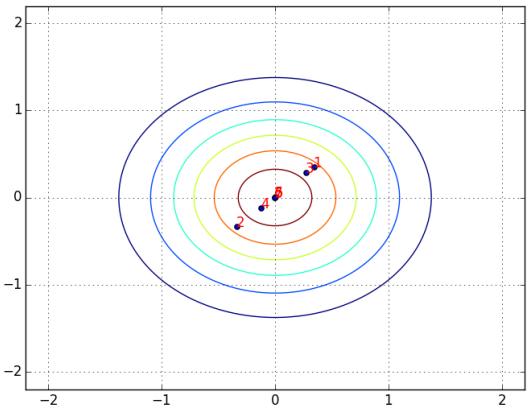
  $$\nabla^2 f_{i,j} = \frac{\nabla f_i(\boldsymbol{x} + h_j \boldsymbol{e}_j) - \nabla f_i(\boldsymbol{x})}{h_j}$$

  where $h_j \approx x_j\sqrt{\epsilon}$, e.g. $\epsilon = 2^{-53}$.
- The searching terminates if $\|\nabla f\| < \epsilon$.

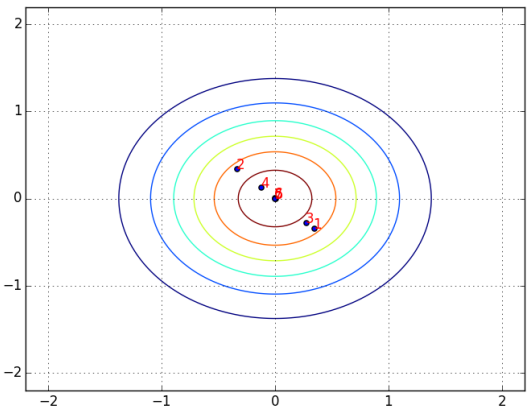
3 Results

3.1 Task 1, 2

Set the maximum of the function f at $(\mu_1, \mu_2) = (0, 0)$ and the initial guess $\boldsymbol{x}_0$ to be $(0.35, 0.35)$ for Newton method and $(0.35, -0.35)$ for Secant method, the convergence to the maximum is shown in Fig.(1). The gradient-based method highly depends on the initial guess, which is therefore a local optimization.



(a) Newton method



(b) Secant method

Figure 1: Gradient-based method

1 Introduction

Inclined throw with air resistance. To solve the following dynamical systems numerically,

$$\begin{aligned}\dot{x} &= v_x \\ \dot{z} &= v_z \\ \dot{v}_x &= -\gamma\sqrt{v_x^2+v_z^2}v_x \\ \dot{v}_z &= -g-\gamma\sqrt{v_x^2+v_z^2}v_z\end{aligned}$$

where γ is the air resistance in the range $[0,5]$ and $g \approx 9.8$ is the gravity acceleration. The initial values are

$$\begin{aligned}x(t_0) &= x_0 \\ z(t_0) &= z_0 \\ v_x(t_0) &= v_0\cos\alpha \\ v_z(t_0) &= v_0\sin\alpha\end{aligned}$$

where α is the angle of initial velocity with respect to the z axis

2 Algorithm Description

classical RK4 method [1] is to integrate the ODE $\dot{\boldsymbol{y}} = \boldsymbol{f}(t,\boldsymbol{y})$ with the initial value $\boldsymbol{y}(t_0) = \boldsymbol{y}_0$,

1. for $n = 0, 1, 2, ...$
2. $\boldsymbol{k}_1 = \boldsymbol{f}(t_n, \boldsymbol{y}_n)$
3. $\boldsymbol{k}_2 = \boldsymbol{f}(t_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_1)$
4. $\boldsymbol{k}_3 = \boldsymbol{f}(t_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_2)$
5. $\boldsymbol{k}_4 = \boldsymbol{f}(t_n + h, \boldsymbol{y}_n + h\boldsymbol{k}_3)$
6. $\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{h}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4)$
7. $t_{n+1} = t_n + h$

where h is the step size. Here,

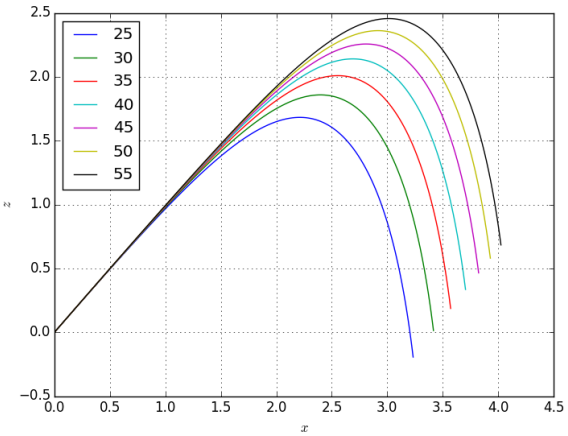
$$\begin{aligned}\boldsymbol{y} &= [x, z, v_x, v_z] \\ \boldsymbol{f}(t,\boldsymbol{y}) &= \left[v_x, v_z, -\gamma\sqrt{v_x^2+v_z^2}v_x, -g-\gamma\sqrt{v_x^2+v_z^2}v_z\right]\end{aligned}$$

3 Results

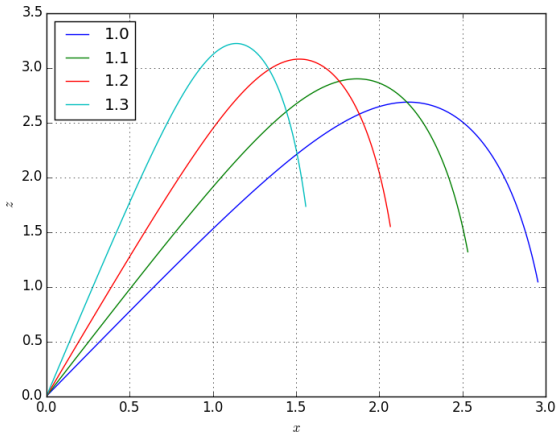
3.1 Task 1

Choose $h = 0.01$ and $\gamma = 0.7$. By varying the initial velocity v_0 or the angle α, the trajectories are plot in Fig.(1).

[1] https://en.wikipedia.org/wiki/Runge-Kutta_methods



(a) $\alpha = 45°$



(b) $v_0 = 40$

Figure 1: Trajectories with various α and v_0, $\gamma = 0.7$, $h = 0.01$

3.2 Task 2

Choose $h = 1\text{e} - 4$ and $v_0 = 40$, the angle α corresponding to the maximum x range are plot with respect to the friction coefficient γ in Fig.(2).

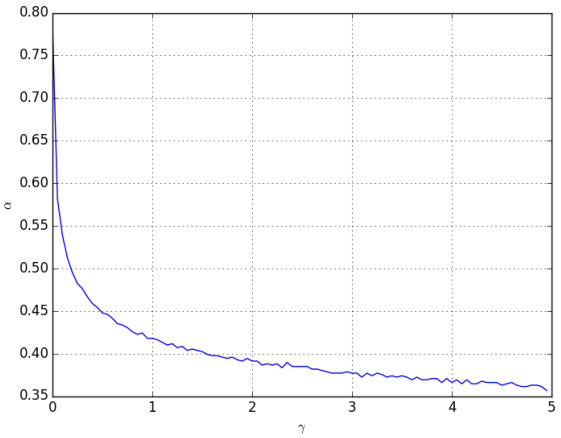


Figure 2: $\alpha_{max}(\gamma)$, $v_0 = 40$, $h = 1\text{e} - 4$

1 Introduction

2D Poisson's equation:

$$\frac{\partial^2\phi(x,y)}{\partial x^2}+\frac{\partial^2\phi(x,y)}{\partial y^2}=-\rho(x,y)$$

where $\rho(x,y)=\delta(x-x_0)\delta(y-y_0)$ and $\phi=0$ at the boundary.

2 Algorithm Description

The 2D Poisson's equation is solved numerically by finite difference method. For example, Jacobi or Gauss-Seidel method.

- The box domain $[0,1]\times[0,1]$ is discretized by $N\times N$ lattice, the grid size $h=\frac{1}{N-1}$.
- Initialize the potential $\phi_{ij}=0$ and the charge density $\rho_{ij}=\frac{1}{h^2}$ if the grid point (i,j) nearest to the point charge (x_0,y_0), otherwise $\rho_{ij}=0$.
- Time iteration $t=0,1,...,$
  - using Jacobi method:

$$\phi_{ij}^{t+1}=\frac{1}{4}\left[\phi_{i+1j}^{t}+\phi_{i-1j}^{t}+\phi_{ij+1}^{t}+\phi_{ij-1}^{t}\right]+\frac{h^2}{4}\rho_{ij}$$

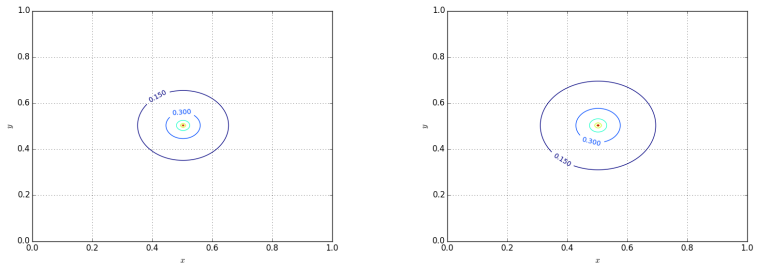
  - using Gauss-Seidel method:

$$\phi_{ij}^{t+1}=\frac{1}{4}\left[\phi_{i+1j}^{t}+\phi_{i-1j}^{t+1}+\phi_{ij+1}^{t}+\phi_{ij-1}^{t+1}\right]+\frac{h^2}{4}\rho_{ij}$$

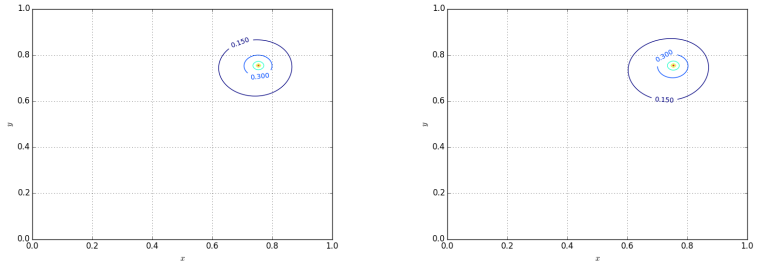
3 Results

Choose $N=200$, $t<$ 1e4, the contours of the potential for $(x_0,y_0)=(0.5,0.5)$ or $(x_0,y_0)=(0.75,0.75)$ are plot in Fig.(1a, 1b). For 2 point charges $(x_0,y_0)=(0.5,0.5)$ and $(x_1,y_1)=(0.75,0.75)$, the contours of the potential are plot in Fig.(1c). The potential contour is propagated faster by the Gauss-Seidel method than the Jacobi method.

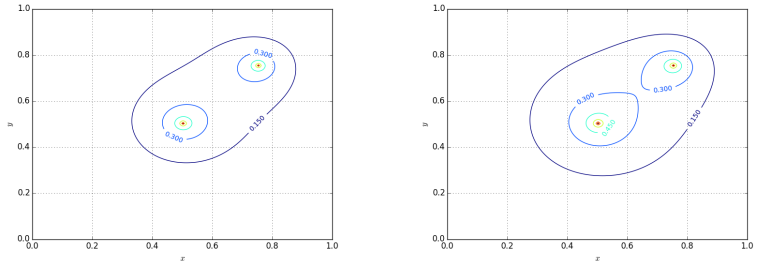
3.1 Task



(a) $(x_0, y_0) = (0.5, 0.5)$



(b) $(x_0, y_0) = (0.75, 0.75)$



(c) $(x_0, y_0) = (0.5, 0.5)$, $(x_1, y_1) = (0.75, 0.75)$

Figure 1: 2D Poisson's equation $\frac{\partial^2 \phi(x,y)}{\partial x^2} + \frac{\partial^2 \phi(x,y)}{\partial y^2} = -\rho(x, y)$ using Jacobi (left) or Gauss-Seidel method (right)

1 Introduction

Poisson's equation for the uniform charge distribution.

$$\frac{\partial^2\phi(x,y)}{\partial x^2} + \frac{\partial^2\phi(x,y)}{\partial y^2} = -\rho(x,y)$$

where $\rho(x,y) = 1.0$ and $\phi = 0$ at the boundary.

2 Algorithm Description

Conjugate gradient method[1] is used here to solve the discretized Poisson's equation,

$$\boldsymbol{A\phi} = \boldsymbol{b}$$

The finite difference discretization $\boldsymbol{A}$ and the source $\boldsymbol{\phi}$ have the following non-zero components for the interior point $(i,j) \in \{2,...,N_y-1\} \times \{2,...,N_x-1\}$,

$$\begin{aligned}
\boldsymbol{A}_{II} &= -\frac{2}{\mathrm{d}x^2} - \frac{2}{\mathrm{d}y^2} \\
\boldsymbol{A}_{I(I-1)} &= A_{I(I+1)} = \frac{1}{\mathrm{d}y^2} \\
\boldsymbol{A}_{I(I-N_y)} &= A_{I(I+N_y)} = \frac{1}{\mathrm{d}x^2} \\
\boldsymbol{b}_I &= -\rho_{ij} = -1
\end{aligned}$$

where the index $I = (j-1)N_y + i$. Besides, $\boldsymbol{A}_{II} = 1$, $\boldsymbol{b}_I = 0$ for the boundary points [1].

In detail, the conjugate gradient method takes the algorithm form,

- $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$
- $\boldsymbol{z}_0 = \boldsymbol{M}^{-1}\boldsymbol{r}_0$
- $\boldsymbol{p}_0 = \boldsymbol{z}_0$
- do $k = 0, 1, ..., \mathrm{maxsteps}$
  - $\alpha_k = \frac{\boldsymbol{r}_k^T \boldsymbol{z}_k}{\boldsymbol{p}_k^T \boldsymbol{A} \boldsymbol{p}_k}$
  - $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$
  - $\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{A}\boldsymbol{p}_k$
  - if $\|\boldsymbol{r}_{k+1}\| < \epsilon$ then exit endif
  - $\boldsymbol{z}_{k+1} = \boldsymbol{M}^{-1}\boldsymbol{r}_{k+1}$
  - $\beta_k = \frac{\boldsymbol{z}_{k+1}^T \boldsymbol{r}_{k+1}}{\boldsymbol{z}_k^T \boldsymbol{r}_k}$
  - $\boldsymbol{p}_{k+1} = \boldsymbol{z}_{k+1} + \beta_k \boldsymbol{p}_k$
- enddo

where $\boldsymbol{M}$ is the preconditioner (i.e. $M_{ij} = A_{ii}\delta_{ij}$ here) and $\boldsymbol{x}_{k+1}$ is the result (i.e. $\boldsymbol{\phi}$ here).

[1] `https://en.wikipedia.org/wiki/Conjugate_gradient_method`

3 Results

3.1 Task

As the implementation of the conjugate gradient method requires the matrix and vector operations, FORTRAN language is used in this task. Choose the box size 10×10 and its discretized grid 101×51 and set $\epsilon = 1\text{e} - 3$, the heat map of ϕ is shown in Fig.(1).

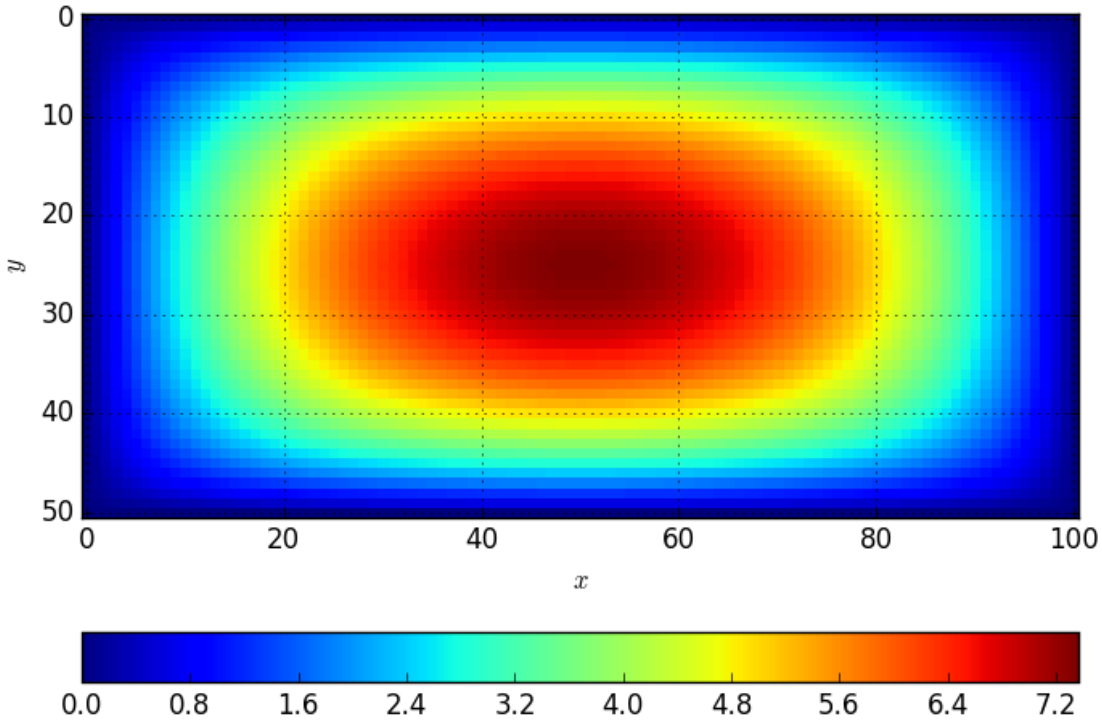


Figure 1: Poisson's equation for the uniform charge distribution using conjugate gradient method

References

[1] Gerya, Taras. Introduction to numerical geodynamic modelling. Cambridge University Press, 2009.

1 Introduction

1D wave equation $\frac{\partial^2 u(x,t)}{\partial t^2} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2}$ with the periodic boundary condition solved by the finite difference method.

2 Algorithm Description

The 1D wave equation is discretized as

$$\frac{u(x,t+\Delta t) - 2u(x,t) + u(x,t-\Delta t)}{\Delta t^2} = \frac{c^2[u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)]}{\Delta x^2}$$

let $b = (\frac{c\Delta t}{\Delta x})^2$, then

$$u(x,t+\Delta t) = 2(1-b)u(x,t) + b[u(x+\Delta x,t) + u(x-\Delta x,t)] - u(x,t-\Delta t)$$

The initial condition of $u(x,t)$ is

$$u(x,t=0) = \exp\left[-(x-10)^2\right]$$
$$u(x,t=-\Delta t) = \exp\left[-(x-c\Delta t-10)^2\right]$$

Indeed, the exact solution is $u(x,t) = \exp[-(x+ct-10)^2]$ according to d' Alembert's formula[1]. The wave will move to the left.

3 Results

3.1 Task

Set the wave speed $c = 1$, $\Delta x = 6\text{e}-4$ and $b = 0.5$, the wave at $t = 2$ is compared with the initial wave in Fig.(1). The wave moves to the left as indicated by the analytical solution. Fix $\Delta x = 1$,

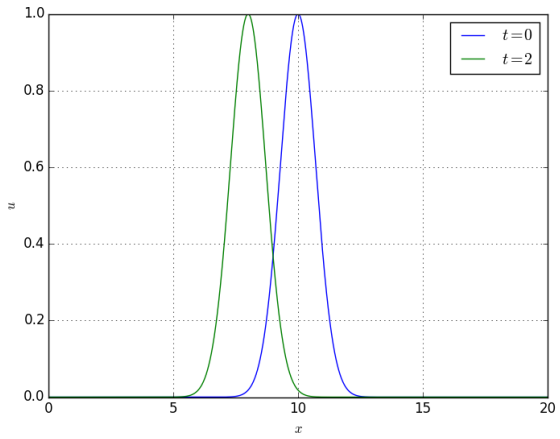


Figure 1: $u(x,t) = \exp[-(x+ct-10)^2]$ solved by the finite element method with the parameters $c = 1$, $\Delta x = 6\text{e}-4$ and $b = 0.5$.

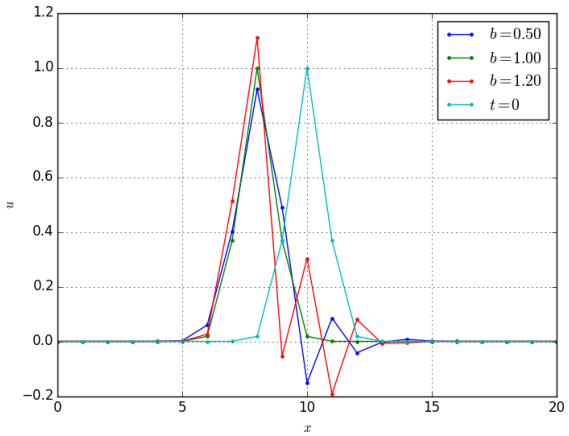


Figure 2: $u(x, t = 2)$ by the fixed parameters $c = 1$, $\Delta x = 1$ and various b

$c = 1$, different values of b result different Δt. Fig.(2) shows that the optimal value for b is $b = 1$, which preserves the wave packet[2]. The wave becomes unstable (i.e. vibration) when $b > 1$.

Change the initial condition of $u(x, t)$ to the superposition of two gaussian packets,

$$u(x, t = 0) = 2 \exp\left[-(x - 10)^2\right]$$
$$u(x, t = -\Delta t) = \exp\left[-(x - c\Delta t - 10)^2\right] + \exp\left[-(x + c\Delta t - 10)^2\right]$$

the analytical solution is $u(x, t) = \exp[-(x + ct - 10)^2] + \exp[-(x - ct - 10)^2]$, which are two gussian packets moving to the left and right correspondingly. The numerical solution in Fig.(3) shows the superposition of theses two waves at $t = 2$.

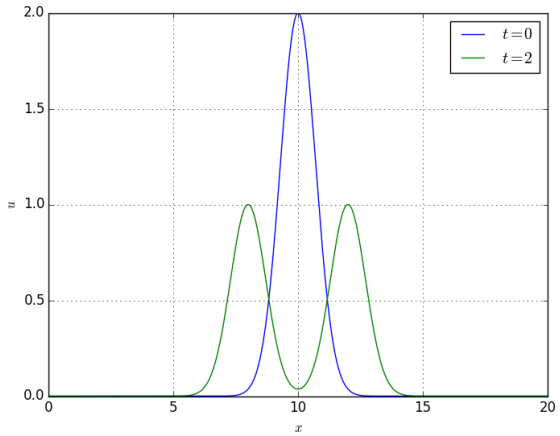


Figure 3: $u(x, t) = \exp[-(x + ct - 10)^2] + \exp[-(x - ct - 10)^2]$ solved by the finite element method with the parameters $c = 1$, $\Delta x = 6\text{e} - 4$ and $b = 1$.

[1] https://en.wikipedia.org/wiki/D'Alembert's_formula

[2] https://en.wikipedia.org/wiki/Courant-Friedrichs-Lewy_condition