Introduction to Computational Physics: 15 Questions

Herbstsemester 2013

Giuseppe Accaputo

Rechnergestützte Wissenschaften, BSc

Congruential and lagged-Fibbonacci random number generators

Congruential RNG (Multiplicative)

- Let us assume that we chose two integer numbers c and p and a seed value x_0 with $c, p, x_0 \in \mathbb{Z}$. We then create the sequence $x_i \in \mathbb{Z}$, $i \in \mathbb{N}$ iteratively by

$$x_i = (c \cdot x_{i-1}) \mod p$$

- In order to transform these random numbers to the interval $[0, 1)$ we simply divide by p.

$$0 \leq z_i = \frac{x_i}{p} < 1$$

- The sequence must repeat after at least $p - 1$ iterations. Thus, the maximal period of this random number generator (RNG) is $p - 1$.

- Also, by picking $x_0 = 0$ the sequence sits on a fixed point 0. This means $x_0 = 0$ cannot be used.

- R.D. Carmicheal proved 1910 that one gets maximal period if p is a Mersenne prime number ($M_q = 2^q - 1$) and the smallest number for which $c^{p-1} \mod p = 1$

Lagged Fibonacci RNG (Additive)

- Consider a sequence of binary numbers $x_i \in 0, 1, 1 \leq i \leq b$. Next bit in our sequence, x_{b+1} is then given by

$$x_{b+1} = \left(\sum_{j \in \mathcal{J}} x_{b+1-j} \right) \mod 2 \qquad \mathcal{J} \subset [1, \ldots, b]$$

- Two element lagged Fibonacci generator:

$$x_{i+1} = (x_{i-a} + x_{i-b}) \mod 2 \qquad a < b$$

  If (a, b) Zierler trinomial, i.e. $T_{a,b}(z) = 1 + z^a + z^b$, then the sequence has maximal period $2^b - 1$ and

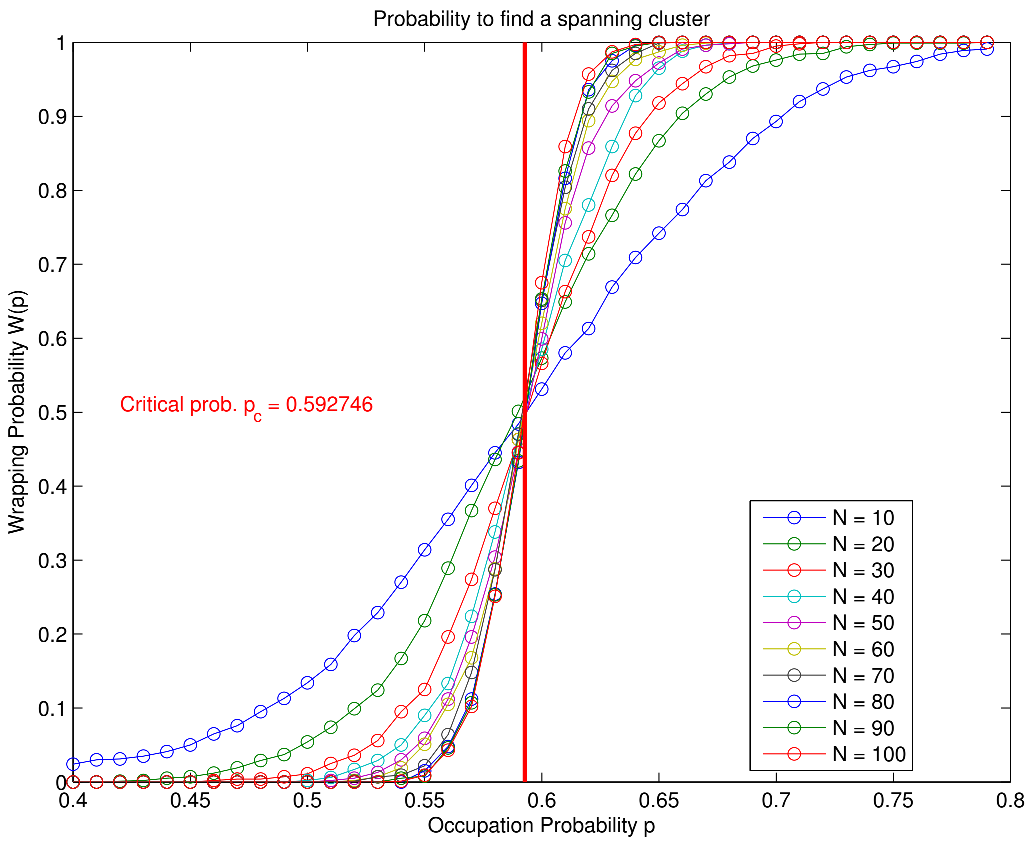
How good is a RNG?

1. Square test: plot two consecutive random numbers. The plot should be distributed homogeneously. Any sign of lines or clustering shows the non-randomness and correlation of the sequence.
2. Cube test: similar to square test, but this time the plot is three-dimensional.
3. Average value: Aritthmetic mean should correspond to the analytical mean value.

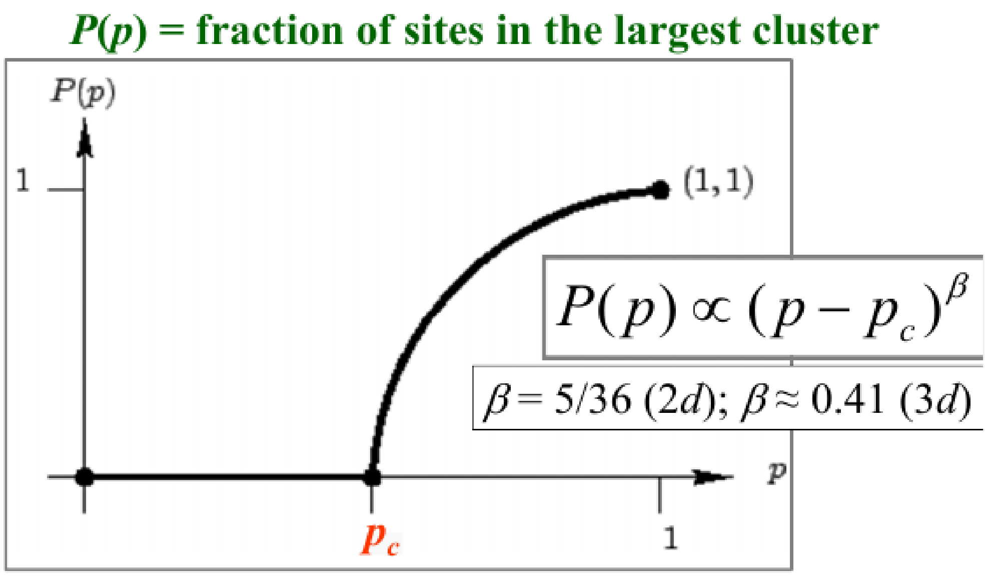
Definition of percolation

- Percolation describes the movement or filtering of fluids through porous media and is used to study phase transitions
  - A phase transition occurs when a material changes its properties in a dramatic way (Ising model: a magnet, when heated loses its magnetism)
  - Phase transitions are characterized by an order parameter
    * Ising: magnetization $M(T)$
    * Percolation: fraction of sites which belong to the biggest cluster $P(p)$
  - Order parameter behave like a power law in the region close to a critical point
    * Ising: $M \propto (T - T_c)^\beta$
    * Percolation: $P(p) \propto (p - pc)^\beta$

- Simplest percolation algorithm: lattice of size L with cells that are either occupied or not occupied. Occupy a cell with probability p.
- For small p most cells are empty and for large p most cells are occupied. At the critical probability (or percolation threshold) $p_c = 0.592$ a percolating cluster appears (cluster of cells that spans over two opposite sides of the box). p_c is the average probability at which a percolating cluster first occurs:



- $P(p)$ is the fraction of sites which belong to the biggest cluster and is the order parameter.
  - Close to the percolation threshold we have $P(p) \propto (p_c - p)^\beta$. This behaviour is called universal criticality.



Fractal dimension and sand-box method

Fractal dimension

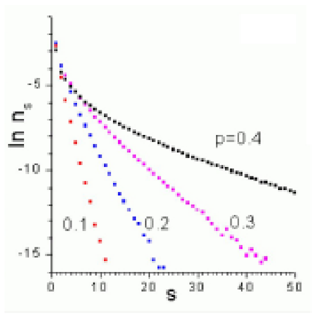
- The underlying idea of the fractal dimension is to find a measure to describe how well a given (fractal) object fills a certain space
- Self-similarity: an object is *self-similar* if it is built up of smaller copies of itself (e.g. Sierpinski-triangle)

Sandbox method

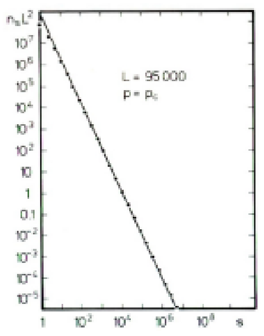
- Sanbox method:
  1. Place small box of size R in the center of the picture and count the number $N(R)$ of occupied sites in the box
  2. Successively increase the box size R in small steps until the whole picture is covered with our box
  3. Plot $N(R)$ vs. R in a log-log plot where the fractal dimension d_f is the slope.

Hoshen-Kopelman algorithm

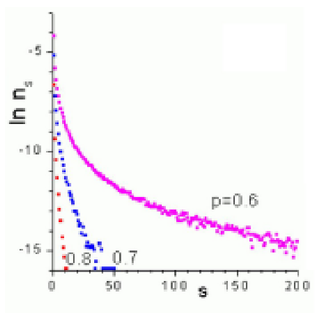
- We have clusters of different sizes s; the cluster size distribution n_s is defined as
$$n_s = \frac{N_s}{N}$$
- Hoshen-Koppelman Algorithm: define matrix N_{ij} with $N_{ij} = l \in \{0, 1\}$, cluster number $k = 2$ and array M_k (mass of cluster k). Comb through the lattice from top-left to bottom-right.
    1. Set $k = 2, M_k = 1$
    2. For all entries of N_{ij} do:
        (a) If a site is occupied and top and left neighbors are empty, we have found a new cluster. $k = k + 1, N_{ij} = k, M_k = 1$
        (b) If either the top or left site has value k_0, we increase the value of M_{k_0} by one and set $N_{ij} = k_0$.
        (c) If top and left site are occupied with k_1 and k_2 respectively (with $k_1 \neq k_2$), choose one of them (for example k_1). Set $N_{ij} = k_1, M_{k_1} = M_{k_1} + M_{k_2} + 1, M_{k_2} = -k_1$. When encountering a negative value for the mass, one can multiply it with -1 to recursively get the correct cluster (recursion stops when $M_k \geq 0$)
    3. For $k = 2 \dots k_{\text{max}}$ do:
        (a) If $M_k > 0$ then $n(M_k) = n(M_k) + 1$, i.e. construct the histogram of the differen cluster sizes
- $n_p(s) = s^{-\tau}\mathcal{R}_{\pm}\left[(p - p_c)s^{\sigma}\right]$, where $\mathcal{R}_{\pm}$ are scaling functions, where the subscript $\pm$ stands for $p > p_c$ $(+)$ and $p < p_c$ $(-)$, respectively.
    – A scaling function is a function $f(x, y)$ of two variables that can be expressed as a function of one variable $f(x')$



(a) $n_{p<p_c}(s) \propto s^{-\theta}e^{as}$



(b) $n_{p_c}(s) = s^{-\tau}$



(c) $n_{p>p_c}(s) \propto e^{-bs^{1-1/d}}$

-

- $$n_p(s) \propto \begin{cases} s^{-\theta}\exp\{as\}, & \text{if } p < p_c \\ s^{-\tau}, & \text{if } p = p_c \\ \exp\{-bs^{1-1/d}\}, & \text{if } p > p_c \end{cases}$$

Finite size scaling

Correlation function $c(r)$ and correlation length ξ

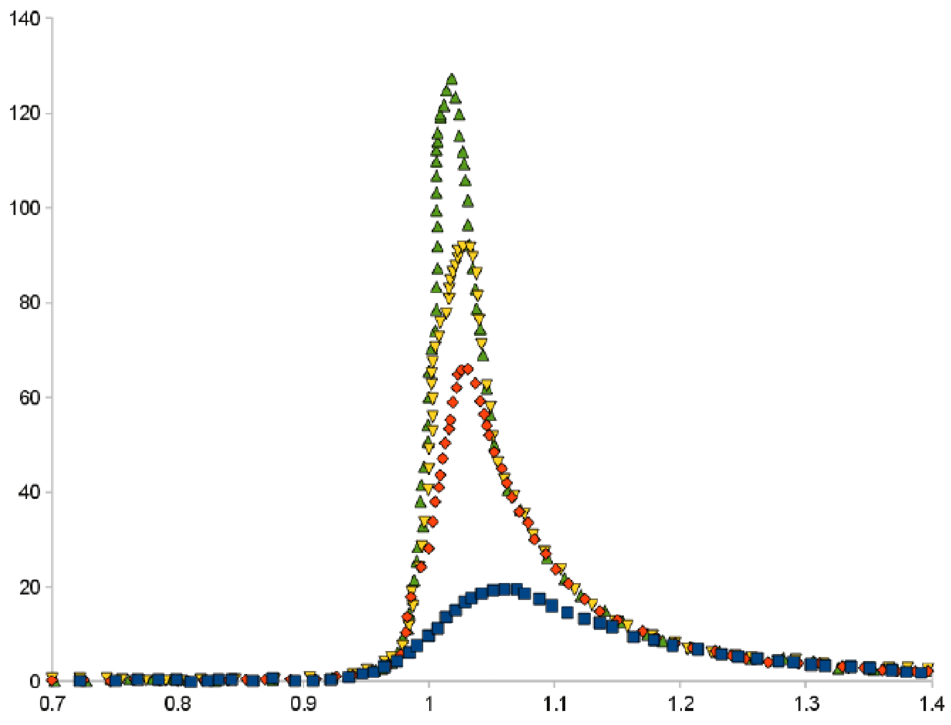
- The correlation function $c(r)$ is a measure for the amount of order in a system. It describes how microscopic variables are correlated over various distances.
- Let $\rho(x)$ be the density of the object at point x. The correlation function of the density at the origin and at a distance r is then

  $$c(r) = \langle \rho(0) \cdot \rho(r) \rangle$$

- Correlation length ξ: $c(r) \propto C + \exp\{-r/\xi\}$. It describes the typical length scale over which the correlation function of a given system decays. For $p < p_c$ ξ is proportional to the radius of a typical cluster.
- ξ has a singularity at the critical occupation probability p_c: $\xi \propto |p - p_c|^{-\nu}$
- $c(r)$ at p_c: $c(r) \propto r^{-(d-2-\nu)}$

Finite size effects and finite size scaling

- Finite size effects: we encounter problems when the system size L is smaller than the correlation length ξ. instead of obtaining a singularity we obtain a maximum.
- Finite size scaling: consider the second moment χ as a function of p and L, i.e. plot χ against the occupation probability p for several values of L we obtain plots that differ around the critical point:



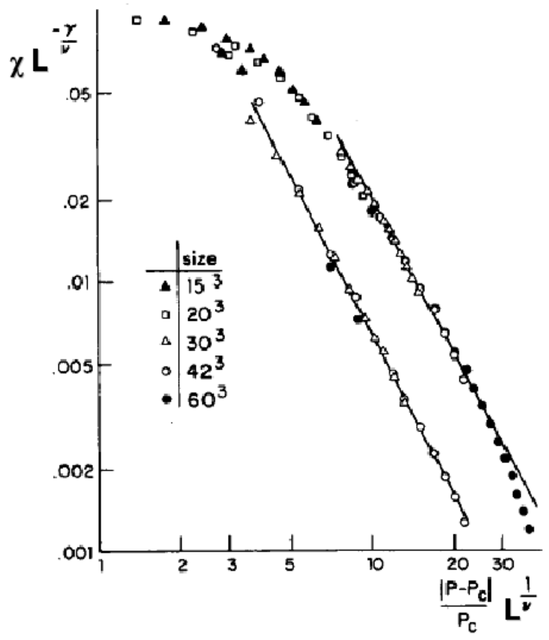
- Power law: $\chi \propto |p - p_c|^{-\gamma}$
- The second moment χ is a very strong indicator of p_c as we can see a very clear divergence around p_c. We get a plot that makes p_c much better visible than plotting the probability of finding a spanning cluster for a given occupation probability p

• χ was originally a function of two parameters p and L and now behaves as though it was a one-parameter function:

$$\chi(p, L) = L^{\gamma/\nu}\mathfrak{N}_\chi\left[(p - p_c)L^{1/\nu}\right] \Longleftrightarrow \mathfrak{N}_\chi\left[(p - p_c)L^{1/\nu}\right] = \chi(p, L)L^{-\gamma/\nu}$$

where $\mathfrak{N}$ is the so-called scaling function. Plot $(p-p_c)L^{1/\nu}$ against $\chi(p, L)L^{-\gamma/\nu}$ and observe the data collapse; the slope of the line gives $-\gamma$. At p_c we have $\chi_{\text{max}}(L) \propto L^{\gamma/\nu}$

- Finite size scaling of χ: We can observe data collapse when plotting $\chi L^{-\gamma/\nu}$ against $|p - p_c|L^{1/\nu}$. The straight lines have a slope of $-\gamma$:



- For the fraction of sites P we have $P(p,L) = L^{-\beta/\nu}\mathfrak{N}_P[(p-p_c)L^{1/\nu}]$. At p_c we find that $P \propto L^{-\beta/\nu}$ and that the number of sites is also system size dependent, i.e. $M \propto L^{d_f}$. Combine it to get $M \propto PL^d \propto L(-\beta/\nu + d) \propto L^{d_f} \implies d_f = d - \beta/\nu$
  - $\chi \propto |p-p_c|^{-\gamma}$
  - Order parameter $P(p)$: fraction of sites in largest cluster

Integration with Monte Carlo

- Major advantages of Monte Carlo methods: error Δ decreases with increasing number of samples N as follows: $\Delta \propto \dfrac{1}{\sqrt{N}}$ and Δ is not dependent on the dimension.
- $\displaystyle\int_b^a g(x)\mathrm{d}x \approx (b-a)\left[\frac{1}{N}\sum_{i=1}^{N} g(x_i)\right]$
  - Simple sampling: If we choose x_i completely at random we get a very good approximation. If $g(x)$ is not smooth, e.g. it has a singularity we get a rather bad approximation, but we get a good approximation if $g(x)$ is smooth.

  - Importance sampling: Introduce a function $p(x)$. Sampling points are now distributed according to $p(x)$. $p(x)$ helps us to pick our sampling points x_i according to their importance (e.g. select more points close to singularity). If $\frac{g(x)}{p(x)}$ is smooth we get a very good approximation.
    * If $g(x)$ has a singularity at $\hat{x}$, then define $p(x)$ such that it covers the singularity, i.e. use a Gaussian distribution with the peak of the curve at at the singularity $\hat{x}$ ($\sigma = \hat{x}$)

Detailed balance and Metropolis algorithm

- Canonical Monte Carlo
  - Probability (at thermal equilibrium) for a system to be in X is given by the Boltzmann factor: $P_{\text{eq}}(X) = \frac{1}{Z_T} \exp\{-E(X)/(k_B T)\}$
  - Ensemble average: $\langle Q \rangle = \sum_X Q(X) P_{\text{eq}}(X)$
- Properties of the probability of a Markov chain:
  - Ergodicity: $\forall X, Y : W(X \to Y) > 0$ (each configuration is reachable)
  - Normality: $\sum_Y W(X \to Y) = 1$
  - Homogeneity: $\sum_Y p_{\text{st}}(Y) W(Y \to X) = p_{\text{st}}(X)$ (the probability for a system to be in X is simply a result of systems coming from other configurations over to X)
- The Master equation:

$$\frac{\mathrm{d}p(X,t)}{\mathrm{d}t} = \sum_Y P(Y) W(Y \to X) - \sum_Y P(X) W(X \to Y)$$

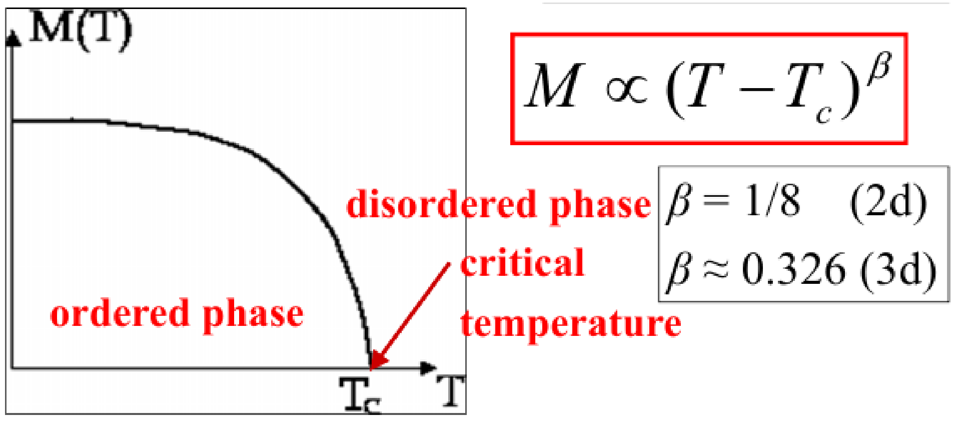
- Detailed balance:
  - Consider the stationary state: $\frac{\mathrm{d}p(X,t)}{\mathrm{d}t}$ (note: all Markov processes reach a steady state). Since we want to model the thermal equilibrium we use the Boltzmann distribution, i.e. $P_{\text{st}}(X) = P_{\text{eq}}(X)$. We get $\sum_Y P_{\text{eq}}(Y) W(Y \to$

$X) = \sum_Y P_{\text{eq}}(X)W(X \to Y)$. The detailed balance condition $P_{\text{eq}}(Y)W(Y \to X) = P_{\text{eq}}(X)W(X \to Y)$ is a sufficient condition and states that the steady state of the Markov process is the thermal equilibrium (we have achieved this by using the Boltzmann distribution for $p(X)$)

- Metropolis algorithm: $A(X \to Y) = \min(1, \exp\{-\Delta E/(k_B T)\})$
- Glauber dynamics: $A(X \to Y) = \dfrac{\exp\{-\Delta E/(k_B T)\}}{1 + \exp\{-\Delta E/(k_B T)\}}$
- Both Metropolis and Glauber fulfill detailed balance

Ising model

- The Ising model is a mathematical model of ferromagnetism in statistical mechanics
- Hamiltonian: $\mathcal{H} = E = -J \sum_{j\text{: nn of } i} \sigma_i \sigma_j - H \sum_i \sigma_i$
- Monte Carlo of the Ising model:
  1. Choose a random site i with spin σ_i
  2. Calculate $\Delta E = E(Y) - E(X) = 2J\sigma_i h_i$
     - $h_i = \sum_{\text{near. neighb. of} \sigma_i} \sigma_j$ is the local field at site i
  3. If $\Delta E \leq 0$ then flip spin, i.e. $\sigma_i \to -\sigma_i$
  4. If $\Delta E > 0$ flip spin with probability $\exp\{-\beta \Delta E\}$
- Susceptibility $\chi = \dfrac{M}{H}$ where M is the magnetization and H is the magnetic field strength. $\chi \propto |T - T_c|^{-\gamma}$
- Spontaneous magnetization: $M(T) = \lim_{H \to 0} \left\langle \dfrac{1}{N} \sum_{i=1}^{N} \sigma_i \right\rangle, M \propto |T - T_c|^{\beta}$



–

Simulate random walk

- 1D random walk
  - Number of visited sites $N_{\text{cov}} \propto \sqrt{t}$
    - $d = 1$: $N_{\text{cov}} \propto \sqrt{t}$
    - $d = 2$: $N_{\text{cov}} \propto \log t$
    - $d > 2$: $N_{\text{cov}} \propto t$
  - Mean square distance $r^2 = \frac{1}{2}Dt$

Euler method

- First-order ordinary differential equations (ODE): $\frac{\mathrm{d}y}{\mathrm{d}t} = f(y,t), y(t_0) = y_0$
- Euler recipe:
  1. Take the initial value $y(t_0) = y_0$
  2. Calculate $\frac{\mathrm{d}y}{\mathrm{d}t}$
  3. Advance linearly for Δt with the derivative at the initial value as the slope: $y(t + \Delta t) = y(t) + \Delta t \cdot y'(t)$
  4. Take the point reached in the previous step as new initial value and repeat steps 3-4
- We have the initial value problem $\frac{\mathrm{d}y(t)}{\mathrm{d}t} = f(t, y(t)), y(t_0) = y_0$. Taylor expansion: $y(t_0 + \Delta t) = y(t_0) + \Delta t \cdot f(y_0, t_0) + O((\Delta t)^2)$

  - One timestep ($y_{n+1} = t_0 + n \cdot \Delta t$) in the Euler method corresponds to $y_{n+1} = y_n + \Delta t \cdot f(y_n, t_n)$
  - The error for each time step is $\propto (\Delta t)^2$. Since we accumulate this error during every time step, i.e. $m = t_{\text{end}}/\Delta t$, the error is of order $m \cdot O((\Delta t)^2) = t_{\text{end}}/\Delta t \cdot O((\Delta t)^2) = O(\Delta t) \implies$ the Euler method is globally of order 1
  - Use small Δt to minimize error, but a small Δt is numerically very expensive (i.e. resulting in many time steps and thus large number of iteration)

2nd order Runge-Kutta

- Runge-Kutta (RK) methods achieve the same accuracy as the Euler method for much larger time steps. RK methods are numerically less expensive, yet more complicated to implement
- RK methods are derived using a Taylor expansion for $y(t+\Delta t)$ keeping all terms up to order $(\Delta t)^q$.
- 2nd order RK method: $y(t+\Delta t) = y(t) + \frac{(\Delta t)}{1!}\frac{\mathrm{d}y}{\mathrm{d}t} + \frac{(\Delta t)^2}{2!}\frac{\mathrm{d}^2y}{\mathrm{d}^2t} + O((\Delta t)^3)$

  1. Perform Euler step of size $\frac{\Delta t}{2}$, starting at the initial value $y_i(t_0)$:

$$y_i(t+1/2\Delta t) = y_i(t) + 1/2\Delta t \cdot f(y_i(t), t)$$
$$y_i(t+\Delta t) = y_i(t) + \Delta t \cdot f(y_i(t+1/2\Delta t), t+1/2\Delta t) + O((\Delta t)^3)$$

- 4th order RK method:

$$\begin{aligned} k_1 &= \Delta t \cdot f(y_n, t_n) \\ k_2 &= \Delta t \cdot f(y_n + k_1/2, t_n + \Delta t/2) \\ k_3 &= \Delta t \cdot f(y_n + k_2/2, t_n + \Delta t/2) \\ k_4 &= \Delta t \cdot f(y_n + k_3, t_n + \Delta t) \\ &\implies y_{n+1} = y_n + k_1/6 + k_2/3 + k_3/3 + k_4/6 + O((\Delta t)^5) \end{aligned}$$

2nd order predictor-corrector

- 2nd order predictor-corrector: $y(t+\Delta t) \approx y(t) + \Delta t \cdot \frac{f(y(t)) + f(y(t+\Delta t))}{2}$. This is an implicit equation which cannot be solved directly (equation depends on $y(t+\Delta t)$. We make a prediction of $y(t+\Delta t)$ by using the Taylor expansion

$y^p(t+\Delta t) = y(t) + \Delta t \cdot \frac{\mathrm{d}y(t)}{\mathrm{d}t} + O((\Delta t)^2)$. We now can compute $y^c(t+\Delta t) = y(t) + \Delta t \cdot \frac{f(y(t)) + f(y^p(t+\Delta t))}{2} + O((\Delta t)^3)$. The corrected value y^c can itself be inserted into the corrector as the predicted value for a better result, which can be don several times.

- 3rd order predictor-corrector: $y^p(t+\Delta t) = y(t) + \frac{\Delta t}{1!}\frac{\mathrm{d}y(t)}{\mathrm{d}t} + \frac{(\Delta t)^2}{2!}\frac{\mathrm{d}^2 y(t)}{\mathrm{d}^2 t} + \frac{(\Delta t)^3}{3!}\frac{\mathrm{d}^3 y(t)}{\mathrm{d}^3 t} + O((\Delta t)^4)$. Now we use $\left(\frac{\mathrm{d}y}{\mathrm{d}t}\right)^c (t+\Delta t) = f(y^p(t+\Delta t))$
  - The error is defined as $\delta = \left(\frac{\mathrm{d}y}{\mathrm{d}t}\right)^c (t+\Delta t) - \left(\frac{\mathrm{d}y}{\mathrm{d}t}\right)^p (t+\Delta t)$
  - To get a complete corrector we have to adapt the function itself, its first, its second and its third derivative, too:

$$y^c(t+\Delta t) = y^p + c_0\delta$$
$$\left(\frac{\mathrm{d}^2 y}{\mathrm{d}^2 t}\right)^c (t+\Delta t) = \left(\frac{\mathrm{d}^2 y}{\mathrm{d}^2 t}\right)^p (t+\Delta t) + c_2\delta$$
$$\left(\frac{\mathrm{d}^3 y}{\mathrm{d}^3 t}\right)^c (t+\Delta t) = \left(\frac{\mathrm{d}^2 y}{\mathrm{d}^2 t}\right)^p (t+\Delta t) + c_3\delta$$

    with the so-called Gear coefficients $c_0 = 3/8, c_2 = 3/4, c_3 = 1/6$. The Gear coefficients can be read from a precalculated table.

    * $c_1 = 1$, since the first derivative of the corrector is not listed, i.e. is 0

Jacobi und Gauss-Seidel relaxation

Jacobi relaxation

- We want to solve $A\vec{\Phi} = \vec{b}$
- Decompose $A = D + U + L$ where D is the diagonal matrix of A, L the lower triangular matrix of A (without the diagonal) and U is the upper triangular matrix (without the diagonal)
- Jacobi method: $\vec{\Phi}(t+1) = D^{-1}(\vec{b} - (U+L)\vec{\Phi}(t))$
  - Not very exact, and the exact solution is only reached if $t \to \infty$

  - Define required precision ϵ and stop when $\delta'_n = \frac{\|\vec{\Phi}(t+1)-\vec{\Phi}(t)\|}{\|\vec{\Phi}(t)\|} \leq \epsilon$
  - Error is defined as $\vec{\delta}(t+1) = -D^{-1}(U+L)\vec{\delta}(t) = -\Lambda\vec{\delta}(t)$ with Λ being the error evolution operator
    * For k time steps we may write the approx. solution as the sum of the exact solution with an error: $\vec{\Phi}_k \approx \vec{\Phi}^* + \lambda^n \cdot \vec{c}$, with λ being the largest eigenvalue of Λ

Gauss-Seidel relaxation

- Jacobi method computes the new values based on the old ones
- Gauss-Seidel method will simply calculate the value of the function at a given site using all adjacent sites (does not care if sites are updated or not)
- Decompose matrix A in the same way as before, i.e. $A = D + U + L$. Now combine elements in a different way: $\vec{\Phi}(t+1) = (D+U)^{-1}(\vec{b} - L\vec{\Phi}(t))$
  - The error evolution operator Λ for the Gauss-Seidel relaxation is defined as $\Lambda = (D+U)^{-1}L$
    * $(D+U)^{-1}$ makes the largest eigenvalue λ_{max} of Λ smaller, consequently decreasing the error at each time step and increasing the convergence speed of the method
  - Stopping criteria (using precision ϵ): $\delta_n = \frac{\|\vec{\Phi}(t+1)-\vec{\Phi}(t)\|}{(1-\lambda)\|\vec{\Phi}(t)\|} \leq \epsilon$ with λ being the largest eigenvalue of Λ

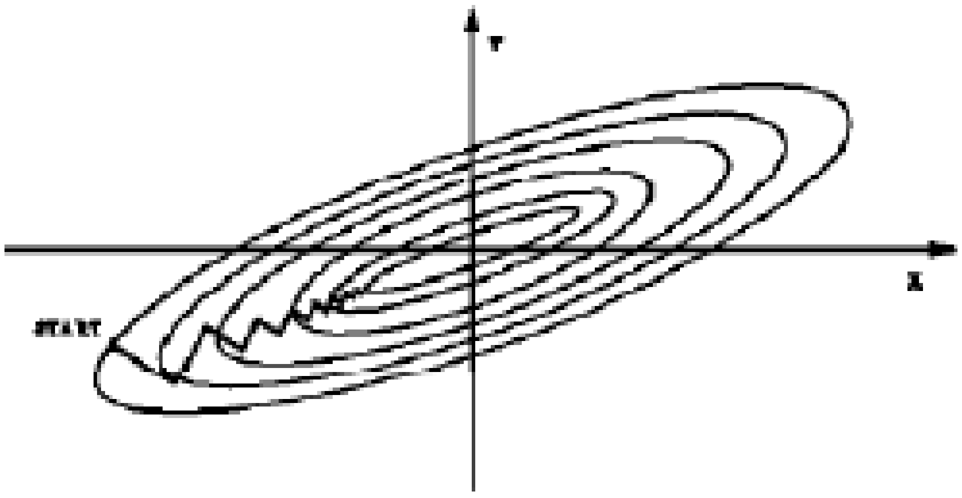
Gradient methods

- Use a functional which measures the error of a solution of the system of equations. If a system of equations has one unique solution, the functional is a paraboloid with its minimum at the exact solution.
- Define functional by residual $\vec{r}$ (estimation of the error $\vec{\delta}$): $\vec{r} = A\vec{\delta} = b - A\vec{\Phi}$
  - If the residual is small, then the error is also going to be small
  - We minimize the functional $\mathfrak{J} = \vec{r}^T A^{-1}\vec{r}$ which is 0 if $\vec{\Phi} = \vec{\Phi}^*$ and > 0 otherwise.
  - $\vec{\Phi}_i$ is the i-th approximation of the solution and let us define $\vec{\Phi} = \vec{\Phi}_i + \alpha_i\vec{d}_i$, where $\vec{d}_i$ is the direction of the step and α_i the step length

- $*$ If we minimize the functional $\mathfrak{J}$ along the lines given by $\vec{d}$ we will find the best value for α (denoted as α_{best}) at which $\mathfrak{J}$ is minimal, i.e. $\frac{\partial \mathfrak{J}}{\partial \alpha} = 2\vec{d}_i(\alpha_{\text{best},i} A\vec{d}_i - \vec{r}) \overset{!}{=} 0 \Longleftrightarrow \alpha_{\text{best},i} = \frac{\vec{d}_i^T \vec{r}_i}{\vec{d}_i^T A \vec{d}_i}$
- $*$ $\alpha_{\text{best},i}$ has to be calculated for each step.

Steepest descent

- Analogy of the steepest descent method: trying to get down to the valley from the mountain top the quickest way possible. To do this, chose the steepest possible direction. It's not the most comfortable way, but it gets the job done.
- For the steepest descent method choose $\vec{d}_i = \vec{r}_i$
- Disadvantage: it does not take the optimal direction if the functional is not a regular paraboloid:



- Steepest descent algorithm:
  1. Start with $\vec{\Phi}_i$ and choose $\vec{d}_i = \vec{r}_i = \vec{b} - A\vec{\Phi}_i$
  2. Evaluate $\vec{u}_i = A\vec{r}_i$ and store $\vec{r}_i$. Calculate the length of the step: $\alpha_i = \frac{\vec{r}_i^2}{\vec{r}_i \vec{u}_i}$
  3. Advance in the direction of $\vec{d}_i$ for the length α_i: $\vec{\Phi}_{i+1} = \vec{\Phi}_i + \alpha_i \vec{r}_i$
  4. Update the residual for the next step: $\vec{r}_{i+1} = \vec{r}_i - \alpha_i \vec{u}_i$
  5. Repeat steps until the residual is sufficiently small

- Most costly part of the algorithm is the calculation of $\vec{u}$ ($O(N^2)$ if A is a dense matrix, $O(N)$ if A is a sparse matrix where N is the number of equations to be solved)

Conjugate Gradient

- The conjugate gradient method fixes the problem of having a functional that is a irregular paraboloid
- Conjugate gradient takes the functional and deforms it in such a way that it looks like a regular paraboloid, and only then it carries out the steepest descent method. We get rid of irregular paraboloids in this way.
- New direction is conjugate to all previous ones by using Gram-Schmidt orthogonalization process: $\vec{d}_i = \vec{r}_i - \sum_{j=1}^{i-1} \frac{\vec{d}_j^T A \vec{r}_i}{\vec{d}_j^T A \vec{d}_j} \vec{d}_j$. Now all directions $\vec{d}_i$ are conjugate to each other, i.e. $\vec{d}_i^T A \vec{d}_j = \delta_{ij}$ with $\delta_{ij} = 0$ if $i \neq j$, else $\delta_{ij} = 1$
- **Important note:** Conjugate gradient method and gradient methods in general are only a valuable option for positive-definite and symmetric matrices A, since this implies that the eigenvalues of A will be real, i.e. $\lambda_i \in \mathbb{R}$
- Conjugate gradient algorithm:
  1. Compute first residual $\vec{r}_1$ by using some vector $\vec{\Phi}_1$. Then use the residual as the first direction, i.e. $\vec{d}_1 = \vec{r}_1 = \vec{b} - A\vec{\Phi}_1$
  2. Compute the temporary scalar $c = (\vec{d}_i^T A \vec{d}_i)^{-1}$ (do not inverse A; first calculate the scalar product, resulting in the inversion of a scalar value)
  3. Compute the length of the step: $\alpha_i = c\vec{r}_i^T \vec{d}_i$
  4. Carry out the step: $\vec{\Phi}_{i+1} = \vec{\Phi}_i + \alpha_i \vec{d}_i$
     - If the residual is sufficiently small, e.g. $\vec{r}_i^T \vec{r}_i < \epsilon$ we can stop
  5. Update the residula for the error estimation and for the next step: $\vec{r}_{i+1} = \vec{b} - A\vec{\Phi}_{i+1}$
  6. Compute the direction of the next step: $\vec{d}_{i+1} = \vec{r}_{i+1} - (c\vec{r}_{i+1}^T A \vec{d}_i)\vec{d}_i$

Startegy of finite elements, finite volumes and spectral methods

- Problem of finite differences: it is not possible to take region-dependent mesh sizes, since the regular mesh cannot be adapted to the problem.

- With finite element methods, PDEs can be solved for irregular geometries, inhomogeneous fields (e.g. with moving boundaries) and non-linear PDEs. The mesh can be adapted during the computation of the solution to speed up convergence.
  - Convergence depends on the ratio of gradient to mesh size, so it's preferable to refine the mesh locally in regions of big gradients
- Poisson equation: $\frac{\mathrm{d}^2\Phi}{\mathrm{d}^2 x}(x) = -4\pi\rho(x)$. Try to expand the field Φ in terms of localized basis functions $u_i(x)$: $\Phi(x) = \sum_{i=1}^{\infty} a_i u_i(x) \approx \Phi_N(x) = \sum_{i=1}^{N} a_i u_i(x)$