Simulating the Ising model using the Metropolis algorithm

Johan Nereng

Department of Physics, University of Oslo, Norway

Nov 21, 2018

1 Abstract

2 Introduction

For the purpose of approximating the derivatives of u, three methods are applied: explicit forward Euler, implicit Backward Euler, and the implicit Crank-Nicolson with a time-centered scheme.

3 Methods

$$\nabla^2 u(x,t) = \frac{\partial u(x,t)}{\partial t} \tag{1}$$

$$u_x x = u_t \tag{2}$$

Initial conditions, boundry conditions

3.1 Discretization and substitution of spatial variable

In order to numerically solve (2), the problem is discretized. The spacial domain, $x \in [0, L]$, is discretized over $n+1$ grid points, such that $x_i = i\Delta x$, $i = 0, 1, ..., n+1$, where $\Delta x = \frac{1}{n+1}$. Similarly, $t = [0, T]$, is is expressed as $t_j = j\Delta t$, $j \geq 0$. The dizcrete approximation of the function u is then defined as $u(x_i, t_j) = u_{i,j}$, with boundary conditions (B.C) $u_{0,j} = 0$ and $u_{n+1,j} = 1$.

The solution to the problem is assumed to be u. In order to extract it's numerical approximation more easily, a change of variables from u to v is applied:

$v_{i,j} = u_{i,j} - x_i$. Recalling the initial conditions (I.C), $u_{i,0} = 0$, for $i < L$, new I.Cs are now obtained: $v_{i,0} = u_{i,0} - x_i$, as well as the B.Cs:

$$v_{0,j} = v_{1,j} = 0$$

3.2 Approximations methods

Using the discretized equation v, the standard approximation for derivatives is now applied, yielding [1]:

$$u_t \approx \frac{u(x_i, t_j + \Delta t) - u(x_i, t_j)}{\Delta t} \tag{3}$$

$$u_{xx} \approx \frac{u(x_i + \Delta x, t_j) - 2u(x_i, t_j) + u(x_i - \Delta x, t_j)}{\Delta x^2} \tag{4}$$

$$\tag{5}$$

Applying compact notation introduced in the discretization:

$$u_t \approx \frac{u_{i,j+1} - u_{i,j}}{\Delta t} \tag{6}$$

$$u_{xx} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} \tag{7}$$

$$\tag{8}$$

Where u_t has local truncation error (LTE) $O(\Delta t)$ and $u_x x$ has $O(\Delta x^2)$.

Setting $u_t = u_{xx}$, defining $\alpha = \Delta t / \Delta x^2$, and solving for $v_{i,j+1}$ yields what is known as the explicit scheme (ref lec notes):

$$v_{i,j+1} = \alpha v_{i-1,j} + (1 - 2\alpha) v_{i,j} + \alpha v_{i+1,j} \tag{9}$$

The terms v_0 and v_{n+1} are now excluded, as they are the B.C's, and thereby part of the solution.

$$V_j = \begin{bmatrix} v_{1,i} \\ v_{2,i} \\ \dots \\ v_{n,i} \end{bmatrix} \tag{10}$$

Which corresponds to the following matrix product:

$$V_{j+1} = \mathbf{A} V_j \tag{11}$$

In which $\mathbf{A} = \begin{bmatrix} 1-2\alpha & \alpha & 0 & \dots & \dots & 0 \\ \alpha & 1-2\alpha & \alpha & 0 & \dots & \dots \\ & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & & 0 & \alpha & 1-2\alpha \end{bmatrix}$

Skriv ut ligningene

$V_{j+1} = \mathbf{A}V_h = \mathbf{A}\mathbf{A}V_{j-1} = ... = \mathbf{A}^{j+1}V_0$

implicit backward euler: Similarly, implicit backward euler:

$$u_t \approx \frac{u_{i,j} - u_{i,j-1}}{\Delta t} \tag{12}$$

$$u_{xx} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} \tag{13}$$

$$\tag{14}$$

3.2.1 Approximations methods and their matrix representations

The second derivatives **Explicit forward Euler** (assignment paper/lecture notes):

$$u_t \approx \frac{u(x_i, t_j + \Delta t) - u(x_i, t_j)}{\Delta t} \tag{15}$$

$$u_{xx} \approx \frac{u(x_i + \Delta x, t_j) - 2u(x_i, t_j) + u(x_i - \Delta x, t_j)}{\Delta x^2} \tag{16}$$

$$\tag{17}$$

Applying compact notation introduced in the discretization:

$$u_t \approx \frac{u_{i,j+1} - u_{i,j}}{\Delta t} \tag{18}$$

$$u_{xx} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} \tag{19}$$

$$\tag{20}$$

Where u_t has local truncation error (LTE) $O(\Delta t)$ and $u_x x$ has $O(\Delta x^2)$.

Setting $u_t = u_{xx}$, defining $\alpha = \Delta t/\Delta x^2$, and solving for $v_{i,j+1}$ yields what is known as the explicit scheme (ref lec notes):

$$v_{i,j+1} = \alpha v_{i-1,j} + (1 - 2\alpha)v_{i,j} + \alpha v_{i+1,j} \tag{21}$$

The terms v_0 and v_{n+1} are now excluded, as they are the B.C's, and thereby part of the solution.

$$V_j = \begin{bmatrix} v_{1,i} \\ v_{2,i} \\ ... \\ v_{n,i} \end{bmatrix} \tag{22}$$

Which corresponds to the following matrix product:

$$V_{j+1} = \mathbf{A}V_j \tag{23}$$

In which $\mathbf{A} = \begin{bmatrix} 1-2\alpha & \alpha & 0 & \dots & \dots & 0 \\ \alpha & 1-2\alpha & \alpha & 0 & \dots & \dots \\ & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & & 0 & \alpha & 1-2\alpha \end{bmatrix}$

Skriv ut ligningene

$V_{j+1} = \mathbf{A}V_h = \mathbf{AA}V_{j-1} = ... = \mathbf{A}^{j+1}V_0$

For a specific number of steps, n, (**??**) yields a set of n linear equations;

$$
\begin{aligned}
-\frac{v_2 + v_0 - 2v_1}{h^2} &= f_1 \\
-\frac{v_3 + v_1 - 2v_2}{h^2} &= f_2 \\
&\cdots\cdots \\
-\frac{v_{i+1} + v_{i-1} - 2v_i}{h^2} &= f_i \\
&\cdots\cdots \\
-\frac{v_{n+1} + v_{n-1} - 2v_n}{h^2} &= f_n
\end{aligned}
$$

The equations above may be represented by matrices. The terms v_0 and v_{n+1}, in the the first and n'th equations, are excluded, as they are the B.C's, and thereby part of the solution.

$$
\frac{1}{h^2}\begin{bmatrix} -v_2 - v_0 + 2v_1 \\ -v_3 - v_1 + 2v_2 \\ \cdots\cdots \\ -v_{i+1} - v_{i-1} + 2v_i \\ \cdots\cdots \\ -v_{n+1} - v_{n-1} + 2v_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \cdots \\ f_i \\ \cdots \\ f_n \end{bmatrix}.
$$

Which corresponds to the following matrix product

$$
\mathbf{Av} = \begin{bmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & \dots \\ 0 & -1 & 2 & -1 & 0 & \dots \\ & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & & -1 & 2 & -1 \\ 0 & \dots & & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ \dots \\ \dots \\ v_n \end{bmatrix} = \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \dots \\ \dots \\ \dots \\ \tilde{b}_n \end{bmatrix} . = \tilde{\mathbf{b}}
$$

Where $\mathbf{A}$ contains the coefficients of the left hand side of the set of equations, $\mathbf{v}$ holds the unknown entities (in this case, the value of the function), and $\tilde{\mathbf{b}}$ holds the solutions $b_i = h^2 f_i$.

3.2.2 Reducing complexity of equations

In general, a set of linear equations may be simplified by Gaussian Elimination, which corresponds to bringing the augmented matrix representing the coefficients

from the set of equations and the vector holding the solutions (right hand side) side to row echelon form.

In the specific case described above the non-diagonal elements are identical, so the the matrix $\mathbf{A}$ is a tridiagonal Toeplitz matrix. In order to produce a more general numerical solver, the Gaussian Elimination below and the algorithms based upon the result, will be founded upon an unspecified $n \times n$ tridiagonal $\mathbf{A}$ and a $1 \times n$ column vector, $\mathbf{b}$.

$$[\mathbf{Ab}] = \mathbf{B} = \begin{bmatrix} d_1 & c_1 & 0 & \dots & \dots & \dots & b_1 \\ a_1 & d_2 & c_2 & \dots & \dots & \dots & b_2 \\ & a_2 & d_3 & c_3 & \dots & \dots & \dots \\ & \dots & \dots & \dots & \dots & \dots & \dots \\ & & & a_{n-2} & d_{n-1} & c_{n-1} & \dots \\ & & & & a_{n-1} & d_n & b_n \end{bmatrix} \sim$$

$$\tilde{\mathbf{B}} = \begin{bmatrix} \tilde{d_1} & c_1 & 0 & \dots & \dots & \dots & \tilde{b_1} \\ 0 & \tilde{d_2} & c_2 & \dots & \dots & \dots & \tilde{b_2} \\ & 0 & d_3 & c_3 & \dots & \dots & \dots \\ & \dots & \dots & \dots & \dots & \dots & \dots \\ & & & 0 & \tilde{d_{n-1}} & c_{n-1} & \dots \\ & & & & 0 & \tilde{d_n} & \tilde{b_n} \end{bmatrix} = [\tilde{\mathbf{A}}\mathbf{b}]$$

Where

$$\tilde{d_i} = \begin{cases} d_i - \frac{c_{i-1}a_{i-1}}{\tilde{d}_{i-1}}, , & \text{for i=2,3,...,n} \\ d_i, & \text{for i=1} \end{cases} \tag{24}$$

$$\tilde{b_i} = \begin{cases} b_i - \frac{\tilde{b}_{i-1}a_{i-1}}{\tilde{d}_{i-1}}, , & \text{for i=2,3,...,n} \\ b_i, & \text{for i=1} \end{cases} \tag{25}$$

In order to derive an algorithm from the equation coefficients, a $1 \times n$ vector, $\mathbf{v}$, holding the variables in the equations, is introduced. Similar to before, but in this case using the row echelon form of the general tridiagonal matrix and general vectors, the matrix product representing the (general) equations can be written as;

$$\mathbf{A} = \begin{bmatrix} \tilde{d_1} & c_1 & 0 & \dots & \dots & \dots \\ 0 & \tilde{d_2} & c_2 & \dots & \dots & \dots \\ & 0 & d_3 & c_3 & \dots & \dots \\ & \dots & \dots & \dots & \dots & \dots \\ & & & 0 & \tilde{d_{n-1}} & c_{n-1} \\ & & & & 0 & \tilde{d_n} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ \dots \\ \dots \\ v_n \end{bmatrix} = \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_2 \\ \dots \\ \dots \\ \dots \\ \tilde{b}_n \end{bmatrix}.$$

The matrix product directly above yields the relation $v_n\tilde{d_n} = \tilde{b}_n$. Since $\tilde{d}_{n-1}v_{n-1} + c_{n_1}v_n = \tilde{b}_{n-1}$ and so on, v_i may be backwards substituted:

$$v_i = \begin{cases} \frac{\tilde{b}_n}{\tilde{d}_n}, & \text{for i=n} \\ \frac{\tilde{b}_i - c_{i-1}v_{i-1}}{\tilde{d}_i}, , & \text{for i=1,2,...,n-1} \end{cases} \tag{26}$$

4 Algorithms

[2]

References

[1] Morten Hjorth-Jensen. Lecture notes fall 2015: Lecture notes fall 2015, Aug 2015.

[2] Conrad Sanderson and Ryan Curtin. Armadillo: a template-based c++ library for linear algebra. http://arma.sourceforge.net/docs.html#part_classes, 2016.