Assignment 5
Numerical solution of a non-linear coupled reaction-diffusion problem

Numerical Analysis For PDE's (WI3730TU)
Kaloyan Yanchev

October 2024

Schnakenberg Model

$$\begin{cases} \frac{\partial u}{\partial t} = D_u \triangle u + k(a - u + u^2 v) \\ \frac{\partial v}{\partial t} = D_v \triangle v + k(b - u^2 v) \end{cases} \quad (x,y) \in \Omega, t \in (0,T]$$

$$\begin{cases} u(x,y,0) = u_0(x,y) \\ v(x,y,0) = v_0(x,y) \end{cases} \quad (x,y) \in \Omega$$

$$\begin{cases} -D_u \nabla u \cdot \mathbf{n} = 0 \\ -D_v \nabla v \cdot \mathbf{n} = 0 \end{cases} \quad (x,y) \in \partial\Omega$$

$D_u = 0.05, D_v = 1.0, k = 5, a = 0.1305, b = 0.7695$
$u_0(x,y) = a + b + r(x,y)$
$v_0(x,y) = \frac{b}{(a+b)^2}$
$\Omega = (0,4) \times (0,4), T = 20$

1. Finite-Difference Method

   (a) Approximation of inner points:

$$-\triangle u_{i,j} = \frac{4u_{i,j} - u_{i-1,j} - u_{i,j-1} - u_{i+1,j} - u_{i,j+1}}{h^2}$$

Neumann Boundary Condition states that the dot product of the gradient of the function around the boundary with the normal vector of the boundary at that point is 0. Therefore this can be rephrased as the derivative of the function value at the boundary with respect to the normal of the boundary at that point is 0. Next, it should be used that the boundary surface is a square, therefore boundary surface normals are: left boundary - negative x direction; right boundary - positive x direction; bottom boundary - negative y direction; top boundary - positive y direction; respectively $\begin{bmatrix} -1 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ -1 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$. Since the boundary normal vectors are only x or y oriented, the Neumann Boundary Condition can be simplified to $u_{boundary} = u_{adjacent}$. Let's consider a single boundary to derive the general rule to properly substitute $u_{boundary} = u_{adjacent}$ in this problem.

Left Boundary ($i = 0, 1 \le j \le N-1$):

$$\mathbf{n_{left}} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

$$\nabla u_{0,j} \cdot \mathbf{n_{left}} = 0$$

$$=> -1 * \frac{\partial u_{0,j}}{\partial x} + 0 * \frac{\partial u_{0,j}}{\partial y} = 0$$

$$=> \frac{\partial u_{0,j}}{\partial x} = D_x u_{0,j} = 0$$

In order to estimate the value of $D_x u_{0,j}$, Finite-Difference approximations should be considered:

$$\begin{cases} D_x^- u_{0,j} = \frac{u_{0,j} - u_{-1,j}}{h} \\ D_x^+ u_{0,j} = \frac{u_{1,j} - u_{0,j}}{h} \\ D_x^c u_{0,j} = \frac{u_{1,j} - u_{-1,j}}{2h} \end{cases}$$

Since the left and central finite-difference derivatives rely on $u_{-1,j}$ which is not part of the system, it would be intuitive to rely on the right (one-sided forward) finite-difference derivative.

$$D_x^+ u_{0,j} = \frac{u_{1,j} - u_{0,j}}{h} = 0$$

$$=> u_{0,j} = u_{1,j}$$

$$u_{boundary} = u_{adjacent}$$

Let's take a look back at the inner point approximation:

$$-\triangle u_{i,j} = \frac{4u_{i,j} - u_{i-1,j} - u_{i,j-1} - u_{i+1,j} - u_{i,j+1}}{h^2}$$

$$=> -\triangle u_{1,j} = \frac{4u_{1,j} - u_{0,j} - u_{1,j-1} - u_{2,j} - u_{1,j+1}}{h^2} = \frac{3u_{1,j} - u_{1,j-1} - u_{2,j} - u_{1,j+1}}{h^2}$$

Thus for inner points whose approximation depends on boundary point values, those boundary point values are substituted with the value of the inner point for which the approximation is derived. Now the matrix A of the negative 2D FD Laplacian with zero Neumann BC's can be derived for N(internal points in direction) = 3, in order to examine doubly boundary dependent, singly, and boundary independent (doubly inner points) approximations of inner points.

$$A = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 3 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 3 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$

(b)

$$D_x = D_y = \frac{1}{h}\begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

$$=> D_x^T = D_y^T = \frac{1}{h}\begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 0 & -1 \end{bmatrix}$$

$$=> A_{xx} = D_x^T D_x = D_y^T D_y = A_{yy} = \frac{1}{h^2}\begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

$$=> I_y \otimes A_{xx} + A_{yy} \otimes I_x = A$$

So $D_x, D_y \in \mathbb{R}^{(N)\times(N-1)}$ with 1s on the main diagonal and -1s on the right from the main one.

(c)

$$\mathbf{u}' = \mathbf{f_u}(\mathbf{u}, \mathbf{v})$$

$$\mathbf{v}' = \mathbf{f_v}(\mathbf{u}, \mathbf{v})$$

$$=> \begin{cases} \mathbf{u}' = \mathbf{f_u}(\mathbf{u}, \mathbf{v}) = -D_u A\mathbf{u} + k(a - \mathbf{u} + \mathbf{u}^2 \circ \mathbf{v}) \\ \mathbf{v}' = \mathbf{f_v}(\mathbf{u}, \mathbf{v}) = -D_v A\mathbf{v} + k(b - \mathbf{u}^2 \circ \mathbf{v}) \end{cases}$$

2. Numerical Time-Integration

(a)

$$\begin{cases} \mathbf{u}' = -D_u A\mathbf{u} + k(a - \mathbf{u} + \mathbf{u}^2 \circ \mathbf{v}) \\ \mathbf{v}' = -D_v A\mathbf{v} + k(b - \mathbf{u}^2 \circ \mathbf{v}) \end{cases}$$

Forward-Euler:

$$=> \begin{cases} \mathbf{u}^{k+1} = \mathbf{u}^k + h[-D_u A\mathbf{u}^k + k(a - \mathbf{u}^k + (\mathbf{u}^k)^2 \circ \mathbf{v}^k)] \\ \mathbf{v}^{k+1} = \mathbf{v}^k + h[-D_v A\mathbf{v}^k + k(b - (\mathbf{u}^k)^2 \circ \mathbf{v}^k)] \end{cases}$$

(b) $\frac{\partial u}{\partial t} = \triangle u - ku, k > 0$, zero Dirichlet BC's

$$\mathbf{u}' = -A\mathbf{u} - ku = -(A + kI)\mathbf{u}$$

$$=> \mathbf{u}^m = \mathbf{u}^{m-1} + h[-(A + kI)\mathbf{u}^{m-1}] = [(1 - kh)I - hA]\mathbf{u}^{m-1} = [(1 - kh)I - hA]^m\mathbf{u}^0$$

$$\mathbf{u}^m = \sum_{j=1}^{n} b_j(0)[1 - h(k + \lambda_j)]^m\mathbf{v}_j \qquad \text{(method of lines)}$$

CFL Stability Condition:

$$|1 - h(k_{max} + \lambda_{max})| < 1$$

$$-1 < 1 - h(k_{max} + \lambda_{max}) < 1$$

$$-2 < -h(k_{max} + \lambda_{max}) < 0$$

$$0 < h(k_{max} + \lambda_{max}) < 2$$

$$=> 0 < h < \frac{2}{k_{max} + \lambda_{max}}$$

(c) Backward-Euler:

$$\begin{cases} \mathbf{u}^{k+1} = \mathbf{u}^k + h[-D_uA\mathbf{u}^{k+1} + k(a + \mathbf{u}^{k+1} + (\mathbf{u}^{k+1})^2 \circ \mathbf{v}^{k+1})] \\ \mathbf{v}^{k+1} = \mathbf{v}^k + h[-D_vA\mathbf{v}^{k+1} + k(b - (\mathbf{u}^{k+1})^2 \circ \mathbf{v}^{k+1})] \end{cases}$$

3. Newton-Raphson Algorithm

(a)

$$\begin{cases} \mathbf{u}^{k+1} = h\mathbf{f}_u(\mathbf{u}^{k+1}, \mathbf{v}^{k+1}) + \mathbf{u}^k, & \mathbf{f}_u(\mathbf{u}, \mathbf{v}) = -D_uA\mathbf{u} + k(a - \mathbf{u} + \mathbf{u}^2 \circ \mathbf{v}) \\ \mathbf{v}^{k+1} = h\mathbf{f}_v(\mathbf{u}^{k+1}, \mathbf{v}^{k+1}) + \mathbf{v}^k, & \mathbf{f}_v(\mathbf{u}, \mathbf{v}) = -D_vA\mathbf{v} + k(b - \mathbf{u}^2 \circ \mathbf{v}) \end{cases}$$

Residuals:

$$\begin{cases} ||\mathbf{u}^k + h\mathbf{f}_u(\mathbf{u}_i^{k+1}, \mathbf{v}_i^{k+1}) - \mathbf{u}_i^{k+1}|| \\ ||\mathbf{v}^k + h\mathbf{f}_v(\mathbf{u}_i^{k+1}, \mathbf{v}_i^{k+1}) - \mathbf{v}_i^{k+1}|| \end{cases}$$

Iteration:

$$\begin{bmatrix} \mathbf{u}_{i+1}^{k+1} \\ \mathbf{v}_{i+1}^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_i^{k+1} \\ \mathbf{v}_i^{k+1} \end{bmatrix} + [I - hJ]^{-1} \cdot \begin{bmatrix} \mathbf{u}^k + h\mathbf{f}_u(\mathbf{u}_i^{k+1}, \mathbf{v}_i^{k+1}) - \mathbf{u}_i^{k+1} \\ \mathbf{v}^k + h\mathbf{f}_v(\mathbf{u}_i^{k+1}, \mathbf{v}_i^{k+1}) - \mathbf{v}_i^{k+1} \end{bmatrix}$$

(b)

$$J(\mathbf{u}, \mathbf{v}) = \begin{bmatrix} \frac{\partial \mathbf{f}_u(\mathbf{u},\mathbf{v})}{\partial \mathbf{u}} & \frac{\partial \mathbf{f}_u(\mathbf{u},\mathbf{v})}{\partial \mathbf{v}} \\ \frac{\partial \mathbf{f}_v(\mathbf{u},\mathbf{v})}{\partial \mathbf{u}} & \frac{\partial \mathbf{f}_v(\mathbf{u},\mathbf{v})}{\partial \mathbf{v}} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{u}}[-D_uA\mathbf{u} + k(a - \mathbf{u} + \mathbf{u}^2 \circ \mathbf{v})] & \frac{\partial}{\partial \mathbf{v}}[-D_uA\mathbf{u} + k(a - \mathbf{u} + \mathbf{u}^2 \circ \mathbf{v})] \\ \frac{\partial}{\partial \mathbf{u}}[-D_vA\mathbf{v} + k(b - \mathbf{u}^2 \circ \mathbf{v}]) & \frac{\partial}{\partial \mathbf{v}}[-D_vA\mathbf{v} + k(b - \mathbf{u}^2 \circ \mathbf{v})] \end{bmatrix}$$

For arbitrary p and q:

$$\begin{bmatrix} \frac{\partial}{\partial u_q}[D_u \sum_{i=1}^n (a_{p,i}u_i) + k(a - u_p + u_p^2 v_p)] & \frac{\partial}{\partial v_q}[D_u \sum_{i=1}^n (a_{p,i}u_i) + k(a - u_p + u_p^2 v_p)] \\ \frac{\partial}{\partial u_q}[D_v \sum_{i=1}^n (a_{p,i}v_i) + k(b - u_p^2 v_p)] & \frac{\partial}{\partial v_q}[D_v \sum_{i=1}^n (a_{p,i}v_i) + k(b - u_p^2 v_p)] \end{bmatrix}$$

$$\begin{bmatrix} D_u \sum_{i=1}^n (-a_{p,i}\frac{\partial u_i}{\partial u_q}) + k(-\frac{\partial u_p}{\partial u_q} + 2u_pv_p\frac{\partial u_p}{\partial u_q}) & ku_p^2\frac{\partial v_p}{\partial v_q} \\ -2ku_pv_p\frac{\partial u_p}{\partial u_q} & D_v \sum_{i=1}^n (-a_{p,i}\frac{\partial v_i}{\partial v_q}) - ku_p^2\frac{\partial v_p)}{\partial v_q} \end{bmatrix}$$

$$\begin{bmatrix} D_u \sum_{i=1}^n (-a_{p,i}\delta_{i,q}) + k(-\delta_{p,q} + 2u_pv_p\delta_{p,q}) & ku_p^2\delta_{p,q} \\ -2ku_pv_p\delta_{p,q} & D_v \sum_{i=1}^n (-a_{p,i}\delta_{i,q}) - ku_p^2\delta_{p,q} \end{bmatrix}$$

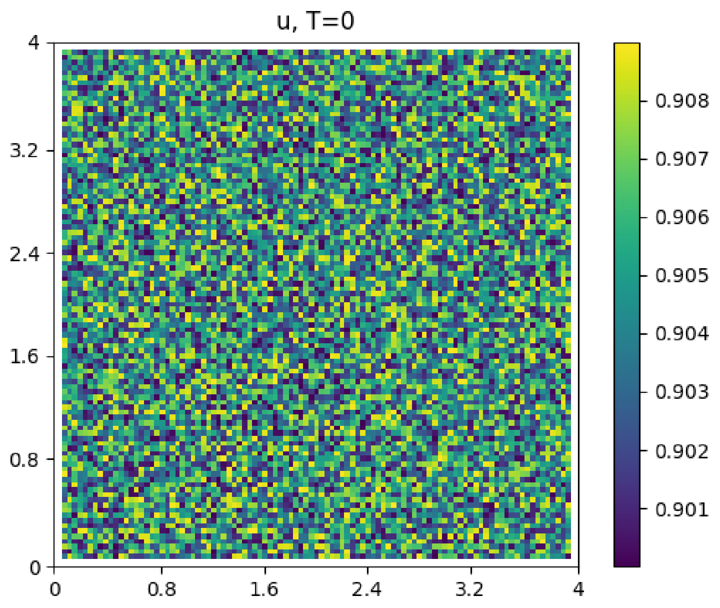
$$\begin{bmatrix} -D_ua_{p,q} + k(-\delta_{p,q} + 2u_pv_p\delta_{p,q}) & ku_p^2\delta_{p,q} \\ -2ku_pv_p\delta_{p,q} & -D_va_{p,q} - ku_p^2\delta_{p,q} \end{bmatrix}$$

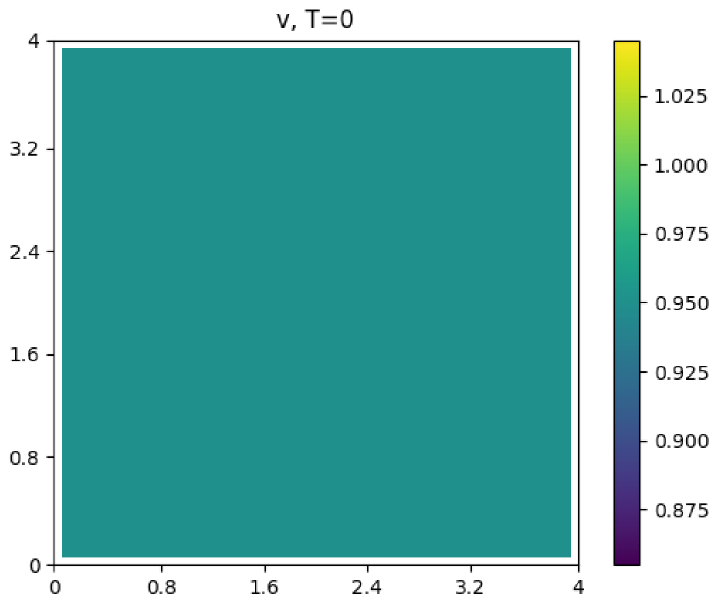
In the general case, $\delta_{p,q} -> I$, $a_{p,q} -> A$. $(diag(\mathbf{x}) = I \circ \mathbf{x})$

$$=> J(\mathbf{u}, \mathbf{v}) = \begin{bmatrix} -D_uA + k[\text{diag}(2\mathbf{u} \circ \mathbf{v}) - I] & k \text{ diag}(\mathbf{u}^2) \\ -2k \text{ diag}(\mathbf{u} \circ \mathbf{v}) & -D_vA - k \text{ diag}(\mathbf{u}^2) \end{bmatrix}$$

4. Implementation

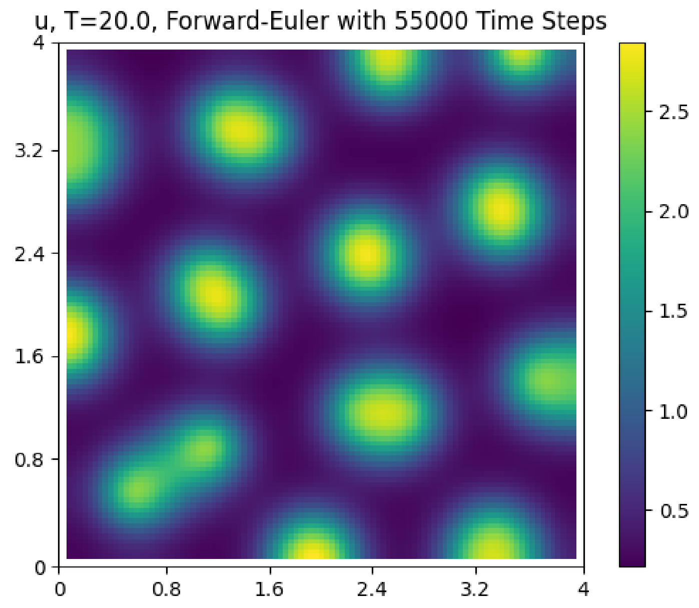
   Note: values of u and v are intentionally visualized with margin around to account for the fact that approximation are based on inner grid points only. Boundary points are not included since due to Neumann Boundary Conditions, those would be the same as the adjacent inner points.

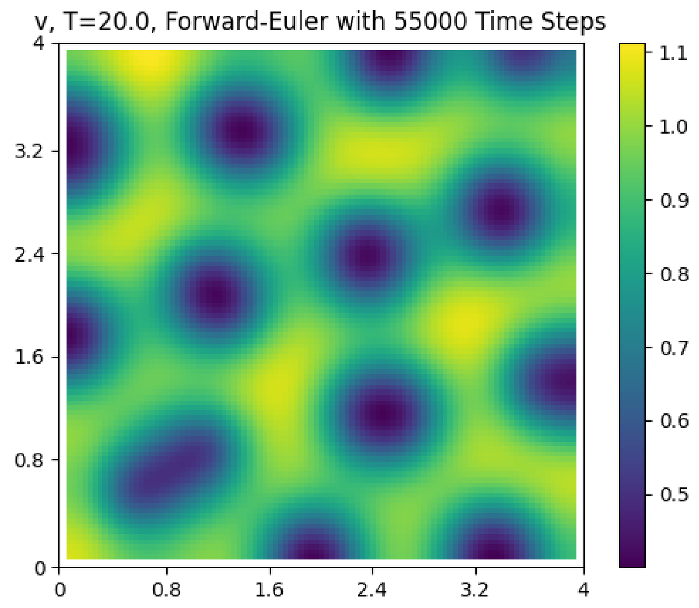




(a) Forward-Euler

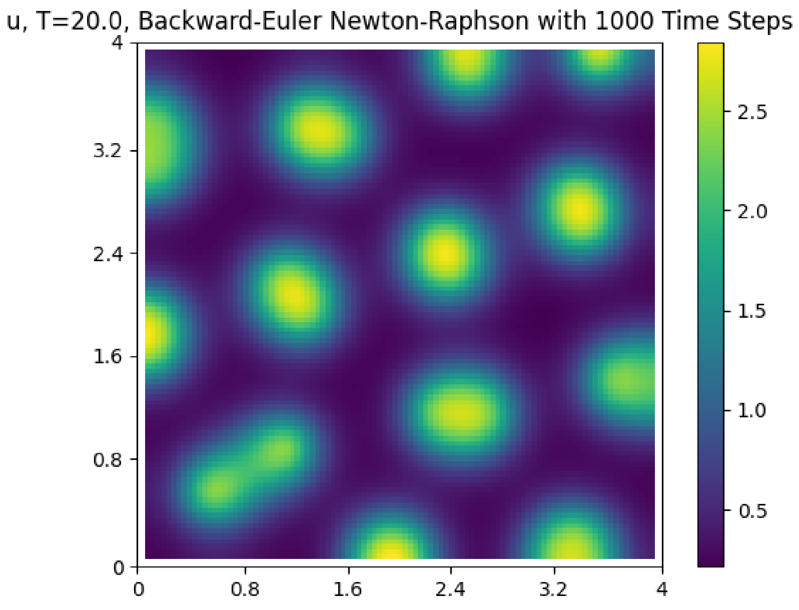
Number of time steps is based on the previous assignment approximation of $\lambda_{max} = \frac{8}{h_{FDM}^2}$. The empirical tests performed have checked for $k \in \{0, 2, 5, 10, 50, 100, 200, 500, 1000\}$. For $k \in \{0, 2, 5\}$, it can be observed how u starts to form a pattern, however, the same is not properly observed for v, so Forward-Euler is not really stable. For values of $k \geq 10$, the pattern appears to be stably the same. Further analysis shows that the difference between approximations for $k \geq 10$ are on average of magnitude of 10^{-6}, so stability is indeed reached and the main difference observed is in terms of computational time. However, for the purpose of rigorousness, results are going to be obtained using $k = 500$, thus $Nt = 55000$.

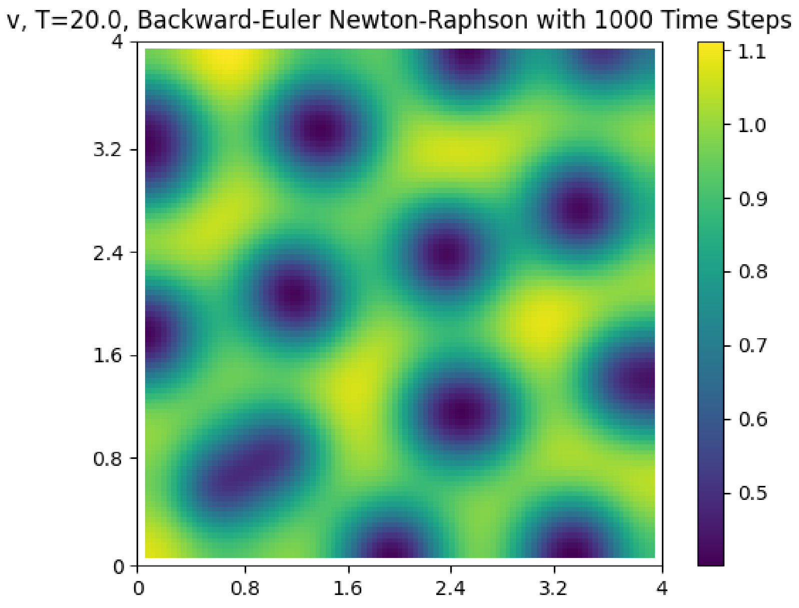




(b) Backward-Euler Newton-Raphson

The empirical tests performed have checked for $Nt \in \{10, 50, 100, 200, 500, 1000\}$. For Nt=10 or Nt=50, the pattern already appears in both u and v, however, compared to the stable Forward-Euler big differences can be observed. For Nt=100, the pattern becomes more similar to the Forward-Euler one, however, some spots seem to be slightly misplaced. Further analysis based on the difference between the Backward-Euler Newton-Raphson approximation and that of the stable Forward-Euler shows that there are indeed some significant point differences that decrease with the more time steps performed for Backward-Euler Newton-Raphson. However, it should be noted that this also requires a lot of computational time (which is also tolled by the printing of the residual norms). For Nt=1000, Backward-Euler Newton-Raphson can be considered to be convergent on the stable Forward-Euler approximation as difference drops to magnitude of 10^{-3} for both u and v, thus this approximation is going to be used as its execution is still considered decently timed (under 5 mins).


u, T=20.0, Backward-Euler Newton-Raphson with 1000 Time Steps


v, T=20.0, Backward-Euler Newton-Raphson with 1000 Time Steps