MA 580 Assignment 5

Kyle Hansen

Turned in 25 Nov. 2025

AI Use Statement: ChatGPT was used to help recall the name and properties of the quadratic form in problem 2

Question 1 GMRES Residual Estimate

An estimate for GMRES residual is given by

$$\frac{\|r_k\|}{\|r_0\|} \leq \|p_k(A)\| \tag{1}$$

for every $p_k \in \mathcal{P}_k$ ($p_k(0) = 1$). Then, for diagonizable matrix $A = V\Lambda V^{-1}$,

$$\frac{\|r_k\|}{\|r_0\|} \leq \left\|V p_k(\Lambda) V^T\right\| \leq \|V\| \, \|p_k(\Lambda)\| \, \left\|V^{-1}\right\| \tag{2}$$

and since $\|V\| \, \|V^{-1}\| = \kappa_2(V)$,

$$\frac{\|r_k\|}{\|r_0\|} \leq \kappa_2(V) \, \|p_k(\Lambda)\| = \kappa_2(V) \left\| \begin{bmatrix} p_k(\lambda_1) & & & \\ & p_k(\lambda_2) & & \\ & & \ddots & \\ & & & p(\lambda_n) \end{bmatrix} \right\| \tag{3}$$

Then, since the norm of a diagonal matrix is its largest eigenvalue,

$$\frac{\|r_k\|}{\|r_0\|} \leq \kappa_2(V) \max_{z \in \sigma(A)} |p_k(z)|. \tag{4}$$

Then an appropriate choice of polynomial p_k would be one that minimizes the the value about the eigenvalues of A. Here $(\frac{10-z}{10}\frac{20-z}{20})^{k/2}$ is chosen, since $p_k = 0$ at the center of the range of possible eivenvalues for this problem. For this choice of p_k, the maximum absolute value $|\max p_k(z)|$ for $z \subset (9, 11) \cup (19, 21)$ occurs at $z = 9$ and is equal to $(11/200)^{k/2}$. Then, for the given properties of A, the residual at iteration k can be bounded by

$$\frac{\|r_k\|}{\|r_0\|} \leq 100 \cdot \left(\frac{11}{200}\right)^{\frac{k}{2}}. \tag{5}$$

Using a relative residual of 10^{-6}, the number of iterations required is

$$10^{-6} = 100 \cdot \left(\frac{11}{200}\right)^{\frac{k}{2}} \tag{6}$$

$$\frac{10^{-6}}{100} = \left(\frac{11}{200}\right)^{\frac{k}{2}} \tag{7}$$

$$\ln 10^{-8} = \frac{k}{2} \ln \frac{11}{200} \tag{8}$$

$$2\frac{\ln 10^{-8}}{\ln \frac{11}{200}} = k = 12.702 \tag{9}$$

And thus about 13 iterations are required for the desired level of convergence.

Question 2 CG Error Estimate

The CG iterative error is

$$e_k = x^\star - x_k \tag{10}$$

$$= x^\star - x_0 - \sum_{j=0}^{k-1} r_j A^j r_0 \tag{11}$$

$$= e_0 - \sum_{j=0}^{k-1} r_j A^j r_0 \leq p(A)e_0, \qquad \forall p \in \mathcal{P}_k, \tag{12}$$

and its norm is bounded by

$$\|e_k\|_A \leq \|p(A)e_0\|_A \tag{13}$$

$$\|e_k\|_A^2 \leq \|p(A)e_0\|_A^2 = (p(A)e_0)^T A p(A)e_0. \tag{14}$$

Since $A = U\Lambda U^T$, where Λ is a diagonal matrix composed of the eigenvalues of A and U is an orthonormal matrix composed of the corresponding eigenvectors. Then $p(A) = Up(\Lambda)U^T$ and

$$\|e_k\|_A^2 \leq (p(A)e_0)^T U\Lambda U^T (p(A)e_0) \tag{15}$$
$$\leq (Up(\Lambda)U^T e_0)^T U\Lambda U^T (Up(\Lambda)U^T e_0) \tag{16}$$
$$\leq e_0^T Up(\Lambda)\cancel{U^T U}^{I}\Lambda\cancel{U^T U}^{I}p(\Lambda)U^T e_0 \tag{17}$$
$$\leq e_0^T Up(\Lambda)\Lambda p(\Lambda)U^T e_0 \tag{18}$$

This product is of quadratic form and can be represented as a sum

$$\|e_k\|_A^2 \leq (U^T e_0)^T p(\Lambda)\Lambda p(\Lambda)U^T e_0 \tag{19}$$
$$\leq \sum_{i=0}^{n}\sum_{j=1}^{n}(U^T e_0)_i (p(\Lambda)\Lambda p(\Lambda))_{i,j}(U^T e_0)_j \tag{20}$$

where, since $p(\Lambda)\Lambda p(\Lambda)$ is a diagonal matrix,

$$\|e_k\|_A^2 \leq \sum_{j=1}^{n}(U^T e_0)_j \lambda_j p(\lambda_j)^2 (U^T e_0)_j \tag{21}$$
$$\|e_k\|_A^2 \leq \sum_{j=1}^{n}\lambda_j p(\lambda_j)^2 \langle u_j, e_0\rangle , \tag{22}$$

where u_j are the columns of U, equal to the eigenvectors of A.

Question 3 Convergence Proof

Here $x^\star$ denotes the exact solution $x^\star = A^{-1}b$.

The CG iterative error can be estimated by

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq \|p_k(A)\|_A . \tag{23}$$

Then one valid choice for the residual polynomial is

$$\overline{p}_k(z) = \prod_{i=1}^{k}\frac{\lambda_i - z}{\lambda_i} \tag{24}$$

where $p_k(0) = 1$ and $p_k(\lambda_i) = 0$ for $1 \leq i \leq k$. Then, with matrix V composed of the eigenvectors of A and diagonal matrix Λ where the diagonal entries are the corresponsing eigenvalues of A, $A = V\Lambda V^{-1}$, and $p_k(A) = Vp_K(\Lambda)V^{-1}$:

$$p_k(A) = \sum_{j=0}^{k} c_j A^j \tag{25}$$

$$p_k(A) = \sum_{j=0}^{k} c_j V D^j V^{-1} \tag{26}$$

$$p_k(A) = V \left(\sum_{j=0}^{k} c_j D^j \right) V^{-1} \tag{27}$$

$$p_k(A) = V p_K(\Lambda) V^{-1}, \tag{28}$$

and since $\overline{p}_k(\lambda_i) = 0$ for $1 \leq i \leq k$,

$$\overline{p}_n(\lambda_l) = \frac{\lambda_1 - \lambda_l}{\lambda_1} \cdots \frac{\lambda_l - \lambda_l}{\lambda_l} \cdots \frac{\lambda_n - \lambda_l}{\lambda_n} = 0 \text{ for } 1 \leq l \leq n \tag{29}$$

and

$$\frac{\|e_k\|_A}{\|e_0\|_A} \leq \|\overline{p}_n(A)\|_A = \left\|V \overline{p}_n(\Lambda) V^{-}1\right\|_A = 0 \tag{30}$$

Then the error of the nth iterate is zero, so the algorithm converges in at most n iterations.

Question 4 CG Inverse Problem

4(a) Normal Equation From the regularized inverse equation:

$$\min_m \frac{1}{2} \|Fm - u_d\|_2^2 + \frac{\alpha}{2} m^T R m, \tag{31}$$

the regularization term can be written as the R-norm,

$$\min_m \frac{1}{2} \|Fm - u_d\|_2^2 + \frac{\alpha}{2} \|m\|_R^2, \tag{32}$$

and again using the property that $\|x\|_A = \left\|A^{1/2} x\right\|_2$,

$$\min_m \frac{1}{2} \|Fm - u_d\|_2^2 + \frac{\alpha}{2} \left\|R^{1/2} m\right\|_2^2. \tag{33}$$

Next, since for squared norms $\|u\|_2^2 + \|v\|_2^2 = \|u + v\|_2^2$, this becomes

$$\min_m \frac{1}{2} \left\| Fm + \sqrt{\alpha} R^{1/2} m - u_d \right\|_2^2. \tag{34}$$

Multiplying by 2, taking the square root, and arranging the terms as composite matrices, the expression becomes

$$\min_m \left\| \begin{pmatrix} F \\ \sqrt{\alpha R} \end{pmatrix} m - \begin{pmatrix} u_d \\ 0 \end{pmatrix} \right\|_2 \tag{35}$$

which is a standard least-squares problem, and can be solved by finding the solution to the linear system

$$M^T M m = M^T b, \tag{36}$$

where

$$M = \begin{pmatrix} F \\ \sqrt{\alpha R} \end{pmatrix} \tag{37}$$

$$b = \begin{pmatrix} u_d \\ 0 \end{pmatrix} \tag{38}$$

For the code formulation, where the matrix form of F cannot be accesed, these matrix multiplications (using $R^T = R$ and $F^T = F$) are equivalent to

$$M^T M m = \begin{pmatrix} F & \sqrt{\alpha R} \end{pmatrix} \begin{pmatrix} F \\ \sqrt{\alpha R} \end{pmatrix} m = FFm + \alpha R m \tag{39}$$

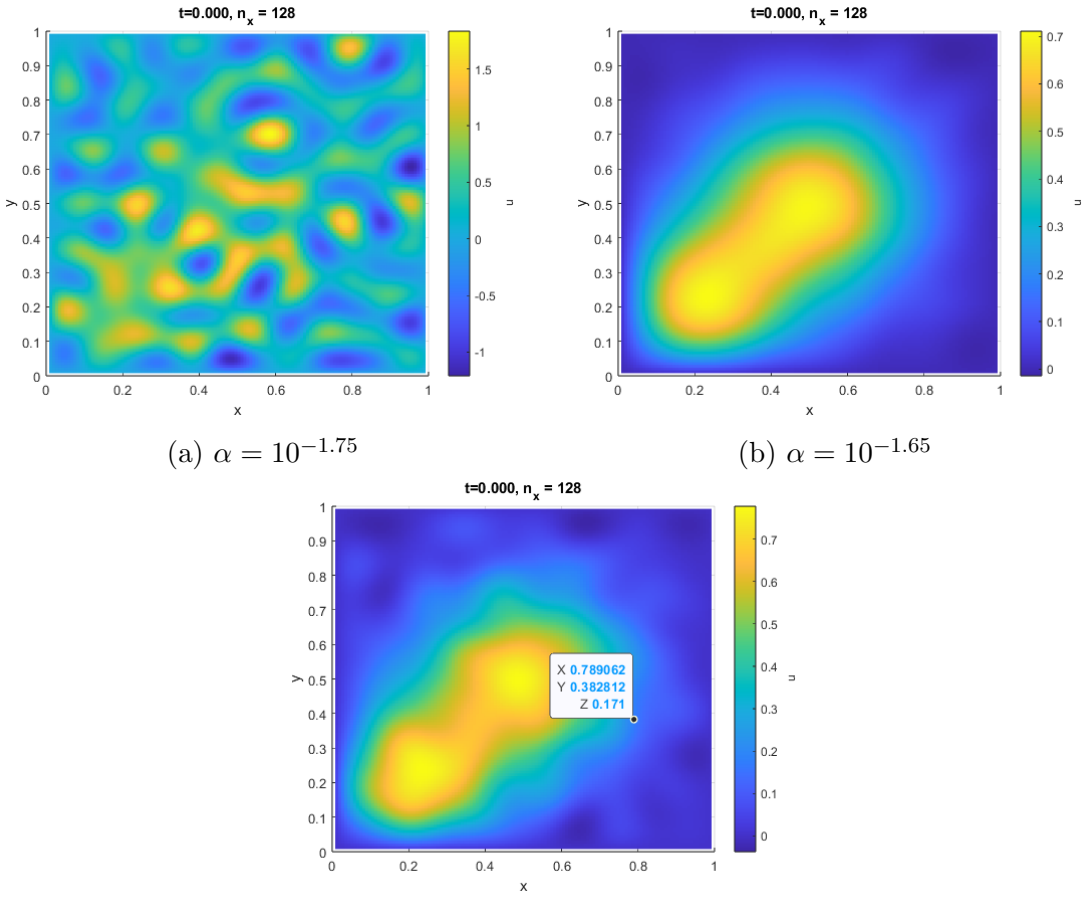
$$M^T b = \begin{pmatrix} F & \sqrt{\alpha R} \end{pmatrix} \begin{pmatrix} u_d \\ 0 \end{pmatrix} = F u_d + 0 = F u_d. \tag{40}$$

4(b) MATLAB code overview The MATLAB implementation of the regularized least sqaures intializes problem data, and steps backward in time for `nt` steps, and solves Equation 36 using

```
pcg(@normal_forward_heat, b, tol, 40,[],[],U(:, n))
```

where `normal_forward_heat` corresponds with Equation 39 and `b` corresponds with Equation 40. The tolerance is set to 10^{-8}, though this is likely more precision than is needed (since the noise already introduces error), and the maximum iterations is set to 40, though this is not exceeded. The solution from the previous step is set to the initial guess.

The regularization matrix R was selected to be A, the discretized differential operator from the forward problem, which penalizes solutions with a high second derivative, which is introduced by the noise in the final condition.



(a) $\alpha = 10^{-1.75}$

(b) $\alpha = 10^{-1.65}$

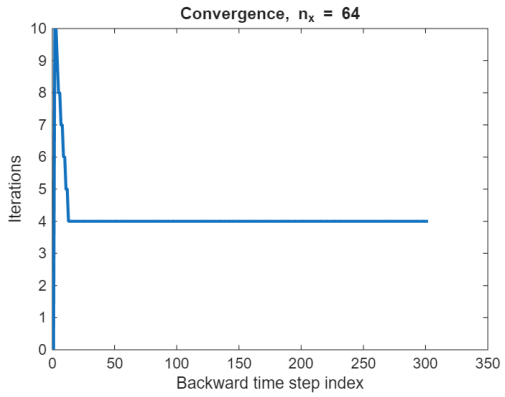
(c) $\alpha = 10^{-1.69}$

Figure 1: Reg. parameter range

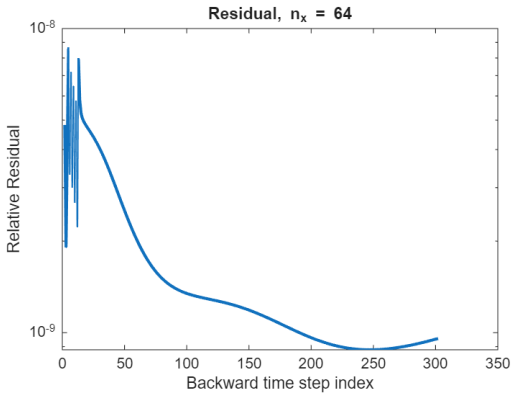
4(c) Regularization parameter With $R = A$, the ideal regularization parameter clearly lies in the range of 10^{-2} to 10^{-1}. Figure 1a clearly shows and under-smoothed solution, since the regularization parameter was too low, and Figure 1b shows an over-smoothed solution, where the initial-time solution shows little difference from the final condition.

In this example, since the true solution is known, α can be selected using "guess-and-check," but in real problems this may not be appropriate, and α may need to be selected using more care. Here $\alpha = 10^{-1.69}$ was chosen, since it was the value of α where the effect of noise could still be seen in the solution, but not overwhelming the smoother, more physical solution.

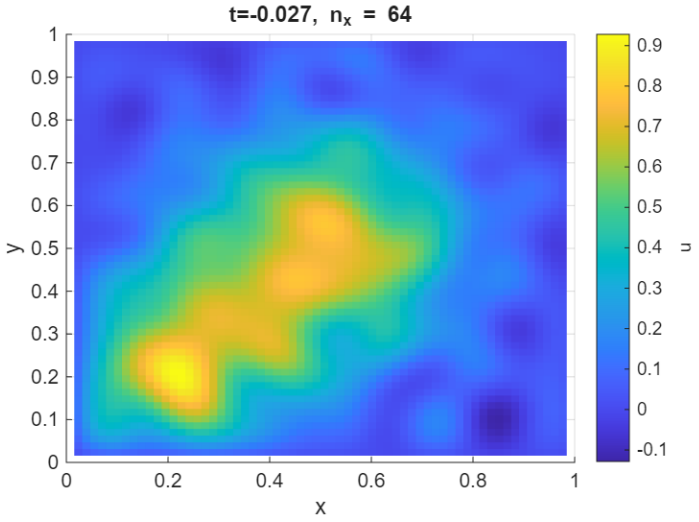
4(d) Solutions and convergence Figures 2, 3, and 4 show the solutions, convergence, and iterations for each resolution. Figure 2b most clearly shows the convergence behavior of the pcg iteration, since the solution from the previous time step is taken as the



(a) Number of pcg iterations



(b) Relative residual



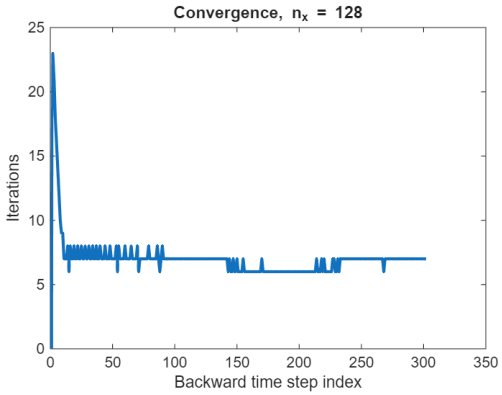
(c) Solution

Figure 2: Solution and convergence, $n_x = 2^6$

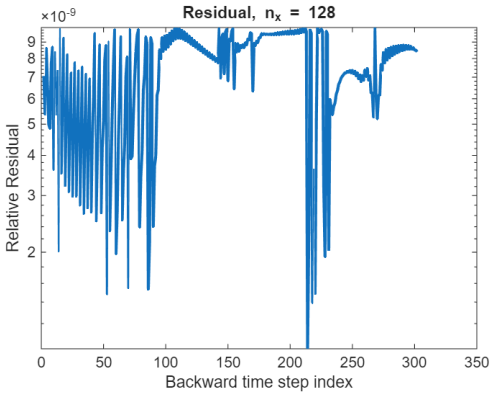
initial iteration, the intial residual $\|r_0\|$ is highest for the first time steps, so these require more iterations to converge, and have a higher relative residual after iterations.

Once the noise has been mostly smoothed (about 20 iterations for $n_x = 2^6$), the inital guess is very close to the actual solution for most iterations.

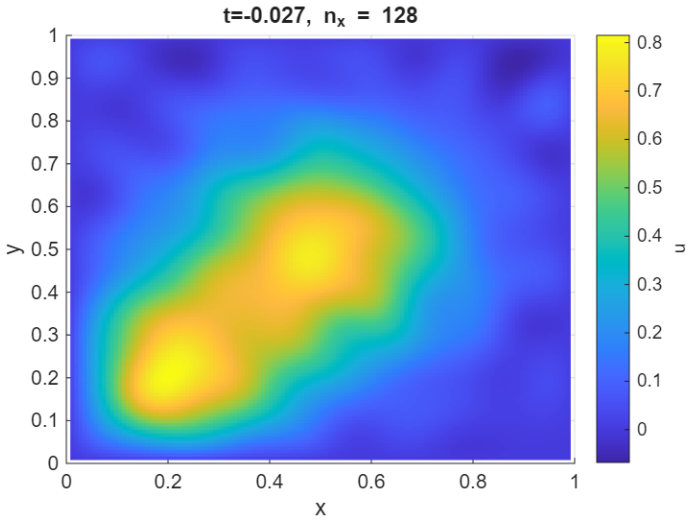
The solutions show some non-physical behavior, where the intial temperature is negative in some places. This could be improved by a different choice of R.



(a) Number of pcg iterations

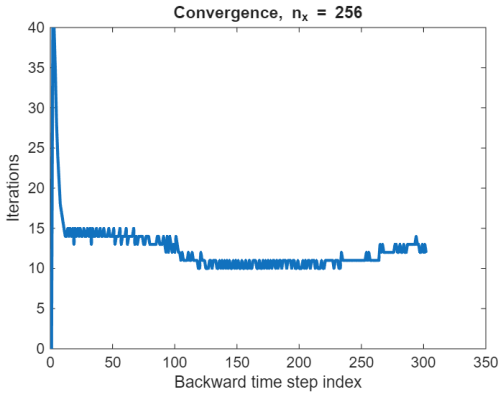


(b) Relative residual

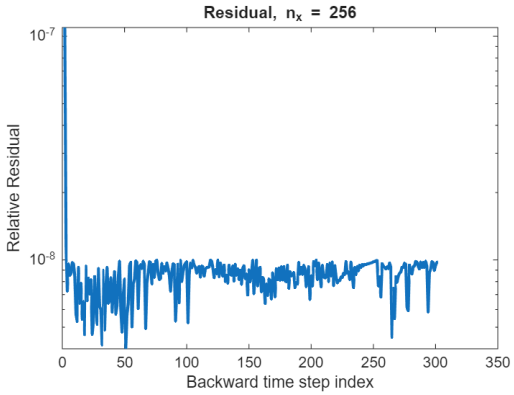


(c) Solution

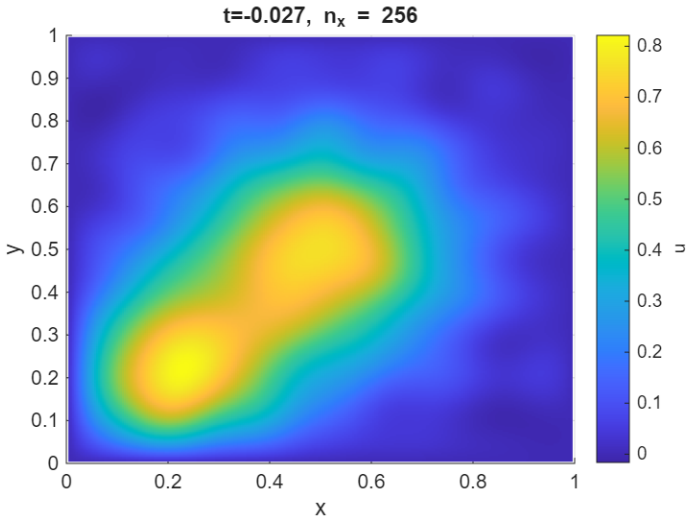
Figure 3: Solution and convergence, $n_x = 2^7$



(a) Number of pcg iterations



(b) Relative residual



(c) Solution

Figure 4: Solution and convergence, $n_x = 2^8$

Question A MATLAB code

```
clear all
close all

global R alpha model_data
init_heat_2d()

%%
nt = model_data.nt;
nx = model_data.nx;

s = length(model_data.A);
q = sqrt(s);

R = model_data.A;

alpha = 10^(-1.69);
tol = 1e-8;

start = tic;

iter = zeros(1, nt+1);
flag = 0;
relres = iter;

U = zeros((nx-1)^2,nt+1);
U(:, 1) = ud;

model_data.nt = 1;


for n = 1:nt
    b = heat_forward_solve_2d(U(:, n), model_data);
    [U(:, n+1), flag, relres(n+1), iter(n+1)] = ...
        pcg(@normal_forward_heat, b, tol, 40,[],[],U(:, n));
    fprintf("time step %i \n", n)
end

fprintf("%.2f second runtime", toc(start))

%% plots
close all

figure()
```

```
plot(iter, LineWidth=2)
xlabel("Backward time step index")
ylabel("Iterations")
title(sprintf("Convergence, n_x = %i", nx))
set(gcf, "Position", [100 100 500 350])
saveas(gcf, sprintf("convergence_%i.png", nx))

figure()
plot(relres, LineWidth=2)
xlabel("Backward time step index")
ylabel("Relative Residual")
yscale("log")
title(sprintf("Residual, n_x = %i", nx))
set(gcf, "Position", [100 100 500 350])
saveas(gcf, sprintf("residual_%i.png", nx))

figure()
s = surf(xi(2:end-1), yi(2:end-1), reshape(ud, [q, q]));
s.EdgeColor = 'none';
view(0,90)
c = colorbar;
c.Label.String = 'u';
set(gcf, "Position", [100 100 500 350])

figure()
s = surf(xi(2:end-1), yi(2:end-1), reshape(ut, [q, q]));
s.EdgeColor = 'none';
view(0,90)
c = colorbar;
c.Label.String = 'u';
set(gcf, "Position", [100 100 500 350])
saveas(gcf, sprintf("reality_%i.png", nx))

for t = [301]
    figure()
    s = surf(xi(2:end-1), yi(2:end-1), reshape(U(:, t), [q, q]));
    s.EdgeColor = 'none';
    view(0,90)
    xlabel("x")
    ylabel("y")
    c = colorbar;
    c.Label.String = 'u';
    title(sprintf("t=%.3f, n_x = %i", tf - t*dt, nx))
    set(gcf, "Position", [100 100 500 350])
    saveas(gcf, sprintf("starting_%i.png", nx))
```

```
end

function M = normal_forward_heat(v)
    global R alpha model_data

    M = heat_forward_solve_2d(...
        heat_forward_solve_2d(v, model_data), model_data) ...
        + (alpha*R*v);
end
```