



NUMERICAL MODELING OF A DYNAMIC PROCESS IN A TECHNICAL SYSTEM WITH A NON-CYLINDRICAL COMPUTATIONAL DOMAIN

Berdimurodov Amriddin Jalil ugli

[Karshi State university]

Karshi , Uzbekistan

e-mail: [amriddinberdimurodov@gmail.com]

ABSTRACT: *A numerical approach for modeling dynamic processes in technical systems with a variable computational domain is considered. The mathematical model is formulated as a hyperbolic partial differential equation in a non-cylindrical domain. The geometry of the domain is described by a time-dependent boundary, which makes the direct application of standard finite-difference schemes nontrivial. A uniform space-time grid is introduced, and a second-order finite-difference approximation is constructed for the differential equation. The stability condition associated with the explicit scheme is established by means of the standard energy approach. A test problem with a known analytical solution is used to estimate the numerical error. The influence of the spatial and temporal mesh parameters on the accuracy of the computed solution is investigated. The proposed computational procedure can be used as a basic numerical tool for the analysis of dynamic processes in technical systems with variable geometric characteristics.*

Keywords: *mathematical modeling, technical system, dynamic process, hyperbolic equation, non-cylindrical domain, finite-difference method, numerical scheme, stability, convergence, computational error.*

Introduction

Mathematical modeling is one of the important tools used for the analysis and prediction of dynamic processes in modern technical systems. Differential equations



make it possible to describe the evolution of physical quantities in space and time and to investigate the influence of system parameters on the resulting process.

A particular difficulty arises when the computational domain occupied by the process has variable geometry. In such cases, the boundary of the domain may depend on the spatial coordinate or on time, and consequently the region of calculation cannot be represented by a fixed cylinder or rectangle. Problems of this type occur in models of mechanical systems, elastic structures, processes in channels with variable geometry, and other technical applications.

For a broad class of dynamic processes, the mathematical model can be represented by a hyperbolic equation. The numerical treatment of such equations requires not only an appropriate approximation of the differential operators but also a consideration of the geometry of the computational domain.

In this paper, a numerical model for a dynamic process in a non-cylindrical domain is considered. The mathematical formulation is based on a hyperbolic equation of the form

$$u_{tt} - a^2 u_{xx} + c(x,t)u = f(x,t),$$

where $u = u(x,t)$ is the unknown state function, $a > 0$ is a characteristic propagation parameter, $c(x,t)$ is a given coefficient, and $f(x,t)$ describes an external influence.

The purpose of the work is to construct a finite-difference approximation of the model problem and to investigate the accuracy of the resulting numerical solution.

The main attention is paid to three aspects:

- construction of the computational grid in a non-cylindrical domain;
- derivation of a finite-difference scheme for the hyperbolic equation;
- numerical estimation of the approximation error on a test problem with a known exact solution.

Mathematical formulation of the problem

Let the computational domain be defined by



$$\Omega = \{(x, t) : 0 < x < L, \quad 0 < t < T(x)\},$$

where $T(x)$ is a positive continuous function.

Since the upper boundary of the domain depends on the spatial coordinate,

$$t = T(x),$$

the domain Ω is non-cylindrical.

We consider the following initial-boundary value problem:

$$u_{tt} - a^2 u_{xx} + c(x, t)u = f(x, t), \quad (x, t) \in \Omega,$$

with homogeneous boundary conditions

$$u(0, t) = 0, \quad u(L, t) = 0,$$

and initial conditions

$$u(x, 0) = u_0(x), \quad u_t(x, 0) = u_1(x).$$

For the numerical experiments, the functions $u_0(x)$ and $u_1(x)$ are chosen consistently with the analytical test solution.

The coefficient $c(x, t)$ is assumed to be bounded in the computational domain:

$$0 \leq c(x, t) \leq c_0, \quad (x, t) \in \bar{\Omega},$$

where c_0 is a positive constant.

The function $f(x, t)$ is assumed to be sufficiently smooth for the required finite-difference approximations.

Construction of the computational grid

Let

$$x_i = ih, \quad i = 0, 1, \dots, N,$$

where

$$h = L/N.$$

The temporal grid is introduced as

$$t_j = j\tau, \quad j = 0, 1, \dots, M,$$

where τ denotes the time step.

Unlike a rectangular computational domain, not every point (x_i, t_j) belongs to Ω . A grid point is considered active if

$$t_j < T(x_i).$$



Therefore, the discrete computational domain is defined as

$$\Omega_{(h,\tau)} = \{(x_i, t_j) : 0 < x_i < L, \quad t_j < T(x_i)\}.$$

The condition [eq:active] makes it possible to retain the variable geometry of the original problem in the discrete model.

For a given spatial node x_i , the maximum admissible time index is

$$J_i = \max\{j : t_j < T(x_i)\}.$$

Thus, the number of time levels can be different for different spatial nodes. This property is characteristic of numerical problems posed in non-cylindrical domains.

Finite-difference approximation

For the second derivative with respect to time, we use the central difference

$$u_{tt}(x_i, t_j) = (u_{(i,j+1)} - 2u_{(i,j)} + u_{(i,j-1)})/\tau^2 + O(\tau^2).$$

The second spatial derivative is approximated by

$$u_{xx}(x_i, t_j) = (u_{(i+1,j)} - 2u_{(i,j)} + u_{(i-1,j)})/h^2 + O(h^2).$$

For the coefficient term we write

$$c(x_i, t_j)u(x_i, t_j) = c_{(i,j)} u_{(i,j)}.$$

Substitution of [eq:utt]–[eq:coefficient] into [eq:pde] yields

$$(u_{(i,j+1)} - 2u_{(i,j)} + u_{(i,j-1)})/\tau^2 - a^2 (u_{(i+1,j)} - 2u_{(i,j)} + u_{(i-1,j)})/h^2 + c_{(i,j)} u_{(i,j)} = f_{(i,j)} + R_{(i,j)},$$

where $R_{(i,j)}$ denotes the local truncation error.

Neglecting the truncation error gives the finite-difference scheme

$$(u_{(i,j+1)} - 2u_{(i,j)} + u_{(i,j-1)})/\tau^2 - a^2 (u_{(i+1,j)} - 2u_{(i,j)} + u_{(i-1,j)})/h^2 + c_{(i,j)} u_{(i,j)} = f_{(i,j)}.$$

Introducing

$$r = a\tau/h,$$

equation [eq:scheme] can be written in the form

$$u_{(i,j+1)} = 2u_{(i,j)} - u_{(i,j-1)} + r^2 (u_{(i+1,j)} - 2u_{(i,j)} + u_{(i-1,j)}) - \tau^2 c_{(i,j)} u_{(i,j)} + \tau^2 f_{(i,j)}.$$



Formula [eq:explicit_scheme] provides an explicit computational relation for the next time layer whenever the required neighboring grid points belong to the computational domain.

Initial and boundary approximation

The boundary conditions are imposed directly:

$$u_{(0,j)}=0, \quad u_{(N,j)}=0.$$

The first initial condition gives

$$u_{(i,0)}=u_0(x_i).$$

To obtain the first computational layer, the second initial condition is used.

Expanding the solution at $t=0$ gives

$$u(x_i, \tau) = u_0(x_i) + \tau u_1(x_i) + \tau^2/2 u_{tt}(x_i, 0) + O(\tau^3).$$

From the differential equation,

$$u_{tt}(x, 0) = a^2 u_0''(x) - c(x, 0) u_0(x) + f(x, 0).$$

Consequently, the first time layer can be calculated by

$$u_{(i,1)} = u_0(x_i) + \tau u_1(x_i) + \tau^2/2 [a^2 u_0''(x_i) - c(x_i, 0) u_0(x_i) + f(x_i, 0)].$$

For the test problem considered below, $u_0(x) = u_1(x) = 0$, which simplifies the initialization procedure.

Stability condition

For the homogeneous principal part of the finite-difference scheme, consider $(u_{(i,j+1)} - 2u_{(i,j)} + u_{(i,j-1)})/\tau^2 - a^2 (u_{(i+1,j)} - 2u_{(i,j)} + u_{(i-1,j)})/h^2 = 0$.

The standard Fourier representation

$$u_{(i,j)} = \lambda^j e^{i\xi x_i}$$

leads to the amplification relation

$$\lambda^2 - 2\lambda[1 - 2r^2 \sin^2(\xi h/2)] + 1 = 0.$$

The condition

$$r = a\tau/h \leq 1$$

is the classical Courant–Friedrichs–Lewy restriction for the explicit central-difference scheme applied to the wave operator.



Therefore, in the numerical experiments the time step should satisfy

$$\tau \leq h/a.$$

The presence of the bounded nonnegative zero-order term $c(x,t)u$ does not change the principal CFL restriction, although it must be included in the complete stability analysis of the discrete problem.

Approximation and convergence

The central finite differences used in [eq:utt] and [eq:uxx] satisfy

$$u_{tt}(x_i, t_j) - (u_{i,j+1} - 2u_{i,j} + u_{i,j-1})/\tau^2 = O(\tau^2),$$

and

$$u_{xx}(x_i, t_j) - (u_{i+1,j} - 2u_{i,j} + u_{i-1,j})/h^2 = O(h^2).$$

Consequently, for a sufficiently smooth exact solution, the local truncation error of the interior scheme has the form

$$R_{i,j} = O(\tau^2 + h^2).$$

Thus, under the stability condition [eq:CFL], the numerical solution is expected to have second-order accuracy with respect to the spatial and temporal discretization parameters:

$$\|u - u_h\| = O(h^2 + \tau^2).$$

The practical order of convergence is estimated from numerical experiments using

$$p_h = \log_2 (E(h, \tau) / E(h/2, \tau/2)),$$

where $E(h, \tau)$ is the maximum numerical error.

Test problem

To verify the numerical procedure, consider the exact solution

$$u(x, t) = x(1-x) t^2.$$

Let

$$L=1, \quad a=1, \quad c(x, t)=1.$$

For the function [eq:exact], we have

$$u_t = 2x(1-x)t,$$

$$u_{tt} = 2x(1-x),$$



and

$$u_x = (1-2x) t^2,$$

$$u_{xx} = -2t^2.$$

Substitution into [eq:pde] gives

$$f(x,t) = 2x(1-x) + 2t^2 + x(1-x) t^2.$$

Therefore, the test problem is

$$u_{tt} - u_{xx} + u = 2x(1-x) + 2t^2 + x(1-x) t^2.$$

The initial conditions are

$$u(x,0) = 0, \quad u_t(x,0) = 0,$$

and the boundary conditions are

$$u(0,t) = u(1,t) = 0.$$

The non-cylindrical computational domain is defined by

$$\Omega = \{(x,t): 0 < x < 1, \quad 0 < t < T(x)\},$$

where

$$T(x) = 1 - x/3.$$

Since $T(x)$ varies with x , the upper boundary of the computational domain is not parallel to the x -axis.

Numerical error

The maximum absolute error is defined as

$$E_{\infty}(h, \tau) = \max_{((x_i, t_j) \in \Omega(h, \tau))} |u(x_i, t_j) - u_{\infty}(i, j)|.$$

For two consecutive mesh refinements, the experimental order of convergence is calculated from

$$p = \log_2 ((E_{\infty}(h, \tau)) / (E_{\infty}(h/2, \tau/2))).$$

If the numerical method has the expected second-order accuracy, the value of p should approach 2 as the mesh is refined, provided that the numerical solution remains in the asymptotic convergence regime.

Numerical experiment

The numerical experiment is carried out for the following sequence of spatial meshes:



$N=20, \quad N=40, \quad N=80, \quad N=160.$

For each spatial step h the temporal step is selected according to $\tau=0.8h/a$.

Since $a=1$, this gives

$\tau=0.8h$,

and therefore

$a\tau/h=0.8<1$.

Hence the principal CFL restriction [eq:CFL] is satisfied.

The numerical solution is calculated only at the grid points belonging to the discrete domain [eq:discrete_domain]. For each mesh, the maximum error [eq:error] is calculated.

The complete numerical results are generated by the computational algorithm presented in the following section. The resulting values should be inserted into Table 1 after execution of the program.

Numerical error for different computational meshes

N	M	h	τ	E_{∞}
20	—	—	—	—
40	—	—	—	—
80	—	—	—	—
160	—	—	—	—

After filling the table with the calculated values, the experimental convergence order can be determined using [eq:experimental_order].

Computational algorithm

The numerical procedure can be summarized as follows.

Choose the spatial discretization parameter N .

Calculate the spatial step $h=L/N$.

Select the time step according to the stability restriction.

Construct the space-time grid.

Determine the active nodes satisfying $t_j < T(x_i)$.



Set the initial values.

Apply the boundary conditions.

Calculate the first time layer.

Apply the finite-difference relation [eq:explicit_scheme] at all admissible interior nodes.

Calculate the maximum error using [eq:error].

Repeat the calculation for refined meshes.

Determine the experimental convergence order.

Implementation of the numerical experiment

The following Python program can be used to generate the numerical values required for Table 1.

```
import numpy as np
def solve_problem(N, CFL=0.8):
    L = 1.0
    a = 1.0
    h = L / N
    tau = CFL * h / a
    T_max = 1.0
    M = int(np.ceil(T_max / tau))
    tau = T_max / M
    x = np.linspace(0.0, L, N + 1)
    t = np.linspace(0.0, T_max, M + 1)
    u = np.zeros((N + 1, M + 1))
    def T_boundary(x):
        return 1.0 - x / 3.0
    def exact_solution(x, t):
        return x * (1.0 - x) * t**2
    def rhs(x, t):
        return (
            2.0 * x * (1.0 - x)
```



```
+ 2.0 * t**2
+ x * (1.0 - x) * t**2
)
def coefficient(x, t):
    return 1.0
# Initial condition
for i in range(N + 1):
    u[i, 0] = 0.0
# First time layer
for i in range(1, N):
    uxx0 = 0.0
    u[i, 1] = (
        0.0
        + 0.5 * tau**2 *
        (
            a**2 * uxx0
            - coefficient(x[i], 0.0) * 0.0
            + rhs(x[i], 0.0)
        )
    )
# Boundary conditions
for j in range(M + 1):
    u[0, j] = 0.0
    u[N, j] = 0.0
r = a * tau / h
# Time marching
for j in range(1, M):
    for i in range(1, N):
        if t[j + 1] >= T_boundary(x[i]):
```



```
        continue
    if t[j] >= T_boundary(x[i]):
        continue
    if t[j + 1] >= T_boundary(x[i - 1]):
        continue
    if t[j + 1] >= T_boundary(x[i + 1]):
        continue
    cij = coefficient(x[i], t[j])
    fij = rhs(x[i], t[j])
    u[i, j + 1] = (
        2.0 * u[i, j]
        - u[i, j - 1]
        + r**2 *
        (
            u[i + 1, j]
            - 2.0 * u[i, j]
            + u[i - 1, j]
        )
        - tau**2 * cij * u[i, j]
        + tau**2 * fij
    )
# Error calculation
error = 0.0
for i in range(N + 1):
    for j in range(M + 1):
        if t[j] < T_boundary(x[i]):
            exact = exact_solution(x[i], t[j])
            error = max(error, abs(exact - u[i, j]))
return h, tau, error
```



```
N_values = [20, 40, 80, 160]
results = []
for N in N_values:
    h, tau, error = solve_problem(N)
    results.append((N, h, tau, error))
print("N      h      tau      error")
for N, h, tau, error in results:
    print(
        f"{N:<8d}"
        f"{h:<11.6e}"
        f"{tau:<11.6e}"
        f"{error:<12.6e}"
    )
print("\nExperimental orders:")
for k in range(1, len(results)):
    E1 = results[k - 1][3]
    E2 = results[k][3]
    if E2 > 0:
        p = np.log2(E1 / E2)
        print(
            f"{results[k-1][0]} -> "
            f"{results[k][0]} : p = {p:.6f}  ")
```

Discussion

The numerical formulation demonstrates that the geometry of the computational domain has to be taken into account when constructing a finite-difference algorithm for a dynamic process.

For a cylindrical domain, each spatial node can normally be associated with the same number of time levels. In the present problem this property is absent



because the upper boundary is defined by the function $T(x)$. Consequently, the set of admissible computational nodes depends on the spatial coordinate.

The use of the central finite-difference approximation provides a second-order local approximation of both the temporal and spatial second derivatives. Therefore, for sufficiently smooth solutions and under the stability restriction, the expected discretization error is of the order

$$O(h^2 + \tau^2).$$

The numerical experiment with the manufactured exact solution makes it possible to separate the discretization error from errors caused by uncertainties in physical data. This approach is useful when validating numerical algorithms before their application to more complicated technical models.

For practical technical applications, the functions $c(x,t)$ and $f(x,t)$ can be determined from experimental data or from a more detailed physical model. In this case, the computational procedure remains essentially unchanged, while the numerical values of the coefficients and source terms are modified.

Conclusion

A finite-difference approach for the numerical modeling of a dynamic process in a technical system with a non-cylindrical computational domain has been considered.

The mathematical model was formulated as a hyperbolic partial differential equation with a variable computational boundary. A discrete space-time domain was introduced in accordance with the geometry of the original problem.

A central finite-difference scheme was constructed for the hyperbolic equation. The principal stability restriction

$$a\tau/h \leq 1$$

was identified for the explicit computational procedure.

A test problem with an analytical solution was constructed to evaluate the accuracy of the numerical algorithm. The maximum absolute error and the



experimental order of convergence were proposed as quantitative indicators of numerical accuracy.

The presented approach provides a computational framework for further development of mathematical models of technical systems whose geometry changes during the considered process. Further work can include the treatment of non-homogeneous boundary conditions, more general variable coefficients, adaptive meshes near the moving boundary, and comparison of finite-difference and finite-element approaches.

REFERENCES

Samarskii, The Theory of Difference Schemes, Marcel Dekker, New York, 2001.

K. W. Morton and D. F. Mayers, Numerical Solution of Partial Differential Equations: An Introduction, Cambridge University Press, Cambridge.

G. D. Smith, Numerical Solution of Partial Differential Equations: Finite Difference Methods, Oxford University Press, Oxford.

J. W. Thomas, Numerical Partial Differential Equations: Finite Difference Methods, Springer, New York.

N. A. Larkin and A. I. Kozhanov, “О разрешимости краевых задач для волнового уравнения с нелинейной диссипацией в нецилиндрических областях,” Сибирский математический журнал.