

**APLIKASI PHYSICS-INFORMED NEURAL
NETWORK (PINN) DALAM FENOMENA
PERSEBARAN PANAS**

TUGAS AKHIR

**Karya tulis sebagai salah satu syarat
untuk memperoleh gelar Sarjana dari**

Institut Teknologi Bandung

Oleh

**MICHAEL ALFARINO BELLA PHUNGPUTRA
NIM 10219111**



**PROGRAM STUDI SARJANA FISIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI BANDUNG
2023**

**APLIKASI PHYSICS-INFORMED NEURAL NETWORK (PINN)
DALAM FENOMENA PERSEBARAN PANAS**

TUGAS AKHIR

Diajukan untuk memenuhi syarat kelulusan tahap pendidikan strata-1 pada
Program Studi Fisika – Institut Teknologi Bandung

Oleh:

Michael Alfarino Bella Phungputra
10219111

Pembimbing:

Dr. Irfan Dwi Aditya, S.Si., M.Si.
Prof. Dr. Suprijadi, M.Eng.

**PROGRAM STUDI SARJANA FISIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI BANDUNG
2023**

ABSTRAK

APLIKASI PHYSICS-INFORMED NEURAL NETWORK (PINN) DALAM FENOMENA PERSEBARAN PANAS

Oleh

Michael Alfarino Bella Phungputra

NIM: 10219111

Persebaran panas pada daerah tertentu dideskripsikan oleh persamaan panas yang merupakan salah satu contoh persamaan diferensial parsial. Terdapat beberapa metode untuk memperoleh solusi dari persamaan diferensial parsial, yaitu metode analitik, metode benda hingga dengan skema *Forward-Time Centered-Space* (FTCS), serta *Physics-Informed Neural Network* (PINN). Telah dilakukan simulasi hasil persamaan panas menggunakan metode numerik FTCS dan PINN. Kemudian, berdasarkan hasil simulasi FTCS dan PINN, dilakukan visualisasi solusi persamaan panas serta penentuan nilai *mean-squared error* (MSE) untuk tiap metode. Hasil penyelesaian persamaan panas menggunakan PINN memiliki *mean-squared error* yang kecil jika dibandingkan dengan metode analitik. Dapat disimpulkan bahwa PINN secara umum bisa dijadikan salah satu alternatif untuk menyelesaikan persamaan diferensial parsial, khususnya persamaan panas. Salah satu keunggulan dari PINN adalah solusinya yang kontinu serta tidak terikat oleh kisi (*grid*). Selanjutnya, setelah dibandingkan performa PINN terhadap metode analitik dan FTCS, telah dipelajari penyelesaian persamaan panas 2 dimensi dengan sumber panas menggunakan *Physics-Informed Neural Network* (PINN). Dipelajari dua variasi keadaan, yaitu sumber panas di tengah domain dan sumber panas di pinggir domain. Setelah disimulasikan kedua variasi, diperoleh kesimpulan bahwa temperatur akan mengalami kenaikan dan menuju saturasi pada titik dengan jarak tertentu dari sumber panas. Titik yang lebih dekat dengan sumber panas akan mengalami saturasi pada temperatur yang lebih tinggi.

Kata kunci: *deep learning*, FTCS, persamaan panas, *physics-informed neural network*.

ABSTRACT

PHYSICS-INFORMED NEURAL NETWORK (PINN) AND ITS APPLICATION TO HEAT DISTRIBUTION

by

Michael Alfarino Bella Phungputra

NIM: 10219111

Distribution of heat is described by the heat equation which is an example of a partial differential equation. There are several methods for obtaining solutions of partial differential equations, namely analytic methods, finite difference methods with the Forward-Time Centered-Space (FTCS) scheme, and Physics-Informed Neural Network (PINN). Heat equation simulation has been done using finite difference method (FTCS scheme) and PINN. Based on the results of the FTCS and PINN simulations, the solution to the heat equation was visualized and the mean-squared error (MSE) value was determined for each method. Solution to the heat equation using PINN has a small mean-squared error when compared to the analytical method. It is concluded that PINN can generally be an alternative for solving partial differential equations, the heat equation in particular. One of the advantages of PINN is that the solution is continuous and not bound by a grid. After comparing the performance of PINN against analytical and FTCS methods, 2-dimensional heat equation with a heat source was studied using Physics-Informed Neural Network (PINN). Two variations of the heat source location were studied, in the middle of the domain and at the edge of the domain. After simulating these two variations, it was concluded that temperatures on a point with a certain distance from the heat source will increase and saturate towards a limit. Points closer to the heat source will experience saturation at a higher temperature.

Keywords: deep learning, FTCS, heat equation, physics-informed neural network.

PENGESAHAN

APLIKASI PHYSICS-INFORMED NEURAL NETWORK (PINN) DALAM FENOMENA PERSEBARAN PANAS

Oleh

Michael Alfarino Bella Phungputra

NIM 10219111

Program studi Sarjana Fisika

Fakultas Matematika dan Ilmu Pengetahuan Alam

Institut Teknologi Bandung

Menyetujui

Bandung, 23 Februari 2023

Dosen Pembimbing 1,



Dr. Irfan Dwi Aditya, S.Si., M.Si.
NIP. 19870424 201504 1 003

Dosen Pembimbing 2,



Prof. Dr. Suprijadi, M.Eng.
NIP. 19670711 199303 1 001

Tim Penguji:

1. Fahdzi Muttaqien, S.Si., M.Si., M.Sc., M.Eng., Ph.D.
2. Dr. rer. nat. Sparisoma Viridi, S.Si.

PEDOMAN PENGGUNAAN BUKU TUGAS AKHIR

Buku Tugas Akhir Sarjana ini tidak dipublikasikan, namun terdaftar dan tersedia di Perpustakaan Institut Teknologi Bandung. Buku ini dapat diakses umum, dengan ketentuan bahwa penulis memiliki hak cipta dengan mengikuti aturan HaKI yang berlaku di Institut Teknologi Bandung. Referensi kepustakaan diperkenankan dicatat, tetapi pengutipan atau peringkasan hanya dapat dilakukan seizin penulis, dan harus disertai dengan kebiasaan ilmiah untuk menyebutkan sumbernya.

Memperbanyak atau menerbitkan sebagian atau seluruh buku Tugas Akhir harus atas izin Program Studi Sarjana Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Teknologi Bandung.

*Tugas Akhir ini dipersembahkan untuk
Tuhan, Bangsa, dan Almamater*

KATA PENGANTAR

Puji syukur kepada Tuhan yang Maha Esa yang telah melimpahkan rahmat dan karunia-Nya sehingga penyusunan buku tugas akhir yang berjudul *Aplikasi Physics-Informed Neural Network (PINN)* dalam Fenomena Persebaran Panas ini dapat diselesaikan. Tujuan penulisan buku ini salah satunya adalah sebagai syarat untuk memperoleh gelar sarjana pada Program Studi Fisika, Institut Teknologi Bandung. Penulis berharap agar tugas akhir ini bermanfaat bagi seluruh pihak dan dapat menjadi salah satu alternatif metode untuk memperoleh solusi persamaan diferensial parsial, khususnya persamaan panas.

Penyusunan tugas akhir ini tidak lepas dari bantuan dan dukungan berbagai pihak baik secara langsung maupun tidak langsung. Untuk itu penulis menyampaikan penghargaan dan terima kasih kepada:

- Kedua orang tua penulis, Thomas Becket Sudjono dan Fransisca Dwi Astuti Setyorini, atas kasih sayang, dukungan, dan doa yang selalu menyertai dalam perjalanan perkuliahan penulis.
- Dr. Irfan Dwi Aditya, S.Si., M.Si. dan Prof. Dr. Suprijadi, M.Eng. selaku dosen pembimbing tugas akhir yang telah memberikan ilmu, saran, serta motivasi dalam penyusunan tugas akhir.
- Fahdzi Muttaqien, S.Si., M.Si., M.Sc., M.Eng., Ph.D. dan Dr. rer. nat. Sparisoma Viridi, S.Si. atas kesediaannya menjadi dosen penguji dalam sidang Tugas Akhir serta atas masukannya yang bermanfaat bagi penelitian ini.
- Seluruh dosen dan staff Program Studi Fisika yang telah mengajar dan mendidik penulis selama menjalani perkuliahan di ITB.
- Monika Agata, yang senantiasa menemani serta memberikan perhatian dan semangat selama dua tahun ini.
- Andreas dan Yuki, sahabat sejati yang selalu ada untuk penulis.
- Teman-teman seerbimbingan, Aldian, Avima, Ervandy, Kemal, dan Shafira, yang telah saling membantu, berbagi ilmu, dan berbagi semangat dalam peny-

sunan tugas akhir.

- Teman-teman "Biskuat", Monik, Cindy, Fio, Yemi, Grace, Jape, MB, Ave, yang telah menemani dan mendukung penulis sejak tahap persiapan bersama (TPB).
- Teman-teman ITB Padmanaba 74, Yozi, Romi, Zada, Fais, Wildan, Rafif, Bona, Nisa, Fitrah, David, Wibi, yang selalu menjadi rumah kedua bagi penulis selama perantauan di Bandung.
- Clarissa dan Shabrina yang menjadi sahabat dekat dalam menjalani kehidupan perkuliahan di Fisika.
- Teman-teman HIMAFI ITB, khususnya angkatan 2019 yang telah menjadi teman seperjuangan dan tempat belajar dan berbagi ilmu selama menjalani masa perkuliahan.
- Semua pihak yang secara langsung maupun tidak langsung membantu penyelesaian tugas akhir ini yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa penulisan tugas akhir ini masih terdapat kekurangan. Oleh karena itu, penulis mengharapkan kritik dan saran yang sifatnya membangun agar tugas akhir ini lebih baik dan bermanfaat di masa yang akan datang.

Bandung, 23 Februari 2023

Michael Alfarino Bella Phungputra
10219111

DAFTAR ISI

ABSTRAK	i
ABSTRACT	ii
LEMBAR PENGESAHAN	iii
PEDOMAN PENGGUNAAN BUKU TUGAS AKHIR	iv
KATA PENGANTAR	vi
DAFTAR ISI	viii
DAFTAR NOTASI	xi
DAFTAR GAMBAR	xiii
DAFTAR TABEL	xv
BAB I PENDAHULUAN	1
I.1 Latar Belakang	1
I.2 Rumusan dan Batasan Masalah.....	1
I.3 Tujuan	2
I.4 Metodologi.....	2
I.5 Sistematika Penulisan	3
BAB II TINJAUAN PUSTAKA	4
II.1 Persamaan Diferensial Parsial	4
II.2 Persamaan Panas	4
II.3 Solusi dari Persamaan Panas	5
II.3.1 <i>Separation of Variables</i>	5
II.3.2 Metode <i>Forward-Time Centered-Space</i> (FTCS)	5
II.3.3 <i>Physics-Informed Neural Network</i> (PINN)	7
BAB III METODOLOGI PENELITIAN	12

III.1	Penyelesaian Persamaan Panas 1 Dimensi	12
III.1.1	Metode Separasi Variabel.....	12
III.1.2	Metode <i>Forward-Time Centered-Space</i> (FTCS)	13
III.1.3	Metode <i>Physics-Informed Neural Network</i> (PINN)	15
III.1.4	Ukuran Evaluasi Kinerja	16
III.2	Penyelesaian Persamaan Panas 2 Dimensi	17
III.2.1	Metode Separasi Variabel.....	17
III.2.2	Metode <i>Forward-Time Centered-Space</i> (FTCS)	18
III.2.3	Metode <i>Physics-Informed Neural Network</i> (PINN)	20
III.2.4	Ukuran Evaluasi Kinerja	21
III.3	Penyelesaian Persamaan Panas 2 Dimensi dengan Sumber Panas	21
III.3.1	Metode <i>Physics-Informed Neural Network</i> (PINN)	22
III.4	Spesifikasi <i>Hardware</i> yang Digunakan	23
BAB IV HASIL DAN ANALISIS	24	
IV.1	Simulasi Persebaran Panas 1D.....	24
IV.1.1	Penyelesaian Metode Separasi Variabel	24
IV.1.2	Penyelesaian Metode FTCS	25
IV.1.3	Penyelesaian Metode PINN	26
IV.1.4	Evaluasi Kinerja Metode FTCS dan PINN pada Persamaan Pa- nas 1D.....	28
IV.2	Simulasi Persebaran Panas 2D.....	30
IV.2.1	Penyelesaian Metode Separasi Variabel	30
IV.2.2	Penyelesaian Metode FTCS	31
IV.2.3	Penyelesaian Metode PINN	32
IV.2.4	Evaluasi Kinerja Metode FTCS dan PINN pada Persamaan Pa- nas 2D.....	33
IV.3	Simulasi Persebaran Panas 2D dengan Sumber Panas	35
IV.3.1	Penyelesaian Metode PINN	35
BAB V SIMPULAN DAN PELUANG RISET LANJUTAN	42	
V.1	Simpulan	42
V.2	Peluang Riset Lanjutan	43
DAFTAR PUSTAKA	44	

LAMPIRAN A: KONFIGURASI PINN DAN FTCS	46
LAMPIRAN B: VARIASI STRUKTUR PINN	47
LAMPIRAN C: KODE PROGRAM	48
C.1 PROGRAM PERSAMAAN PANAS 1D	48
C.2 PROGRAM PERSAMAAN PANAS 2D	55
C.3 PROGRAM PERSAMAAN PANAS DENGAN SUMBER PANAS	67
C.3.1 Variasi Sumber Panas di Tengah Domain	67
C.3.2 Variasi Sumber Panas di Pinggir Domain	74

DAFTAR NOTASI

Notasi	Arti
α	difusivitas termal
u	temperatur
$N^L(\mathbf{x})$	<i>neural network</i> dengan jumlah <i>layer</i> L
$\sigma(x)$	fungsi aktivasi
\mathbf{W}^ℓ	matriks beban/ <i>weights</i> pada <i>layer</i> ke ℓ
\mathbf{b}^ℓ	vektor <i>bias</i> pada <i>layer</i> ke ℓ
\mathcal{B}	syarat batas PDP
θ	kumpulan dari beban/ <i>weights</i> dan <i>bias</i>
\mathcal{L}	<i>loss function</i>
\mathbb{R}^n	himpunan bilangan real pada dimensi n

DAFTAR SINGKATAN

Notasi	Arti
BC	<i>boundary condition / syarat batas</i>
FDM	<i>Finite Difference Method</i>
FTCS	<i>Forward-Time Centered-Space</i>
IC	<i>initial condition / nilai awal</i>
MSE	<i>mean squared error</i>
NN	<i>neural network</i>
PDP	persamaan diferensial parsial
PINN	<i>Physics-Informed Neural Network</i>

DAFTAR GAMBAR

Gambar II.1	Ilustrasi sebuah <i>Perceptron</i> [Nielsen, 2015]	7
Gambar II.2	Ilustrasi sederhana dari <i>Deep Neural Network</i> [Nielsen, 2015].	8
Gambar II.3	Ilustrasi dari PINN untuk persamaan panas 1 dimensi (III.1) ..	11
Gambar III.1	<i>Flowchart</i> proses perolehan data metode FTCS untuk persamaan panas 1 dimensi (III.1).....	14
Gambar III.2	<i>Flowchart</i> proses <i>training</i> PINN	16
Gambar III.3	<i>Flowchart</i> proses perolehan data metode FTCS untuk persamaan panas 2 dimensi (III.12)	19
Gambar III.4	Ilustrasi dari PINN untuk persamaan panas 2 dimensi(III.12)..	21
Gambar III.5	Ilustrasi dari PINN untuk persamaan panas 2 dimensi dengan sumber panas (III.21)	23
Gambar IV.1	Solusi persamaan panas 1 dimensi (III.1) menggunakan metode analitik (separasi variabel)	24
Gambar IV.2	Solusi persamaan panas 1 dimensi (III.1) menggunakan metode FTCS	25
Gambar IV.3	Solusi persamaan panas 1 dimensi (III.1) menggunakan PINN	26
Gambar IV.4	Perkembangan nilai <i>loss function</i> (\mathcal{L}) pada PINN untuk persamaan panas 1 dimensi (III.1).....	27
Gambar IV.5	Perbandingan antara metode analitik, FTCS, dan PINN pada penyelesaian persamaan panas 1 dimensi (III.1).....	28
Gambar IV.6	Solusi persamaan panas 2 dimensi (III.12) menggunakan metode analitik (separasi variabel)	30
Gambar IV.7	Solusi persamaan panas 2 dimensi (III.12) menggunakan metode FTCS	31
Gambar IV.8	Solusi persamaan panas 2 dimensi (III.12) menggunakan PINN	32
Gambar IV.9	Perkembangan nilai <i>loss function</i> (\mathcal{L}) pada PINN untuk persamaan panas 2 dimensi (III.12)	33
Gambar IV.10	Solusi persamaan panas 2 dimensi dengan sumber panas (III.21) dengan variasi sumber panas di titik (0.5, 0.5) menggunakan PINN	36

Gambar IV.11 Perkembangan nilai <i>loss function</i> (\mathcal{L}) pada PINN untuk persamaan panas 2 dimensi dengan sumber panas (III.21) dengan variasi sumber panas di titik (0.5, 0.5)	37
Gambar IV.12 Perkembangan temperatur terhadap waktu pada 6 titik di dalam domain solusi persamaan panas 2 dimensi dengan sumber panas (III.21) dengan variasi sumber panas di titik (0.5, 0.5) ..	38
Gambar IV.13 Solusi persamaan panas 2 dimensi dengan sumber panas (III.21) dengan variasi sumber panas di titik (0.25, 0.25) menggunakan PINN	39
Gambar IV.14 Perkembangan nilai <i>loss function</i> (\mathcal{L}) pada PINN untuk persamaan panas 2 dimensi dengan sumber panas (III.21) dengan variasi sumber panas di titik (0.25, 0.25)	40
Gambar IV.15 Perkembangan temperatur terhadap waktu pada 6 titik di dalam domain solusi persamaan panas 2 dimensi dengan sumber panas (III.21) dengan variasi sumber panas di titik (0.25, 0.25)	41

DAFTAR TABEL

Tabel II.1 Contoh penggunaan AD untuk mengevaluasi diferensial parsial $\frac{\partial y}{\partial x_1}$ dan $\frac{\partial y}{\partial x_2}$ pada $(x, y) = (2, 1)$	10
Tabel IV.1 Perbandingan tiap metode dalam penyelesaian persamaan panas 1 dimensi (III.1)	29
Tabel IV.2 Perbandingan tiap metode dalam penyelesaian persamaan panas 2 dimensi (III.12)	34
Tabel A.1 Konfigurasi PINN yang digunakan pada persamaan panas 1D dan 2D	46
Tabel A.2 Konfigurasi FTCS yang digunakan pada persamaan panas 1D dan 2D	46
Tabel B.1 Perbandingan hasil variasi PINN pada penyelesaian persamaan panas 1 dimensi (III.1)	47

BAB I

PENDAHULUAN

I.1 Latar Belakang

Persamaan diferensial parsial (PDP) adalah persamaan yang menyatakan hubungan antara fungsi yang memiliki dua atau lebih variabel bebas dan turunan parsial dari fungsi terhadap variabel bebas tersebut. PDP muncul di sebagian besar bidang teknik dan sains. Terdapat banyak proses fisis diatur oleh persamaan diferensial parsial. Salah satunya adalah persamaan panas yang mendeskripsikan persebaran panas pada daerah tertentu.

Solusi dari persamaan diferensial parsial dapat diperoleh secara analitik ketika permasalahannya sederhana dan tidak rumit. Jika permasalahannya rumit, dapat dilakukan diskritisasi domain masalah. Selanjutnya, metode *finite difference* (FDM) umum digunakan dalam menyelesaikan permasalahan tersebut.

Salah satu teknik penyelesaian PDP yang baru muncul akhir-akhir ini adalah *Physics-Informed Neural Network* (PINN). PINN sendiri bekerja dengan prinsip dasar seperti *neural network*, namun memanfaatkan hukum-hukum fisika sebagai salah satu fungsi objektif nya. Akan dianalisis performa PINN dalam menyelesaikan permasalahan persebaran panas. Hasil dari PINN akan dibandingkan bersama metode FDM dengan hasil analitik sebagai acuan. Selanjutnya, akan dipelajari solusi persamaan panas dua dimensi dengan kondisi adanya sumber panas menggunakan *Physics-Informed Neural Network* (PINN).

I.2 Rumusan dan Batasan Masalah

Rumusan masalah pada penelitian ini adalah pencarian alternatif solusi PDP yang lebih teliti melalui *Physics-Informed Neural Network* (PINN) dan membandingkan hasilnya dengan metode *finite difference* (metode beda hingga) dengan skema *Forward-Time Centered-Space* (FTCS) serta mempelajari solusi persamaan panas dua dimensi dengan sumber panas menggunakan *Physics-Informed Neural Network* (PINN). Masalah pene-

litian ini dibatasi pada persamaan yang dianalisis, yang merupakan persamaan panas pada satu dan dua dimensi, dengan syarat batas Dirichlet dan syarat awal fungsi sinus, beserta persamaan panas pada dua dimensi dengan adanya sumber panas.

I.3 Tujuan

Tujuan dari penelitian ini adalah:

1. Menentukan solusi persamaan panas satu dimensi dan dua dimensi dengan metode analitik (separasi variabel); metode *finite difference* / beda hingga dengan skema *Forward-Time Centered-Space* (FTCS); dan dengan *Physics-Informed Neural Network* (PINN).
2. Membandingkan hasil penyelesaian persamaan panas metode analitik, FTCS, dan PINN.
3. Mempelajari solusi persamaan panas dua dimensi dengan sumber panas menggunakan *Physics-Informed Neural Network* (PINN).

I.4 Metodologi

Secara umum, penelitian ini dibagi menjadi beberapa tahapan:

1. Studi literatur

Studi literatur dilakukan dalam mempelajari persebaran panas dan persamaan panas beserta metode-metode penyelesaiannya.

2. Simulasi solusi persamaan panas

Dilakukan simulasi hasil persamaan panas menggunakan metode numerik FTCS dan model PINN.

3. Pengolahan data

Berdasarkan hasil simulasi FTCS dan PINN, dilakukan visualisasi solusi persamaan panas serta penentuan nilai *mean-squared error* MSE untuk tiap metode.

I.5 Sistematika Penulisan

Setelah Pendahuluan pada Bab I ini, Bab II akan mengulas tentang teori dari persamaan panas serta masing-masing metode solusi yang akan disimulasikan pada penelitian ini. Bab III menjelaskan terkait metode yang digunakan pada penelitian ini yang meliputi penyelesaian persamaan panas 1 dan 2 dimensi menggunakan metode separasi variabel, FTCS, dan PINN; ukuran evaluasi kinerja yang akan digunakan; serta penyelesaian persamaan panas 2 dimensi dengan sumber panas menggunakan PINN. Bab IV memaparkan tentang hasil yang diperoleh dari pengolahan data, visualisasi, hingga analisis kinerjanya. Terakhir, Bab V menyajikan simpulan yang diperoleh dari hasil penelitian serta peluang riset lanjutan.

BAB II

TINJAUAN PUSTAKA

II.1 Persamaan Diferensial Parsial

Persamaan diferensial parsial (PDP) adalah persamaan yang menyatakan hubungan antara fungsi yang memiliki dua atau lebih variabel bebas dan turunan parsial dari fungsi terhadap variabel bebas tersebut. Umumnya, dalam sebagian besar masalah di bidang teknik dan sains, variabel bebasnya adalah ruang (x, y, z) atau ruang dan waktu (x, y, z, t) . Variabel terikat tergantung pada masalah fisis yang dimodelkan. Solusi dari persamaan diferensial parsial adalah fungsi tertentu, misal $f(x, y)$ atau $f(x, t)$, yang memenuhi PDE dalam domain tertentu, dan memenuhi kondisi awal dan/atau batas yang ditentukan pada batas domain yang diminati [Hoffman and Frankel, 2001].

II.2 Persamaan Panas

Persamaan diferensial parsial (II.1), atau yang sering dikenal dengan persamaan panas, mendeskripsikan sebaran temperatur u dengan difusivitas termal medium α dan dituliskan sebagai berikut: [Widder, 1976]

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u \quad (\text{II.1})$$

dengan $u = u(t, \mathbf{x})$, t bervariasi pada interval \mathbb{R} , dan $\mathbf{x} \in \mathbb{R}^n$.

Persamaan panas memiliki aplikasi penting pada berbagai permasalahan sains seperti konduksi panas [Widder, 1976], transpor dan difusi polutan [Karima and Magdalena, 2021, Maitisa et al., 2021], serta deteksi polutan [Badia and Ha-Duong, 2002].

Difusivitas termal medium (α)

Difusivitas termal medium adalah ukuran kemampuan suatu medium untuk menghantarkan energi panas relatif terhadap kemampuannya untuk menyimpan energi panas.

Difusivitas termal dapat diperoleh melalui persamaan berikut. [Bird et al., 2006]

$$\alpha = \frac{k}{\rho c_p} \quad (\text{II.2})$$

k merupakan konduktivitas termal, ρ massa jenis medium, serta c_p kapasitas panas spesifik pada tekanan tetap. Diketahui konduktivitas termal, massa jenis, serta kapasitas panas spesifik udara pada suhu 25°C dan tekanan 1 atm adalah masing-masing $2.612 \times 10^{-2} Jm^{-1}s^{-1}K^{-1}$, $1.168 \times 10^{-3} kgm^{-3}$, dan $1005 Jkg^{-1}K^{-1}$ [Lemmon et al., 2000]. Sehingga difusivitas termal udara pada suhu 25°C dan tekanan 1 atm adalah:

$$\alpha = 2.225 \times 10^{-5} m^2 s^{-1} \quad (\text{II.3})$$

II.3 Solusi dari Persamaan Panas

II.3.1 *Separation of Variables*

Metode yang umum digunakan untuk menyelesaikan persamaan diferensial parsial seperti persamaan (II.1) secara analitis adalah separasi variabel. Pada metode ini, solusi temperatur dipisah menjadi dua fungsi, fungsi yang bergantung hanya pada koordinat spasial dan fungsi yang bergantung hanya pada koordinat waktu. Sehingga dapat dituliskan sebagai berikut [Boas, 2005]:

$$u = F(\mathbf{x})T(t), \quad (\text{II.4})$$

Dalam beberapa kasus khusus, solusi dari persamaan diferensial parsial dapat dinyatakan dalam ekspresi bentuk tertutup dengan menggunakan metode ini. Namun, pada kenyataannya, sebagian besar masalah di bidang teknik dan sains yang melibatkan persamaan diferensial parsial tidak bisa diselesaikan dengan metode ini [Hoffman and Frankel, 2001].

II.3.2 Metode *Forward-Time Centered-Space* (FTCS)

Metode *Forward-Time Centered-Space* adalah salah satu jenis pendekatan persamaan diferensial parsial menggunakan *Finite Difference Method* (FDM). FDM atau metode beda hingga adalah salah satu metode penyelesaian persamaan diferensial parsial (seperti persamaan II.1) secara numerik. Tujuan dari metode beda hingga untuk menyelesaikan

persamaan diferensial parsial adalah mengubah persoalan kalkulus menjadi persoalan aljabar dengan cara:

1. Diskritisasi domain ruang dan waktu yang kontinu ke dalam kisi yang diskrit
2. Dekati masing-masing turunan parsial eksak yang ada dalam persamaan diferensial parsial dengan pendekatan beda hingga
3. Substitusikan pendekatan ini ke dalam persamaan diferensial parsial untuk memperoleh persamaan beda hingga
4. Selesaikan persamaan beda hingga yang dihasilkan

Pada metode FTCS, titik dasar untuk pendekatan beda hingga dari persamaan diferensial parsial adalah titik ke (i, n) pada kisi. Turunan parsial pertama terhadap waktu didekati dengan aproksimasi *forward-time* orde pertama, sedangkan turunan parsial kedua terhadap ruang didekati dengan aproksimasi *centered-space* orde kedua. Diberikan pendekatan FTCS untuk dua dimensi ruang dan satu dimensi waktu sebagai berikut. [Hoffman and Frankel, 2001]

$$\frac{\partial u}{\partial t} = \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} \quad (\text{II.5a})$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} \quad (\text{II.5b})$$

$$\frac{\partial^2 u}{\partial y^2} = \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2} \quad (\text{II.5c})$$

Metode von Neumann dalam Analisis Stabilitas Persamaan Beda Hingga

Persamaan beda hingga dikatakan stabil jika menghasilkan solusi terbatas untuk persamaan diferensial parsial yang stabil, dan dikatakan tidak stabil jika menghasilkan solusi tak terbatas untuk persamaan diferensial parsial yang stabil.

Untuk menganalisis kestabilan persamaan beda hingga, digunakan metode von Neumann yang menyatakan bahwa aproksimasi FTCS dari persamaan panas (II.1) untuk kasus satu dimensi ruang dan satu dimensi waktu ($u(t, \mathbf{x}) = u(t, x)$) harus memenuhi

persyaratan berikut: [Hoffman and Frankel, 2001]

$$\frac{\alpha \Delta t}{(\Delta x)^2} \leq \frac{1}{2} \quad (\text{II.6})$$

sedangkan untuk kasus dua dimensi ruang dan satu dimensi waktu ($u(t, \mathbf{x}) = u(t, x, y)$) harus memenuhi persyaratan berikut: [Hoffman and Frankel, 2001]

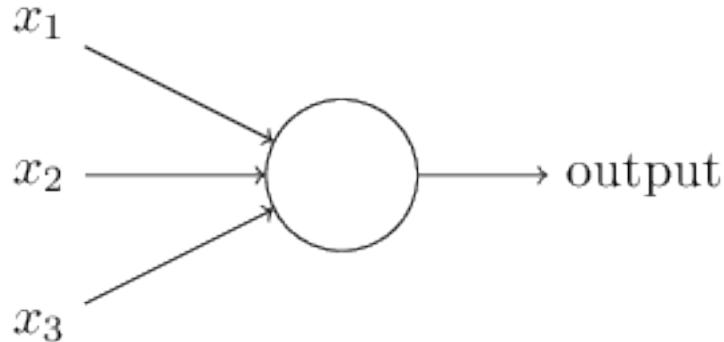
$$\alpha \Delta t \frac{(\Delta x)^2 + (\Delta y)^2}{(\Delta x \Delta y)^2} \leq \frac{1}{2} \quad (\text{II.7})$$

II.3.3 *Physics-Informed Neural Network (PINN)*

Metode *Physics-Informed Neural Network* (PINN) memiliki empat bagian penting, yaitu *perceptron*, *Deep Neural Network*, *Automatic Differentiation* (AD), dan bagian *Physics-Informed* dari PINN [Lu et al., 2021].

Perceptron

Perceptron adalah sebuah fungsi yang mengambil sejumlah input (x_1, x_2, x_3, \dots) dan menghasilkan satu buah keluaran. Di dalam sebuah *perceptron* terdapat sejumlah be-



Gambar II.1: Ilustrasi sebuah *Perceptron* [Nielsen, 2015]

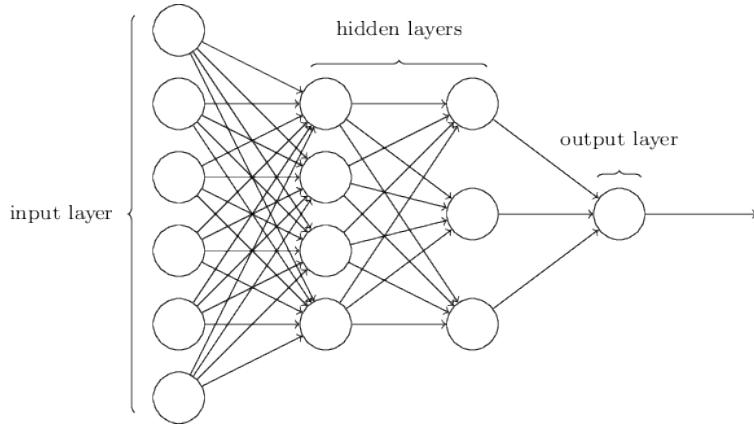
an (*weights*; (w_1, w_2, w_3, \dots)) dan *bias* (b_1, b_2, b_3, \dots) yang menggambarkan tingkat kepentingan dari suatu input terhadap output *perceptron*. Kemudian, hasil dari perhitungan input tersebut dimasukkan ke sebuah fungsi aktivasi ($\sigma(x)$). Secara matematis

output setiap *perceptron* dievaluasi seperti berikut: [Nielsen, 2015]

$$\text{output} = \sigma \left(\sum_i (w_i x_i + b_i) \right) \quad (\text{II.8})$$

Deep Neural Network

Deep Neural Network adalah suatu struktur jaringan kompleks dari *perceptron* yang memiliki *hidden layer* lebih dari satu.



Gambar II.2: Ilustrasi sederhana dari *Deep Neural Network* [Nielsen, 2015]

Setiap input pada *input layer* (lapisan pertama) terhubung ke setiap perceptron pada *layer* berikutnya. Begitu seterusnya hingga diperoleh output pada *layer* terakhir.

$\mathcal{N}^L(\mathbf{x}) : \mathbb{R}^{d_{in}} \rightarrow \mathbb{R}^{d_{out}}$ didefinisikan sebagai *neural network* dengan L buah *layer*, dengan *layer* ℓ memiliki N^ℓ buah *neuron* ($N_0 = d_{in}$ dan $N_L = d_{out}$). Matriks beban (*weight*) dan vektor *bias* pada *layer* ℓ masing-masing didefinisikan sebagai $\mathbf{W}^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ dan $\mathbf{b}^\ell \in \mathbb{R}^{N_\ell}$. Menggunakan fungsi aktivasi yang didefinisikan sebagai σ , *neural network* didefinisikan secara rekursif seperti berikut: [Lu et al., 2021]

$$\text{input layer:} \quad \mathcal{N}^0(\mathbf{x}) = \mathbf{x} \in \mathbb{R}^{d_{in}} \quad (\text{II.9a})$$

$$\text{hidden layers:} \quad \mathcal{N}^\ell(\mathbf{x}) = \sigma \left(\mathbf{W}^\ell \mathcal{N}^{\ell-1}(\mathbf{x}) + \mathbf{b}^\ell \right) \in \mathbb{R}^{N^\ell} \quad (\text{II.9b})$$

$$\text{output layer:} \quad \mathcal{N}^L(\mathbf{x}) = \mathbf{W}^L \mathcal{N}^{L-1}(\mathbf{x}) + \mathbf{b}^L \in \mathbb{R}^{d_{out}} \quad (\text{II.9c})$$

Fungsi aktivasi (σ) yang biasanya digunakan adalah fungsi logistik $1/(1 + e^{-x})$, fungsi tangen hiperbolik ($\tanh(x)$), dan fungsi *rectified linear unit* (ReLU; $\max\{x, 0\}$).

Perlu dilakukan proses *training* pada *neural network* dalam aproksimasi $\mathcal{N}(\mathbf{x})$ untuk semua input \mathbf{x} . Proses ini melibatkan *data training* yang berupa input bersamaan dengan keluaran yang tepat. Untuk mengukur seberapa baik model yang dibentuk, didefinisikan fungsi kerugian (*loss function*). Beberapa contoh dari *loss function* diantaranya adalah *mean-squared error* (MSE) dan *cross-entropy loss function* [Nielsen, 2015].

Automatic Differentiation (AD)

Dalam PINN, untuk mengevaluasi *loss function* dari *neural network* diperlukan nilai diferensial dari output terhadap input yang diberikan. Umumnya, terdapat empat cara untuk mengevaluasi nilai diferensial tersebut: [Baydin et al., 2015] (1) menghitung turunan tiap output terhadap input secara manual; (2) menghitung turunan secara numerik menggunakan FDM; (3) menghitung turunan secara simbolik menggunakan manipulasi ekspresi (seperti pada software *Mathematica*); (4) *automatic differentiation* (AD). Pada PINN, digunakan metode AD dalam mengevaluasi nilai diferensial.

AD berlandaskan pada fakta bahwa semua perhitungan numerik yang dilakukan oleh komputer merupakan komposisi dari sekumpulan operasi elementer yang turunannya sudah diketahui [Verma, 2000]. Dengan menggabungkan turunan dari sekumpulan operasi tersebut melalui aturan rantai, dapat diperoleh turunan dari keseluruhan komposisi. Berikut adalah contoh dari penggunaan AD.

$$v = -2x_1 + 3x_2 + 0.5 \quad (\text{II.10a})$$

$$h = \tanh v \quad (\text{II.10b})$$

$$y = 2h - 1 \quad (\text{II.10c})$$

Tabel II.1: Contoh penggunaan AD untuk mengevaluasi diferensial parsial $\frac{\partial y}{\partial x_1}$ dan $\frac{\partial y}{\partial x_2}$ pada $(x, y) = (2, 1)$

<i>Forward pass</i>	<i>Backward pass</i>
$x_1 = 2$	$\frac{\partial y}{\partial y} = 1$
$x_2 = 1$	
$v = -2x_1 + 3x_2 + 0.5 = -0.5$	$\frac{\partial y}{\partial h} = \frac{\partial 2h-1}{\partial h} = 2$
$h = \tanh v \approx -0.462$	$\frac{\partial y}{\partial v} = \frac{\partial y}{\partial h} \frac{\partial h}{\partial v} = \frac{\partial y}{\partial h} \operatorname{sech} v^2 \approx 1.573$
$y = 2h - 1 = -1.924$	$\frac{\partial y}{\partial x_1} = \frac{\partial y}{\partial v} \frac{\partial v}{\partial x_1} = \frac{\partial y}{\partial v} \times (-2) = -3.416$ $\frac{\partial y}{\partial x_2} = \frac{\partial y}{\partial v} \frac{\partial v}{\partial x_2} = \frac{\partial y}{\partial v} \times 3 = 4.719$

Bagian *Physics-Informed* dari PINN

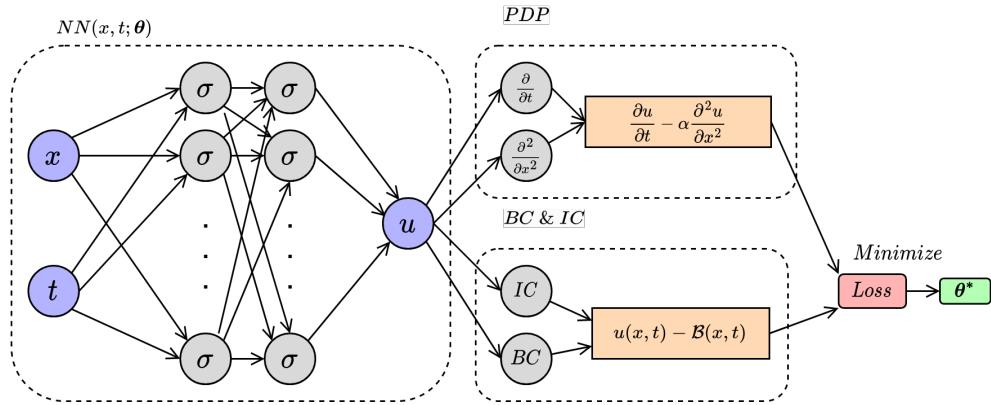
Physics-Informed Neural Network bekerja sama seperti *neural network* pada biasanya. Akan tetapi, proses *training neural network* ini dibantu oleh hukum-hukum yang sudah mapan, yaitu hukum fisika. PINN memanfaatkan hukum fisika sebagai *loss function*-nya. [Raissi et al., 2017]

Tinjau suatu hukum fisika dalam bentuk PDP lengkap dengan syarat batasnya seperti berikut:

$$f \left(\mathbf{x}; \frac{\partial u}{\partial x_1} \dots \frac{\partial u}{\partial x_d}; \frac{\partial u}{\partial x_1 \partial x_1} \dots \frac{\partial u}{\partial x_1 \partial x_d}; \dots; \lambda \right) = 0, \quad \mathbf{x} \in \Omega \quad (\text{II.11a})$$

$$\mathcal{B}(u, \mathbf{x}) = 0, \quad \mathbf{x} \in \partial\Omega \quad (\text{II.11b})$$

dengan parameter λ dan solusi $u(\mathbf{x})$ dengan $\mathbf{x} = (x_1, \dots, x_d)$ yang terdefinisi pada domain $\Omega \subset \mathbb{R}^d$. $\mathcal{B}(u, \mathbf{x})$ merupakan syarat batas Dirichlet, Neumann, Robin, atau periodik, sesuai dengan model yang di tinjau [Lu et al., 2021].



Gambar II.3: Ilustrasi dari PINN untuk persamaan panas 1 dimensi (III.1)

Dengan adanya PDP beserta BC dan IC-nya sebagai *loss function* ini, mereka berfungsi sebagai "jawaban" keluaran yang tepat dari *neural network* yang dibentuk. *Loss function* tersebut akan dioptimasi (diminimalkan) hingga ditemukan sekumpulan beban (*weights*) dan *bias* ($\theta = \{\mathbf{W}^\ell, \mathbf{b}^\ell\}_{1 \leq \ell \leq L}$) yang ideal (θ^*) untuk mendeskripsikan PDP tersebut. [Lu et al., 2021].

BAB III

METODOLOGI PENELITIAN

III.1 Penyelesaian Persamaan Panas 1 Dimensi

Akan dicari solusi persamaan panas (II.1) dalam satu dimensi ruang dan satu dimensi waktu dengan syarat batas Dirichlet dan nilai awal sebagai berikut.

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \quad x \in [0, 1], \quad t \in [0, 1] \quad (\text{III.1a})$$

$$u(0, t) = u(1, t) = 0 \quad (\text{III.1b})$$

$$u(x, 0) = \sin(\pi x) \quad (\text{III.1c})$$

Digunakan tiga metode untuk menyelesaikan persamaan panas tersebut, yaitu metode separasi variabel, metode *Forward-Time Centered-Space* (FTCS), serta metode *Physics-Informed Neural Network* (PINN).

III.1.1 Metode Separasi Variabel

Substitusi persamaan (II.4) ke persamaan (III.1a), sehingga diperoleh

$$\alpha T \frac{\partial^2 F}{\partial x^2} = F \frac{\partial T}{\partial t} \quad (\text{III.2a})$$

$$\frac{1}{F} \frac{\partial^2 F}{\partial x^2} = \frac{1}{\alpha T} \frac{\partial T}{\partial t} \quad (\text{III.2b})$$

Ruas kiri dan kanan persamaan (III.2b) adalah fungsi dari x saja atau t saja, sehingga keduanya bernilai konstan dan dapat dituliskan:

$$\frac{1}{F} \frac{\partial^2 F}{\partial x^2} = -k^2 \quad \Rightarrow \quad \frac{\partial^2 F}{\partial x^2} + k^2 F = 0 \quad (\text{III.3a})$$

$$\frac{1}{\alpha T} \frac{\partial T}{\partial t} = -k^2 \quad \Rightarrow \quad \frac{\partial T}{\partial t} = -k^2 \alpha T \quad (\text{III.3b})$$

Dengan k adalah suatu konstanta. Solusi dari kedua persamaan tersebut adalah:

$$F(x) = A \sin(kx - \phi) \quad (\text{III.4a})$$

$$T(t) = e^{-k^2\alpha t} \quad (\text{III.4b})$$

Sehingga,

$$u(x, t) = F(x)T(t) = e^{-k^2\alpha t} A \sin(kx - \phi) \quad (\text{III.5})$$

Substitusikan syarat batas dan nilai awal (III.1b) dan (III.1c). Diperoleh $k = \pi$ sehingga:

$$u(x, t) = e^{-\pi^2\alpha t} \sin(\pi x) \quad (\text{III.6})$$

dengan dipilih $\alpha = 0.4$.

III.1.2 Metode *Forward-Time Centered-Space* (FTCS)

Substitusi persamaan (II.5a) dan (II.5b) ke persamaan (III.1a). Sehingga diperoleh solusi:

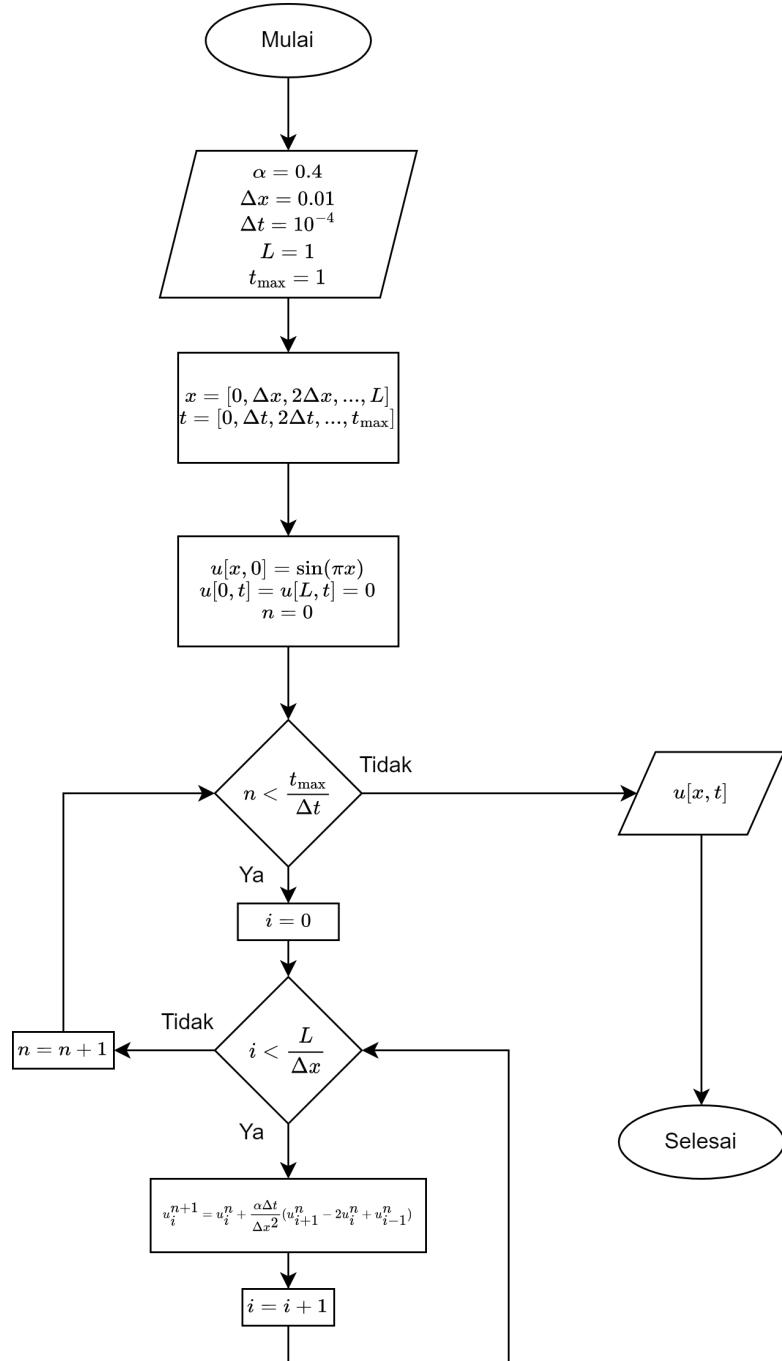
$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \quad (\text{III.7a})$$

$$u_i^{n+1} = u_i^n + \frac{\alpha \Delta t}{\Delta x^2} (u_{i+1}^n - 2u_i^n + u_{i-1}^n) \quad (\text{III.7b})$$

Dipilih nilai $\alpha = 0.4$ dan $\Delta x = 0.01$, maka sesuai metode von Neumann (II.6) diperoleh:

$$\Delta t \leq 1.25 \times 10^{-4} \quad (\text{III.8})$$

Dipilih nilai Δt yaitu $\Delta t = 1 \times 10^{-4}$. Dari hasil tersebut, dilakukan komputasi secara numerik untuk memperoleh data solusi persamaan panas dengan metode FTCS. Alur *flowchart* proses komputasi dan perhitungan dalam melakukan pengambilan data divisualisasikan pada Gambar III.1.



Gambar III.1: Flowchart proses perolehan data metode FTCS untuk persamaan panas 1 dimensi (III.1)

III.1.3 Metode *Physics-Informed Neural Network* (PINN)

Untuk metode PINN, digunakan model *neural network* yang sama seperti persamaan (II.9). Kemudian, dipilih sejumlah titik acak sebanyak $|\mathcal{T}|$ yang akan menjadi *data training* ($\mathcal{T} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{|\mathcal{T}|}\}$). \mathcal{T} terdiri dari dua set, $\mathcal{T}_b \subset \partial\Omega$ (pada batas daerah) dan $\mathcal{T}_f \subset \Omega$ (di dalam daerah) dengan $(\mathcal{T}_f, \mathcal{T}_b) = (2540, 240)$. Selanjutnya, ditentukan PDP, BC, dan IC yang digunakan sebagai *loss function* (\mathcal{L}), yaitu seperti pada persamaan (II.11) dan (III.1) [Lu et al., 2021].

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = \omega_f \mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) + \omega_b \mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b) \quad (\text{III.9a})$$

$$\mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \left\| \frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} \right\|_2^2 \quad (\text{III.9b})$$

$$\mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b) = \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \|\mathcal{B}(u, \mathbf{x})\|_2^2 \quad (\text{III.9c})$$

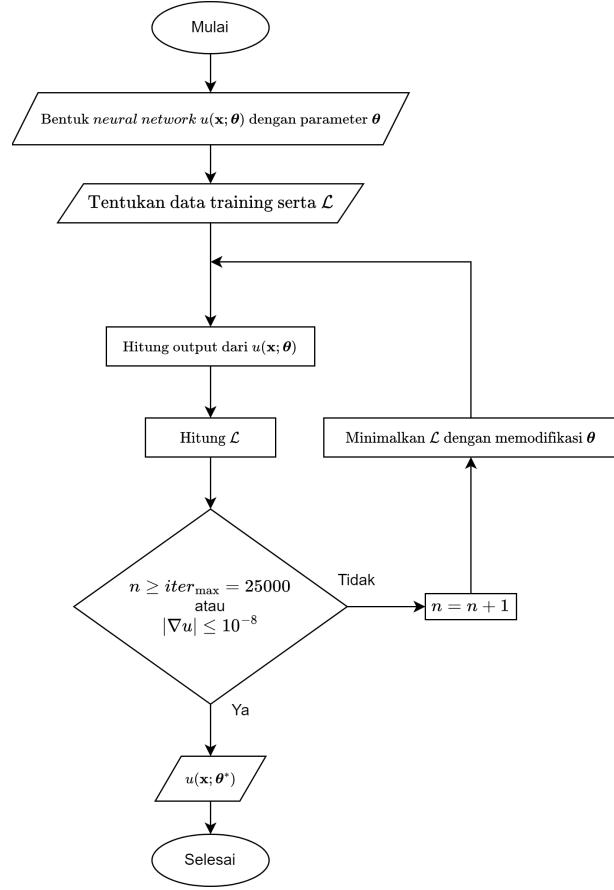
Disini \mathcal{L}_f dan \mathcal{L}_b merupakan kontribusi *loss function* masing-masing dari PDP serta BC & IC. *Loss function* kemudian didefinisikan sebagai *weighted sum* dengan beban $(\omega_f, \omega_b) = (1, 1)$. Turunan parsial yang terlibat dalam *loss function* dievaluasi menggunakan metode *automatic differentiation* (AD).

Fungsi aktivasi yang akan digunakan adalah fungsi tangen hiperbolik:

$$\sigma(x) = \tanh x \quad (\text{III.10})$$

serta digunakan metode optimasi berbasis gradien: Adam dan L-BFGS [Kingma and Ba, 2014, Byrd et al., 1995]

Seperti yang telah dijelaskan sebelumnya, *loss function* (III.9) akan dioptimasi (diminimalkan) hingga ditemukan sekumpulan beban (*weights*) dan *bias* ($\boldsymbol{\theta} = \{\mathbf{W}^\ell, \mathbf{b}^\ell\}_{1 \leq \ell \leq L}$) yang ideal ($\boldsymbol{\theta}^*$) untuk memodelkan persamaan panas (III.1). Struktur dari PINN untuk persamaan panas ini ditunjukkan pada Gambar II.3. Alur *flowchart* proses *training* dari PINN divisualisasikan pada Gambar III.2.



Gambar III.2: Flowchart proses training PINN

III.1.4 Ukuran Evaluasi Kinerja

Untuk mengevaluasi kinerja metode FTCS dan metode PINN, keduanya akan dibandingkan dengan hasil analitik (separasi variabel) dengan mengevaluasi galat menggunakan *mean-squared error* (MSE).

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{u}(x, t) - u(x, t))^2 \quad (\text{III.11})$$

Dimana $\hat{u}(x, t)$ adalah solusi dari FTCS dan PINN, dan n adalah jumlah titik pada solusi FTCS dan PINN.

III.2 Penyelesaian Persamaan Panas 2 Dimensi

Akan dicari solusi persamaan panas (II.1) dalam dua dimensi ruang dan satu dimensi waktu dengan syarat batas Dirichlet dan nilai awal sebagai berikut.

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right), \quad x \in [0, 1], \quad y \in [0, 1], \\ t \in [0, 0.5] \quad (\text{III.12a})$$

$$u(0, y, t) = u(1, y, t) = u(x, 0, t) = u(x, 1, t) = 0 \quad (\text{III.12b})$$

$$u(x, y, 0) = \sin(\pi x) \sin(\pi y) \quad (\text{III.12c})$$

Digunakan tiga metode untuk menyelesaikan persamaan panas tersebut, yaitu metode separasi variabel, metode *Forward-Time Centered-Space* (FTCS), serta metode *Physics-Informed Neural Network* (PINN).

III.2.1 Metode Separasi Variabel

Substitusi persamaan (II.4) (dengan $F(\mathbf{x}) = F(x)G(y)$) ke persamaan (III.12a), sehingga diperoleh

$$\alpha T \left(G \frac{\partial^2 F}{\partial x^2} + F \frac{\partial^2 G}{\partial y^2} \right) = FG \frac{\partial T}{\partial t} \quad (\text{III.13a})$$

$$\frac{1}{F} \frac{\partial^2 F}{\partial x^2} + \frac{1}{G} \frac{\partial^2 G}{\partial y^2} = \frac{1}{\alpha T} \frac{\partial T}{\partial t} \quad (\text{III.13b})$$

Ruas kiri dan kanan persamaan (III.13b) adalah fungsi dari x saja, y saja, atau t saja, sehingga keduanya bernilai konstan dan dapat dituliskan:

$$\frac{1}{F} \frac{\partial^2 F}{\partial x^2} + \frac{1}{G} \frac{\partial^2 G}{\partial y^2} = -(k^2 + l^2) \quad \Rightarrow \quad \begin{aligned} \frac{\partial^2 F}{\partial x^2} + k^2 F &= 0 \\ \frac{\partial^2 G}{\partial y^2} + l^2 G &= 0 \end{aligned} \quad (\text{III.14a})$$

$$\frac{1}{\alpha T} \frac{\partial T}{\partial t} = -(k^2 + l^2) \quad \Rightarrow \quad \frac{\partial T}{\partial t} = -(k^2 + l^2) \alpha T \quad (\text{III.14b})$$

Dengan k dan l adalah suatu konstanta. Solusi dari ketiga persamaan tersebut adalah:

$$F(x) = A \sin(kx - \phi) \quad (\text{III.15a})$$

$$G(x) = B \sin(lx - \phi) \quad (\text{III.15b})$$

$$T(t) = e^{-(k^2+l^2)\alpha t} \quad (\text{III.15c})$$

Sehingga,

$$u(x, y, t) = F(x)G(y)T(t) = AB e^{-(k^2+l^2)\alpha t} \sin(kx - \phi) \sin(lx - \phi) \quad (\text{III.16})$$

Substitusikan syarat batas dan nilai awal (III.12b) dan (III.12c). Diperoleh $k = \pi$ dan $l = \pi$ sehingga:

$$u(x, y, t) = e^{-2\pi^2\alpha t} \sin(\pi x) \sin(\pi y) \quad (\text{III.17})$$

dengan dipilih $\alpha = 0.4$.

III.2.2 Metode *Forward-Time Centered-Space* (FTCS)

Substitusi persamaan (II.5a), (II.5b), dan (II.5c) ke persamaan (III.12a). Sehingga diperoleh solusi:

$$\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} = \alpha \left(\frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} + \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2} \right) \quad (\text{III.18a})$$

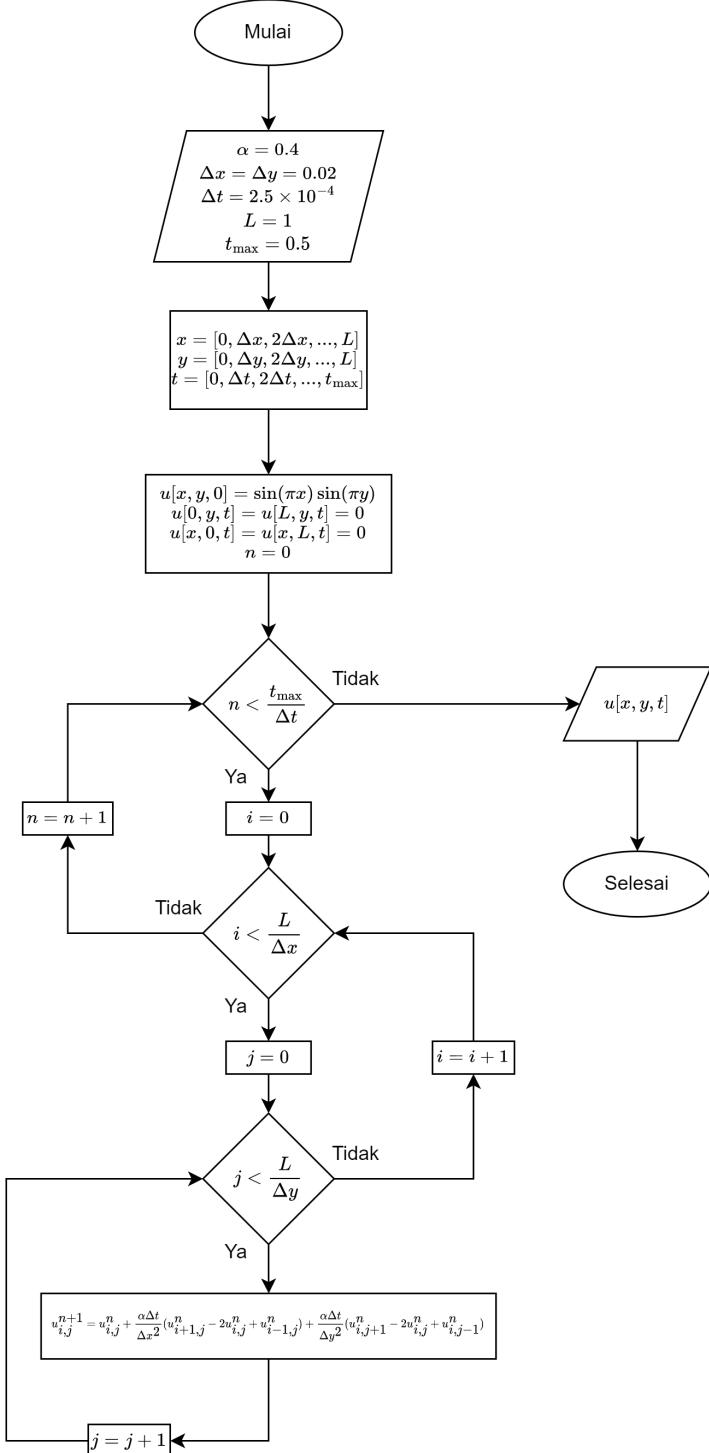
$$\begin{aligned} u_{i,j}^{n+1} &= u_i^n + \frac{\alpha \Delta t}{\Delta x^2} (u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) \\ &\quad + \frac{\alpha \Delta t}{\Delta y^2} (u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n) \end{aligned} \quad (\text{III.18b})$$

Dipilih nilai $\alpha = 0.4$ dan $\Delta x = \Delta y = 0.02$, maka sesuai metode von Neumann (II.7) diperoleh:

$$\Delta t \leq 2.5 \times 10^{-4} \quad (\text{III.19})$$

Dipilih nilai Δt terbesar, yaitu $\Delta t = 2.5 \times 10^{-4}$. Dari hasil tersebut, dilakukan komputasi secara numerik untuk memperoleh data solusi persamaan panas dengan metode FTCS. Alur *flowchart* proses komputasi dan perhitungan dalam melakukan pengambilan data

divisualisasikan pada Gambar III.3.



Gambar III.3: Flowchart proses perolehan data metode FTCS untuk persamaan panas 2 dimensi (III.12)

III.2.3 Metode *Physics-Informed Neural Network* (PINN)

Untuk metode PINN, digunakan model *neural network* yang sama seperti persamaan (II.9). Kemudian, dipilih sejumlah titik acak sebanyak $|\mathcal{T}|$ yang akan menjadi *data training* ($\mathcal{T} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{|\mathcal{T}|}\}$). \mathcal{T} terdiri dari dua set, $\mathcal{T}_b \subset \partial\Omega$ (pada batas daerah) dan $\mathcal{T}_f \subset \Omega$ (di dalam daerah) dengan $(\mathcal{T}_f, \mathcal{T}_b) = (5000, 600)$. Selanjutnya, ditentukan PDP, BC, dan IC yang digunakan sebagai *loss function* (\mathcal{L}), yaitu seperti pada persamaan (II.11) dan (III.12) [Lu et al., 2021].

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = \omega_f \mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) + \omega_b \mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b) \quad (\text{III.20a})$$

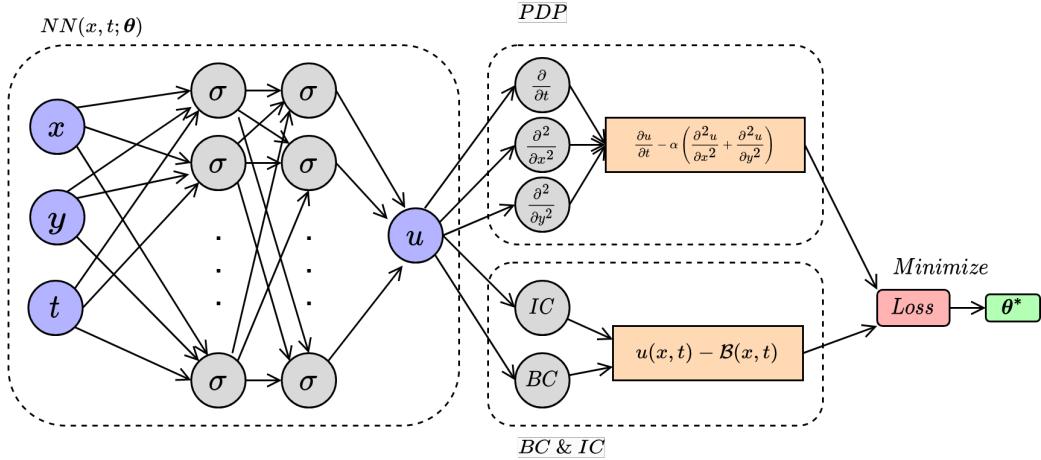
$$\mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \left\| \frac{\partial u}{\partial t} - \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \right\|_2^2 \quad (\text{III.20b})$$

$$\mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b) = \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \|\mathcal{B}(u, \mathbf{x})\|_2^2 \quad (\text{III.20c})$$

Disini \mathcal{L}_f dan \mathcal{L}_b merupakan kontribusi *loss function* masing-masing dari PDP serta BC & IC. *Loss function* kemudian didefinisikan sebagai *weighted sum* dengan beban $(\omega_f, \omega_b) = (1, 1)$. Turunan parsial yang terlibat dalam *loss function* dievaluasi menggunakan metode *automatic differentiation* (AD).

Fungsi aktivasi yang akan digunakan adalah fungsi tangen hiperbolik (III.10), serta digunakan metode optimasi berbasis gradien: Adam dan L-BFGS [Kingma and Ba, 2014, Byrd et al., 1995]

Seperti yang telah dijelaskan sebelumnya, *loss function* (III.20) akan dioptimasi (diminimalkan) hingga ditemukan sekumpulan beban (*weights*) dan *bias* ($\boldsymbol{\theta} = \{\mathbf{W}^\ell, \mathbf{b}^\ell\}_{1 \leq \ell \leq L}$) yang ideal ($\boldsymbol{\theta}^*$) untuk memodelkan persamaan panas (III.12). Struktur dari PINN untuk persamaan panas ini ditunjukkan pada Gambar III.4. Alur *flowchart* proses *training* dari PINN divisualisasikan pada Gambar III.2.



Gambar III.4: Ilustrasi dari PINN untuk persamaan panas 2 dimensi(III.12)

III.2.4 Ukuran Evaluasi Kinerja

Untuk mengevaluasi kinerja metode FTCS dan metode PINN, keduanya akan dibandingkan dengan hasil analitik (separasi variabel) dengan mengevaluasi galat menggunakan *mean-squared error* (MSE) (III.11) Dimana $\hat{u}(x, t)$ adalah solusi dari FTCS dan PINN, dan n adalah jumlah titik pada solusi FTCS dan PINN.

III.3 Penyelesaian Persamaan Panas 2 Dimensi dengan Sumber Panas

Akan dicari solusi persamaan panas (II.1) dalam dua dimensi ruang dan satu dimensi waktu dengan sumber panas, syarat batas Dirichlet, dan nilai awal sebagai berikut.

$$\frac{\partial u}{\partial t} = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + \frac{Q(x, y)}{\rho c_p}, \quad x \in [0, 1], \quad y \in [0, 1], \quad (III.21a)$$

$$t \in [0, 10000]$$

$$u(0, y, t) = u(1, y, t) = u(x, 0, t) = u(x, 1, t) = 25 \quad (III.21b)$$

$$u(x, y, 0) = 25 \quad (III.21c)$$

Persamaan (III.21) mendeskripsikan sumber panas yang tiba-tiba muncul di suatu titik. Disini $Q(x, y)$ merupakan fungsi sumber panas. Digunakan nilai ρ , c_p , dan α sesuai seperti pada (II.3) Akan dimodelkan sumber panas dari satu titik di tengah domain (titik

(0.5, 0.5)) dan satu titik di pinggir domain (titik (0.25, 0.25)) dengan menggunakan fungsi distribusi normal dalam dua variabel

$$Q(x, y) = \frac{A}{2\pi\sigma^2} \exp \left[-\frac{1}{2} \left(\left(\frac{x - \mu}{\sigma} \right)^2 + \left(\frac{y - \mu}{\sigma} \right)^2 \right) \right] \quad (\text{III.22})$$

dengan dipilih nilai $A = \frac{1000}{63.65}$ (agar puncak distribusi normal bernilai 1000), $\sigma = 0.05$, $\mu_1 = 0.5$ (titik (0.5, 0.5)), serta $\mu_2 = 0.25$ (titik (0.25, 0.25)). Akan digunakan metode *Physics-Informed Neural Network* (PINN) untuk menyelesaikan persamaan panas tersebut.

III.3.1 Metode *Physics-Informed Neural Network* (PINN)

Untuk metode PINN, digunakan model *neural network* yang sama seperti persamaan (II.9). Kemudian, dipilih sejumlah titik acak sebanyak $|\mathcal{T}|$ yang akan menjadi *data training* ($\mathcal{T} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{|\mathcal{T}|}\}$). \mathcal{T} terdiri dari dua set, $\mathcal{T}_b \subset \partial\Omega$ (pada batas daerah) dan $\mathcal{T}_f \subset \Omega$ (di dalam daerah) dengan $(\mathcal{T}_f, \mathcal{T}_b) = (5000, 600)$. Selanjutnya, ditentukan PDP, BC, dan IC yang digunakan sebagai *loss function* (\mathcal{L}), yaitu seperti pada persamaan (II.11) dan (III.21) [Lu et al., 2021].

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = \omega_f \mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) + \omega_b \mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b) \quad (\text{III.23a})$$

$$\mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \left\| \frac{\partial u}{\partial t} - \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - \frac{Q(x, y)}{\rho c_p} \right\|_2^2 \quad (\text{III.23b})$$

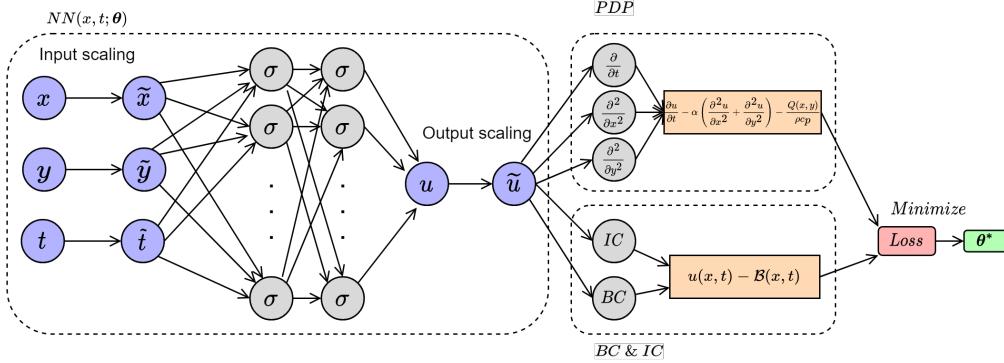
$$\mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b) = \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \|\mathcal{B}(u, \mathbf{x})\|_2^2 \quad (\text{III.23c})$$

Disini \mathcal{L}_f dan \mathcal{L}_b merupakan kontribusi *loss function* masing-masing dari PDP serta BC & IC. *Loss function* kemudian didefinisikan sebagai *weighted sum* dengan beban $(\omega_f, \omega_b) = (1000, 10^{-2})$. Turunan parsial yang terlibat dalam *loss function* dievaluasi menggunakan metode *automatic differentiation* (AD).

Fungsi aktivasi yang akan digunakan adalah fungsi tangen hiperbolik (III.10), serta digunakan metode optimasi berbasis gradien: Adam dan L-BFGS [Kingma and Ba, 2014, Byrd et al., 1995]

Seperti yang telah dijelaskan sebelumnya, *loss function* (III.23) akan dioptimasi (diminimalkan) hingga ditemukan sekumpulan beban (*weights*) dan *bias* ($\boldsymbol{\theta} = \{\mathbf{W}^\ell, \mathbf{b}^\ell\}_{1 \leq \ell \leq L}$)

yang ideal (θ^*) untuk memodelkan persamaan panas (III.21). Struktur dari PINN untuk persamaan panas ini ditunjukkan pada Gambar III.5.



Gambar III.5: Ilustrasi dari PINN untuk persamaan panas 2 dimensi dengan sumber panas (III.21)

Karena input dan output dari PINN yang memiliki skala berbeda, maka diperlukan proses *scaling* agar memudahkan proses *training*. Oleh karena itu diperlukan penambahan *scaling layer* pada input dan output seperti pada gambar diatas. Dilakukan proses input *scaling* ($x \times 1, y \times 1, t \times 10^{-4}$) karena domain waktu memiliki nilai dengan orde 4. Kemudian, dilakukan proses output *scaling* $u \times 100$ karena keluaran temperatur memiliki nilai dengan orde 2. Alur *flowchart* proses *training* dari PINN divisualisasikan pada Gambar III.2.

III.4 Spesifikasi *Hardware* yang Digunakan

Untuk melakukan seluruh proses perhitungan dan *training*, digunakan dua jenis *platform*, yaitu Jupyter Notebook dan Google Colab. Jupyter Notebook dijalankan oleh penulis menggunakan Intel Core i5-4200M CPU @ 2.50GHz dengan 2 *core* serta 8 GB RAM. Google Colab dijalankan oleh *server* Google dengan spesifikasi 2 buah Intel(R) Xeon(R) vCPU @ 2.00GHz dengan masing-masing 1 *core*, 13.3 GB RAM, serta fitur *GPU-accelerated notebook*.

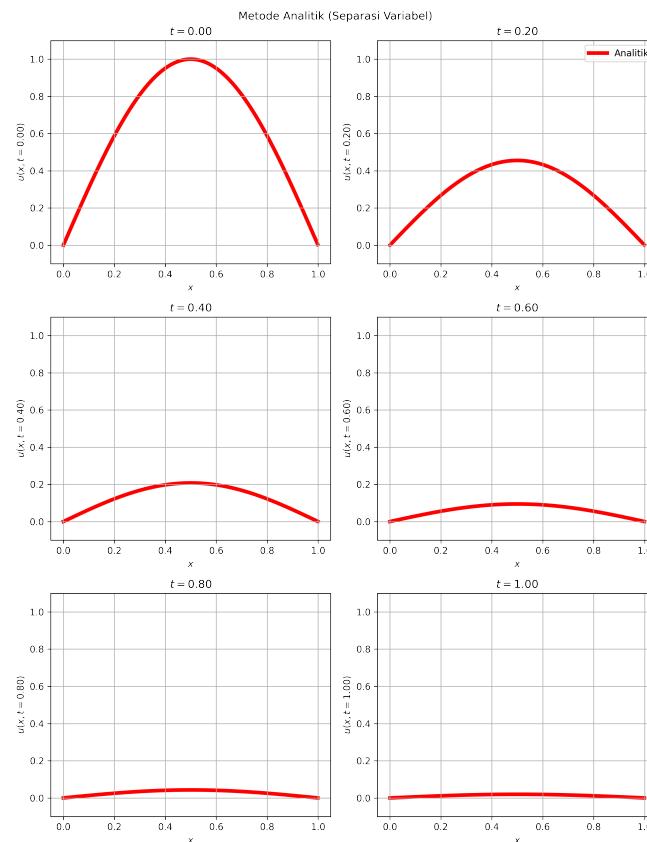
BAB IV

HASIL DAN ANALISIS

IV.1 Simulasi Persebaran Panas 1D

IV.1.1 Penyelesaian Metode Separasi Variabel

Telah diselesaikan persamaan panas (III.1) menggunakan metode separasi variabel dan diperoleh hasil seperti pada persamaan (III.6). Berikut merupakan visualisasi hasil tersebut pada grafik yang merepresentasikan sebaran temperatur $u(x, t)$ terhadap posisi x untuk tiap waktu t .

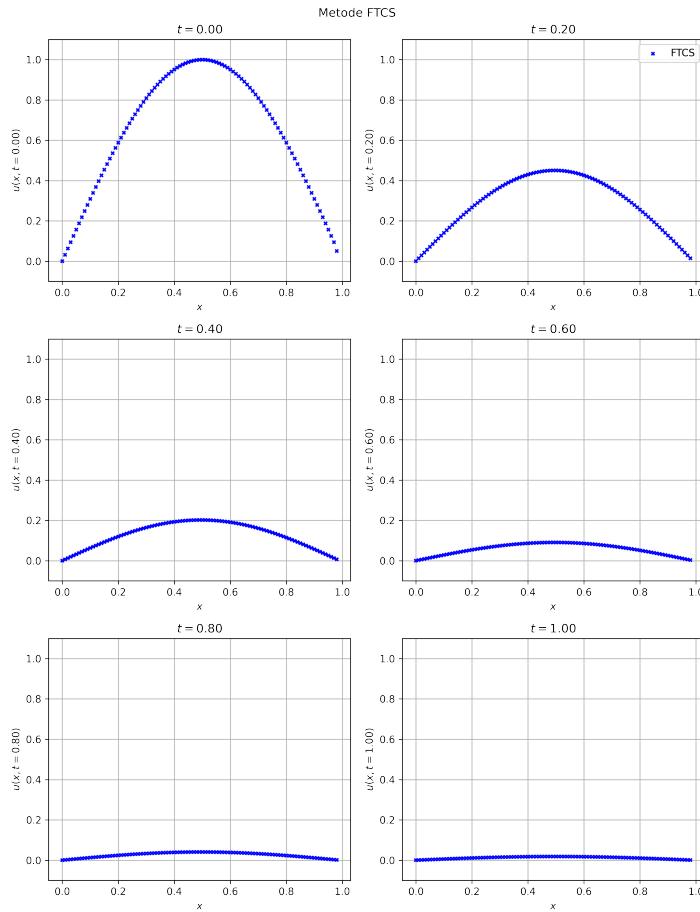


Gambar IV.1: Solusi persamaan panas 1 dimensi (III.1) menggunakan metode analitik (separasi variabel)

Hasil ini digunakan sebagai tolok ukur untuk metode FTCS dan PINN.

IV.1.2 Penyelesaian Metode FTCS

Dilakukan simulasi dengan langkah-langkah seperti pada Gambar III.1. Simulasi berjalan selama 60.52 detik (16.36 detik menggunakan Google Colab). Diperoleh hasil sebagai berikut.

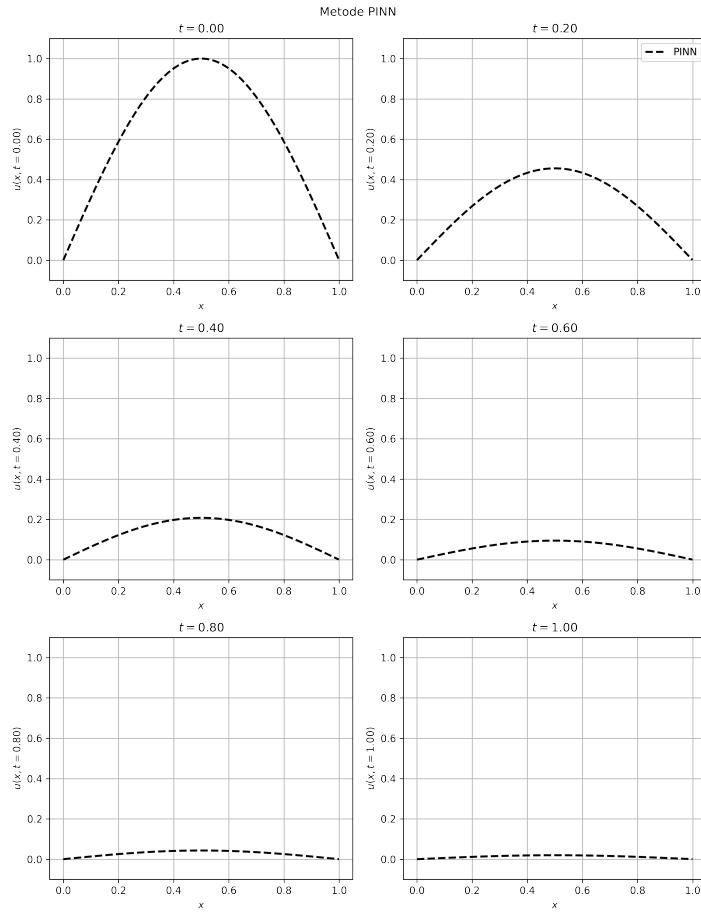


Gambar IV.2: Solusi persamaan panas 1 dimensi (III.1) menggunakan metode FTCS

Salah satu konsep dasar dari metode beda hingga adalah diskritisasi domain ruang dan waktu yang kontinu ke dalam kisi yang diskrit. Sehingga dapat dilihat bahwa pada metode FTCS, hasil yang diperoleh berupa titik-titik pada kurva, bukan kurva yang kontinu. Hasil dari metode ini akan selalu terikat pada kisi yang diskrit.

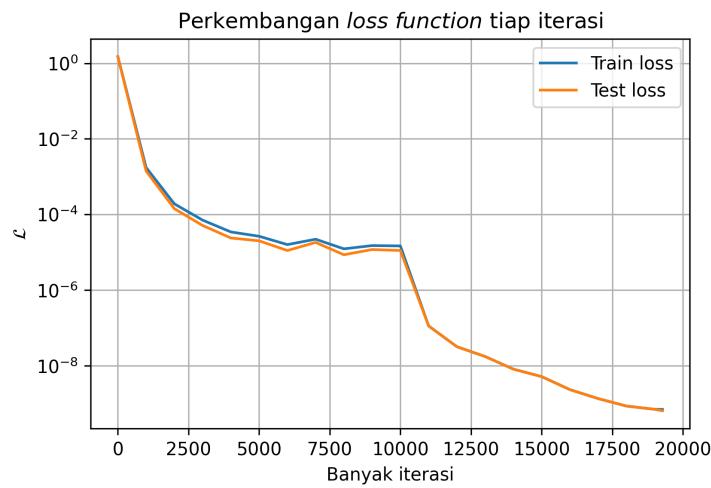
IV.1.3 Penyelesaian Metode PINN

Dilakukan simulasi dengan langkah-langkah seperti pada Gambar III.2 serta Gambar II.3. Digunakan struktur PINN 4 *layer*, dengan jumlah *neuron* per *layer* sebanyak 32. Simulasi berjalan selama 20.48 menit (3.78 menit menggunakan Google Colab). Diperoleh hasil sebagai berikut.



Gambar IV.3: Solusi persamaan panas 1 dimensi (III.1) menggunakan PINN

Loss function (III.9) tiap 1000 iterasi disimpan nilainya, kemudian disajikan pada Gambar IV.4.

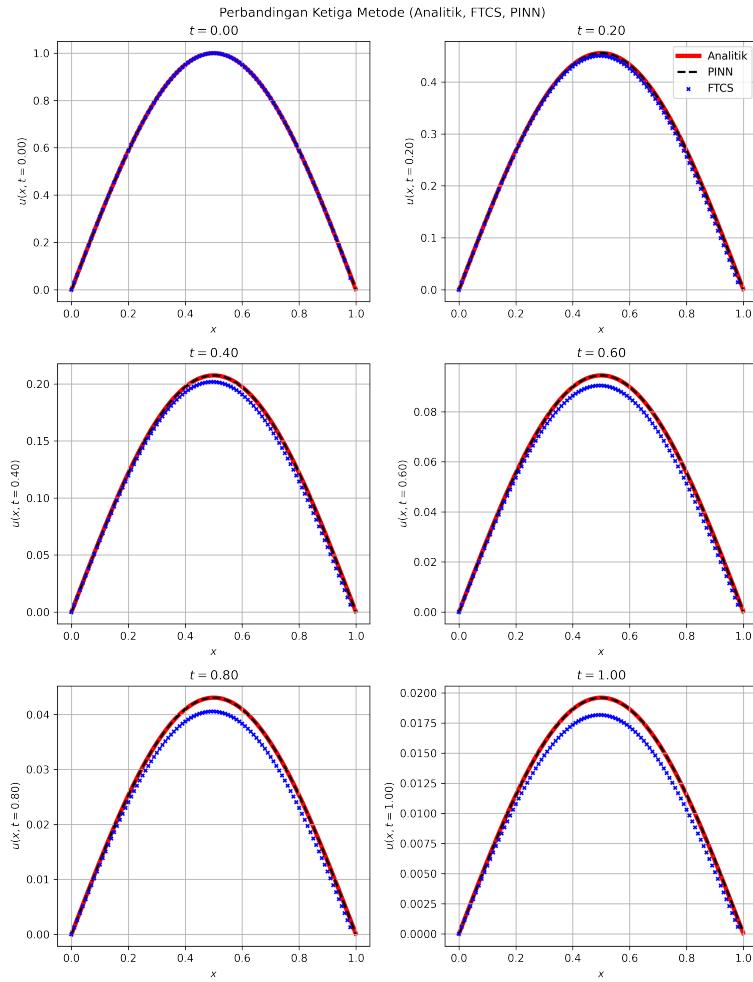


Gambar IV.4: Perkembangan nilai *loss function* (\mathcal{L}) pada PINN untuk persamaan panas 1 dimensi (III.1)

Sehingga dapat diamati perkembangan *loss function* yang turun seiring dengan naiknya iterasi program. Hasil penyelesaian persamaan panas melalui PINN menghasilkan solusi yang kontinu, sehingga dapat dievaluasi pada titik acak.

IV.1.4 Evaluasi Kinerja Metode FTCS dan PINN pada Persamaan Panas 1D

Hasil dari metode analitik, FTCS, dan PINN kemudian dibandingkan secara bersamaan. Berikut merupakan visualisasi hasil ketiga metode pada grafik yang merepresentasikan sebaran temperatur $u(x, t)$ terhadap posisi x untuk tiap waktu t



Gambar IV.5: Perbandingan antara metode analitik, FTCS, dan PINN pada penyelesaian persamaan panas 1 dimensi (III.1)

Kemudian, hasil dievaluasi dengan mencari puncak dari kurva serta selisih kuadrat rata-rata antara nilai estimasi (FTCS dan PINN) dan nilai sebenarnya (Analitik). Digunakan persamaan (III.11) untuk memperoleh nilai *mean-squared error*. Hasilnya disajikan pada Tabel IV.1

Tabel IV.1: Perbandingan tiap metode dalam penyelesaian persamaan panas 1 dimensi (III.1)

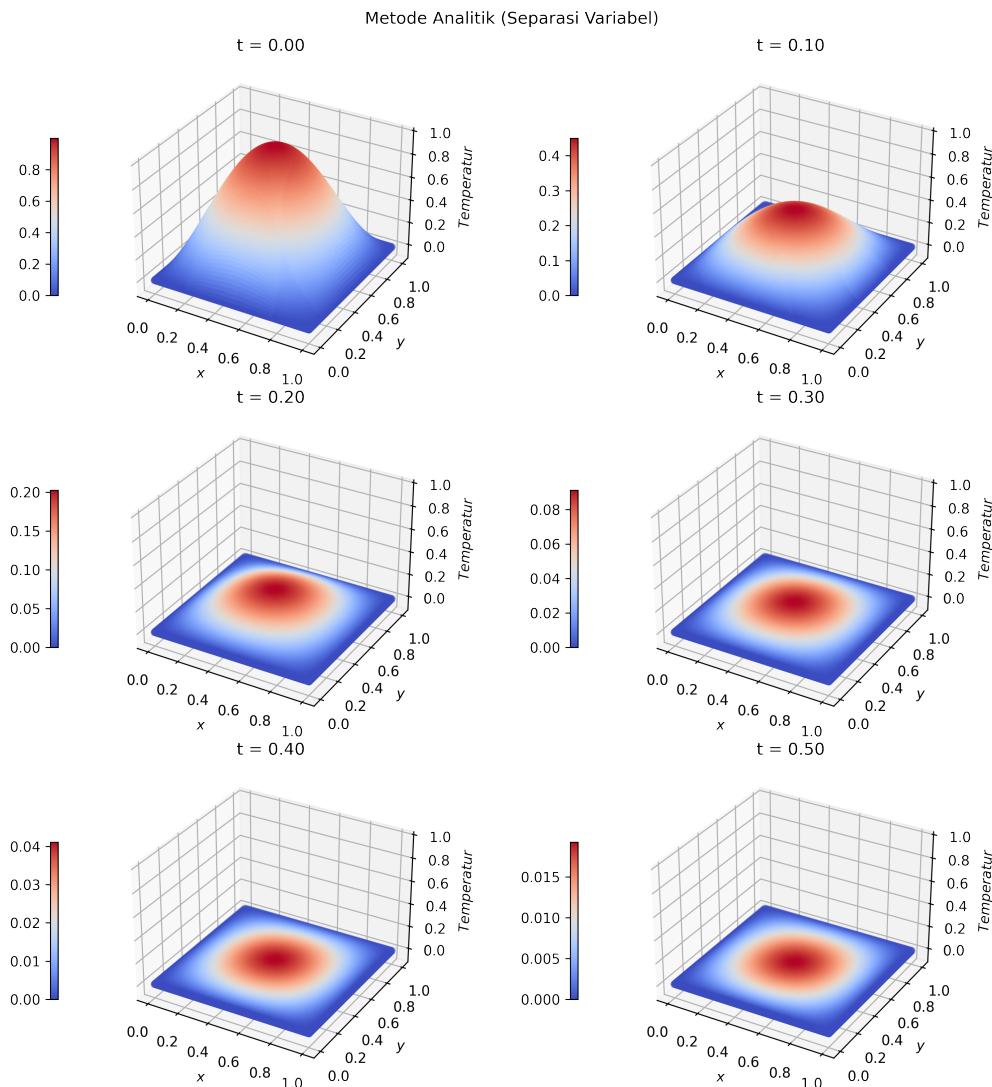
Metode	Puncak	MSE	Waktu Kalkulasi	
			menggunakan Jupyter Notebook	menggunakan Google Colab
Analitik	1.00000			
FTCS	0.99961	2.827×10^{-5}	49.76 detik	16.36 detik
PINN	0.99999	9.834×10^{-12}	20.48 menit	3.78 menit

Tampak bahwa hasil dari PINN dengan konfigurasi seperti pada Tabel A.1 memiliki MSE yang 10^6 kali lebih kecil jika dibandingkan metode FTCS dengan konfigurasi seperti pada Tabel A.2. Dapat disimpulkan bahwa PINN bisa menjadi salah satu alternatif untuk menyelesaikan persamaan panas. Salah satu keunggulan dari PINN adalah solusi yang kontinu karena solusinya yang berbentuk fungsi. Hal ini menjadikan hasil dari PINN berguna untuk permasalahan lebih lanjut yang memerlukan fungsi kontinu. PINN juga tidak terikat oleh kisi (*grid*), sehingga hasil $u(\mathbf{x})$ dapat diperoleh pada titik acak pada domain input. Namun, ketelitian yang dicapai oleh PINN dan hasilnya yang terbebas dari kisi dibayar dengan waktu komputasinya. Tampak sangat jelas bahwa waktu kalkulasi dari FTCS jauh lebih cepat dibanding waktu kalkulasi dari PINN.

IV.2 Simulasi Persebaran Panas 2D

IV.2.1 Penyelesaian Metode Separasi Variabel

Telah diselesaikan persamaan panas (III.12) menggunakan metode separasi variabel dan diperoleh hasil seperti pada persamaan (III.16). Berikut merupakan visualisasi hasil tersebut pada grafik yang merepresentasikan sebaran temperatur $u(x, t)$ terhadap posisi x untuk tiap waktu t .

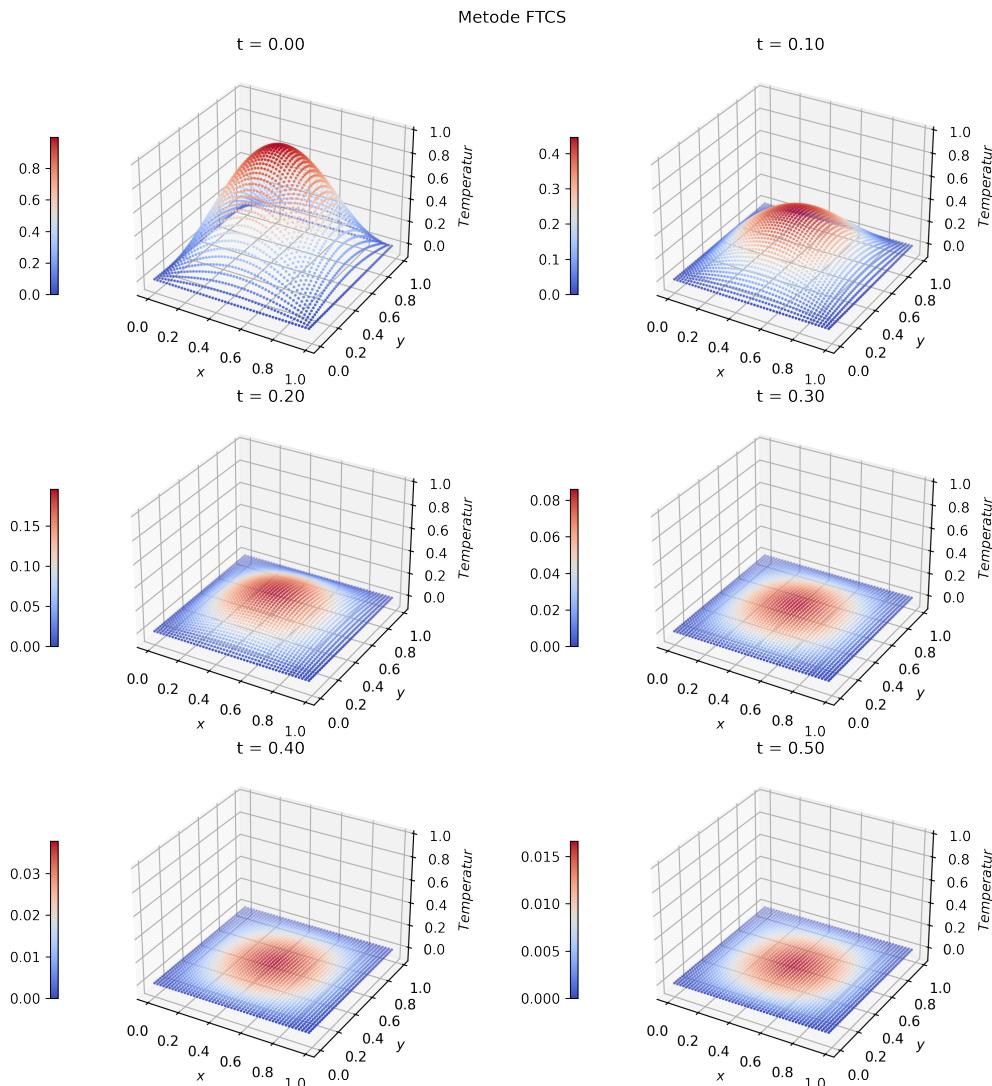


Gambar IV.6: Solusi persamaan panas 2 dimensi (III.12) menggunakan metode analitik (separasi variabel)

Hasil ini digunakan sebagai tolok ukur untuk metode FTCS dan PINN.

IV.2.2 Penyelesaian Metode FTCS

Dilakukan simulasi dengan langkah-langkah seperti pada Gambar III.3. Simulasi berjalan selama 74.22 detik (28.58 detik menggunakan Google Colab). Diperoleh hasil sebagai berikut.



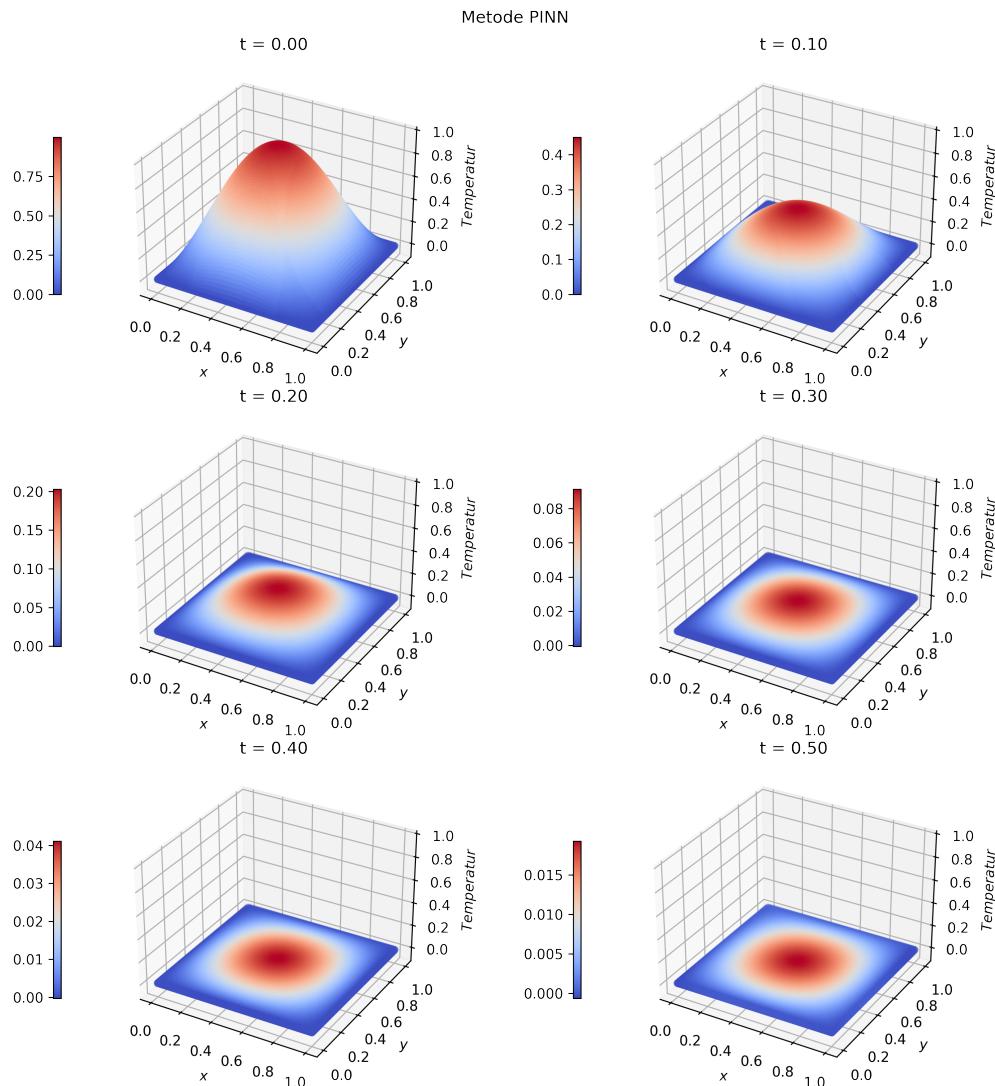
Gambar IV.7: Solusi persamaan panas 2 dimensi (III.12) menggunakan metode FTCS

Salah satu konsep dasar dari metode beda hingga adalah diskritisasi domain ruang dan waktu yang kontinu ke dalam kisi yang diskrit. Sehingga dapat dilihat bahwa pada

metode FTCS, hasil yang diperoleh berupa titik-titik pada kurva, bukan kurva yang kontinu. Hasil dari metode ini akan selalu terikat pada kisi yang diskrit.

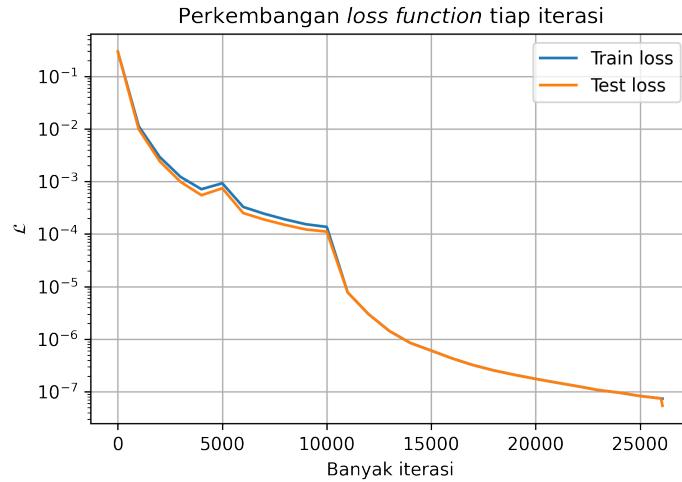
IV.2.3 Penyelesaian Metode PINN

Dilakukan simulasi dengan langkah-langkah seperti pada Gambar III.2 serta Gambar II.3. Digunakan struktur PINN 4 *layer*, dengan jumlah *neuron* per *layer* sebanyak 32. Simulasi berjalan selama 79.56 menit (45.72 menit menggunakan Google Colab). Diperoleh hasil sebagai berikut.



Gambar IV.8: Solusi persamaan panas 2 dimensi (III.12) menggunakan PINN

Loss function (III.20) tiap 1000 iterasi disimpan nilainya, kemudian disajikan pada Gambar IV.9.



Gambar IV.9: Perkembangan nilai *loss function* (\mathcal{L}) pada PINN untuk persamaan panas 2 dimensi (III.12)

Sehingga dapat diamati perkembangan *loss function* yang turun seiring dengan naiknya iterasi program. Hasil penyelesaian persamaan panas melalui PINN menghasilkan solusi yang kontinu, sehingga dapat dievaluasi pada titik acak.

IV.2.4 Evaluasi Kinerja Metode FTCS dan PINN pada Persamaan Panas 2D

Hasil dievaluasi dengan mencari puncak dari kurva serta selisih kuadrat rata-rata antara nilai estimasi (FTCS dan PINN) dan nilai sebenarnya (Analitik). Digunakan persamaan (III.11) untuk memperoleh nilai *mean-squared error*. Hasilnya disajikan pada Tabel IV.2

Tabel IV.2: Perbandingan tiap metode dalam penyelesaian persamaan panas 2 dimensi (III.12)

Metode	Puncak	MSE	Waktu Kalkulasi	
			menggunakan Jupyter Notebook	menggunakan Google Colab
Analitik	1.00000			
FTCS	0.99803	7.889×10^{-5}	74.22 detik	28.58 detik
PINN	0.99968	8.283×10^{-9}	79.56 menit	45.72 menit

Tampak bahwa hasil dari PINN dengan konfigurasi seperti pada Tabel A.1 memiliki MSE yang 10^4 kali lebih kecil jika dibandingkan metode FTCS dengan konfigurasi seperti pada Tabel A.2. Dapat disimpulkan bahwa PINN bisa menjadi salah satu alternatif untuk menyelesaikan persamaan panas. Salah satu keunggulan dari PINN adalah solusi yang kontinu karena solusinya yang berbentuk fungsi. Hal ini menjadikan hasil dari PINN berguna untuk permasalahan lebih lanjut yang memerlukan fungsi kontinu. PINN juga tidak terikat oleh kisi (*grid*), sehingga hasil $u(\mathbf{x})$ dapat diperoleh pada titik acak pada domain input. Namun, ketelitian yang dicapai oleh PINN dan hasilnya yang terbebas dari kisi dibayar dengan waktu komputasinya. Tampak sangat jelas bahwa waktu kalkulasi dari FTCS jauh lebih cepat dibanding waktu kalkulasi dari PINN.

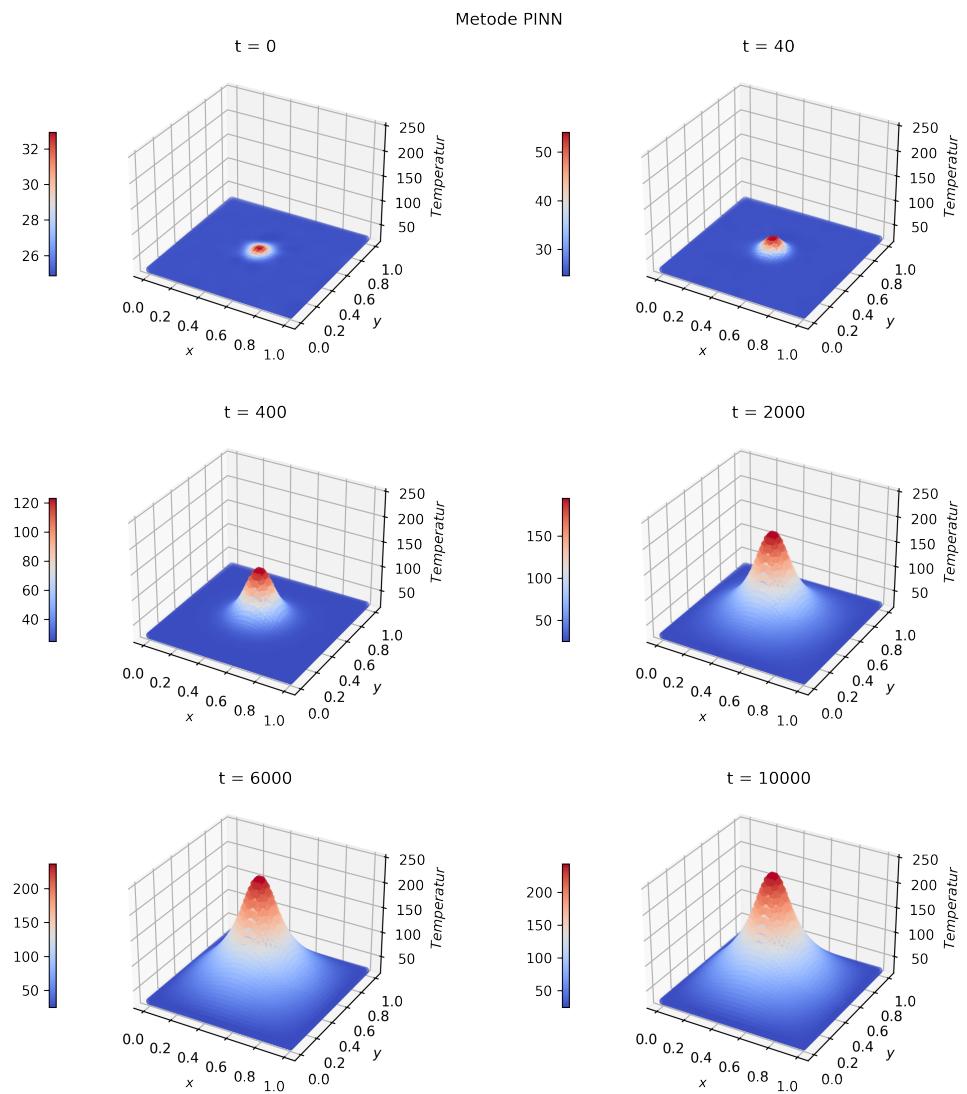
IV.3 Simulasi Persebaran Panas 2D dengan Sumber Panas

IV.3.1 Penyelesaian Metode PINN

Telah diselesaikan persamaan panas (III.21) dengan dilakukannya simulasi melalui langkah-langkah seperti pada Gambar III.2 serta Gambar III.5. Akan disimulasikan dua variasi problem, yaitu sumber panas pada tengah domain (titik (0.5, 0.5)) dan sumber panas pada pinggir domain (titik (0.25, 0.25)).

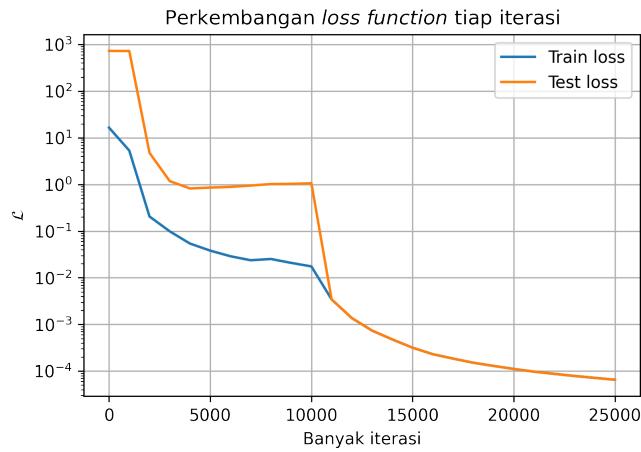
Sumber Panas pada Titik (0.5, 0.5)

Digunakan struktur PINN 4 *layer*, dengan jumlah *neuron* per *layer* sebanyak 32. Simulasi berjalan selama 88.45 menit (9.37 menit menggunakan Google Colab). Diperoleh hasil sebagai berikut.



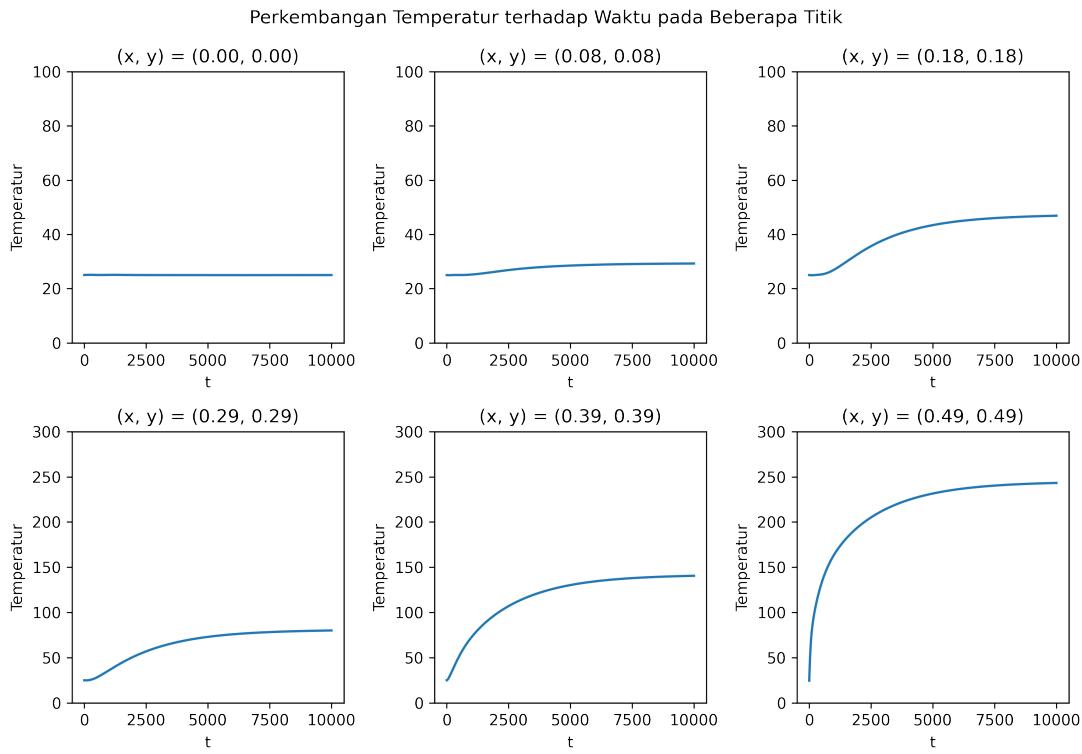
Gambar IV.10: Solusi persamaan panas 2 dimensi dengan sumber panas (III.21) dengan variasi sumber panas di titik $(0.5, 0.5)$ menggunakan PINN

Loss function (III.20) tiap 1000 iterasi disimpan nilainya, kemudian disajikan pada Gambar IV.11.



Gambar IV.11: Perkembangan nilai *loss function* (\mathcal{L}) pada PINN untuk persamaan panas 2 dimensi dengan sumber panas (III.21) dengan variasi sumber panas di titik (0.5, 0.5)

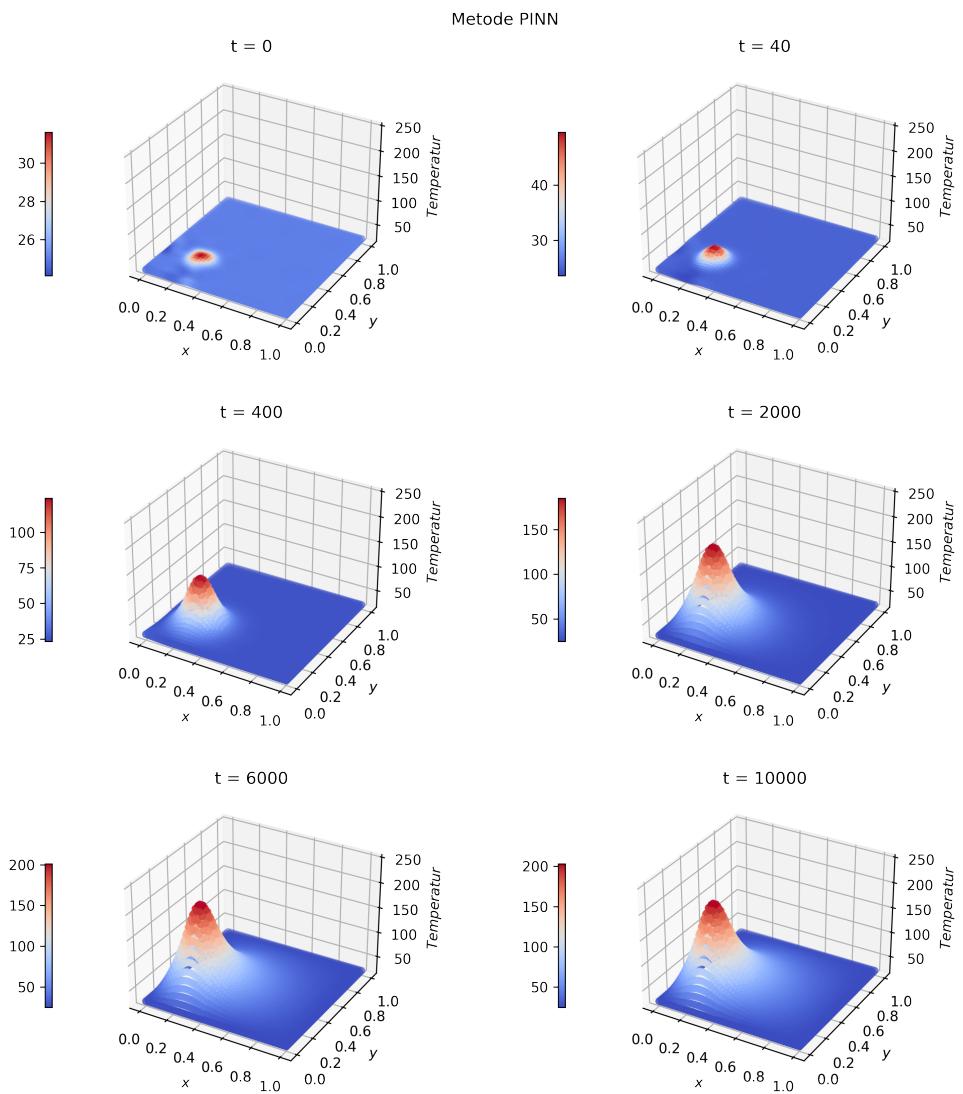
Sehingga dapat diamati perkembangan *loss function* yang turun seiring dengan naiknya iterasi program. Kemudian diamati perkembangan temperatur terhadap waktu pada 6 titik di dalam domain. Hasilnya disajikan pada Gambar IV.12.



Gambar IV.12: Perkembangan temperatur terhadap waktu pada 6 titik di dalam domain solusi persamaan panas 2 dimensi dengan sumber panas (III.21) dengan variasi sumber panas di titik $(0.5, 0.5)$

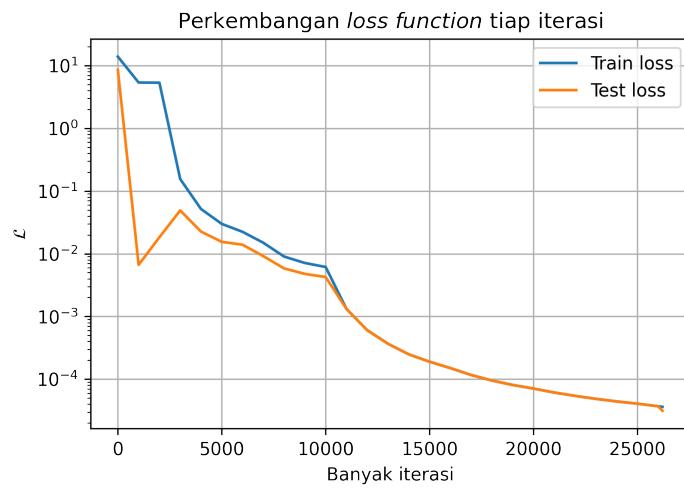
Sumber Panas pada Titik $(0.25, 0.25)$

Digunakan struktur PINN 4 *layer*, dengan jumlah *neuron* per *layer* sebanyak 32. Simulasi berjalan selama 80.85 menit (7.21 menit menggunakan Google Colab). Diperoleh hasil sebagai berikut.



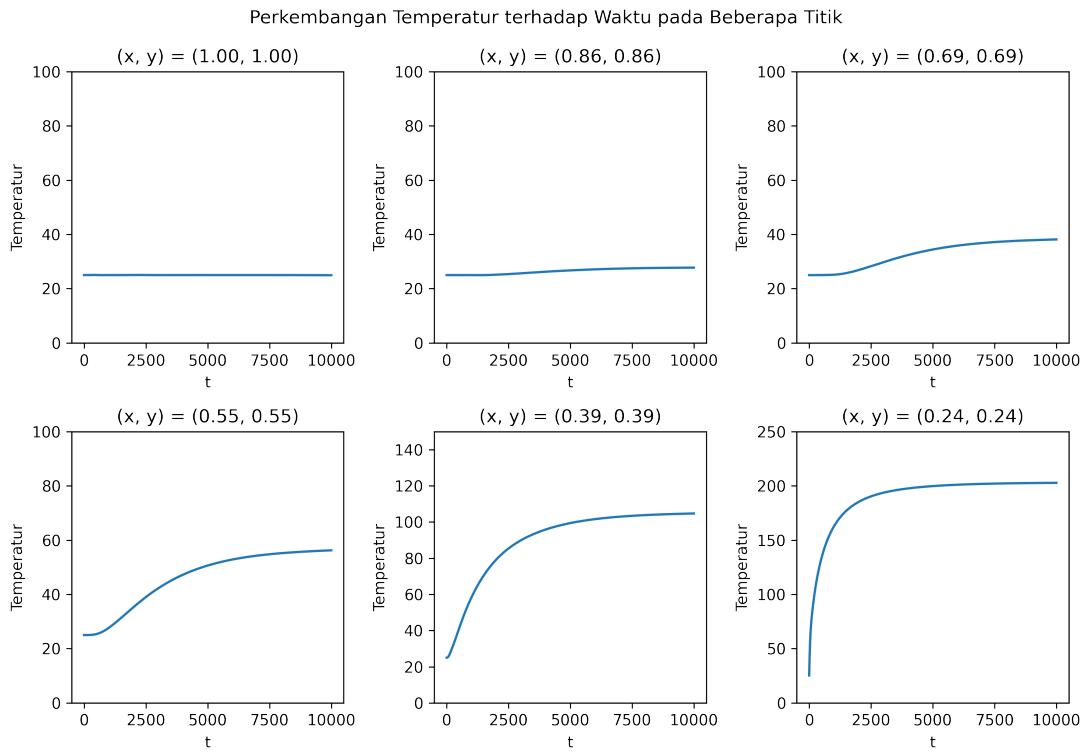
Gambar IV.13: Solusi persamaan panas 2 dimensi dengan sumber panas (III.21) dengan variasi sumber panas di titik $(0.25, 0.25)$ menggunakan PINN

Loss function (III.20) tiap 1000 iterasi disimpan nilainya, kemudian disajikan pada Gambar IV.14.



Gambar IV.14: Perkembangan nilai *loss function* (\mathcal{L}) pada PINN untuk persamaan panas 2 dimensi dengan sumber panas (III.21) dengan variasi sumber panas di titik (0.25, 0.25)

Sehingga dapat diamati perkembangan *loss function* yang turun seiring dengan naiknya iterasi program. Kemudian diamati perkembangan temperatur terhadap waktu pada 6 titik di dalam domain. Hasilnya disajikan pada Gambar IV.15.



Gambar IV.15: Perkembangan temperatur terhadap waktu pada 6 titik di dalam domain solusi persamaan panas 2 dimensi dengan sumber panas (III.21) dengan variasi sumber panas di titik $(0.25, 0.25)$

Setelah kedua variasi disimulasikan, diperoleh kesimpulan sebagai berikut. Temperatur akan mengalami kenaikan dan menuju saturasi pada titik dengan jarak tertentu dari sumber panas. Titik yang lebih dekat dengan sumber panas akan mengalami saturasi pada temperatur yang lebih tinggi.

BAB V

SIMPULAN DAN PELUANG RISET LANJUTAN

V.1 Simpulan

Berdasarkan penelitian yang telah dilakukan, telah diselesaikan persamaan panas satu dan dua dimensi dengan syarat batas Dirichlet dan nilai awal fungsi sinus menggunakan tiga metode: analitik, FTCS, dan PINN. Hasil penyelesaian persamaan panas menggunakan PINN memiliki *mean-squared error* yang kecil jika dibandingkan dengan metode analitik. Dapat disimpulkan bahwa PINN bisa menjadi salah satu alternatif untuk menyelesaikan persamaan diferensial parsial secara umum, serta persamaan panas secara khusus.

Selain itu, solusi yang dihasilkan metode FTCS berupa titik-titik pada kurva, bukan kurva yang kontinu. Sedangkan PINN memberikan solusi yang kontinu serta tidak terikat oleh kisi (*grid*), sehingga menjadikan hasil dari PINN berguna untuk permasalahan lebih lanjut yang memerlukan fungsi kontinu.

Kemudian, telah diselesaikan persamaan panas dua dimensi dengan sumber panas menggunakan *Physics-Informed Neural Network* (PINN). Dipelajari dua variasi keadaan, yaitu sumber panas di tengah domain dan sumber panas di pinggir domain. Setelah kedua variasi disimulasikan, diperoleh kesimpulan bahwa temperatur akan mengalami kenaikan dan menuju saturasi pada titik dengan jarak tertentu dari sumber panas. Titik yang lebih dekat dengan sumber panas akan mengalami saturasi pada temperatur yang lebih tinggi.

V.2 Peluang Riset Lanjutan

Pengembangan lanjutan untuk riset ini dapat dilakukan pada pemaparan lebih detail mengenai parameter-parameter *Physics-Informed Neural Network* (PINN), seperti jumlah *layer*, jumlah *neuron* per *layer*, banyak iterasi, metode optimasi, dan sebagainya. Selanjutnya, dapat dilakukan variasi masing-masing parameter untuk mengamati pengaruhnya terhadap hasil yang diperoleh. Selain itu, diperlukan penyesuaian model agar lebih mendekati hasil eksperimen.

DAFTAR PUSTAKA

- [Badia and Ha-Duong, 2002] Badia, A. E. and Ha-Duong, T. (2002). On an inverse source problem for the heat equation. application to a pollution detection problem. *Journal of Inverse and Ill-posed Problems*, 10(6):585–599.
- [Baydin et al., 2015] Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. (2015). Automatic differentiation in machine learning: a survey.
- [Bird et al., 2006] Bird, R., Stewart, W., and Lightfoot, E. (2006). *Transport Phenomena*. Number v. 1 in Transport Phenomena. Wiley.
- [Boas, 2005] Boas, M. (2005). *Mathematical Methods in the Physical Sciences*. Wiley.
- [Byrd et al., 1995] Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16:1190–1208.
- [Hoffman and Frankel, 2001] Hoffman, J. and Frankel, S. (2001). *Numerical Methods for Engineers and Scientists, Second Edition*,. Taylor & Francis.
- [Karima and Magdalena, 2021] Karima, N. and Magdalena, I. (2021). The 2nd, 4th, and 6th-order finite difference schemes for pollutant transport equation. In *Proceedings of the 1st International Conference on Mathematics and Mathematics Education (ICMMEd 2020)*, pages 29–33. Atlantis Press.
- [Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.
- [Lemmon et al., 2000] Lemmon, E., Jacobsen, R., Penoncello, S., and Friend, D. (2000). Thermodynamic properties of air and mixtures of nitrogen, argon, and oxygen from 60 to 2000 k at pressures to 2000 mpa.
- [Lu et al., 2021] Lu, L., Meng, X., Mao, Z., and Karniadakis, G. E. (2021). Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228.

- [Maitsa et al., 2021] Maitsa, T., Hafiyyan, Q., Adityawan, M., Magdalena, I., Adi Kuntoro, A., and Kardhana, H. (2021). Development of a 2d numerical model for pollutant transport using ftcs scheme and numerical filter. *Makara Journal of Technology*, 25:119.
- [Nielsen, 2015] Nielsen, M. (2015). *Neural Networks and Deep Learning*. Determination Press.
- [Raissi et al., 2017] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017). Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations.
- [Verma, 2000] Verma, A. (2000). An introduction to automatic differentiation. *Current Science*, 78(7):804–807.
- [Widder, 1976] Widder, D. (1976). *The Heat Equation*. Pure and Applied Mathematics. Elsevier Science.

LAMPIRAN A

KONFIGURASI PINN DAN FTCS

Tabel A.1: Konfigurasi PINN yang digunakan pada persamaan panas 1D dan 2D

Parameter PINN	1D	2D
Struktur PINN	4×32	4×32
Loss Function	(III.9)	(III.20)
Jumlah titik training ($\mathcal{T}_f, \mathcal{T}_b$)	(2540, 240)	(5000, 600)
Fungsi aktivasi	tanh	tanh
Metode optimasi	Adam & L-BFGS	Adam & L-BFGS

Tabel A.2: Konfigurasi FTCS yang digunakan pada persamaan panas 1D dan 2D

Parameter metode FTCS	1D	2D
Δx	0.01	0.02
Δy	-	0.02
Δt	10^{-4}	2.5×10^{-4}
Jumlah titik perhitungan	10^6	5×10^6

LAMPIRAN B

VARIASI STRUKTUR PINN

Tabel B.1: Perbandingan hasil variasi PINN pada penyelesaian persamaan panas 1 dimensi (III.1)

Struktur PINN		Puncak	MSE	Waktu Kalkulasi (menggunakan Google Colab)
Jumlah <i>layer</i>	Jumlah neuron per <i>layer</i>			
2	16	0.99999	1.706×10^{-11}	2.97 menit
2	32	0.99999	1.276×10^{-11}	3.38 menit
4	32	0.99999	9.834×10^{-12}	3.78 menit
4	64	0.99999	3.283×10^{-11}	5.47 menit
6	64	0.99999	1.990×10^{-11}	11.91 menit

LAMPIRAN C

KODE PROGRAM

Untuk kode program dalam bentuk .ipynb, kunjungi:

<https://github.com/michaelalfarino/PINNheatEq>

C.1 PROGRAM PERSAMAAN PANAS 1D

```
1 # # PERSAMAAN PANAS 1 DIMENSI
2 import deepxde as dde
3 from deepxde.backend import tf
4 import numpy as np
5
6 # library untuk membuat animasi:
7 from IPython.display import HTML
8 import matplotlib.pyplot as plt
9 from matplotlib.animation import FuncAnimation
10 from matplotlib.animation import PillowWriter
11
12 # pengaturan
13 plt.rcParams['animation.ffmpeg_path'] = r'C:\FFmpeg\bin\ffmpeg.exe'
14 dde.config.set_default_float("float64")
15 dde.optimizers.config.set_LBFGS_options(maxiter=15000)
16
17
18
19 # # PINN
20
21 # solusi analitik persamaan panas 1d
22 def heat_eq_exact_solution(x, t):
23     return np.exp(-(np.pi**2 * a * t)) * np.sin(np.pi * x)
24
25 # fungsi untuk menghitung solusi analitik
26 def gen_exact_solution():
27     # jumlah titik tiap dimensi:
28     x_dim, t_dim = (1000, 1000)
29
```

```

30 # batas dari x dan t:
31 x_min, t_min = (0, 0.0)
32 x_max, t_max = (L, maxtime)
33
34 # titik titik yg akan dicari solusinya:
35 t = np.linspace(t_min, t_max, num=t_dim).reshape(t_dim, 1)
36 x = np.linspace(x_min, x_max, num=x_dim).reshape(x_dim, 1)
37 usol = np.zeros((x_dim, t_dim)).reshape(x_dim, t_dim)
38
39 # peroleh solusi tiap titik:
40 for i in range(x_dim):
41     for j in range(t_dim):
42         usol[i][j] = heat_eq_exact_solution(x[i], t[j])
43
44 # Save:
45 np.savez("heat_eq_data", x=x, t=t, usol=usol)
46
47 # fungsi untuk membaca data analitik
48 def gentestdata():
49     # Load data:
50     data = np.load("heat_eq_data.npz")
51     t, x, exact = data["t"], data["x"], data["usol"].T
52     # ratakan data:
53     xx, tt = np.meshgrid(x, t)
54     X = np.vstack((np.ravel(xx), np.ravel(tt))).T
55     y = exact.flatten()[:, None]
56     return X, y
57
58 # Problem parameters:
59 a = 0.4 # Thermal diffusivity
60 L = 1 # panjang interval
61 maxtime = 1
62
63 # peroleh solusi analitik
64 gen_exact_solution()
65
66 def pde(x, y):
67     # persamaan diferensial parsial: persamaan panas 1d
68     dy_t = dde.grad.jacobian(y, x, i=0, j=1)
69     dy_xx = dde.grad.hessian(y, x, i=0, j=0)
70     return dy_t - a * dy_xx

```

```

71
72 # geometri domain permasalahan
73 geom = dde.geometry.Interval(0, L)
74 timedomain = dde.geometry.TimeDomain(0, maxtime)
75 geomtime = dde.geometry.GeometryXTime(geom, timedomain)
76
77 # Initial and boundary conditions:
78 bc = dde.icbc.DirichletBC(geomtime, lambda x: 0, lambda _,
    on_boundary: on_boundary)
79 ic = dde.icbc.IC(
80     geomtime,
81     lambda x: np.sin(np.pi * x[:, 0:1]),
82     lambda _, on_initial: on_initial,
83 )
84
85 # definisi PDP dan konfigurasi neural network
86 data = dde.data.TimePDE(
87     geomtime,
88     pde,
89     [bc, ic],
90     num_domain=2540,
91     num_boundary=80,
92     num_initial=160,
93     num_test=2540,
94 )
95 net = dde.nn.FNN([2] + [32] * 4 + [1], "tanh", "Glorot normal")
96 model = dde.Model(data, net)
97
98 # Bangun dan latih model dengan 2 metode optimasi
99 model.compile("adam", lr=1e-3, loss_weights=[1, 1, 1])
100 model.train(iterations=10000)
101 model.compile("L-BFGS", loss_weights=[1, 1, 1])
102 losshistory, train_state = model.train()
103
104 # Plot/print hasil
105 dde.saveplot(losshistory, train_state, issave=True, isplot=True)
106 X, y_true = gentestdata()
107 y_PINN = model.predict(X)
108 print("Mean squared error:", dde.metrics.mean_squared_error(y_true,
    y_PINN))
109 np.savetxt("test.dat", np.hstack((X, y_true, y_PINN)))

```

```

110
111
112
113 # Numeric FTCS
114
115 # library untuk menghitung waktu running FTCS
116 import time
117 start = time.time()
118
119 # deklarasi ukuran diferensial
120 dt = 1e-4
121 dx = 0.01
122
123 # inisialisasi domain
124 xs = np.arange(0, L, dx)
125 ts = np.arange(0, maxtime, dt)
126 u = np.sin(np.pi*xs)      #initial condition
127
128 u[0] = 0      #boundary condition
129 u[-1] = 0
130
131 y_num = np.empty(3)
132
133 for n in ts:  #iterasi melalui waktu
134     un = u.copy()
135     u[1:-1] = un[1:-1] + a * dt / dx**2 * (un[2:] - 2 * un[1:-1] + un
136     [0:-2])
137     u[0] = 0
138     u[-1] = 0
139     temp = np.hstack((xs.reshape(-1,1), np.full(len(xs), n).reshape
140     (-1,1), u.reshape(-1,1)))
141     y_num = np.vstack((y_num, temp))
142
143 y_num = np.delete(y_num, 0, 0)
144 end = time.time()
145
146
147
148 # plot hasil dari persamaan panas

```

```

149
150 fig, axs = plt.subplots(3, 2, figsize=(10,13))
151 figTime = 0
152 fig.suptitle("Perbandingan Ketiga Metode")
153 for ax in axs.flat:
154     ax.grid()
155     ax.set_xlabel('$x$', ylabel='$u(x, t = {:.2f})$'.format(time[figTime]))
156     ax.set_ylim([-1, 1.1])
157     markersize = [10 for i in range(len(y_num[np.where(np.isclose(y_num[:, 1], time[figTime]))[0][0]:np.where(np.isclose(y_num[:, 1], time[figTime]))[0][-1], 0]]))]
158     ax.plot(result[np.where(result[:, 1] == time[figTime])[0][0]:np.where(result[:, 1] == time[figTime])[0][-1], 0], result[np.where(result[:, 1] == time[figTime])[0][0]:np.where(result[:, 1] == time[figTime])[0][-1], 2], 'r-', linewidth=4, label = 'Analitik', zorder = 0)
159     ax.plot(result[np.where(result[:, 1] == time[figTime])[0][0]:np.where(result[:, 1] == time[figTime])[0][-1], 0], result[np.where(result[:, 1] == time[figTime])[0][0]:np.where(result[:, 1] == time[figTime])[0][-1], 3], 'k--', linewidth=2, label = 'PINN', zorder = 5)
160     ax.scatter(y_num[np.where(np.isclose(y_num[:, 1], time[figTime]))[0][0]:np.where(np.isclose(y_num[:, 1], time[figTime]))[0][-1], 0], y_num[np.where(np.isclose(y_num[:, 1], time[figTime]))[0][0]:np.where(np.isclose(y_num[:, 1], time[figTime]))[0][-1], 2], markersize, marker='x', c='b', label = 'FTCS', zorder = 10)
161     ax.set_title("$t = {:.2f}$".format(time[figTime]))
162     figTime += 199
163
164 axs[0,1].legend()
165
166 plt.tight_layout()
167 plt.savefig("result.png", dpi=300)
168
169
170
171 # Animasi hasil
172
173 result = np.hstack((X, y_true, y_PINN))
174 time = np.unique(result[:, 1])

```

```

175 time2 = np.unique(y_num[:, 1])
176
177 fig, ax = plt.subplots(figsize=(7,5))
178
179 def update(frame):
180     ax.clear()
181     ax.set_title(" ")
182     ax.set_ylim([-1, 1.1])
183     ax.set_ylabel("Temperatur")
184     ax.set_xlabel("x")
185     ax.grid()
186
187     a = np.where(result[:, 1] == time[frame])[0][0]
188     b = np.where(result[:, 1] == time[frame])[0][-1]
189     c = np.where(np.isclose(y_num[:, 1], time[frame], atol=5e-4))
190     [0][0]
191     d = np.where(np.isclose(y_num[:, 1], time[frame], atol=5e-4))
192     [0][-1]
193     ax.plot(result[a:b, 0], result[a:b, 2], 'r-', linewidth=4, label
194     = 'Analitik', zorder = 0)
195     ax.plot(result[a:b, 0], result[a:b, 3], 'k--', linewidth=2, label
196     = 'PINN', zorder = 5)
197     ax.scatter(y_num[c:d, 0], y_num[c:d, 2], marker='x', label =
198     'FTCS', zorder = 10)
199     ax.set_title("Persebaran panas t = {:.2f}".format(result[a, 1]))
200     ax.legend()
201
202 ani = FuncAnimation(fig, update, frames=int(len(time)))
203
204 # Save animasi sebagai gif
205 ani.save("heat_eq_1d.gif", dpi=200, fps=30)
206
207
208 # mengukur MSE tiap metode
209 from sklearn.metrics import mean_squared_error as MSE
210 print('Method' + "Peak" + "MSE" + ' ')
211
212 print("\nAnalytic", heat_eq_exact_solution(result[:, 0], result
213 [:, 1]).max(),
214      "\nFTCS", y_num[:, 2].max(), " ", MSE(y_num[:, 2],
215      heat_eq_exact_solution(y_num[:, 0], y_num[:, 1])), )

```

```

207     "\nPINN           ", result[:, 3].max(),"   ", MSE(result[:, 3],
208     heat_eq_exact_solution(result[:, 0], result[:, 1]))
209
210
211
212 # simpan plot loss history
213 loss_train = np.sum(losshistory.loss_train, axis=1)
214 loss_test = np.sum(losshistory.loss_test, axis=1)
215
216 plt.figure(figsize=(6,4))
217 plt.grid()
218 plt.title("Perkembangan $loss$ $function$ tiap iterasi")
219 plt.semilogy(losshistory.steps, loss_train, label="Train loss")
220 plt.semilogy(losshistory.steps, loss_test, label="Test loss")
221 for i in range(len(losshistory.metrics_test[0])):
222     plt.semilogy(
223         loss_history.steps,
224         np.array(loss_history.metrics_test)[:, i],
225         label="Test metric",
226     )
227 plt.xlabel("Banyak iterasi")
228 plt.ylabel("$\mathcal{L}$")
229 plt.legend()
230 plt.savefig('loss.png', dpi=300)

```

C.2 PROGRAM PERSAMAAN PANAS 2D

```
1 # # PERSAMAAN PANAS 2 DIMENSI
2 import deepxde as dde
3 import numpy as np
4 from deepxde.backend import tf
5
6 # library untuk membuat animasi:
7 from IPython.display import HTML
8 import matplotlib.pyplot as plt
9 from matplotlib.animation import FuncAnimation
10 from matplotlib.animation import PillowWriter
11 from matplotlib import cm
12
13 # pengaturan
14 plt.rcParams['animation.ffmpeg_path'] = r'C:\FFmpeg\bin\ffmpeg.exe'
15 dde.config.set_default_float("float64")
16 dde.optimizers.config.set_LBFGS_options(maxiter=15000)
17
18 # # PINN
19
20 # solusi analitik persamaan panas 2d
21 def heat_eq_exact_solution(x, y, t):
22     return np.exp(-(2 * np.pi**2 * a * t)) * np.sin(np.pi * x) * np.
23     sin(np.pi * y)
24
25 # fungsi untuk menghitung solusi analitik
26 def gen_exact_solution():
27     # jumlah titik tiap dimensi:
28     x_dim, y_dim, t_dim = (100, 100, 100)
29
30     # batas dari x dan t:
31     x_min, y_min, t_min = (0, 0, 0.0)
32     x_max, y_max, t_max = (L, L, maxtime)
33
34     # titik titik yg akan dicari solusinya:
35     t = np.linspace(t_min, t_max, num=t_dim).reshape(t_dim, 1)
36     x = np.linspace(x_min, x_max, num=x_dim).reshape(x_dim, 1)
37     y = np.linspace(y_min, y_max, num=y_dim).reshape(y_dim, 1)
38     usol = np.zeros((x_dim, y_dim, t_dim)).reshape(x_dim, y_dim,
t_dim)
```

```

38
39     # peroleh solusi tiap titik:
40     for i in range(x_dim):
41         for j in range(y_dim):
42             for k in range(t_dim):
43                 usol[i][j][k] = heat_eq_exact_solution(x[i], y[j], t[k])
44
45     # Save:
46     np.savez("heat_eq_data", x=x, y=y, t=t, usol=usol)
47
48 # fungsi untuk membaca data analitik
49 def gentestdata():
50     # Load data:
51     data = np.load("heat_eq_data.npz")
52     t, x, y, exact = data["t"], data["x"], data["y"], data["usol"].T
53     # ratakan data:
54     yy, tt, xx = np.meshgrid(y, t, x)
55     X = np.vstack((np.ravel(xx), np.ravel(yy), np.ravel(tt))).T
56     Y = exact.flatten()[:, None]
57     return X, Y
58
59 # Problem parameters:
60 a = 0.4 # Thermal diffusivity
61 L = 1 # panjang interval
62 maxtime = .5
63
64 # peroleh solusi analitik
65 gen_exact_solution()
66
67 def pde(x, u):
68     # persamaan diferensial parsial: persamaan panas 2d
69     du_xx = dde.grad.hessian(u, x, i=0, j=0)
70     du_yy = dde.grad.hessian(u, x, i=1, j=1)
71     du_t = dde.grad.jacobian(u, x, i=0, j=2)
72     return (du_t - (a * (du_xx + du_yy)))
73
74 # geometri domain permasalahan
75 geom = dde.geometry.Rectangle((0,0), (L,L))
76 timedomain = dde.geometry.TimeDomain(0, maxtime)
77 geomtime = dde.geometry.GeometryXTime(geom, timedomain)

```

```

78
79 # Initial and boundary conditions:
80 bc = dde.icbc.DirichletBC(geomtime, lambda x: 0, lambda _,
81     on_boundary: on_boundary)
82 ic = dde.icbc.IC(
83     geomtime,
84     lambda x: np.sin(np.pi * x[:, 0:1])*np.sin(np.pi * x[:, 1:2]),
85     lambda _, on_initial: on_initial,
86     )
87
88 # definisi PDP dan konfigurasi neural network
89 data = dde.data.TimePDE(
90     geomtime,
91     pde,
92     [bc, ic],
93     num_domain=5000,
94     num_boundary=200,
95     num_initial=400,
96     num_test=5000,
97 )
98 net = dde.nn.FNN([3] + [32] * 4 + [1], "tanh", "Glorot normal")
99 model = dde.Model(data, net)
100
101 # Bangun dan latih model dengan 2 metode optimasi
102 model.compile("adam", lr=1e-3, loss_weights=[1, 1, 1])
103 model.train(iterations=10000)
104 losshistory, train_state = model.train()
105
106 # Plot/print hasil
107 dde.saveplot(losshistory, train_state, issave=True, isplot=True)
108 X, y_true = gentestdata()
109 y_PINN = model.predict(X)
110 print("Mean squared error:", dde.metrics.mean_squared_error(y_true,
111     y_PINN))
112 np.savetxt("test.dat", np.hstack((X, y_true, y_PINN)))
113
114 # Plot hasil yang diperoleh
115 fig = plt.figure(figsize=(11,11))
116 plt.suptitle("Metode PINN")

```

```

117 ax = fig.add_subplot(3, 2, 1, projection='3d')
118 surf = ax.scatter(X[0:10000, 0], X[0:10000, 1], y_PINN[0:10000], c=
119     y_PINN[0:10000], cmap=cm.coolwarm)
120 ax.set_title("t = {:.2f}".format(X[0, 2]))
121 ax.set_zlim(-0.1, 1)
122 ax.set_xlabel('$x$')
123 ax.set_ylabel('$y$')
124 ax.set_zlabel('$Temperatur$')
125 fig.colorbar(surf, shrink=.5, location='left')
126
126 ax = fig.add_subplot(3, 2, 2, projection='3d')
127 surf = ax.scatter(X[200000:210000, 0], X[200000:210000, 1], y_PINN
128     [200000:210000], c=y_PINN[200000:210000], cmap=cm.coolwarm)
129 ax.set_title("t = {:.2f}".format(X[200000, 2]))
130 ax.set_zlim(-0.1, 1)
131 ax.set_xlabel('$x$')
132 ax.set_ylabel('$y$')
133 ax.set_zlabel('$Temperatur$')
134 fig.colorbar(surf, shrink=.5, location='left')
135
135 ax = fig.add_subplot(3, 2, 3, projection='3d')
136 surf = ax.scatter(X[400000:410000, 0], X[400000:410000, 1], y_PINN
137     [400000:410000], c=y_PINN[400000:410000], cmap=cm.coolwarm)
138 ax.set_title("t = {:.2f}".format(X[400000, 2]))
139 ax.set_zlim(-0.1, 1)
140 ax.set_xlabel('$x$')
141 ax.set_ylabel('$y$')
142 ax.set_zlabel('$Temperatur$')
143 fig.colorbar(surf, shrink=.5, location='left')
144
144 ax = fig.add_subplot(3, 2, 4, projection='3d')
145 surf = ax.scatter(X[600000:610000, 0], X[600000:610000, 1], y_PINN
146     [600000:610000], c=y_PINN[600000:610000], cmap=cm.coolwarm)
147 ax.set_title("t = {:.2f}".format(X[600000, 2]))
148 ax.set_zlim(-0.1, 1)
149 ax.set_xlabel('$x$')
150 ax.set_ylabel('$y$')
151 ax.set_zlabel('$Temperatur$')
152 fig.colorbar(surf, shrink=.5, location='left')
153
153 ax = fig.add_subplot(3, 2, 5, projection='3d')

```

```

154 surf = ax.scatter(X[800000:810000, 0], X[800000:810000, 1], y_PINN
155 [800000:810000], c=y_PINN[800000:810000], cmap=cm.coolwarm)
156 ax.set_title("t = {:.2f}".format(X[800000, 2]))
157 ax.set_zlim(-0.1, 1)
158 ax.set_xlabel('$x$')
159 ax.set_ylabel('$y$')
160 ax.set_zlabel('$Temperatur$')
161 fig.colorbar(surf, shrink=.5, location='left')
162
162 ax = fig.add_subplot(3, 2, 6, projection='3d')
163 surf = ax.scatter(X[990000:1000000, 0], X[990000:1000000, 1], y_PINN
164 [990000:1000000], c=y_PINN[990000:1000000], cmap=cm.coolwarm)
165 ax.set_title("t = {:.2f}".format(X[990000, 2]))
166 ax.set_zlim(-0.1, 1)
167 ax.set_xlabel('$x$')
168 ax.set_ylabel('$y$')
169 ax.set_zlabel('$Temperatur$')
170 fig.colorbar(surf, shrink=.5, location='left')
171
171 plt.tight_layout()
172 plt.savefig("PINN2d.png", dpi=300)
173
174
175
176 # plot hasil metode analitik
177 fig = plt.figure(figsize=(11,11))
178 plt.suptitle("Metode Analitik (Separasi Variabel)")
179
180 ax = fig.add_subplot(3, 2, 1, projection='3d')
181 surf = ax.scatter(X[0:10000, 0], X[0:10000, 1], y_true[0:10000], c=
182 y_true[0:10000], cmap=cm.coolwarm)
183 ax.set_title("t = {:.2f}".format(X[0, 2]))
184 ax.set_zlim(-0.1, 1)
185 ax.set_xlabel('$x$')
186 ax.set_ylabel('$y$')
187 ax.set_zlabel('$Temperatur$')
188 fig.colorbar(surf, shrink=.5, location='left')
189
189 ax = fig.add_subplot(3, 2, 2, projection='3d')
190 surf = ax.scatter(X[200000:210000, 0], X[200000:210000, 1], y_true
191 [200000:210000], c=y_true[200000:210000], cmap=cm.coolwarm)

```

```

191 ax.set_title("t = {:.2f}".format(X[200000, 2]))
192 ax.set_zlim(-0.1, 1)
193 ax.set_xlabel('$x$')
194 ax.set_ylabel('$y$')
195 ax.set_zlabel('$Temperatur$')
196 fig.colorbar(surf, shrink=.5, location='left')
197
198 ax = fig.add_subplot(3, 2, 3, projection='3d')
199 surf = ax.scatter(X[400000:410000, 0], X[400000:410000, 1], y_true
200 [400000:410000], c=y_true[400000:410000], cmap=cm.coolwarm)
201 ax.set_title("t = {:.2f}".format(X[400000, 2]))
202 ax.set_zlim(-0.1, 1)
203 ax.set_xlabel('$x$')
204 ax.set_ylabel('$y$')
205 ax.set_zlabel('$Temperatur$')
206 fig.colorbar(surf, shrink=.5, location='left')
207
208 ax = fig.add_subplot(3, 2, 4, projection='3d')
209 surf = ax.scatter(X[600000:610000, 0], X[600000:610000, 1], y_true
210 [600000:610000], c=y_true[600000:610000], cmap=cm.coolwarm)
211 ax.set_title("t = {:.2f}".format(X[600000, 2]))
212 ax.set_zlim(-0.1, 1)
213 ax.set_xlabel('$x$')
214 ax.set_ylabel('$y$')
215 ax.set_zlabel('$Temperatur$')
216 fig.colorbar(surf, shrink=.5, location='left')
217
218 ax = fig.add_subplot(3, 2, 5, projection='3d')
219 surf = ax.scatter(X[800000:810000, 0], X[800000:810000, 1], y_true
220 [800000:810000], c=y_true[800000:810000], cmap=cm.coolwarm)
221 ax.set_title("t = {:.2f}".format(X[800000, 2]))
222 ax.set_zlim(-0.1, 1)
223 ax.set_xlabel('$x$')
224 ax.set_ylabel('$y$')
225 ax.set_zlabel('$Temperatur$')
226 fig.colorbar(surf, shrink=.5, location='left')
227
228 ax = fig.add_subplot(3, 2, 6, projection='3d')
229 surf = ax.scatter(X[990000:1000000, 0], X[990000:1000000, 1], y_true
230 [990000:1000000], c=y_true[990000:1000000], cmap=cm.coolwarm)
231 ax.set_title("t = {:.2f}".format(X[990000, 2]))

```

```

228 ax.set_zlim(-0.1, 1)
229 ax.set_xlabel('$x$')
230 ax.set_ylabel('$y$')
231 ax.set_zlabel('$Temperatur$')
232 fig.colorbar(surf, shrink=.5, location='left')
233
234 plt.tight_layout()
235 plt.savefig("analytic2d.png", dpi=300)
236
237
238
239 # # Numeric FTCS
240
241 # library untuk menghitung waktu running FTCS
242 import time
243 start = time.time()
244 dx = 0.02
245 dy = dx
246 dt = dx**2 / (4*a)
247
248 # inisialisasi domain
249 xs = np.arange(0, L, dx)
250 ys = np.arange(0, L, dy)
251 ts = np.arange(0, maxtime, dt)
252 u = np.empty((len(xs), len(ys)))
253 for i in range(len(xs)):
254     for j in range(len(ys)):
255         u[i, j] = np.sin(np.pi*xs[i]) * np.sin(np.pi*ys[j]) #initial
256         condition
257 u[0, :] = 0
258 u[-1, :] = 0
259 u[:, 0] = 0
260 u[:, -1] = 0    #boundary condition
261
262 XS, YS = np.meshgrid(xs, ys)
263 y_num = np.empty(4)
264
265 for n in ts: #iterasi melalui waktu
266     un = u.copy()
267     u[1:-1, 1:-1] = (un[1:-1, 1:-1] + a * dt / dx**2 *

```

```

268                     (un[1:-1, 2:] - 2 * un[1:-1, 1:-1] + un[1:-1,
269 0:-2]) +
270                     a * dt / dy**2 * (un[2:, 1: -1] - 2 * un[1:-1,
271 1:-1] + un[0:-2, 1:-1]))
272 u[0, :] = 0
273 u[-1, :] = 0
274 u[:, 0] = 0
275 u[:, -1] = 0
276
277 temp = np.hstack((XS.reshape(-1,1), YS.reshape(-1,1), np.full((
278 len(xs) * len(ys)), n).reshape(-1,1), u.reshape(-1,1)))
279 y_num = np.vstack((y_num, temp))
280
281 y_num = np.delete(y_num, 0, 0)
282
283 end = time.time()
284
285 print('total waktu kalkulasi FTCS', end-start)
286
287
288
289 markersize = [1 for i in range(2500)]
290 ax = fig.add_subplot(3, 2, 1, projection='3d')
291 surf = ax.scatter(y_num[0:2500, 0], y_num[0:2500, 1], y_num[0:2500,
292 3], s=markersize, c=y_num[0:2500, 3], cmap=cm.coolwarm)
293 ax.set_title("t = {:.2f}".format(y_num[0, 2]))
294 ax.set_zlim(-0.1, 1)
295 ax.set_xlabel('$x$')
296 ax.set_ylabel('$y$')
297 ax.set_zlabel('$Temperatur$')
298 fig.colorbar(surf, shrink=.5, location='left')
299
300 ax = fig.add_subplot(3, 2, 2, projection='3d')
301 surf = ax.scatter(y_num[1000000:1002500, 0], y_num[1000000:1002500,
302 1], y_num[1000000:1002500, 3], s=markersize, c=y_num
[1000000:1002500, 3], cmap=cm.coolwarm)
303 ax.set_title("t = {:.2f}".format(y_num[1000000, 2]))
304 ax.set_zlim(-0.1, 1)

```

```

303 ax.set_xlabel('$x$')
304 ax.set_ylabel('$y$')
305 ax.set_zlabel('$Temperatur$')
306 fig.colorbar(surf, shrink=.5, location='left')
307
308 ax = fig.add_subplot(3, 2, 3, projection='3d')
309 surf = ax.scatter(y_num[2000000:2002500, 0], y_num[2000000:2002500,
1], y_num[2000000:2002500, 3], s=markersize, c=y_num
[2000000:2002500, 3], cmap=cm.coolwarm)
310 ax.set_title("t = {:.2f}".format(y_num[2000000, 2]))
311 ax.set_zlim(-0.1, 1)
312 ax.set_xlabel('$x$')
313 ax.set_ylabel('$y$')
314 ax.set_zlabel('$Temperatur$')
315 fig.colorbar(surf, shrink=.5, location='left')
316
317 ax = fig.add_subplot(3, 2, 4, projection='3d')
318 surf = ax.scatter(y_num[3000000:3002500, 0], y_num[3000000:3002500,
1], y_num[3000000:3002500, 3], s=markersize, c=y_num
[3000000:3002500, 3], cmap=cm.coolwarm)
319 ax.set_title("t = {:.2f}".format(y_num[3000000, 2]))
320 ax.set_zlim(-0.1, 1)
321 ax.set_xlabel('$x$')
322 ax.set_ylabel('$y$')
323 ax.set_zlabel('$Temperatur$')
324 fig.colorbar(surf, shrink=.5, location='left')
325
326 ax = fig.add_subplot(3, 2, 5, projection='3d')
327 surf = ax.scatter(y_num[4000000:4002500, 0], y_num[4000000:4002500,
1], y_num[4000000:4002500, 3], s=markersize, c=y_num
[4000000:4002500, 3], cmap=cm.coolwarm)
328 ax.set_title("t = {:.2f}".format(y_num[4000000, 2]))
329 ax.set_zlim(-0.1, 1)
330 ax.set_xlabel('$x$')
331 ax.set_ylabel('$y$')
332 ax.set_zlabel('$Temperatur$')
333 fig.colorbar(surf, shrink=.5, location='left')
334
335 ax = fig.add_subplot(3, 2, 6, projection='3d')
336 surf = ax.scatter(y_num[4997500:5000000, 0], y_num[4997500:5000000,
1], y_num[4997500:5000000, 3], s=markersize, c=y_num

```

```

[4997500:5000000, 3], cmap=cm.coolwarm)
337 ax.set_title("t = {:.2f}".format(y_num[4997500, 2]))
338 ax.set_zlim(-0.1, 1)
339 ax.set_xlabel('$x$')
340 ax.set_ylabel('$y$')
341 ax.set_zlabel('$Temperatur$')
342 fig.colorbar(surf, shrink=.5, location='left')
343
344 plt.tight_layout()
345 plt.savefig("FTCS2d.png", dpi=300)
346
347 print('mean squared error FTCS: ', dde.metrics.mean_squared_error((
    heat_eq_exact_solution(y_num[:, 0], y_num[:, 1], y_num[:, 2]),
    y_num[:, 3]))
348
349
350
351 # Animasi hasil
352 result = np.hstack((X, y_true, y_PINN))
353 time = np.unique(result[:, 2])
354 time2 = np.unique(y_num[:, 2])
355
356 fig = plt.figure(figsize=(7,5))
357 ax = fig.add_subplot(1, 1, 1, projection='3d')
358
359 def update(frame):
360     ax.clear()
361     ax.set_title(" ")
362     ax.set_zlim([-1, 1.1])
363     ax.set_zlabel("Temperatur")
364     ax.set_ylabel("y")
365     ax.set_xlabel("x")
366     ax.grid()
367     # untuk animasi PINN uncomment 4 line berikut
368     # a = np.where(result[:, 2] == time[frame])[0][0]
369     # b = np.where(result[:, 2] == time[frame])[0][-1]
370     # surface = ax.scatter(result[a:b, 0], result[a:b, 1], result[a:b,
371     # , 4], c=result[a:b, 4], cmap=cm.coolwarm)
372     # ax.set_title("Persebaran panas PINN t = {:.2f}".format(result[a
372     , 4]))
```

```

373     # untuk animasi analitik uncomment 4 line berikut
374     #     a = np.where(result[:, 2] == time[frame])[0][0]
375     #     b = np.where(result[:, 2] == time[frame])[0][-1]
376     #     surface = ax.scatter(result[a:b, 0], result[a:b, 1], result[a:b,
377     , 3], c=result[a:b, 3], cmap=cm.coolwarm)
378     #     ax.set_title("Persebaran panas analitik t = {:.2f}".format(
379     #         result[a, 3]))
380
381     # untuk animasi FTCS uncomment 4 line berikut
382     a = np.where(y_num[:, 2] == time2[frame])[0][0]
383     b = np.where(y_num[:, 2] == time2[frame])[0][-1]
384     surface = ax.scatter(y_num[a:b, 0], y_num[a:b, 1], y_num[a:b, 3],
385     c=y_num[a:b, 3], cmap=cm.coolwarm)
386     ax.set_title("Persebaran panas FTCS t = {:.2f}".format(y_num[a,
387     2]))
388
389 ani = FuncAnimation(fig, update, frames=int(len(time2)))
390
391
392 # simpan plot loss history
393 loss_train = np.sum(losshistory.loss_train, axis=1)
394 loss_test = np.sum(losshistory.loss_test, axis=1)
395
396 plt.figure(figsize=(6,4))
397 plt.grid()
398 plt.title("Perkembangan $loss$ $function$ tiap iterasi")
399 plt.semilogy(losshistory.steps, loss_train, label="Train loss")
400 plt.semilogy(losshistory.steps, loss_test, label="Test loss")
401 for i in range(len(losshistory.metrics_test[0])):
402     plt.semilogy(
403         loss_history.steps,
404         np.array(loss_history.metrics_test)[:, i],
405         label="Test metric",
406     )
407 plt.xlabel("Banyak iterasi")
408 plt.ylabel("$\mathcal{L}$")
409 plt.legend()

```

```
410 plt.savefig('loss.png', dpi=300)
```

C.3 PROGRAM PERSAMAAN PANAS DENGAN SUMBER PANAS

C.3.1 Variasi Sumber Panas di Tengah Domain

```
1 # # PERSAMAAN PANAS 2 DIMENSI DENGAN SUMBER PANAS (VARIASI PANAS DI
2 # TENGAH DOMAIN)
3 import deepxde as dde
4 import numpy as np
5 from deepxde.backend import tf
6
7 # library untuk membuat animasi:
8 from IPython.display import HTML
9 import matplotlib.pyplot as plt
10 from matplotlib.animation import FuncAnimation
11 from matplotlib.animation import PillowWriter
12 from matplotlib import cm
13
14 # pengaturan
15 plt.rcParams['animation.ffmpeg_path'] = r'C:\FFmpeg\bin\ffmpeg.exe'
16 dde.config.set_default_float("float64")
17 dde.optimizers.config.set_LBFGS_options(maxiter=15000)
18
19 # # PINN
20 # fungsi untuk memperoleh titik2 domain
21 def gen_points():
22     # jumlah titik tiap dimensi:
23     x_dim, y_dim, t_dim = (50, 50, 1000)
24
25     # batas dari x dan t:
26     x_min, y_min, t_min = (0, 0, 0.0)
27     x_max, y_max, t_max = (L, L, maxtime)
28
29     # titik-titik domain
30     t = np.linspace(t_min, t_max, num=t_dim).reshape(t_dim, 1)
31     x = np.linspace(x_min, x_max, num=x_dim).reshape(x_dim, 1)
32     y = np.linspace(y_min, y_max, num=y_dim).reshape(y_dim, 1)
33
34     # Save hasil
35     np.savez("heat_eq_data", x=x, y=y, t=t)
```

```

36
37 # baca titik2 domain
38 def gentestdata():
39     # Load data:
40     data = np.load("heat_eq_data.npz")
41     t, x, y = data["t"], data["x"], data["y"]
42     # ratakan data
43     yy, tt, xx = np.meshgrid(y, t, x)
44     X = np.vstack((np.ravel(xx), np.ravel(yy), np.ravel(tt))).T
45     return X
46
47 # Problem parameters:
48 a = 2.225e-5 # Thermal diffusivity
49 L = 1 # panjang interval
50 maxtime = 10000
51 var = .05
52 mean = .5
53
54 # peroleh titik2 domain
55 gen_points()
56
57
58 def pde(x, u):
59     # persamaan diferensial parsial: persamaan panas 2d dengan sumber
      panas
60     du_xx = dde.grad.hessian(u, x, i=0, j=0)
61     du_yy = dde.grad.hessian(u, x, i=1, j=1)
62     du_t = dde.grad.jacobian(u, x, i=0, j=2)
63     K = 0.00085*1000/63.65
64     S = K / (var*var*(2*np.pi)) * tf.exp(-0.5*((x[:, 0:1]-mean)/var)
      **2 + ((x[:, 1:2]-mean)/var)**2))
65     return du_t - (a * (du_xx + du_yy)) - S
66
67 # geometri domain permasalahan
68 geom = dde.geometry.Rectangle((0,0), (L,L))
69 timedomain = dde.geometry.TimeDomain(0, maxtime)
70 geomtime = dde.geometry.GeometryXTime(geom, timedomain)
71
72 # Initial and boundary conditions:
73 bc = dde.icbc.DirichletBC(geomtime, lambda x: 25, lambda _,
      on_boundary: on_boundary)

```

```

74 ic = dde.icbc.IC(
75     geomtime,
76     lambda x: 25,
77     lambda _, on_initial: on_initial,
78         )
79
80 # definisi PDP dan konfigurasi neural network
81 data = dde.data.TimePDE(
82     geomtime,
83     pde,
84     [bc, ic],
85     num_domain=5000,
86     num_boundary=400,
87     num_initial=200,
88     num_test=5000,
89 )
90 net = dde.nn.FNN([3] + [32] * 4 + [1], "tanh", "Glorot normal")
91 model = dde.Model(data, net)
92
93
94 # fungsi feature transform
95 def feature_transform(X):
96     return tf.concat([tf.reshape(X[:, 0:1], (-1,1)), tf.reshape(X[:, 1:2], (-1,1)), tf.reshape(X[:, 2:], (-1,1))*1e-3], axis=1)
97
98 # tambahkan layer feature transform
99 net.apply_feature_transform(feature_transform)
100
101 # tambahkan layer output transform
102 net.apply_output_transform(lambda x, y: y*100)
103
104 # Bangun dan latih model dengan 2 metode optimasi
105 model.compile("adam", lr=1e-3, loss_weights=[1e3,1e-2,1e-2])
106 model.train(iterations=10000)
107 model.compile("L-BFGS", loss_weights=[1e3,1e-2,1e-2])
108 losshistory, train_state = model.train()
109
110 # Plot/print hasil
111 dde.saveplot(losshistory, train_state, issave=True, isplot=True)
112 X = gentestdata()
113 y_PINN = model.predict(X)

```

```

114 np.savetxt("test.dat", np.hstack((X, y_PINN)))
115
116
117
118 # plot metode PINN
119 fig = plt.figure(figsize=(11,11))
120 plt.suptitle("Metode PINN")
121
122 ax = fig.add_subplot(3, 2, 1, projection='3d')
123 surf = ax.scatter(X[2500:5000, 0], X[2500:5000, 1], y_PINN
124 [2500:5000], c=y_PINN[2500:5000], cmap=cm.coolwarm)
125 ax.set_title("t = {:.0f}".format(X[0, 2]))
126 ax.set_zlim(20, 250)
127 ax.set_xlabel('$x$')
128 ax.set_ylabel('$y$')
129 ax.set_zlabel('$Temperatur$')
130 fig.colorbar(surf, shrink=.5, location='left')
131
132 ax = fig.add_subplot(3, 2, 2, projection='3d')
133 surf = ax.scatter(X[10000:12500, 0], X[10000:12500, 1], y_PINN
134 [10000:12500], c=y_PINN[10000:12500], cmap=cm.coolwarm)
135 ax.set_title("t = {:.0f}".format(X[10000, 2]))
136 ax.set_zlim(20, 250)
137 ax.set_xlabel('$x$')
138 ax.set_ylabel('$y$')
139 ax.set_zlabel('$Temperatur$')
140 fig.colorbar(surf, shrink=.5, location='left')
141
142 ax = fig.add_subplot(3, 2, 3, projection='3d')
143 surf = ax.scatter(X[100000:102500, 0], X[100000:102500, 1], y_PINN
144 [100000:102500], c=y_PINN[100000:102500], cmap=cm.coolwarm)
145 ax.set_title("\n \nt = {:.0f}".format(X[100000, 2]))
146 ax.set_zlim(20, 250)
147 ax.set_xlabel('$x$')
148 ax.set_ylabel('$y$')
149 ax.set_zlabel('$Temperatur$')
150 fig.colorbar(surf, shrink=.5, location='left')
151
152 ax = fig.add_subplot(3, 2, 4, projection='3d')
153 surf = ax.scatter(X[500000:502500, 0], X[500000:502500, 1], y_PINN
154 [500000:502500], c=y_PINN[500000:502500], cmap=cm.coolwarm)

```

```

151 ax.set_title(" \n \nt = 2000)".format(X[500000, 2]))
152 ax.set_zlim(20, 250)
153 ax.set_xlabel('$x$')
154 ax.set_ylabel('$y$')
155 ax.set_zlabel('$Temperatur$')
156 fig.colorbar(surf, shrink=.5, location='left')
157
158 ax = fig.add_subplot(3, 2, 5, projection='3d')
159 surf = ax.scatter(X[1497500:1500000, 0], X[1497500:1500000, 1],
160 y_PINN[1497500:1500000], c=y_PINN[1497500:1500000], cmap=cm.
161 coolwarm)
162 ax.set_title(" \n \nt = 6000)".format(X[1497500, 2]))
163 ax.set_zlim(20, 250)
164 ax.set_xlabel('$x$')
165 ax.set_ylabel('$y$')
166 ax.set_zlabel('$Temperatur$')
167 fig.colorbar(surf, shrink=.5, location='left')
168
169 ax = fig.add_subplot(3, 2, 6, projection='3d')
170 surf = ax.scatter(X[2497500:2500000, 0], X[2497500:2500000, 1],
171 y_PINN[2497500:2500000], c=y_PINN[2497500:2500000], cmap=cm.
172 coolwarm)
173 ax.set_title(" \n \nt = {:.0f}".format(X[2497500, 2]))
174 ax.set_zlim(20, 250)
175 ax.set_xlabel('$x$')
176 ax.set_ylabel('$y$')
177 ax.set_zlabel('$Temperatur$')
178 fig.colorbar(surf, shrink=.5, location='left')
179
180
181 # Plot temperatur terhadap waktu pada 6 titik
182 idx1 = np.where(np.isclose(X[:, 0], 0) & (np.isclose(X[:, 0], X[:, 1])))[0]
183 idx2 = np.where((X[:, 0] >= 0.07) & (X[:, 0] <= 0.09) & (np.isclose(X
184 [:, 0], X[:, 1])))[0]
185 idx3 = np.where((X[:, 0] >= 0.17) & (X[:, 0] <= 0.19) & (np.isclose(X
186 [:, 0], X[:, 1])))[0]

```

```

185 idx4 = np.where((X[:, 0] >= 0.27) & (X[:, 0] <= 0.29) & (np.isclose(X
186 [:, 0], X[:, 1])))[0]
187 idx5 = np.where((X[:, 0] >= 0.37) & (X[:, 0] <= 0.39) & (np.isclose(X
188 [:, 0], X[:, 1])))[0]
189 idx6 = np.where((X[:, 0] >= 0.47) & (X[:, 0] <= 0.49) & (np.isclose(X
190 [:, 0], X[:, 1])))[0]
191
192 result1 = X[idx1]
193 result2 = X[idx2]
194 result3 = X[idx3]
195 result4 = X[idx4]
196 result5 = X[idx5]
197 result6 = X[idx6]
198
199 fig = plt.figure(figsize=(10,7))
200 plt.suptitle("Perkembangan Temperatur terhadap Waktu pada Beberapa
201 Titik")
202
203 ax = fig.add_subplot(2, 3, 1)
204 ax.set_ylabel("Temperatur")
205 ax.set_xlabel("t")
206 ax.plot(result1[:, 2], y_PINN[idx1])
207 plt.title("(x, y) = ({:.2f}, {:.2f})".format(result1[0, 1], result1
208 [0, 1]))
209 plt.ylim((0,100))
210
211 ax = fig.add_subplot(2, 3, 2)
212 ax.set_ylabel("Temperatur")
213 ax.set_xlabel("t")
214 ax.plot(result2[:, 2], y_PINN[idx2])
215 plt.title("(x, y) = ({:.2f}, {:.2f})".format(result2[0, 1], result2
216 [0, 1]))
217 plt.ylim((0,100))
218

```

```

219
220 ax = fig.add_subplot(2, 3, 4)
221 ax.set_ylabel("Temperatur")
222 ax.set_xlabel("t")
223 ax.plot(result4[:, 2], y_PINN[idx4])
224 plt.title("(x, y) = {:.2f}, {:.2f})".format(result4[0, 1], result4
225 [0, 1]))
226 plt.ylim((0,300))
227
228 ax = fig.add_subplot(2, 3, 5)
229 ax.set_ylabel("Temperatur")
230 ax.set_xlabel("t")
231 ax.plot(result5[:, 2], y_PINN[idx5])
232 plt.title("(x, y) = {:.2f}, {:.2f})".format(result5[0, 1], result5
233 [0, 1]))
234 plt.ylim((0,300))
235
236 ax = fig.add_subplot(2, 3, 6)
237 ax.set_ylabel("Temperatur")
238 ax.set_xlabel("t")
239 ax.plot(result6[:, 2], y_PINN[idx6])
240 plt.title("(x, y) = {:.2f}, {:.2f})".format(result6[0, 1], result6
241 [0, 1]))
242 plt.ylim((0,300))
243
244
245
246 # Animasi hasil
247 result = np.hstack((X, y_PINN))
248 time = np.unique(result[:, 2])
249
250 fig = plt.figure(figsize=(7,5))
251 ax = fig.add_subplot(1, 1, 1, projection='3d')
252
253 def update(frame):
254     ax.clear()
255     ax.set_title(" ")
256     ax.set_zlim([20, 150])

```

```

257     ax.set_zlabel("Temperatur")
258     ax.set_ylabel("y")
259     ax.set_xlabel("x")
260     ax.grid()
261     a = np.where(result[:, 2] == time[frame])[0][0]
262     b = np.where(result[:, 2] == time[frame])[0][-1]
263     surface = ax.scatter(result[a:b, 0], result[a:b, 1], result[a:b,
264     3], c=result[a:b, 3], cmap=cm.coolwarm)
265     ax.set_title("Persebaran panas t = {:.0f}".format(result[a, 2]))
266
267
268 # Save animasi sebagai video
269 ani.save("heat eq 2dhsvar1.mp4", bitrate=6000, dpi=200, fps=30)
270
271
272
273 # simpan plot loss history
274 loss_train = np.sum(losshistory.loss_train, axis=1)
275 loss_test = np.sum(losshistory.loss_test, axis=1)
276
277 plt.figure(figsize=(6,4))
278 plt.grid()
279 plt.title("Perkembangan $loss$ $function$ tiap iterasi")
280 plt.semilogy(losshistory.steps, loss_train, label="Train loss")
281 plt.semilogy(np.concatenate((losshistory.steps[:12], losshistory.
282     steps[-1:])), np.concatenate((loss_test[:12], loss_test[-1:])),
283     label="Test loss")
284 for i in range(len(losshistory.metrics_test[0])):
285     plt.semilogy(
286         loss_history.steps,
287         np.array(loss_history.metrics_test)[:, i],
288         label="Test metric",
289     )
290 plt.xlabel("Banyak iterasi")
291 plt.ylabel("$\mathcal{L}$")
292 plt.legend()
293 plt.savefig('loss.png', dpi=300)

```

C.3.2 Variasi Sumber Panas di Pinggir Domain

```
1 # # PERSAMAAN PANAS 2 DIMENSI DENGAN SUMBER PANAS (VARIASI PANAS DI
```

```

    TENGAH DOMAIN)
2 import deepxde as dde
3 import numpy as np
4 from deepxde.backend import tf
5
6 # library untuk membuat animasi:
7 from IPython.display import HTML
8 import matplotlib.pyplot as plt
9 from matplotlib.animation import FuncAnimation
10 from matplotlib.animation import PillowWriter
11 from matplotlib import cm
12
13 # pengaturan
14 plt.rcParams['animation.ffmpeg_path'] = r'C:\FFmpeg\bin\ffmpeg.exe'
15 dde.config.set_default_float("float64")
16 dde.optimizers.config.set_LBFGS_options(maxiter=15000)
17
18
19 # # PINN
20 # fungsi untuk memperoleh titik2 domain
21 def gen_points():
22     # jumlah titik tiap dimensi:
23     x_dim, y_dim, t_dim = (50, 50, 1000)
24
25     # batas dari x dan t:
26     x_min, y_min, t_min = (0, 0, 0.0)
27     x_max, y_max, t_max = (L, L, maxtime)
28
29     # titik-titik domain
30     t = np.linspace(t_min, t_max, num=t_dim).reshape(t_dim, 1)
31     x = np.linspace(x_min, x_max, num=x_dim).reshape(x_dim, 1)
32     y = np.linspace(y_min, y_max, num=y_dim).reshape(y_dim, 1)
33
34     # Save hasil
35     np.savez("heat_eq_data", x=x, y=y, t=t)
36
37 # baca titik2 domain
38 def gentestdata():
39     # Load data:
40     data = np.load("heat_eq_data.npz")
41     t, x, y = data["t"], data["x"], data["y"]

```

```

42     # ratakan data
43     yy, tt, xx = np.meshgrid(y, t, x)
44     X = np.vstack((np.ravel(xx), np.ravel(yy), np.ravel(tt))).T
45     return X
46
47 # Problem parameters:
48 a = 2.225e-5 # Thermal diffusivity
49 L = 1 # panjang interval
50 maxtime = 10000
51 var = .05
52 mean = .25
53
54 # peroleh titik2 domain
55 gen_points()
56
57
58 def pde(x, u):
59     # persamaan diferensial parsial: persamaan panas 2d dengan sumber
60     # panas
61     du_xx = dde.grad.hessian(u, x, i=0, j=0)
62     du_yy = dde.grad.hessian(u, x, i=1, j=1)
63     du_t = dde.grad.jacobian(u, x, i=0, j=2)
64     K = 0.00085*1000/63.65
65     S = K / (var*var*(2*np.pi)) * tf.exp(-0.5*((x[:, 0:1]-mean)/var)
66     **2 + ((x[:, 1:2]-mean)/var)**2))
67     return du_t - (a * (du_xx + du_yy)) - S
68
69 # geometri domain permasalahan
70 geom = dde.geometry.Rectangle((0,0), (L,L))
71 timedomain = dde.geometry.TimeDomain(0, maxtime)
72 geomtime = dde.geometry.GeometryXTime(geom, timedomain)
73
74 # Initial and boundary conditions:
75 bc = dde.icbc.DirichletBC(geomtime, lambda x: 25, lambda _,
76                             on_boundary: on_boundary)
77 ic = dde.icbc.IC(
78     geomtime,
79     lambda x: 25,
80     lambda _, on_initial: on_initial,
81 )

```

```

80 # definisi PDP dan konfigurasi neural network
81 data = dde.data.TimePDE(
82     geomtime,
83     pde,
84     [bc, ic],
85     num_domain=5000,
86     num_boundary=400,
87     num_initial=200,
88     num_test=5000,
89 )
90 net = dde.nn.FNN([3] + [32] * 4 + [1], "tanh", "Glorot normal")
91 model = dde.Model(data, net)
92
93
94 # fungsi feature transform
95 def feature_transform(X):
96     return tf.concat([tf.reshape(X[:, 0:1], (-1,1)), tf.reshape(X[:, 1:2], (-1,1)), tf.reshape(X[:, 2:], (-1,1))*1e-3], axis=1)
97
98 # tambahkan layer feature transform
99 net.apply_feature_transform(feature_transform)
100
101 # tambahkan layer output transform
102 net.apply_output_transform(lambda x, y: y*100)
103
104 # Bangun dan latih model dengan 2 metode optimasi
105 model.compile("adam", lr=1e-3, loss_weights=[1e3,1e-2,1e-2])
106 model.train(iterations=10000)
107 model.compile("L-BFGS", loss_weights=[1e3,1e-2,1e-2])
108 losshistory, train_state = model.train()
109
110 # Plot/print hasil
111 dde.saveplot(losshistory, train_state, issave=True, isplot=True)
112 X = gentestdata()
113 y_PINN = model.predict(X)
114 np.savetxt("test.dat", np.hstack((X, y_PINN)))
115
116
117
118 # plot metode PINN
119 fig = plt.figure(figsize=(11,11))

```

```

120 plt.suptitle("Metode PINN")
121
122 ax = fig.add_subplot(3, 2, 1, projection='3d')
123 surf = ax.scatter(X[2500:5000, 0], X[2500:5000, 1], y_PINN
124 [2500:5000], c=y_PINN[2500:5000], cmap=cm.coolwarm)
125 ax.set_title("t = {:.0f}".format(X[0, 2]))
126 ax.set_zlim(20, 250)
127 ax.set_xlabel('$x$')
128 ax.set_ylabel('$y$')
129 ax.set_zlabel('$Temperatur$')
130 fig.colorbar(surf, shrink=.5, location='left')
131
132 ax = fig.add_subplot(3, 2, 2, projection='3d')
133 surf = ax.scatter(X[10000:12500, 0], X[10000:12500, 1], y_PINN
134 [10000:12500], c=y_PINN[10000:12500], cmap=cm.coolwarm)
135 ax.set_title("t = {:.0f}".format(X[10000, 2]))
136 ax.set_zlim(20, 250)
137 ax.set_xlabel('$x$')
138 ax.set_ylabel('$y$')
139 ax.set_zlabel('$Temperatur$')
140 fig.colorbar(surf, shrink=.5, location='left')
141
142 ax = fig.add_subplot(3, 2, 3, projection='3d')
143 surf = ax.scatter(X[100000:102500, 0], X[100000:102500, 1], y_PINN
144 [100000:102500], c=y_PINN[100000:102500], cmap=cm.coolwarm)
145 ax.set_title("\n \nt = {:.0f}".format(X[100000, 2]))
146 ax.set_zlim(20, 250)
147 ax.set_xlabel('$x$')
148 ax.set_ylabel('$y$')
149 ax.set_zlabel('$Temperatur$')
150 fig.colorbar(surf, shrink=.5, location='left')
151
152 ax = fig.add_subplot(3, 2, 4, projection='3d')
153 surf = ax.scatter(X[500000:502500, 0], X[500000:502500, 1], y_PINN
154 [500000:502500], c=y_PINN[500000:502500], cmap=cm.coolwarm)
155 ax.set_title("\n \nt = 2000")#{:.0f}".format(X[500000, 2]))
156 ax.set_zlim(20, 250)
157 ax.set_xlabel('$x$')
158 ax.set_ylabel('$y$')
159 ax.set_zlabel('$Temperatur$')
160 fig.colorbar(surf, shrink=.5, location='left')

```

```

157
158 ax = fig.add_subplot(3, 2, 5, projection='3d')
159 surf = ax.scatter(X[1497500:1500000, 0], X[1497500:1500000, 1],
160 y_PINN[1497500:1500000], c=y_PINN[1497500:1500000], cmap=cm.
161 coolwarm)
162 ax.set_title(" \n \nt = 6000)".format(X[1497500, 2]))
163 ax.set_zlim(20, 250)
164 ax.set_xlabel('$x$')
165 ax.set_ylabel('$y$')
166 ax.set_zlabel('$Temperatur$')
167 fig.colorbar(surf, shrink=.5, location='left')
168
169 ax = fig.add_subplot(3, 2, 6, projection='3d')
170 surf = ax.scatter(X[2497500:2500000, 0], X[2497500:2500000, 1],
171 y_PINN[2497500:2500000], c=y_PINN[2497500:2500000], cmap=cm.
172 coolwarm)
173 ax.set_title(" \n \nt = {:.0f}".format(X[2497500, 2]))
174 ax.set_zlim(20, 250)
175 ax.set_xlabel('$x$')
176 ax.set_ylabel('$y$')
177 ax.set_zlabel('$Temperatur$')
178 fig.colorbar(surf, shrink=.5, location='left')
179
180
181 # Plot temperatur terhadap waktu pada 6 titik
182 idx1 = np.where(np.isclose(X[:, 0], 1) & (np.isclose(X[:, 0], X[:, 1])))[0]
183 idx2 = np.where((X[:, 0] >= 0.84) & (X[:, 0] <= 0.86) & (np.isclose(X
184 [:, 0], X[:, 1])))[0]
185 idx3 = np.where((X[:, 0] >= 0.68) & (X[:, 0] <= 0.7) & (np.isclose(X
186 [:, 0], X[:, 1])))[0]
187 idx4 = np.where((X[:, 0] >= 0.54) & (X[:, 0] <= 0.56) & (np.isclose(X
188 [:, 0], X[:, 1])))[0]
189 idx5 = np.where((X[:, 0] >= 0.38) & (X[:, 0] <= 0.4) & (np.isclose(X
190 [:, 0], X[:, 1])))[0]
191 idx6 = np.where((X[:, 0] >= 0.24) & (X[:, 0] <= 0.26) & (np.isclose(X
192 [:, 0], X[:, 1])))[0]

```

```

188
189 result1 = X[idx1]
190 result2 = X[idx2]
191 result3 = X[idx3]
192 result4 = X[idx4]
193 result5 = X[idx5]
194 result6 = X[idx6]

195
196 fig = plt.figure(figsize=(10,7))
197 plt.suptitle("Perkembangan Temperatur terhadap Waktu pada Beberapa Titik")
198
199 ax = fig.add_subplot(2, 3, 1)
200 ax.set_ylabel("Temperatur")
201 ax.set_xlabel("t")
202 ax.plot(result1[:, 2], y_PINN[idx1])
203 plt.title("(x, y) = {:.2f}, {:.2f})".format(result1[0, 1], result1[0, 1]))
204 plt.ylim((0,100))
205
206 ax = fig.add_subplot(2, 3, 2)
207 ax.set_ylabel("Temperatur")
208 ax.set_xlabel("t")
209 ax.plot(result2[:, 2], y_PINN[idx2])
210 plt.title("(x, y) = {:.2f}, {:.2f})".format(result2[0, 1], result2[0, 1]))
211 plt.ylim((0,100))
212
213 ax = fig.add_subplot(2, 3, 3)
214 ax.set_ylabel("Temperatur")
215 ax.set_xlabel("t")
216 ax.plot(result3[:, 2], y_PINN[idx3])
217 plt.title("(x, y) = {:.2f}, {:.2f})".format(result3[0, 1], result3[0, 1]))
218 plt.ylim((0,100))
219
220 ax = fig.add_subplot(2, 3, 4)
221 ax.set_ylabel("Temperatur")
222 ax.set_xlabel("t")
223 ax.plot(result4[:, 2], y_PINN[idx4])
224 plt.title("(x, y) = {:.2f}, {:.2f})".format(result4[0, 1], result4[0, 1]))

```

```

[0, 1]))
225 plt.ylim((0,100))
226
227 ax = fig.add_subplot(2, 3, 5)
228 ax.set_ylabel("Temperatur")
229 ax.set_xlabel("t")
230 ax.plot(result5[:, 2], y_PINN[idx5])
231 plt.title("(x, y) = {:.2f}, {:.2f})".format(result5[0, 1], result5
232 [0, 1]))
233 plt.ylim((0,150))
234
235 ax = fig.add_subplot(2, 3, 6)
236 ax.set_ylabel("Temperatur")
237 ax.set_xlabel("t")
238 ax.plot(result6[:, 2], y_PINN[idx6])
239 plt.title("(x, y) = {:.2f}, {:.2f})".format(result6[0, 1], result6
240 [0, 1]))
241 plt.ylim((0,250))
242
243
244
245
246 # Animasi hasil
247 result = np.hstack((X, y_PINN))
248 time = np.unique(result[:, 2])
249
250 fig = plt.figure(figsize=(7,5))
251 ax = fig.add_subplot(1, 1, 1, projection='3d')
252
253 def update(frame):
254     ax.clear()
255     ax.set_title(" ")
256     ax.set_zlim([20, 150])
257     ax.set_zlabel("Temperatur")
258     ax.set_ylabel("y")
259     ax.set_xlabel("x")
260     ax.grid()
261     a = np.where(result[:, 2] == time[frame])[0][0]
262     b = np.where(result[:, 2] == time[frame])[0][-1]

```

```

263     surface = ax.scatter(result[a:b, 0], result[a:b, 1], result[a:b,
264         3], c=result[a:b, 3], cmap=cm.coolwarm)
265     ax.set_title("Persebaran panas t = {:.0f}".format(result[a, 2]))
266
267     ani = FuncAnimation(fig, update, frames=int(len(time)))
268
269 # Save animasi sebagai video
270 ani.save("heat eq 2dhsvar2.mp4", bitrate=6000, dpi=200, fps=30)
271
272
273 # simpan plot loss history
274 loss_train = np.sum(losshistory.loss_train, axis=1)
275 loss_test = np.sum(losshistory.loss_test, axis=1)
276
277 plt.figure(figsize=(6,4))
278 plt.grid()
279 plt.title("Perkembangan $loss$ $function$ tiap iterasi")
280 plt.semilogy(losshistory.steps, loss_train, label="Train loss")
281 plt.semilogy(np.concatenate((losshistory.steps[:12], losshistory.
282     steps[-1:])), np.concatenate((loss_test[:12], loss_test[-1:])),
283     label="Test loss")
284 for i in range(len(losshistory.metrics_test[0])):
285     plt.semilogy(
286         loss_history.steps,
287         np.array(loss_history.metrics_test)[:, i],
288         label="Test metric",
289     )
290 plt.xlabel("Banyak iterasi")
291 plt.ylabel("$\mathcal{L}$")
292 plt.legend()
293 plt.savefig('loss.png', dpi=300)

```