平成 28 年度　修士論文

# カンボジア国･トンレサップ湖の
# 流動把握に向けた 2 次元数値流動モデルの開発

## Development of 2D Numerical Flow Model to Grasp Hydrodynamics of Tonle Sap Lake, Combodia


指導教員　　中村　恭志　准教授
　　木内　豪　教授

東京工業大学
大学院総合理工学研究科
環境理工学創造専攻

15M29320

村上　駿

# 目次


第 1 章　序論 1

1-1.　研究背景 1
1-2.　既往研究 2
1-3.　研究概要 2
第 1 章 参考文献 4
**第 1 章 図表集**

第 2 章　流動計算モデル 5

2-1.　浅水方程式の導出 5
2-2.　離散化手法 6
2-2-1.　時間発展式の導出
2-2-2.　河床面勾配の考慮
2-2-3.　外力項の付与
2-3.　水域･陸域判定 13
2-4.　ベンチマークテスト計算 13
2-4-1.　1 次元テスト計算
2-4-2.　2 次元テスト計算
第 2 章 参考文献 16
**第 2 章 図表集**

第 3 章　GPGPU による計算速度高速化処理 17

3-1.　GPU による高速化原理 17
3-2.　CUDA プログラミング 18
3-2-1.　CUDA プログラミングモデル
3-2-2.　CUDA スレッドモデル
3-2-3.　CUDA メモリモデル
3-3.　CUDA による流動モデルの実装 21
3-3-1.　ホストコードプログラム
3-3-2.　デバイスコードプログラム
第 3 章 参考文献 24
**第 3 章 図表集**

第 4 章　トンレサップ湖を対象とした再現計算_____ 25

4-1.　再現計算条件 25
4-1-1.　計算期間･入力条件
4-1-2.　計算地形
4-1-3.　計算条件
4-2.　計算再現精度検証用データ 26
4-3.　計算結果 28
4-3-1.　水位変化
4-3-2.　水域変化
4-4.　GPGPU による高速化の評価 29
4-4-1.　CPU 計算速度と GPU 計算速度の比較
4-4-2.　GPU 計算速度の高速化
第 4 章 参考文献 32
**第 4 章 図表集**

第 5 章　結論と今後の課題_____ 33

5-1.　結論 33
5-2.　課題 34

**謝辞**_____ 35

# 第１章 序論

本研究はカンボジア国･トンレサップ湖の環境保全の一環として，水理学的状況把握のための数値流動モデルを開発するものである．モデル開発にあたっては，現地の技術水準，同研究対象スケールなどの要求に最適な手法を用いた．ここでは本研究の背景と既往研究に触れた上で，本研究の概要について述べる．

## 1-1．研究背景

カンボジア王国に位置するトンレサップ湖は全長最大 250km，最大面積 16,000km$^2$ の東南アジア最大の淡水湖である．同湖南部には唯一の流出河川である全長 120km のトンレサップ河があり，首都プノンペンでチベット高原を源流とする東南アジア最長の国際河川･メコン河に合流している(**図 1-1**)．また湖周囲はダムレイ山脈，ダンレク山脈，カルダモン山脈, エレファント山脈に囲まれ, 湖との間の丘から Stung Sen 等の支流が湖に注いでいる．湖底地形は平坦であり，乾季には水深が最低 0.5m 程度と非常に浅いが，一方で雨季にはメコン川増水の影響でトンレサップ河が逆流し水深 9m 程度まで水位が大きく上昇するなど，季節単位で水理学的状況が大きく変化する特徴を持っている(**図 1-2**，**図 1-3**)．雨季の湖面積は乾季の 6 倍程度に増加し，周辺に広大な氾濫原を持っている(**図 1-4**)．この氾濫作用により氾濫原には浸水林や灌木林が形成され広大な湿地帯となっている[1].

9 世紀から 15 世紀にかけてのクメール王朝時代からトンレサップ湖は文明繁栄の重要基盤として位置づけられ, 世界遺産のアンコール･ワットが同湖北西部のシェムリアップ市に建造されている．現在でも氾濫原には 120 万人が居住し，そのうち 30 万人程度は高床式や筏上の家などで内水面上に居住し，水位に合わせて年に何度かの移動を行っている．湖を囲む 6 県で国内総就業人口の約 30%が同湖を経済基盤とするといわれ，周辺氾濫原における浮稲の栽培等の農業と水域内での漁業は主要な食料供給源となっている[2]．特にメコン河も含めた内水面漁獲量は総漁獲量の約 8~9 割を占め，漁業が国内総生産(GDP)の 10%以上を占めることから同湖･河の環境保全が重要な課題となっている[3].

その一方でカンボジア王国は 1949 年にフランス領から独立したものの，ベトナム戦争の影響でアメリカと南北ベトナムの介入によりカンボジア内戦が勃発し，親米派のクーデターによって当時のシハヌーク国王が追放，クメール共和国の成立となった．しかしその後も内戦は激化の一途を辿り，ポル・ポト率いるクメール・ルージュ(共産党)がクメール共和国を打倒し 1975 年民主カンプチアを設立．その後 1979 年までの約 4 年間の間にクメール・ルージュが原始共産制の実現を目指し，200 万人以上の大量虐殺を行った．特にポル・ポトは徹底的な国家の改造を望み, 死者の中には教師, 医者, 資本家といった有識者が多かった.

そのためカンボジア内戦による政情不安，法整備の不備，内戦終了後の急激な人口増加などの様々な背景を受けて，森林破壊や水文特性の変化等が同湖における環境問題として挙げられている．近年では観光都市シェムリアップ市や首都プノンペンの都市･工場排水や水域内住居からの排水による富栄養化(**図 1-4**)と水質の悪化が報告され，有害な植物プランクトンの増殖や病原性細菌の蔓延などの危険性が増加している[4].

## 1-2. 既往研究

笠井[2]によると，トンレサップ湖における環境問題は JICA の調査により 7 つに分類されているが，そのうち水理に関わる問題としてラムサール条約に指定されている湿地帯及び浸水林の破壊，水文特性の変化と土砂堆積，生物多様性の劣化，過剰漁業による漁業資源の劣化，水質汚染と富栄養化などが挙げられている．しかしながら先述した内戦の影響によりカンボジアには技術研究者が慢性的に不足しており，国内研究者よりも国外研究者による研究が多く報告されている．

その中でも藤井[5]はカンボジアメコン氾濫域の水理を対象に流量観測や衛星モニタリング，水文水理モデルなどを利用して氾濫原の水収支や氾濫緩和量の解析を行っている．大八木ら[6]はシェムリアップ川における水質特性に着目し，乾季には水位が低下し湖水量が減少するため，都市部からの排水の影響が表面化すると報告している．また，社会的な研究としても水上集落の住居･集落の空間構成の解明を扱った例[7]やコミュニティー漁業を取り扱った例[8]も報告されている．

しかしながらトンレサップ湖を対象に水理の解析を行った例は少なく，また観測体制も整っていないため同湖の水理的構造は解明されていないのが現状である．

## 1-3. 研究概要

今後，健全な水環境を維持していくには，同湖における流動など水理学的な状況を把握することが必要であるが，経済的･人的資源の制約から現地観測体制は現状ではほとんど整備されていない．そこでごく限られた観測データを補うには，数値流動解析モデルが有効である．しかし，数百 km に広がり，季節的変動を考慮するために長期間の解析が必要となるため高計算負荷となる解析を高速に実現できることが数値解析モデルに必要となる．また，商用モデルを含め現在の既往数値解析モデルの多くは高度な計算手法を用いているが，長い内戦により専門知識を持つ技術者が極度に不足するカンボジアでは，今後カンボジア人が自ら解析を実行し水質管理に供していくには出来るだけ簡便な手法を用いたモデルが必要とされる．

そこで本研究では，以上のような要求を満たす数値解析モデルとして直交格子と単純な

計算手法を用いる一方，高速化手法である GPGPU を適用して高速な計算が可能な数値流動解析モデルを新たに開発することとした．同湖は平坦な地形を持ち，水深は浅いことから，流動は水深方向に一様性が高いと考え，水深平均値を計算対象とする浅水 2 次元流モデルを対象とした．

本論の構成は次の通りである．第 2 章「流動計算モデル」では基礎方程式となる水平 2 次元の浅水流方程式の導出について述べ，先述した直交格子に合わせた離散化手法，並びに計算精度確認のため解析解の存在する一様斜面への遡上問題と 2 次元ボウル形状内スロッシング問題の 2 つのベンチマークテストを行った．

第 3 章「GPGPU による計算速度高速化処理」では第 2 章の計算モデルをトンレサップ湖へ適用するため，GPU(画像処理用演算プロセッサ)を計算ユニットとして利用する GPGPU(GPU による汎用計算)によって計算速度の高速化を図った．本研究以前にも GPGPU は流体計算や天体物理などのシミュレーションに適用されており，特に迅速な計算が求められる津波シミュレーションでも研究が進んでいる[9][10]．計算モデルの GPGPU 化においては GPU ベンダーである NVIDIA 社により無償で提供されている CUDA ライブラリを使用した．

第 4 章「トンレサップ湖を対象とした再現計算」では以上の高速化されたモデルを用いて，2002 年の湖の氾濫･縮小現象を対象に再現計算を行った．計算結果の再現精度確認にはメコン川委員会(MRC)の水位データと Landsat 衛星画像データを使用し，水位変化と水面形変化の再現性について検証を行った．またここでは先述した GPGPU 化により計算時間がどの程度短縮化されたかについても述べる．

第 5 章「結論と今後の課題」では以上で得られた結論についてまとめ，今後改善すべき課題について述べる．

## 第 1 章 参考文献


1. I. C. Campbell, C. Pool, W. Giesen, J. V. Jorgensen: Species diversity and ecology of Tonle Sap Great Lake, *Aquatic Science*, Vol. 68, pp.355-353, 2006
2. 笠井 利之: カンボジア・トンレサップ湖地域の環境保全についての予備的考察，立命館国際地域研究，第 21 号，pp.41-64，2003
3. 高橋 信吾，石川 智士，黒倉 寿: カンボジアの内水面漁業，水文･水資源学会誌，大 18 巻，2 号，pp.185-193，2005
4. 科学技術振興機構(JST)，トンレサップ湖における環境保全基盤の構築，https://www.jst.go.jp/global/kadai/h2703_cambodia.html
5. 藤井 秀人: トンレサップ湖および周辺氾濫域の洪水緩和機能，農業土木学会誌，第 72 巻，7 号，pp.599-603，2004
6. 大八木 英夫, 遠藤 修一, 奥村 康昭, 石川 俊之, 塚脇 真二: カンボジア・トンレサップ湖とその集水域における水質特性について―シェムリアプ川河口域を中心として―，日本陸水学会大会講演要旨集，第 73 巻，pp.179，2008
7. 脇田 祥尚, 前田 幸大: 水上集落における住居・集落の空間構成－アンロン・タ・ウー村（カンボジア・トンレサップ湖）を事例にして－，日本建築学会計画系論文集，第 75 巻，pp.2107-2114，2010
8. 環境省，カンボジア･トンレサップ湖のコミュニティー漁業，http://www.env.go.jp/nature/satoyama/satonavi/initiative/kaigai/109.html
9. 青木 尊之: 双曲型方程式の GPU による高速計算，応用数理，第 20 巻，2 号，pp.32-45，2008
10. 曾 新翔，赤穂 良輔，石川忠晴，中村恭志: 非構造三角形格子に基づく津波数値解析の GPGPU による高速化，水工学論文集，第 57 巻，pp.619-624，2013

# 第1章
# 図表集

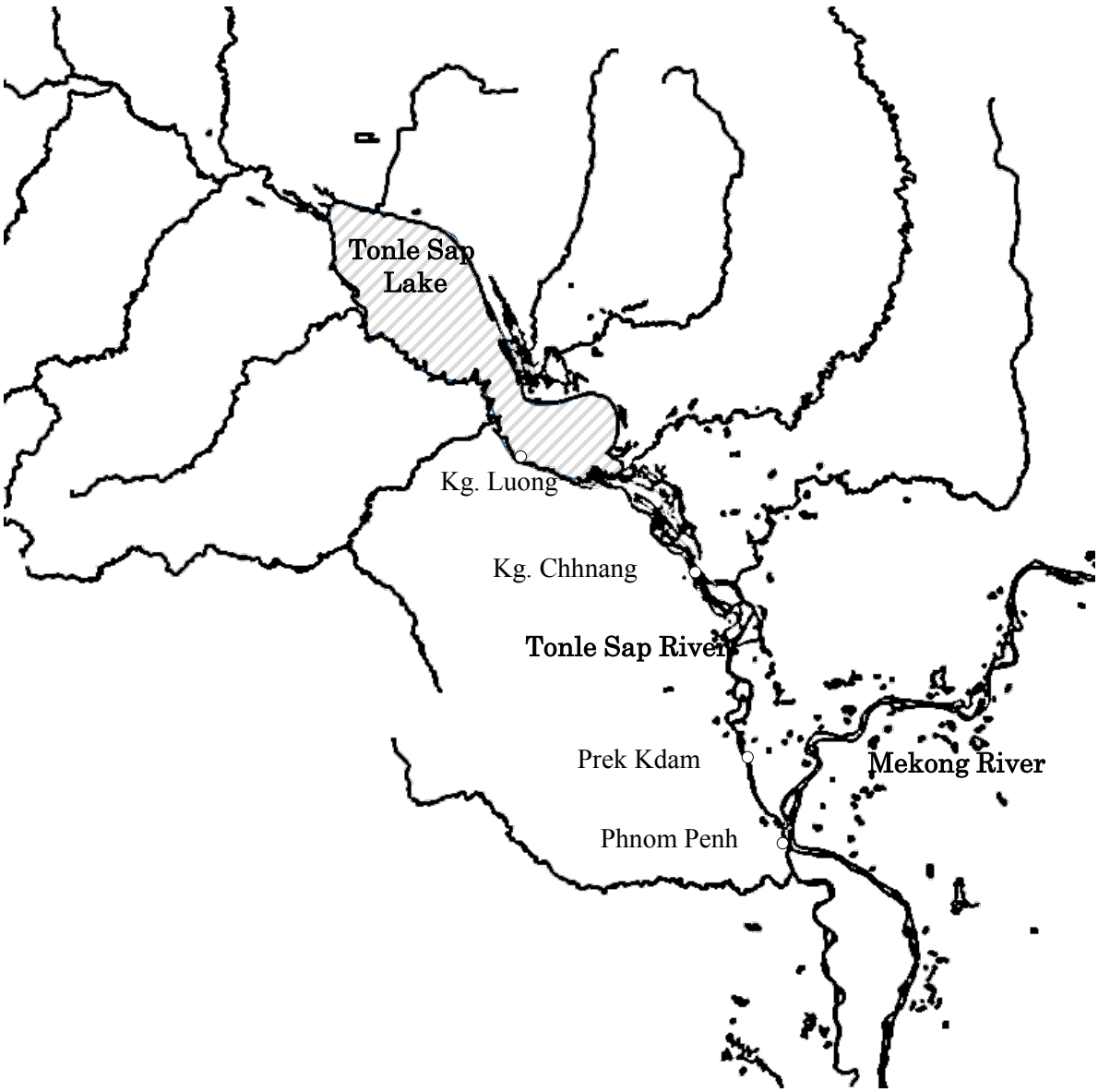
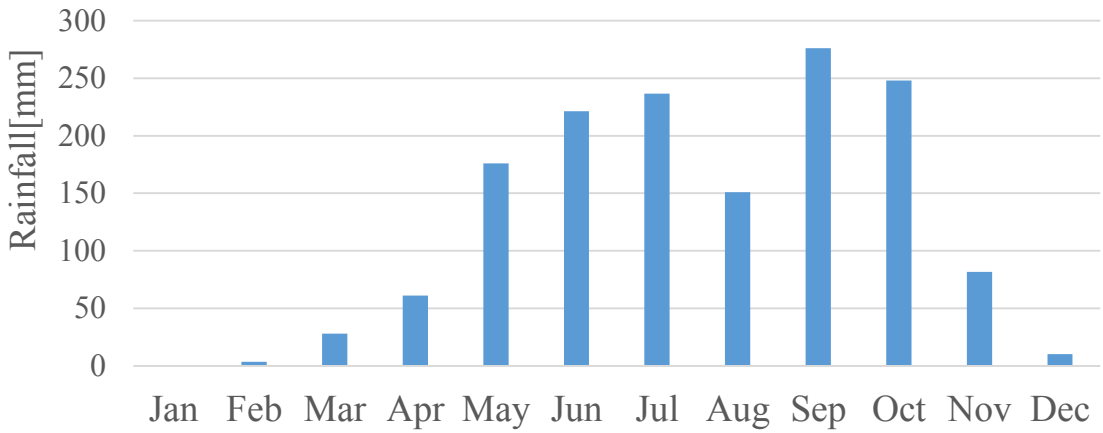



図1-1 トンレサップ水域概要図

月別降水量




湖水深
(Kompong Luong地点)

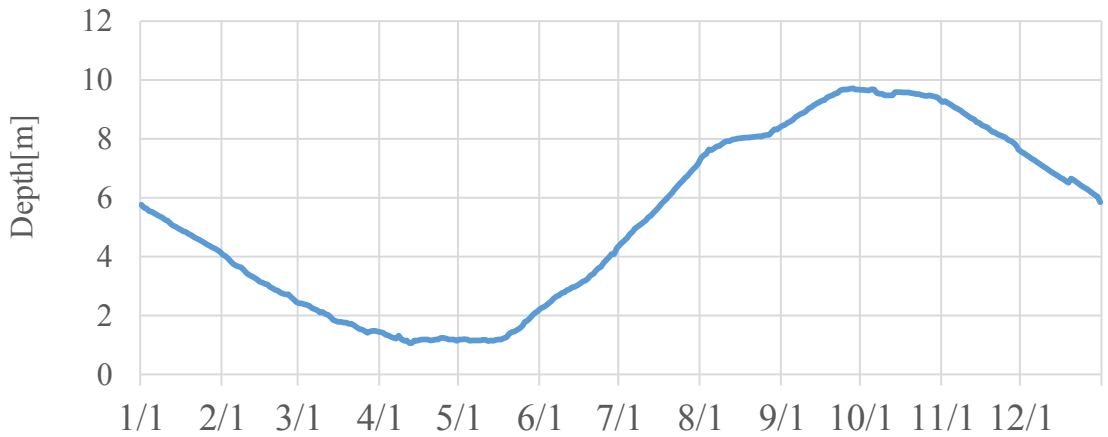


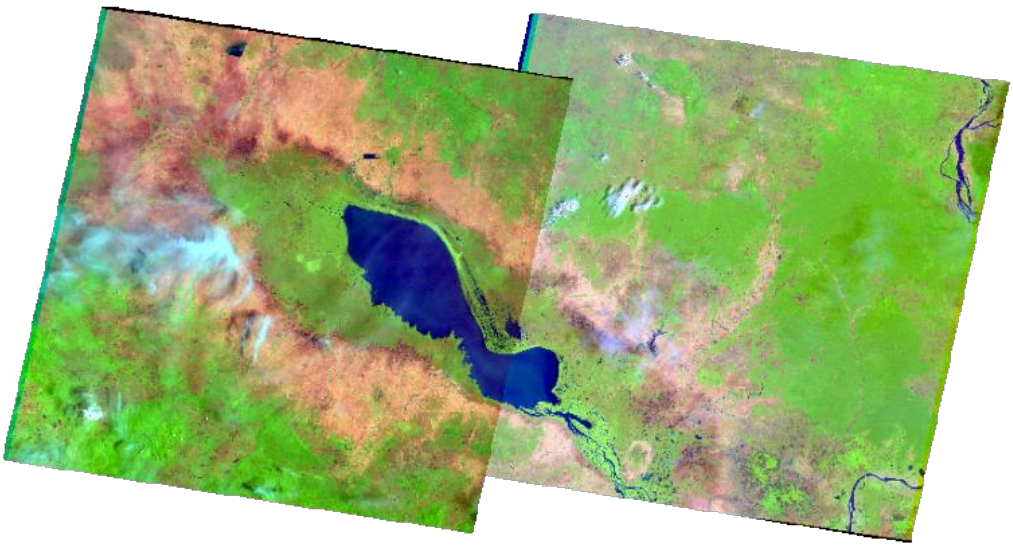

図1-2 湖の季節的変化




(Kompong Chhnangにて2016/9撮影)

図1-3 トンレサップ河の逆流

乾季



雨季

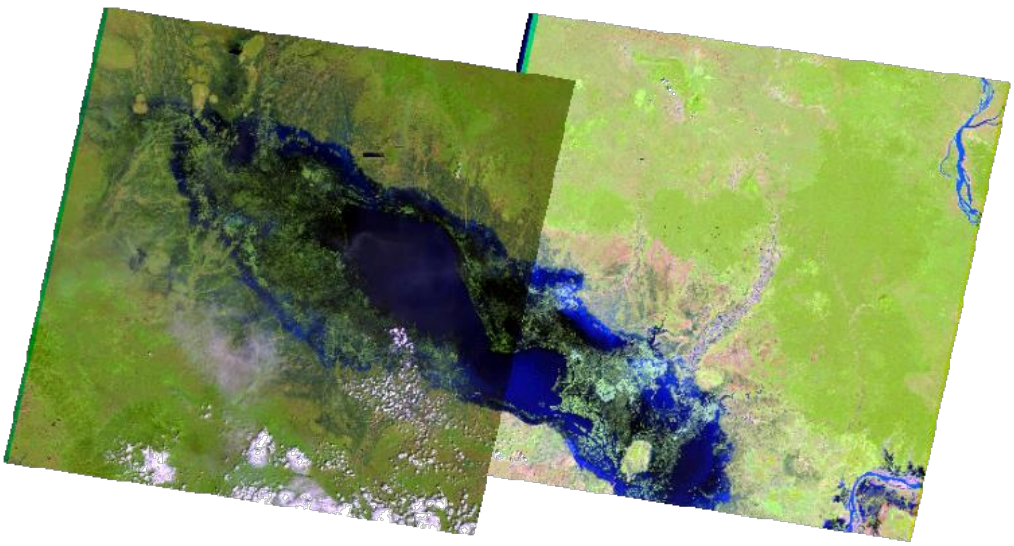


図1-4 乾季･雨季におけるトンレサップ湖のLandsat衛星画像



図1-5 トンレサップ湖での富栄養化[4]

# 第 2 章 流動計算モデル

流動計算モデル構築にあたり，対象領域が広大かつ現象の時間スケールが長期間であることから計算コストが非常に大きくなることが想定される．そこで可能な限り単純化したモデルを用いて計算コストが小さくなるようにした．そのためモデルを水平 2 次元のモデルとし，計算格子はシンプルな直交格子型で格子サイズを固定とした．

本章ではモデルに用いる基礎方程式の導出過程を述べた上で，先述した計算格子に合わせて離散化し，計算精度の確認のためベンチマークテストを行った．

## 2-1. 浅水方程式の導出

本モデルの基礎方程式は Navier-Stokes 方程式と連続の式から導出される[1]．それぞれの式は以下のように表される．

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}=-\frac{1}{\rho}\frac{\partial p}{\partial x} \tag{2-1}$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}=-\frac{1}{\rho}\frac{\partial p}{\partial y} \tag{2-2}$$

$$\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}=-\frac{1}{\rho}\frac{\partial p}{\partial z}-g \tag{2-3}$$

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \tag{2-4}$$

ここに，$u,v,w$は$x,y,z$方向流速，$\rho$は密度，$p$は圧力，$g$は重力加速度である．ここで(2-4)式を用いると(2-1)式は以下のように保存型へ変形できる．

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}+u\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)=-\frac{1}{\rho}\frac{\partial p}{\partial x}$$

$$\frac{\partial u}{\partial t}+\frac{\partial u^2}{\partial x}+\frac{\partial uv}{\partial y}+\frac{\partial uw}{\partial z}=-\frac{1}{\rho}\frac{\partial p}{\partial x} \tag{2-5}$$

(2-2),(2-3)式も同様に，

$$\frac{\partial v}{\partial t}+\frac{\partial uv}{\partial x}+\frac{\partial v^2}{\partial y}+\frac{\partial vw}{\partial z}=-\frac{1}{\rho}\frac{\partial p}{\partial y} \tag{2-6}$$

$$\frac{\partial w}{\partial t}+\frac{\partial uw}{\partial x}+\frac{\partial vw}{\partial y}+\frac{\partial w^2}{\partial z}=-\frac{1}{\rho}\frac{\partial p}{\partial z} \tag{2-7}$$

と変形できる．

一方，(2-4)式を$z=b$(河床面)から$z=\eta$(水面)まで鉛直積分すると，

$$\frac{\partial}{\partial x}\left(\int_b^\eta u dz\right) - u\frac{\partial \eta}{\partial x} + u\frac{\partial b}{\partial x} + \frac{\partial}{\partial y}\left(\int_b^\eta v dz\right) - v\frac{\partial \eta}{\partial y} + v\frac{\partial b}{\partial y} + w(x,y,\eta) - w(x,y,b) = 0 \quad (2\text{-}8)$$

となる．ここで，

$$H = \eta - b, \qquad U = \frac{1}{H}\int_b^\eta u dz, \qquad V = \frac{1}{H}\int_b^\eta v dz$$

$$\boldsymbol{u_\eta} = \begin{pmatrix} u_\eta \\ v_\eta \\ w_\eta \end{pmatrix}, \qquad \boldsymbol{u_b} = \begin{pmatrix} u_b \\ v_b \\ w_b \end{pmatrix}$$

$$\boldsymbol{n_\eta} = \begin{pmatrix} -\dfrac{\partial \eta}{\partial x} \\ -\dfrac{\partial \eta}{\partial y} \\ 1 \end{pmatrix}, \qquad \boldsymbol{n_b} = \begin{pmatrix} -\dfrac{\partial b}{\partial x} \\ -\dfrac{\partial b}{\partial y} \\ 1 \end{pmatrix}$$

($H$: 水深，$U$,V: 水深平均流速，$\boldsymbol{u}$: 流速ベクトル，$\boldsymbol{n}$: 法線ベクトル)

とすると(2-8)式は

$$\frac{\partial HU}{\partial x} + \frac{\partial HV}{\partial y} + \boldsymbol{u_\eta}\cdot\boldsymbol{n_\eta} - \boldsymbol{u_b}\cdot\boldsymbol{n_b} = 0 \quad (2\text{-}9)$$

と表され，更に水面での動力学条件($\boldsymbol{u_\eta}\cdot\boldsymbol{n_\eta} = \frac{\partial \eta}{\partial t} = \frac{\partial H}{\partial t}$)，河床面での湧き出しなしの条件($\boldsymbol{u_b}\cdot\boldsymbol{n_b} = 0$)から

$$\frac{\partial H}{\partial t} + \frac{\partial HU}{\partial x} + \frac{\partial HV}{\partial y} = 0 \quad (2\text{-}10)$$

が得られる．

また(2-5)~(2-7)式に対しても同様に鉛直積分すると，

$$\frac{\partial}{\partial t}\left(\int_b^\eta u dz\right) + \frac{\partial}{\partial x}\left(\int_b^\eta u^2 dz\right) + \frac{\partial}{\partial y}\left(\int_b^\eta uv dz\right) = -\frac{1}{\rho}\frac{\partial}{\partial x}\left(\int_b^\eta p dz\right) + \frac{1}{\rho}p_\eta\frac{\partial \eta}{\partial x} - \frac{1}{\rho}p_b\frac{\partial b}{\partial x} \quad (2\text{-}11)$$

$$\frac{\partial}{\partial t}\left(\int_b^\eta v dz\right) + \frac{\partial}{\partial x}\left(\int_b^\eta uv dz\right) + \frac{\partial}{\partial y}\left(\int_b^\eta v^2 dz\right) = -\frac{1}{\rho}\frac{\partial}{\partial y}\left(\int_b^\eta p dz\right) + \frac{1}{\rho}p_\eta\frac{\partial \eta}{\partial y} - \frac{1}{\rho}p_b\frac{\partial b}{\partial y} \quad (2\text{-}12)$$

$$\frac{\partial}{\partial t}\left(\int_b^\eta w dz\right) + \frac{\partial}{\partial x}\left(\int_b^\eta uw dz\right) + \frac{\partial}{\partial y}\left(\int_b^\eta vw dz\right) = -\frac{1}{\rho}(p_\eta - p_b) - gH \quad (2\text{-}13)$$

ここで圧力を静水圧($p = \rho g(\eta - z)$)とすると，(2-11)~(2-13)式は

$$\frac{\partial HU}{\partial t} + \frac{\partial}{\partial x}\left(\int_b^\eta u^2 dz\right) + \frac{\partial}{\partial y}\left(\int_b^\eta uv dz\right) = -g\frac{\partial}{\partial x}\left(\frac{1}{2}H^2\right) - gH\frac{\partial b}{\partial x} \quad (2\text{-}14)$$

$$\frac{\partial HV}{\partial t} + \frac{\partial}{\partial x}\left(\int_b^\eta uv dz\right) + \frac{\partial}{\partial y}\left(\int_b^\eta v^2 dz\right) = -g\frac{\partial}{\partial y}\left(\frac{1}{2}H^2\right) - gH\frac{\partial b}{\partial y} \quad (2\text{-}15)$$

$$\frac{\partial HW}{\partial t} + \frac{\partial}{\partial x}\left(\int_b^\eta uw dz\right) + \frac{\partial}{\partial y}\left(\int_b^\eta vw dz\right) = 0 \quad (2\text{-}16)$$

今流速を平均成分とその偏差に分離すれば，

$$\frac{\partial}{\partial x}\left(\int_b^\eta uvdz\right)=\frac{\partial}{\partial x}\left(\int_b^\eta (U+u')(V+v')dz\right)$$
$$=\frac{\partial}{\partial x}(HUV)+\frac{\partial}{\partial x}\left(U\int_b^\eta v'dz+V\int_b^\eta u'dz+\int_b^\eta u'v'dz\right)$$
$$\cong\frac{\partial}{\partial x}(HUV)$$

よって，(2-14)~(2-16)式は

$$\frac{\partial HU}{\partial t}+\frac{\partial}{\partial x}(HU^2)+\frac{\partial}{\partial y}(HUV)=-g\frac{\partial}{\partial x}\left(\frac{1}{2}H^2\right)-gH\frac{\partial b}{\partial x} \quad (2\text{-}17)$$

$$\frac{\partial HV}{\partial t}+\frac{\partial}{\partial x}(HUV)+\frac{\partial}{\partial y}(HV^2)=-g\frac{\partial}{\partial y}\left(\frac{1}{2}H^2\right)-gH\frac{\partial b}{\partial y} \quad (2\text{-}18)$$

$$\frac{\partial HW}{\partial t}+\frac{\partial}{\partial x}(HUW)+\frac{\partial}{\partial y}(HVW)=0 \quad (2\text{-}19)$$

以上の(2-10)，(2-17)，(2-18)式によって H,U,V について方程式が閉じる．ここで改めて H,U,V を h,u,v と置き換えれば，水平 2 次元の浅水方程式は(2-10)，(2-17)，(2-18)式より以下のように表される．

$$\frac{\partial}{\partial t}\boldsymbol{w}+\frac{\partial}{\partial x}\boldsymbol{F}+\frac{\partial}{\partial y}\boldsymbol{G}=\boldsymbol{S} \quad (2\text{-}20)$$

$$\boldsymbol{w}=\begin{pmatrix}w_1\\w_2\\w_3\end{pmatrix}=\begin{pmatrix}h\\hu\\hv\end{pmatrix}$$

$$\boldsymbol{F}=\begin{pmatrix}hu\\hu^2+\frac{1}{2}gh^2\\huv\end{pmatrix}=\begin{pmatrix}w_2\\{w_2}^2+\frac{1}{2}g{w_1}^2\\w_2w_3/w_1\end{pmatrix},\qquad \boldsymbol{G}=\begin{pmatrix}hv\\huv\\hv^2+\frac{1}{2}gh^2\end{pmatrix}=\begin{pmatrix}w_3\\w_2w_3/w_1\\{w_3}^2+\frac{1}{2}g{w_1}^2\end{pmatrix}$$

$$\boldsymbol{S}=\begin{pmatrix}0\\-gh\frac{\partial b}{\partial x}+\tau_x\\-gh\frac{\partial b}{\partial y}+\tau_y\end{pmatrix}$$

ここに$\boldsymbol{w}$は h,u,v からなる保存量ベクトル，$\boldsymbol{F}$，$\boldsymbol{G}$は流束，$\boldsymbol{S}$はソース項，$b$は河床高，$\tau_x$，$\tau_y$は外力である(**図 2-1**).

## 2-2. 離散化手法

先に述べたように計算格子を直交格子型としたモデル化を行う．質量保存性を考慮するため，離散化手法は有限体積法とした．

### 2-2-1. 時間発展式の導出

(2-20)式における流束$\boldsymbol{F}$，$\boldsymbol{G}$の Jacobian 行列$\boldsymbol{J}_x$，$\boldsymbol{J}_y$は以下のように表される．

$$\boldsymbol{J}_x=\frac{\partial(F_1,F_2,F_3)}{\partial(w_1,w_2,w_3)}=\begin{pmatrix}0 & 1 & 0\\ -\frac{{w_2}^2}{{w_1}^2}+gw_1 & \frac{2w_2}{w_1} & 0\\ -\frac{w_2w_3}{{w_1}^2} & \frac{w_3}{w_1} & \frac{w_2}{w_1}\end{pmatrix}=\begin{pmatrix}0 & 1 & 0\\ -u^2+gh & 2u & 0\\ -uv & v & u\end{pmatrix} \tag{2-21}$$

$$\boldsymbol{J}_y=\frac{\partial(G_1,G_2,G_3)}{\partial(w_1,w_2,w_3)}=\begin{pmatrix}0 & 0 & 1\\ -\frac{w_2w_3}{{w_1}^2} & \frac{w_3}{w_1} & \frac{w_2}{w_1}\\ -\frac{{w_3}^2}{{w_1}^2}+gw_1 & 0 & \frac{2w_3}{w_1}\end{pmatrix}=\begin{pmatrix}0 & 0 & 1\\ -uv & v & u\\ -v^2+gh & 0 & 2v\end{pmatrix} \tag{2-22}$$

$\boldsymbol{J}_x$の固有値$\lambda$，固有ベクトル$\boldsymbol{L}$は$\boldsymbol{J}_x\boldsymbol{L}=\lambda\boldsymbol{L}$より

$$\lambda_1=u,\lambda_2=u+c,\lambda_3=u-c \tag{2-23}$$

$$\boldsymbol{L}_1=\begin{pmatrix}0\\0\\1\end{pmatrix},\boldsymbol{L}_2=\begin{pmatrix}1\\u+c\\v\end{pmatrix},\boldsymbol{L}_3=\begin{pmatrix}1\\u-c\\v\end{pmatrix} \tag{2-24}$$

となる．ここで，$c(=\sqrt{gh})$は波速である．Jacobian 行列$\boldsymbol{J}_x$と固有行列$\boldsymbol{L}=(\boldsymbol{L}_1,\boldsymbol{L}_2,\boldsymbol{L}_3)$を用いれば，

$$\boldsymbol{L^{-1}J}_x\boldsymbol{L}=\boldsymbol{L^{-1}}(\lambda_1\boldsymbol{L}_1,\lambda_2\boldsymbol{L}_2,\lambda_3\boldsymbol{L}_3)=\begin{pmatrix}\lambda_1 & 0 & 0\\ 0 & \lambda_2 & 0\\ 0 & 0 & \lambda_3\end{pmatrix}=\boldsymbol{\Lambda} \tag{2-25}$$

の関係が得られる．ここに$\boldsymbol{\Lambda}$は特性速度である．この関係から，

$$\boldsymbol{J}_x=\boldsymbol{L\Lambda L^{-1}} \tag{2-26}$$

が得られる．$\boldsymbol{J}_x$，$\boldsymbol{J}_y$を用いれば，(2-20)式は以下のように表される．

$$\frac{\partial}{\partial t}\boldsymbol{w}+\boldsymbol{J}_x\frac{\partial}{\partial x}\boldsymbol{w}+\boldsymbol{J}_y\frac{\partial}{\partial y}\boldsymbol{w}=\boldsymbol{S} \tag{2-27}$$

Euler の同次定理により

$$\frac{\partial}{\partial t}\boldsymbol{w}+\frac{\partial}{\partial x}(\boldsymbol{J}_x\boldsymbol{w})+\frac{\partial}{\partial y}(\boldsymbol{J}_y\boldsymbol{w})=\boldsymbol{S} \tag{2-28}$$

となり，(2-20)式と合わせて

$$\boldsymbol{F}=\boldsymbol{J}_x\boldsymbol{w},\qquad \boldsymbol{G}=\boldsymbol{J}_y\boldsymbol{w} \tag{2-29}$$

が言える．

ここで N 個に等分割された 1 次元の計算領域を考える．分割された小領域をセルとし，セル i(i=1,2,…,N)の範囲を$[x_{i-1/2},x_{i+1/2}]$,中心点を$x_i$とおく．このセル i において，浅水方程式(2-20)の x 方向の伝播は流束差を用いて

$$\boldsymbol{w}_i^{n+1}=\boldsymbol{w}_i^n+\frac{\Delta t}{\Delta x}(\boldsymbol{F}_{i-1/2}-\boldsymbol{F}_{i+1/2}) \tag{2-30}$$

で表すことができる．ここに$n$は時間ステップ，$i$はセルのインデックス，$\Delta x$はセルの幅，$\Delta t$は時間ステップ幅，$\boldsymbol{F}_{i+1/2}$はセル境界$x_{i+1/2}$における数値流束である．この数値流束について(2-26)，(2-29)式より

$$\boldsymbol{F}_{i+1/2} = (\boldsymbol{J}_x \boldsymbol{w})_{i+1/2} = (\boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^{-1}\boldsymbol{w})_{i+1/2} \tag{2-31}$$

となる．またこの式は$|\boldsymbol{\Lambda}| = \begin{pmatrix} |\lambda_1| & 0 & 0 \\ 0 & |\lambda_2| & 0 \\ 0 & 0 & |\lambda_3| \end{pmatrix}$を用いて以下のように表される．

$$\boldsymbol{F}_{i+1/2} = \left(\boldsymbol{L}\frac{1}{2}(\boldsymbol{\Lambda} + |\boldsymbol{\Lambda}|)\boldsymbol{L}^{-1}\boldsymbol{w} + \boldsymbol{L}\frac{1}{2}(\boldsymbol{\Lambda} - |\boldsymbol{\Lambda}|)\boldsymbol{L}^{-1}\boldsymbol{w}\right)_{i+1/2} \tag{2-32}$$

$\frac{1}{2}(\boldsymbol{\Lambda} + |\boldsymbol{\Lambda}|)$は正の方向に，$\frac{1}{2}(\boldsymbol{\Lambda} - |\boldsymbol{\Lambda}|)$は負の方向に進行する成分を表すので各々の成分について上流化すると，

$$\begin{aligned} \boldsymbol{F}_{i+1/2} &\cong \left(\boldsymbol{L}\frac{1}{2}(\boldsymbol{\Lambda} + |\boldsymbol{\Lambda}|)\boldsymbol{L}^{-1}\boldsymbol{w}^{-} + \boldsymbol{L}\frac{1}{2}(\boldsymbol{\Lambda} - |\boldsymbol{\Lambda}|)\boldsymbol{L}^{-1}\boldsymbol{w}^{+}\right)_{i+1/2} \\ &= \frac{1}{2}(\boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^{-1}\boldsymbol{w}^{-} + \boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^{-1}\boldsymbol{w}^{+})_{i+1/2} + \frac{1}{2}(\boldsymbol{L}|\boldsymbol{\Lambda}|\boldsymbol{L}^{-1}\boldsymbol{w}^{-} - \boldsymbol{L}|\boldsymbol{\Lambda}|\boldsymbol{L}^{-1}\boldsymbol{w}^{+})_{i+1/2} \\ &= \frac{1}{2}\left(\boldsymbol{F}(\boldsymbol{w}^{-}_{i+1/2}) + \boldsymbol{F}(\boldsymbol{w}^{+}_{i+1/2})\right) + \frac{1}{2}\left|\boldsymbol{F}_{i+1/2}\right|\left(\boldsymbol{w}^{-}_{i+1/2} - \boldsymbol{w}^{+}_{i+1/2}\right) \end{aligned} \tag{2-33}$$

となる．ここに$\boldsymbol{w}^{\pm}_{i+1/2}\left(= \lim_{x \to x_{i+1/2}{}^{\pm}} \boldsymbol{w}\right)$はセル境界$x_{i+1/2}$における極限値である．また$\left|\boldsymbol{F}_{i+1/2}\right|$は固有値$\lambda_k (k = 1,2,3)$を用いて

$$\left|\boldsymbol{F}_{i+1/2}\right| = \frac{1}{2\tilde{c}}\begin{pmatrix} -\lambda_3|\lambda_2| + \lambda_2|\lambda_3| & |\lambda_2| - |\lambda_3| & 0 \\ -\lambda_2\lambda_3(|\lambda_2| - |\lambda_3|) & \lambda_2|\lambda_2| - \lambda_3|\lambda_3| & 0 \\ \tilde{v}(-2\tilde{c}|\lambda_1| - \lambda_3|\lambda_2| + \lambda_2|\lambda_3|) & \tilde{v}(|\lambda_2| - |\lambda_3|) & 2\tilde{c}|\lambda_1| \end{pmatrix}_{i+1/2} \tag{2-34}$$

$$\left(\tilde{c} = \sqrt{g\tilde{h}}, \quad \lambda_1 = \tilde{u}, \quad \lambda_2 = \tilde{u} + \tilde{c}, \quad \lambda_3 = \tilde{u} - \tilde{c}\right)$$

で表される．”~”記号は以下のように境界$x_{i+1/2}$における Roe 平均を意味する．

$$\begin{aligned} \tilde{h} &= \frac{1}{2}(h^{+} + h^{-}) \\ \tilde{u} &= \frac{\sqrt{h^{+}}u^{+} + \sqrt{h^{-}}u^{-}}{\sqrt{h^{+}} + \sqrt{h^{-}}} \\ \tilde{v} &= \frac{\sqrt{h^{+}}v^{+} + \sqrt{h^{-}}v^{-}}{\sqrt{h^{+}} + \sqrt{h^{-}}} \end{aligned} \tag{2-35}$$

Roe の近似リーマン解法[2]では物理量がセル内で空間的に一定とし，物理量$f$に関して境界$x_{i+1/2}$の左側における極限値$f^{-}$を$f_i$，右側に於ける極限値$f^{+}$を$f_{i+1}$とする．

以上のように，セル i における時間発展(2-30)式はセル境界面での数値流束(2-33)式の流入出により計算され，保存性が満足されるものとなっている．

### 2-2-2. 河床面勾配の考慮

ここでは時間発展式(2-30)を，以下のように河床勾配と外力を表現するソース項$\boldsymbol{S}(=\boldsymbol{S}_x+\boldsymbol{S}_y)$を含めた形に拡張する．

$$\boldsymbol{w}_i^{n+1}=\boldsymbol{w}_i^n+\frac{\Delta t}{\Delta x}\left(\boldsymbol{F}_{i-1/2}-\boldsymbol{F}_{i+1/2}\right)+\frac{\Delta t}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}}\boldsymbol{S}_x dx \tag{2-36}$$

浅水方程式(2-20)において，”C-property”と呼ばれる静止状態($h=const.,\ u=v=0,\ \tau_x=\tau_y=0$)では以下の関係が得られる[3]．

$$\frac{\partial}{\partial x}\left(\frac{1}{2}gh^2\right)=-gh\frac{\partial b}{\partial x} \tag{2-37}$$

この関係を満たさずに定式化した場合，運動量に物理的に誤った数値振動が生じてしまうため，この関係を離散化レベルで保持できるように定式化する必要がある．そこでソース項を，水位$H(=b+h)$を用いて以下のように変形する．

$$\begin{aligned}\boldsymbol{S}_x&=\begin{pmatrix}0\\-gh\dfrac{\partial b}{\partial x}+\tau_x\\0\end{pmatrix}=\begin{pmatrix}0\\\dfrac{\partial}{\partial x}\left(\dfrac{1}{2}gh^2\right)\\0\end{pmatrix}+\begin{pmatrix}0\\-gh\dfrac{\partial H}{\partial x}+\tau_x\\0\end{pmatrix}\\&=\boldsymbol{S}_x^*+\boldsymbol{S}_x^{Force}\end{aligned} \tag{2-38}$$

この変形により，静止条件では第 2 項$\boldsymbol{S}_x^{Force}$が 0 となり第 1 項$\boldsymbol{S}_x^*$によって(2-36)式を満たすことができる．(2-38)式の第 1 項$\boldsymbol{S}_x^*$をセル内で積分すると，

$$\begin{aligned}&\int_{x_{i-1/2}}^{x_{i+1/2}}\boldsymbol{S}_x^*\,dx=-\boldsymbol{F}_{i+1/2}^*+\boldsymbol{F}_{i-1/2}^*\\&\boldsymbol{F}_{i\pm1/2}^*=\begin{pmatrix}0\\-\dfrac{1}{2}g{h_{i\pm1/2}}^2\\0\end{pmatrix}\end{aligned} \tag{2-39}$$

となり(2-36)式の形から$\boldsymbol{F}_{i\pm1/2}^*$を$x_{i\pm1/2}$における流束と同様に扱うことができる．そこで$\boldsymbol{F}_{i\pm1/2}^*$に関して$\boldsymbol{\Phi}(h)=\begin{pmatrix}0\\-\frac{1}{2}gh^2\\0\end{pmatrix}$として上流化を行うと，

$$\begin{aligned}\boldsymbol{F}_{i+1/2}^*&=(\boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^{-1})^{-1}(\boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^{-1})\boldsymbol{\Phi}\\&=(\boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^{-1})^{-1}\left\{\frac{1}{2}\boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^{-1}(\boldsymbol{\Phi}^+ +\boldsymbol{\Phi}^-)+\frac{1}{2}\boldsymbol{L}|\boldsymbol{\Lambda}|\boldsymbol{L}^{-1}(\boldsymbol{\Phi}^- -\boldsymbol{\Phi}^+)\right\}\\&=\frac{1}{2}(\boldsymbol{\Phi}^+ +\boldsymbol{\Phi}^-)+\frac{1}{2}sgn\boldsymbol{F}(\boldsymbol{\Phi}^- -\boldsymbol{\Phi}^+)\end{aligned} \tag{2-40}$$

となる．ここに，

$$sgn\boldsymbol{F} = \boldsymbol{L}\boldsymbol{\Lambda}^{-1}|\boldsymbol{\Lambda}|\boldsymbol{L}^{-1}$$
$$= \frac{1}{2\tilde{c}}\begin{pmatrix} -\lambda_3 sgn(\lambda_2) + \lambda_2 sgn(\lambda_3) & sgn(\lambda_2) - sgn(\lambda_3) & 0 \\ -\lambda_2\lambda_3(sgn(\lambda_2) - sgn(\lambda_3)) & \lambda_2 sgn(\lambda_2) - \lambda_3 sgn(\lambda_3) & 0 \\ \tilde{v}(-2\tilde{c}(sgn(\lambda_1) - \lambda_3 sgn(\lambda_2) + \lambda_2 sgn(\lambda_3)) & \tilde{v}(sgn(\lambda_2) - sgn(\lambda_3)) & 2\tilde{c}sgn(\lambda_1) \end{pmatrix} \quad (2\text{-}41)$$

である．(2-33)式と(2-40)式を合わせて$x_{i+1/2}$における流束は，

$$\boldsymbol{F}_{i+1/2} + \boldsymbol{F}^*_{i+1/2} \cong \frac{1}{2}\left(\boldsymbol{F}(\boldsymbol{w}^-_{i+1/2}) + \boldsymbol{F}(\boldsymbol{w}^+_{i+1/2})\right) + \frac{1}{2}\left|\boldsymbol{F}_{i+1/2}\right|\left(\boldsymbol{w}^-_{i+1/2} - \boldsymbol{w}^+_{i+1/2}\right)$$
$$+ \frac{1}{2}(\boldsymbol{\Phi}^+ + \boldsymbol{\Phi}^-) + \frac{1}{2}sgn\boldsymbol{F}(\boldsymbol{\Phi}^- - \boldsymbol{\Phi}^+) \quad (2\text{-}42)$$
$$= \frac{1}{2}\left(\widehat{\boldsymbol{F}}^+ + \widehat{\boldsymbol{F}}^-\right) + \frac{1}{2}\left|\boldsymbol{F}_{i+1/2}\right|\left(\boldsymbol{w}^-_{i+1/2} - \boldsymbol{w}^+_{i+1/2}\right) + \frac{1}{2}sgn\boldsymbol{F}(\boldsymbol{\Phi}^- - \boldsymbol{\Phi}^+)$$

となる．ここで，

$$\widehat{\boldsymbol{F}}^{\pm} = \boldsymbol{F}(\boldsymbol{w}^{\pm}_{i+1/2}) + \boldsymbol{\Phi}^{\pm} = \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{pmatrix}_{i+1/2^{\pm}} + \begin{pmatrix} 0 \\ -\frac{1}{2}gh^2 \\ 0 \end{pmatrix}_{i+1/2^{\pm}} = \begin{pmatrix} hu \\ hu^2 \\ huv \end{pmatrix}_{i+1/2^{\pm}} \quad (2\text{-}43)$$

である．

(2-38)式の第 2 項$\boldsymbol{S}_x^{Force}$についても，同様にセル内で積分･上流化すると

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \boldsymbol{S}_x^{Force}\,dx = \int_{x_{i-1/2}}^{x_{i+1/2}} (\boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^{-1})^{-1}(\boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^{-1})\boldsymbol{S}_x^{Force}\,dx$$
$$= \int_{x_{i-1/2}}^{x_{i+1/2}} (\boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^{-1})^{-1}\left\{\frac{1}{2}\boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{L}^{-1}[(\boldsymbol{S}_x^{Force})^- + (\boldsymbol{S}_x^{Force})^+] + \frac{1}{2}\boldsymbol{L}|\boldsymbol{\Lambda}|\boldsymbol{L}^{-1}[(\boldsymbol{S}_x^{Force})^- - (\boldsymbol{S}_x^{Force})^+]\right\}dx \quad (2\text{-}44)$$
$$= \frac{1}{2}\int_{x_{i-1/2}}^{x_{i+1/2}} \{[(\boldsymbol{S}_x^{Force})^- + (\boldsymbol{S}_x^{Force})^+] + sgn\boldsymbol{F}[(\boldsymbol{S}_x^{Force})^- - (\boldsymbol{S}_x^{Force})^+]\}dx$$
$$= \frac{\Delta x}{2}(\boldsymbol{S}_x^{Force})_{i-1/2} + \frac{\Delta x}{2}(\boldsymbol{S}_x^{Force})_{i+1/2} + \frac{\Delta x}{2}(sgn\boldsymbol{F})_{i-1/2}(\boldsymbol{S}_x^{Force})_{i-1/2} - \frac{\Delta x}{2}(sgn\boldsymbol{F})_{i+1/2}(\boldsymbol{S}_x^{Force})_{i+1/2}$$

となる．

以上をまとめると，時間発展式は

$$\boldsymbol{w}_i^{n+1} = \boldsymbol{w}_i^n + \frac{\Delta t}{\Delta x}\left(\boldsymbol{F}_{i-1/2} - \boldsymbol{F}_{i+1/2}\right) + \frac{\Delta t}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \boldsymbol{S}_x^{Force}dx \quad (2\text{-}45)$$
$$\boldsymbol{F}_{i+1/2} = \frac{1}{2}\left(\widehat{\boldsymbol{F}}^+ + \widehat{\boldsymbol{F}}^-\right) + \frac{1}{2}\left|\boldsymbol{F}_{i+1/2}\right|\left(\boldsymbol{w}^-_{i+1/2} - \boldsymbol{w}^+_{i+1/2}\right) + \frac{1}{2}sgn\boldsymbol{F}(\boldsymbol{\Phi}^- - \boldsymbol{\Phi}^+)$$
$$\int_{x_{i-1/2}}^{x_{i+1/2}} \boldsymbol{S}_x^{Force}\,dx = \frac{\Delta x}{2}(\boldsymbol{S}_x^{Force})_{i-1/2} + \frac{\Delta x}{2}(\boldsymbol{S}_x^{Force})_{i+1/2} + \frac{\Delta x}{2}(sgn\boldsymbol{F})_{i-1/2}(\boldsymbol{S}_x^{Force})_{i-1/2} - \frac{\Delta x}{2}(sgn\boldsymbol{F})_{i+1/2}(\boldsymbol{S}_x^{Force})_{i+1/2}$$

で表される．またここまでは x 方向についてのみ述べたが，y 方向についても同様に処理を行う．

### 2-2-3. 外力項の付与

流動計算モデル開発にあたり必要と考えられる外力項を，コリオリ力，底面摩擦力，レイノルズ応力とした．ここではそれぞれの外力項の定式化について述べる．

- コリオリ力の導入

コリオリ力を考慮した外力項は以下のように表される．

$$\tau_{c_x} = f_c w_3, \qquad \tau_{c_y} = -f_c w_2, \tag{2-46}$$

ここにコリオリ力$f_c$は地球の自転角速度$\omega = 7.292 \times 10^{-5}$[rad/sec]，緯度$\phi$を用いて，

$$f_c = 2\omega sin\phi \tag{2-47}$$

で表される．定式化において，(2-46)式中において保存量$\boldsymbol{w}$の導関数は含まれないため，特別な処理をせず(2-38)式における$\boldsymbol{S}_x^{Force}$の外力項として取り扱う事ができる．

- 底面摩擦項の導入

底面摩擦を考慮した外力項は以下のように表される．

$$\tau_{f_x} = -\frac{gn^2}{h^{7/3}} w_2 \sqrt{{w_2}^2 + {w_3}^2}, \qquad \tau_{f_y} = -\frac{gn^2}{h^{7/3}} w_3 \sqrt{{w_2}^2 + {w_3}^2}, \tag{2-48}$$

ここに，$n$は Manning 係数である．この係数は地形状態によって異なるが本モデルでは自然水路の標準値[1]として$n = 0.03$とした．

定式化においてコリオリ力と同様に$\boldsymbol{S}_x^{Force}$の外力項として取り扱うと，(2-20)式から$\partial \boldsymbol{w} / \partial t$との間に次の関係が成り立つ．(ここでは$w_2$のみについて考える)

$$\frac{\partial w_2}{\partial t} = \tau_{f_x} = -\alpha w_2 \tag{2-49}$$

$$\left( \alpha = \frac{gn^2}{h^{7/3}} \sqrt{{w_2}^2 + {w_3}^2} > 0 \right)$$

この関係式において$\alpha$を定数とみなすとその理論解は$w_2(t) = w_2(0)e^{-\alpha t}$となる．これまでの手法は陽解法的に離散化をするため，時間ステップ n と n+1 の間に(2-49)式は以下のように離散化される．

$$\frac{{w_2}^{n+1} - {w_2}^n}{\Delta t} = -\alpha {w_2}^n \tag{2-50}$$

$${w_2}^{n+1} = {w_2}^n (1 - \alpha \Delta t)$$

このように離散化してしまうと，$\alpha \Delta t > 1$となったときに${w_2}^{n+1}$の符号が${w_2}^n$と逆になり，数値発散が生じてしまう．そこで陰解法で離散化すると，

$$\frac{{w_2}^{n+1} - {w_2}^n}{\Delta t} = -\alpha {w_2}^{n+1} \tag{2-51}$$

$${w_2}^{n+1} = {w_2}^n / (1 + \alpha \Delta t)$$

となり，先ほどの数値誤差を防ぐことができる．そのためモデル実装においてはこれまでの時間発展とは別に底面摩擦項の計算フェーズを設ける必要がある．

- レイノルズ応力

  レイノルズ応力は以下のように表される．

$$\tau_{s_x} = \frac{\partial}{\partial x}\left(\nu h \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu h \frac{\partial u}{\partial y}\right), \qquad \frac{\partial}{\partial x}\left(\nu h \frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial x}\left(\nu h \frac{\partial v}{\partial y}\right) \tag{2-52}$$

ここに$\nu$は渦動粘性係数で Richardson の 4/3 乗則[4]に従い以下の式で表される．

$$\nu = 0.01\Delta x^{4/3}[\mathrm{cm}^2/\mathrm{s}] \tag{2-53}$$

定式化においては，拡散形であるため上流化は行わず

$$\begin{aligned} \frac{\partial}{\partial x}\left(\nu h \frac{\partial u}{\partial x}\right)\Big|_{x=x_i} &= \frac{1}{\Delta x}\left\{\left(\nu h \frac{\partial u}{\partial x}\right)_{i+1/2} - \left(\nu h \frac{\partial u}{\partial x}\right)_{i-1/2}\right\} \\ &= \frac{\nu}{\Delta x^2}\{\tilde{h}_{i+1/2}(u_{i+1} - u_i) - \tilde{h}_{i-1/2}(u_i - u_{i-1})\} \end{aligned} \tag{2-54}$$

とする．

## 2-3. 水域･陸域判定

本モデルのように氾濫･遡上現象を取り扱う場合，水陸境界が移動し陸域セルが水域セルに変化，あるいはその逆となるケースを表現しなければならないため，計算において各セルは水域あるいは陸域であると判定する必要がある．そこで水深の閾値として限界水深$h_\varepsilon$を定め，以下の式のようにセルの水深$h$が限界水深を下回るときは陸域セル，それ以上となるときは水域セルとして扱う．

氾濫過程においては対象陸域セルの 4 つの隣接セルのうち 1 つでも「水域セルであり，水位 H が対象セルの底面高 z を上回っている」場合は対象セルを新たに水域とし水深$h_{i,j} = h_\varepsilon$を与える．なお保存性を満足するため，水域化した場合その増加分$(h_\varepsilon \Delta x \Delta y)$の水深を隣接水域セルから差し引く．**図 2-2** にその表現を図示する．図の場合，対象の陸域セル(i,j)の 4 隣接セルの内，セル(i,j-1)とセル(i+1,j)が水域となっており，セル(i+1,j)の水位が対象セルの底面高を上回っている$(H_{i+1.j} > z_{i,j})$．そのため対象セルが新たに水域となり水深として$h_\varepsilon$を与え，同時にセル(i+1,j)の水深から$h_\varepsilon$を差し引くことによってセル(i+1,j)からの遡上を表現している．また陸地化過程においても対象セルの水深が限界水深を下回った場合$(h_{i,j} < h_\varepsilon)$,そのセルを陸域セルとみなし，流動計算対象から除外する．

## 2-4. ベンチマークテスト計算

以上の構造の流動モデルの計算精度を確認するため，一次元及び二次元の 2 ケースの遡

上問題を扱ったベンチマークテストを行った．なおここでは先述した外力項は無視する物とする．

### 2-4-1. 一次元遡上問題ベンチマークテスト

ここでは Carrier と Greenspan[5]によって提案された厳密解を持つ一様斜面上の 1 次元遡上モデルの検証問題を取り扱う．この問題では一様な傾きの斜面に向かって沿岸部から波を入射させたときの水面形変化を取り扱う．この流れ場の時刻$t$，座標$x$における解析解（流速$u$，水深$h$）はパラメータ$A,\sigma,\lambda$とポテンシャル関数$\Phi(\sigma,\lambda)$を用いて以下のように与えられる．

$$\begin{aligned}\Phi(\sigma,\lambda) &= AJ_0(\sigma)cos(\lambda)\\ x &= l_0\left(\frac{1}{4}\frac{\partial\Phi}{\partial\lambda}-\frac{u^2}{2}-\frac{\sigma^2}{16}\right)\\ t &= \sqrt{\frac{l_0}{g\theta}}\left(\frac{\lambda}{2}-u\right)\\ u &= \sqrt{g\theta l_0}\frac{1}{\sigma}\frac{\partial\Phi}{\partial\sigma}\\ h &= l_0\theta\left(\frac{1}{4}\frac{\partial\Phi}{\partial\lambda}-\frac{u^2}{2}\right)-z_0\end{aligned} \tag{2-55}$$

ここに$J_0(\sigma)$は次数 0 の第 1 種 Bessel 関数，$l_0$は代表長さ，$g$は重力加速度，$\theta$は斜面の勾配，$z_0$は底面高である．

計算条件は工藤[6]を参考に，$l_0 = 100m$，$\theta = 0.01$，入射波の周期$T = 100\pi/\sqrt{g}$と設定した．また(2-57)式より全領域で$u = 0$となる場合の解析解を初期条件として与え，メッシュサイズ$\Delta x = 0.5m$，時間ステップは$\mathrm{CFL} = \frac{u\Delta t}{\Delta x} < 0.1$とし，境界条件として$x = -100m$に波高$H$の解析解を与えた．

**図 2-3** に水位と流速の分布について解析解と計算値の比較を 1/8 周期ずつ示す．水際先端の流速が理論値に比べやや誤差が生じているが，全体的に見ると水位･流速ともに良好な結果が得られている事がわかる．

### 2-4-2. 二次元遡上問題ベンチマークテスト

次に Thacker[7]によって提案された厳密解を持つボウル状地形の 2 次元遡上モデルの懸賞問題を取り扱う．計算領域$[-4000,4000]\times[-4000,4000]$における以下のような最大半径$R_z = 4000$mのボウル状地形を設定する．(**図 2-4**)

$$z(x,y)=z(r)=\begin{cases}-D_0\left(1-\dfrac{r^2}{L^2}\right), & if\ r\le R_z\\ -D_0\left(1-\dfrac{{R_z}^2}{L^2}\right), & otherwise\end{cases} \tag{2-56}$$

ここに$D_0$は原点における水位基準面($H=0m$)から底面までの深さ，$L$は底面高$z_0=0m$となる半径である．この地形は原点からの距離$r(=\sqrt{x^2+y^2})$によって表される原点対称地形である．

初期条件を以下のように与える．

$$H_0(x,y)=H_0(r)=D_0\left[\frac{\sqrt{1-A^2}}{1-A}-1-\frac{r^2}{L^2}\left\{\frac{1-A^2}{(1-A)^2}-1\right\}\right] \tag{2-57}$$

$$u_0(x,y)=v_0(x,y)=0$$

ここにパラメータ$A$は

$$A=\frac{L^4-{R_0}^4}{L^4+{R_0}^4} \tag{2-58}$$

で表される．$R_0$は初期水域の半径である．

この問題の水位$H$の解析解は以下のように表される．

$$H(x,y,t)=H(r,t)=D_0\left[\frac{\sqrt{1-A^2}}{1-Acos\omega t}-1-\frac{r^2}{L^2}\left\{\frac{1-A^2}{(1-Acos\omega t)^2}-1\right\}\right] \tag{2-59}$$

ここにパラメータ$\omega$は

$$\omega=\frac{2}{L}\sqrt{2gD_0} \tag{2-60}$$

で表され，周期$T=2\pi/\omega$の振動する解を与える．

計算条件は赤穂[3]を参考に$D_0=1000m$，$L=2500m$，$R_0=2500m$と設定した．ここではメッシュサイズ$\Delta x=100m$，$50m$，$25m$の 3 ケースで計算を行った．また時間ステップはCFL < 0.1とした．**図 2-5** に各ケースにおける水位分布を$y=0$断面，$x=y$断面について計算結果と解析解の比較を 1/8 周期ずつ示す．各ケースにおいて$y=0$断面と$x=y$断面で同様な水位変動が見られ，セルの配置方向(x，y 方向)だけでなく動径方向(r 方向)にも伝播していることがわかる．また，解析解と同様に計算波形が対照的に広がり，再び縮小する様子が再現されている．しかしメッシュサイズによっては特に原点位置における水位の解析解との差が顕著に現れる．そこで**図 2-6** に原点における水位時間変動を示す．メッシュサイズが小さいケースほど解析解との誤差が大きく，水位振動が早く減衰する．**図 2-7** にピーク時における水位の解析値との誤差$H_{err.}$と$\Delta x$の関係を示す．この図から，誤差はメッシュサイズに対して 1 次のオーダーで生じているがこれは **2-2-1** 節で述べた Roe 近似が 1 次精度の近似であることに起因すると考えられる．

## 第 2 章 参考文献


1. 池田 駿介: 詳述 水理学，技報堂出版，1999
2. P. L. Roe: Approximate Riemann solvers, parameter vectors, and difference schemes, *Journal of Computational Physics*, Vol.43, pp.357-372, 1981
3. 赤穂 良輔: 高精度津波シミュレーションに向けた数値計算手法の研究，東京工業大学大学院 総合理工学研究科博士論文, 2009
4. 有田 正光，中井 正則，道奥 康治，池田 裕一，中村 由行，村上 和男: 水圏の環境，東京電機大学出版局，1998
5. G. F. Carrier and H. P. Greenspan: Water waves of finite amplitude on a sloping beach, *Journal of Fluid Mechanics*, Vol.4, pp.97-109, 1958
6. 工藤 健太郎: 内部境界条件有限要素法による流体運動計算手法の開発，東京工業大学大学院 総合理工学研究科博士論文，2003
7. W. C. Thacker: Some extent solutions to the nonlinear shallow-water wave equations, *Journal of Fluid Mechanics*, Vol.107, pp.499-508, 1981

# 第2章
# 図表集

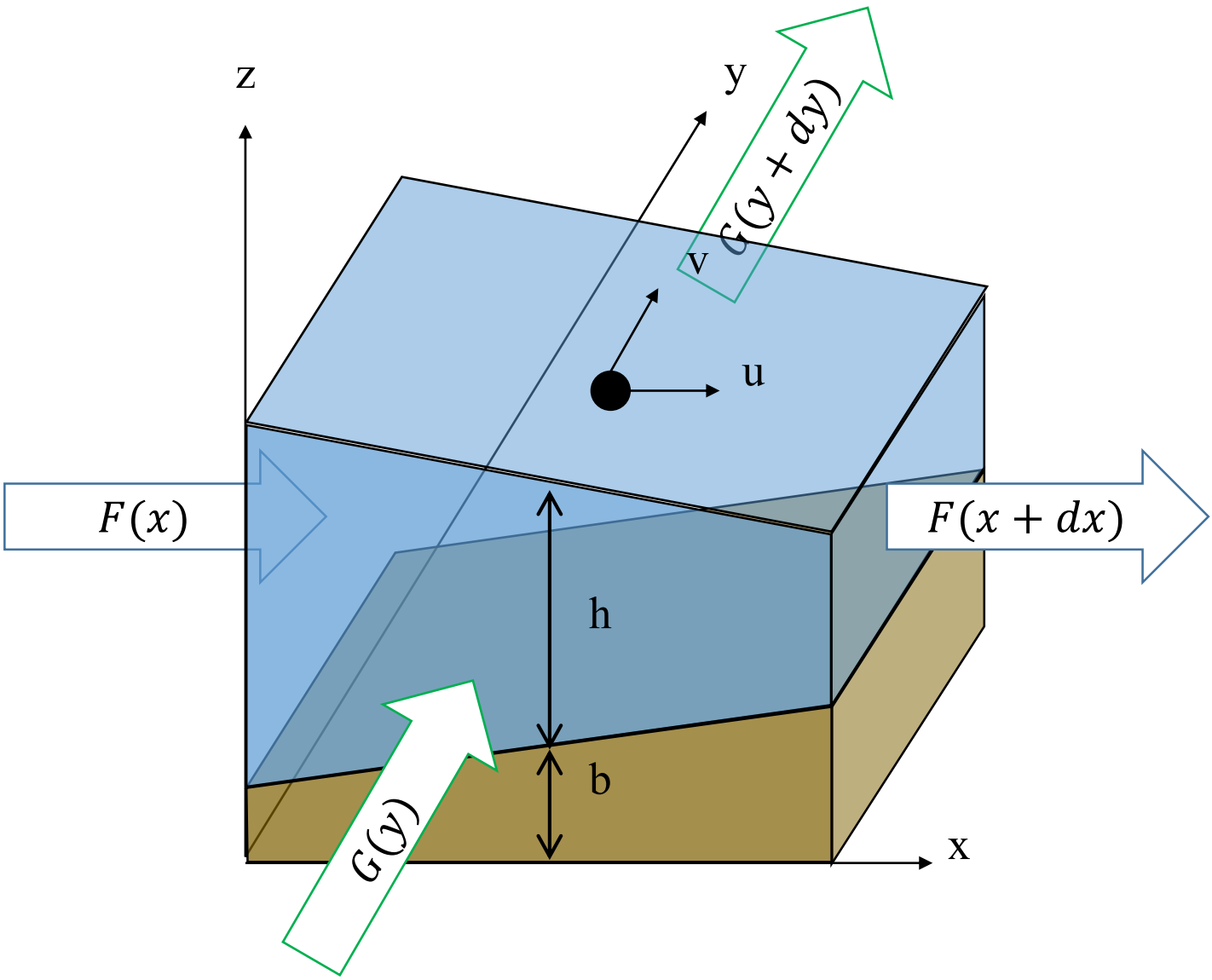



図2-1 浅水方程式における各種変数

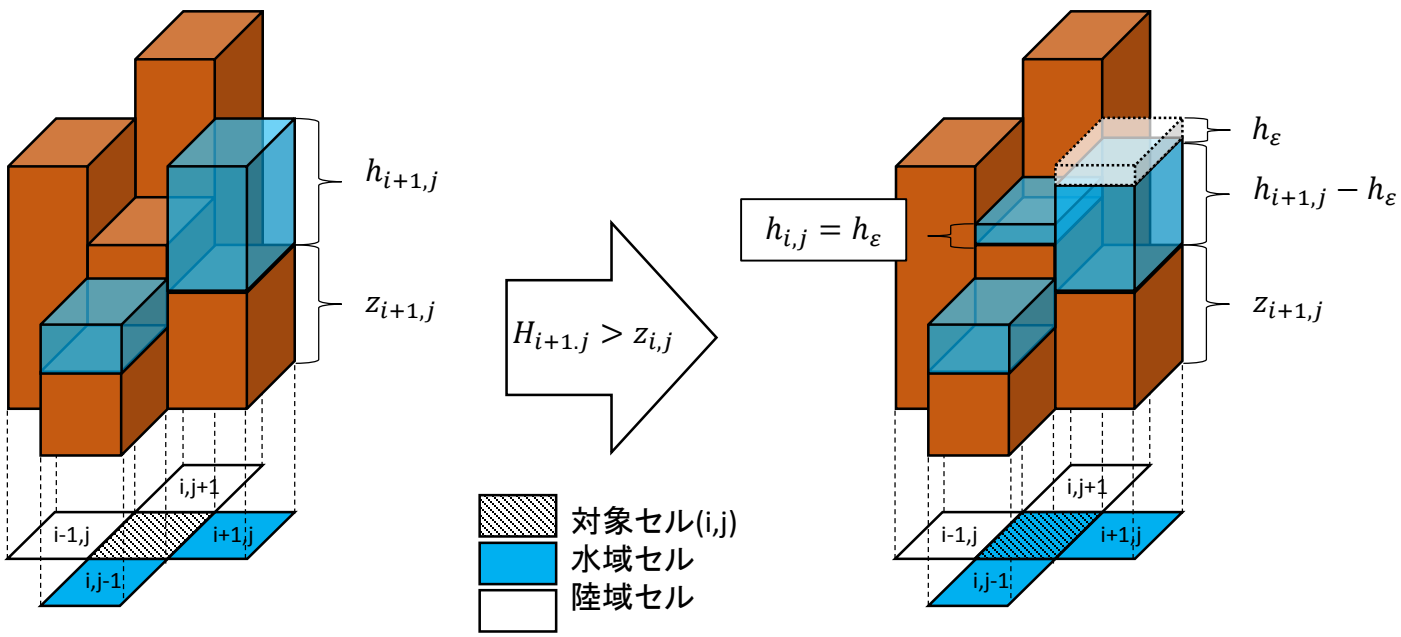
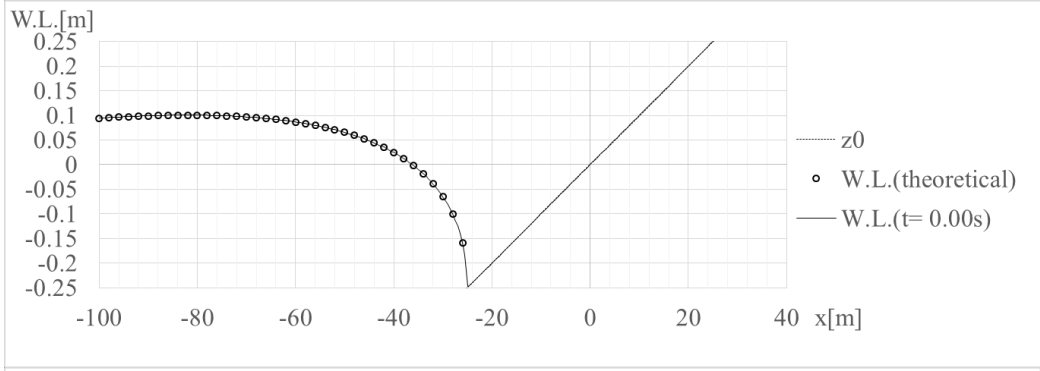
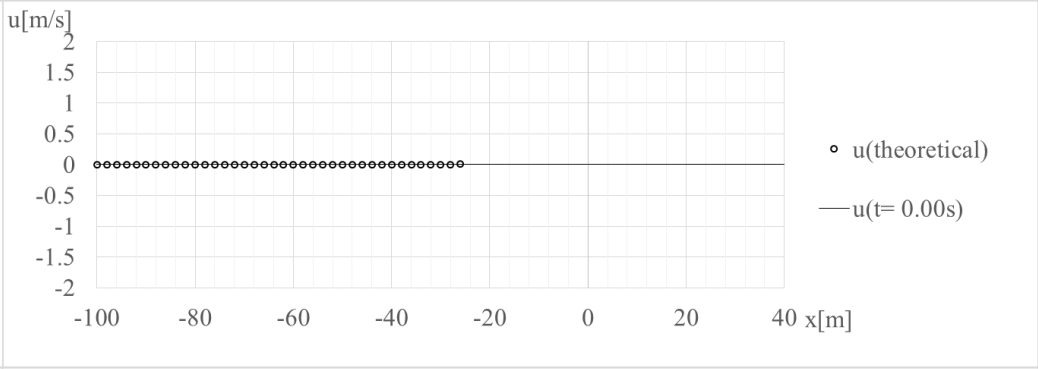
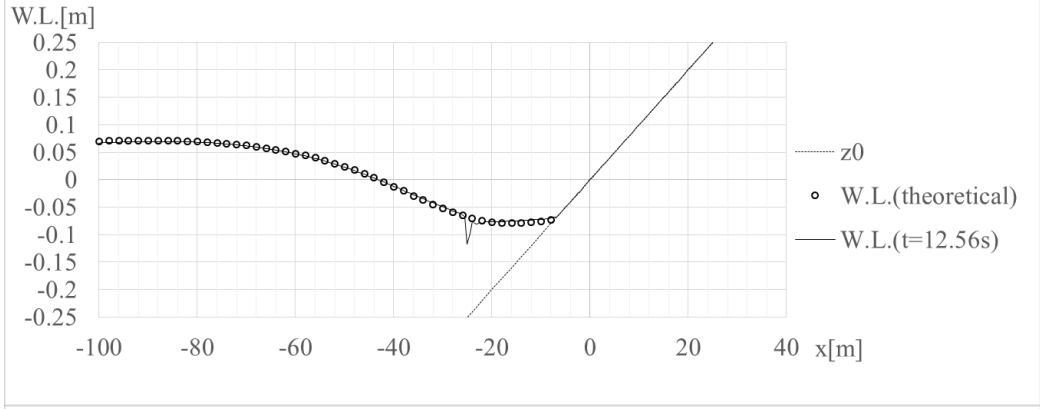
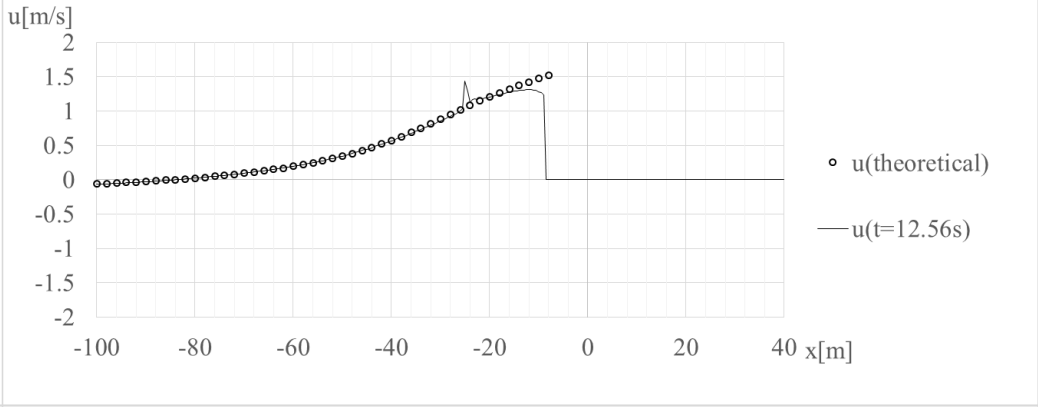

$h_{i+1,j}$
$z_{i+1,j}$
i,j+1
i-1,j
i+1,j
i,j-1
$H_{i+1.j} > z_{i,j}$
対象セル(i,j)
水域セル
陸域セル
$h_{i,j} = h_\varepsilon$
$h_\varepsilon$
$h_{i+1,j} - h_\varepsilon$
$z_{i+1,j}$
i,j+1
i-1,j
i+1,j
i,j-1

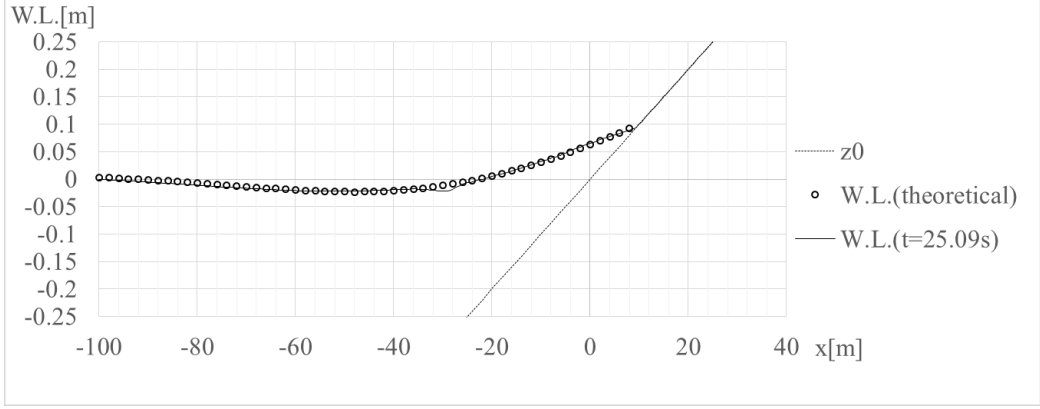
図2-2 氾濫過程の表現



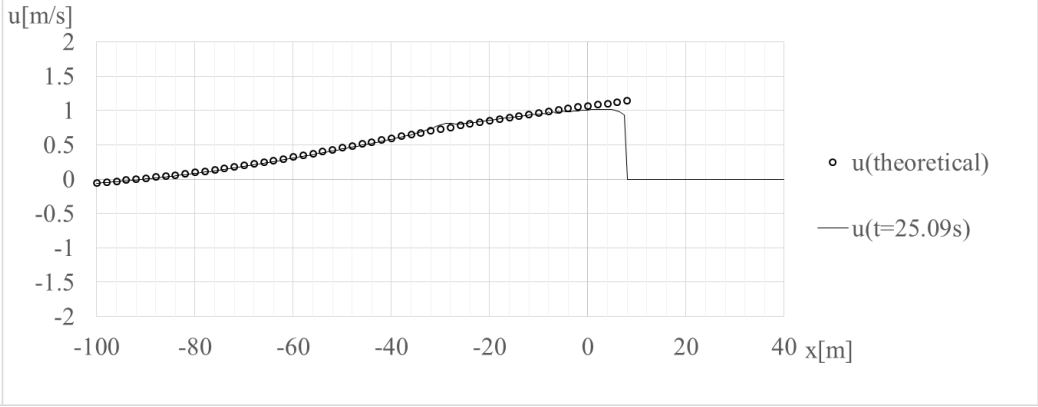
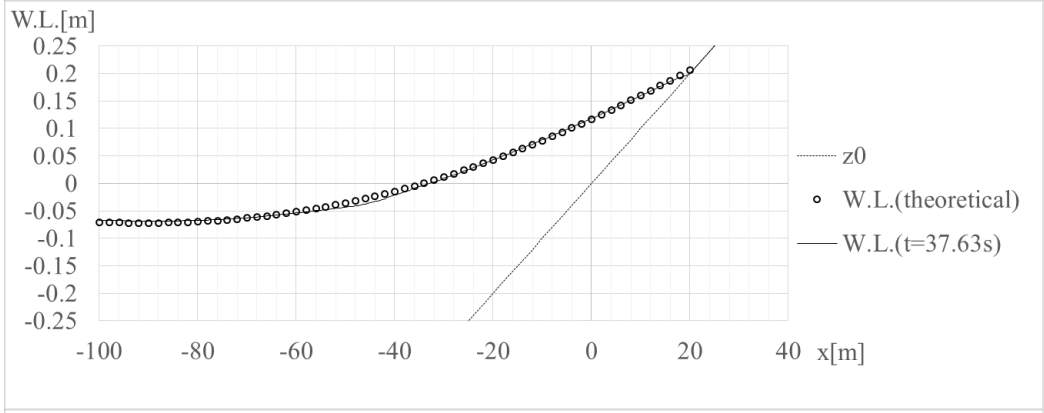
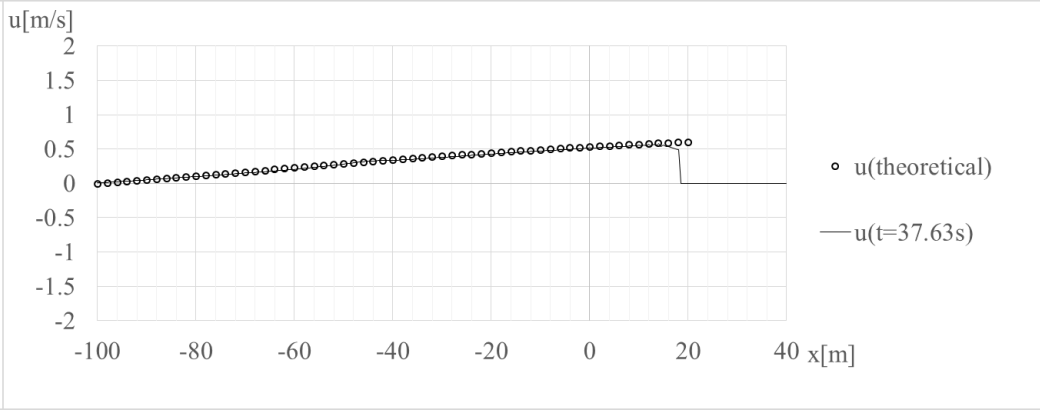
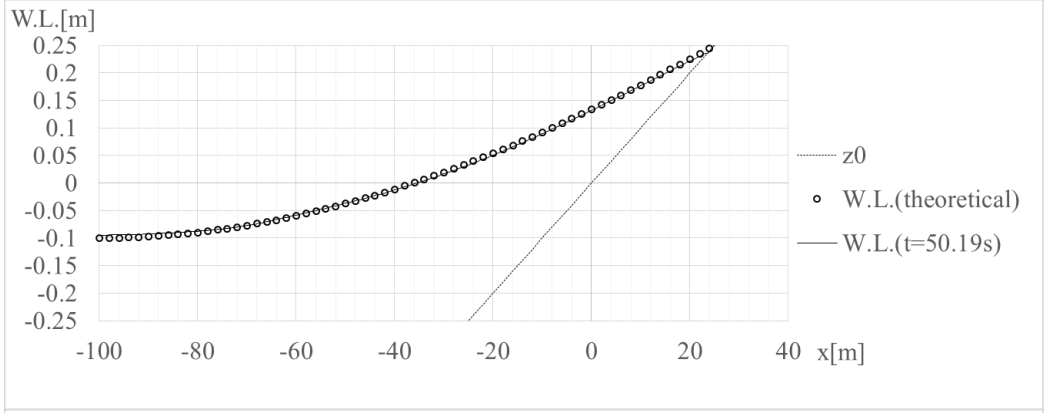
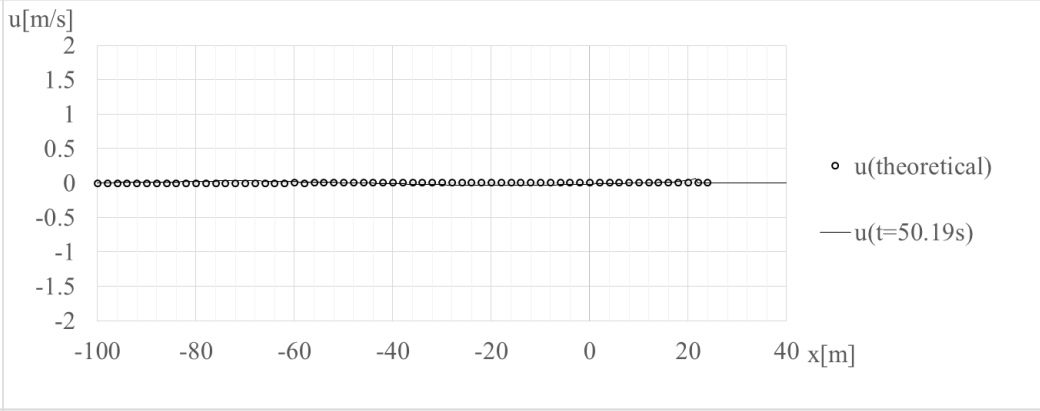



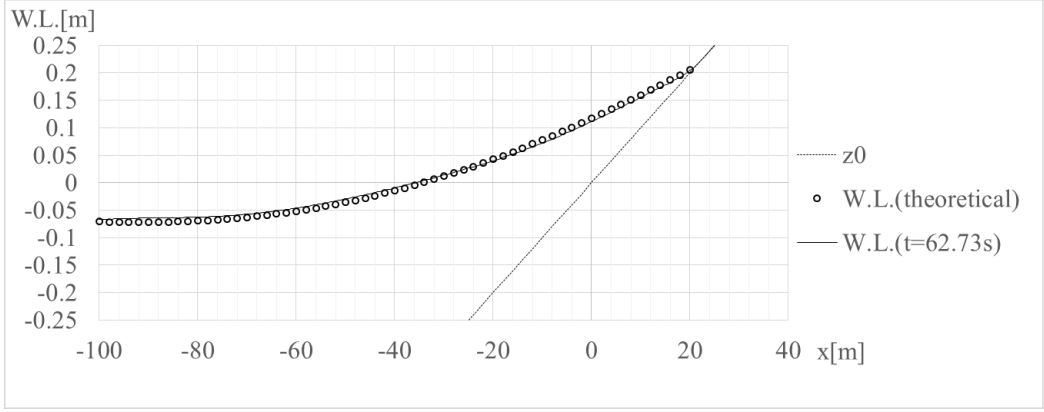

図2-3 一次元遡上計算の水位･流速の分布



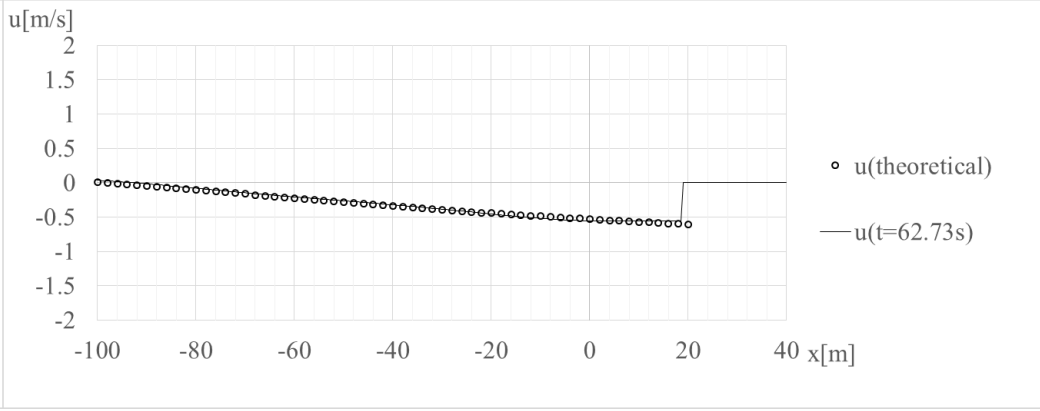
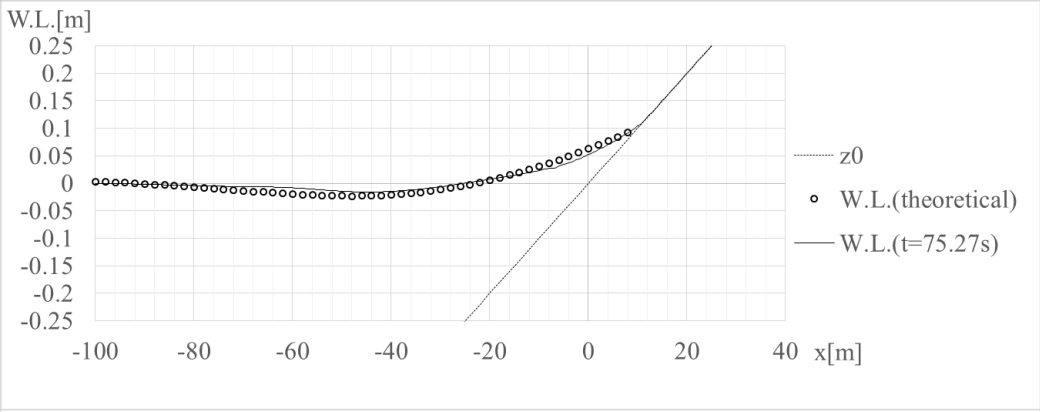
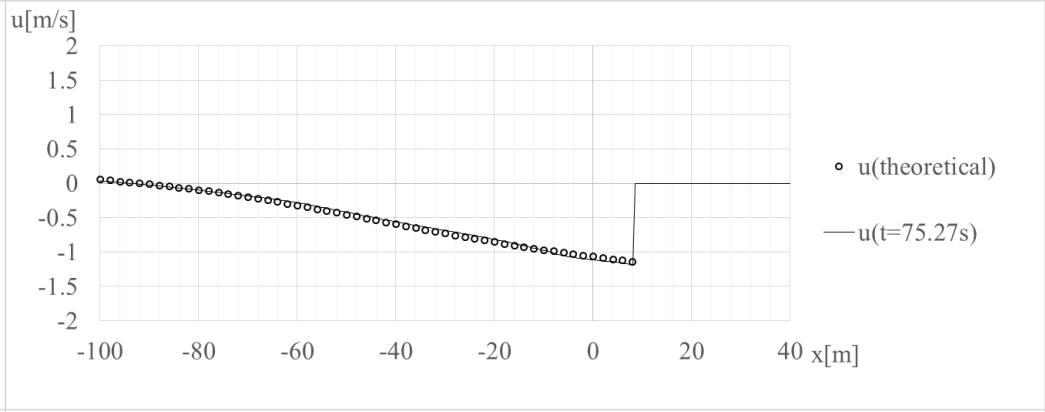
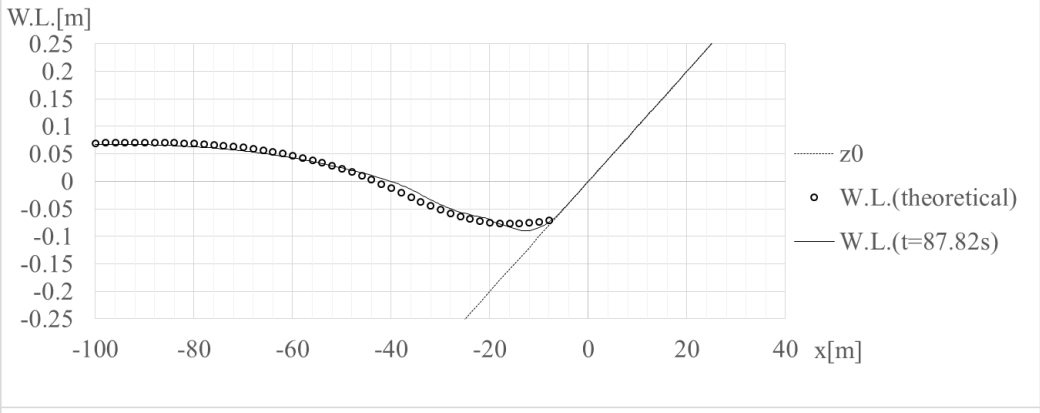
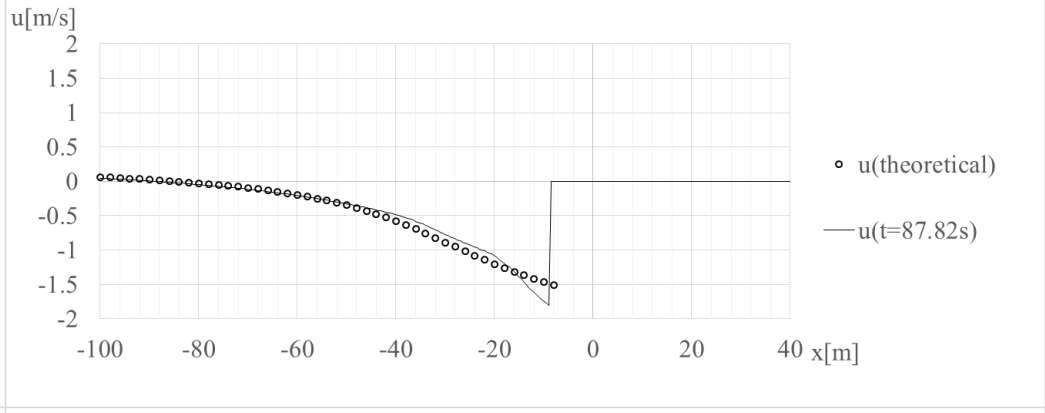



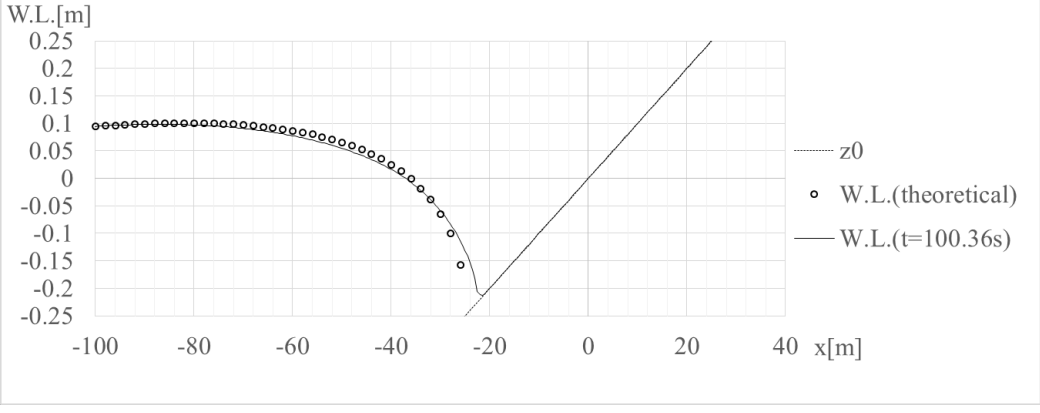

図2-3 一次元遡上計算の水位･流速の分布



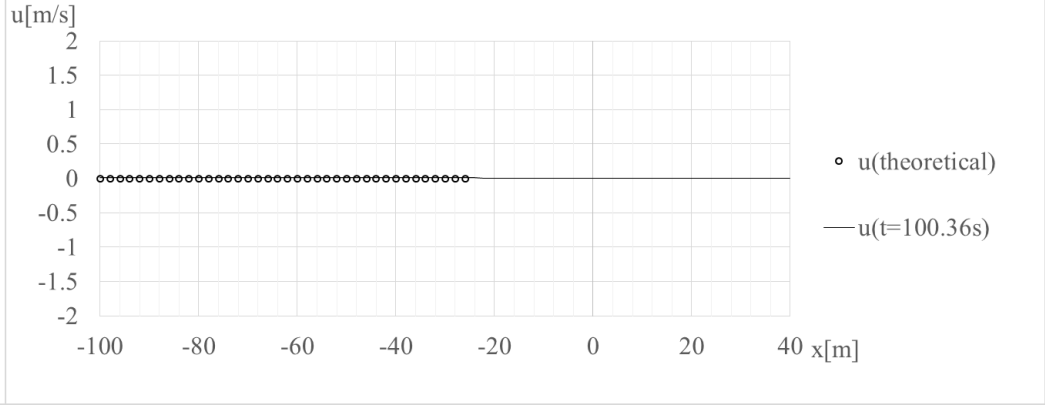




図2-3 一次元遡上計算の水位･流速の分布

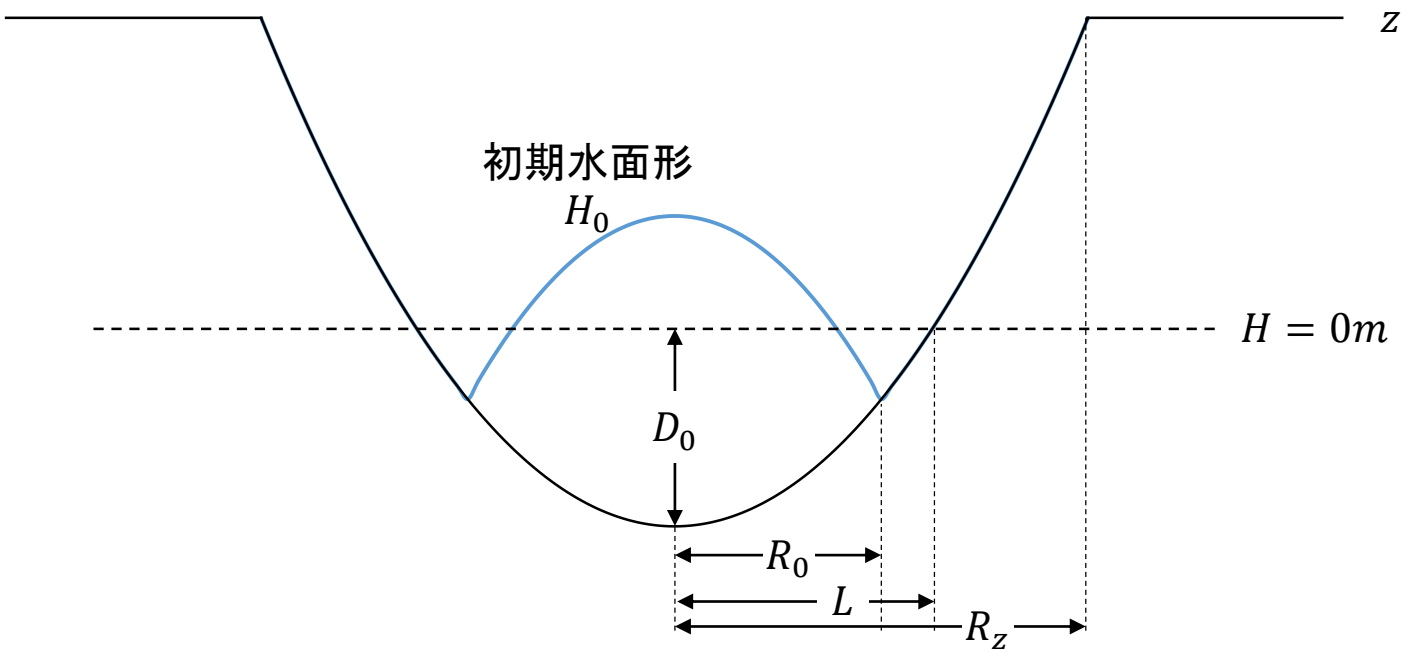



図2-4 二次元遡上問題の定数

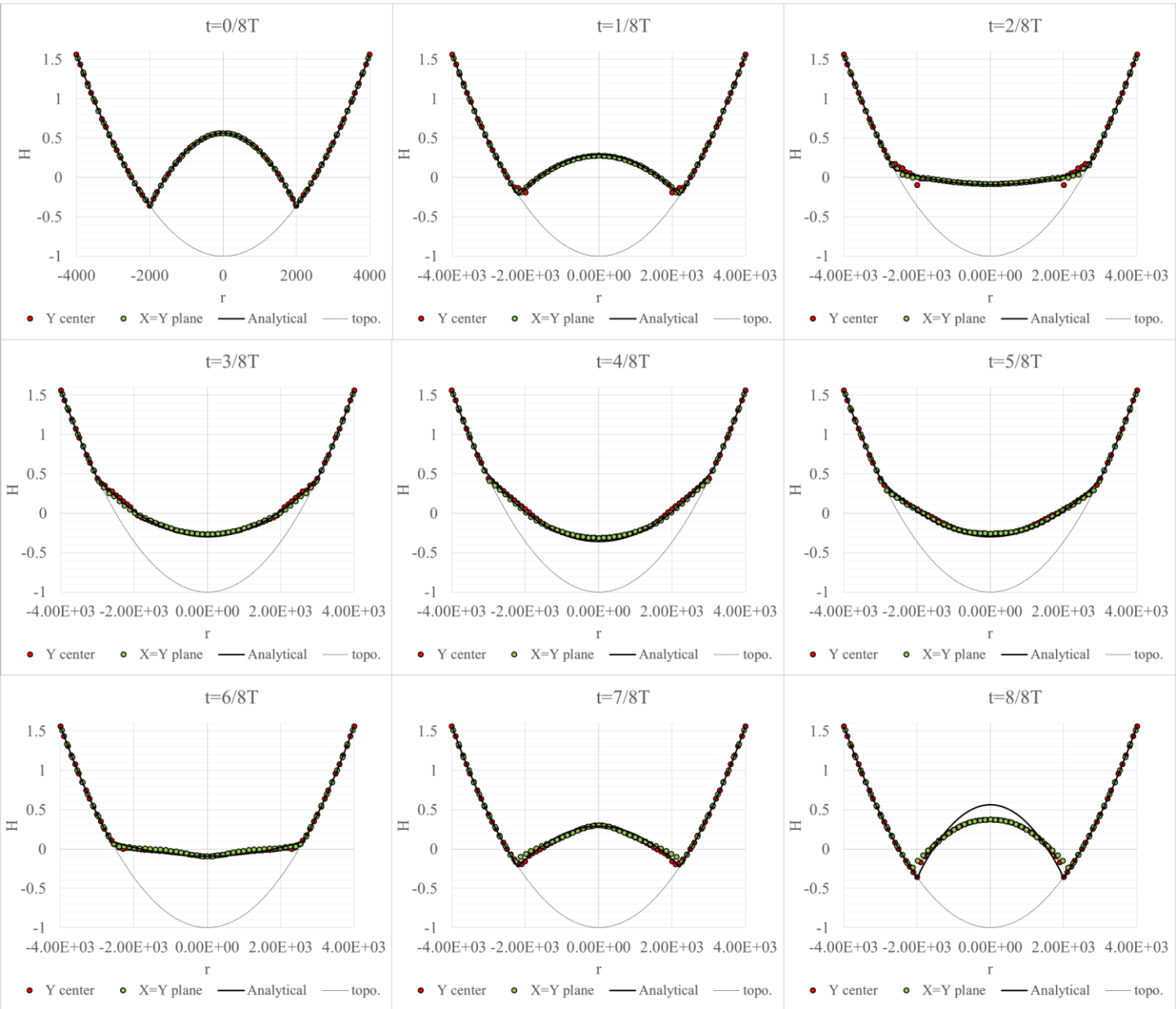



図2-5 二次元遡上問題の水位分布(case.1 $\Delta x = 100m$)

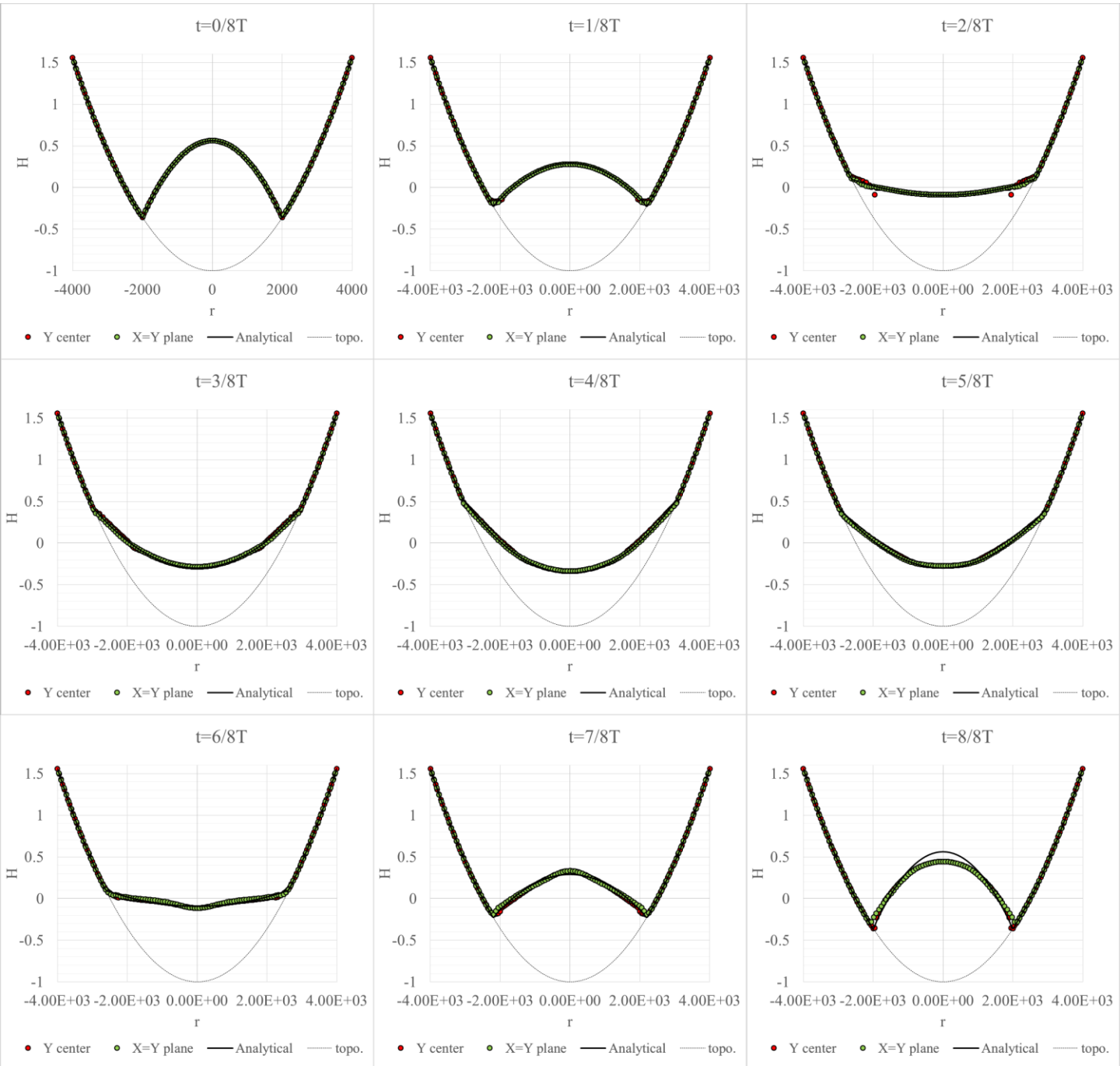



図2-5 二次元遡上問題の水位分布(case.2 $\Delta x = 50m$)

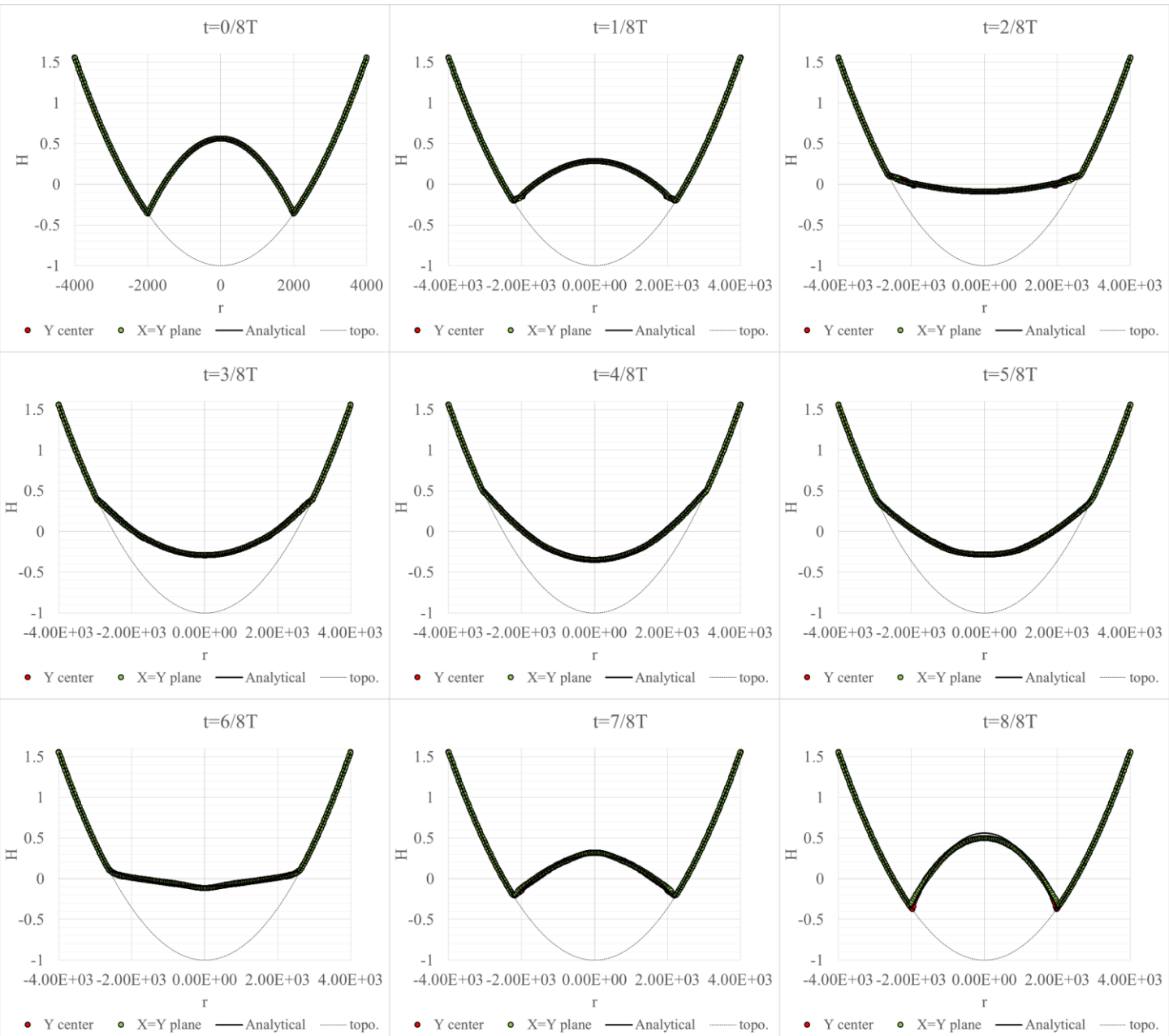



図2-5 二次元遡上問題の水位分布(case.3 $\Delta x = 25m$)

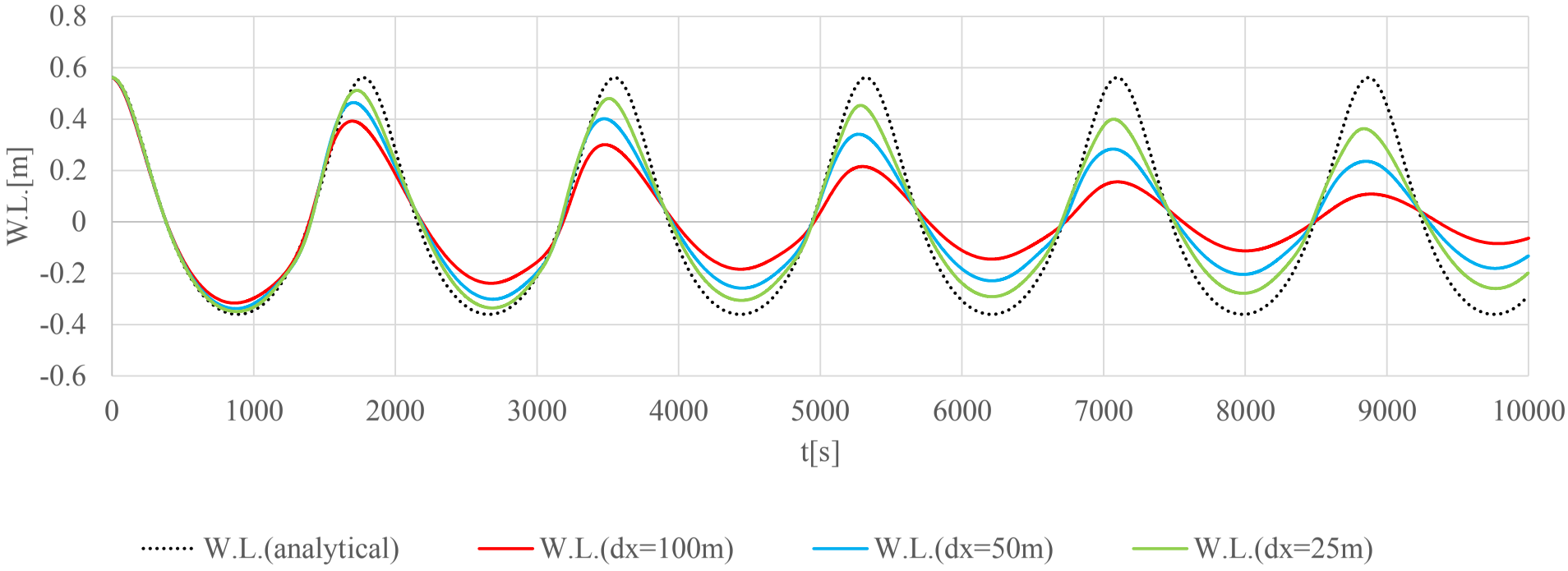
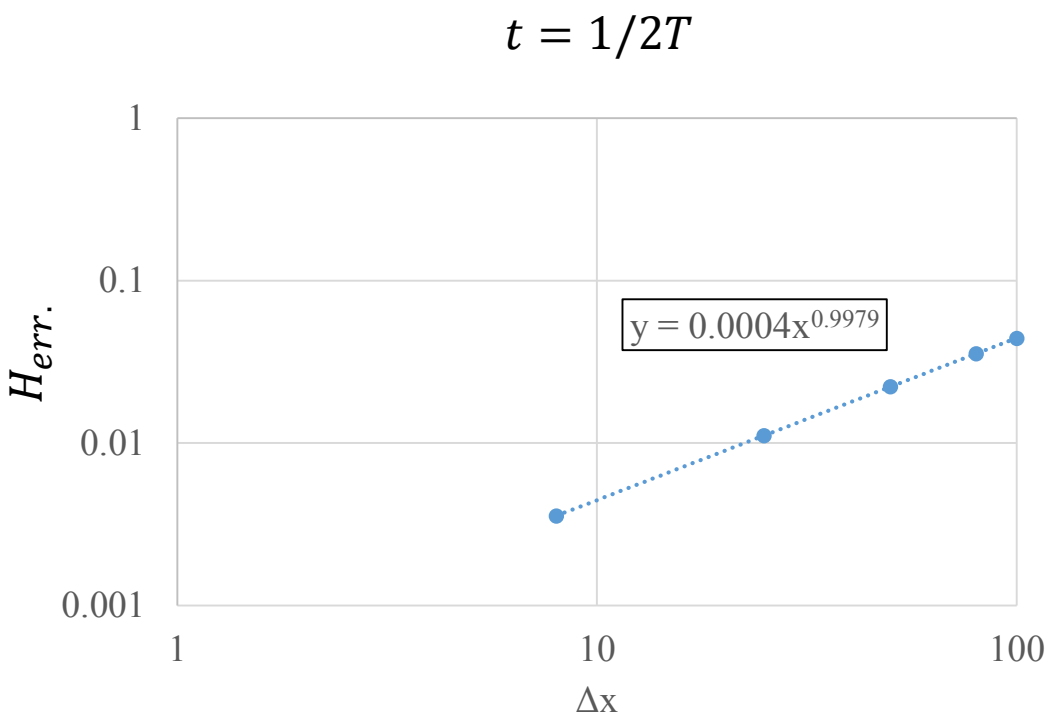



図2-6 原点における水位時間変動



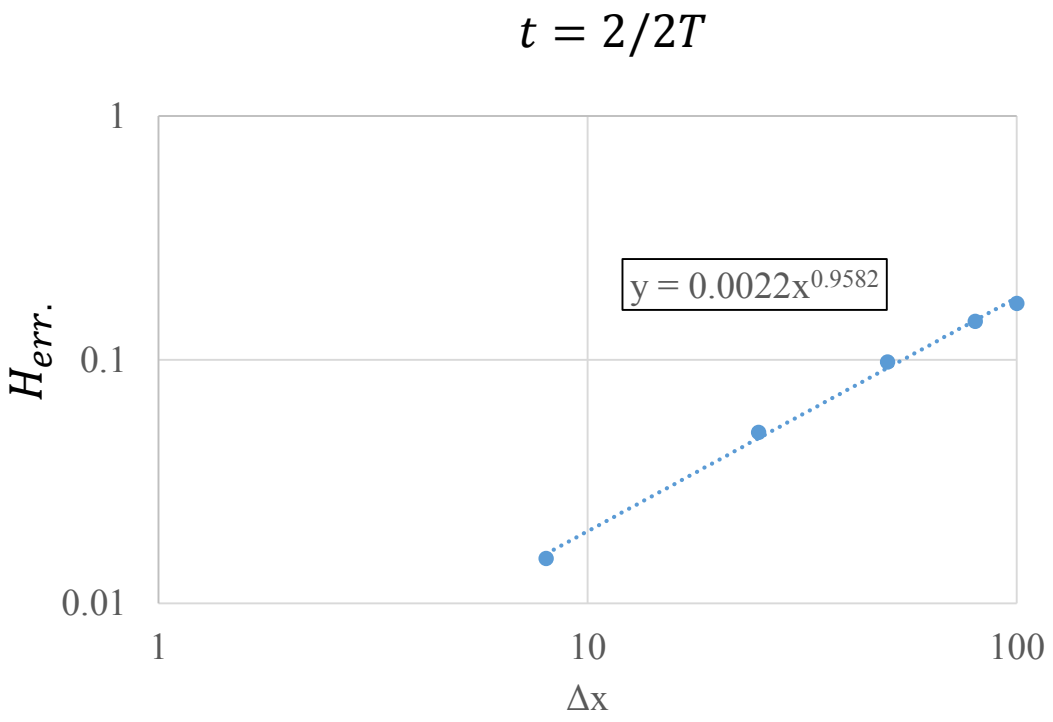




図2-7 ピーク時における$H_{err.} - \Delta x$

# 第 3 章 GPGPU による計算速度高速化処理

GPGPU(General-Purpose computing on Graphics Processing Units: GPU による汎用計算)は一般的に画像処理に用いられる GPU プロセッサを並列計算に応用し計算を高速に行う手法である．ここでは資料[1]を参考に GPU の構造と高速化の原理，その性能を十分に引き出すための CUDA と呼ばれる統合開発環境，そして前章のモデルを CUDA によって実装する方法について述べる．

## 3-1. GPU による高速化原理

GPU はこれまで汎用 PC における画像処理に特化した専用演算プロセッサであった．中央演算処理装置(CPU)が数コア程度なのに対し，GPU は CPU よりも 1 コア当たりの演算性能は劣るが演算ユニットを数百~数千コアのレベルで多数搭載しており，画像処理のような膨大なデータを並列に同一処理する作業に適している．その演算性能は日を追うごとに成長し，数 TFLOPS もの演算速度を実現している．一方で CPU は GPU のようなシンプルな並列処理よりも条件分岐等複雑なシーケンス的処理に適しており，その演算性能自体は数十~数百 GFLOPS となっている. そのためより高速並列計算が可能な GPU を用いて，計算機の数値計算に GPU を使用することで高速シミュレーションが実現された．

多数の GPU ベンダーの性能競争の結果，現在 GPU はアメリカの半導体メーカーである NVIDIA 社と AMD 社による寡占的状況に置かれており，両社とも 3D ゲーム，CAD，計算科学など様々な用途に適したシリーズの GPU ラインナップを展開している[2]．多くの GPU は汎用 PC におけるビデオカードとして提供されており，数万~数十万と低コストで簡単に導入できることがメリットの一つにもなっている．その中で本研究では数値計算に適した GPU として NVIDIA 社から提供されている Tesla シリーズ[3]を使用し，また同社からリリースされた CUDA と呼ばれる統合開発環境を用いて C 言語プログラムをベースに GPGPU プログラミングを行う．

**図 3-1** に NVIDIA 社製 Tesla 世代 GPU, Tesla C1060 のアーキテクチャを示す．他の世代においてもコア数の違いなどはあるがアーキテクチャはほぼ同様である．GPU はビデオカード上に搭載されており，また同じくビデオカード上にあるビデオメモリと接続されている．ビデオメモリは PCI Express バスと呼ばれる内部バスによって CPU と接続されており 8GB/sec の速度でデータ転送が可能となっている．CUDA プログラミングでは CPU からビデオメモリ(CUDA のメモリモデルではグローバルメモリという領域として確保される)に各種変数の数値を転送し，GPU 内部で計算を行いファイル出力などのタイミングで GPU 側から CPU 側に転送し直す．

一方 GPU には 30 個の SM(Streaming Multiprocessor)が搭載されており，また 1 個の SM 内には 8 個の SP(Streaming Processor)と 16KB のシェアードメモリ(Shared Memory)がある．SP は GPU を用いた数値計算における最小単位の演算処理ユニットである．結果として GPU1 チップに 8×30=240 個の SP が搭載されており，一般的な CPU(Intel Core i7 シリーズで 4~10 コア)に比べ大量の計算コアを保持している．各 SM はグローバルメモリにアクセス可能となっているが，SP の計算速度に比べると非常に低速であるため，より高速にアクセス可能なシェアードメモリにデータを確保し, SM 内で処理を行うのが計算高速化に重要となっている．

## 3-2. CUDA プログラミング

先述した GPU の性能を最大限発揮できるように用いるのが CUDA である．CUDA(クーダ: Compute Unified Device Architecture)は NVIDIA 社が提供する GPU 向けの C 言語の統合開発環境であり、コンパイラ(nvcc)やライブラリなどから構成されている。CUDA は NVIDIA 社から 2016 年現在 CUDA Toolkit 8.0 が無償で提供されており，同社の Web ページからダウンロードすることができる[4]．ここでは GPGPU 化における CUDA のプログラミングモデル，スレッドモデル，メモリモデルについて述べる．

### 3-2-1. CUDA プログラミングモデル

CUDA は従来の C 言語プログラムを拡張し，GPU へのデータ転送や GPU 自体の操作・管理などライブラリを含んでいる．また CUDA 用コンパイラ nvcc により実行可能形式へコンパイルされる．

CUDA プログラムでは CPU, GPU をそれぞれ別のデバイスとして扱うため CPU を「ホスト」, GPU を「デバイス」と呼称する. **図 3-2** に CUDA プログラミングのイメージを示す. GPGPU プログラムは 2 種類のコードで構成され, ホスト(CPU)側で動くコードを「ホスト・コード」，デバイス(GPU)側で動くコードを「デバイス・コード」と呼ぶ．ホスト・コードには通常の C 言語プログラムに加えて「CUDA ランタイム API」や GPU の動作実行を命令する「カーネル関数コール」等が含まれる．デバイス・コードには実際に GPU 上で，スレッド単位で動く「カーネル関数」が記述される．

実際にプログラムは CPU 上で実行開始される．ホスト・コード中で, 各種変数データ f を保存するポインタとしてホスト・メモリ用の f_h ポインタとデバイス・メモリ用の f_d ポインタが宣言される．ホスト・メモリについては標準的な C 言語プログラムと同様に「malloc 関数」を用いて配列 f_h[i]が確保されるが，デバイス・メモリについては CUDA ランタイム API の「cudaMalloc」関数を用いてデバイス上に配列 f_d[i]を確保する．ホスト・メモリに入

力ファイル等からデータが入力された後，ホストからデバイスにデータを転送する時にも，CUDA ランタイム API の「cudaMemcpy」関数を使用して f_h[i]から f_d[i]にデータをコピーする．

次に GPU 上で計算するためのカーネル関数を呼び出し，デバイス上にプログラムの最小実行単位であるスレッドが生成される．スレッドはセル 1 つずつのデータ処理を行うように並列化され，後述する「ブロック」の単位で実行される．

最後にデバイスでの計算が終了し，計算結果をホスト側へ転送するときには再び CUDA ランタイム API の「cudaMemcpy」関数を使用して f_d[i]から f_h[i]にデータをコピーする．ただしこの CPU と GPU 間の転送速度は GPU 内部でのグローバルメモリ―SP 間のアクセス速度に比べ非常に遅い．そのため計算を早く終わらせるためにデータ転送は極力回数を減らし，デバイス上で計算を繰り返す必要が有る．

### 3-2-2. CUDA スレッドモデル

前節のカーネル関数の呼び出しにより GPU 上にはスレッドが大量に生成される．GPU はこの大量のスレッドの並列処理を行うが，効率的に処理するために CUDA ではスレッドを階層的に管理している．**図 3-3** に CUDA のスレッドモデルを示す．

CUDA では「スレッド(thread)」の塊を「ブロック(block)」，「ブロック(block)」の塊を「グリッド(grid)」というように上から「グリッド―ブロック―スレッド」の順で階層的にスレッドを取りまとめている．グリッドの中ではブロックが，ブロックの中ではスレッドがマトリクス上に配置され，(x，y，z)方向の最大 3 次元構造で配置することができる．ただし，グリッド中の z 方向ブロック数は 1 で固定されるため実質的に 2 次元構造となっている．**図 3-3** ではスレッド，ブロックともに 2 次元構造で配置した場合のスレッドモデルを表している．

これらのスレッドをプログラム中で構造的に管理するための変数として，カーネル関数の中には宣言せずに使用可能なビルトイン変数が用意されている．ビルトイン変数はスレッドのブロックにおける ID(配置番号)と，その含まれるブロック自体のグリッドにおける ID を示す変数，そしてブロックとグリッドのサイズを示す変数として以下のものが用意されている．

[スレッドのブロックにおける ID]
threadIdx.x: 各ブロックにおけるそのスレッドの x 方向配置番号
threadIdx.y: 各ブロックにおけるそのスレッドの y 方向配置番号
threadIdx.z: 各ブロックにおけるそのスレッドの z 方向配置番号
[ブロックのグリッドにおける ID]
blockIdx.x: グリッドにおけるそのブロックの x 方向配置番号
blockIdx.y: グリッドにおけるそのブロックの y 方向配置番号

blockIdx.z: グリッドにおけるそのブロックの z 方向配置番号(0 のみ)
[ブロックのサイズ]
blockDim.x: ブロックに含まれる x 方向スレッド数
blockDim.y: ブロックに含まれる y 方向スレッド数
blockDim.z: ブロックに含まれる z 方向スレッド数
[グリッドのサイズ]
gridDim.x: グリッドに含まれる x 方向ブロック数
gridDim.y: グリッドに含まれる y 方向ブロック数
gridDim.z: グリッドに含まれる z 方向ブロック数(1 に固定)

スレッド，ブロックの ID の最大値は含まれるブロック，グリッドのサイズによって決定されるので「$0 \leq \mathrm{threadIdx.x} \leq \mathrm{blockDim.x} - 1$」という関係が成り立つ．**図 3-3** に示す例では，グリッドのサイズは(5，4)となり，総ブロック数は 5×4=20 となる．またブロックのサイズは(4，3)となり 1 ブロックに 4×3=12 のスレッドが含まれる構造となっている．

カーネル関数を使用するときには，

```
dim3 Dg (5，4，1)，  Db(4，3，2)
```

のように 3 次元ベクトルの整数型宣言である dim3 宣言を使用してグリッドのサイズ Dg，ブロックのサイズ Db を設定する．このとき Dg,Db は dim3 型変数となり，gridDim.x=5，gridDim.y=4,…とブロック及びグリッドの各方向のサイズが決定される．この 2 つのサイズ変数を使用して，

```
func_kernel <<< Dg, Db >>> (f_d);
```

と「<<< >>>」で挟むことによりカーネル関数を呼び出す事ができる．

また CUDA プログラミングではこれらのビルトイン変数を用いて各スレッドの担当する 2 次元セル配置を決定する．同一ブロック中に含まれる全スレッドは 1 つの SM に割り振られる．SM は 8 個の SP を 4 サイクルかけて同一の命令を実行するので同時に 8×4=32 個のスレッドが SM 内で処理される．この 32 スレッドを 1 単位として，ウォープ(Warp)と呼称する．効率よく計算するためには，1 ブロック中のスレッド数をウォープの整数倍にして SP をフルに利用できるようにしなければならない．ただし 1 ブロック中にはスレッドは最大 512 個までしか格納できないのでこちらの条件も満たすように設定しなければならない．

### 3-2-3. CUDA メモリモデル

CUDA ではハードウェアの命名とは異なる独自のメモリモデルを導入している．**図 3-4** に CUDA のメモリモデルを図示する．CUDA ではレジスタ，ローカルメモリ，シェアードメモリ，グローバルメモリ，コンスタントメモリ，テクスチャメモリを使用することができる．以下に各メモリの機能について述べる．

- レジスタ
  GPU 内に実装されるオンチップ型メモリ．チップ内にあるため高速に読み書きができる．カーネル関数で使用される変数はレジスタ上に保持される．このメモリは各スレッドによって占有されるため，他のスレッドからのアクセスは不可能になっている．
- ローカルメモリ
  GPU 外に実装されるオフチップ型メモリ．レジスタが足りなくなったときにデータの退避場所として使われるが，チップ外にあるためレジスタと比較すると 100 倍以上アクセスが遅い．
- シェアードメモリ
  オンチップ型メモリ．各 SM に 16,384byte 備わっており，1 つのブロックで共有して使うことができブロック内では異なるスレッド間でもデータアクセスが可能である．保存期間はスレッド処理中に限られるが，アクセスが非常に高速なので繰り返し使用するデータやスレッド間で共有するデータなどを保存するのに適している．
- グローバルメモリ
  オフチップ型メモリ．レジスタやシェアードメモリと比較すると 100 倍以上アクセスが遅いが，**図 3-1** のビデオメモリ上に置かれるため 4GB と大量のデータを保持することができる．CUDA ではブロック間で共有されており，すべてのスレッドからアクセスすることができる．
- コンスタントメモリ･テクスチャメモリ
  コンスタントキャッシュ･テクスチャキャッシュを使用するためのメモリ

GPGPU 化においてこれらの特徴の異なるメモリを組み合わせて使用することで効率よく計算高速化をすることが可能となる．

ここまでの CUDA における各モデルの上下構造を**表 3-1** に示す．計算モデルでは全計算領域を多数のサブドメインに分割する．サブドメインにはまた多数の計算セルが含まれるが，スレッドがセルを処理し，ブロック単位でサブドメインを担当する．1 ブロックの計算は 1 つの SM が処理し，SP がスレッドを読み込んでセルごとの計算を行う．また，SP は自身の持つレジスタにデータを格納しスレッドの処理を行い，ブロック(サブドメイン)内で共有されるデータを使用するときには SM が持つシェアードメモリに保存されたデータを参照する．このように CUDA における各モデルは上下構造をなしており，それぞれ横一列に相当する関係を持っている．

## 3-3. CUDA による流動モデルの実装

ここでは前章で述べた流動モデルを，実際に CUDA を用いて GPGPU 化する手法につい

て述べる.

### 3-3-1. ホストコードプログラム

**図 3-5** に GPGPU 化されたプログラムのフローチャートを，**図 3-6** にホストコードの main 関数を示す．**図 3-6** の①では変数宣言を行うが，各変数 h,u,v,...についてホスト用の変数とデバイス用の変数(_d)を用意している. これらの変数を, ②で malloc 関数及び CUDA ランタイムの cudaMalloc 関数を用いてメモリ確保を行う．ここに NX_ALL 及び NY_ALL は x 方向及び y 方向セル配置数で，ヘッダーファイル(~.h)内で定義しておく．③で地形データや初期条件をホスト側でセットした後，④で cudaMemcpy 関数を使ってデバイス側に同じ値をコピーする．cudaMemcpyHostToDevice はこの関数の転送タイプを指定する定数で, これによりホストからデバイスへ転送することを指定している．ここまでが時間発展計算前の準備である. ⑤から時間発展ループを開始する. ここから GPU 内で計算を行うので, ⑥から⑭ではすべてカーネル関数を呼び出している．ここで grid 及び block は dim3 型の変数で③内でそのサイズを指定している．ファイル出力時間になると⑮でデバイス側からホスト側へ cudaMemcpy 関数を使用してコピーする．この時④とは逆方向にデータを転送するので転送タイプを cudaMemcpyDeviceToHost としている. そしてホスト側でファイル出力を終えたら次のファイル出力時間を設定し次の時間ループに進む．すべての計算が終了したのち，⑯で free 関数及び cudaFree 関数を使用してホスト･デバイス両方のメモリを解放しプログラムは終了となる.

### 3-3-2. デバイスコードプログラム

デバイスコードプログラムの例として**図 3-7** に x 方向フラックス計算カーネル関数を示す．カーネル関数の関数名(Flux_x_kernel)には関数に対する修飾子として__global__修飾子を添えてある．これにより，ホスト側から呼び出されたときにこの関数がデバイス上でのみ実行される．①ではスレッド間で共有する必要のあるデータをシェアードメモリ上で宣言する．ここでは__shared__修飾子を添えることでこの変数がシェアードメモリ上に確保されることを表す．また N_IN_BLOCK はブロックのサイズを指定しておりヘッダーファイル内で定義している. シェアードメモリ上に確保される変数はブロック単位で確保され，ブロック内のスレッドから読み書きが可能となる．②では各スレッドの持つ変数としてレジスタ上に変数を確保する．ここでは①とは異なり修飾子は添えない. ③では対象スレッドが受け持つ格子点の番号をスレッド ID,ブロック ID を用いて計算する．CUDA では多次元配列よりも 1 次元配列でデータを管理するほうが圧倒的に効率がよいとされている．そこで

```
i_global=blockIdx.x*(N_IN_BLOCK-2)+threadIdx.x;
j_global=blockIdx.y*(N_IN_BLOCK-2)+threadIdx.y;
```

i_1d_global=NX_ALL*j_global+i_global;

として，**図 3-8**，**図 3-9** のようにスレッドが構成される．ここで N_IN_BLOCK-2 としているのはブロックの両端を隣接ブロックと重ね合わせて「のりしろ」部分を作ることにより，サブドメイン間でのデータ転送を可能にするためである．④ではグローバルメモリからシェアードメモリへデータの読み込みを行う．このフラックス計算関数のように 1 つのスレッドで複数のセルの値を用いて計算を行う場合には，グローバルメモリ上の値を直接参照するのではなくシェアードメモリに各値を格納してスレッドごとに利用することでより高速化される．また__syncthreads( );はスレッドの同期をとる CUDA 関数で，ブロック中の全スレッドがこの関数に到達してから次の処理に移ることができる．⑤では各スレッドにおいて計算に必要なデータをシェアードメモリからレジスタに読み込み，⑥で(2-35)式に基づいて Roe 平均値を計算，⑦で(2-42)式に基づいてフラックスを計算する．すべての計算が終わった後，⑧で再びグローバルメモリにアクセスし計算されたデータを書き出してカーネル関数は終了となる．

このようにカーネル関数ではシェアードメモリを利用し効率の良いメモリアクセスによって計算を高速に行うことができる．

## 第 3 章 参考文献


1. 青木 尊之，額田 彰: はじめての CUDA プログラミング，工学社，2009
2. GDEP 日本 GPU コンピューティングパートナーシップ，ゼロから始める GPU Computing: http://www.gdep.jp/page/index/market
3. NVIDIA，高性能コンピューティング(HPC)およびスーパーコンピューティング: http://www.nvidia.co.jp/object/tesla-supercomputing-solutions-jp.html
4. NVIDIA, CUDA Toolkit: https://developer.nvidia.com/cuda-toolkit

# 第3章
# 図表集

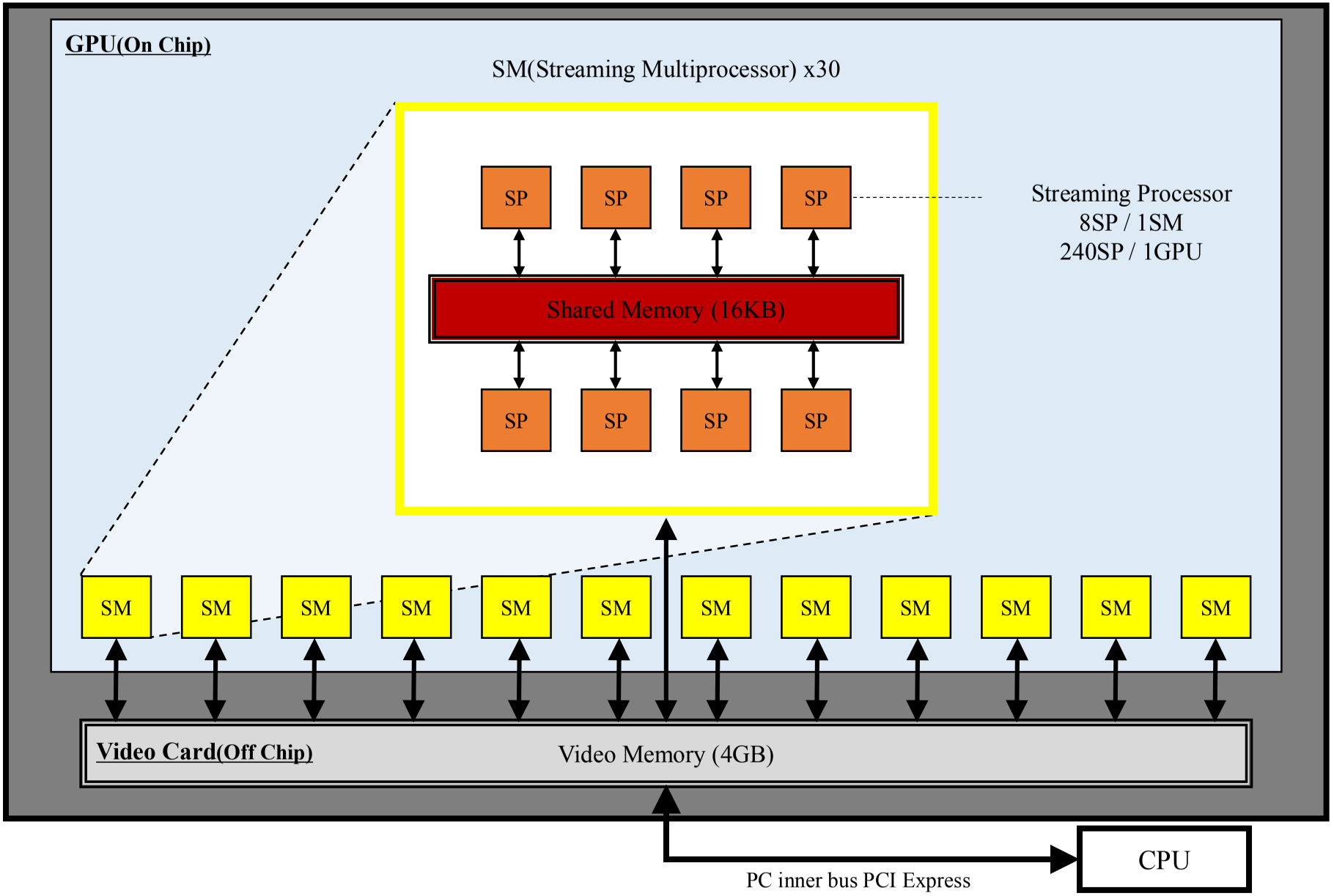



図3-1 GPUアーキテクチャ

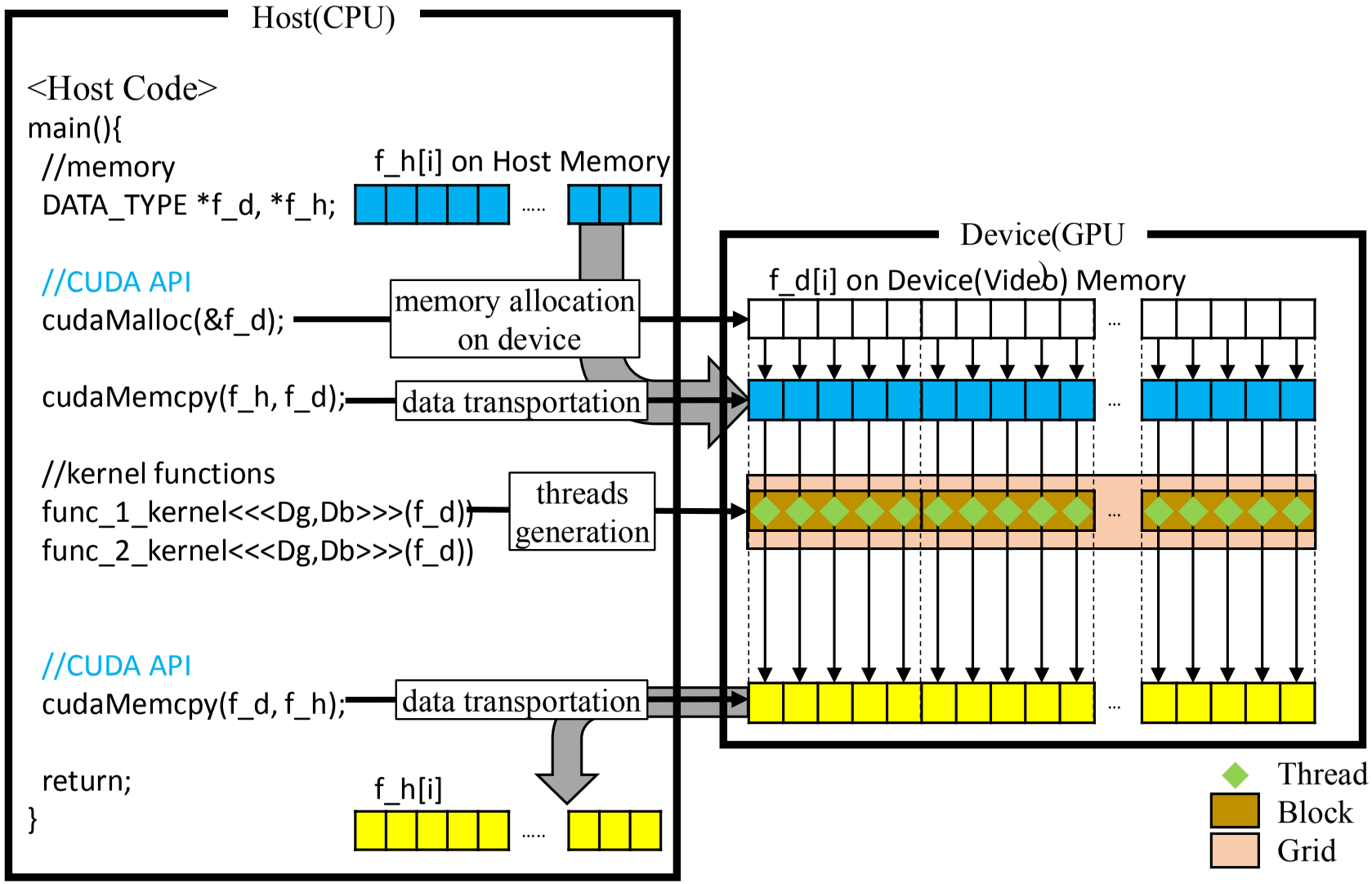



図3-2 CUDAプログラミングイメージ

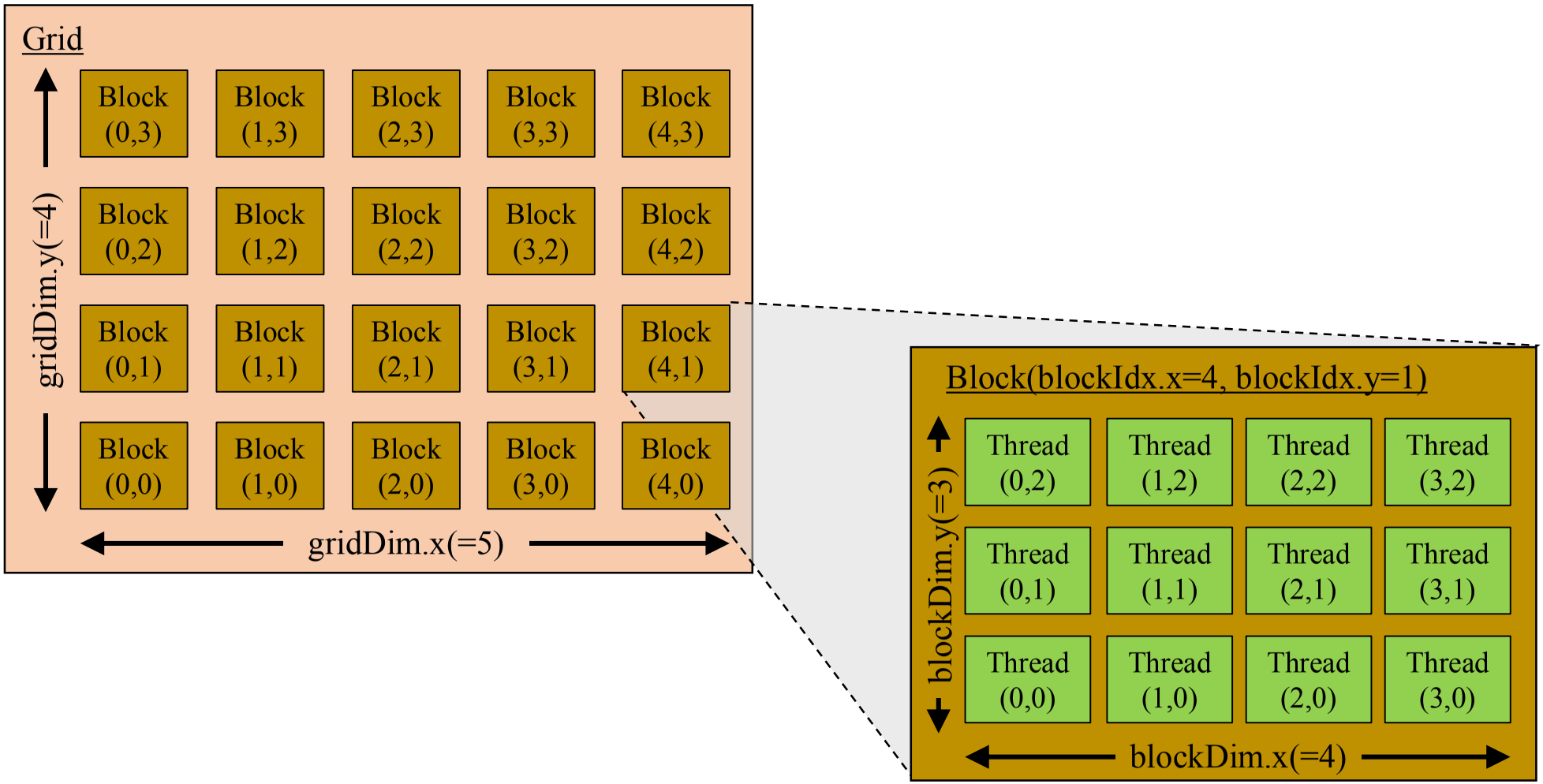



図3-3 CUDAスレッドモデル

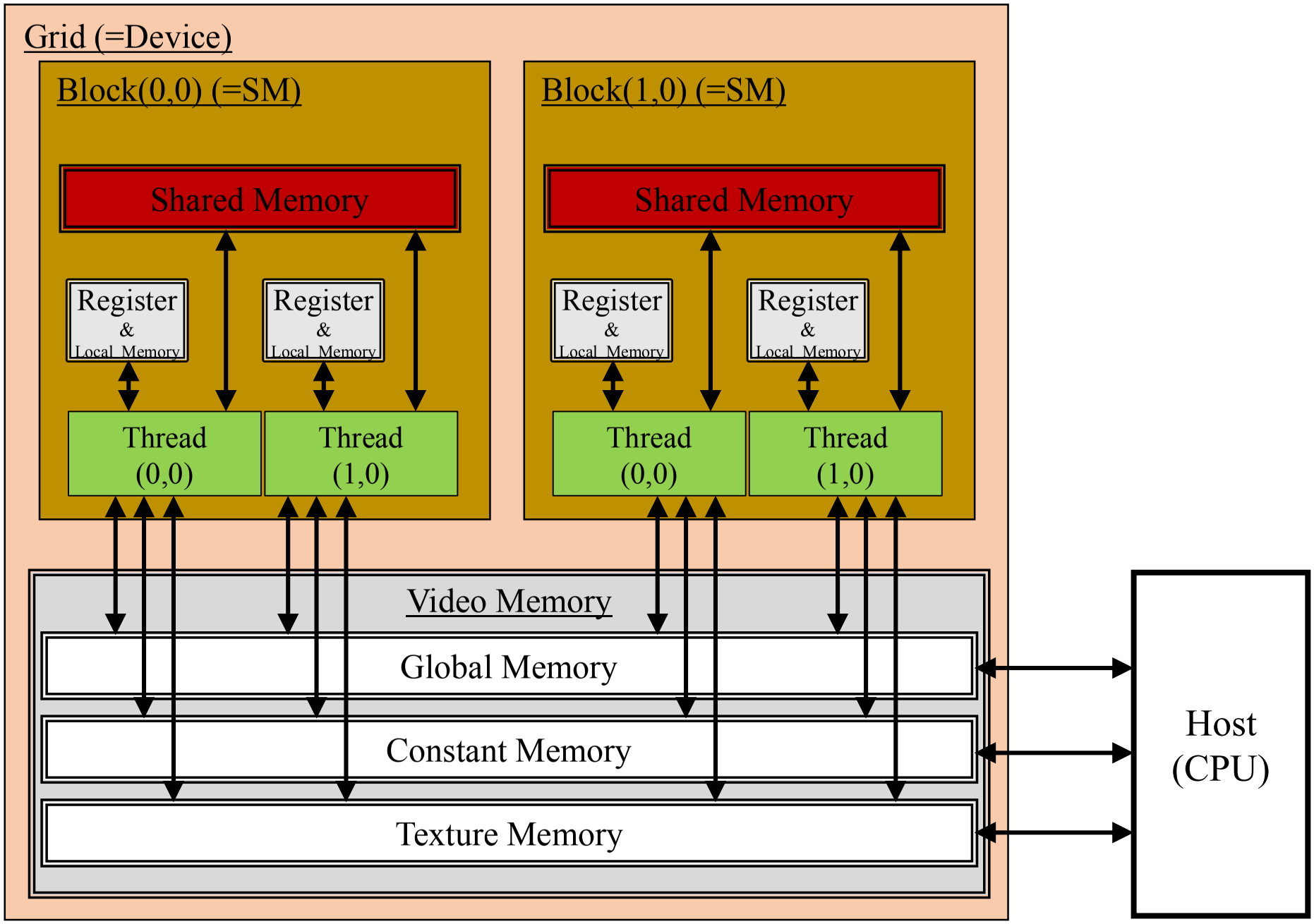



図3-4 CUDAメモリモデル

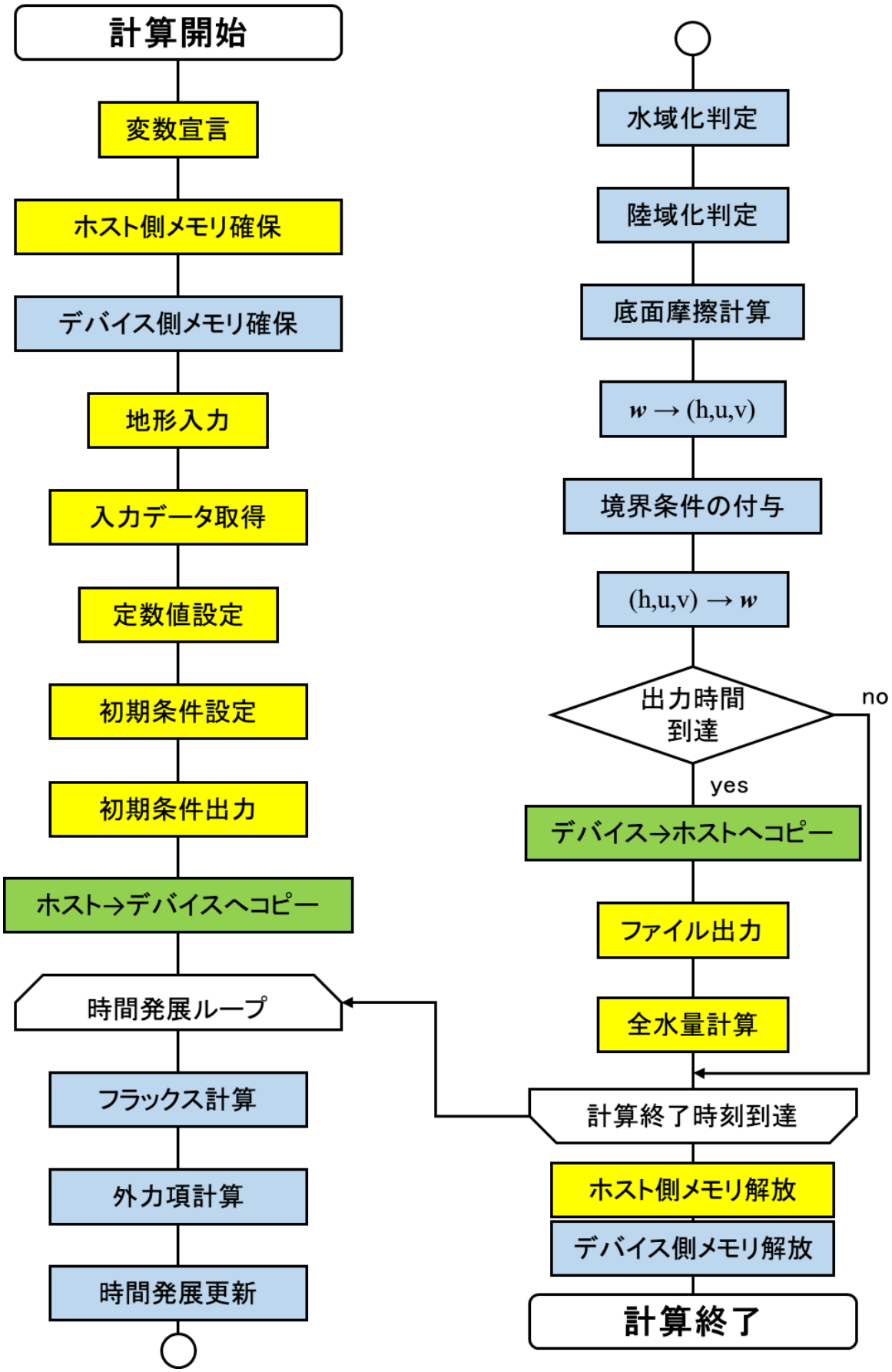



図3-5 計算フローチャート

```
#include "shallow2d.cuh“
int main(int argc, char *argv[ ] )
{
//①変数宣言
  DATA_TYPE *h;
  DATA_TYPE *u;
  DATA_TYPE *v;
                ･･･省略･･･
  DATA_TYPE *h_d;
  DATA_TYPE *u_d;
  DATA_TYPE *v_d;
                ･･･省略･･･

//②メモリ確保
  h=(DATA_TYPE *)malloc(sizeof(DATA_TYPE)*NX_ALL*NY_ALL);
  u=(DATA_TYPE *)malloc(sizeof(DATA_TYPE)*NX_ALL*NY_ALL);
  v=(DATA_TYPE *)malloc(sizeof(DATA_TYPE)*NX_ALL*NY_ALL);
                ･･･省略･･･
  cudaMalloc((void **)&(*h_d),sizeof(DATA_TYPE)*NX_ALL*NY_ALL);
  cudaMalloc((void **)&(*u_d),sizeof(DATA_TYPE)*NX_ALL*NY_ALL);
  cudaMalloc((void **)&(*v_d),sizeof(DATA_TYPE)*NX_ALL*NY_ALL);
                ･･･省略･･･

//③地形等データ入力
                ･･･省略･･･

//④ホスト→デバイスへコピー
  cudaMemcpy(h_d, h, sizeof(DATA_TYPE)*NX_ALL*NY_ALL,
                                                cudaMemcpyHostToDevice);
                ･･･省略･･･

//⑤時間発展ループ開始
  for(time_step=0; time<time_end; time_step++){

//⑥フラックス計算のカーネル関数
  Flux_x_kernel<<<grid,block>>>(h_d, u_d, v_d, F1_d, F2_d, F3_d, land_flag_d);
  Flux_y_kernel<<<grid,block>>>(h_d, u_d, v_d, G1_d, G2_d, G3_d, land_flag_d);
```

図3-6 ホストコードのmain関数

```
//⑦外力計算のカーネル関数
 ExtraForce_x_kernel<<<grid,block>>>(h_d, z_d, u_d, v_d, S1_d, …, land_flag_d);
 ExtraForce_y_kernel<<<grid,block>>>(h_d, z_d, u_d, v_d, S1_d, …, land_flag_d);

//⑧時間発展更新のカーネル関数
 update_kernel<<<grid,block>>>(w1_d, w2_d, w3_d, F1_d, F2_d, F3_d, G1_d, G2_d, G3_d,
                                                    S1_d, S2_d, S3_d, land_flag_d);

//⑨水域化判定のカーネル関数
 submergence_kernel<<<grid,block>>>(z_d, temp_d, w1_d, w2_d, w3_d, land_flag_d,
                                                        new_land_flag_d);

//⑩陸域化判定のカーネル関数
 emergence_kernel<<<grid,block>>>(w1_d, w2_d, w3_d, land_flag_d);

//⑪底面摩擦計算のカーネル関数
 BtmFric_kernel<<<grid,block>>>(w1_d, w2_d, w3_d, land_flag_d);

//⑫ wから(h,u,v)に更新するカーネル関数
 trans_from_w_variables_kernel<<<grid,block>>>(h_d, u_d, v_d, w1_d, w2_d, w3_d);

//⑬境界条件付与のカーネル関数
 update_bc_kernel<<<grid_river,block_river>>>(…);

//⑭ (h,u,v) からwに更新するカーネル関数
 set_w_variables_kernel<<<grid,block>>>(h_d, u_d, v_d, w1_d, w2_d, w3_d);

//⑮出力時間にファイル出力
 if(time>=output_next_time){
  cudaMemcpy(h, h_d, sizeof(DATA_TYPE)*NX_ALL*NY_ALL,
                                                    cudaMemcpyDeviceToHost);
  (ファイル出力関数の呼び出し)
  output_next_time+=output_time_interval;
 }

//⑯メモリ解放
 free(h);
・・・省略・・・
 cudaFree(h_d);
・・・省略・・・
```

図3-6 ホストコードのmain関数

```
#include "shallow2d.cuh“
__global__ void Flux_x_kernel(DATA_TYPE *h_global,… )
{
//①シェアードメモリ上の変数宣言
  __shared__ DATA_TYPE h_sh[N_IN_BLOCK][N_IN_BLOCK];
                ･･･省略･･･

//②レジスタ上の変数宣言
  DATA_TYPE hup;
  DATA_TYPE hdn;
                ･･･省略･･･

//③スレッドが受け持つ格子点の番号
  i_global=blockIdx.x*(N_IN_BLOCK-2)+threadIdx.x;
  j_global=blockIdx.y*(N_IN_BLOCK-2)+threadIdx.y;
  i_1d_global=NX_ALL*j_global+i_global;

//④グローバルメモリからブロック内シェアードメモリへ読み込み
  h_sh[threadIdx.x][threadIdx.y]=h_global[i_1d_global];
                ･･･省略･･･
__syncthreads( );

//⑤sharedメモリからregisterメモリへ格納
  hdn=h_sh[threadIdx.x][threadIdx.y];
  udn=u_sh[threadIdx.x][threadIdx.y];
  vdn=v_sh[threadIdx.x][threadIdx.y];

  if(threadIdx.x==blockDim.x-1){
    hup=hdn;
    uup=udn;
    vup=vdn;
  } else{
    hup=h_sh[threadIdx.x+1][threadIdx.y];
    uup=u_sh[threadIdx.x+1][threadIdx.y];
    vup=v_sh[threadIdx.x+1][threadIdx.y];
  }
```

図3-7 デバイスコードのx方向フラックス計算カーネル関数

```
 //⑥Roe平均の計算
 roe_h=0.5*(hup+hdn);
 tmp1=SQRT(hup);
 tmp2=SQRT(hdn);
 roe_u=(tmp1*uup+tmp2*udn)/(tmp1+tmp2);
 roe_v=(tmp1*vup+tmp2*vdn)/(tmp1+tmp2);
 roe_c=SQRT(roe_h*(DATA_TYPE)GRAVITY_ACC);

//⑦フラックス計算
・・・省略・・・
 tmp1=0.5*((w2dn+w2up)+(abs_F11*lamda1+abs_F12*lamda2+abs_F13*lamda3));
 tmp2=…
 tmp3=…
・・・省略・・・

//⑧グローバルメモリに書き出し
 F1_global[i_1d_global]=tmp1;
 F2_global[i_1d_global]=tmp2;
 F3_global[i_1d_global]=tmp3;

 return ;
}
```

図3-7 デバイスコードのx方向フラックス計算カーネル関数

| | アーキテクチャ | メモリモデル | スレッドモデル | 計算モデル |
|---|---|---|---|---|
| 上位 | GPU | グローバルメモリ | グリッド | ドメイン(計算領域) |
| | ストリーミング<br>マルチプロセッサ(SM) | シェアードメモリ | ブロック | サブドメイン<br>(小領域) |
| 下位 | ストリーミング<br>プロセッサ(SP) | レジスタ | スレッド | セル |

表3-1 GPGPUにおける各モデルの上下構造

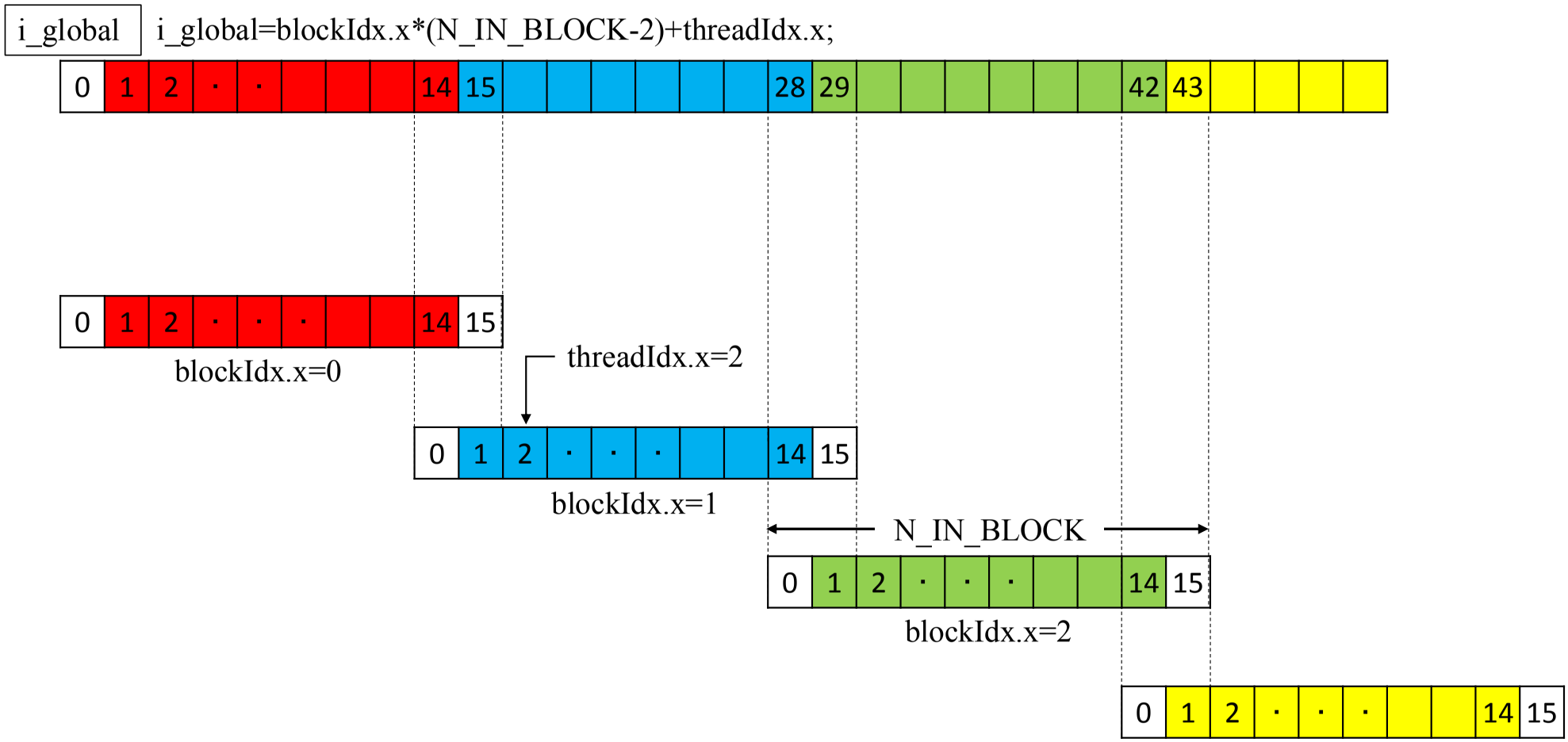



図3-8 1次元配列へのアクセス(例:N_IN_BLOCK=16)

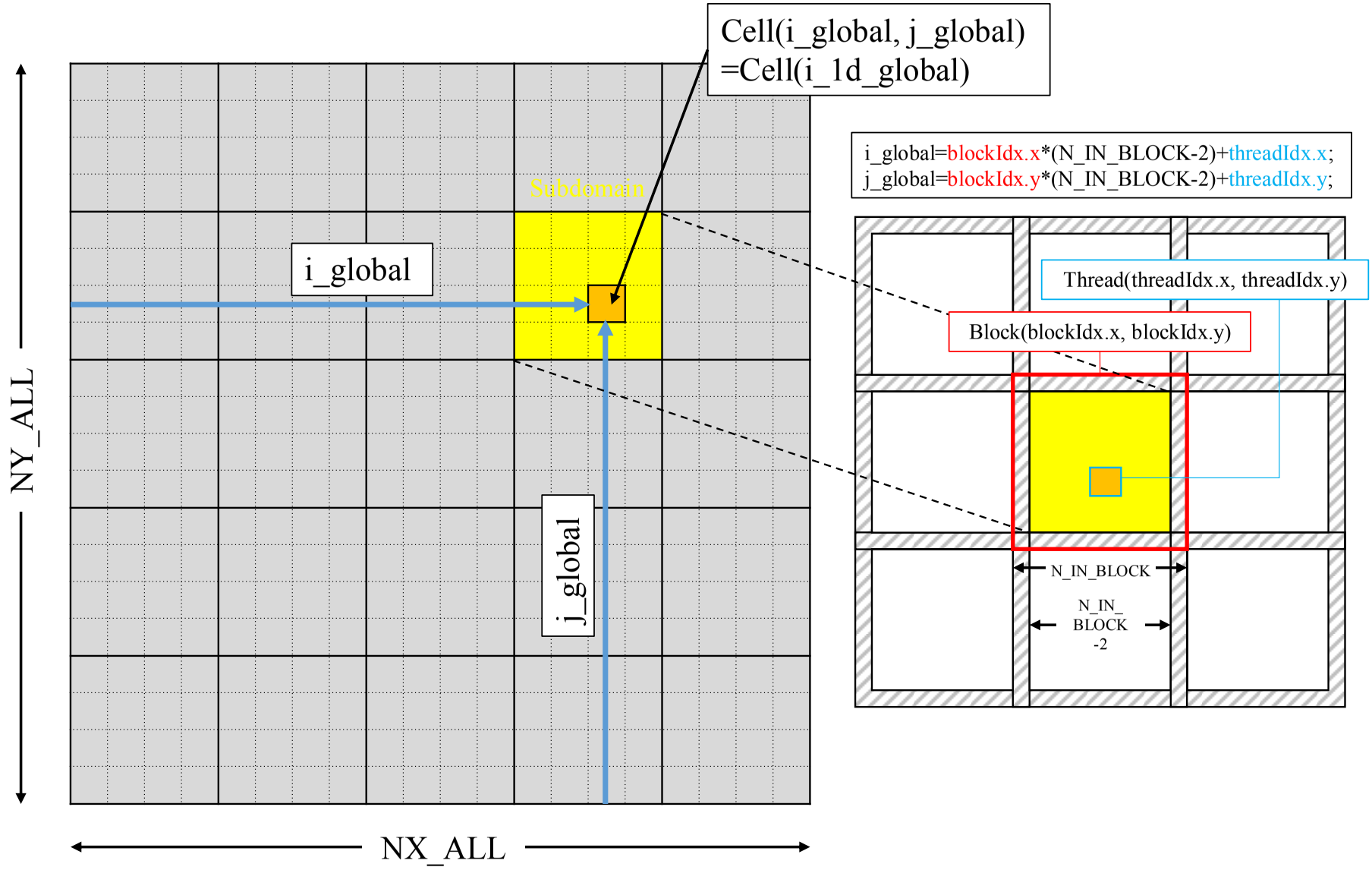

Cell(i_global, j_global)
=Cell(i_1d_global)
i_global=blockIdx.x*(N_IN_BLOCK-2)+threadIdx.x;
j_global=blockIdx.y*(N_IN_BLOCK-2)+threadIdx.y;
Subdomain
i_global
j_global
NY_ALL
NX_ALL
Thread(threadIdx.x, threadIdx.y)
Block(blockIdx.x, blockIdx.y)
N_IN_BLOCK
N_IN_
BLOCK
-2

図3-9 2次元的配列

# 第 4 章 トンレサップ湖を対象とした再現計算

先述した GPGPU 化された流動モデルを用いて，トンレサップ湖を対象に流動再現計算を行った．ここではその再現計算に使用した各種データ，計算条件について述べた後，再現計算の結果と観測データならびに衛星画像との比較を行う．また本再現計算における GPGPU 化による高速化の評価も行った．

## 4-1. 再現計算条件

### 4-1-1. 計算期間･入力条件

メコン川委員会(MRC)は水文プロジェクトとして TSLV プロジェクト(トンレサップ湖及び周辺地域の持つ多面的水文機能の評価と水文データの強化: Consolidation of hydro-meteorological data and its Vicinities)を 2001 年から実施している．また JICA の開発調査「メコン川流域線水文モニタリング計画調査(WUP-JICA プロジェクト)」が時を同じくして開始しており，特に 2002 年にメコン川本川，バサック川，トンレサップ川の主要水位観測点で ADCP を用いて隔週木曜日に流量観測を行っている[1].

本計算では入力境界条件として，トンレサップ川の Prek Kdam 地点における流量データを与えるため，再現計算期間をその観測日に合わせ 2002/7/18~2002/12/19 とした．**図 4-1** にその観測流量を示す．ここに流量の値が正のときは流向がトンレサップ湖方向，負のときは流向がメコン川方向である．流量観測は 2 週間おきに行われたため 3 次スプライン補間により観測日の間の流量値を内挿して与えた．また，流量だけでなく同時期に MRC で観測されている水位データ(**図 4-1**)も同様に境界条件として与えた．さらにトンレサップ湖はトンレサップ川を通じた流入出だけでなく上流側 11 支川からも流入するため，MRC で公開されている流量データを与えた(**図 4-2**)．

なお入力条件として流量を与える際は，流速一様とし，流路断面積で流量を除して流速を各計算セルに与えた．領域端における境界条件は非反射条件とした．

### 4-1-2. 計算地形

計算地形を作成するにあたり，トンレサップ湖･川の水域の河床データ，陸域の標高データの 2 種類の地形データを使用した．河床データに関しては MRC 作成の測深図を用いて河床標高を数値化した．また，陸域の標高データに関してはスペースシャトル立体地形データ(SRTM)の 3 秒(約 90m)メッシュデータと MRC 作成の DTM(Digital Terrain Model)50m

メッシュデータを用意した．**図 4-3** にそれぞれの標高を示す．両者を比べると，DTM データでは SRTM よりも小さい値が記録されている．そこで雨季の Landsat 衛星写真を用いて比較を行う．**図 4-4** の観測日(2002/10/18)では水位は 9.26m となっているが，標高 9m 以下の部分を各地形データから抽出すると，DTM では水面形が衛星写真のものと合致しているが SRTM では乾季水面周りで空白部分が多く見られ水面形が合致しない．そこで湖周辺で無作為に選んだポイント上での標高データを**図 4-5** に示す．同図から SRTM の値は DTM の値よりも平均して 3-4m ほど大きく評価されている．この原因としては，SRTM が EGM96 ジオイド面からの値を標高として記録しているのに対し DTM がベトナムの Ha Tien 平均海水位を基準面として標高を記録しているが，両基準面の関係性が不明，また DTM の使用している回転楕円形が不明であり，両データ間でバイアスがかかってしまっているためと考えられる[2]．一方で DTM データは**図 4-3** からわかるように，湖南部~トンレサップ河でデータの乱れが見られる．

そこで再現計算に用いる地形データの作成においては，**図 4-5** にも示す以下の回帰式により，SRTM データにバイアス補正を施した．

$$\mathrm{SRTM_revised} = 0.9815 \times \mathrm{SRTM} - 3.2457 \qquad (4\text{-}1)$$

なお，SRTM データは整数値で保存されているため，8 近傍平均を用いてスムージング処理施した．

### 4-1-3. 計算条件

先述したとおり本モデルでは直交格子型の正方形格子を使用するため，すべてのセルサイズが同一となる．湖が幅数十 km スケールであるのに対し，トンレサップ川では狭小部で約 400m 幅のスケールとなっている．そこで計算コストを抑えた上で川の通水能力をモデルでも再現できるように，格子幅を$\Delta x = \Delta y = 250m$とした．時間刻み幅に関しては$\Delta t = 10s$とした．

また，**2-2-3** 節で述べた外力項の計算に必要な各パラメータ値は**表 4-1** に示す値を用いた．

## 4-2. 計算再現精度検証用データ

計算結果の再現精度確認用データとして，トンレサップ川中間部(Kompong Chhnang 地点)，湖南部(Kompong Luong 地点)における水位データを使用する．**図 4-6** に下流端(Prek Kdam 地点)の水位データと合わせて 3 地点間の水位比較を示す．雨季に入る直前(5 月)では 3 地点間で水位差は小さいが，雨季に入り全体的に水位が上昇し始めると下流端側から水位勾配が生じ，メコン川から湖方向への流れになるので水位勾配を持って湖への流入が生じている．本計算期間はその水位上昇期の半ばから水位ピーク，さらに水位降下期の半ばまでと

なっている．

また，水面形の変動を確認できるデータとして，アメリカ地質調査所(USGS)で配布されている Landsat 衛星画像を用意した[3]．今回は 1999 年に打ち上げられた Landsat7 号機の ETM+(Enhanced Thematic Mapper, Plus)センサーで撮影された画像を使用した．**表 4-2** にその諸元[4]を示す. 表に示すように Landsat7 号機は回帰日数 16 日周期で同じエリア上に飛来している．そこでトンレサップ湖南部~メコン川エリアにおける衛星画像のうち，水 2002 年 6 月から 12 月の雲によるノイズが少ない画像を**図 4-7** に示す．6 月当初は水面が乾季時と変わらないが, 7 月になるとメコン川から氾濫が生じ始め 9 月には湖も大きく氾濫している．やがて 10 月に入り水位が低下し始め，12 月になると湖は多少氾濫したままであるが，メコン川では乾季時に近い形まで戻っている．

本章では水面形変動の再現性を確認するために，画像撮影日の水域を抽出する手段として正規化水指数 NDWI(Normalized Difference Water Index)を用いた．竹内ら[5]によると，以下の式により NDWI は定義される．

$$NDWI = \frac{VIS - SWIR}{VIS + SWIR} \tag{4-2}$$

ここに$VIS$は可視光，$SWIR$は短波長赤外を意味する．水の分光反射率は可視で最大値をとり，短波長赤外で最小となる．ETM+センサーにおいて，**表 4-2** に示すマルチスペクトルの波長から短波長赤外はバンド 5 の値を使用するが，可視光にはバンド 1~3 の値を使用することができる．そこでトンレサップ湖の乾季, 雨季において土地被膜別に無作為に抽出した地点の NDWI 値をバンド組み合わせ別に**図 4-8，図 4-9** に示す．まず乾季において，水域エリアはどの組み合わせにおいても植生エリア, 農地エリアよりも高い NDWI が検出され，水域を容易に判断することができる．次に雨季において，水域はナチュラルカラー画像からわかるように判断されにくい．そこで水域エリアは暗色部(水深が深い部分)，明色部(水深が浅い部分)，湿地部(乾季は植生域だったが氾濫により浸水林となった部分)と分類した．その結果，乾季同様水域エリアは植生エリア，農地エリアに比べ高い NDWI が検出されるがバンド 2，3 を可視光とした場合では湿地部が明色部よりも低く検出される．これらの湿地部は明色部よりも標高が低く湖氾濫により生じた部分と考えられるが，明色部は農地エリアに水が薄く溜まっている状況と考えられ，湿地部の方がより深い水域である可能性が高い．そこでこの 2 ケースよりも湿地部の NDWI が高く検出されるバンド 1 を可視光とした．そのため NDWI 値は(4-2)式から以下の式に改めた．

$$NDWI = \frac{Band1 - Band5}{Band1 + Band5} \tag{4-3}$$

また，この(4-3)式から計算される NDWI を用いて観測水位データとの比較を行った．**図 4-10** に湖南部の Kompong Luong 地点近辺における陸域(図中 A)から沿岸部(図中 B)にかけての標高断面図と，断面に沿った(4-3)により計算される NDWI 値を示す．実際にこの時点の観測水位(9.66m)と標高の重なる地点(図中 C)で NDWI が 0.4 程度となっており，それより

高い位置の明色部において暗色部，浸水部よりも低い NDWI 値が検出されている．そのため NDWI=0.4 を閾値として，衛星画像からこの値より大きいエリアを水域として判定できると考えられる．

## 4-3. 計算結果

以上の計算条件，入力データを元にトンレサップ湖並びに周辺の流入河川，唯一の流出河川であるトンレサップ川を対象範囲として流動再現計算を行った．ここではその精度確認として，水位データを用いて水位変動の再現性，衛星画像データを用いて水面形変化の再現性を確認する．

### 4-3-1. 水位変化

**図 4-11** に各地点における水位の観測値と計算値の比較を示す．観測値と計算値を比較すると，水位上昇期において計算値でも水位勾配を持ちつつ水位上昇しており，観測値同様 10 月初旬にピークを迎え水位差が最小となり，そこから流量･水位勾配が逆転し水位低下が始まっている．

また，**図 4-12** に Kompogn Chhnang 地点，Kompong Luong 地点における水位の計算値と観測値の比較を示す．トンレサップ川中間点の Kompogn Chhnang 地点では水位上昇期はよく再現されているが，ピーク以降の降下期については観測値と計算値の間で約 60cm 程度の差が生じている．一方湖南部の Kompong Luong 地点では水位上昇期･降下期ともに比較的良好な精度で水位変化が再現されている．

### 4-3-2. 水域変化

前節では，点データとして観測水位による計算再現精度の検証を行った．ここでは面的に水域の変化を確認するため計算された水面形変化並びに Landsat 衛星画像を用いて検証を行う．

**図 4-13** に計算水位分布，**図 4-14, 15** に計算流速分布の変化を 4 週間毎に示す．先に述べた水位変化の様子と同様に，水位上昇期には下流端側から氾濫が進行し蛇行しながら湖へと水が流入している．湖は比較的流速が小さく穏やかに水域が拡大している．10 月前後の水位ピーク時には領域全体で氾濫が生じており初期状態から水面形が大きく拡大している．一方水位降下期にはトンレサップ河の流向が逆転し，下流端側から水域の縮小が進んでいる．

以上のように，計算結果においても**図 4-7** に示した衛星画像と同様に水域が拡大･縮小し

ている様子が確認できた．そこでこの計算結果を(4-3)式により衛星画像から抽出された水域と比較し，水域変化の再現性について確認する．**図 4-16** に雲によるノイズが少なく鮮明に撮影されている衛星画像と NDWI で判断される水域，ならびに再現計算で得られた水陸境界を示す．同図において，左側はナチュラルカラー画像，右側は水色で示された部分が水域(NDWI>0.4)，実線が計算水陸境界である．計算水陸境界は抽出された水域と概ね合致しており，水域の拡大･縮小が再現できている事がわかる．ただし湖南部に比べ湖北部では氾濫域の一部が水域と判断されておらず，浸水林や浮草などが水の分光反射を遮り NDWI 値が低下していると考えられる．9/7 では水域の水際線は確認できるものの，11/26，12/12 と水域が縮小していくにつれて水際線が不明瞭になっていく．この点に関して，氾濫した植生域も水域と判断できるように今後 NDWI 評価基準を改める必要があると考えられる．

## 4-4. GPGPU による計算高速化の評価

先述した GPGPU による並列計算によって計算速度が非常に高速化され，長期間の計算が短時間で可能となった．ここではその GPGPU 化により本計算対象を計算する際に計算速度がどの程度高速化されたのか定量的に評価するため，CPU のみを用いた計算での計算時間と GPU を用いた計算での計算時間の比較を行った．また GPGPU 化においてより効率よく高速化するための手法についても述べる．

### 4-4-1. CPU 計算速度と GPU 計算速度の比較

CPU のみを用いた計算と GPU を用いた計算との場合で計算速度がどの程度高速化されたか評価を行った．計算に用いる CPU は Intel Core i7 920，GPU は Nvidia Tesla C 1060 とした．**表 4-3** に各諸元[6][7]を示す．なお計算時間の計測には CUDA ライブラリの CUDA Event API[8]を使用し，時間発展ループ中の各計算プロセスにかかる計算時間を計測した．

**図 4-17** に計算期間 6 時間分にかかる CPU による計算時間と GPU による計算時間を各計算プロセス別に，**図 4-18** にその速度比を示す．実際に GPU 内で処理される計算プロセスはすべて高速化されており，計算プロセス全体で約 50 倍の高速化となっている．特に保存量$\boldsymbol{w}$と h,u,v の変換プロセスは約 150 倍程度とより高速化されているが，このプロセスでは 1 つのスレッドが 1 つのセルを担当するのでサブドメイン間の影響を考慮せず，変換自体も非常にシンプルであるため GPU で効率よく処理されたためと考えられる．一方境界条件付与のプロセスでは計算対象セルが限定的であるため約 3 倍程度と効率よく高速化されなかった．

また，この条件では計算期間 6 時間(=2160 ステップ)に 1 回の出力を行っている．仮に毎ステップに CPU から GPU へのファイル転送と出力を行ったとするとファイル転送にかか

る時間が 0.017 秒から 36.5 秒に，出力にかかる時間が 4.43 秒から 9565 秒となり，計算時間が増大してしまう．そのためファイル転送及び出力回数はなるべく少なくするように出力時間を設定する必要がある．

### 4-4-2. GPU 計算速度の高速化

前節では GPU 計算が CPU 計算に比べ十分に高速化された．本節では GPU 計算に着目し，より高速に計算できる工夫について検討した．

- ブロックサイズの変更

  3-2-2 節で述べたように GPU 内では 1 つの SM が 1 つのブロックを担当し，32 スレッドを 1 単位(ウォープ:Warp)として処理を行っている．本モデルではブロックを重ね合わせてのりしろを作るように配置している．そのためブロックサイズを変化させると同じ計算領域でものりしろ部分のセル(スレッド)数が変化し，計算時間に差が生じる．ここではブロック幅 N_IN_BLOCK( N )に対し N=4,8,12,16,20 の 5 ケースでブロックサイズを変化させたときの計算時間について検討した．**表 4-4** に各ケースのブロック数，セル数等の設定値を示す．同表からわかるように，ブロックサイズを大きくするほどのりしろ部分が少なくなり，計算される総セル(スレッド)数も減少している．それと同時に 1 ブロックの計算に SM が行うウォープ数は増加している．**図 4-19** にケースごとの計算時間を示す．計算全体でみると，ブロックサイズを大きくするほど計算時間が短くなり，高速化されている事がわかる．ただし，陸域化プロセスや底面摩擦計算プロセス，$\boldsymbol{w}$と h,u,v の変換プロセスなどサブドメイン間の影響を考慮しないプロセスでは，N=16 のケースが N=20 のケースよりも計算が早い．

- バンクコンフリクトの解消

  計算プロセスのうち，フラックス計算，外力計算，時間発展更新，水域化のプロセスではシェアードメモリを使用している．シェアードメモリは 16 個のバンクによって構成されており，SM がシェアードメモリにアクセスするときは 16 スレッドずつ処理される．各バンクには一度に 1 箇所しかアクセス出来ないため，複数のスレッドが同じバンクの複数の箇所にアクセスすると順番に処理され時間がかかってしまう．これを「バンクコンフリクト」と呼ぶ．そのため**図 4-20** のようにスレッドが連続的にメモリにアクセスできるようにすることでバンクコンフリクトを防ぐことが望ましい．しかしシェアードメモリ上で 2 次元配列 f_sh[16][16] を用いると，スレッド[threadIdx.x][threadIdx.y]がメモリ[threadIdx.x][threadIdx.y]に 16 要素ごとにメモリにアクセスするので**図 4-21** では threadIdx.y=0 のスレッドが全て Bank0 のメモリにアクセスしてしまい，「16-way バンクコンフリクト」が生じてしまう．そこで配列確

保を f[16][17]とすると，シェアードメモリの使用量が増加してしまうが 17 要素ごとにメモリにアクセスできるので，**図 4-22** のように各スレッドがすべて別のバンクにアクセスするため効率よく処理する事ができる．

以上のようにシェアードメモリの配列確保を

__shared__ DATA_TYPE h_sh[N_IN_BLOCK][N_IN_BLOCK+1];

とすることでバンクコンフリクトを解消することができる．**図 4-23** は N_IN_BLOCK=16 とした時の，各プロセス別のバンクコンフリクト解消前後での計算時間比較である．どのプロセスでも計算速度は向上し，プロセス全体では 30%ほど高速化することができた．

## 第 4 章 参考文献


1. 藤井 秀人: メコン河カンボジア氾濫域の水文観測と水収支，農業工学研究所技報，第 202 号，pp.127-140, 2004
2. 標高データの管理，ESRI JAPAN: http://desktop.arcgis.com/ja/arcmap/10.3/manage-data/raster-and-images/wkflw-elevation-part1.htm
3. USGS, LandsatLook Viewer: https://landsatlook.usgs.gov/viewer.html
4. 宇宙技術開発株式会社，ランドサット(Landsat)衛星・センサの概要・諸元: http://www.sed.co.jp/sug/contents/satellite/satellite_landsat.html
5. 竹内 渉，安岡 善文: 衛星リモートセンシングデータを用いた正規化植生，土壌，水指数の開発，写真測量とリモートセンシング，第 43 巻，6 号，pp.7-19, 2004
6. 環境総合研究所，パソコン CPU の浮動小数点倍精度演算速度，http://eritokyo.jp/independent/aoyama-col8025.html
7. NVIDIA, Tesla C1060 コンピューティングプロセッサー，http://www.nvidia.co.jp/object/tesla_c1060_jp.html
8. J. Sanders, E. Kandrot，株式会社クイープ: CUDA by Example 汎用 GPU プログラミング入門，インプレスジャパン，2011

# 第4章
# 図表集

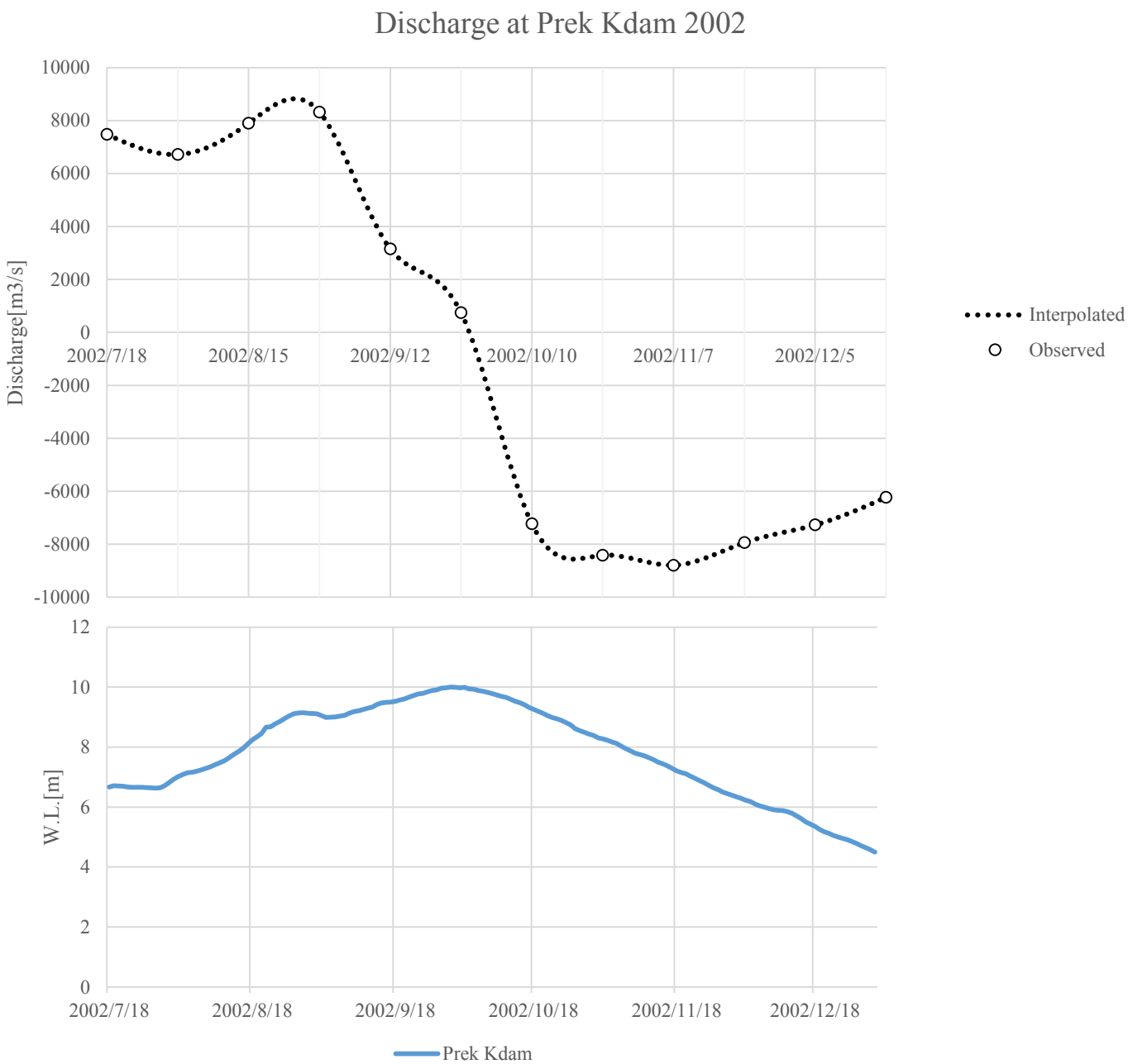
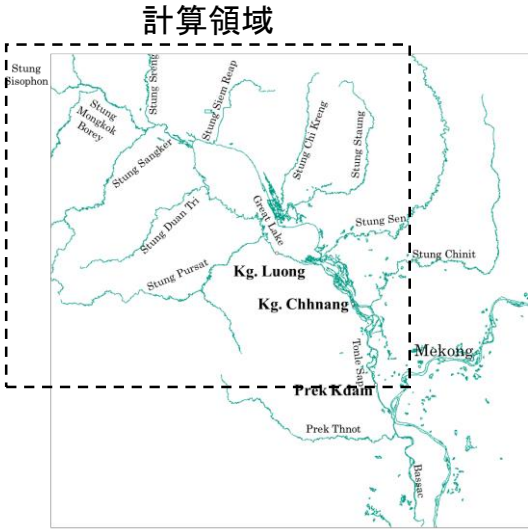
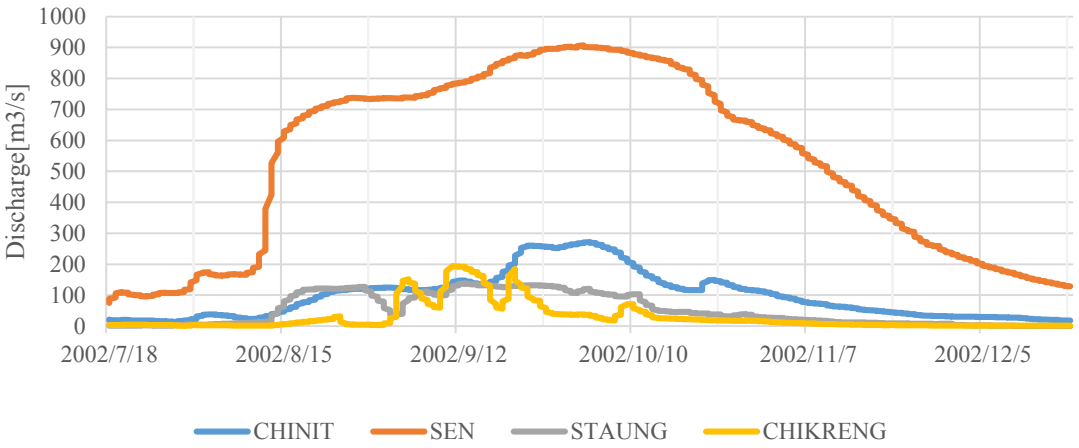
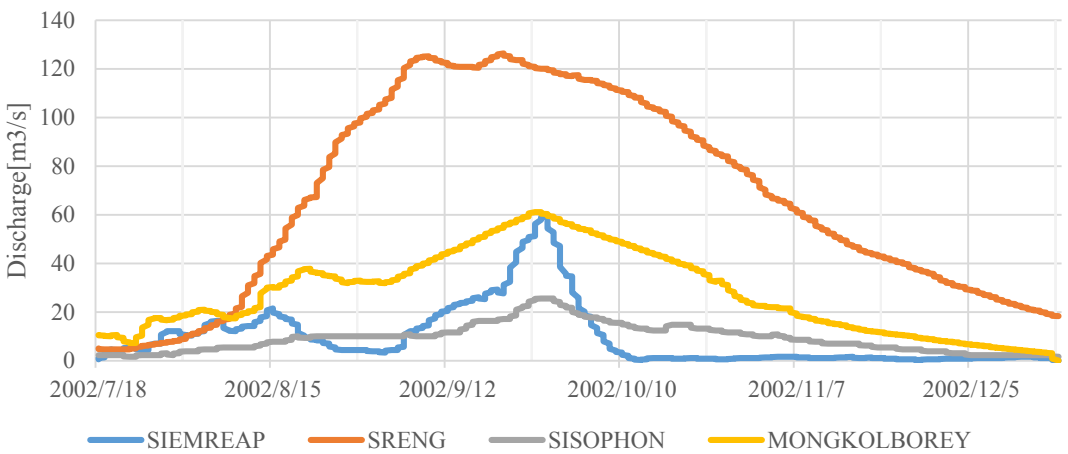



図4-1 Prek Kdam地点観測流量･水位



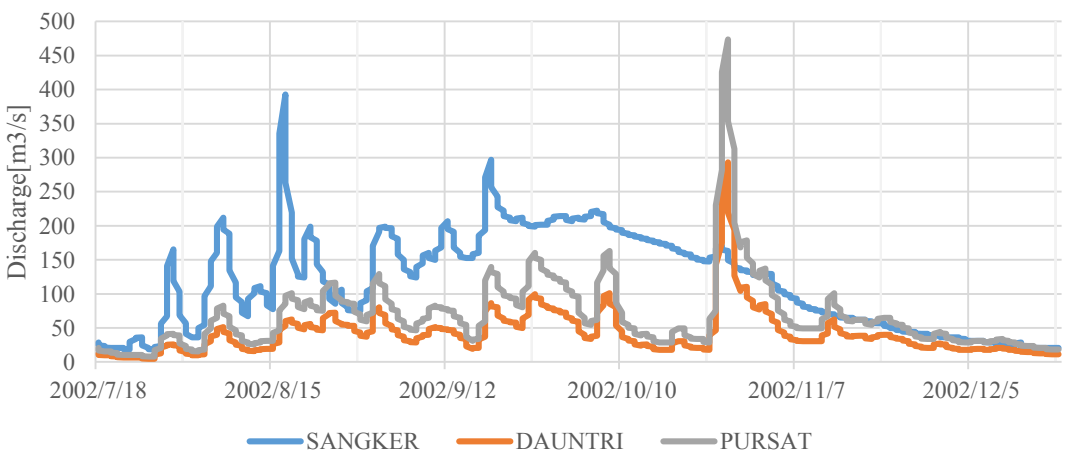
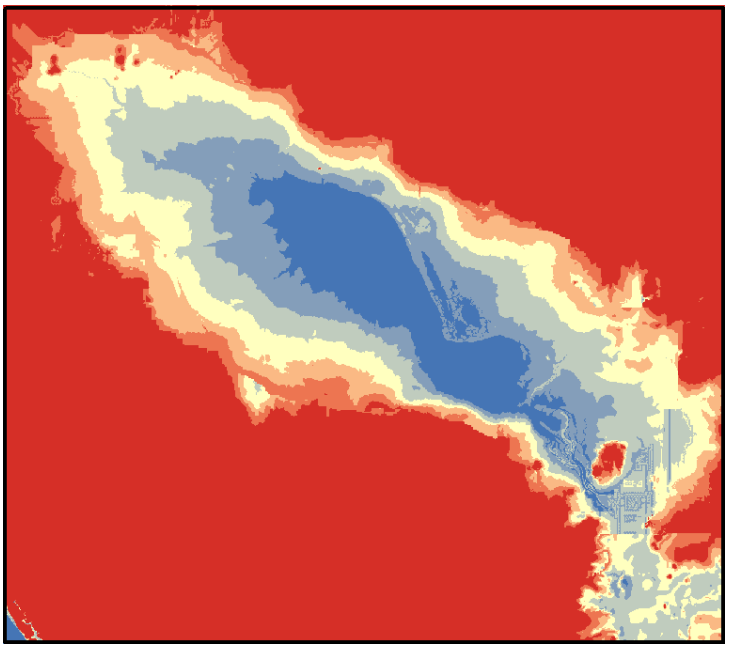
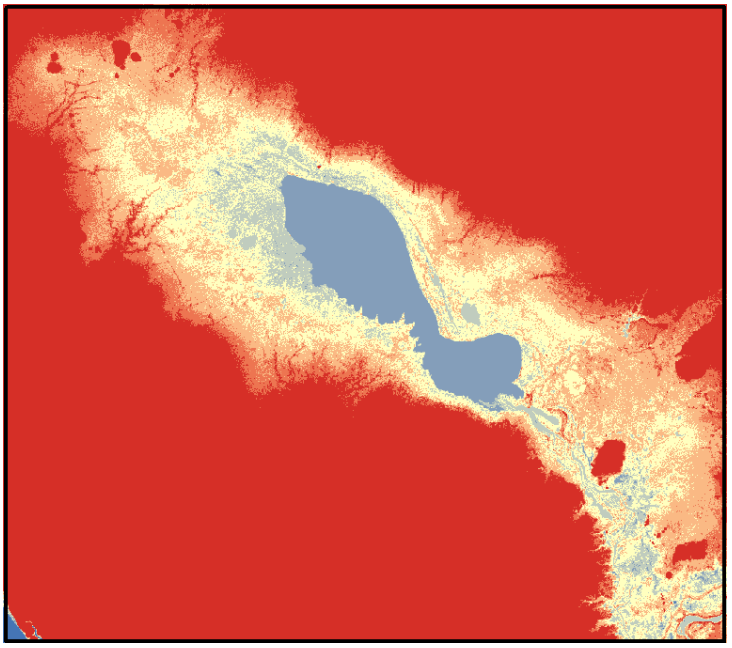




図4-2 MRC上流流量データ

DTM



SRTM



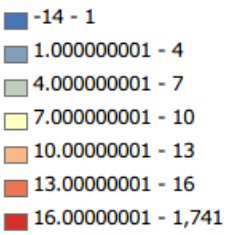
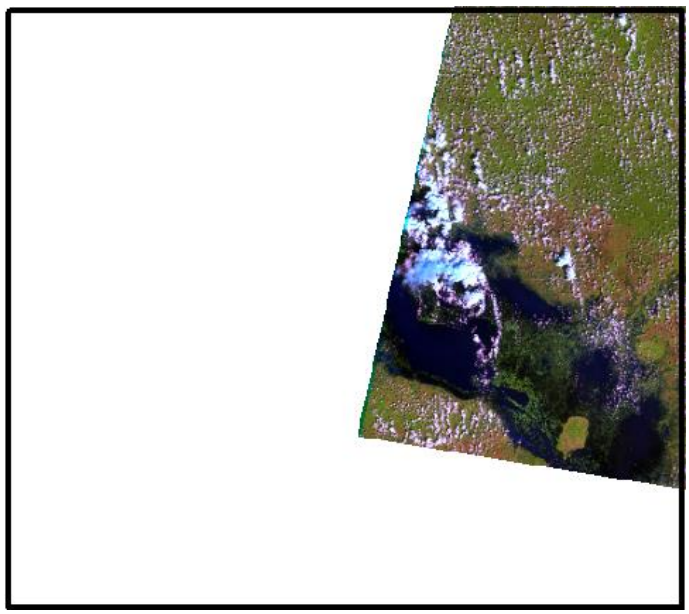



図4-3 標高データ

Landsat衛星画像
(2002/10/18,
Kompog Luong水位9.26m)



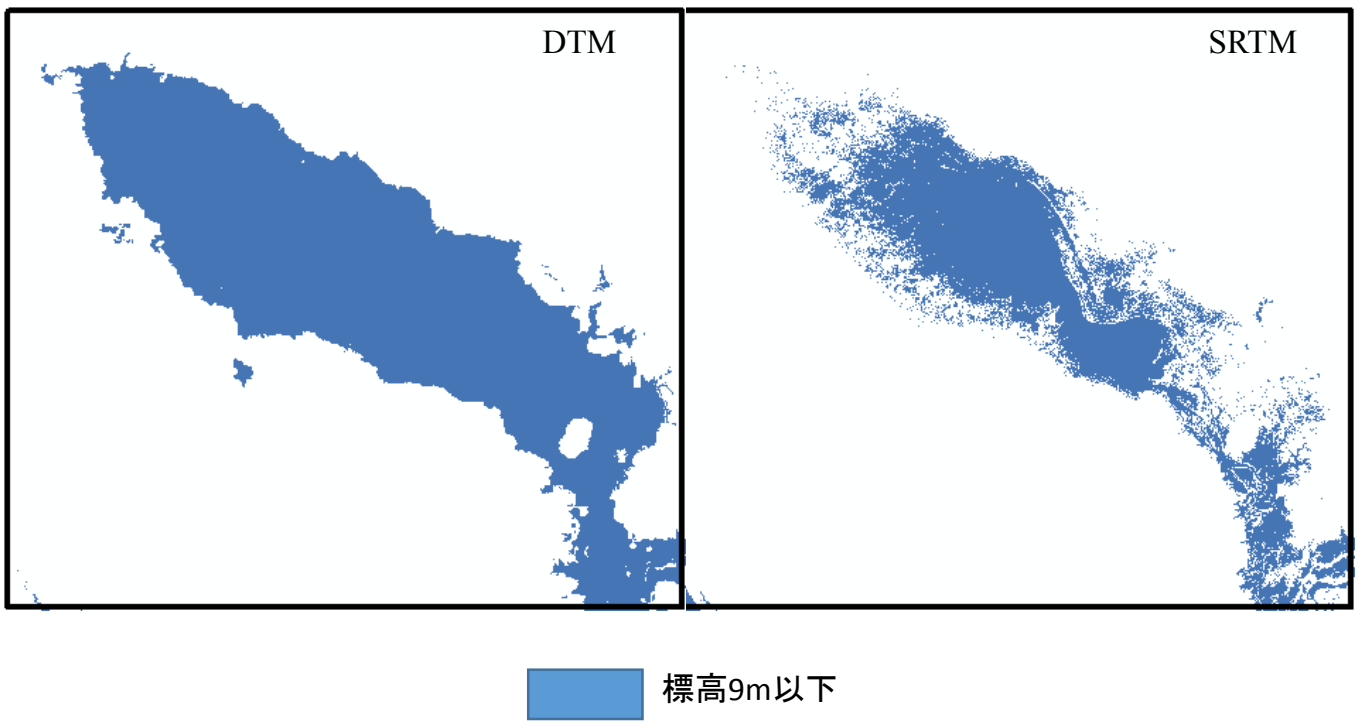



図4-4 標高データと衛星画像の比較

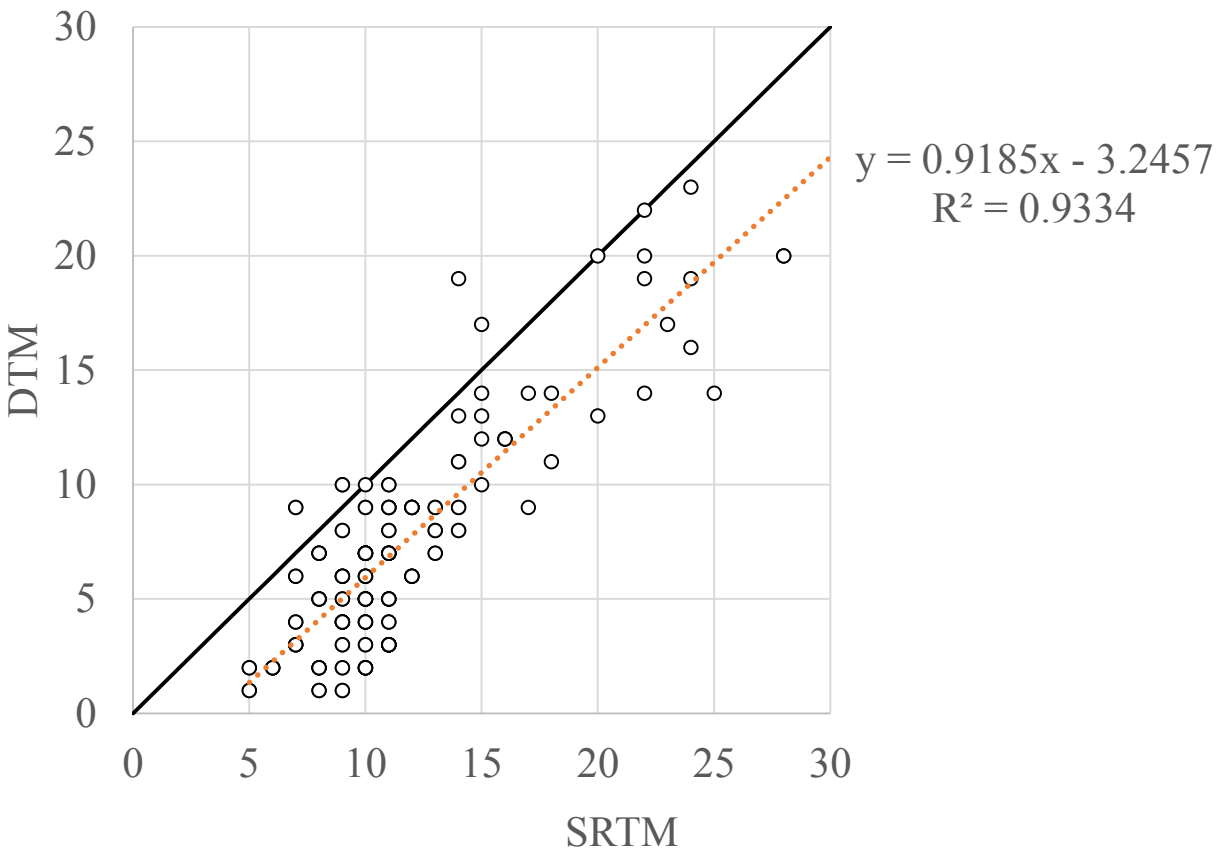



図4-5 SRTMデータとDTMデータの比較

| 記号 | パラメータ | 設定値 | 単位 |
|---|---|---|---|
| $g$ | 重力加速度 | 9.8 | $\mathrm{m \cdot s^{-2}}$ |
| $f_c$ | コリオリ力 | $3.16 \times 10^{-5}$ | $\mathrm{s^{-1}}$ |
| $n$ | マニング係数 | 0.03 | $\mathrm{m^{-1/3} \cdot s}$ |
| $\nu$ | 渦動粘性係数 | 3.39 | $\mathrm{m^2 \cdot s}$ |

表4-1 外力計算パラメータ

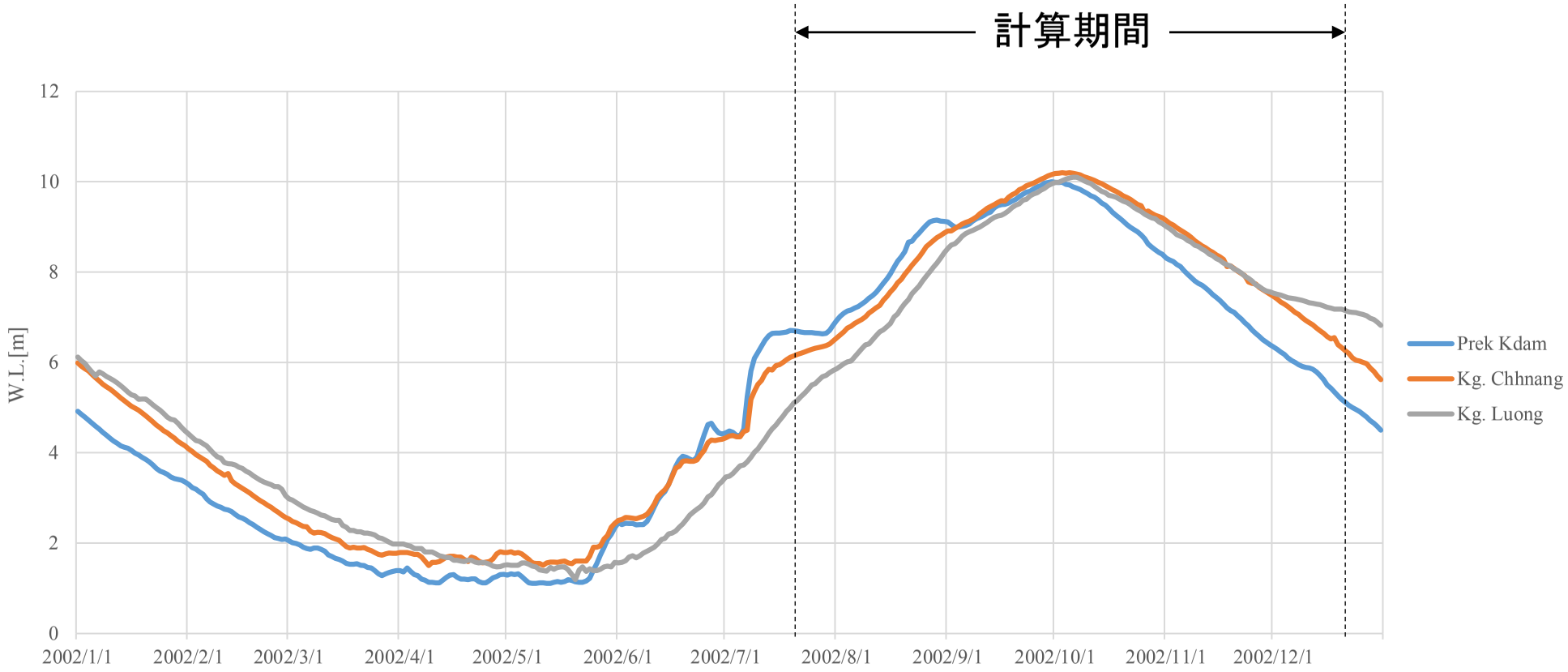

計算期間
W.L.[m]
12
10
8
6
4
2
0
2002/1/1
2002/2/1
2002/3/1
2002/4/1
2002/5/1
2002/6/1
2002/7/1
2002/8/1
2002/9/1
2002/10/1
2002/11/1
2002/12/1
Prek Kdam
Kg. Chhnang
Kg. Luong

図4-6 観測水位

| | |
|---|---|
| 打ち上げ | 1999年4月15日 |
| 軌道高度 | 705km<br>（太陽同期準回帰軌道、回帰日数16日、約99分／1周回） |
| マルチスペクトルの波長 | band1 青 450-520nm<br>band2 緑 530-610nm<br>band3 赤 630-690nm<br>band4 近赤外 750-900nm<br>band5 中間赤外 1550-1750nm<br>band6 熱赤外 10400-12500nm<br>band7 中間赤外 2090-2350nm |

表4-2 Landasat7号機 ETM+センサーの諸元

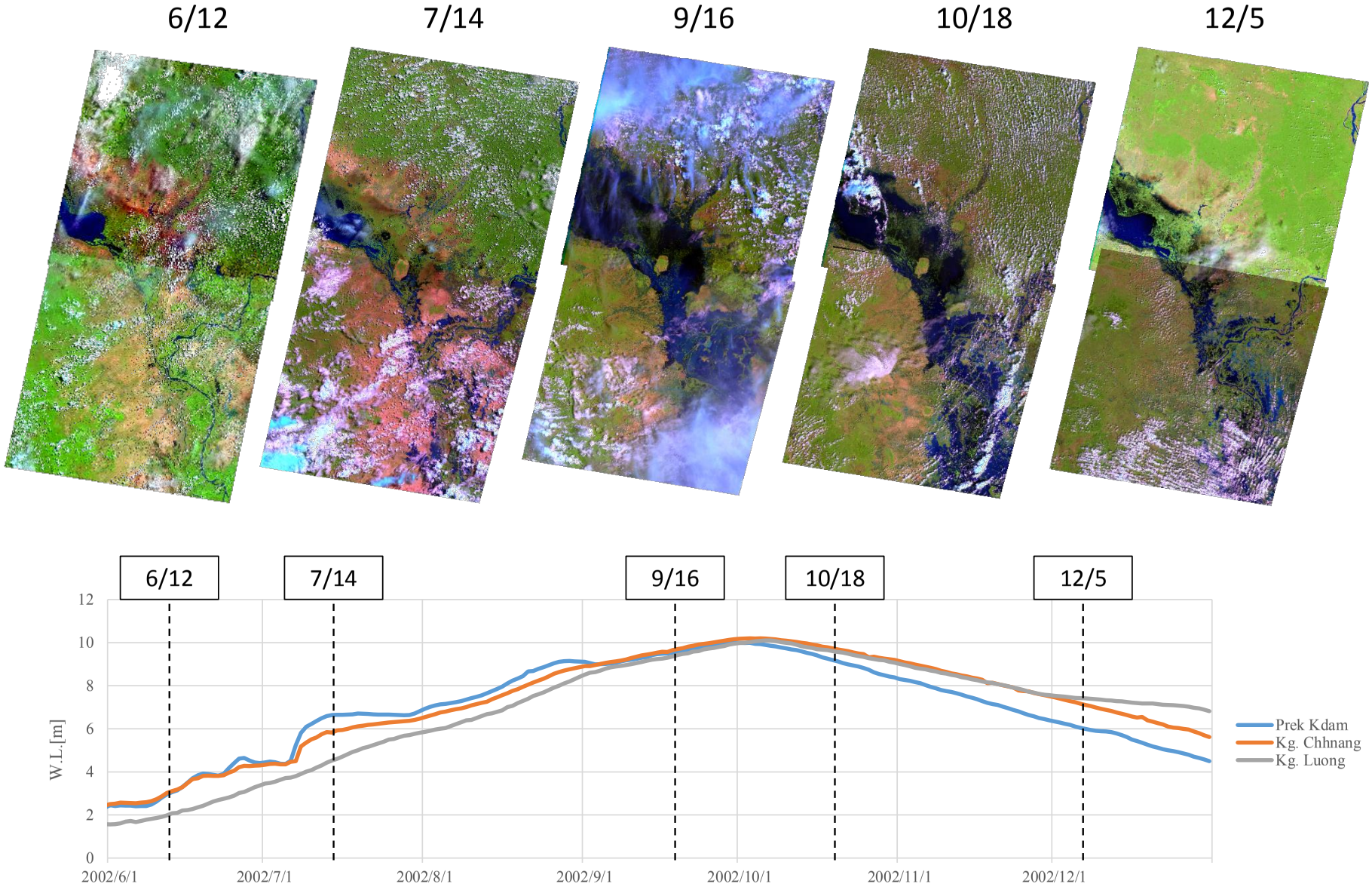
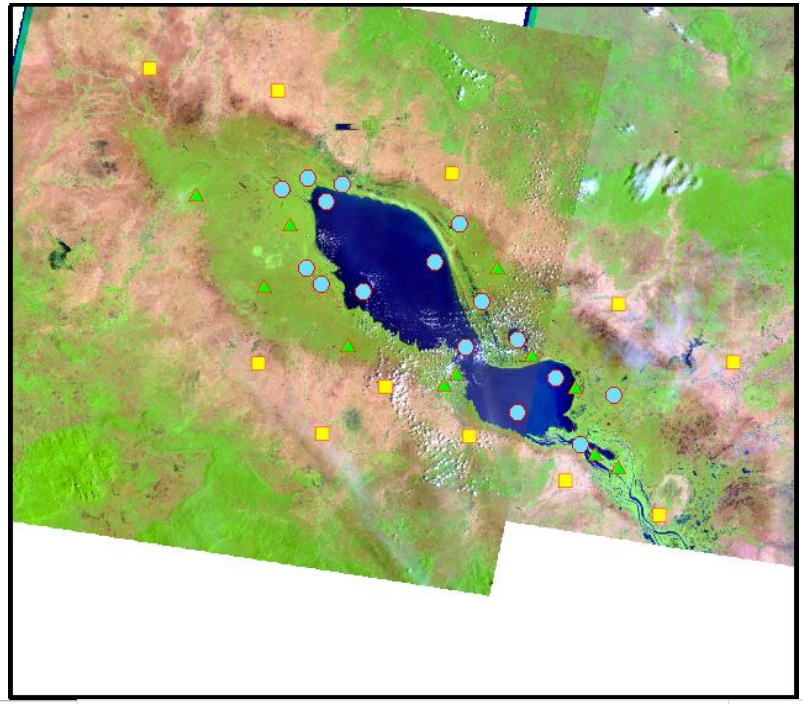



図4-7 衛星画像による水域変化



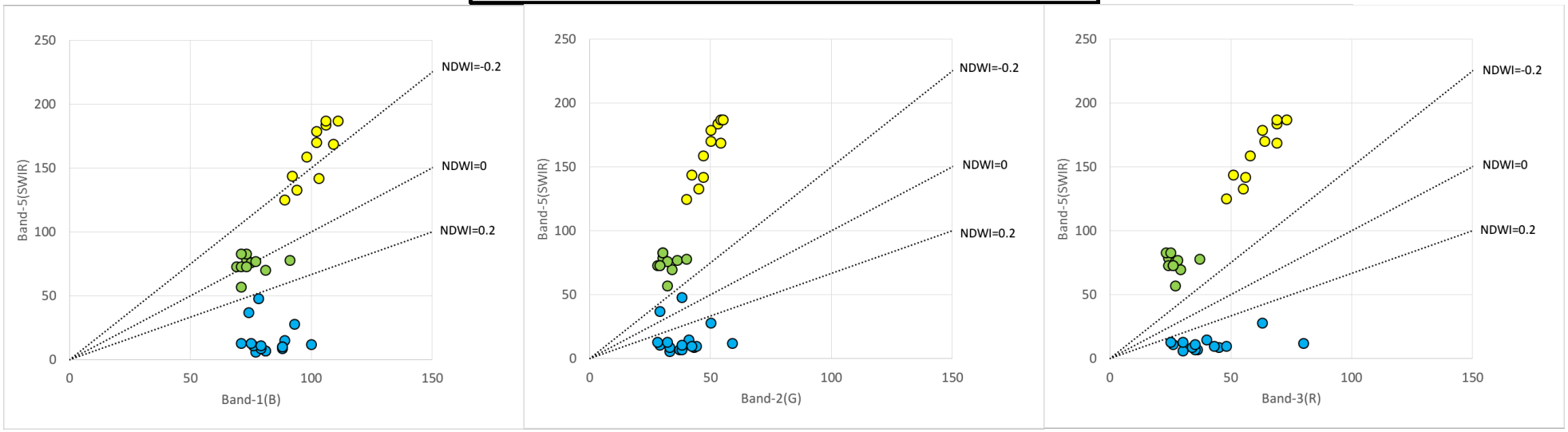
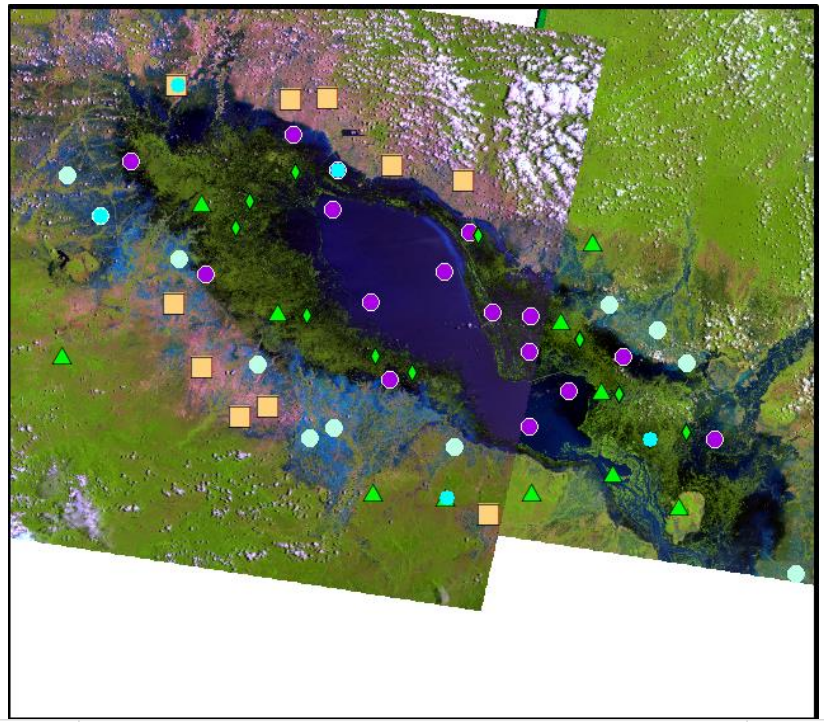



図4-8 乾季NDWIプロット



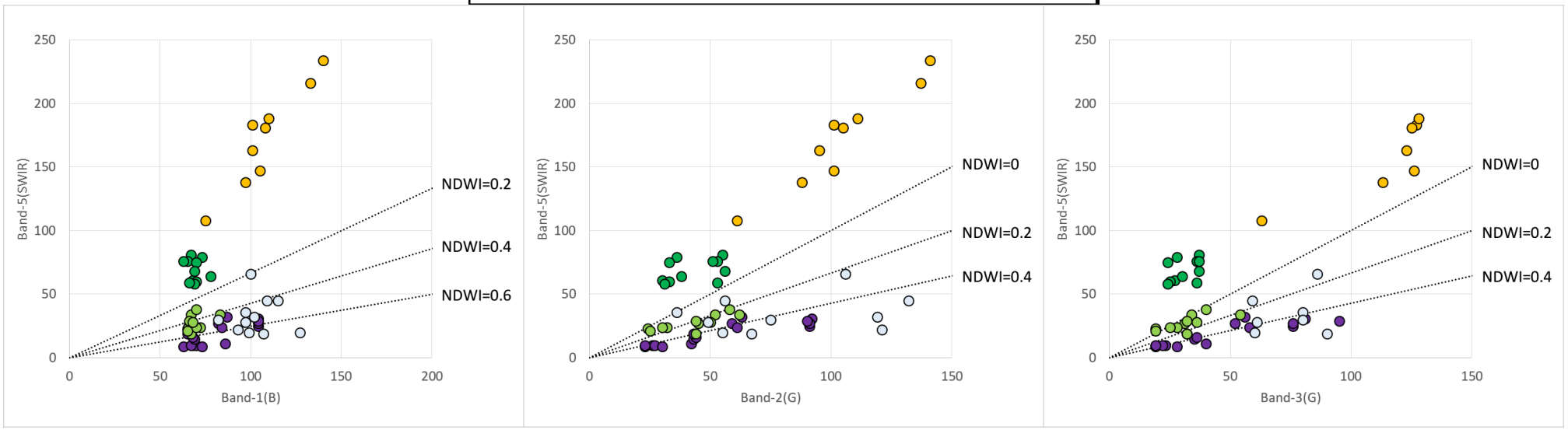
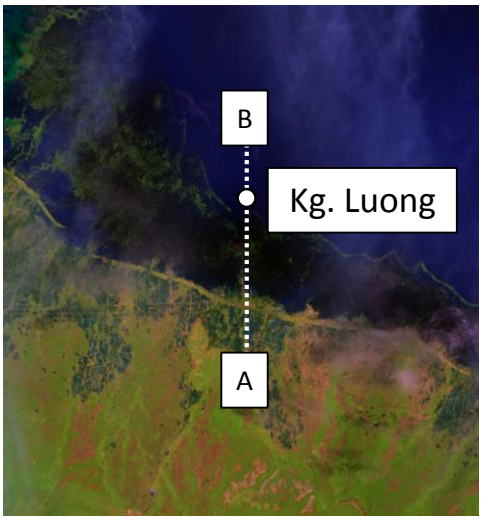




図4-9 雨季NDWIプロット



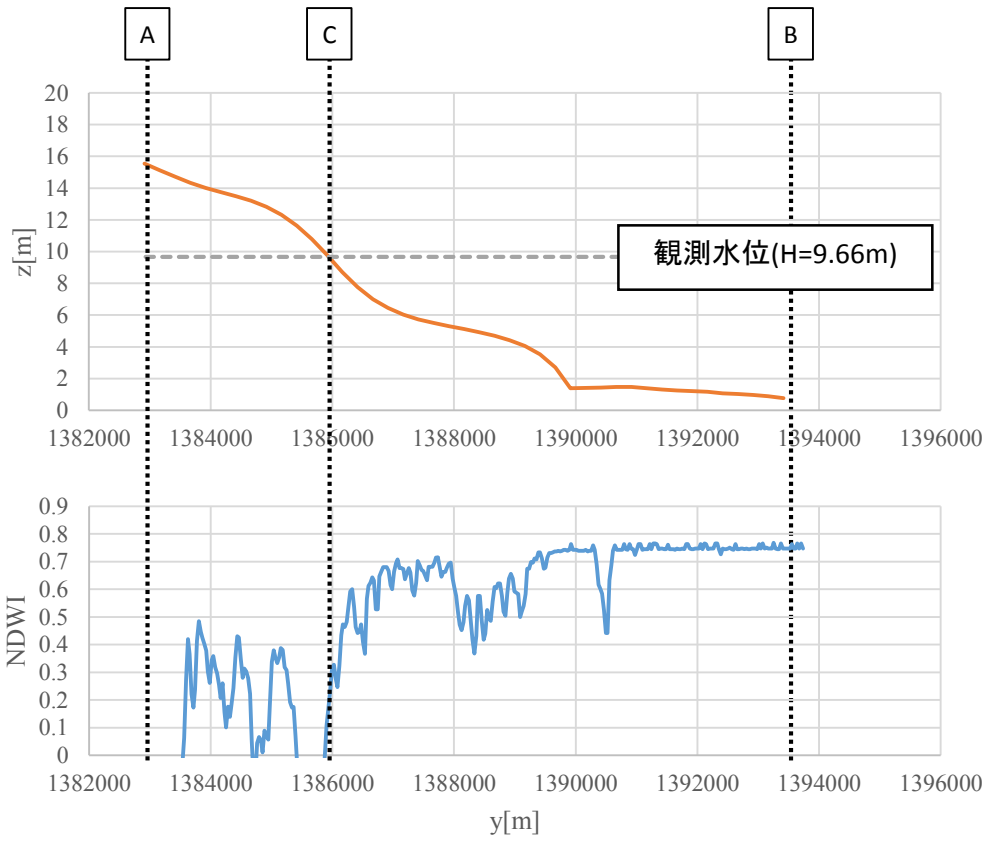
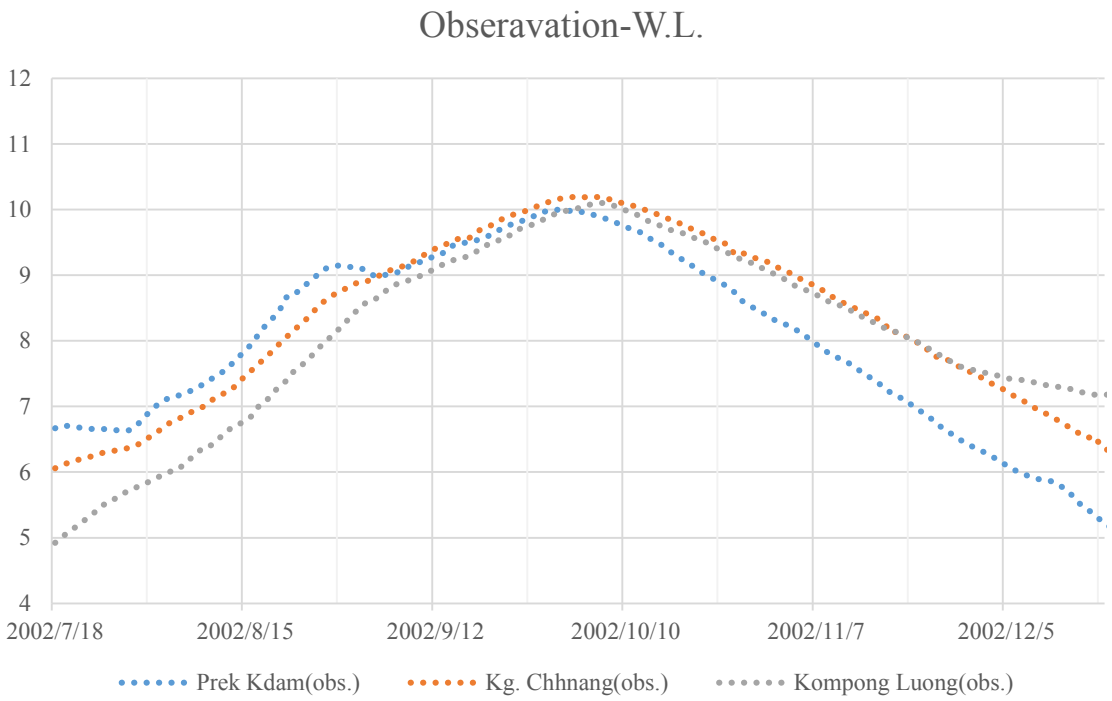




図4-10 Kompong Luong 地域での標高断面とNDWI



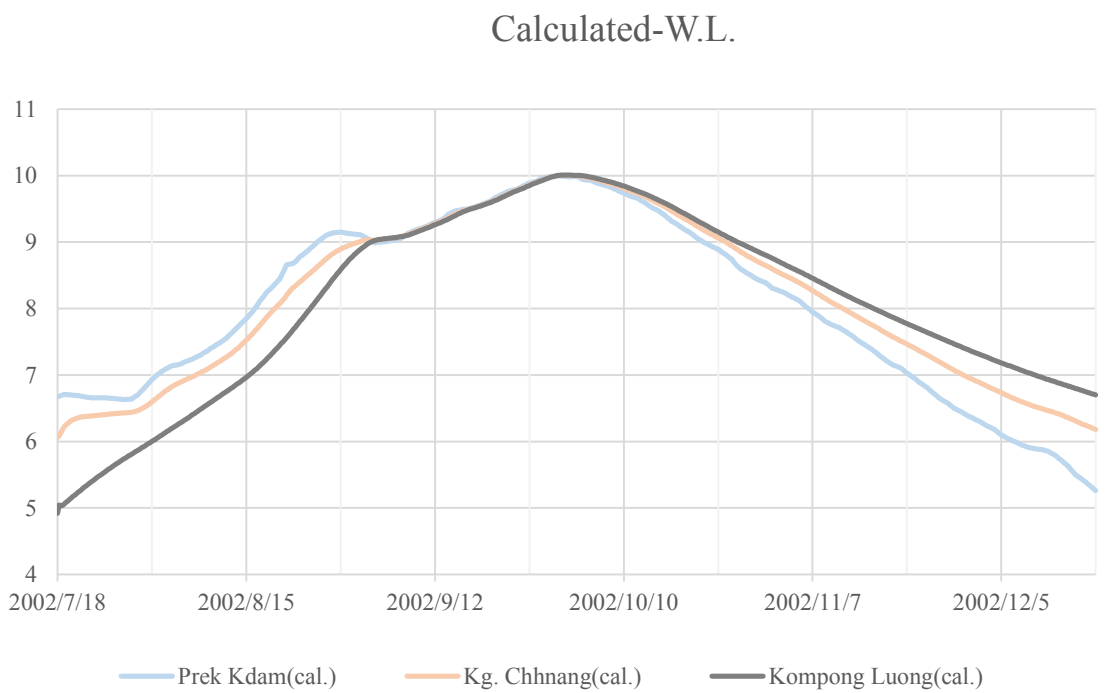
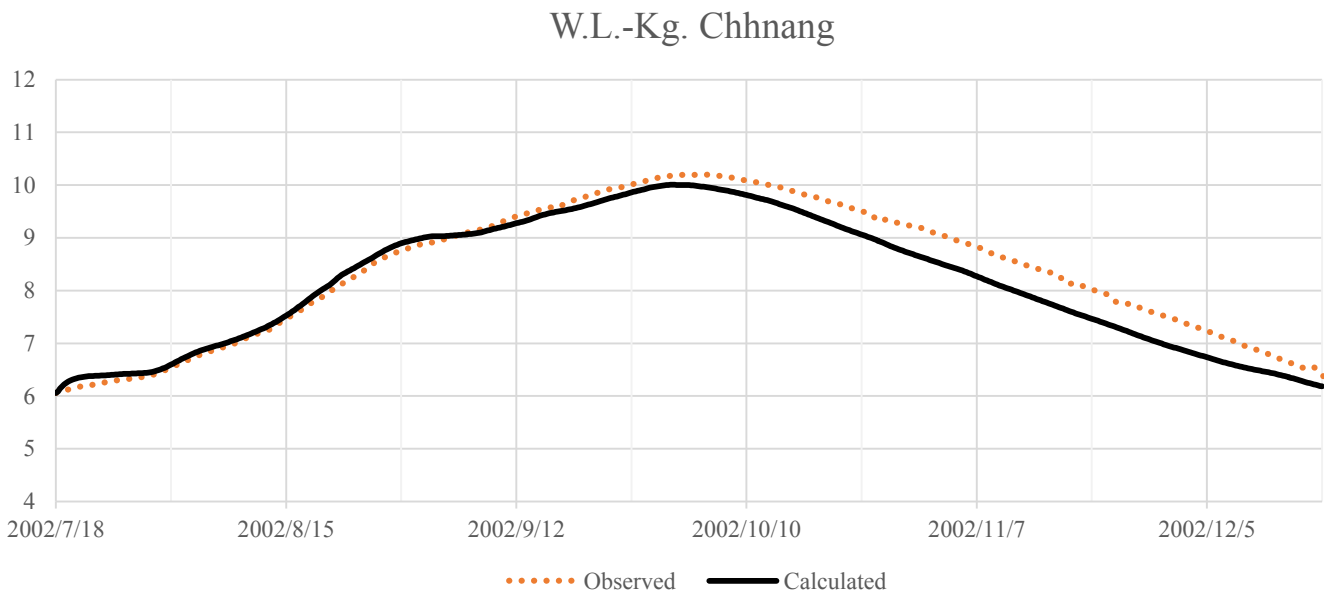




図4-11 水位の計算値と観測値の比較



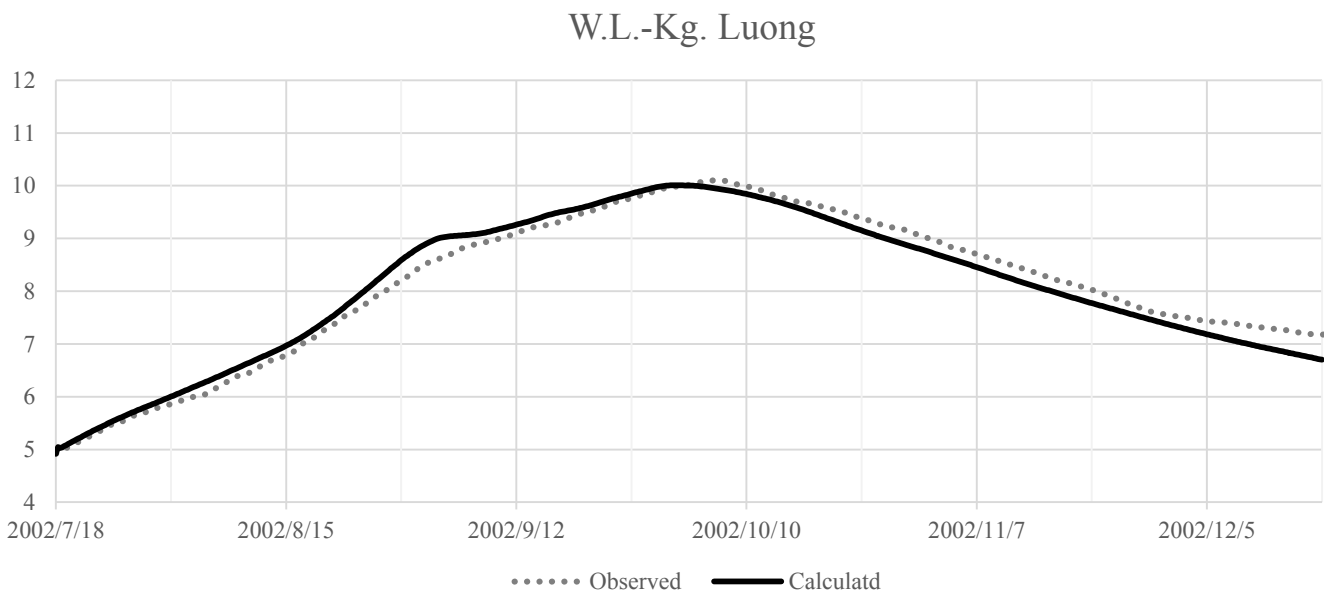




図4-12 Kg. Chhnang, Kg. Luongでの水位の計算値と観測値の比較

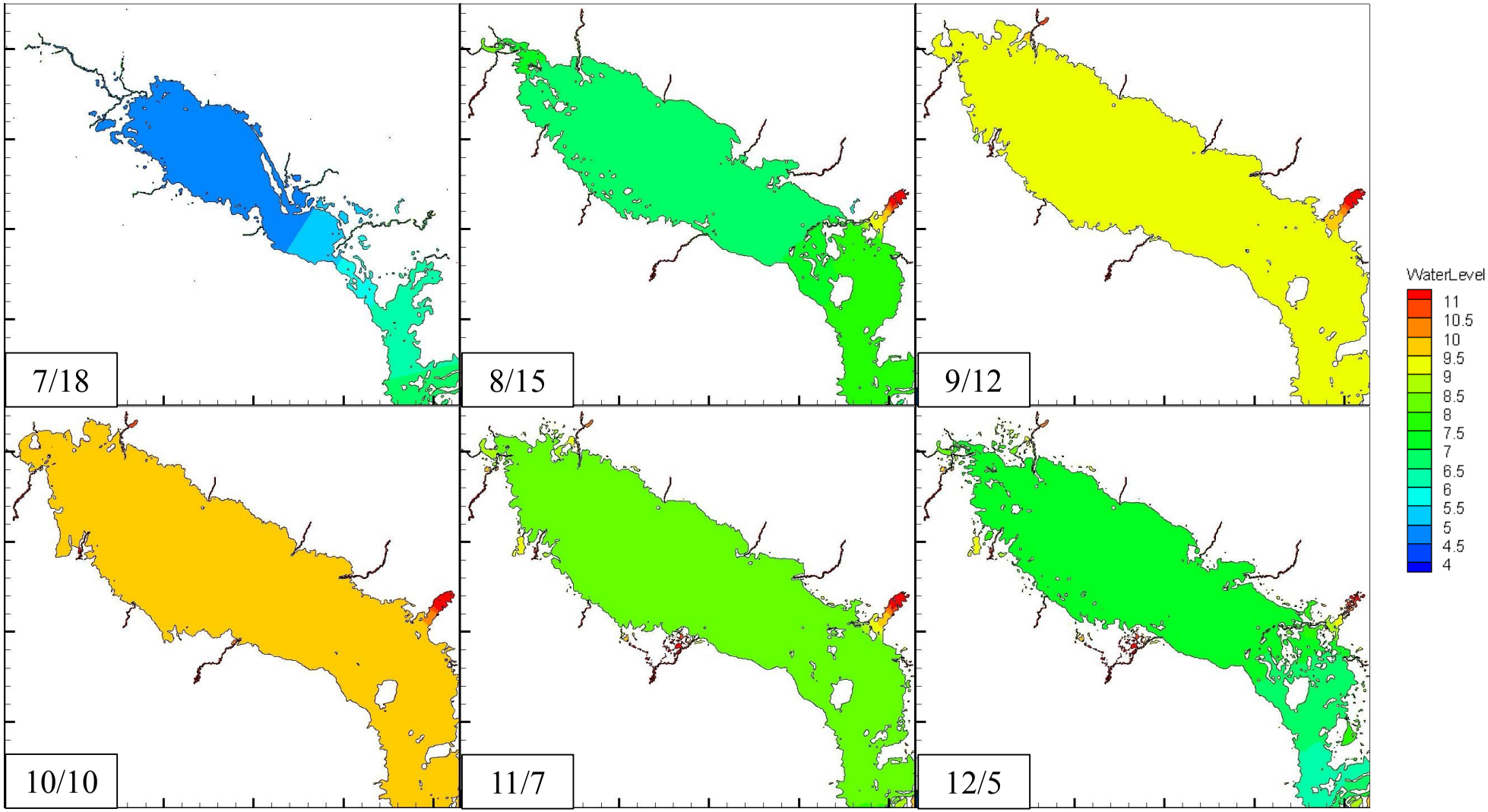



図4-13 計算水位分布変化

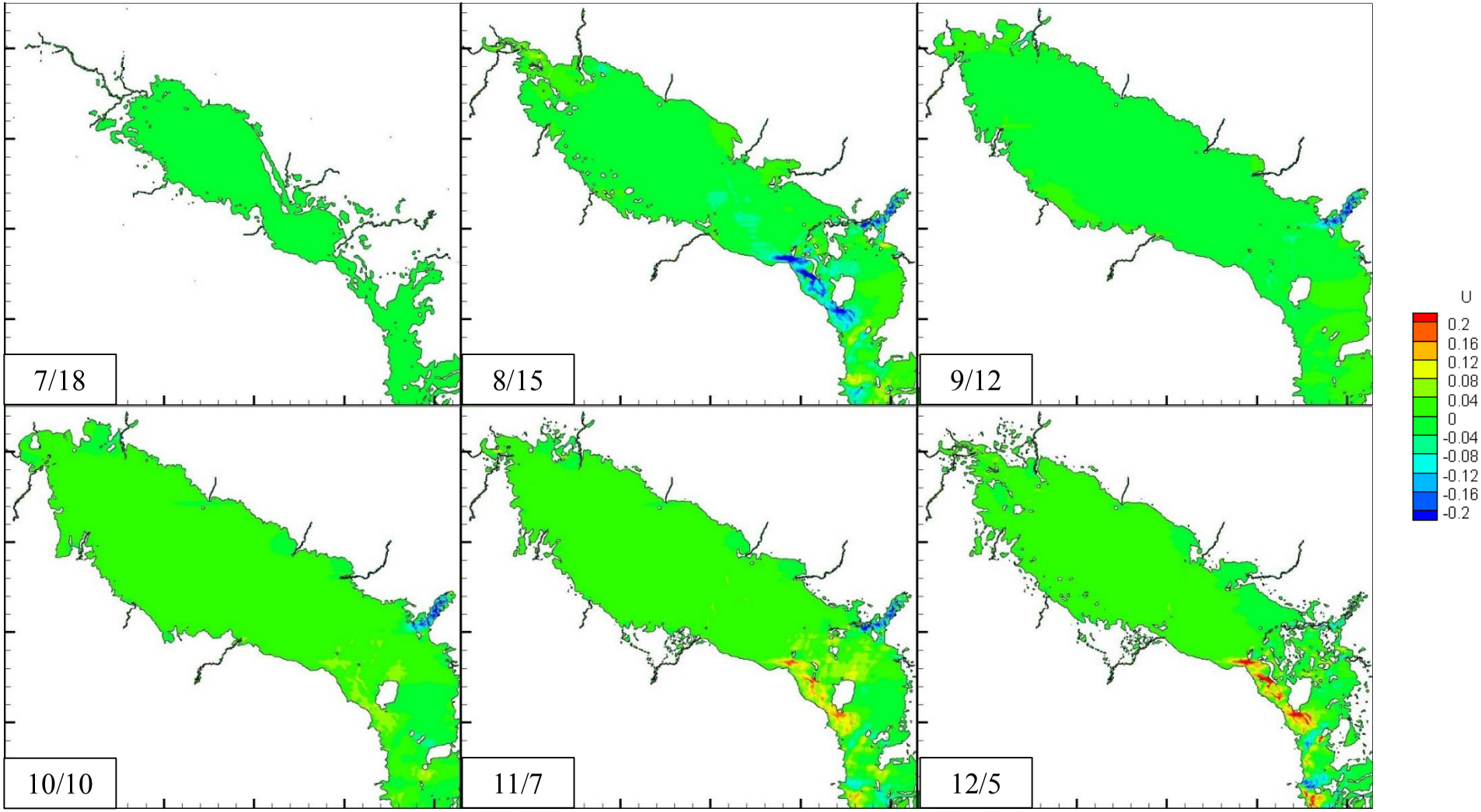



図4-14 計算x方向流速分布変化

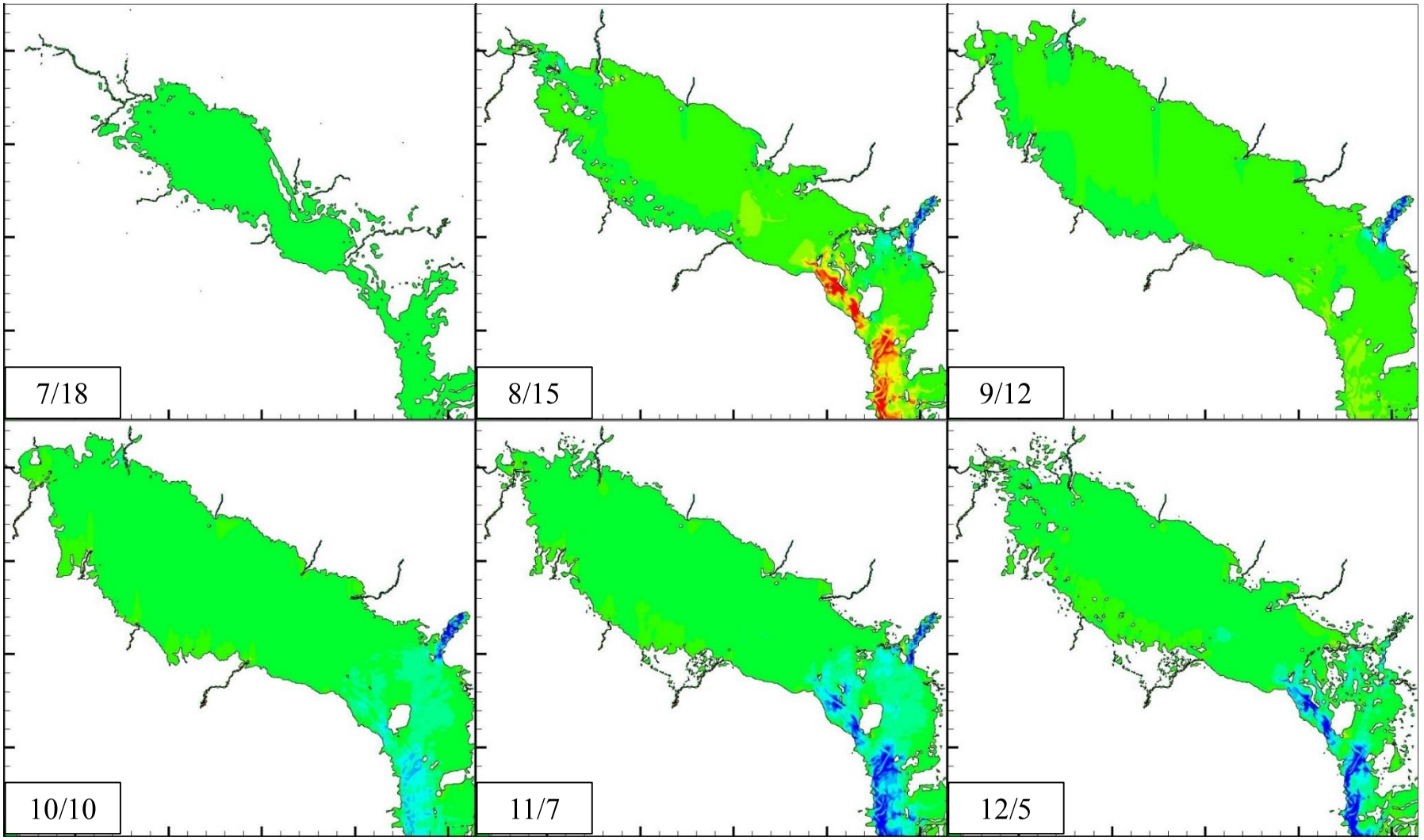



図4-15 計算y方向流速分布変化

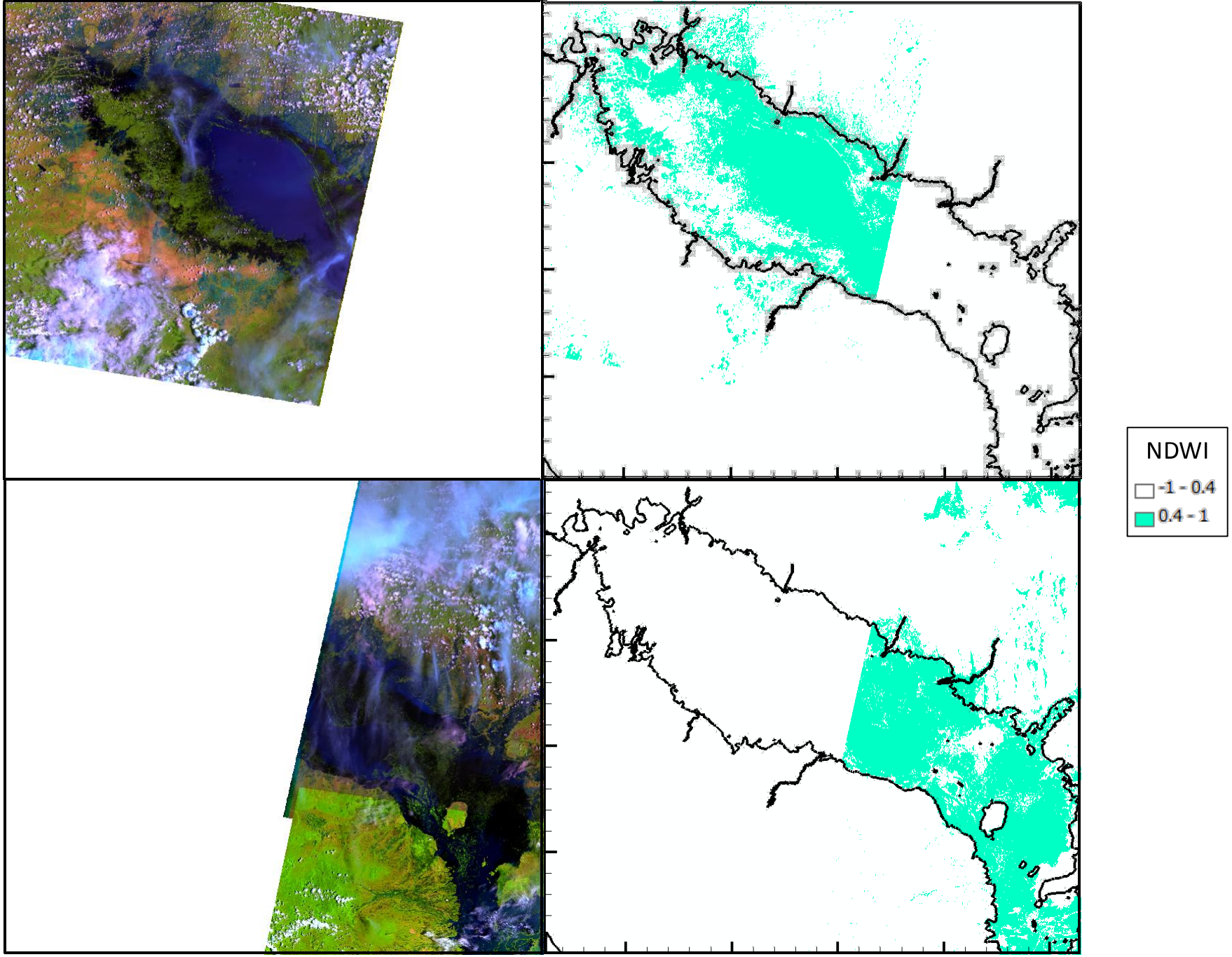



図4-16 Landsat衛星画像から抽出された水域，計算水陸境界線

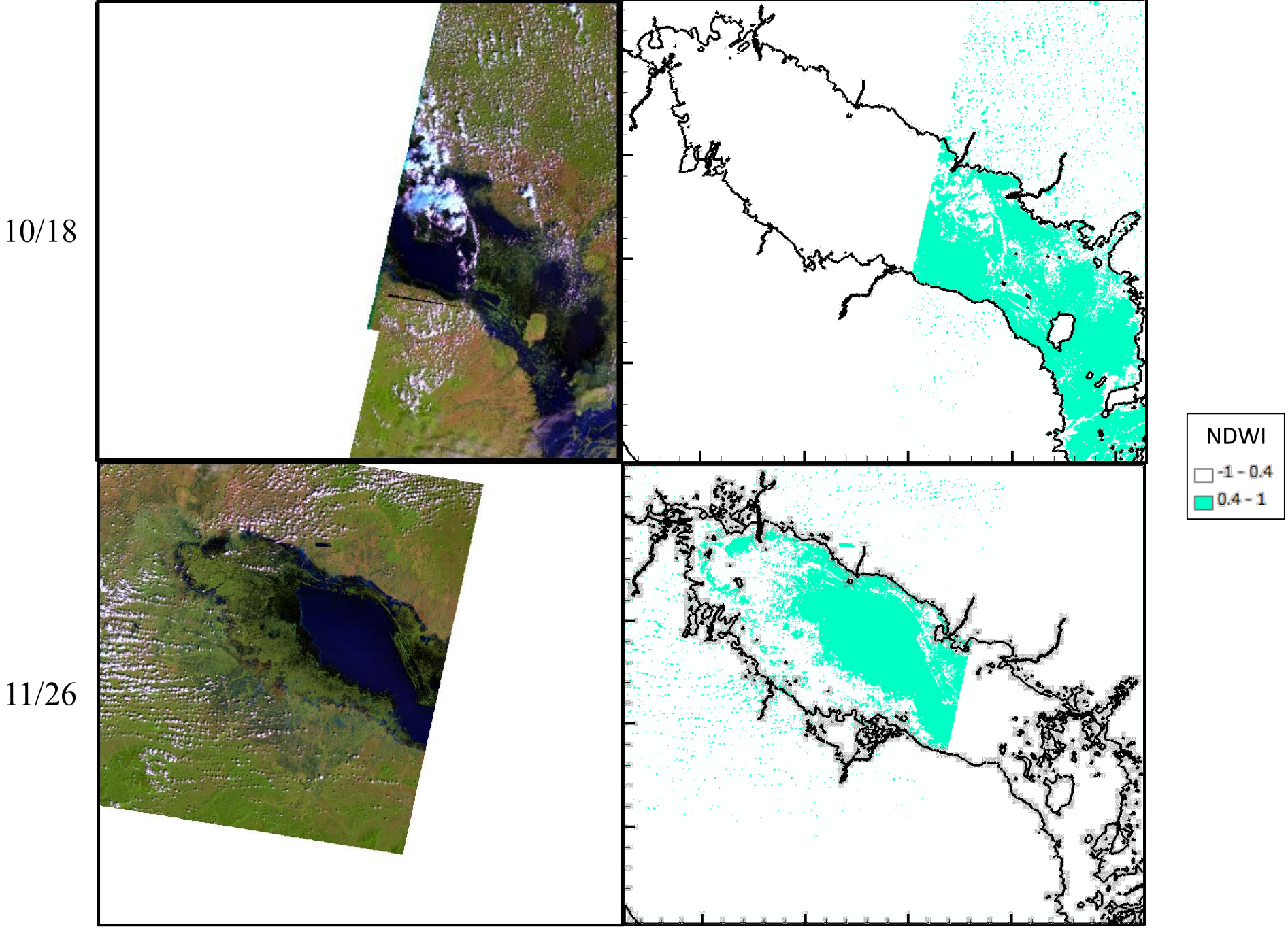



図4-16 Landsat衛星画像から抽出された水域，計算水陸境界線

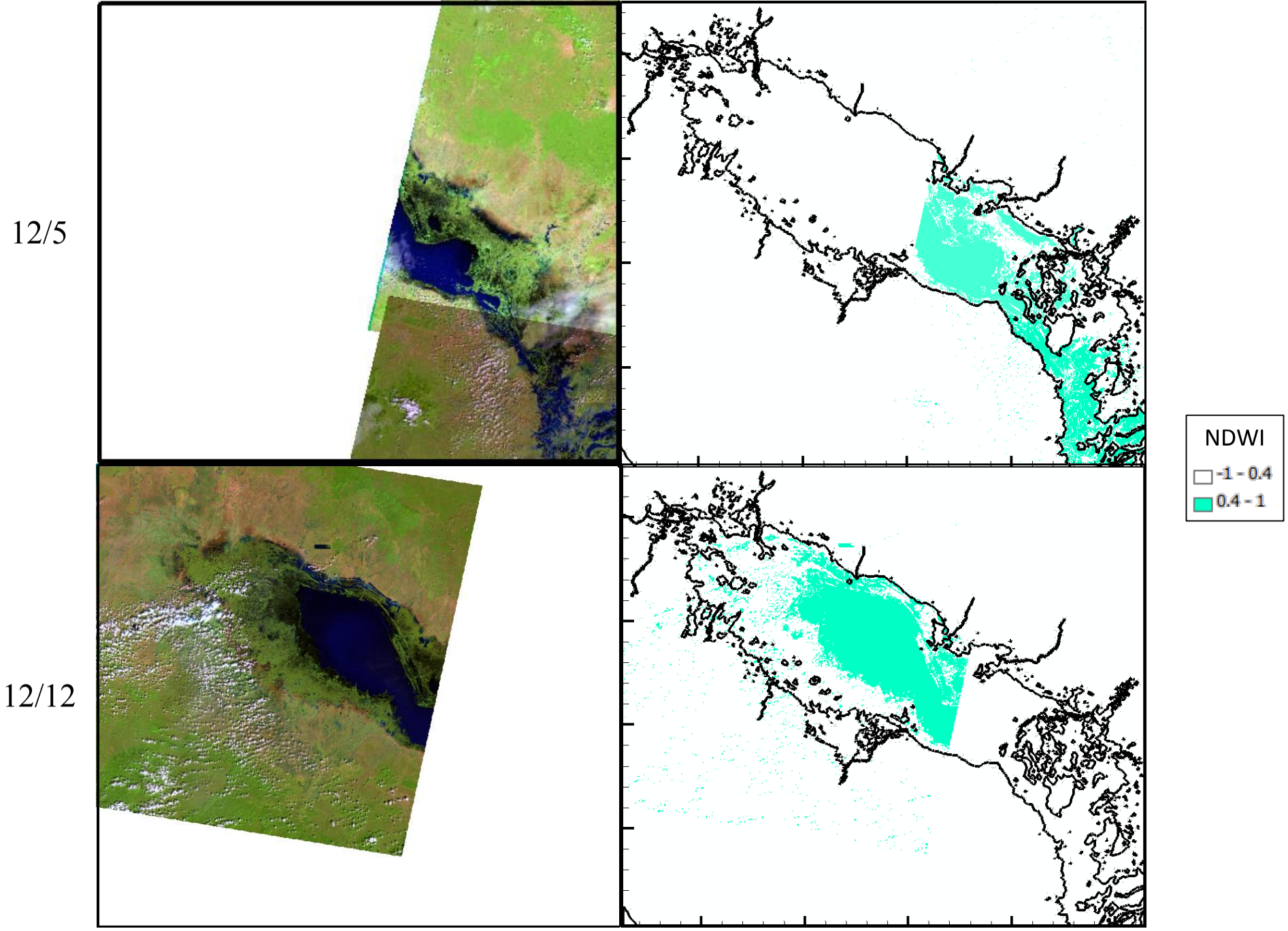



図4-16 Landsat衛星画像から抽出された水域，計算水陸境界線

| | | | |
|---|---|---|---|
| プロセッサ名 | | Intel® Core™ i7 920 | NVIDIA Tesla C1060 |
| プロセッサタイプ | | CPU | GPU |
| クロック数 | | 2.66GHz | 1.30GHz |
| コア数 | | 4コア | 240コア |
| メモリ転送帯域 | | 25.6GB/s | 102GB/sec |
| 演算性能 | 単精度(float) | 85.12GFLOPS* | 933GFLOPS |
| | 倍精度(double) | 42.56GFLOPS | 78GFLOPS |

*倍精度演算性能から推定

表4-3 計算に使用したCPUとGPUの諸元

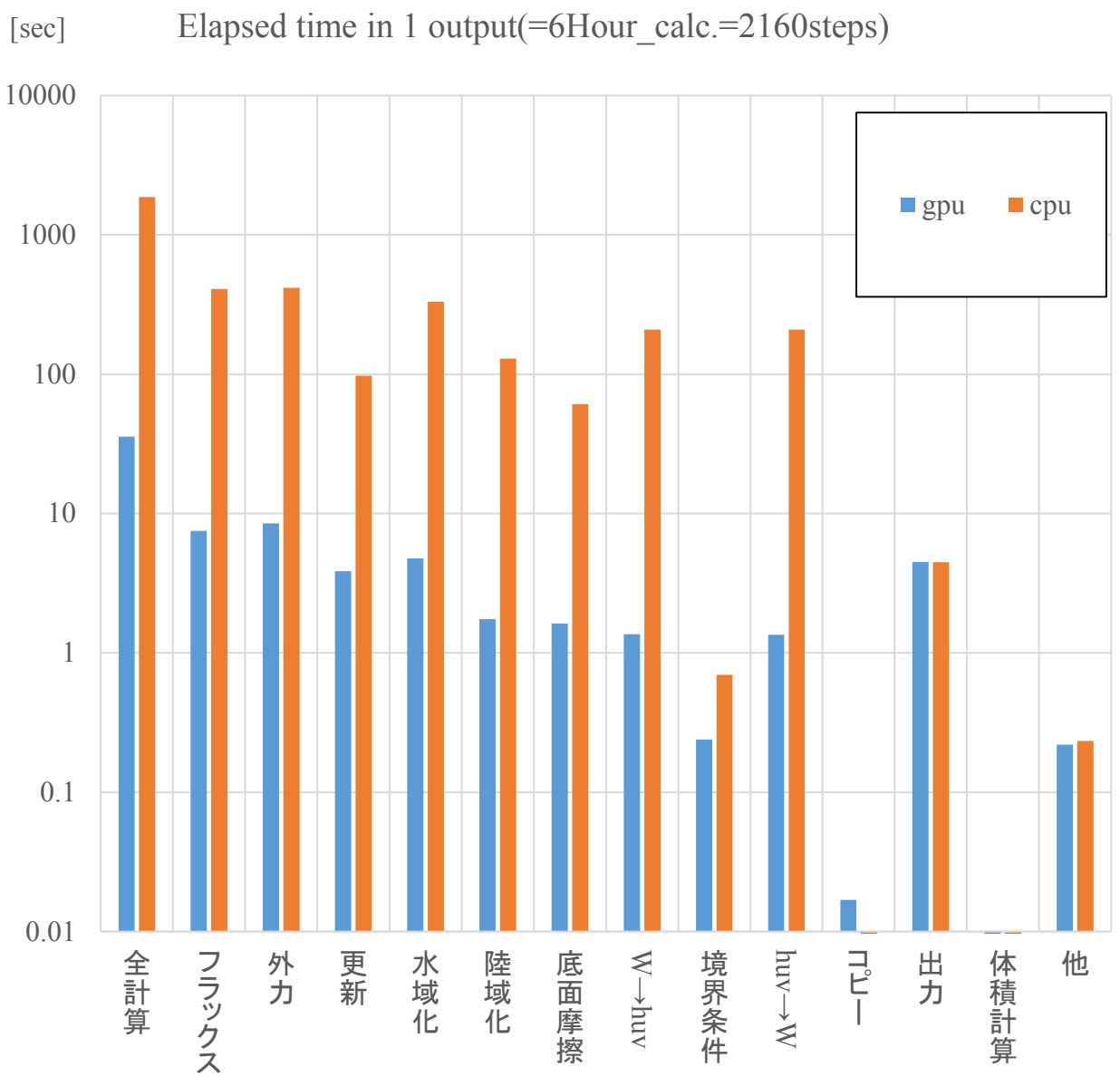



図4-17 各計算プロセス別CPU/GPUによる計算時間

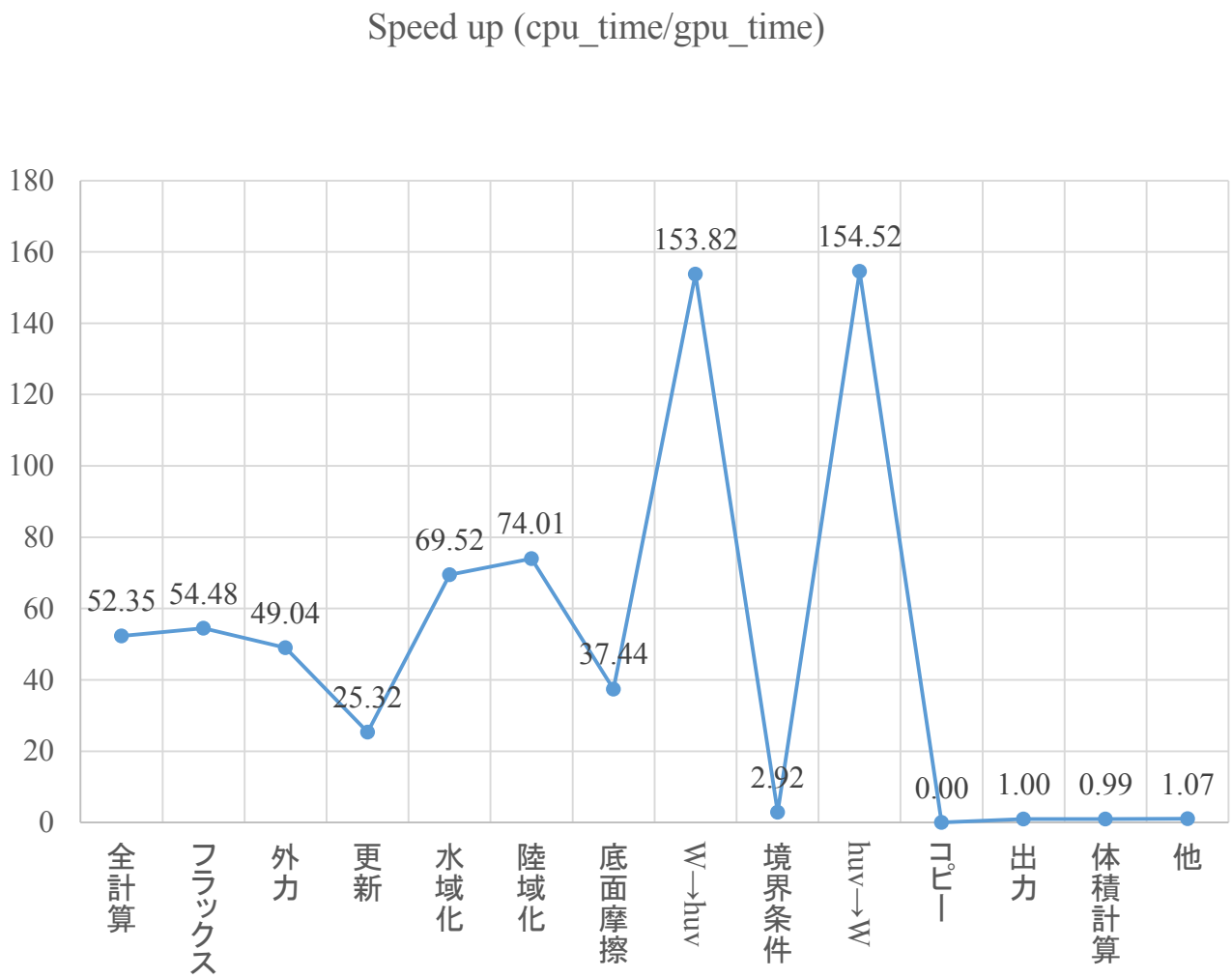
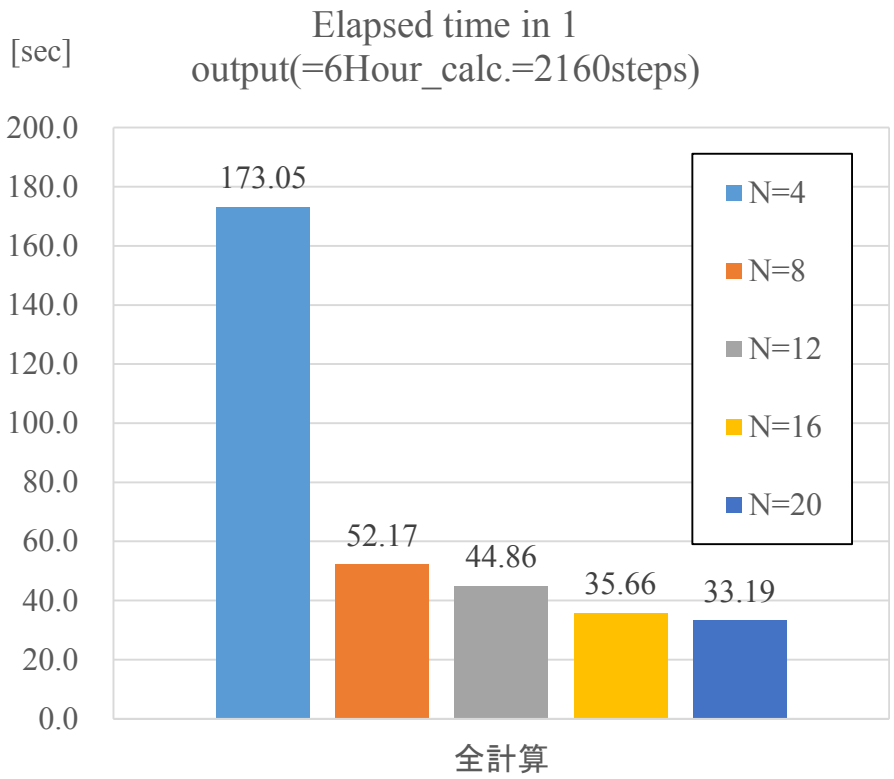



図4-18 GPU計算による高速化程度

| N_IN_BLOCK | RATIO_X | RATIO_Y | Cells | Overlap Cells | Blocks | alpha | alpha * Blocks | total elapsed time |
|---|---|---|---|---|---|---|---|---|
| ブロックの幅 (N^2<=512) | ブロックのx方向数 | ブロックのy方向数 | 総セル数 | のりしろセル数 | 総ブロック数 | ブロックのwarp回数(N^2<32*alpha) | | |
| 4 | 505 | 447 | 3611760 | 2708820 | 225735 | 1 | 225735 | 173.05 |
| 8 | 169 | 149 | 1611584 | 705068 | 25181 | 2 | 50362 | 52.17 |
| 12 | 101 | 90 | 1308960 | 399960 | 9090 | 5 | 45450 | 44.86 |
| 16 | 73 | 64 | 1196032 | 280320 | 4672 | 8 | 37376 | 35.66 |
| 20 | 57 | 50 | 1140000 | 216600 | 2850 | 13 | 37050 | 33.19 |

表4-4 各ブロックサイズにおける設定



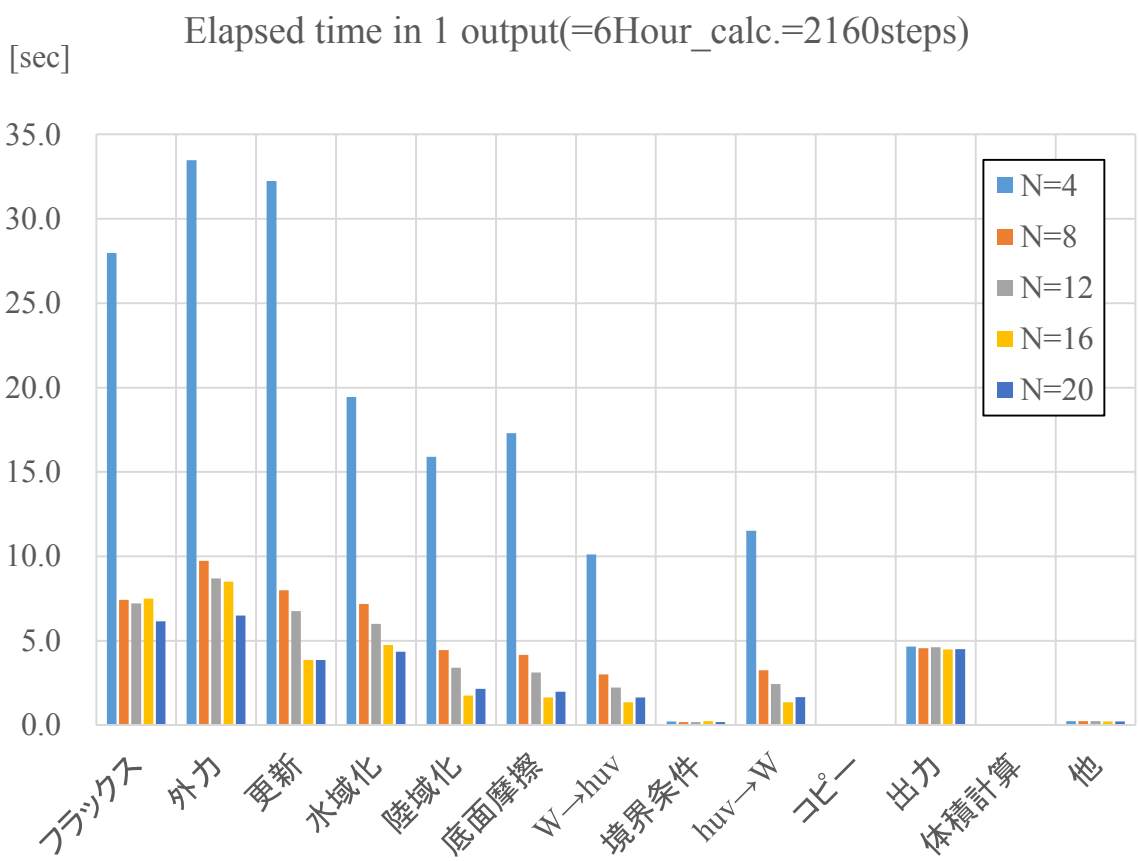




図4-19 ブロックサイズ別計算時間

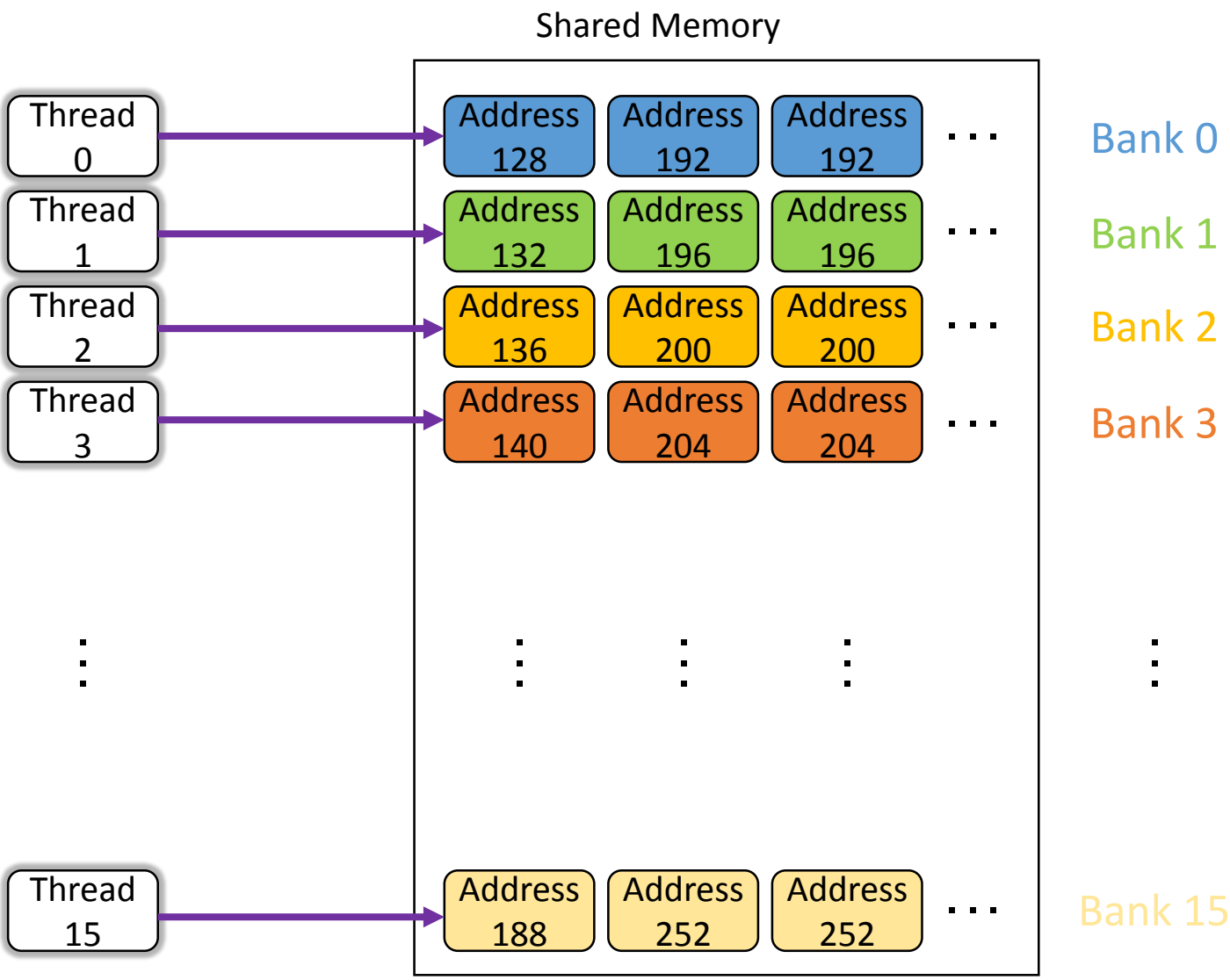



図4-20 シェアードメモリへのアクセス

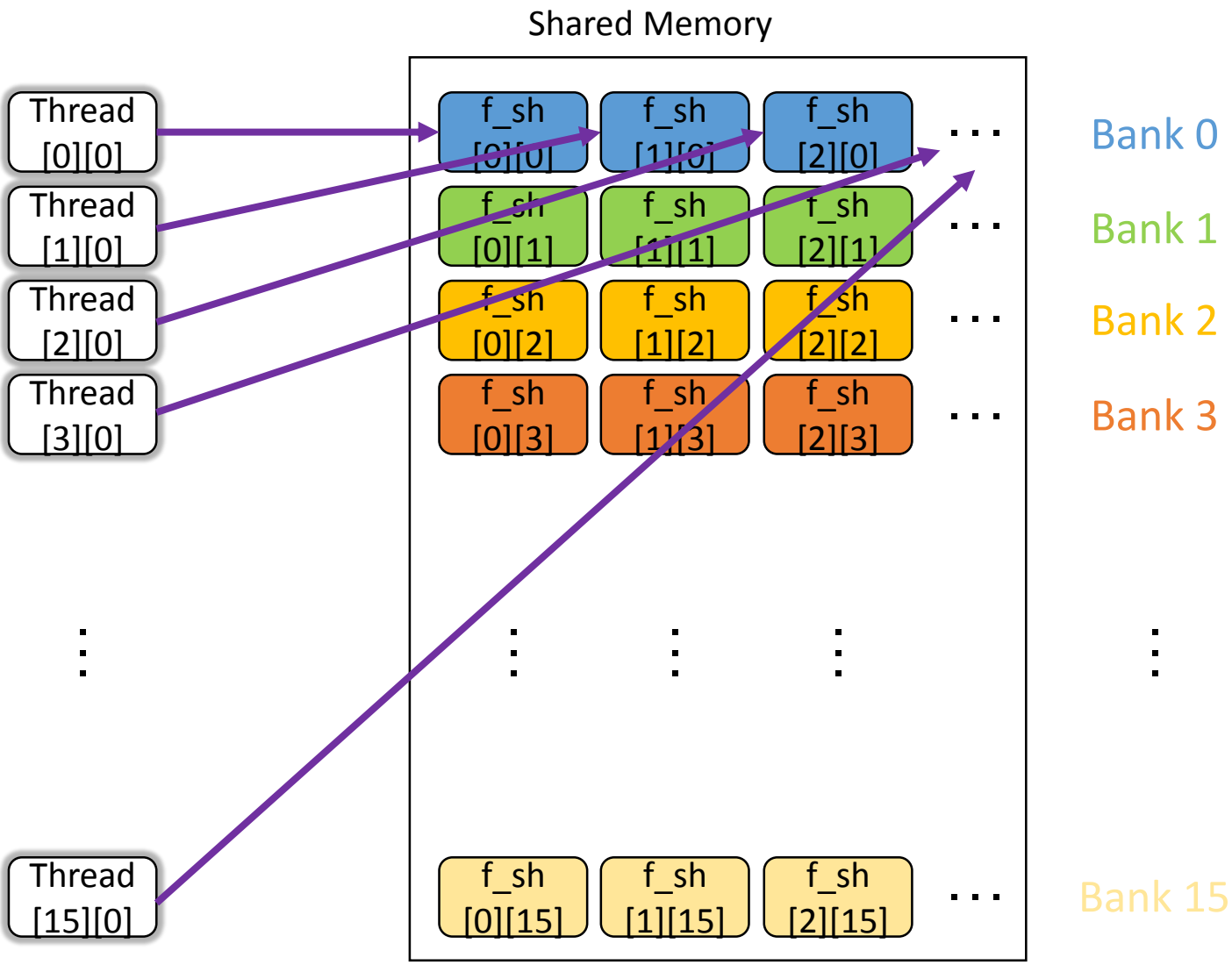



図4-21 16-wayバンクコンフリクト

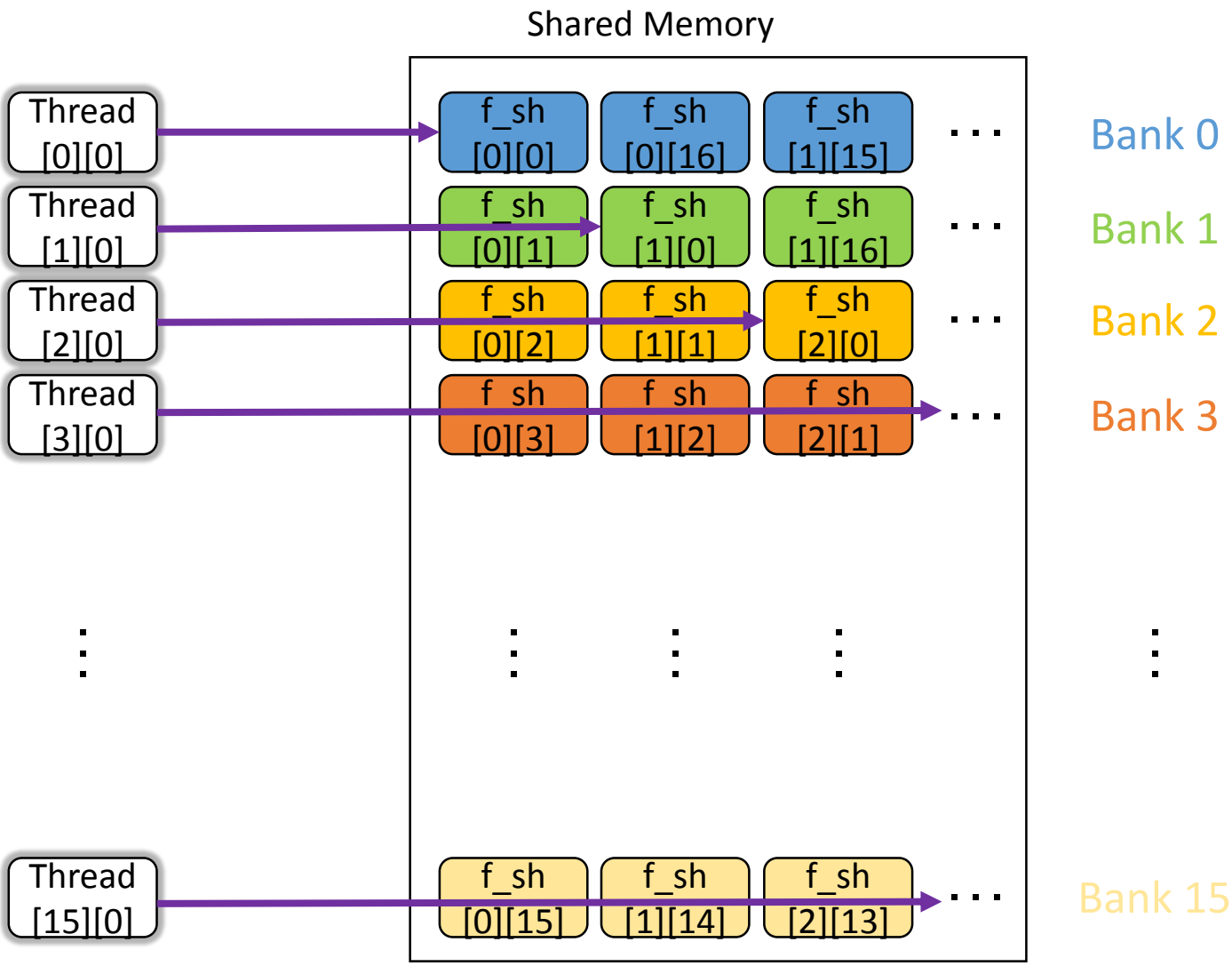
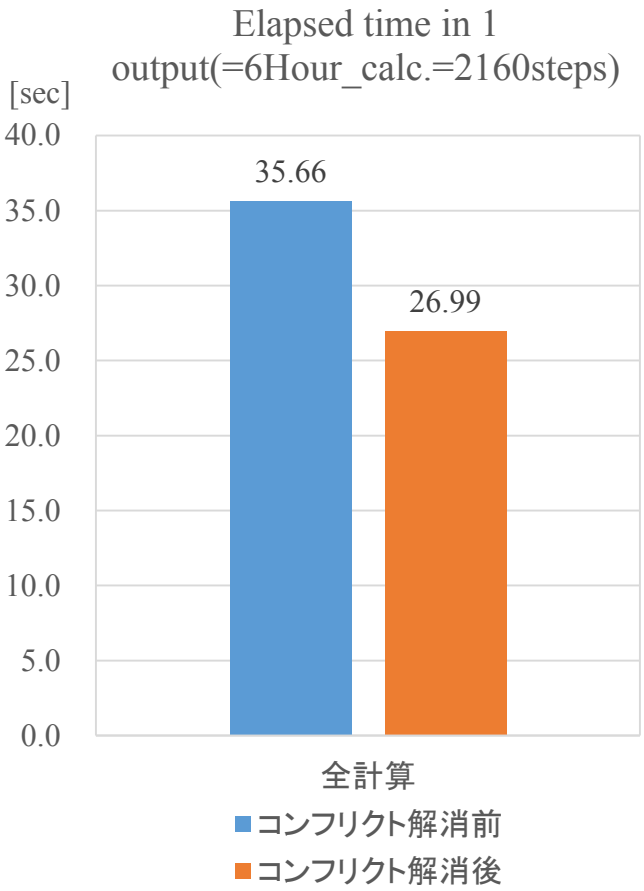



図4-22 16-wayバンクコンフリクトの解消



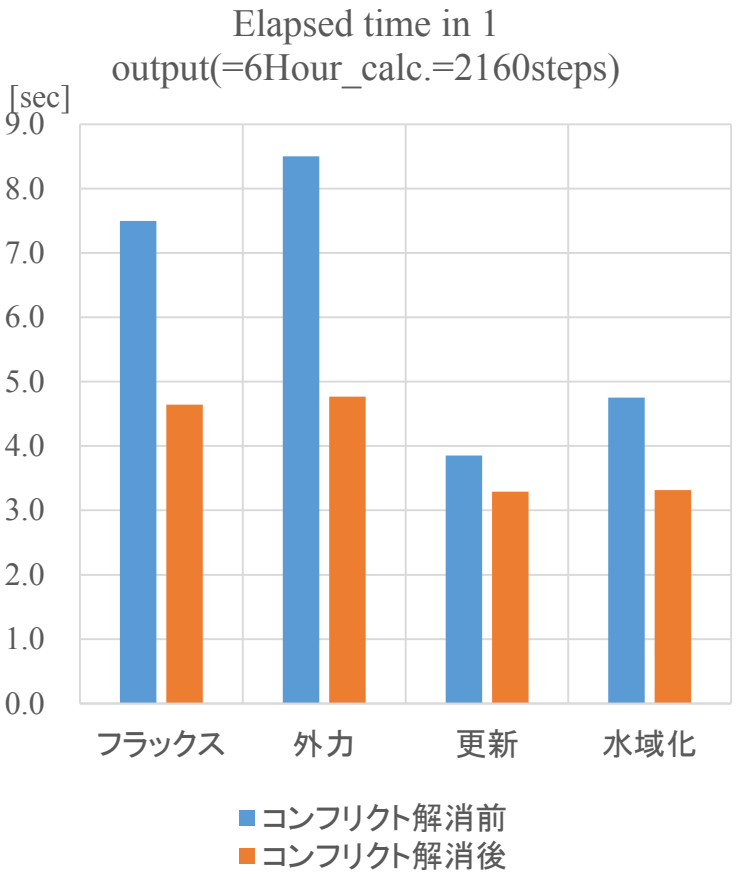




図4-23 バンクコンフリクト解消による計算時間の短縮

# 第 5 章 結論と今後の課題

## 5-1. 結論

本研究ではカンボジア国･トンレサップ湖を対象に，流動把握を目的として数値流動解析モデルを開発した．モデル開発においては，計算対象の時間的･空間的コスト，さらに現地技術者水準を考慮し，浅水 2 次元流モデル，直交格子，GPGPU による高速化とできるだけ簡便化された手法を用いた．さらにそのモデルを使用して 2002 年のトンレサップ湖水域変化を対象に再現計算を行い，観測水位データ，衛星画像を用いて計算再現精度の検証を行った．

第 2 章では流動数値計算における基礎方程式となる水平 2 次元浅水流方程式を導出し，直交格子に合わせた離散化を，有限体積法を用いて行った．またここでは解析解を持つ 1 次元･2 次元の 2 種類の遡上問題を取扱い，ベンチマークテストを行った．同テストにおいて再現精度は良好であり，トンレサップ湖への適用可能性が示された．

第 3 章では第 2 章のモデルを高速に計算できるようにするため，NVIDIA 社提供の CUDA ライブラリを使用して GPU 計算に適したプログラム化を行った．ここでは GPU による計算高速化原理，CUDA 独自のプログラミングモデル，メモリモデル，スレッドモデルについて述べた後，前章の浅水流方程式に基づいた水平 2 次元･直交格子型流動モデルを実装する手順について述べた．モデル実装の際には，サブドメイン間のデータ転送のため，各領域端を重ね合わせてのりしろ部分を作るように領域分割を行う，計算データを GPU 内で保存し極力 CPU への転送を行わない，SM 内での計算はシェアードメモリを利用するなどの工夫を用いて最適化し，GPU の高い演算性能を十分に引き出せるようにした．

第 4 章ではその GPU 化された流動モデルを用いてトンレサップ湖を対象に再現計算を行った．計算期間を境界条件である下流端の流量が観測されている 2002/7/18~12/19 とし，下流端から氾濫最大域，上流河川を十分に含む領域を計算領域として計算を行った．ポイントでの観測水位や Landsat 衛星画像から見られる水位変化･水域変化の様子が，多少の再現精度の低さはあるものの計算でも同様に再現されており，計算コストを考慮して水平 2 次元･直交格子型と単純化した流動計算モデルではあったが同湖への適用性が高いモデルであった．またここでは GPU 計算と CPU 計算の比較を行い，GPU 高速化を定量的に評価した．計算期間を 6 時間と設定した場合の計算時間は GPU 計算で 35.66sec，CPU 計算で 1866.66sec となり，約 50 倍程度の高速化が実現された．この結果を考慮し，今後 1 年スケールでの再現計算を行ったとすると，CPU で約 1 ヶ月以上かかる計算が GPU で約半日程度で終えることができると想定される．

以上の結果により，本研究で開発されたモデルは短い計算時間でトンレサップ湖の流動構造

把握が可能なモデルであることが示された.

## 5-2. 今後の課題

以上の結論を踏まえ，本研究で今後改善すべき課題点を以下に示す.

- 再現計算においてはSRTMデータを陸域の標高データとして使用したが,正確にHa Tien 基準面への変換を行う事ができず，代替的に DTM データとの回帰式から標高値を与えた. そのため実際の地形を再現できているとは必ずしも言えず, 再現計算への影響も無視することができない. よって今後より正確な地形データを入手し再現計算に適した標高値を与える必要がある.

- 本計算モデルでは外力項を底面摩擦, コリオリ力, レイノルズ応力の 3 つとしたが, トンレサップ湖への適用を考えると，氾濫域の湿地帯における樹木による抵抗力は無視することができない. そのため, 氾濫域における樹木の植生分布, 形状などを明確にし，樹木抵抗として新しく外力項に与えることが望ましい.

- 今回行った再現計算は約 5 ヶ月の期間であった. しかし実現象スケールを考慮するとこれでは短く，1 年スケールでの計算が求められる. そのためには計算に必要な各種条件，特に下流端における流量データを収集する必要がある.

- GPU 計算に使用した Tesla C 1060 は Tesla シリーズの中では比較的旧式(2008 年製)のものである. 現行ラインナップの最新 GPU である Tesla P 100(2016 年製)は単精度演算性能 9.5TFLOPS，倍精度演算性能 4.8TFLOPS と Tesla C 1060(単精度演算性能 0.9TFLOPS)よりも遥かに性能が高く，より高速に再現計算が可能である.

# 謝　辞

本研究を進めるにあたり，主指導教官である中村恭志准教授には，研究の進め方から，研究の構想，プレゼンテーションの方法など，様々な面で懇切丁寧にご指導いただきました．ここに深く感謝いたします．私一人の力だけでは到底ここまで来られなかったと感じております．そして研究生活を送る中で今の自分にかけているもの，今後身につけていくべき力などに，社会人となる前に気づくことができ非常に有意な時間であったと感じています．副指導教官である木内豪教授にはゼミ発表のたびに鋭いご指摘･アドバイスを下さり，本当にこの方法で大丈夫かと研究を振り返りながら進める事ができました．ありがとうございました．そして学部四年生時にお世話になり，昨年ご退職された石川忠晴教授には水理分野の初心者であった私に水理の基礎，研究者としての心構えなど熱心にご指導くださり，小川原湖の研究を通じて研究の面白さ，難しさ，有意義さ，そういったものを学ぶことができました．修士課程に入り本研究を進める際にもその経験が役立つ場面が多かったと思います．ありがとうございました．研究室の先輩･後輩･同期と学生の皆さんには同じ学生の立場として日々の生活，研究の相談など，この 2 年間の支えとなっていました．ありがとうございました．

昨年 9 月のカンボジアでの現地調査の際にご一緒に行動させていただいた山形大学･藤井秀人教授，富山県立大学･星川圭介教授，石川県立大学･藤原洋一教授と横村さん，本大学院土木工学専攻吉村研究室の Yang さん，そしてカンボジア当局の関係者皆様方には，一週間ほどの滞在でしたがカンボジアでの過ごし方，東南アジア中心の研究のお話，またトンレサップの現状など多くの面で大変お世話になりました．実際に現地を訪問し，初めてトンレサップを目の前にし，今自分はこの雄大な自然環境を相手にして研究を行っているのだと，研究モチベーションの向上につながりました．

最後に，私に様々な方面から助言してくださった皆様方に再度お礼申し上げます．ありがとうございました．