Some Numerical and Theoretical Results on a Two-dimensional Dam Overflow Problem

Li Tian Cheng

M.Phil.

The Hong Kong Polytechnic University

2000

Acknowledgment

I would like to thank the Department of Applied Mathematics, The Hong Kong Polytechnic University for the support of my project "A Study of Flexible, Fast and Efficient Numerical Simulation of Reservoirs" (Work Programme V324). I owe much to my supervisor, Dr. K. C. Chung for his valuable advice, constant patience and encouragement. Special thanks are given to my former supervisor, Mr. T. M. Shih who retired in December 1997, for his care and support. I am also indebted to Prof. Liang Gou-ping who developed the FEPG and passed it to me unreservedly. Meanwhile, Mr. Lin Zhou Chen and Mr. Dai Min are deserved thanks for their selfless help and beneficial discussion. Finally, I should show my gratitude to all the staff in the Department of Applied Mathematics who offer me a harmonious and pleasant two years of study.

Abstract

The thesis consists of four chapters and two appendixes. In the first chapter, background of the Dam Overflow problem, the Sobolev spaces, the Finite Element Method as well as other basic PDE concepts are introduced. This chapter also includes an extensive review of the existing related materials. In Chapter 2, the proposed algorithm of the potential function model is presented. Furthermore, a relaxation factor method is adopted in the thesis to achieve a better algorithm for running faster and more stably. In Chapter 3, some theoretical results as well as numerical results are obtained for the stream function model. In Chapter 4, the two methods using the potential function model and the stream function model are compared. Chapter 2, Chapter 3 and Chapter 4 are my major work on the field of the Dam Overflow Problem. In Chapter 5 some reviews on my work are given and some possible directions for future research are indicated. In Appendix A, all the theorems used in this thesis are listed, and in Appendix B, a powerful software called "Finite Element Program Generator (FEPG)" is briefly introduced. The FEPG provides a powerful and convenient tool for coding my programs for the present work.

Contents

List of Symbols

R^n	real n-dimensional Euclidean space
R	real line R^1
Ω	domain in R^n, an open set
$\partial\Omega$	boundary of Ω
$a(u,v)$	bilinear form corresponding to the differential
Υ	relaxation factor
m	Lebesgue measure
$\mathcal{L}(A)$	$\{f : A \to R \mid f$ is summable on A$\}$, for A bounded and measurable
$\xrightarrow{\text{w}}$	weak convergence
Δ	Laplace operator
∇	gradient
α	$\alpha = (\alpha_1, \cdots, \alpha_n)$, multi-index
$\|\alpha\|$	length of multi-index, $= \alpha_1 + \cdots + \alpha_n$
D^α	$D^\alpha = D_1^{\alpha_1} \cdots D_n^{\alpha_n} = \frac{\partial^{\|\alpha\|}}{\partial x_1^{\alpha_1} \cdots \partial x_n^{\alpha_n}}$
$C^0(\Omega)$	space of continuous functions on Ω
$C^k(\Omega)$	$C^k(\Omega) = \{u : D^\alpha u \in C^0(\Omega), 0 \le \|\alpha\| \le k\}$
$C_0^k(\Omega)$	$C_0^k(\Omega) = C^k(\Omega) \cap \{u : \operatorname{spt} u$ compact, $\operatorname{spt} u \subset \Omega\}$
$C^k(\bar{\Omega})$	$\{u : D^\alpha u$ uniformly continuous on $\Omega, \|\alpha\| \le k\}$
$C^k(\Omega \cup \Gamma)$	$\{u : u \in C^k(\bar{\Omega})$ and $\operatorname{spt} u \subset \Omega \cup \Gamma\}$
$\|u\|_{p;\Omega}$	L^p-norm on Ω
$\|u\|$	L^2-norm on Ω
$L^\infty(\Omega)$	$\{f : f \in L^p(\Omega), \forall p < \infty\}$
$\|u\|_{\infty;\Omega}$	sup norm on Ω
$W^{k,p}(\Omega)$	Sobolev space on Ω
$\|u\|_{k,p;\Omega}$	Sobolev norm on Ω
$\|u\|_1$	$\|u\|_1 = \|u\|_{1,2;\Omega}$
$W_0^{k,p}(\Omega)$	norm closure of $C_0^\infty(\Omega)$
$H^{m,p}(\Omega)$	$H^{m,p}(\Omega) = W^{m,p}(\Omega)$
$H^m(\Omega)$	$H^m(\Omega) = H^{m,2}(\Omega)$
$f(x)^-$	$f(x)^- = \dfrac{f(x) - \|f(x)\|}{2}$
$f(x)^+$	$f(x)^+ = \dfrac{f(x) + \|f(x)\|}{2}$

Chapter 1

Introduction

1.1 Historical Background

Dams are constructed for economic development, and their construction involves large investments of money, and natural and human resources. Of the various types of dams constructed around the globe, earth dams are the most common type and constitute a vast majority of dams. Dams are used to store enormous amount of water. When a dam fails, it causes a sudden release of water which is a potential threat to virtually everything downstream. The dam failure may result in loss of life and property. So programs of dam research have been developed in most countries of the world and the dam overflow problem is one of them.

Following the development of electronic technology and computer hardware, more and more large scale scientific computing, which consumed much CPU time before, have become practically possible.

In the fluid computing, one can obtain various parameters of flow using mathematical models combined with physical models. Meanwhile, computing the flow field by the finite element method has been developing rapidly. The method not only allows us to obtain data satisfying various needs in engineering, but also is economical in the sense that it saves machine time and manpower.

The Dam Overflow Problem have been developed since 1970's and many related results can be found from both mathematical and engineering journals. For example Ding Dao Yang proposed a method to compute potential function model in 1982

and Jiang Li Shang constructed the stream function model in 1986.

1.2 Sobolev Spaces

The Sobolev spaces, i.e. the classes of functions with generalized derivatives in L_p, occupy an outstanding place in functional analysis, for its important use in numerical solutions of PDE using finite elements. During the last two decades, a substantial contribution to the study of these spaces has been made; so now solutions to many important problems connected with them are known. For details please refer to [24]–[25].

1.2.1 Motivation of Introducing Sobolev Spaces

As an independent subject, the study of PDE can be dated to the eighteen century. After two centuries of development, classical research has achieved fruitfully in both theory and application. Until the twentieth century, following the development of functional analysis and other subjects, people have gradually recognized that the classical PDE theory has many limitations. For instance, consider the following Dirichlet problem on a plain open set Ω.

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega. \end{cases} \tag{1.1}$$

In the classical theory the solution u of (1.1) is unique in $C^2(\Omega)$. In other word, under the classical theory, the domain of definition of the differential operator Δ is $D(\Delta) = \{v \in C^2(\Omega) \cap C(\bar{\Omega}) : v|_{\partial\Omega=0}\}$ and the range of Δ, $R(\Delta) \in C(\bar{\Omega})$. Obviously, $D(\Delta)$ is a dense subset of $C(\bar{\Omega})$ with respect of the ∞-norm. The drawback of the classical theory comes from the fact that $\Delta : D(\Delta) \longrightarrow C(\bar{\Omega})$ is not a surjection, i.e. it is not true that (1.1) has a solution for every continuous function $f \in C(\bar{\Omega})$. Here a counter example is constructed to illustrate this.

Assume Ω is the unit circle in the xy-plane. Suppose that for each $f \in C(\bar{\Omega})$, (1.1) has a solution $u \in D(\Delta)$. Then $f \longmapsto D_x D_y u$ is a mapping from $C(\bar{\Omega})$ to itself.

Regarding $D_xD_yu(0,0)$ as a linear functional, there exists a measure μ on Ω, such that

$$D_xD_yu(0,0) = \int_\Omega f d\mu. \tag{1.2}$$

To prove this, we see that the functional $P : f \longmapsto f(0,0)$ is a linear functional on $C(\bar{\Omega})$ in the sense of L^∞-norm. In light of the Hahn–Banach theorem (see A.5), P can be extended to the linear functional P^* on $L^\infty(\Omega)$ such that

$$P^*(x) = P(x) \text{ for all } x \in C(\bar{\Omega})$$

Then there is $g \in L^1(\Omega)$ satisfying

$$P^*(f) = \int_\Omega f(x)g(x)dx$$

Introduce a new measure $\mu(B) = \int_B g(x)dx$, where B is a Borel-set. Then for any $f \in C(\bar{\Omega})$

$$f(0,0) = P(f) = P^*(f) = \int_\Omega f(x)g(x)dx = \int_\Omega f(x)d\mu$$

Therefore μ is the required measure.

Meanwhile,

$$u(x,y) = \int_\Omega G(x,y;\xi,\eta)f(\xi,\eta)d\xi d\eta \tag{1.3}$$

where $G(x,y;\xi,\eta)$ is the Green function of the Laplace operator on unit circle and is given by

$$G(x,y;\xi,\eta) = \frac{1}{2\pi}\log\frac{r_1}{r_2} + \frac{1}{2\pi}\log\rho,$$

where

$$\begin{aligned} r_1 &= [(x-\xi)^2 + (y-\eta)^2]^{\frac{1}{2}}, \\ \rho &= (\xi^2+\eta^2)^{\frac{1}{2}}, \\ r_2 &= \left[(x-\frac{\xi}{\rho^2})^2 + (y-\frac{\eta}{\rho^2})^2\right]^{\frac{1}{2}} \end{aligned}$$

Therefore $D_xD_yG(0,0;\xi,\eta) = \frac{1}{\pi}\xi\eta\frac{1-\rho^4}{\rho^4}$. From (1.3),

$$D_xD_yu(0,0) = \lim_{\epsilon\to 0}\int_{\epsilon<\rho<1}\frac{1}{\pi}\xi\eta\frac{(1-\rho^4)}{\rho^4}f(\xi,\eta)d\xi d\eta, \tag{1.4}$$

which is infinite for some f and hence contradicts (1.2).

From the theory of functional, the contradiction arises from the fact that the classical differential operator is not a close operator.

Since $D(\Delta)$ is a dense subset of $C(\bar{\Omega})$, there always exists a close extension of Δ. By this close extension, the classical derivatives is substituted by the generalized derivatives.

1.2.2 Definitions and Basic Properties

We define a functional $\|\cdot\|_{m,p}$, where m is a nonnegative integer and $1 \leq p < \infty$, as follows:

$$\|u\|_{m,p} = \left\{ \sum_{0 \leq |\alpha| \leq m} \|D^\alpha u\|_p^p \right\}^{\frac{1}{p}} \text{ if } 1 \leq p < \infty, \tag{1.5}$$

$$\|u\|_{m,\infty} = \max_{0 \leq |\alpha| \leq m} \|D^\alpha u\|_\infty \tag{1.6}$$

for any function u for which the right side makes sense, $\|\cdot\|_p$ being the $L^p(\Omega)$-norm. (In situations where confusion of domains may occur we shall write $\|u\|_{m,p,\Omega}$ in place of $\|u\|_{m,p}$). It is clear that (1.5) or (1.6) defines a norm on any vector space of functions for which the right side takes finite values provided that functions are identified in the space if they are equal almost everywhere in Ω. We consider three such spaces corresponding to any given values m and p:

$H^{m,p}(\Omega)$ = the completion of $\{u \in C^m(\bar{\Omega}) : \|u\|_{m,p} < \infty\}$ with respect of the norm $\|\cdot\|_{m,p}$,

$W^{m,p}(\Omega) = \{u \in L^p(\Omega) : D^\alpha u \in L^p(\Omega) \text{ for } 0 \leq |\alpha| \leq m\}$, where D^α is the generalized partial derivative, and

$W_0^{m,p}(\Omega)$ = the closure of $C_0^\infty(\Omega)$ in the space $W^{m,p}(\Omega)$.

Equipped with the appropriate norm (1.5) or (1.6), these are called *Sobolev spaces* over Ω. Clearly $W^{0,p} = L^p(\Omega)$, and if $1 \leq p < \infty$, $W_0^{0,p}(\Omega) = L^p(\Omega)$ by Theorem 2.19 in [24]. For any m the chain of imbeddings

$$W_0^{m,p}(\Omega) \to W^{m,p}(\Omega) \to L^p(\Omega)$$

is also clear. Theorem 3.16 in [24] shows that $H^{m,p}(\Omega) = W^{m,p}(\Omega)$.

In the following, we detail one of the most important theorems of Sobolev space, the Sobolev imbedding theorem. Before stating the theorem, the definition of the cone property is given as follows:

Definition We say that Ω has the cone property if there exists a bounded open cone K such that each $x^0 \in \partial\Omega$ is the vertex of a cone $K(x^0) \subset \Omega$ and $K(x^0)$ is congruent to K.

THEOREM (*The Sobolev imbedding theorem*) Let Ω be a domain in R^n and let Ω^k be the k-dimensional domain obtained by intersecting Ω with a k-dimensional plane in R^n, $1 \leq k \leq n$. (Thus $\Omega^n = \Omega$.) Let j and m be nonnegative integers and let p satisfy $1 \leq p < \infty$.

PART I If Ω has the cone property, then we have the following imbeddings:

Case A Suppose $mp < n$ and $n - mp < k \leq n$, then

$$W^{j+m,p}(\Omega) \rightarrow W^{j,q}(\Omega^k), \quad p \leq q \leq kp/(n - mp), \tag{1.7}$$

and in particular,

$$W^{j+m,p}(\Omega) \rightarrow W^{j,q}(\Omega), \quad p \leq q \leq np/(n - mp), \tag{1.8}$$

or

$$W^{m,p}(\Omega) \rightarrow W^{p}(\Omega), \quad p \leq q \leq np/(n - mp). \tag{1.9}$$

Moreover, if $p = 1$, so that $m < n$, imbedding (1.7) also exists for $k = n - m$.

Case B Suppose $mp = n$, Then for each k, $1 \leq k \leq n$,

$$W^{j+m,p}(\Omega) \rightarrow W^{j,q}(\Omega^k), \quad p \leq q < \infty, \tag{1.10}$$

so that in particular

$$W^{m,p}(\Omega) \rightarrow L^{q}(\Omega^k), \quad p \leq q < \infty. \tag{1.11}$$

Moreover, if $p = 1$ so that $m = n$, imbedding (1.10) and (1.11) exist with $q = \infty$ as well; in fact,

$$W^{j+n,1}(\Omega) \rightarrow C_0^j(\Omega). \tag{1.12}$$

Case C Suppose $mp > n$. Then

$$W^{j+m,p}(\Omega) \to C_0^j(\Omega) \tag{1.13}$$

PART II If Ω has the strong local Lipschitz property, then Case C of Part I can be refined as follows:

Case C′ Suppose $mp > n > (m-1)p$. Then

$$W^{j+m,p}(\Omega) \to C^{j,\lambda}(\bar{\Omega}), \quad 0 < \lambda \leq m - (n/p). \tag{1.14}$$

Case C″ Suppose $n = (m-1)p$. Then

$$W^{j+m,p}(\Omega) \to C^{j,\lambda}(\bar{\Omega}), \quad 0 < \lambda < 1. \tag{1.15}$$

Also, if $n = m - 1$ and $p = 1$, then (1.15) holds for $\lambda = 1$ as well. For the proof of this theorem please refer to [24].

1.3 Free Boundary Problem, Variational Principle and the Finite Element Method

1.3.1 Free Boundary Problem

Important developments in the study of free boundary problems have been achieved in recent years by introducing variational approach wherever possible. This enables one to conclude without great effort that a solution to the free-boundary problem exists in some "weak" sense. One can proceed to establish the regularity of the solution and then, hopefully, study the smoothness of the free boundary itself. In fact, within the last few years, significant new methods have been developed for analyzing the free boundary and the theory has now reached a certain stage of maturity; its future looks even more exciting. An increasing number of physical and engineering problems are becoming accessible to this growing body of methods.

In the following we use an example to illustrate what a free boundary problem is.

The Dirichlet problem for the Laplace operator Δ seeks a solution u of $\Delta u = f$ in a given domain Ω satisfying $u = \phi$ on the boundary $\partial\Omega$. Suppose that only a portion S of $\partial\Omega$ is given whereas the remaining portion Γ is not prescribed, and an additional condition is imposed on the unknown part of the boundary, such as $\nabla(u - \phi) = 0$ on Γ (here ϕ is a given function in the entire domain Ω). Thus we seek to determine u and Γ satisfying

$$\Delta u = f \quad \text{in} \quad \Omega, \tag{1.16}$$

$$u = \phi \quad \text{on} \quad \partial\Omega, \tag{1.17}$$

$$\nabla(u - \phi) = 0 \quad \text{on} \quad \Gamma \tag{1.18}$$

where Ω is bounded by the given S and the unknown Γ. This problem is an example of a free boundary problem. (1.18) is the free boundary condition and (1.17) is the ordinary boundary condition.

For a two-dimensional ideal fluid, the density function u satisfies, on the interface Γ between the fluid and the air, the free boundary conditions

$$u = C_1, \quad |\nabla u| = C_2 \quad (C_1, C_2 \text{ constants})$$

where normally either C_1 or C_2 is not unknown; Γ is also not prescribed.

1.3.2 Variational Principles

The classical variational method is the Ritz-Galerkin method which relates directly to the finite element method in mathematical sense. We briefly introduce these methods respectively as follows.

The Ritz Method The method is derived from the theorem of the minimum energy. For instance, through theorem of the minium energy, the solution u of the equations

$$\begin{cases} -\Delta u = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases}$$

must satisfy

$$J(u) = \min J(v), \quad v \in H_0^1(\Omega) \tag{1.19}$$

where

$$J(v) = \frac{1}{2}\int_\Omega \nabla v \cdot \nabla v \, dx\, dy - \int_\Omega f\, v \, dx\, dy.$$

Assume H is a separable Hilbert space and let

$$S_N = \left\{ \omega_N | \omega_N = \sum_{i=1}^{N} C_i \varphi_i, \text{ where } (C_1, \cdots, C_N) \in R^N \right\} \tag{1.20}$$

being expanded by N linear independent vectors $\varphi_1, \cdots, \varphi_N$, be a finite dimensional subset of H. We replace H by S_N and find the extreme value of the functional $J(\omega)$ on S_N. i.e. find $u_N \in S_N$ satisfying

$$J(u_N) = \min_{\omega_N \in S_N} J(\omega_N). \tag{1.21}$$

This is **the Ritz Method**.

In fact u_N can be obtained by solving linear equations. Since $J(\omega)$ is

$$J(\omega) = \frac{1}{2} D(\omega, \omega) - F(\omega) \tag{1.22}$$

where D is a bilinear form and F is a linear functional. And in (1.19)

$$\begin{aligned} D(u, v) &= \int_\Omega \nabla u \cdot \nabla v \, dx\, dy \\ F(v) &= \int_\Omega f\, v \, dx\, dy \end{aligned}$$

If $\omega_N = \sum_{i=1}^{N} c_i \varphi_i$,

$$\begin{aligned} J(\omega_N) &= \frac{1}{2} D\left(\sum_{i=1}^{N} c_i \varphi_i, \sum_{i=1}^{N} c_i \varphi_i \right) - F(\sum_{i=1}^{N} c_i \varphi_i) \\ &= \frac{1}{2} \sum_{i,j=1}^{N} D(\varphi_i, \varphi_j) c_i c_j - \sum_{i=1}^{N} F(\varphi_i) c_i \end{aligned}$$

Therefore, $J(\omega_N)$ is a quadratic polynomial $j(c_1, \cdots, c_N)$ of variable $c_1, \cdots, c_N$. If the coefficient matrix of the second degree terms, $[D(\varphi_i, \varphi_j)]_{i,j=1,\cdots,N}$ is positive definite, then $J(\omega_N)$ has a unique extremum point $(c_1^0, \cdots, c_N^0)$ such that

$$j(c_1^0, \cdots, c_N^0) = \min_{c_1, \cdots, c_N} j(c_1, \cdots, c_N) \tag{1.23}$$

From (1.23) and the necessary condition for an extremum, we have

$$\left.\frac{\partial j}{\partial c_i}\right|_{(c_1^0,\cdots,c_N^0)} = 0 \quad (i = 1, \cdots, N) \tag{1.24}$$

i.e. $c = [c_1^0, \cdots, c_N^0]$ satisfies

$$Kc = F$$

where $F = [F(\varphi_1), \cdots, F(\varphi_N)]^T$ and $K = [D(\varphi_i, \varphi_j)]_{i,j=1,\cdots,N}$.

The Galerkin Method The method is derived from the principle of virtual energy. Here we find $u_N \in S_N$ such that

$$D(u_N, \omega_N) = F(\omega_N), \quad \forall \omega_N \in S_N, \tag{1.25}$$

where the definition of D and F are the same as in **the Ritz Method**. Assume $u_N = \sum_{i=1}^{N} c_i^0 \varphi_i$, $\omega_N = \sum_{i=1}^{N} c_i \varphi_i$, then

$$\begin{aligned} 0 &= \sum_{i,j=1}^{N} D(\varphi_i, \varphi_j) c_j^0 c_i - \sum_{i=1}^{N} c_i F(\varphi_i) \\ &= \sum_{i=1}^{N} \left[\sum_{j=1}^{N} D(\varphi_i, \varphi_j) c_j^0 - F(\varphi_i) \right] c_i \end{aligned}$$

Because c_i are arbitrary,

$$\sum_{j=1}^{N} D(\varphi_i, \varphi_j) c_j^0 = F(\varphi_i) \quad (i = 1, \cdots, N)$$

It is the same as the linear equations deduced in the Ritz Method

$$Kc = F.$$

This also indicates that the theorem of minimum energy and the principle of virtual energy are equivalent.

1.3.3 The Finite Element Method

Perhaps no other family of approximation methods has had such a great impact on the theory and practice of numerical methods during the twentieth century as the

finite element method. The method has been used in virtually every conceivable area of engineering that can make use of models of nature characterized by partial differential equations. There are dozens of textbooks, monographs, handbooks, memoirs, and journals devoted to the method. Further studies, numerous conferences, symposia, and workshops on various aspects of the finite element methodology are held regularly throughout the world. There exist easily over thousands of references on finite element today, and this is growing exponentially with further revelations of the power and versatility of the method. Today finite element methodology is making significant inroads into fields in which many thought were outside its realm; for example, computation fluid dynamics. In time, finite element methods may assume a position in this area of comparable or great importance rather than classical difference schemes which have long dominated the subject.

The finite element method is the development and modification of Ritz-Galerkin method. Its key is choosing piece-wise polynomial functions as the basis function for the subset S_N. In the method, we decompose the domain Ω into a series of elements first, then we construct interpolating polynomial on every element, which satisfy some continuous conditions on the line (surface) of intersection between two neighboring element. This ensure that the S_N is the subset of H. In this sense, the basis of the finite element method is the variational principle and interpolation of piece-wise polynomial. Here we use an example to illustrate the finite element algorithm. For simplicity we consider the Dirichlet problem of the Poisson equation.

$$\begin{cases} \Delta u = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{1.26}$$

For simplicity and without loss of generality, assume Ω is a polygonal domain. The generalized solution of (1.26) is the function $u \in H_0^1(\Omega)$ satisfying

$$D(u,\omega) = (f,\omega) \quad \forall \omega \in H_0^1(\Omega)$$
$$(f,\omega) = \int_D f\,\omega\,dx\,dy$$

Decompose Ω into triangles and denote the triangles by $e_1, \cdots, e_{N_E}$. The nodes inside Ω are denoted by $P_1, \cdots, P_{N_P}$ and the nodes on the boundary by $P_{N_P+1}, \cdots, P_{N_P+M}$. The maximum line length of e_i is h.

Linear interpolation is used for every element $e = \triangle P_i P_j P_m$ so that the values at the nodes P_i, P_j, P_m are equal to known values $\omega_i, \omega_j, \omega_m$ respectively. If a node is on the boundary $\partial\Omega$, the corresponding known value is 0. Through this way, we construct a piecewise linear function $\omega_h(x, y)$ (the subscript h is associated with the decomposition). While $(\omega_1, \cdots, \omega_{N_P})$ going through all the vectors of space R_{N_P}, we can obtain an N_P-dimensional space S_h consisting of all such $\omega_h(x, y)$.

Obviously, S_h is a linear space and is a subset of $H_0^1(\Omega)$.

Since S_h is a N_P-dimensional subset of $H_0^1(\Omega)$, there must exist N_P basis function $\varphi_1(x, y)$, $\cdots$, $\varphi_{N_P}(x, y)$

$$\omega_h(x, y) = \sum_{i=1}^{N_P} \omega_i \varphi_i(x, y)$$

where the $\varphi_i(x, y)$ satisfies

1. $\varphi_i(P_j) = \delta_{ij}$

2. $\varphi_i(x, y)$ is a linear function on every element.

Obviously, $\varphi_i(x, y) \in H_0^1(\Omega)$, $\varphi_i(x, y) \not\equiv 0$ inside the triangle including node P_i and $\varphi_i(x, y) \equiv 0$ elsewhere. Therefore the support of $\varphi_i(x, y)$ is a polygonal domain containing P_i as its interior point.

The finite element method consists of finding $u_1, \cdots, u_{N_P}$ such that

$$u_h(x, y) = \sum_{\alpha=1}^{N_P} u_\alpha \varphi_\alpha(x, y)$$

satisfying

$$D(u_h, \omega_h) - (f, \omega_h) = 0 \quad \forall \omega_h \in S_h. \tag{1.27}$$

Assume $w_h = \sum_{\alpha=1}^{N_P} \omega_\alpha \varphi_\alpha(x, y)$ and (1.27) is

$$\sum_{\alpha,\beta=1}^{N_P} \omega_\alpha u_\beta D(\varphi_\alpha, \varphi_\beta) - \sum_{\alpha=1}^{N_P} (f, \omega_\alpha) = 0,$$

where

$$\begin{aligned} D(\varphi_\alpha, \varphi_\beta) &= \sum_{n=1}^{N_E} \int_{e_n} \Big(\frac{\partial \varphi_\alpha}{\partial x} \frac{\partial \varphi_\beta}{\partial x} + \frac{\partial \varphi_\alpha}{\partial y} \frac{\partial \varphi_\beta}{\partial y} \Big) dx\, dy \\ (f, \varphi_\alpha) &= \sum_{n=1}^{N_E} \int_{e_n} f\, \varphi_\alpha\, dx\, dy. \end{aligned}$$

Introduce

$$\begin{aligned} D_n(\varphi_\alpha, \varphi_\beta) &= \int_{e_n} \Big(\frac{\partial \varphi_\alpha}{\partial x} \frac{\partial \varphi_\beta}{\partial x} + \frac{\partial \varphi_\alpha}{\partial y} \frac{\partial \varphi_\beta}{\partial y} \Big) \, dx \, dy \\ (f, \varphi_\alpha)_n &= \int_{e_n} f \, \varphi_\alpha \, dx \, dy \end{aligned}$$

and assume the element $e_n = \triangle P_i P_j P_m$. Therefore there are only three basis functions φ_i, φ_i and φ_m, which do not equal 0 over the element e_n. Then

$$\begin{aligned} &\sum_{\alpha,\beta=1}^{N_P} w_\alpha u_\beta D_n(\varphi_\alpha, \varphi_\beta) = \sum_{\alpha,\beta=1}^{N_P} w_\alpha u_\beta \sum_{n=1}^{N_E} D_n(\varphi_\alpha, \varphi_\beta) \\ = &\sum_{n=1}^{N_E} \sum_{\alpha,\beta=1}^{N_P} w_\alpha u_\beta D_n(\varphi_\alpha, \varphi_\beta) = \sum_{n=1}^{N_E} \sum_{\alpha,\beta=i,j,m} w_\alpha u_\beta D_n(\varphi_\alpha, \varphi_\beta) \\ = &\sum_{n=1}^{N_E} \begin{bmatrix} w_i \\ w_j \\ w_m \end{bmatrix}^T \begin{bmatrix} D_n(\varphi_i, \varphi_i) & D_n(\varphi_i, \varphi_j) & D_n(\varphi_i, \varphi_m) \\ D_n(\varphi_j, \varphi_i) & D_n(\varphi_j, \varphi_j) & D_n(\varphi_j, \varphi_m) \\ D_n(\varphi_m, \varphi_i) & D_n(\varphi_m, \varphi_j) & D_n(\varphi_m, \varphi_m) \end{bmatrix} \begin{bmatrix} u_i \\ u_j \\ u_m \end{bmatrix}. \end{aligned}$$

Let

$$\{\delta^*\}_{e_n} = \begin{bmatrix} w_i \\ w_j \\ w_m \end{bmatrix}, \quad \{\delta\}_{e_n} = \begin{bmatrix} u_i \\ u_j \\ u_m \end{bmatrix}$$

and

$$[k]_{e_n} = \begin{bmatrix} D_n(\varphi_i, \varphi_i) & D_n(\varphi_i, \varphi_j) & D_n(\varphi_i, \varphi_m) \\ D_n(\varphi_j, \varphi_i) & D_n(\varphi_j, \varphi_j) & D_n(\varphi_j, \varphi_m) \\ D_n(\varphi_m, \varphi_i) & D_n(\varphi_m, \varphi_j) & D_n(\varphi_m, \varphi_m) \end{bmatrix}.$$

The $[k]_{e_n}$ is the element stiff matrix. Then

$$D(u_h, w_h) = \sum_{n=1}^{N_E} \{\delta^*\}_{e_n}^T [k]_{e_n} \{\delta\}_{e_n}. \tag{1.28}$$

Meanwhile,

$$\begin{aligned} (f, w_h) &= \sum_{\alpha=1}^{N_P} w_\alpha (f, \varphi_\alpha) \\ &= \sum_{\alpha=1}^{N_P} w_\alpha \sum_{n=1}^{N_E} (f, \varphi_\alpha)_n \\ &= \sum_{n=1}^{N_E} \begin{bmatrix} w_i \\ w_j \\ w_m \end{bmatrix}^T \begin{bmatrix} (f, \varphi_i)_n \\ (f, \varphi_j)_n \\ (f, \varphi_m)_n \end{bmatrix}. \end{aligned}$$

Let

$$\{F\}_{e_n} = \begin{bmatrix} (f,\varphi_i)_n \\ (f,\varphi_j)_n \\ (f,\varphi_m)_n \end{bmatrix}.$$

The $\{F\}_{e_n}$ is the element load. Then

$$(f, w_h) = \sum_{n=1}^{N_E} \{\delta^*\}_{e_n}^T \{F\}_{e_n} \tag{1.29}$$

After putting (1.28) and (1.29) into (1.27), we have

$$\sum_{n=1}^{N_E} \{\delta^*\}_{e_n}^T [k]_{e_n} \{\delta\}_{e_n} - \sum_{n=1}^{N_E} \{\delta^*\}_{e_n}^T \{F\}_{e_n} = 0. \tag{1.30}$$

After superimposing $[k]_{e_n}$, $\{\delta^*\}_{e_n}$, $\{\delta\}_{e_n}$, and $\{F\}_{e_n}$ to $[K]$, $\{\delta^*\}$, $\{\delta\}$ and $\{F\}$ respectively, we obtain

$$\{\delta^*\}^T [K]\{\delta\} = \{\delta^*\}^T F.$$

Since $\{\delta^*\}$ is arbitrary, we can get a linear system

$$[K]\{\delta\} = F.$$

1.4 Current Work on the Dam Overflow Problem

Based on [4] and [14], Chapter 2, Chapter 3 and Chapter 4 of this thesis will expose the author's work on the dam overflow problem. Two mathematical models are adopted in Chapter 2 and Chapter 3 in which numerical results as well as theoretical results are presented. The work, divided into three parts, is detailed in the following.

1. In Chapter 2, the equations of the potential function model,

$$-\Delta\phi(x,y) = 0 \quad \text{in} \quad \Omega \tag{1.31}$$

$$\frac{\partial\phi(x,y)}{\partial n}\Big|_{s1} = -\frac{q}{h} \tag{1.32}$$

$$\phi(x,y)\Big|_{s2} = 0 \tag{1.33}$$

$$\frac{\partial\phi(x,y)}{\partial n}\Big|_{s3} = 0 \tag{1.34}$$

$$\frac{\partial\phi(x,y)}{\partial n}\Big|_{s4} = 0 \tag{1.35}$$

$$\frac{\partial\phi(x,y)}{\partial s}\Big|_{s4} = \sqrt{2g(E_0 - y)} \tag{1.36}$$

is presented, the ready-made algorithm of (1.31 – 1.36) (see 2.2 of this thesis) usually preferred by engineers will be adopted. As the original algorithm becomes unstable when iterating and consumes relatively long time, we modify the original algorithm so that it runs faster and be more stable by introducing the relaxation factor method. Moreover the programs are developed on the FEPG (Finite Element Program Generator) platform aiming at verifying the effectiveness of the relaxation factor and its practical use as well.

2. Since the free boundary condition (1.36) of the potential function model is given with tangential derivative, it is difficult to transform the potential function model into the variational form. We naturally think about using the stream function, which is the conjugate function of the potential function. Because the tangential derivative of the potential function is exactly the minus of the normal derivative of the stream function, the obstacle of using the potential function for transforming into variational form can be averted. The stream function model constructed in [14] is considered in Chapter 3. The equations for the stream function model are:

$$-\Delta\psi(x,y) = 0 \tag{1.37}$$

$$\psi(x,y)\Big|_{s1} = (c-q) + \frac{hy}{h} \tag{1.38}$$

$$\frac{\partial\psi(x,y)}{\partial n}\Big|_{s2} = 0 \tag{1.39}$$

$$\psi(x,y)\Big|_{s3} = c - q \tag{1.40}$$

$$\psi(x,y)\Big|_{s4} = c \tag{1.41}$$

$$\frac{\partial\psi(x,y)}{\partial n}\Big|_{s4} = -\sqrt{2(\lambda - gy)}. \tag{1.42}$$

From the stream function model we get

(a) the existence of the solution to the original equations (1.31)–(1.36);

(b) the existence of the discrete solution to the corresponding numerical scheme;

(c) the convergence of the discrete solutions to the exact solution of (1.31)–(1.36).

In the meantime, we develop the programs for computing the stream function model.

3. In Chapter 4, after getting the computational outputs of the potential function model and the stream function model, we compare the advantages or disadvantages between the two algorithms.

Chapter 2

The Potential Function Model

In this chapter, we focus on improving the original algorithm and developing programs to obtain the numerical results of the potential function model. Section 1 and section 2 consist of the preliminary work. We deduce the mathematical model in section 1 and depict in detail the original algorithm in section 2. Section 3 and section 4 consist of my major work. In section 3 we introduce how to use the relaxation factor and its advantage and the implementation in modifying the original algorithm. Moreover, to demonstrate the advantage of using relaxation factor, we compare a series of data taken from computation of the modified algorithm with the relaxation factor adding in and the original algorithm without the relaxation factor. In the last section, we show the numerical results by graphics.

2.1 Mathematical Model

The potential function model describes how the water is flowing over a vertical dam. In Figure 2.1, we depict the cross-section of the river water and the dam. The water is assumed flowing from left to right and passing over the vertical dam of height H. The height of the water level h at a great distance $h1$ from the dam is assumed known. All the data used in the computation in this thesis are in consistent units.

2.1.1 The Diagram for the Dam Overflow Problem

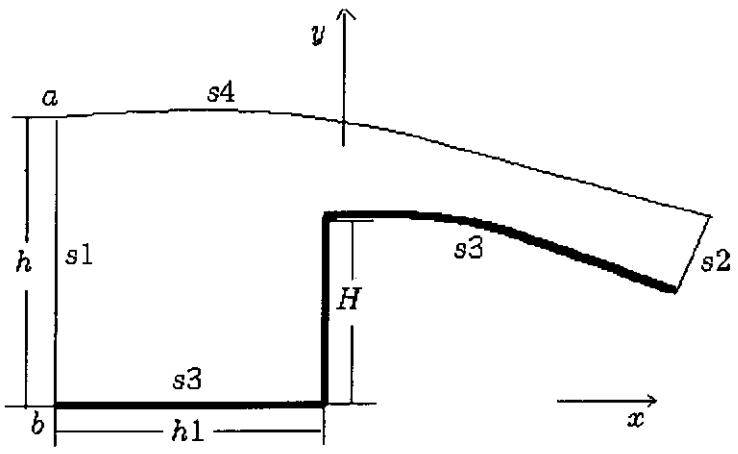


Figure 2.1

In the figure, we use the following notations:

$s1$ – entrance	$s2$ – exit
$s3$ – the body of the dam	$s4$ – free boundary (the water flow surface)
H – height of the dam	h – height of the water

The unknown parameters are the flux (q), the flow velocity ($\vec{V}$), the flow energy (E) and the free boundary ($s4$). To obtain these, mathematical model should be constructed.

2.1.2 Construction of the Mathematical Model

Since the water flow is a very complicated problem, for simplicity we assume that the water flow is a incompressible and irrotational here. Hence one can introduce the potential function $\phi(x, y)$ with the property

$$\begin{aligned} \vec{V} &= (V_1(x,y), V_2(x,y)) \\ &= \nabla\phi(x,y) \\ &= \left(\frac{\partial\phi(x,y)}{\partial x}, \frac{\partial\phi(x,y)}{\partial y}\right) \end{aligned}$$

where $\vec{V}$ is the flow velocity, which is a vector consisting of two components, V_1 and V_2.

As the water is incompressible, the divergence of the potential function is zero, giving

$$\Delta\phi(x,y) = \mathrm{div}(\nabla\phi(x,y)) = \mathrm{div}\vec{V} = 0. \tag{2.1}$$

The boundary conditions are given as follows:

(1) On $s1$

From Figure 2.1, one can assume that $h1$ is large enough, which means the flow entrance is far away from the dam. So the hypothesis that the water around the entrance $s1$ flows at average speed is reasonable, which means $\vec{V}\cdot\vec{n} = -\frac{q}{h} = \text{const.}$ That is to say,

$$\frac{\partial\phi(x,y)}{\partial n}\Big|_{s1} = -\frac{q}{h} \tag{2.2}$$

where $\vec{n}$ is the outward normal direction and q is the flux.

(2) On $s2$

The shape of exit can be made arbitrarily and is insignificant in the model. For simplicity, s2 can be considered as an equi-potential line. On the other hand, the value of equi-potential line is relative, therefore, for convenience, its value is set to zero, i.e.

$$\phi(x,y)\Big|_{s2} = 0 \tag{2.3}$$

(3) On $s3$

Since $s3$ is the fixed boundary, it is easy to know that there is no variation of the fluid along the normal direction of $s3$, i.e.

$$\frac{\partial\phi(x,y)}{\partial n}\Big|_{s3} = 0 \tag{2.4}$$

(4) On $s4$

Because $s4$ is the free boundary, it needs two conditions, the ordinary boundary condition and the free boundary condition. As the same reason given in (3) that there is no variation of the fluid along the normal direction of $s4$,

$$\frac{\partial\phi(x,y)}{\partial n}\Big|_{s4} = 0. \tag{2.5}$$

Meanwhile, by the Bernoulli-Energy Equation (see A.1),

$$\frac{\|\vec{V}\|^2}{2g} = E_0 - y - \frac{P_0}{\rho g}$$

where $\vec{V}$ is the flow velocity, E_0 the initial energy, y the vertical coordinate, P_0 the air pressure, g the gravity and ρ the water density.

Since E_0 and $\frac{P_0}{\rho g}$ are constant, we merge them into one term and for simplicity denote $E_0 - \frac{P_0}{\rho g}$ by E_0. Let s be the length parameter such that s is increasing as we traverse along the boundary in the counter-clockwise sense. Then, along $s4$,

$$\begin{aligned}\frac{\partial\phi(x,y)}{\partial s} &= \sqrt{\Big(\frac{\partial\phi(x,y)}{\partial n}\Big)^2 + \Big(\frac{\partial\phi(x,y)}{\partial s}\Big)^2 - \Big(\frac{\partial\phi(x,y)}{\partial n}\Big)^2} \\ &= \sqrt{\|\vec{V}\|^2 - \Big(\frac{\partial\phi(x,y)}{\partial n}\Big)^2} \\ &= \sqrt{\|\vec{V}\|^2} \\ &= \sqrt{2g(E_0 - y)} \qquad (2.6)\end{aligned}$$

By combining all equations from (2.1) to (2.6), the potential function model is

$$-\Delta\phi(x,y) = 0 \quad \text{in} \quad \Omega \qquad (2.1)$$

$$\frac{\partial\phi(x,y)}{\partial n}\Big|_{s1} = -\frac{q}{h} \qquad (2.2)$$

$$\phi(x,y)\Big|_{s2} = 0 \qquad (2.3)$$

$$\frac{\partial\phi(x,y)}{\partial n}\Big|_{s3} = 0 \qquad (2.4)$$

$$\frac{\partial\phi(x,y)}{\partial n}\Big|_{s4} = 0 \qquad (2.5)$$

$$\frac{\partial\phi(x,y)}{\partial s}\Big|_{s4} = \sqrt{2g(E_0 - y)} \qquad (2.6)$$

where q is the unknown flux and $s4$ is the unknown free boundary to be determined.

The above equations are non-linear due to its free boundary which brings more difficulty in either developing theoretical results or getting numerical solutions.

2.2 The Algorithm

2.2.1 Outline of the Algorithm

The algorithm for the iteration processes are briefly described in the following steps:

1. Consider only the equations (2.1) – (2.5) and form the following boundary

value problem:

$$\begin{cases} -\Delta\phi(x,y) = 0 \quad \text{in} \quad \Omega \\ \dfrac{\partial\phi(x,y)}{\partial n}\Big|_{s1} = -\dfrac{q_0}{h} \\ \phi(x,y)\Big|_{s2} = 0 \\ \dfrac{\partial\phi(x,y)}{\partial n}\Big|_{s3} = 0 \\ \dfrac{\partial\phi(x,y)}{\partial n}\Big|_{\Gamma_0} = 0 \end{cases} \tag{2.7}$$

2. Give q_0 and Γ_0 (here Γ_0 in place of $s4$ in Figure 2.1) initial values;
3. Solve (2.7) by the finite element method to get a numerical solution $\phi(x,y)$.
4. Obtain the new flux q and new boundary Γ using the algorithm provided in [4].
5. Repeat step 2, step 3 and step 4 until convergence is reached (it means the errors of the free boundary and the flux between the two consecutive iterating process are less than the tolerance ϵ given in advance).

2.2.2 Anatomy of the Algorithm

1. The displacement feature of the free boundary

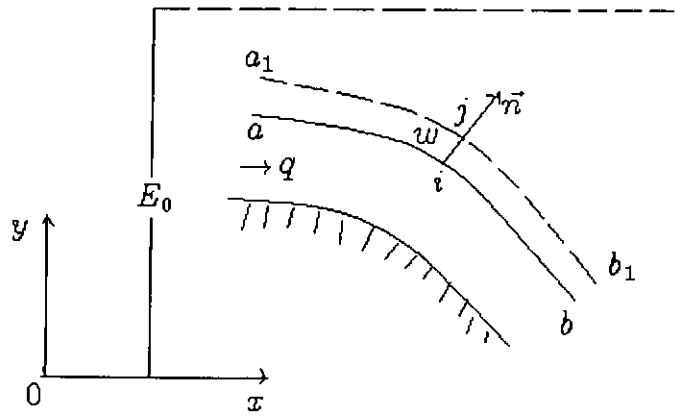


Figure 2.2

In Figure 2.2, H_p (water pressure) of the ith point is changing with the normal displacement w (outward is positive). Consider

$$H_p(w) \equiv f(w) - F(w) \tag{2.8}$$

where

$$f(w) = E_0 - y - w\cos\theta, \tag{2.9}$$

$$F(w) = [u(w)]^2/2g. \tag{2.10}$$

In (2.10) $u(w)$ is the speed of the ith point and is a function of w. In (2.9) θ is the angle between y-axis and $\vec{ij}$. In general, $H_p(0) \neq 0$ along an arbitrary surface. However, using the Bernoulli-Energy Equation, if $\widehat{ab}$ is the actual free boundary surface, we know that

$$H_p(0) = f(0) - F(0) = 0 \tag{2.11}$$

The linear parts of the Taylor expansion of (2.8) and (2.9) are

$$f(w) = f(0) + f'(0)w \tag{2.12}$$

$$F(w) = F(0) + F'(0)w. \tag{2.13}$$

And after substituting into (2.8) with (2.12) and (2.13),

$$H_p(w) = (f'(0) - F'(0))w, \tag{2.14}$$

which is a property of the free surface. Generally, for a free surface,

$$F'(0) < 0,$$

$$f'(0) < 0.$$

Therefore, due to the inequality relation between $F'(0)$ and $f'(0)$, the free surface can be classified into three types below:

1. $F'(0) > f'(0)$ i.e. $|F'(0)| < |f'(0)|$
2. $F'(0) < f'(0)$ i.e. $|F'(0)| > |f'(0)|$
3. $F'(0) = f'(0)$ i.e. $|F'(0)| = |f'(0)|$.

Type 1, type 2 and type 3 are called slow flow, rapid flow and critical flow respectively.

Understanding the property of the free surface is most important for adjusting the free surface. Still referring to Figure 2.2, we assume $\widehat{ab}$ is the nth iterating free surface. Meanwhile, if $H_P(0) \neq 0$, a small value of w should be found to satisfy

$$H_P(w) = f(w) - F(w) = 0$$

If w is very small, through Taylor's expansion,

$$w = -\frac{F(0) - f(0)}{F'(0) - f'(0)} = \frac{H_p(0)}{F'(0) - f'(0)}. \tag{2.15}$$

The following list is the detail of the free surface feature.

	$H_p(0) > 0$	$H_p(0) < 0$
Slow flow $F'(0) > f'(0)$	$w > 0$	$w < 0$
Rapid flow $F'(0) < f'(0)$	$w < 0$	$w > 0$

The above list indicates that the adjusting directions for slow flow and the rapid flow are exactly opposite.

2. Computation of the flux

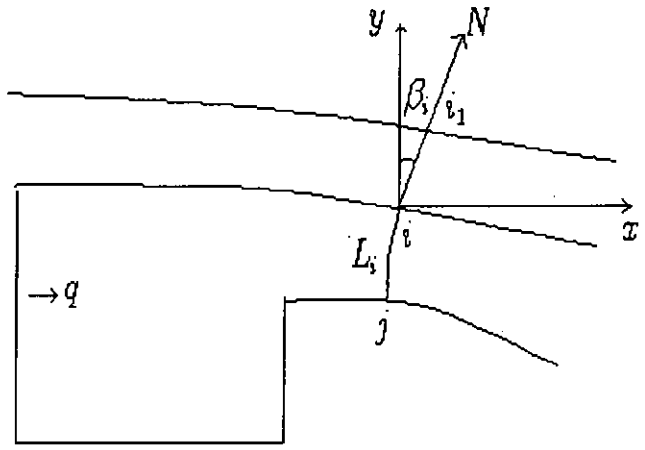


Figure 2.3

The method is to find the approximate flux for the next iteration and to adjust the free surface. In Figure 2.3, after obtaining the nth flow, the speed $u_{0i}(0)$ of ith point can be calculated (since speed is the gradient of the potential function). From the ith point one can draw an equi-potential line $\widehat{ij}$ which is a curve throughout the whole flow field. The length of $\widehat{ij}$ can be computed using interpolation. Now

a small displacement w_i along the normal is given. Under the condition that the flow velocity along $\widehat{i_1 j}$ is similar to that along $\widehat{ij}$, we have

$$\frac{u_{0i}(0)}{\bar{u}_i(0)} = \frac{u_{0i}(w_i)}{\bar{u}_i(w_i)} = K_i \tag{2.16}$$

$$\bar{u}_i(0) = \frac{q}{L_i} \tag{2.17}$$

$$\bar{u}_i(w_i) = \frac{q}{L_i + w_i} \tag{2.18}$$

where K_i a proportional constant.

By combining (2.16), (2.17) and (2.18),

$$u_{0i}(w_i) = \frac{K_i q}{L_i + w_i} \tag{2.19}$$

Since $\frac{\partial \phi}{\partial s} = u_{0i}(w_i)$ in (2.6), and using (2.9)-(2.10),

$$\begin{aligned} H_p(w_i) &= f(w_i) - F(w_i) \\ &= h + \frac{q^2}{2gh^2} - y_i - w_i \cos\theta_i - \frac{k_i^2 q^2}{2g(L_i + w_i)^2} \end{aligned}$$

The requirement that $\widehat{ab}$ is the real free surface is $H_p(w_i) = 0$, from which the equation to reveal the relation between q and w_i,

$$h + \frac{q^2}{2gh^2} = y_i + w_i \cos\theta_i + \frac{k_i^2 q^2}{2g(L_i + w_i)^2} \tag{2.20}$$

is achieved.

For convenience, now introduce the following non-dimensional numbers:

$$\begin{aligned} q_{i*} &= \frac{q^2}{gh^3} \\ h_{i*} &= \frac{L_i + w_i}{h} \\ \theta_{i*} &= \cos\theta_i \\ z_{i*} &= \frac{h - y_i + L_i \cos\theta_i}{h \cos\theta_i} \end{aligned}$$

Consequently, (2.20) is changed to

$$q_{i*} = \frac{2h_{i*}^2 \theta_{i*}(h_{i*} - z_{i*})}{h_{i*}^2 - K_i^2} \tag{2.21}$$

When q_{i*} reaches its maximum,

$$\frac{dq_{i*}}{dh_{i*}} = 0$$

i.e.

$$h_{i*}^3 - 3K_i^2 h_{i*} + 2K_i^2 z_{i*} = 0. \tag{2.22}$$

It can be proved that the condition that the equation (2.22) in h_{i*} has a positive root $h_{m_{i*}} \in (0, K_i)$ is

$$z_{i*} < K_i. \tag{2.23}$$

In other word, on each point of the free surface satisfying (2.23), there must have an extremum $q_{m_{i*}}$ which is the extreme flux. Since $h_{m_{i*}} < K_i$,

$$\left.\frac{d^2 q_{i*}}{d^2 h_{i*}}\right|_{h_{i*}=h_{m_{i*}}} = \frac{6\theta_{i*} h_{m_{i*}}}{h_{m_{i*}}^2 - K_i^2} < 0$$

Hence $q_{m_{i*}}$ is a maximum.

The $h_{m_{i*}}$ can be obtained from (2.22) through either iteration (2.22) or analytic solution as

$$h_{m_{i*}} = 2K_i \cos(60^0 - \frac{\alpha}{3}) \quad \text{where} \quad \cos\alpha = \frac{z_{i*}}{K_i}$$

Therefore

$$q_{m_{i*}} = \frac{\theta_{i*} h_{m_{i*}}^3}{K_i^2} \tag{2.24}$$

and

$$q_{m_i} = \sqrt{g q_{m_{i*}}} h^{\frac{3}{2}}, \tag{2.25}$$

through substituting $h_{m_{i*}}$ to h_{i*} in (2.21), where q_{m_i} is the extremum corresponding to q in (2.20).

From (2.25), one can get the maximum flux of each point suitable to (2.23). Since the actual flux must be less than or equal to the maximum flux of every point, the minimum of all these maximum flux is the possible maximum flux, denoted by $q^{(n+1)}$, for the next iteration. That is

$$q^{(n+1)} = \min_i q_{m_i} = q_{m_k}$$

where k is the number of point at which the possible maximum flux reaches. Since

$$\begin{aligned} f'(w_{m_k}) &= -\cos\theta_k = -\theta_{k*} \quad \text{from (2.9)}, \\ F'(w_{m_k}) &= -\frac{K_k^2 q_{m_k}^2}{g(L_k + w_{m_k})^3} = -\frac{K_k^2 q_{m_{k*}}^2}{h_{m_{k*}}^3} \quad \text{from (2.10) and (2.19)}. \end{aligned}$$

And from (2.24),

$$f'(w_{m_k}) = F'(w_{m_k}).$$

So kth point is the critical point of $(n+1)$th flow and its adjustment of the free surface, w_{m_k}, can be taken through (2.24).

3. Adjustment of the free boundary

After getting nth flow field, the flux q of the next iteration, critical point number k and the adjustment of the critical point w_k, we can adjust the free surface for both slow flow and rapid flow. The method is described below.

For the discrete flow field, consider an arbitrary point i on the free surface. The equation (2.20) can be expressed in the form

$$f(w_i) = F(w_i) \tag{2.26}$$

$$f(w_i) = h + \frac{(q^{(n+1)})^2}{2gh} - y_i - w_i\cos\theta_i \tag{2.27}$$

$$F(w_i) = \frac{K_i^2 (q^{(n+1)})^2}{2g(L_i + w_i)^2} \tag{2.28}$$

where q_{n+1} can be obtained in the above section. So one can solve (2.26) for w_i. (2.26) is a cubic equation,

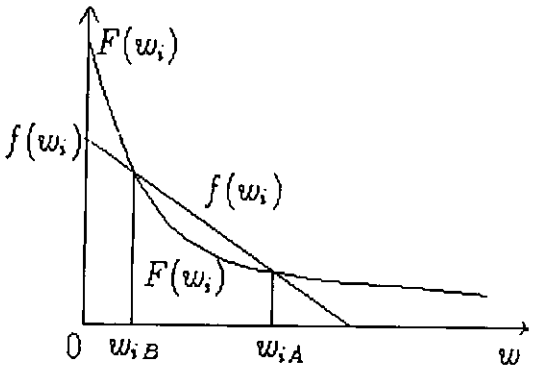


Figure 2.4

and in Figure 2.4, it is known that (2.26) has two kinds of solutions, slow flow solution and rapid flow solution. To get them, one need to adopt different iteration strategies. For the points above the critical point, since $|F'(0)| > |f'(0)|$, the slow flow form

$$F(w_i^r) = f(w_i^{r+1}) \tag{2.29}$$

is adopted. And for the points below the critical point, since $|F'(0)| < |f'(0)|$, the rapid flow form

$$F(w_i^{r+1}) = f(w_i^r) \tag{2.30}$$

is adopted, r and $r+1$ being iteration numbers.

2.3 Relaxation Factor Method

2.3.1 Introduction

We now use an example to explain how to use relaxation factor method. Suppose we want to obtain the solution of the equation

$$X = f(X) \tag{2.31}$$

where the X is a vector. The usual simple iteration $X_{n+1} = f(X_n)$ can be put in the form

$$\begin{cases} \Delta X_{n+1} = f(X_n) - X_n, \\ X_{n+1} = X_n + \Delta X_{n+1}. \end{cases} \tag{2.32}$$

To improve the convergence, instead of the above form we use

$$\begin{cases} \Delta X_{n+1} = f(X_n) - X_n, \\ X_{n+1} = X_n + \kappa_\theta \Delta X_{n+1}. \end{cases} \tag{2.33}$$

where κ_θ is a scalar and $\kappa_\theta = \kappa(\theta)$; θ is the angle between ΔX_n and ΔX_{n+1} (see Figure 2.5).

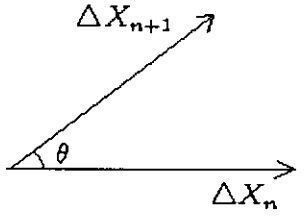


Figure 2.5

The function $\kappa(\theta)$ is a monotone decreasing function. The formula in (2.33) described a relaxation factor method and κ_θ is called the relaxation factor.

Using a good relaxation factor does improve the speed of convergence. In (2.33), when θ is large, which means that the trends of the two consecutive iterations differ considerably. In this case, we need to decrease $\kappa(\theta)$ to slow down these trends. Inversely, we need to increase $\kappa(\theta)$ when θ is small. Through using relaxation factor, we slow down the iterating speed when instability appears whereas we accelerate the iterating speed when the iteration is considerable stable.

2.3.2 Implementation

The purpose of introducing relaxation factor method is to speed up the convergence and stabilize the iteration. It is used in step 4 in 2.2.1 of this thesis to improve the original algorithm. And the following gives the details of the implementation.

We use the vectors Γ_0, Γ_1 and Γ_2 to represent the discrete forms of the former free boundary, the current free boundary and the next free boundary respectively. The elements of Γ_0, Γ_1 and Γ_2 are coordinates. And use vectors $\Delta\Gamma_2$ and $\Delta\Gamma_1$ to represent the increments from Γ_2 to Γ_1 and from Γ_1 to Γ_0, where

$$\Delta\Gamma_2 = (x_{i_2}), \quad \Delta\Gamma_1 = (x_{i_1}),$$

and x_{i_2} and x_{i_1} are the increments of ith point along the normal direction of Γ_1 and Γ_0 respectively (see Figure 2.6).

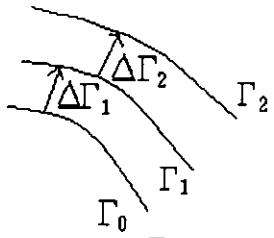


Figure 2.6

The angle between $\Delta\Gamma_1$ and $\Delta\Gamma_2$ (see Figure 2.7) is

$$\theta = \frac{\Delta\Gamma_1 \cdot \Delta\Gamma_2}{\|\Delta\Gamma_1\|\|\Delta\Gamma_2\|}$$

$$= \frac{\sum_{i=1}^{n} x_{i_1} \cdot x_{i_2}}{\sqrt{\sum_{i=1}^{n} x_{i_1}^2} \sqrt{\sum_{i=1}^{n} {x_{i_2}}^2}}$$

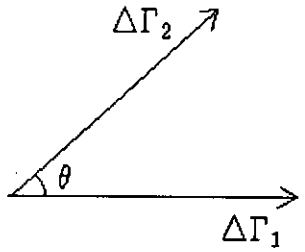


Figure 2.7

In the process of iteration, when θ is large, the trend of the iteration is not beneficial to convergence, whereas the trend speeds up convergence when θ is small. The question is how to avert the drawback caused by large θ and to make use of the good effect due to the small θ. For that, we introduce a scalar Υ called the relaxation factor associated with θ. Using relaxation factor by the approach of replacing $\Delta\Gamma_2$ by $\Upsilon\Delta\Gamma_2$ in the next iteration will achieve this goal. In general, the relaxation factor would be much less than 1 when the trend is disadvantage with θ being large as well as it would be more than 1 when the trend is advantage with θ being small.

The relation of Υ and θ in the computing process is presented by a monotone increasing and piecewise constant function such that

$$\Upsilon = f(\theta) = \begin{cases} c_1, & a_1 \leq \cos\theta < a_2 \\ c_2, & a_2 \leq \cos\theta < a_3 \\ c_3, & a_3 \leq \cos\theta < a_4 \\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots & \\ c_{n-1} & a_{n-1} \leq \cos\theta < a_n, \end{cases} \tag{2.34}$$

where $-1 \leq a_1 < a_2 < \cdots < a_n \leq 1$ and $0 < c_1 < c_2 < \cdots < c_{n-1}$. Under such relaxation factor, the new free boundary is $\Gamma_2 = \Gamma_1 + f(\theta)\Delta\Gamma_i$.

The function is constant over each interval $[a_i, a_{i+1}]$, $i = 1, ..., n-1$. The reason why choosing such a function as a relaxation factor is that it is easy for programming. Since one don't know previously how to choose a series of data as relaxation factor to reach the optimal in the sense of convergence, we may change the relaxation factor many times by modifying the program to meet the optimal.

Choosing such function is convenient for modification.

In the following we give a subroutine written with algorithm language to demonstrate how to use the relaxation factor method in our actual computation.

RX=2

IF $\cos\theta$ <= 0.8 RX=RX*0.5

IF $\cos\theta$ <= 0.6 RX=RX*0.5

IF $\cos\theta$ <= 0.4 RX=RX*0.5

IF $\cos\theta$ <= 0.2 RX=RX*0.5

IF $\cos\theta$ <= 0.0 RX=RX*0.5

IF $\cos\theta$ <= -0.5 RX=RX*0.5

where RX is relaxation factor and θ is the angle between $\Delta\Gamma_1$ and $\Delta\Gamma_2$. The subroutine is the process of assigning value to Υ according to (2.34) with c_i and a_i given as follows:

$$
\begin{array}{ll}
a_1 = -1, & \\
a_2 = -0.5, & c_1 = 2 \times 0.5^6 \\
a_3 = 0.0, & c_2 = 2 \times 0.5^5 \\
a_4 = 0.2, & c_3 = 2 \times 0.5^4 \\
a_5 = 0.4, & c_4 = 2 \times 0.5^3 \\
a_6 = 0.6, & c_5 = 2 \times 0.5^2 \\
a_7 = 0.8, & c_6 = 2 \times 0.5 \\
a_8 = 1.0, & c_7 = 2
\end{array}
$$

In the above subroutine, the relaxation factor RX= 2, if $\cos\theta > 0.8$, RX= 1, if $0.6 < \cos\theta <= 0.8$, RX= 0.5, if $0.4 < \cos\theta <= 0.6$ and so on.

2.3.3 Advantage of Using Relaxation Factor

For explaining the advantage of using relaxation factor, we list the CPU time and iterating number of the iteration added in relaxation factor and the iteration without any relaxation factor.

Iterating number

Error	Iteration with relaxation factor	Iteration without relaxation factor
$e < 0.001$	74	102
$e < 0.1$	32	46

CPU time

Error	Iteration with relaxation factor	Iteration without relaxation factor
$e < 0.001$	47 seconds	65 seconds
$e < 0.1$	20 seconds	29 seconds

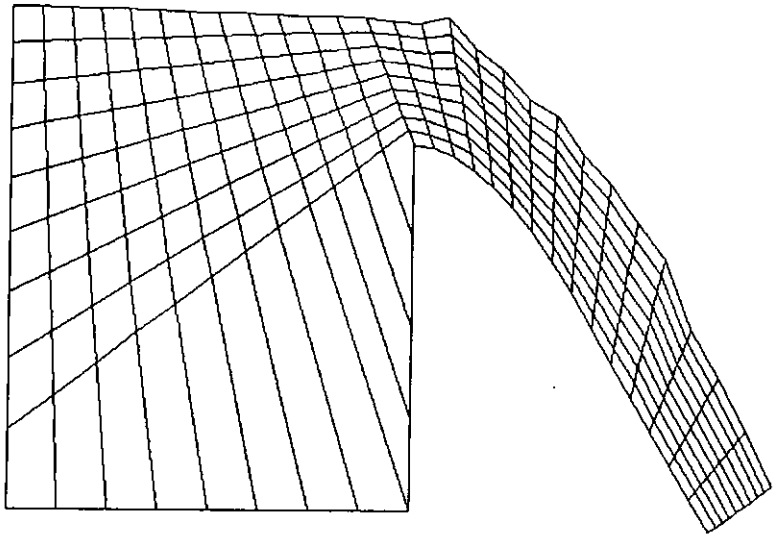
Note: the CPU time is taken from DX486 computer.

The above two lists obviously indicate that using relaxation factor can save time and overcome oscillation, consequently, speed up the iteration to convergence.

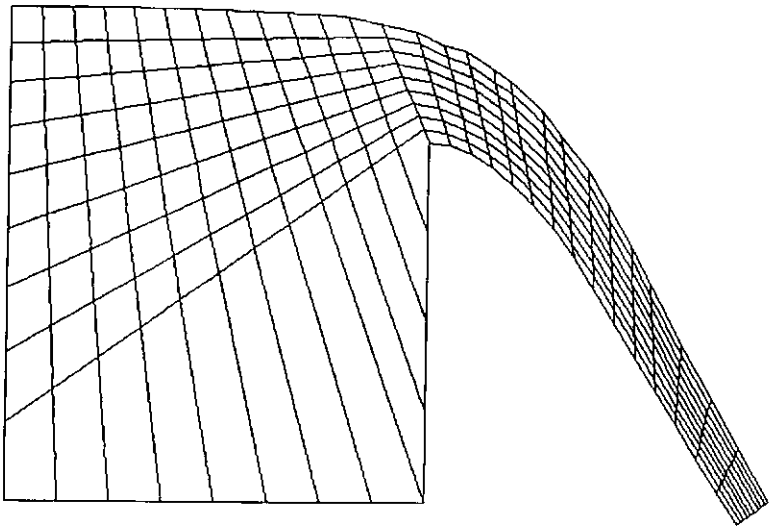
2.4 Numerical and Graphical Results

a. Free Boundary

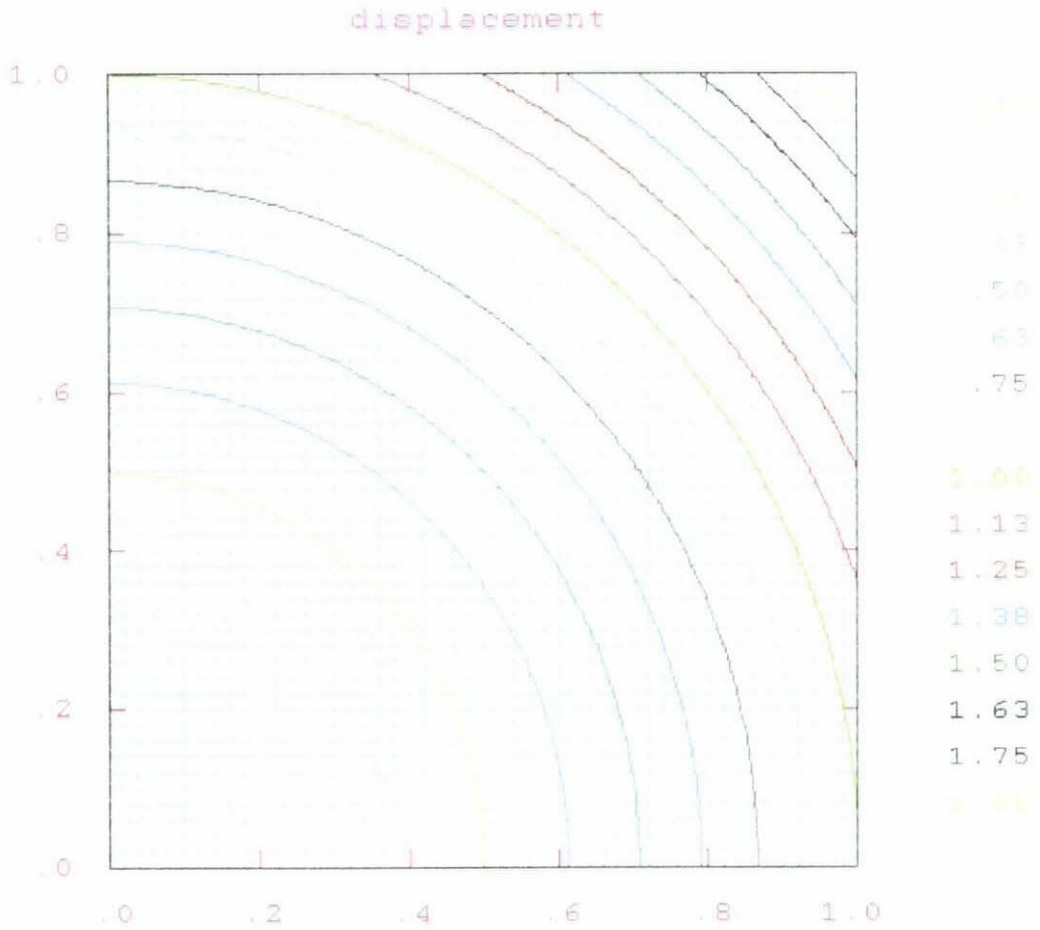
Initial case ($q_0 = 85.0$)



Final case ($q = 298.39947$)



b. Equi-potential lines



Chapter 3

The Stream Function Model

3.1 Mathematical Model

In Chapter 2, the potential function model was established. Considering the drawback of the potential function model for developing theoretical results due to the presence of tangential derivatives in the boundary conditions, the stream function, which is the conjugate function of the potential function $\phi(x, y)$, will be used in this section for theoretical results.

Let the stream function be $\psi(x, y)$, and by the Cauchy-Riemann Condition, we have

$$\frac{\partial \phi(x,y)}{\partial x} = \frac{\partial \psi(x,y)}{\partial y}, \quad \frac{\partial \phi(x,y)}{\partial y} = -\frac{\partial \psi(x,y)}{\partial x} \tag{3.1}$$

Therefore

$$\frac{\partial \phi(x,y)}{\partial n} = \frac{\partial \psi(x,y)}{\partial s} \tag{3.2}$$

$$\frac{\partial \phi(x,y)}{\partial s} = -\frac{\partial \psi(x,y)}{\partial n}. \tag{3.3}$$

In the following, we transform (2.1)-(2.6) one by one by utilizing (3.1)-(3.3).

From (2.1) and (3.1)

$$\begin{aligned}
\Delta\psi(x,y) &= \frac{\partial^2 \psi(x,y)}{\partial x^2} + \frac{\partial^2 \psi(x,y)}{\partial y^2} \\
&= \frac{\partial}{\partial x}\Big(\frac{\partial \psi(x,y)}{\partial x}\Big) + \frac{\partial}{\partial y}\Big(\frac{\partial \psi(x,y)}{\partial y}\Big) \\
&= \frac{\partial}{\partial x}\Big(-\frac{\partial \phi(x,y)}{\partial y}\Big) + \frac{\partial}{\partial y}\Big(\frac{\partial \phi(x,y)}{\partial x}\Big)
\end{aligned}$$

$$= -\frac{\partial^2\phi(x,y)}{\partial x\partial y} + \frac{\partial^2\phi(x,y)}{\partial x\partial y}$$
$$= 0$$

i.e.

$$-\Delta\psi(x,y) = 0 \tag{3.4}$$

From (2.2) and (3.2),

$$\frac{\partial\psi(x,y)}{\partial s} = \frac{\partial\phi(x,y)}{\partial n} = -\frac{q}{h}.$$

Since along s_1

$$\frac{\partial\psi(x,y)}{\partial s} = \Big(\frac{\partial\psi(x,y)}{\partial x}, \frac{\partial\psi(x,y)}{\partial y}\Big)\cdot(0,-1)$$
$$= -\frac{\partial\psi(x,y)}{\partial y},$$

we have, referring to Figure 2.1,

$$\psi(x,y)|_{s1} = \psi(b) + \int_0^y \frac{\partial\psi(x,y)}{\partial y}dy$$
$$= \psi(b) + \int_0^y -\frac{\partial\psi(x,y)}{\partial s}dy$$
$$= \psi(b) + \int_0^y \frac{q}{h}dy$$
$$= \psi(b) + \frac{qy}{h}$$

Let $\psi(a) = c$, then

$$\psi(x,y)|_{s1} = (c-q) + \frac{qy}{h} \tag{3.5}$$

From (2.3), $\frac{\partial\phi(x,y)}{\partial s}|_{s2} = 0$, since $\phi(x,y)|_{s2} = 0$. Then through (3.3)

$$\frac{\partial\psi(x,y)}{\partial n} = 0. \tag{3.6}$$

From (2.4) and (3.2),

$$\frac{\partial\psi(x,y)}{\partial s}|_{s3} = \frac{\partial\phi(x,y)}{\partial n}|_{s3}$$
$$= 0.$$

So $\psi(x,y)|_{s3}$ must be constant and by the continuity of $\psi(x,y)$ and using $\psi(b) = c - q$, we can get

$$\psi(x,y)|_{s3} = c - q \tag{3.7}$$

For the same reason of getting (3.7) and $\psi(a) = c$,

$$\psi(x,y)|_{s4} = c \tag{3.8}$$

from (2.5).

By (3.2), (3.3) and putting λ in place of gE_0, it is easy to get

$$\frac{\partial\psi(x,y)}{\partial n}\Big|_{s4} = -\sqrt{2(\lambda - gy)} \tag{3.9}$$

from (2.6).

Overall, the stream function model is

$$-\Delta\psi(x,y) = 0 \tag{3.4}$$

$$\psi(x,y)\Big|_{s1} = (c-q) + \frac{qy}{h} \tag{3.5}$$

$$\frac{\partial\psi(x,y)}{\partial n}\Big|_{s2} = 0 \tag{3.6}$$

$$\psi(x,y)\Big|_{s3} = c - q \tag{3.7}$$

$$\psi(x,y)\Big|_{s4} = c \tag{3.8}$$

$$\frac{\partial\psi(x,y)}{\partial n}\Big|_{s4} = -\sqrt{2(\lambda - gy)} \tag{3.9}$$

3.2 Variational Form

To facilitate further discussion, we depict the cross-section of the water flowing over vertical dam in Figure 3.1.

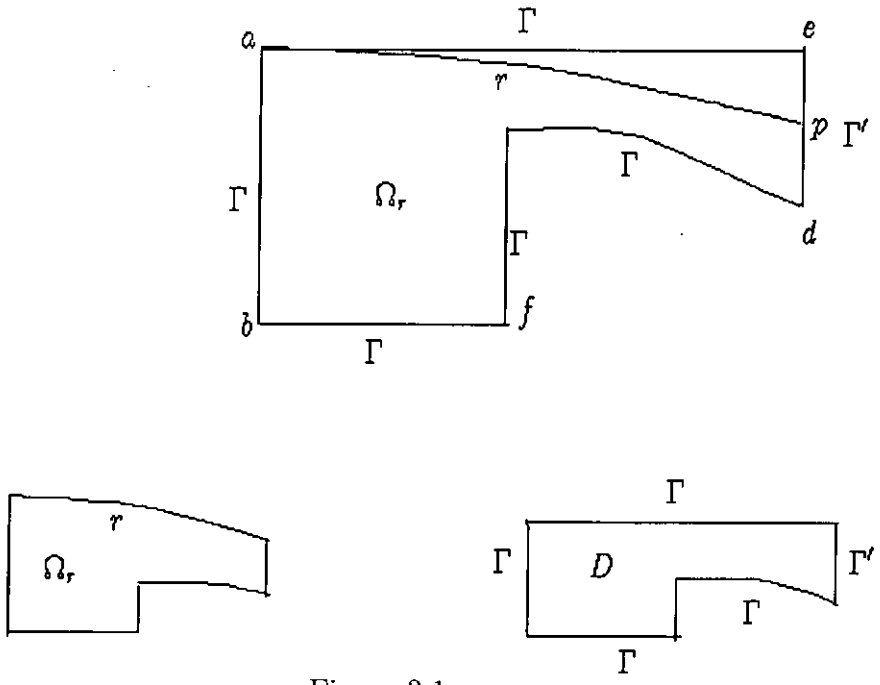


Figure 3.1

In Figure 3.1, $\Gamma = ab \cup ae \cup \widehat{bfd}$ and $\Gamma' = ed$.

Let

$$J(v,r) = \int_{\Omega_r} \left[\frac{1}{2}|\nabla v|^2 + (\lambda - gy)\right] dx\, dy$$

$$K = \left\{v | v \in H^1(D), v|_r = c, v|_{ab} = (c-q) + \frac{qy}{h}, v|_{\widehat{bfd}} = c - q\right\}$$

$$Z = \{r | \text{ the smooth lines connecting } a \text{ to } de\}$$

Ω_r is the domain bounded by $r \in Z$ and $\widehat{abfdp}$.

From [14] we know that $\psi(x,y)$ and $r_0 \in Z$ form the solution of (3.4)−(3.9) if $\psi(x,y)$ and r_0 form the solution of the variational problem:

$$J(u, r_0) = \min_{v \in K, r \in Z} J(v, r). \tag{3.10}$$

From the extremum principle, $\psi(x,y)$ is less than c within Ω. After being extended from Ω_r to D such that $\psi(x,y) = 0, \quad (x,y) \in D - \Omega_r$, $\psi(x,y)$ satisfies

the following variational problem:

$$J^*(u) = \min_{v \in K^*} J^*(v) \tag{3.11}$$

where

$$J^*(v) = \int_D \left[\frac{1}{2}|\nabla v|^2 + (\lambda - gy)\chi(c - v)\right] dx\, dy$$
$$K^* = \{v | v \in H^1(D), v|_\Gamma = G(x,y), v|_{\Gamma'} \leq c\}$$
$$G(x,y)|_{ae} = c$$
$$G(x,y)|_{ab} = (c - q) + \frac{qy}{h}$$
$$G(x,y)|_{\widehat{bcd}} = c - q$$

and

$$\chi(x) = \begin{cases} 0, & x \leq 0 \\ 1, & \text{otherwise} \end{cases}$$

3.3 Theoretical Results

This section is the major content of Chapter 3, where we will prove some theoretical results based on the stream function model given in [14].

3.3.1 Existence of the Variational Form

Theorem 3.1 *If ψ is the solution of (3.11) and there exists a smooth curve r such that $\psi = c$ above r or on r, and $\psi < c$ below r, then ψ and r must be the solution of (3.10).*

Proof We prove the theorem by contradiction. If there exist u and r' satisfying

$$J(u, r') < J(\psi, r)$$

then extend u from domain $\Omega_{r'}$ to D with $u \equiv c$ outside $\Omega_{r'}$. By assumption,

$$J^*(\psi) \leq J^*(u).$$

On the other hand,

$$J^*(u) = J(u, r') \quad \text{and} \quad J^*(\psi) = J(\psi, r)$$

So

$$J(\psi, r) = J^*(\psi) \leq J^*(u) = J(u, r') < J(\psi, r)$$

This is a contradiction, so the theorem is proved. □

Theorem 3.1 not only indicates that (3.10) and (3.11) are equivalent but also give how to get the free boundary. In fact, we see that r satisfying the conditions in the theorem is exactly the free boundary.

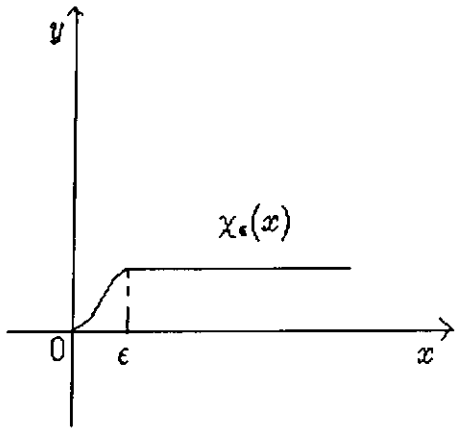


Figure 3.2

Next, we prove that the solution of (3.11) exists. To do this, χ in (3.11) need smoothing first as follows (see Figure 3.2): Choose a smoothed sequence $\{\chi_\epsilon\}$ satisfying

$$\begin{aligned}
&\lim_{\epsilon \to 0} \chi_\epsilon(x) = \chi(x) \\
&\chi'_\epsilon(x) \leq \frac{2}{\epsilon} \\
&\chi''_\epsilon(x) \leq \frac{4}{\epsilon^2} \\
&\chi_{\epsilon_1}(x) \leq \chi_{\epsilon_2}(x) \quad \text{if} \quad \epsilon_1 > \epsilon_2.
\end{aligned}$$

Subsequently, for each $\epsilon > 0$, we introduce the following new variational problem corresponding to (3.11)

$$J^*_\epsilon(\psi) = \min_{v \in K^*} J^*_\epsilon(v) \tag{3.12}$$

where

$$\begin{aligned}
J^*_\epsilon(v) &= \int_D \left[\frac{1}{2}|\nabla v|^2 + (\lambda - gy)\chi_\epsilon(c - v)\right] dx\, dy \\
K^* &= \{v | v \in H^1(D), v|_\Gamma = G(x, y), v|_{\Gamma'} \leq c\}
\end{aligned}$$

Lemma 3.2 *The solution of (3.12) exists.*

Proof Since $J_\epsilon^* \geq 0$, the infimum $\alpha = \inf_{v \in K^*} J_\epsilon^*(v)$ exists and is nonnegative. Also, there exists a sequence $\{v_n\} \subset K^*$ that satisfies $J_\epsilon^*(v_n) \to \alpha \quad (n \to \infty)$. Hence there exists a bound $M > 0$ such that $|J_\epsilon^*(v_n)| \leq M$ for all n. Then $\|\nabla v_n\|$ are bounded as $(\lambda - gy)\chi_\epsilon(c - v_n) \geq 0$.

Let $u \in H^1(D)$, $u|_\Gamma = G(x, y)$, then $(v_n - u)|_\Gamma = 0$. From Poincare-inequality (see Appendix A),

$$\|u - v_n\|_1 \leq C\|\nabla(u - v_n)\| \leq C\|\nabla u\| + C\|\nabla v_n\|.$$

This indicates that $\|u - v_n\|_1$ has a bound. And from

$$\|v_n\|_1 \leq \|u\|_1 + \|v_n - u\|_1,$$

we know $\|v_n\|_1$ are bounded. Therefore, there must exist $v \in H^1(D)$ and a subsequence $\{v_{n_i}\} \subset \{v_n\}$ such that

$$v_{n_i} \xrightarrow{w} v \quad \text{in } H^1(D) \tag{3.13}$$

$$v_{n_i}|_{\partial D} \xrightarrow{w} v|_{\partial D} \quad \text{in } H^{\frac{1}{2}}(\partial D) \tag{3.14}$$

Also because $H^{\frac{1}{2}}(\partial D)$ is compactly imbedded into $L^2(\partial D)$,

$$v_{n_i}|_{\partial D} \longrightarrow v|_{\partial D} \quad \text{in } L^2(\partial D) \tag{3.15}$$

$$v_{n_i} \longrightarrow v \quad \text{for each } x \in \partial D \text{ (in the sense of subsequence)} \tag{3.16}$$

For simplicity, we still denote $\{v_{n_i}\}$ by $\{v_n\}$. Then owing to (3.13),

$$\liminf \|\nabla v_n\|^2 \geq \|\nabla v\|^2 \tag{3.17}$$

Since $H^1(D)$ imbeds into $L^2(D)$ compactly,

$$v_n \longrightarrow v \quad \text{in } L^2(D) \tag{3.18}$$

$$v_n \longrightarrow v \quad \text{for each } x \in D \text{ (in the sense of subsequence).} \tag{3.19}$$

Since $(\lambda - gy)\chi_\epsilon(c - v_n) \leq \max_D(\lambda - gy) < +\infty$, from the Lebesgue dominated convergence theorem (see Appendix A),

$$\lim_{n \to \infty} \int_D (\lambda - gy)\chi_\epsilon(c - v_n)dxdy = \int_D (\lambda - gy)\chi_\epsilon(c - v)dxdy.$$

And from (3.17)

$$\liminf_{n\to\infty} J_\epsilon^*(v_n) \geq J_\epsilon^*(v). \tag{3.20}$$

Since $v \in K^*$ known from (3.16) and (3.19), and $J_\epsilon^*(v) = \alpha$ from (3.20), v is the solution of (3.12). □

Based on Lemma 3.2, the following theorem can be established.

Theorem 3.3 *Let ϵ_n be a decreasing sequence such that $\epsilon_n \to 0$. Let v_n be the solution of (3.12) corresponding to ϵ_n. Then there must exist a subsequence $\{v_{n_i}\} \subset \{v_n\}$ such that $v_{n_i} \to v$ in D and v is the solution of (3.11).*

Proof Since $J_{\epsilon_n}^*(v_n) \geq 0$, let $\alpha = \inf J_{\epsilon_n}^*(v_n)$ and there exists a subsequence $\{v_{n_i}\}$, and $J_{\epsilon_{n_i}}^*(v_{n_i}) \to \alpha \quad (i \to \infty)$. Therefore $\|\nabla v_{n_i}\|$ have a bound. For simplicity, we denote $\{v_{n_i}\}$ with $\{v_n\}$. By the same method used in the proof of Lemma 3.2, there exists $v \in K^*$

$$\begin{aligned} v_n &\xrightarrow{\text{w}} v \quad \text{in } H^1(D) \\ v_n &\longrightarrow v \quad \text{in } L^2(D) \\ v_n &\longrightarrow v \quad \text{for each } x \in D \text{ (in the sense of subsequence)} \end{aligned}$$

Here we prove that v is the solution of (3.11). i.e. $\forall u \in K^*$, $J^*(u) \geq J^*(v)$. Through the monotonicity of χ_ϵ,

$$\limsup_{n\to\infty} J_{\epsilon_n}^*(v_n) \geq \limsup_{n\to\infty} J_{\epsilon_{n_0}}^*(v_n) \geq J_{\epsilon_{n_0}}^*(v) \quad \forall n_0$$

Through the arbitrariness of n_0 and the Lebesgue dominated convergence theorem,

$$\lim_{n_0\to\infty} J_{\epsilon_{n_0}}^*(v) = J^*(v),$$

Therefore,

$$\limsup_{n\to\infty} J_{\epsilon_n}^*(v_n) \geq J^*(v)$$

Through minimum property of v_n,

$$J^*(u) = \lim_{n\to\infty} J_{\epsilon_n}^*(u) \geq \limsup_{n\to\infty}(v_n) \geq J^*(v)$$

So v is the solution of (3.11). □

From Theorem 3.3 the existence of the solution for (3.11) follows.

3.3.2 Equivalent Form

Theorem 3.4 *If u is the solution of (3.12), then u satisfies following equations*

$$-\Delta u - (\lambda - gy)\chi_\epsilon'(c-u) = 0 \quad in\ D \tag{3.21}$$

$$u|_\Gamma = G(x,y) \tag{3.22}$$

$$u|_{\Gamma'} \le c,\ \frac{\partial u}{\partial n} \ge 0,\ (c - u|_{\Gamma'})\left.\frac{\partial u}{\partial n}\right|_{\Gamma'} = 0 \tag{3.23}$$

Proof For each $v \in K^*$, because K^* is a convex set ($tv + (1-t)u \in K^*$, for each $t \ge 0$), $J_\epsilon^*(tv + (1-t)u)$ reaches a minimum when $t = 0$. Hence

$$\left.\frac{dJ_\epsilon^*(tv + (1-t)u)}{dt}\right|_{t=0} \ge 0.$$

i.e.

$$\begin{aligned} 0 \ &\le \ \int_D [\nabla u \nabla(v-u) - (\lambda - gy)\chi_\epsilon'(c-u)(v-u)]\, dx\, dy \\ &= \ \int_D [-\Delta u(v-u) - (\lambda - gy)\chi_\epsilon'(c-u)(v-u)]\, dx\, dy \\ &\quad + \int_{\Gamma'} \frac{\partial u}{\partial n}(v-u)\, dx\, dy \end{aligned} \tag{3.24}$$

Select v arbitrarily such that $v|_{\Gamma'} = u|_{\Gamma'}$, then

$$\int_D [\nabla u \nabla(v-u) - (\lambda - gy)\chi_\epsilon'(c-u)(v-u)]\, dxdy \ge 0$$

Due to the arbitrariness of v, (3.21) is obtained. And (3.24) is changed to

$$\int_{\Gamma'} \frac{\partial u}{\partial n}(v-u)ds \ge 0. \tag{3.25}$$

Let $(x,y) \in \Gamma'$.

(i) Suppose that $u(x,y) = c$.

After multipling both sides of (3.21) by $(c-u)^-$ and integrating,

$$\int_D \left[\nabla(c-u)^-\right]^2 dxdy - \int_D (c-u)^-(\lambda - gy)\chi_\epsilon'(c-u)dxdy = 0.$$

Because $\int_D (c-u)^-(\lambda - gy)\chi_\epsilon'(c-u)dxdy = 0$, $\nabla(c-u)^- = 0$. Then $(c-u)^-$ is constant. As $u|_\Gamma = G(x,y) \le c$, $(c-u)^-|_\Gamma = 0$. So

$$(c-u)^- = 0$$

i.e. $u \geq c$. Therefore

$$\left.\frac{\partial u(x,y)}{\partial n}\right|_{\Gamma'} \geq 0$$

(ii) Suppose that $u(x,y) < c$.

From (3.25) and the arbitrariness of v,

$$\left.\frac{\partial u(x,y)}{\partial n}\right|_{\Gamma'} = 0$$

So, on Γ', u satisfies

$$\left.\frac{\partial u}{\partial n}\right|_{\Gamma'} \geq 0$$
$$c - u|_{\Gamma'} \geq 0$$
$$\left.\frac{\partial u}{\partial n}\right|_{\Gamma'} (c - u|_{\Gamma'}) = 0.$$

In both cases we see that u is the solution of (3.21) - (3.23). □

Condition (3.23) is a two-possibility condition, introduce the penalty function

$$\beta(t) = \begin{cases} 0, & t \geq 0 \\ -\infty, & t < 0 \end{cases} \tag{3.26}$$

then (3.23) is substituted by

$$\left(\frac{\partial u}{\partial n} + \beta(c-u)\right)_{\Gamma'} = 0. \tag{3.27}$$

Lemma 3.5 *If u satisfies (3.27), then u also satisfies (3.23).*

Proof From (3.27) and the definition of $\beta(t)$, we know

$$\left.\frac{\partial u}{\partial n}\right|_{\Gamma'} = -\beta(c-u)|_{\Gamma'} \leq 0$$

And if there is a $(x_0, y_0) \in \Gamma'$ such that $u(x_0, y_0) > c$, hence $\beta(c - u(x_0, y_0)) = -\infty$. The equality (3.27) cannot be true, since $\left.\frac{\partial u(x_0,y_0)}{\partial n}\right|_{\Gamma'}$ is finite. Therefore

$$u|_{\Gamma'} \leq c.$$

For each $(x_0, y_0) \in \Gamma'$, if $u(x_0, y_0) = c$, then $\frac{\partial u(x_0,y_0)}{\partial n}(c - u(x_0, y_0)) = 0$ is obviously true. And if $u(x_0, y_0) < c$, then

$$\frac{\partial u(x_0, y_0)}{\partial n} = -\beta(c - u(x_0, y_0)) = 0.$$

It follows that $\frac{\partial u(x_0,y_0)}{\partial n}(c - u(x_0, y_0)) = 0$. This completes the proof. □

From Lemma 3.5, it is easy to obtain the following theorem.

Theorem 3.6 *If u is the solution of*

$$\begin{cases} -\Delta u - (\lambda - gy) - \chi_\epsilon'(c-u) = 0 \\ u|_\Gamma = G(x,y) \\ \left.\dfrac{\partial u}{\partial n}\right|_{\Gamma'} = -\beta(c - u|_{\Gamma'}) \end{cases} \tag{3.28}$$

then u must satisfy (3.21)–(3.23).

3.3.3 Numerical Form

The penalty function $\beta(t)$ in (3.26) is a jump function. We smooth it out with $\beta_\delta(t) \in C^\infty(R)$ (see Figure 3.3) satisfying

$$\lim_{\delta \to 0} \beta_\delta(t) = \beta(t).$$

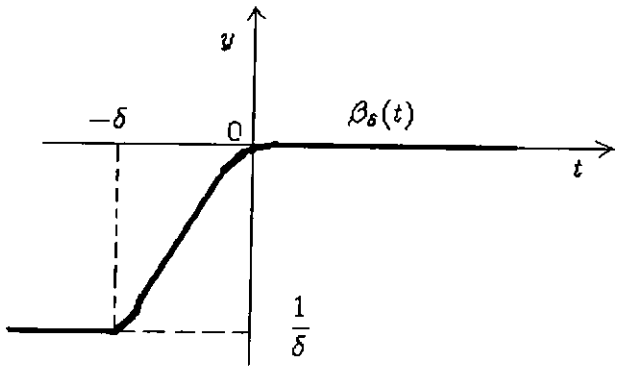


Figure 3.3

Replace $\beta(c - u|_{\Gamma'})$ by $\beta_\delta(c - u|_{\Gamma'})$ in (3.28), we get

$$-\Delta u - (\lambda - gy) - \chi_\epsilon'(c-u) = 0 \tag{3.29}$$

$$u|_\Gamma = G(x,y) \tag{3.30}$$

$$\left.\frac{\partial u}{\partial n}\right|_{\Gamma'} = -\beta_\delta(c - u|_{\Gamma'}) \tag{3.31}$$

Its variational form is:

Find the solution $u \in H_\Gamma^1(D)$ such that for all $v \in H_0^1(D)$, we have

$$(\nabla u, \nabla v) - (\lambda - gy)(\chi_\epsilon', v) + \int_{\Gamma'} \beta_\delta(c - u|_{\Gamma'}) \cdot v ds = 0 \tag{3.32}$$

where

$$H_\Gamma^1(D) = \{v(x) |\ v(x) \in H^1(D),\ v(x)|_\Gamma = G(x,y)\}$$

and

$$H_0^1(D) = \{v(x)|\ v(x) \in H^1(D),\ v(x)|_\Gamma = 0\}.$$

Let

$$V = \{v(x) \in C(D)|\ v(x) \text{ is a piecewise polynomial}\}$$
$$V_{0h} = \{v(x) \in V|\ v(x)|_\Gamma = 0\}$$
$$u_0(x,y) \in H^1(D),\quad u_0|_\Gamma = G(x,y)$$
$$u_{0h} = \Pi u_0 \text{ (interpolation of } u_0)$$
$$V_h = u_{0h} + V_{0h}$$

The numerical form corresponding to (3.32) is:
Find the solution $u_h \in V_h$ such that for all $v_h \in V_{0h}$, we have

$$(\nabla u_h, \nabla v_h) - (\lambda - gy)(\chi'_\epsilon(c - u_h), v_h) + \int_{\Gamma'} \beta_\delta(c - u_h) \cdot v_h ds = 0 \qquad (3.33)$$

To prove the existence of the solution u_h to (3.33), we need the following lemma:

Lemma 3.7 *For $w \in L^2(D)$, the variational form:*

$$(\nabla u_h, \nabla v_h) - (\lambda - gy)(\chi'_\epsilon(c - w), v) + \int_{\Gamma'} \beta_\delta(c - w) \cdot v ds = 0 \qquad (3.34)$$

for all $v_h \in V_{0h}$, has a unique solution.

Proof In (3.34) let $u_h = U_h + u_{0h}$, $U_h \in V_{0h}$, then

$$(\nabla U_h, \nabla v_h) = (\lambda - gy)(\chi'_\epsilon(c - w) - u_{0h}, v_h) - \int_{\Gamma'} \beta_\delta(c - w) v_h ds \qquad (3.35)$$

Let $a(u,v) = (\nabla u, \nabla v)$, $u, v \in V_{0h}$, be a bilinear form, then

$$a(U_h, U_h) = \|\nabla U_h\|^2 \le \|U_h\|_1$$

From Poincare-Inequality we can get

$$\alpha \|U_h\|_1 \le \|\nabla U_h\| = a(U_h, U_h)$$

where $\alpha > 0$.

Since V_{0h} is complete and $a(\cdot,\cdot)$ is symmetric, Lemma 3.7 is proved by using the Lax-Milgram lemma. □

By Lemma 3.7, we get the following theorem for the existence of the solution to (3.33):

Theorem 3.8 *The solution of the variational form (3.33) exists.*

Proof Let $u_h = Tw$, $w \in V_h$, $u_h \in V_h$, and

$$(\nabla u_h, \nabla v_h) - (\lambda - gy)(\chi'_\epsilon(c-w), v) + \int_{\Gamma'} \beta_\delta(c-w)v ds = 0, \quad \forall v_h \in V_{0h}.$$

By Lemma 3.7, T is a mapping of V_h into itself. In the following we will prove that the mapping T suits other conditions of the Schaulder theorem (refer to Appendix B). i.e.

(i) T is continuous

Let $w_1, w_2 \in V_h$, $u_1 = Tw_1, u_2 = Tw_2$, then $\forall v \in V_h$

$$(\nabla u_1, \nabla v) - (\lambda - gy)(\chi'_\epsilon(c-w_1), v) + \int_{\Gamma'} \beta_\delta(c-w_1)v ds = 0 \qquad (3.36)$$

$$(\nabla u_2, \nabla v) - (\lambda - gy)(\chi'_\epsilon(c-w_2), v) + \int_{\Gamma'} \beta_\delta(c-w_2)v ds = 0 \qquad (3.37)$$

We get

$$\begin{aligned}(\nabla(u_1-u_2), \nabla v) &= (\lambda - gy)(\chi'_\epsilon(c-w_1) - \chi'_\epsilon(c-w_2), v) \\ &\quad - \int_{\Gamma'} (\beta_\delta(c-w_1) - \beta_\delta(c-w_2))v ds \qquad (3.38)\end{aligned}$$

by (3.36)–(3.37). Substitute $u_1 - u_3$ for v in (3.38)

$$\begin{aligned}\int_D |\nabla(u_1-u_2)|^2 dx\, dy &\le C \int_D |\chi'_\epsilon(c-w_1) - \chi'_\epsilon(c-w_2)||u_1-u_2|\, dx\, dy \\ &\quad + \int_{\Gamma'} |\beta_\delta(c-w_1) - \beta_\delta(c-w_2)||u_1-u_2| ds.\end{aligned}$$

Using Poincare inequality,

$$\begin{aligned}\|u_1-u_2\|_1^2 &\le C\|\nabla(u_1-u_2)\|^2 \\ &\le \frac{C}{\epsilon^2}\|w_1-w_2\|\|u_1-u_2\| + \max_t |\beta'_\delta(t)|\|w_1-w_2\|_1\|u_1-u_2\|_1 \\ &\le C\|w_1-w_2\|\|u_1-u_2\|\end{aligned}$$

i.e. $\|u_1-u_2\| \le C\|w_1-w_2\|$, which shows the continuity of T.

(ii) TV_h is compact.

Since V_h is a finite dimensional space, TV_h is finite dimensional. It follows that TV_h is compact if we can prove that TV_h is bounded. Indeed by (3.35) in the proof of Lemma 3.7, we can get

$$\begin{aligned} \|U_h\|_1^2 &\leq C_1\|U_h\| + M\|U_h\|_{0,\Gamma'} \\ &\leq C_1\|U_h\|_1 + C_2M\|U_h\|_1. \end{aligned}$$

That is to say

$$\|U_h\|_1 \leq C_1 + C_2M.$$

where $M = \max|\beta_\delta(c-w)|$. Then $\|u_h\|_1 < C_1 + C_2M + \|u_{0h}\|_1$. Hence TV_h is bound.

From the Schaulder theorem (see Appendix A), we know that T has a fixed point u_h which is the solution of (3.33). □

Remark. We compute (3.33) with relatively small δ as the approximation of the stream function model (3.4)–(3.9) on the platform of FEPG (Finite Element Generator Program, see Appendix B).

Chapter 4

Comparison of the Two Models

In this chapter, we will tersely compare the two algorithms of the potential function model and the stream function model mentioned in Chapter 2 and Chapter 3 respectively. First we explain the advantage and drawback of these two algorithms. Secondarily, from the previous two chapters, we know that the potential function model and the stream function are equivalent. So the numerical results must be the same in theoretical sense. A list of speed of the points on the free boundary is used to demonstrate the equivalence of these two models.

The algorithm of the potential function model is an iterating process based on the flux and the free boundary, with which one can achieve more accurate free boundary shape and flux. Moreover this algorithm is verified many times by physical experiments. It is a pretty mature algorithm among fluid computation. Its drawback is that there is no exact theoretical proof and we can't prove the convergence of the iteration. Meanwhile, it needs much computing work and consumes much CPU time.

The algorithm of the stream function model is computing a variational problem obtained through variational method for variational domain by the finite element method. In Chapter 3, we give all theoretical bases needed by this algorithm. So there is no doubt in theory. Since there is no iteration in computing, it just needs less computing work and consumes less CPU time. Its drawback is that we can only get the narrow lace in which the free boundary may locate rather than the definite free boundary obtained from the algorithm of the potential function model.

Before giving the list in the following, we need to do some explain. In Chapter 2 and Chapter 3 we know the relation between the potential function $\phi(x,y)$ and the stream function $\psi(x,y)$ is:

$$\frac{\partial\psi(x,y)}{\partial x} = \frac{\partial\phi(x,y)}{\partial y}, \quad \frac{\partial\psi(x,y)}{\partial y} = -\frac{\partial\phi(x,y)}{\partial x}$$

we denote the flow velocity in x-direction by S_x and in y-direction by S_y. We use the subscript p and q to distinguish the velocity obtained from the potential function model and the stream function model respectively. Then

$$\begin{aligned} S_{x_p} &= \phi_x, \\ S_{y_p} &= \phi_y, \\ S_{x_q} &= \psi_y, \\ S_{y_q} &= -\psi_x. \end{aligned}$$

Let

$$\begin{aligned} e_x &= |S_{x_p} - S_{x_q}|, \\ e_y &= |S_{y_p} - S_{y_q}|. \end{aligned}$$

In theory, if the parameters used in the computation by the potential function model and the stream function model are the same, then $e_x = e_y = 0$. However, practically there are some deviations in the actual computation. The deviations e_x and e_y are shown in the following table.

x	y	S_{x_p}	S_{x_q}	S_{y_p}	S_{y_q}	e_x	e_y
-80.00	85.00	3.402004	3.227526	0.293793	0.0000	0.17448	0.29379
-68.31	84.93	3.591063	4.433717	-0.01815	0.0000	0.84265	0.01815
-53.70	84.77	3.958116	4.660282	-0.02415	0.000	0.70217	0.02415
-39.08	84.53	4.494598	5.038917	-0.08686	0.000	0.54432	0.08686
-24.45	83.97	5.484186	5.273575	-0.27868	-0.271039	0.21061	0.00764
-9.06	82.15	6.787702	6.33834	-1.52473	-0.478973	0.44936	1.04576
5.44	78.05	11.26501	11.0309	-4.67419	-3.74226	0.2341	0.9319
16.73	71.49	14.66012	15.45273	-10.9882	-8.312456	0.79260	2.67576
29.21	58.94	15.40882	17.01303	-18.7186	-14.20831	1.60421	4.51030
40.60	41.95	14.92616	17.04092	-25.0021	-19.5310	2.2148	5.47108
52.58	21.75	18.47039	21.90899	-29.8465	-23.80349	3.43860	6.05303

In the above table, the values S_{x_p}, S_{x_q}, S_{y_p} and S_{y_q} are computed using the following parameters. Referring to Figure 2.1 the parameters being using are:

$$\begin{aligned} h &= 85.0, \\ h1 &= 80.0, \\ H &= 62.0, \end{aligned}$$

and $s3$ is depicted by the following discrete points:

Number	x-coordinate	y-coordinate
1	0.0000000	62.0000000
2	2.0000000	61.7340500
3	4.0000000	61.4680900
4	6.0000000	60.7752900
5	8.0000000	60.0824800
6	10.0000000	59.0113200
7	12.0000000	57.9401600
8	14.0000000	56.5137500
9	16.0000000	55.0873400
10	18.0000000	53.3209300
11	20.0000000	51.5545100
12	22.0000000	49.4594000
13	24.0000000	47.3642800
14	26.0000000	44.9493900
15	28.0000000	42.5345000
16	30.0000000	39.8071700
17	32.0000000	37.0798500
18	34.0000000	34.0463000
19	36.0000000	31.0127500
20	38.0000000	28.0643200
21	40.0000000	25.1158800
22	42.0000000	22.2587400
23	44.0000000	19.4015900
24	46.0000000	16.5444500
25	48.0000000	13.6873100
26	50.0000000	10.8301600
27	52.0000000	7.9730210
28	54.0000000	5.1158780
29	56.0000000	2.2587350
30	58.0000000	-5.984082E-001
31	60.0000000	-3.4555510

Chapter 5

Conclusions

The Dam Overflow Problem is an old problem involved in physics, mechanics, applied mathematics and engineering science. Mathematics has contributed much in this area. Especially following the modern electronic development which makes computers faster and cheaper, large scale computation in low cost now becomes viable. The advantage of using mathematical models to analyze engineering problems is that its cost is low, but the results sometimes are very satisfactory. So using mathematical model is becoming more common in engineering. The Dam Overflow Problem is not exceptional. Further application of mathematics in the Dam Overflow Problem will surely be very promising if more working principles are discovered and more major factors are considered.

The finite element method developed in the seventies is so far one of the most powerful methods to obtain numerical solutions of PDE. With its theory completing increasingly and plenty of practical experience of application accumulated, many large scale computations including fluid computation, which used the traditional finite difference method before, has switched to the finite element method. The finite element method has the advantage that it can be effective over domains of any shape, regular or irregular. Whereas, the finite difference method is difficult to be applied over a relatively irregular domain. Although using the finite element method consumes a little more CPU time than using the finite difference method, it does not matter much since following the swift development of the computer hardware, the speed of the CPU is no longer a bottleneck in many aspects. Based

on the above reasons, the finite element method is an option for either numerical solutions or theoretical results in this thesis.

The main contributions of the present research to the dam overflow problem are described in details in Chapter 2 and Chapter 3. In Chapter 2, the focus is on improving the original algorithm and developing programs to obtain the numerical results of the potential function model. Section 2.1 and section 2.2 consist of the preliminary work. The author deduced the mathematical model in section 2.1 and depicted in details the original algorithm in section 2.2. Section 2.3 and section 2.4 consist of the author's major work. In section 2.3 the author introduced the relaxation factor method and illustrated its advantage and implementation in modifying the original algorithm. Moreover, to demonstrate the advantage of using relaxation factors, the author compared the numerical results of computation based on the modified algorithm with relaxation factors adding in and the original algorithm without the relaxation factor. In the last section, the author showed the numerical results by graphics.

In Chapter 3, the focus is on the theoretical results of the stream function model. Considering the drawback of the potential function model for developing theoretical results due to the presence of tangential derivatives in the boundary conditions, the stream function, which is the conjugate function of the potential function, was used in this chapter. As a byproduct of the theory, the algorithm corresponding to the stream function model was obtained. In section 3.1 the mathematical model based on the stream function was developed. In section 3.2 the variational form of the stream function model was derived. The author's major works for the stream function model included in section 3.3 are listed as follows:

1. The author established the existence of the solution to the variational form of the stream function model.

2. The author established the existence of the discrete solution to the corresponding numerical scheme.

3. The author obtained some numerical results for the stream function model.

The deficiency is that the author cannot prove the solution uniqueness for the stream function model. If the solution uniqueness can be proved, more interesting results such as the error estimation between the discrete form and the original form can be established.

The contents of the Dam Overflow Problem are by far richer than what I have mentioned in this thesis. More work has to be done in order to understand it thoroughly.

Bibliography

[1] Bossavit, A., Damlamian, A., Fremond, M.
"Free Boundary Problems: Applications and Theory"
Vol. III, IV, Pitman, London (1985)

[2] 陈传淼
"有限元方法及其提高精度的分析"
湖南科技出版社 (1982)

[3] 陈亚浙，吴兰成
"而阶椭圆方程与椭圆型方程组"
科学出版社 (1979)

[4] 丁道扬
"应用自由表面位移特性解水工地泄漏问题"
中国科学 1986, 5: 321-356

[5] 丁道扬
"计算三元自由表面孔口水流 "
力学学报， 1982， 6 ： 519-527

[6] 丁道扬
"具有未知流量的二元孔口水流数学模型 "
水利水运科学研究， 1981， 2 ： 32-44

[7] 许协定
"自由表面重力流的一种有限元解法 "
水利学报， 1980, 1: 1-13

[8] Chan, S. T. K., Larock, B. E., Herrman, L. R.
"Free-surface Ideal Flows",
A. S. C. E., 99 HY6 (1973)

[9] Eliott, C. M., Ockendon, J. R.
"Weak and Variational Methods for Moving Boundary Problems"
Pitman, London (1982)

[10] Friedman A.
"Variational Principles and Free-boundary Problems"
John Wiley & Sons (1982)

[11] 关肇直，张恭庆，冯德兴
"线性泛函分析入门"
上海科技出版社 (1979)

[12] Hestenes M. R., Stiefel, E.
"Methods of conjugate gradients for solving linear systems"
J. Res. Nat. Bur. Standards Sect., 1952, B49: 409-436

[13] Alt H. W., Caffarelli L. A.
"Existence and regularity for a minimum problem with free boundary" J.Reine Angew Math., 1981, 325: 105-144

[14] 姜礼尚
"自由边界问题"，现代数学与力学
北京大学出版社 (1986)

[15] 姜礼尚，庞之垣
"有限元方法及其理论基础"
人民教育出版社 (1979)

[16] Barrett, J. W., Shanahan, R. M.
"Finite element approximation of a model reaction-diffusion problem with a non-Lipschtiz nonlinearity"
Numer. Math., 1991, 59: 217-242

[17] Kinderlehrer, D., Stampacchia, G.
"An Introduction to Variational Inequalities and Their Applications" Academic Press, New York (1980)

[18] 梁国平
"有限元程序自动生成系统和有限元语言"
力学进展， 1991, 20: 20-52

[19] Liang G. P.
"An automated generation system for finite element programs "
Proceedings of Symposia on Scientific Software
China University of Science and Technology Press - Hefei, 1989, 159-165

[20] 梁国平
"有限元程序自动生成系统使用说明"
中国科学院数学研究所资料 (1989)

[21] 吕涛，石济民，林振宝
"区域分解算法"
科学出版社 (1992)

[22] Ikegawa M., Washizu K.
"Finite element method applied to analysis of flow over a spillway crest"
International Journal For Numerical Methods In Engineering, 1973, 6: 179-189

[23] Ciarlet P. G.
"Finite Element Methods for Elliptic Problems"
North-Holland Publ., Amsterdam (1978)

[24] Adams, R.
"Soboev Spaces"
Academic Press New York San Francisco London (1975)

[25] Mazja V. G.
"Sobolev Spaces"
Spinger-Verlag Berlin Heidelberg New York Tokyo (1985)

[26] Fortin, M. and Glowinski, R.
"Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-value Problems"
Amsterdam North-Holland (1986)

[27] Smart, D. R.
"Fixed point theorems"
Cambridge University Press (1974)

[28] Ciarlet, P. G. and Lions, J. L.
"Handbook of Numerical Analysis"
Acta Numerica, 1989, 61-143

[29] 应隆安
"有限元方法讲义"
北京大学出版社 (1988)

[30] Heuser, H. G.
"Functional Analysis"
John Wiley & Sons (1982)

[31] Szabo, B. A.
"Finite Element Analysis"
New York: Wiley (1991)

[32] Mitchell, A. R.
"The Finite Difference Method in Partial Differential Equations"
Chichester : Wiley (1980)

Appendix A

Some Theorems used in the Thesis

A.1 Bernoulli-Energy Equation

Since the Bernoulli-Energy Equation plays an important role in the potential function model and the stream function model. Hence we give a brief derivation of the Bernoulli-Energy Equation for a ease of reference.

The Euler's equation of incompressible flow along the streamline under streamline coordinate system is

$$-\frac{\partial p}{\rho \partial s} - g\frac{\partial z}{\partial s} - \vec{V}\frac{\partial \vec{V}}{\partial s} = 0 \tag{A.1}$$

where p is the air pressure, z is vertical coordinate, $\vec{V}$ is velocity field, g is the gravity, and ρ is the density.

Since g and ρ are constants, (1) can be written as

$$\frac{d}{ds}\Big(\frac{p}{\rho} + gz + \frac{\|\vec{V}\|^2}{2}\Big) = 0 \tag{A.2}$$

Therefore

$$\frac{p}{\rho} + gz + \frac{\|\vec{V}\|^2}{2} = \text{constant} \tag{A.3}$$

along the streamline.

After replacing z and p with y and P_0 respectively, the Bernoulli-Energy Equation is obtained.

A.2 Schauder's Second Theorem

The following theorem is known as Schauder's Second Theorem.

Theorem Let μ be a non-empty convex subset of a normed space B. Let T be a continuous mapping of μ into a compact set $\kappa \subset \mu$. Then T has a fixed point.

Schauder (1930) proved this result in the case where B is complete and μ closed. The following proof for the above theorem is from [27].

Definition Let T map a set ϕ into a topological space X. If $T\phi$ is contained in compact subset of X, we say that T is compact.

Notation We write $\text{co}(X)$ for the smallest convex set containing X, and $\overline{\text{co}}(X)$ for the closure of $\text{co}(X)$.

Lemma A (*Schauder's Projections*) If κ is a compact subset of normed space V and $\epsilon > 0$, there is a finite subset x of κ and a continuous mapping P of κ into $\text{co}(X)$ such that

$$\|Px - x\| < \epsilon \quad (x \in \kappa)$$

Proof Choose $x_1, \cdots, x_n$ in κ such that the sets $N(x_i, \epsilon)$ with $1 \leq i \leq n$ cover κ. Put $X = \{x_1, \cdots, x_n\}$. For $1 \leq i \leq n$ put

$$f_i(x) = \max(0, \epsilon - \|x - x_i\|).$$

Then $f_i(x) \neq 0$ if and only if $x \in N(x_i, \epsilon)$. Thus at each x in κ, some $f_i(x) \neq 0$. Now put

$$Px = \sum f_i(x)x_i / \sum f_i(x) \quad (x \in \kappa)$$

Clearly P is continuous. Also, since Px is a convex combination of those points x_i which lie in $N(x_i, \epsilon)$, we have $Px \in N(x, \epsilon)$. □

Proof of Schauder's Second Theorem. For $n = 1, 3, \cdots$ consider P_nT where P_n is the mapping given as in the previous Lemma A with $\epsilon = 1/n$. Since $X \subset \kappa \subset \mu$ we have $\text{co}(X) \subset \mu$. Thus P_nT gives a continuous mapping of the finite-dimensional compact convex set $\text{co}(X)$ into itself. A fixed point x_n exists by Brouwer's theorem (see A.7). From $P_nTx_n = x_n$ we get $\|Tx_n - x_n\| < 1/n$. By contraction mapping theorem, T has a fixed point. □

The following special case of Schauder's Second Theorem is useful for applications.

Corollary Let T be a compact continuous mapping of a normed space B into B, Then T has a fixed point.

For finite-dimensional spaces the above results become:

Theorem (a) Any continuous mapping of a convex subset μ of R^n into a bounded closed set inside μ has a fixed point.

(b) Any continuous mapping of R^n into a bounded subset of R^n has a fixed point.

A.3 Poincare's Inequality

Let Ω be a bounded domain. Suppose Γ is of class C^1 and such that $H_0^{1,p}(\Omega \cup \Gamma)$, where $1 \leq q < \infty$, does not contain any nonzero constant. Then

$$|u|_{p,\Omega} \leq C(\Omega)|\nabla u|_{p,\Omega}$$

holds whenever $u \in H_0^{1,p}(\Omega \cup \Gamma)$.

A.4 Lebesgue Dominated Convergence Theorem

Let A be a bounded Measurable subset of $\Re$ and let f_n be a sequence of measurable functions on A such that $\lim_{n\to\infty} f_n(x) = f(x)$ for every $x \in A$. If there exists a function $g \in \mathcal{L}(A)$ [i.e. g is summable] such that $|f_n(x)| \leq g(x)$ for $n = 1, 2, 3, \ldots$ and all $x \in A$, then $\lim_{n\to\infty} \int_A f_n dm = \int_A f dm$.

A.5 Hahn-Banach Theorem

Let E be a real linear space and let M be a linear subspace of E. Suppose p is a sublinear functional defined on E and f a linear functional defined on M such that $f(x) \leq p(x)$ for every $x \in M$. Then there is a linear functional g defined on E such that g is an extension of f (i.e., $g(x) = f(x)$ for all $x \in M$) and $g(x) \leq p(x)$ for all $x \in E$.

A.6 Lax-Milgram Theorem

Let B be a bounded, coercive bilinear form on a Hilbert Space E. Then for every bounded linear functional $F \in E^*$, there exists a unique element $f \in E$ such that

$$B(x, f) = F(x) \quad \forall x \in E.$$

A.7 Brouwer's theorem

Let B^n be the closed n-ball.

(i) B^n has the fixed point property.

(ii) Every compact convex non-empty subset X of R^n has the fixed point property.

Appendix B

The FEPG

This appendix is devoted to introducing the software FEPG (Finite Element Program Generator). Nearly all who have experience of developing FE (Finite Element)programs said FE programs are very complicated and involved. Albeit the existed FE packages can solve various problem depend on corresponding libraries. Nevertheless, all these packages are not user-friendly and difficult to master. Different from the conventional FE packages, the FEPG, which generates Fortran FE programs by computer directly, break through the limits with which most finite element program systems are restricted to specific finite element problems or specific fields.

In the following, the FEPG and its ability are first introduced. Next, the structure of FEPG is explain briefly. In the third section, an example of using FEPG in Chapter 2 is illustrated. For details of the FEPG, please see [18]-[20]. And in the last part, we give the details of how to compute the potential function model through FEPG.

B.1 Introduction of FEPG

During 1950s and 1960s, FEM in its early stage achieved great success in structural mechanics. During the following thirty years, FEM has developed into a mature computational technique, and engineers and scientists in various fields have gradually understood and accepted FEM. With the rapid advances and popularization of computer science, the cost of computation is greatly reduced. However, by and

large FEM is still used by only a small group of specialists. The reason comes from the complexity of FE programs.

The generality of FEM brings about the extensive applications of FEM, as well as the difficulties in FE programming. Any fairly consummate FE programs must contain well above 10,000 lines of codes, which must cost several manyears. This is unbearable to most people. To overcome the difficulties, many research faculties and software companies offer various FE packages. But most of them are limited to some specific fields. In order to solve problems as extensively as possible, some FE package have to resort to huge program libraries. Despite of this strategy, they still can't cover all problems. For a new problem, the subroutines are usually unavailable in the existing program libraries. The user must rewrite part of the codes to achieve his aims. Without the help of the developers of the FE packages, this task is unlikely to succeed. Moreover, such libraries are voluminous, poor in readability and inconvenient in maintenance and modification. It often takes much time for an engineer to customize and master such software. That is why people prefer the old, even obsolete, but familiar FE packages to new and advanced ones.

On the other hand, theoretically FE programs are canonical. The procedures in all FE programs can be divided into the following steps: first, preprocessing, including grid generation and data preparation; second, compute element stiffness matrices and loads with treating constraints; next, form the global stiffness matrix and right-hand-side terms, and then solve this algebraic system; at last, postprocessing, such as outputting results and visualizing them. Consequently, some FE program codes keep invariant when the problem vary. This examination brings the hope of designing a general FE software.

The FEPG is the very software that is applicable to extremely wide range of FE problems. It has been successfully applied to many fields, such as multi-body mechanics, super-conductivity, multi-phase and the phase control, fluid computation, shape memory alloys and earth dynamic mechanics. The basic idea of designing the FEPG is to separator FE programs into fixed parts and variable parts. The First parts keep unchanged for different problems and are written in advance. The

computer generates the variable parts according to the specific problems. Then, again by computer, a complete FE program is ready after inserting the variable parts into right places of the fixed parts. This design frees FE programs from being constrained to specific kinds of problems and fields.

The FEPG can handle problems from on-dimensional to three dimensional, no matter they are linear or nonlinear, ready-state or time-dependent, parabolic or hyperbolic, a single equation or system of equations. Whenever the order of the PDE (Partial Differential Equation) is not more than four and the weak-formulation is available, the FE programs can be generated quickly. The user can also choose arbitrary finite elements. Even different elements in different subdomains.

The key to make a computer generate the variable parts is how to explain an FE program to a computer. The FEPG in effect provides a kind of FE language, which is very close to the description manner of FE problems. The user needs only to fill in some files describing various expressions and formulae that FEM requires. No more work is necessary, the computer can then produce the desired codes of the variable parts. The work of filling the files, which are very comprehensible to ordinary scientist, is much less that writing a whole FE program. Thus not only the user can save much time, but also the files are of excellent readability, easy to be maintained and modified. For simple problems, such as the example at the end of this chapter, a skillful FEPG user can get the FE program in less than one hour. Even for extremely difficult problems, such as those afore-mentioned, it is estimated that the efficiency of programming can be improved by at least twenty times.

In some sense, the FEPG is a software platform. Users can apply it to develop their own FE software.

B.2 Structures of the FEPG and Files Needed

The FEPG consists of six subsystems: the element subroutine generator (ESG), the data generator (DG), the dynamic memory allocation generator (DMAG), the batch file generator (BFG), the algorithm generator (AG) and the atomic programs

and solvers (APS).

The ESG generates the elements subroutines that compute the element stiffness matrices, the element mass matrices and element load matrices. The element computation is one of the variable parts in FE programs. The user need only input the transforms between the original coordinates and the reference coordinate, the shape functions in the reference coordinate, the weak-formulation of the problem and so on. The files supplying these information are of extended name *.GES.

The DG prepares the mesh, initial values and Dirichlet boundary values of the functions to be solved. The programs fulfilling this task are surely variable parts. The file giving the relations of the required date is a tree-structured and of extended name *.MTI. The system needs another file to instruct the computer how to prepare these data wanted. The file name of this file is the same as the *.MTI file, but with different and arbitrary extended name.

The DMAG helps the user to realize dynamic memory allocation in Fortran programs. At first, the user may write programs using static arrays, without being troubled by their dynamic allocation. Next, the user modify the program by replacing the static array with the dynamic arrays. These files are of extended name *.EPG. Then the user write the memory allocation information files. which are of extended name *.GPG.

The BFG usually does not require any further input form the user. It can generate several batch files that create the Fortran programs, assemble them, compile them, execute the .EXE programs and show the final results graphically.

The AG produces the source code of the solving algorithm that the user designed. The extended name of the input is *.NFE. for instance, if the user wants to apply Grank-Nichoson scheme to solve a parabolic PDE, he may tell the compute the scheme in the *.NFE file.

The APS provides some program hate quite common in FE programs, such as the solvers SOL.FOR, the preprocessor START.FOR and boundary data modifier BFT.FOR which is used in evolution equations. Indeed, for simple problems the user usually need only to modify the example files attached to FEPG and quickly

obtain the files indeed by the FEPG.

B.3 An Example

For instance, consider the elliptic equation

$$\begin{cases} \Delta u = 4 \quad \text{in} \quad \Omega \\ u|_{\partial\Omega} = x^2 + y^2. \end{cases}$$

The solution is $u(x, y) = x^2 + y^2$. The following files can describe this problem sufficiently under FEPG.

1. **.GES File** defi

   disp u

   var u1,u2,u3,u4

   refc p,q

   coor x,y

   dord 1

   node 4

   shap

   u=[(1-p)*(1-q)/4]u1+[(1+p)*(1-q)/4]u2

   +[(1+p)*(1+q)/4]u3+[(1-p)*(1+q)/4]u4

   tran

   x=[(1-p)*(1-q)/4]x(1)+[(1+p)*(1-q)/4]x(2)

   +[(1+p)*(1+q)/4]x(3)+[(1-p)*(1+q)/4]x(4)

   y=[(1-p)*(1-q)/4]y(1)+[(1+p)*(1-q)/4]y(2)

   +[(1+p)*(1+q)/4]y(3)+[(1-p)*(1+q)/4]y(4)

   gaus=4 -1.,-1.,1.; 1.,-1.,1.; 1.,1.,1.,; -1.,1.,1.;

   stif

   dist=[u/x;u/x]+[u/y;u/y]

load=-[u]*4.

end

2. **.MTI File**

bili 5000

COOR;ELEM;ID;DISP

ELEM F

ENOD

table

ENOD 4000

N;NOD1;NOD2;NOD3;NOD4;

I5;I5;I5;I5;I5;

ID F

N;IDU;

I5;I3;

DISP F

N;U

I5;F10.5

COOR F

N;X;Y;

I5;F10.5;F10.5;

DECLAR

COMMON/NXY/NX,NY,NR,nc

3. **DATA File**

BILI 4

COOR

$r &NX;&NY;&XMAX;&YMAX;

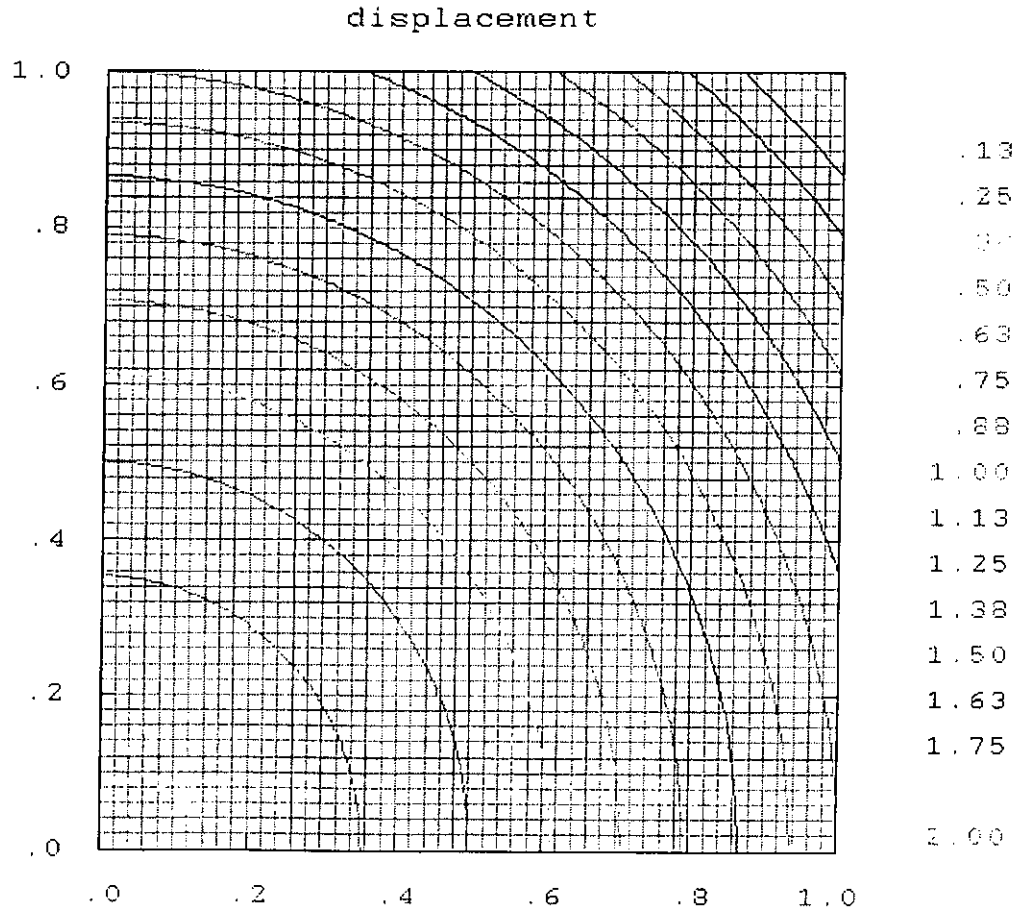
```
&DX=XMAX/NX; &DY=YMAX/NY;
[[(I-1)*(NX+1)+J;DX*(J-1);DY*(I-1);J=1,NX+1];I=1,NY+1]
ID
[I;1;I=1,(NX+1)*(NY+1)]
[I;-1;I=1,NX+1]
[I;-1;I=(NX+1)*(NY+1)-NX,(NX+1)*(NY+1)]
[(NX+1)*(I-1)+1;-1;I=1,NY+1]
[(NX+1)*I;-1;I=1,NY+1]
ELEM 1
ENOD
[[NX*(I-1)+J;(I-1)*(NX+1)+J;(I-1)*(NX+1)+J+1;I*(NX+1)+J+1;
I*(NX+1)+J; J=1,NX];I=1,NY]
DISP
$c6 DIMENSION R(2)
$c6 COMMON /coor/X(5000),Y(5000)
[I;0.0;I=1,(NX+1)*(NY+1)]
[I;&R(1)=X(I);&R(2)=Y(I);BOUND(R,0.0,1);I=1,NX+1]
[I;&R(1)=X(I);&R(2)=Y(I);BOUND(R,0.0,1);I=(NX+1)*(NY+1)-NX,
(NX+1)*(NY+1)]
[N=(NX+1)*(I-1)+1;&R(1)=X(N);&R(2)=Y(N);
U=BOUND(R,0.0,1);I=1,NY+1]
[N=(NX+1)*I;&R(1)=X(N);&R(2)=Y(N);U=BOUND(R,0.0,1);I=1,NY+1]
```

4. **.NFE File**

```
defi
stif S
mass M
load F
type e
```

```
mdty 1
step 0
equation
matrix = [S]
FORC=[F]
SOLUTION U
write(s,unod) U
end
```

As $\Omega = [0,1] \times [0,1]$, the graph of solution is below



B.4 Implementation

In this section, we list the major files needed in computation of potential function model in Chapter 2, with which we can develop the programs through FEPG to get numerical results in Chapter 2.

1. Element Information File

   .GES File: BILI.GES

   defi
   disp u
   var u1, u2, u3, u4, u5, u6 ,u7, u8
   refc p, q
   coor x, y
   dord 1
   node 8

   shape
   u=[-(1-p)*(1+q)*(1-q+p)/4]u1 + [(1-p**2)*(1+q)/2]u2
   + [-(1+p)*(1+q)*(1-p-q)/4]u3 + [(1-q**2)*(1+p)/2]u4
   + [-(1+p)*(1-q)*(1-p+q)/4]u5 + [(1-p**2)*(1-q)/2]u6
   + [-(1-p)*(1-q)*(1+p+q)/4]u7 + [(1-q**2)*(1-p)/2]u8
   tran
   x=[-(1-p)*(1+q)*(1-q+p)/4]x1 + [(1-p**2)*(1+q)/2]x2
   + [-(1+p)*(1+q)*(1-p-q)/4]x3 + [(1-q**2)*(1+p)/2]x4
   + [-(1+p)*(1-q)*(1-p+q)/4]x5 + [(1-p**2)*(1-q)/2]x6
   + [-(1-p)*(1-q)*(1+p+q)/4]x7 + [(1-q**2)*(1-p)/2]x8

   y=[-(1-p)*(1+q)*(1-q+p)/4]y1 + [(1-p**2)*(1+q)/2]y2
   + [-(1+p)*(1+q)*(1-p-q)/4]y3 + [(1-q**2)*(1+p)/2]y4
   + [-(1+p)*(1-q)*(1-p+q)/4]y5 + [(1-p**2)*(1-q)/2]y6

```
+ [-(1-p)*(1-q)*(1+p+q)/4]y7 + [(1-q**2)*(1-p)/2]y8
gaus=8 -1.0,-1.0,0.5;0.0,-1.0,0.5;1.0,-1.0,0.5;1.0,0.0,0.5;
      1.0,1.0,0.5;0.0,1.0,0.5;-1.0,1.0,0.5;-1.0,0.0,0.5
stif
dist=[u/x;u/x]+[u/y;u/y]
load=[u]*0.0
end
```

LINE.GES (Neumann Boundary Condition)

```
LINE
defi
disp u
var u(3)
refc s1
coor y
dord 1
node 3
shap
u=[(s1-1)*(s1-0.5)*2]u1+[-4*s1*(s1-1)]u2+[2*s1*(s1-0.5)]u3
tran
y=[(s1-1)*(s1-0.5)*2]y(1)+[-4*s1*(s1-1)]y(2)+[2*s1*(s1-0.5)]y(3)
gaus=3 0.0.0.333333;0.5,0.333333;1.0,0.333333;
stif
$c6 v0=prmt(1)
dist=[u;u]*0.0

load=[u]*v0
end
```

2. Tree Structural File

.MTI File: BILI.MTI

bili 5000

COOR;ELEM;NODUV;ID;DISP;

ELEM F

NOD8,NOD3

table

NOD8

N;NOD1;NOD2;NOD3;NOD4;NOD5;NOD6;NOD7;NOD8;

I5;I5;I5;I5;I5;I5;I5;I5;I5;

NOD3

N;NOD1;NOD2;NOD3;

I5;I5;I5;I5;

NODUV F

N;NOD1;NOD2;NOD3;NOD4;

I5;I5;I5;I5;I5;

ID F

N;IDU;

I5;I3;

DISP F

N;U

I5;F10.5

COOR F

N;X;Y;

I5;F10.5;F10.5;

DECLAR

COMMON/NXY/NX,NY,NR,nc

3. Data File

BILI. File: bili.rec

```
BILI 5
COOR
$C6 DIMENSION A1(100),A2(100),B1(100),B2(100)
$C6 DIMENSION C1(100),C2(100),D1(100),D2(100)
$c6 OPEN(1,FILE='FREE',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) N1,M1
$C6 K=(M1-1)/2
$C6 DO 1000 I=1,N1
$C0 1000 READ(1,*) A1(I),A2(I)
$C6 CLOSE(1)
$C6 OPEN(1,FILE='SIDE',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) N2
$C6 DO 1200 I=1,N2/2
$C6 READ(1,*) C1(I),C2(I)
$C0 1200 CONTINUE
$C6 CLOSE(1)
$C6 OPEN(1,FILE='start',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) NNN1,NNN2,NNN3,NNN3
$C6 READ(1,*) HHH0,XXX1,YYY1,XXX2
$c6 along=xxx2*0.22
$c6 xxx2=xxx1+xxx2
$c6 close(1)
$c6 open(1,file='start',form='formatted',status='old')
$c6 read(1,*) nnn1,nnn2,nnn3,nnn4
$c6 close(1)
$c6 nnn1=nnn1-1
$c6 nnn2=nnn2+1
$C6 CLOSE(1)
```

```
$C6 DO 1210 I=N2/2+1,N2
$C6 C1(I)=A1(M1)+(XXX1-A1(M1))/(N2/2+1)*(I-N2/2)
$C6 C2(I)=A2(M1)+(YYY1-A2(M1))/(N2/2+1)*(I-N2/2)
$C0 1210 CONTINUE
$C6 OPEN(1,FILE='SIDE',FORM='FORMATTED',STATUS='OLD')
$C6 WRITE(1,*) N2
$C6 DO 1220 I=1,N2
$C6 WRITE(1,*) C1(I),C2(I)
$C0 1220 CONTINUE
$C6 CLOSE(1)
$C6 OPEN(1,FILE='SIDE',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) N2
$C6 DO 1001 I=1,N2
$C0 1001 READ(1,*) C1(I),C2(I)
$C6 CLOSE(1)
$C6 N3=N2/2
$C6 DO 1002 I=1,N3
$C6 D1(I)=C1(I+N3)
$C6 D2(I)=C2(I+N3)
$C0 1002 CONTINUE
$C6 OPEN(1,FILE='DOWN',FORM='FORMATTED',STATUS='OLD')
$C6 DO 1003 I=1,N1
$C0 1003 READ(1,*) B1(I),B2(I)
$C6 CLOSE(1)
$C6 N6=N1+(N1-1)/2+1
$C6 N5=(N3+1)*N6
[I;A1(I);A2(I);I=1,N1]
[I;B1(I-N5);B2(I-N5);I=N5+1,N5+N1 ]
[ [I*N6+2*J+1;& M=2*J+1;
&U=SL1(A1(M),A2(M),B1(M),B2(M),C1(I),C2(I),D1(I),D2(I));
```

```
SL1(A1(M),A2(M),B1(M),B2(M),C1(I),C2(I),D1(I),D2(I));
SL2(C1(I),C2(I),D1(I),D2(I),U);J=0,K];I=1,N3]
[[I*N6+2*J;&M=I*N6+2*J;(X(M+1)+X(M-1))/2;
(M+1)+Y(M-1))/2;J=1,K];I=1,N3]
[[N1+(I-1)*N6+J+1;&M=(I-1)*N6+2*J+1;(X(M)+X(M+N6))/2;
(M)+Y(M+N6))/2;J=0,K];
I=1,N3+1]
[[I*N6+2*(J+K)+1;&M=2*(J+K)+1;A1(M)+I*(B1(M)-A1(M))/(n3+1);
A2(M)+I*(B2(M)-A2(M))/(n3+1);J=0,(n1-1)/2-k];I=1,N3]
[[I*N6+2*(K+J);&M=I*N6+2*(K+J);(X(M-1)+X(M+1))/2;
(Y(M+1)+Y(M-1))/2;J=0,(N1-1)/2-K];I=1,N3]
[[N1+(I-1)*N6+(J+K)+1;&M=(I-1)*N6+2*(J+K)+1;
(X(M)+X(M+n6))/2; (Y(M)+Y(M+N6))/2;J=0,(n1-1)/2-K];
I=1,N3+1]
&N=N5+N1;

ID
$c6 OPEN(1,FILE='FREE',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) N1,M1
$C6 CLOSE(1)
$C6 K=(M1-1)/2
$C6 OPEN(1,FILE='SIDE',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) N2
$C6 CLOSE(1)
$C6 N3=N2/2
$C6 N6=N1+(N1-1)/2+1
$c6 N5=(N3+1)*N6
[I;1;I=1,N5+N1]
[I*N6+N1;-1;I=1,N3+1]
[I*N6+N6;-1;I=0.N3]
&N=N5+N1;
```

```
ELEM 2
NOD8
$c6 OPEN(1,FILE='FREE',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) N1,M1
$C6 CLOSE(1)
$C6 K=(M1-1)/2
$C6 OPEN(1,FILE='SIDE',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) N2
$C6 CLOSE(1)
$C6 N3=N2/2
$C6 N6=N1+(N1-1)/2+1
$c6 N5=(N3+1)*N6
[[I*(N1-1)/2+J;2*(J-1)+1+I*N6;2*J+I*N6;
2*J+1+I*N6;I*N6+N1+J+1;(I+1)*N6+2*J+1;
(I+1)*N6+2*J;(I+1)*N6+2*(J-1)+1;I*N6+N1+J;J=1,(N1-1)/2],I=0,N3]
NOD3
$c6 OPEN(1,FILE='FREE',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) N1,M1
$C6 CLOSE(1)
$C6 K=(M1-1)/2
$C6 OPEN(1,FILE='SIDE',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) N2
$C6 CLOSE(1)
$C6 N3=N2/2
$C6 N6=N1+(N1-1)/2+1
$c6 N5=(N3+1)*N6
[I;(I-1)*N6+1;(I-1)*N6+N1+1;I*N6+1;I=1,N3+1]
NODUV
$c6 OPEN(1,FILE='FREE',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) N1,M1
```

```
$C6 CLOSE(1)
$C6 K=(M1-1)/2
$C6 OPEN(1,FILE='SIDE',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) N2
$C6 CLOSE(1)
$C6 N3=N2/2
$C6 N6=N1+(N1-1)/2+1
$c6 N5=(N3+1)*N6
[[I*(N1-1)/2+J;2*(J-1)+1+I*N6;2*J+1+I*N6;(I+1)*N6+2*J+1;
(I+1)*N6+2*(j-1)+1;J=1,(N1-1)/2],I=0,N3]
&N=(N3+1)*N6+2*(N1-1)/2+1;

DISP
$c6 OPEN(1,FILE='FREE',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) N1,M1
$C6 CLOSE(1)
$C6 K=(M1-1)/2
$C6 OPEN(1,FILE='SIDE',FORM='FORMATTED',STATUS='OLD')
$C6 READ(1,*) N2
$C6 CLOSE(1)
$C6 N3=N2/2
$C6 N6=N1+(N1-1)/2+1
$c6 N5=(N3+1)*N6
[I;0.0;I=1,N5+N1]
```

4. Main File

.NFE File: dam.nfe

```
defi
stif S
mass M
load F
```

```
type e
mdty 1
step 0

equation
matrix = [S]
FORC=[F]

SOLUTION U
write(s,unod) U

end
```