Fluid Dynamicist Workbench

Kenneth Man-kam Yip

Research Report YALEU/DCS/RR-1008
February 1994

Fluid Dynamicist Workbench

KENNETH MAN-KAM YIP*
Department of Computer Science
Yale University
P.O. Box 208285, Yale Station
New Haven, CT 06520-8285.
(yip-ken@cs.yale.edu)

February 16, 1994

I.2.1 Artificial Intelligence: Applications and Expert Systems
I.6 Simulation and Modeling
J.2 Physical Science and Engineering
J.6 Computer-aided Engineering
I.1.4 Algebraic Manipulation: Applications

words: heuristic reasoning, expert systems, qualitative physics, intelligently guided numerical experimentation.

by NSF NYI Award ECS-9357773

Summary– We discuss the design of an intelligent workbench based on a library of powerful heuristic and qualitative methods. We demonstrate the feasibility of the approach by exhibiting computer programs solving a variety of problems in fluid mechanics.

Introduction

Symbolic and numerical techniques incorporating powerful problem solving ideas based on general mathematical and domain-specific knowledge can lead to a new category of computational tools that might dramatically increase the productivity of scientists. Professional scientists routinely apply powerful heuristic and qualitative methods to simplify a problem and get at its essence before any lengthy calculation is attempted. Methods such as dimensional and order of magnitude estimates, exploitation of small parameters, consideration of limiting cases, perturbation expansions, use of analytical properties of physical quantities, consequences of symmetry, diagrammatic representations, and renormalizability of theory constitute the "bag of tricks" that practicing professionals have but are seldom articulated in a systematic, constructive manner that beginners can apply directly in scientific works.

Although developed in highly specialized disciplines, these problem solving methods are generally applicable to many branches of physics or engineering. For instance, the tremendously successful renormalization group techniques developed for the study of phase transitions may be equally impressive when applied to problems in high energy physics as well as theories of turbulence.

The application of these problem solving methods, however, is rarely straightforward. Basic ideas behind these methods are often obscured by technical and mathematical details. An "outsider" has to spend an enormous amount of effort to learn a new method, to abstract the essential points, and to try it out on his or her problems. The difficulty will be compounded if the method originates in an discipline unfamiliar to the practitioner.

We propose to build an intelligent workbench, based on a library of problem solving methods, which allows scientists to quickly assemble a domain-specific program for testing the applicability of the methods. For instance, to apply the renormalized perturbation theory to fluid turbulence, a fluid dynamicist specifies the necessary field equations, field quantities (like velocity and its correlation tensor), and diagrammatic rules for relating quantities and diagram elements. The workbench responds by giving the perturbation expansions for the field quantities and their corresponding diagrams. It classifies the diagrams and checks if subclasses of diagrams can be summed. It displays a list of renormalized irreducible diagrams. It suggests possible approximate models by selecting subclasses of diagrams.

Our approach to the design of problem solving environment for scientists, which might be called the *power toolkit approach* [1], complements the more common task-specific and generic approach. While useful for the intended community of experts, a task-specific program, say, for computing the higher order Feynman diagrams for the electroweak Standard Model, might be of little use to someone interested in using Feynman-like diagrams in statistical mechanics or fluid turbulence. Similarly, the general-purpose symbolic algebra systems and numerical libraries have not been very helpful in the application of heuristic and qualitative methods.

No workbench as sophisticated as the one envisioned exists. But we will show the necessary first steps have been taken. We begin by demonstrating that the core ideas of many of these powerful methods can be automated. The examples we picked are either historically important in the development of a particular problem solving idea, or of some current research interests. For demonstration purposes, we will focus mainly on problems in fluid mechanics, a vital area for research and practical applications.

Next, we explain the technology behind these programs. One recurrent theme in these programs is that their source of power derives as much from the general problem solving methods and domain-specific knowledge as from the symbolic and numerical techniques that implement them.

Finally, we conclude with some lessons we learn in building these programs.

Powerful methods used by professional scientists can be automated.

In a typical theoretical fluid dynamics study, an investigator identifies some interesting fluid phenomena, formulates a fundamental set of equations of motion, derives an approximate model from the full equations, obtain predictions from analytical (which may be approximate too) or numerical solutions to the approximate model, and compares the predictions with experiments or numerical studies of the full equations. Even with the help of powerful numerical computers and computer algebra systems, this process is time-consuming and requires a substantial amount of human expertise and judgement in simplifying the full model, deducing consequences from the simplified models, and preparing and interpreting the numerical simulation.

In this section, we will demonstrate programs that can automate much of this process. The OOM program uses order of magnitude reasoning to simplify boundary layer problems and a problem with multiple time scales. The KAM2 program applies dynamics knowledge geometrically to explain numerical results. The GI program uses the properties of analytic functions to select integration contour in the complex plane for approximating integrals. The BF program uses an elementary version of Feynman-like diagrams to manipulate integrals and renormalize perturbation expansions.

Programs can generate approximate models.

Composite models simplify modeling.

Few problems in fluid dynamics can be solved exactly because of the complexity of the Navier Stokes equations. The full description is necessarily complex because it incorporates all possible

physical effects that might the fluid motion. In most applications, not all of these effects are equally important. By identifying and eliminating the unimportant effects, an investigator can obtain a useful simplification of the the full model.

A powerful idea to produce approximate model is the use of *composite model.* The idea is to divide up the problem domain into different regions where all but a few effects can be safely ignored. The relative importance of the effects can be estimated by comparing the order of magnitude of the physical quantities involved. As a first approximation, quantities that are much smaller than the rest are dropped. This approximation procedure, however, has to be carried out judiciously because quantities that appear small might not actually be small.

Perhaps the most important application of the order of magnitude approximation in the history of fluid dynamics is Prandtl's boundary layer approximation for slightly viscous flows over solid boundaries. Consider as an example a two dimensional steady flow over a semi-infinite flat plate. The full model for this problem is the 2D steady incompressible Navier-Stokes equations – a set of 3 coupled nonlinear PDEs (Fig. 1). No analytical solution method is known for the full model.

In many applications, the parameter Reynolds number of the flow is large (usually of $O(10^3)$ or higher) and so it is reasonable to exploit this parameter for simplifying the equations. In particular, in the limiting case of $Re \rightarrow \infty$, it might appear that the viscous terms in the equations can be dropped because these terms are multiplied by the reciprocal of a large number. However approximate models that neglect the viscous terms will predict zero drag on a solid body in steady flow; results diverge from reality.

Prandtl's idea is that at high Reynolds numbers viscosity remains important near the body surface even if it could be disregarded everywhere else. As long as the "no-slip" condition holds, i.e., that fluids do not slip with respect to solids, there will be a thin layer around the body where rapid changes of velocity produce notable effects, despite the small coefficient $\frac{1}{Re}$. The layer in question is called *boundary layer.* The width of the boundary layer, usually denoted by δ, is an important quantity. By arguing that within the narrow layer the viscous effects must balance the inertia effects, Prandtl concluded that $\delta = O(\frac{1}{\sqrt{Re}})$.

We will show how a computer program automatically generates Prandtl's boundary layer model and derives the same conclusion about the boundary layer thickness. The input to the program is shown in Fig. 1. The problem is specified by (1) a list of quantities: dependent variables, independent variables, and parameters, (2) the full model equations, and (3) order of magnitude estimates of a few quantities. Notice that each input quantities have associated physical features such as space, velocity, and pressure. These features are used to determine the physical meaning of derived quantities by simple rewrite rules. For instance, a velocity quantity differentiated by a space quantity gives a velocity-gradient quantity.

The order of magnitude estimates in the problem defines the geometry of the flow. For instance, the streamwise direction x is scaled to unity, while the transverse direction y is given

Model name: prandtl-boundary-layer
Independent variables:

x	lower-bound = 0 upper-bound = 1
	physical features = space,streamwise
y	...

Dependent variables:

u	depends-on = x, y
	lower-bound = 0 upper-bound = 1
	physical features = velocity,streamwise
v	...

Parameters:

Re	type = large-parameter
	physical features = dimensionless-number
δ	type = small-parameter
	physical features = dimensionless-number,length,transverse

Equations:

Streamwise-momentum: $u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2})$

Transverse-momentum: $u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2})$

Continuity: $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$

Order of Magnitude estimates:

u = O(1)
x = O(1)
y = O(δ)

Figure 1: The problem specification for flow over semi-infinite plate. It consists of three parts: (1) declarations of variables and parameters, (2) full equations of motion, and (3) a few order of magnitude estimates.

a symbolic estimate δ with the constraint that δ is a small parameter, i.e., $\delta \ll 1$. With the problem specification, the program searches for possible simplifications of the full equations (Fig. 2).

The program generates two approximate models because it does not have enough information to estimate the importance of the pressure gradient term $\frac{dp}{dx}$ in the streamwise momentum equation. These models are much simpler: the transverse equation just says the pressure is constant across the boundary layer, and the streamwise equation becomes a parabolic PDE, which is much easier to handle numerically. The program also computes the drag coefficient based on the order of magnitude estimate for the transverse velocity gradient $\frac{du}{dy}$.

Composite models may have interacting subparts.

Prandtl's approximation is an example of a composite model with nearly independent subparts, i.e., the role of the free stream non-viscous flow is limited to setting the external pressure for the boundary layer, and to a first approximation the free stream flow is not affected by what

> (search-simplifications *model*)
Making <MODEL-2: PRANDTL-BOUNDARY-LAYER-1> from <MODEL-1: PRANDTL-BOUNDARY-LAYER>...
Assigning the leading order terms of TRANSVERSE-MOMENTUM-EQUATION to be: DPDY
Significant terms in STREAMWISE-MOMENTUM-EQUATION are:
UDUDX VDUDY D2UDY2/RE
STREAMWISE-MOMENTUM-EQUATION: 2 candidate dominant sets:
(D2UDY2/RE VDUDY UDUDX)
(D2UDY2/RE DPDX VDUDY UDUDX)

Making <MODEL-3: PRANDTL-BOUNDARY-LAYER-1> from <MODEL-2: PRANDTL-BOUNDARY-LAYER-1>...
Balancing 2 terms:
D2UDY2/RE (VISCOUS STRESS TRANSVERSE) and
VDUDY (CONVECTIVE-ACCELERATION INERTIA TRANSVERSE)
in STREAMWISE-MOMENTUM-EQUATION with 1 parameter assumption: $Re = \frac{1}{\delta^2}$
Assigning the leading order terms of STREAMWISE-MOMENTUM-EQUATION to be:
D2UDY2/RE and VDUDY and UDUDX
<MODEL-3: PRANDTL-BOUNDARY-LAYER-1> is self-consistent.

Making <MODEL-4: PRANDTL-BOUNDARY-LAYER-2> from <MODEL-2: PRANDTL-BOUNDARY-LAYER-1>...
Balancing 3 terms:
D2UDY2/RE (VISCOUS STRESS TRANSVERSE) and
DPDX (PRESSURE-GRADIENT) and
VDUDY (CONVECTIVE-ACCELERATION INERTIA TRANSVERSE)
in STREAMWISE-MOMENTUM-EQUATION with 1 parameter assumption: $Re = \frac{1}{\delta^2}$
Assigning the leading order terms of STREAMWISE-MOMENTUM-EQUATION to be:
D2UDY2/RE and DPDX and VDUDY and UDUDX
<MODEL-4: PRANDTL-BOUNDARY-LAYER-2> is self-consistent.

Two simplified models are found.

Model-3:

$$\begin{aligned} u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= \frac{1}{Re}\frac{\partial^2 u}{\partial y^2} \\ \frac{\partial p}{\partial y} &= 0 \\ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \end{aligned}$$

Model-4:

$$\begin{aligned} u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= -\frac{\partial p}{\partial x} + \frac{1}{Re}\frac{\partial^2 u}{\partial y^2} \\ \frac{\partial p}{\partial y} &= 0 \\ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \end{aligned}$$

The drag coefficient is $O(\frac{1}{\sqrt{Re}})$.

Figure 2: The program finds two self-consistent simplified models. It deduces that $Re = \frac{1}{\delta^2}$ in both of these models. It further computes the drag coefficient based on the magnitude of the velocity gradient $\frac{du}{dy}$.

happens inside the thin boundary layer. In more complicated fluid problems, we find that the submodels often interact non-weakly. To determine the structure of each submodel and its region of validity, we have to consider not only the balancing of effects in the region, but also the matching conditions at the boundaries of adjacent regions.

A good example of a composite model with interacting subparts is Stewartson and Messiter's triple deck model which successfully models flows near the trailing edge of a flat plate [20]. The

model is considered a major landmark in theory of laminar flow because the same technique can be applied immediately to many other problems, including flow near a corner, a hump, a tilted trailing edge, and even supersonic flow.

Roughly, the triple deck model has a 3-layer structure near the trailing edge: a "lower deck" reacts to the solid boundary just like Prandtl's thin boundary layer, a "upper deck" is controlled by the free stream condition, and a "main deck" transmits the displacement effect of the lower deck to the upper deck. Given the qualitative structure of the composite model, an investigator has to determine the equations of motion in each deck, and the region of validity for each deck. The reasoning required for such quantitative determination is far from trivial. To get a feel of what's involved, we will show a record of how an expert [19] arrives at the conclusion:

> Consider the basic physical characteristics of the triple-deck structure hand in hand with an order of magnitude argument to support the $O(Re^{-\frac{3}{8}})$ length scaling (of the region around the trailing edge for which the composite model is applicable).
>
> First, close to the plate, in the lower deck, a nonlinear viscous response is forced subject to the action of the induced pressure gradient. So if the lower deck thickness if $Re^{-\frac{1}{2}}\,\delta$ and the length scale is l, where δ and l are unknown but small, then the streamwise velocity is $O(\delta)$ because the oncoming velocity profile gives approximately a uniform shear there. Hence the induced pressure must be $O(\delta^2)$ for its gradient to balance the inertial force, while the balance with the viscous force requires $\delta = O(l^{\frac{1}{3}})$.
>
> Second, the main deck spanning the $O(Re^{-\frac{1}{2}})$ boundary layer is simply displaced in an inviscid manner by an amount $O(Re^{-\frac{1}{2}}\,\delta)$ due to the lower deck thickness, the displaced motion being linearized but rotational because of the oncoming curved profile, while the pressure is virtually unaltered across the main deck.
>
> Third, that displacement is transmitted to the upper deck, outside the boundary layer, where linearized irrotational flow properties hold in the effectively uniform stream there and induce an inviscid pressure response of the order of the displacement slope $O(Re^{-\frac{1}{2}}\,\frac{\delta}{l})$.
>
> The final element of the argument is that this external pressure of order $O(Re^{-\frac{1}{2}}\,\frac{\delta}{l})$ should coincide with the plate pressure of order δ^2...hence since $\delta = O(l^{\frac{1}{3}})$, we recover the triple-deck scaling of $l = O(Re^{-\frac{3}{8}})$, $\delta = O(Re^{-\frac{1}{8}})$, and $p = O(Re^{-\frac{1}{4}})$.

In the following transcript, we will show how a computer program combining the order of magnitude reasoning (as illustrated in the boundary layer example) with symbolic methods of equation and inequality solving can implement this type of reasoning.

The problem specification (Fig. 3) describes the qualitative triple deck structure. Symbolic order of magnitude estimates are assigned to many dependent and independent variables. The convention is to use hat ^ to denote the upper deck quantities, tilde ~ the main deck, and bar - the lower deck. For example, $\hat{\pi}$, $\tilde{\pi}$, and $\bar{\pi}$represent the symbolic order of magnitudes for the pressure in upper, main, and lower deck respectively. Since the problem requires going beyond

the leading order approximation, the horizontal velocity u is given as a perturbation expansion. The `inherit` descriptor tells the program the type of flow it can assume for a particular region. For instance, in the lower deck we specify the flow belongs to Prandtl's boundary layer type, but the horizontal ($x = Re^{\alpha}$) and vertical extent ($y = Re^{\beta}$) for which this flow type holds are left unspecified. In the main deck where we don't have much information, we just say the flow is a general 2D incompressible steady flow.

```
Model name: flat-plate-with-trailing-edge
Dependent variables:
        P0      depends-on: x
                lower-bound = 0 upper-bound = 1
                physical features = pressure,external
        UB      ...
Parameters:
        Re      type = large-parameter
                physical features = dimensionless-number
        σ̃       type = small-parameter
                physical features = dimensionless-number,streamwise,velocity
        ...
Scalars:
        α       type = small-parameter
                physical features = dimensionless-number,length
        β       ...
Relations:
        α < 1/2 , β > 1/2
Regions:
        layout: :vertical (lower-deck, main-deck, upper-deck)
        lower-deck
                inherit: prandtl-boundary-layer
                dependent variables: u1 ...
                expansion: u = Re^(1/2 − β) u1
                ...
        main-deck
                inherit: 2d-incompressible-steady-flow
                dependent variables: u1 ...
                expansion: u = UB + u1
                ...
        upper-deck
                inherit: 2d-incompressible-steady-flow
                dependent variables: u1 ...
                expansion: u = U∞ + u1
                ...
```

Figure 3: The problem specification for flow over finite plate. It is a composite model consisting of three subregions. Subregions can be specified by the type of flow they inherit. Local declarations of parameters and variables are permitted. Some variable declarations and all boundary conditions are omitted.

The output of the program is shown in Fig. 4. The program finds approximate models for each deck, and determines the symbolic estimates as functions of the Reynolds number.

Upper Deck:

$$\begin{aligned} \frac{\partial u_1}{\partial x} &= -\frac{\partial p}{\partial x} \\ \frac{\partial v}{\partial x} &= -\frac{\partial p}{\partial y} \\ \frac{\partial u_1}{\partial x} + \frac{\partial v}{\partial y} &= 0 \end{aligned}$$

Main Deck:

MD-1

$$\begin{aligned} UB\frac{\partial u_1}{\partial x} + v\frac{\partial UB}{\partial y} &= 0 \\ \frac{\partial p}{\partial y} &= 0 \\ \frac{\partial u_1}{\partial x} + \frac{\partial v}{\partial y} &= 0 \end{aligned}$$

MD-2

$$\begin{aligned} UB\frac{\partial u_1}{\partial x} + v\frac{\partial UB}{\partial y} &= -\frac{\partial p}{\partial x} \\ \frac{\partial p}{\partial y} &= 0 \\ \frac{\partial u_1}{\partial x} + \frac{\partial v}{\partial y} &= 0 \end{aligned}$$

MD-3

$$\begin{aligned} UB\frac{\partial u_1}{\partial x} + v\frac{\partial UB}{\partial x} &= -\frac{1}{Re}\frac{d^2UB}{dy^2} \\ \frac{\partial p}{\partial y} &= 0 \\ \frac{\partial u_1}{\partial x} + \frac{\partial v}{\partial y} &= 0 \end{aligned}$$

MD-4

$$\begin{aligned} UB\frac{\partial u_1}{\partial x} + v\frac{\partial UB}{\partial x} &= -\frac{\partial p}{\partial x} - \frac{1}{Re}\frac{d^2UB}{dy^2} \\ \frac{\partial p}{\partial y} &= 0 \\ \frac{\partial u_1}{\partial x} + \frac{\partial v}{\partial y} &= 0 \end{aligned}$$

Constraints found by matching and coincidence:

$$\begin{aligned} \tilde{\sigma} Re^{\alpha-\frac{1}{2}} &= Re^{2\alpha-4\beta+\frac{3}{2}} \\ \tilde{\sigma} Re^{\alpha-\frac{1}{2}} &= \hat{\sigma} \\ \tilde{\pi} &= \bar{\pi} \\ \tilde{\pi} &= \hat{\pi} \\ \hat{\pi} &= \hat{\sigma} \\ \bar{\pi} &= Re^{\beta-\frac{1}{2}-\alpha} \\ \alpha &= 3\beta - \frac{3}{2} \end{aligned}$$

Solution of constraints:

$$\begin{aligned} \alpha &= \frac{3}{8} \\ \beta &= \frac{5}{8} \\ \hat{\pi} = \tilde{\pi} = \bar{\pi} &= Re^{\frac{1}{4}} \\ \hat{\sigma} &= Re^{\frac{1}{4}} \\ \tilde{\sigma} &= Re^{\frac{1}{8}} \end{aligned}$$

Figure 4: Output for the triple deck problem. The simplified model for the upper deck is a potential flow. The main deck has four consistent models. Only MD-1 can produce enough matching constraints for solving the seven unknowns. The lower right corner shows the solution corresponding to MD-1.

Perturbation on a known solution further simplifies models.

A powerful strategy to exploit a small parameter in a full model is to solve the model with the small parameter set to zero to obtain the so-called *unperturbed solution*, and approximate the solution to the full model by a perturbation expansion of the unperturbed solution in terms of powers of the small parameter. This strategy is a generalization of the Taylor series expansion

in elementary calculus. The perturbation method can often lead to drastic simplifications, but the danger is that the method will not reveal anything about other solutions that might exist but don't look like the unperturbed solution.

Computer implementations of various formal perturbation methods (such as the method of multiple scales) exist. While useful for automatically generating higher order approximations (which would be quite tedious to carry out manually), these programs are not particularly helpful in setting up the perturbation scheme in the first place because they require as part of the input the choice of relevant length and time scales in order to determine the form of the perturbation expansion.

We will demonstrate a computer program, which combines order of magnitude reasoning with the method of multiple scale, to solve a problem involving the chaotic wave motion inside a wave tank equipped with a wave-making paddle [21].

The primary question is how to predict the character of the standing waves – their amplitude and period – in response to different controllable parameters such as the amplitude and angular frequency of the paddle, the length-width ratio of the tank, water depth, and so on. Experimentally it is known that both **longitudinal waves**, sinusoidal waves whose crests are parallel to the wave-maker and which oscillate with the same frequency as that of the wave-maker, and **cross waves**, one whose crests are at right angles to the wave-maker with half the frequency of the wave-maker, can be excited [12].

Let us formulate the problem in mathematical terms. The fluid is assumed to be inviscid and the flow irrotational, such that the velocity $\mathbf{v}$ can be expressed as the gradient of a scalar function, the velocity potential ϕ: $\mathbf{v} = \nabla\phi$. We will ignore surface tension. The equations of motion can then be written as Laplace's equation with nonlinear boundary conditions (Fig. 5). The formulation is quite general; it applies to a large class of water wave problems [14]. Note that the nonlinear boundary conditions (1) and (2) involve the unknown free surface ξ; this is why this class of problems are difficult to solve analytically or numerically.

Given the problem specification as shown in Fig. 5, the program automatically generates the amplitude equations governing the evolution of the longitudinal wave and the cross wave.

The output of the program is shown in Fig. 6. Two key features in the output are worth noting. First, by assuming the unperturbed solutions for the longitudinal and cross waves are ordinary sinusoidal waves, the program determines the time scale τ necessary for observing these waves to be resonant with the driving force. Second, the complicated wave tank equation is decomposed into three relatively simple sets of ordinary differential equations, which can be passed on to other programs that know how to analyze or numerically simulate ODEs.

A rectangular tank of water of width W, length L, and filled to a depth D.
The flow is assumed to be incompressible, inviscid, and irrotational. In terms
of dimensionless variables, the governing equation is given by:

$$\nabla^2 \phi = 0$$

where ϕ is the velocity potential.
There are 5 boundary conditions:
(1) A fluid particle at the free surface ($z = \xi(x, y, t)$) always remains at the surface:

$$\phi_z = \xi_t + \phi_x \xi_x + \phi_y \xi_y$$

where the subscripts denote partial derivatives.
(2) The pressure within the fluid motion must conform to Bernoulli's equation:

$$\phi_t + g\xi + \frac{1}{2} \nabla\phi \cdot \nabla\phi = 0$$

(3) On the sidewalls and bottom of the tank, there must be no normal velocity:

$$\begin{aligned} \phi_x &= 0 \ on \quad x = l \\ \phi_y &= 0 \ on \quad y = 0, 1 \\ \phi_z &= 0 \ on \quad z = -d \end{aligned}$$

where the tank width is scaled to 1; length, l; and depth, d.
(4) The wave-maker oscillates with frequency ω and amplitude ϵ:

$$x = \epsilon(1 + \frac{z}{d}) \cos \omega t$$

(5) The mean surface height is taken to be z = 0.

Figure 5: Formulation of the standing wave problem in a rectangular tank.

Renormalizing a perturbation expansion can lead to accurate approximate models.

Almost any physical system can be thought of a collection of many interacting particles – many electrons, many molecules, many fluid particles, for example. We will demonstrate a computer program which can help solve a many-body problem using a powerful technique known as **renormalization.**

Computing the detailed description of a many-particle system, say a box of helium of 10^{23} atoms, in terms of positions and velocities for every particle is totally impractical. Fortunately many important measurable quantities of interest can be calculated from the knowledge of the *average* behavior of one or two particles. The quantities describing the average behavior of one particle and two particles are known as the **1-particle propagator** or **2-particle propagator** respectively. The calculation of these propagators is a fundamental problem in many branches of physics.

There are three cases to consider:
(I) Assume $A \gg B$. Then $A = O(\epsilon^{\frac{1}{3}})$ and $\tau = O(\epsilon^{\frac{2}{3}})$.

$$\mu\frac{dA}{d\tau} + i2\lambda\mu A + i\Gamma A^2 A^* = \tfrac{1}{4}\,\delta$$

(II) Assume $B \gg A$. Then $B = O(\epsilon^{\frac{1}{2}})$ and $\tau = O(\epsilon)$.

$$\mu\frac{dB}{d\tau} + i2\lambda\mu B - i\beta B^* + i\Gamma B^2 B^* = 0$$

(III) Assume $A \sim B$. Then $A = B = O(\epsilon^{\frac{1}{2}})$ and $\tau_1 = O(\epsilon^{\frac{1}{2}}), \tau_2 = O(\epsilon)$.
Let $\frac{\partial}{\partial\tau} = \frac{\partial}{\partial\tau_1} + \epsilon^{\frac{1}{2}}\frac{\partial}{\partial\tau_2}$.

$$\begin{aligned}\mu\frac{dA}{d\tau} + i\gamma A + i\Gamma A^2 A^* - i\Sigma ABB^* &= \delta \\ \mu\frac{dB}{d\tau} + i\gamma B - i\beta B^* - i\Gamma B^2 B^* - i\Sigma BAA^* &= 0\end{aligned}$$

Figure 6: The computer automatically derives three sets of amplitude equations depending on the relative ordering of complex amplitude A of the longitudinal wave, and the complex amplitude B of the cross wave amplitude: (I) only the longitudinal wave is resonant, (II) only the cross wave is resonant, and (III) both the longitudinal and cross waves are resonant. τ is the long time scale.

As a simple example of 1-particle propagator, consider a test particle at position 1 moving through a medium of particles or scattering centers, labeled A, B, C, and so on. What is the probability that the test particle will emerge at position 2? We expect the total probability of the test particle going from 1 to 2 to be the sum of all possible paths through the medium. For example, one path the test particle can propagate is to go freely from 1 to 2 without hitting any other particles. Call this probability $G_0(1,2)$. Another path it can follow is to go freely until it hits one particle, and then go freely from that particle to 2. Call this probability $G_1(1,2)$, which will be a sum over all possible particles:

$$G_1(1,2) = G_0(1,A)P(A)G_0(A,2) + G_0(1,B)P(B)G_0(B,2) + \cdots$$

where P(A) and P(B) are the probability of hitting particle A and B respectively.

A more complicated path the test particle can follow is hitting two particles before it reaches 2, or hitting three particles, and so on. We can write the total probability formally as an expansion series:

$$G(1,2) = G_0(1,2) + G_1(1,2) + G_2(1,2) + \cdots$$

Each $G_k(1,2)$ can be expressed in terms of a sum of G_0 factors over all possible groups of k particles, and the expression gets complicated quickly as k increases. One might be tempted to truncate the series after the first few terms in the expansion for G. The truncated series,

however, can be a poor approximation because although an individual term in the sum for G_k may be small compared to those in G_0 or G_1, there are many more such terms to sum over in G_k. Another reason for poor approximation is that sometimes there are terms that cannot be neglected to all orders of G_k. For example, if particle A is a particularly strong attractive center, its interactions with the test particle have to be summed to all orders of G_k to get an accurate approximation.

The process of rearranging the series for G and partially summing selected classes of terms to improve convergence of the series is called **renormalization**, a routine tool in present day theoretical physics. Renormalization was first introduced into fluid mechanics in 1961 by Wyld as a framework for constructing theories of turbulence [23]. So far the success of the approach has been mixed, and it continues to be a subject of considerable research interest [13].

To illustrate how a computer program can use the renormalization technique to generate approximate models, we will use an example from statistical mechanics, which is considerably simpler than fluid turbulence, and which allows the essential points of the technique to be clearly seen. The example problem is the derivation of the equation of state for moderate density imperfect gas from first principles. This problem was solved by J.E. Mayer in 1937 and was considered a landmark in the development of that subject [3]. The input to the program is shown in Fig. 7, and output in Fig. 8. Note that if the intermolecular potential $\phi = 0$, i.e., gas particles don't interact, then $B(T) = 0$ and the equation of state becomes the ideal gas law.

The partition function Z:

$$Z = \frac{1}{N!} \int e^{-\frac{E}{kT}} d\Omega$$

If $E = \sum \frac{p_i^2}{2m} + \lambda\phi$, **then** $Z = \frac{1}{N!}(\frac{2\pi mkT}{h^2})^{\frac{3N}{2}} Q$
where the configuration integral Q is given by:

$$Q = \int e^{-\lambda \frac{\phi}{kT}} dV_1 \ldots dV_n$$

Assume: $\phi = \sum_{pairs\ i,j} u(r_{ij})$
Equation of state:

$$\frac{P}{kT} = (\frac{\partial \log Z}{\partial V})_T = (\frac{\partial \log Q}{\partial V})_T$$

Figure 7: The problem specification for deriving the equation of state directly from the partition function Z or equivalently the configuration integral Q.

Programs can analyze integral solutions graphically.

Throw a pebble into a quiet pond and observe how the wave crests evolve. The Cauchy-Poisson problem is the problem of predicting the amplitude and location of wave crests as a function

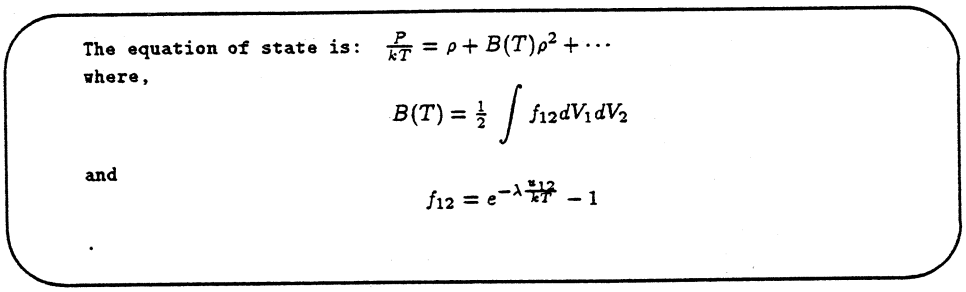


Figure 8: The program derives the second term in the expansion of the equation state. The coefficient B(T) is the integral over irreducible clusters of two particles (to be explained later).

of time given an initial disturbance. If the initial disturbance is not too large, one can linearize the full water wave equations (Fig. 5) and obtain a formal solution by superposition of simple progressive waves of the form $e^{i(kx-\omega t)}$ where k is the wave number and ω is the frequency, and summing over all wavenumbers:

$$\eta(x,t) = \int_{-\infty}^{\infty} A(k) e^{i(kx-\omega t)} dk$$

where $A(k)$ is related to initial conditions. Such formal solution is in general difficult to analyze exactly. Simplification of the integral in terms of elementary functions can often be obtained if we are interested in the long time or farfield behaviors of the integral.

We will demonstrate how a particular asymptotic technique, the method of steepest descent [4], can be automated to produce approximate analytical representations for a wide class of integrals. The technique is selected for two reasons: (1) it is widely used in many branches of physics (in particular, the evaluation of partition function in statistical mechanics and path integrals in quantum field theory), and (2) it gives higher order approximations. The method however can be difficult to apply because it requires a judicious choice of integration contour in the complex plane.

Another novel feature of the program is that it evaluates the higher order terms by a graphical method – a "baby" version of Feynman diagrams. The graphical method is not the most efficient way for solving relatively simple Gaussian-type integrals [7], but its ability to simplify bookkeeping in manipulating integrals, organize integrals into classes, and provide a physical interpretation for algebraic expressions make it an invaluable tool in many areas of theoretical physics.

As our model problem, we use the Airy function (or the rainbow integral), in its complex form:

$$Ai(\lambda) = \frac{1}{2\pi i}\int_C e^{\lambda z - \frac{1}{3}z^3} dz$$

which has found applications in many branches of physics – ray diffraction in optics, tunneling of quantum particles, and evolution of wavefront of tsunamis, just to name a few. The integral can also be expressed as a solution to the linear differential equation:

$$\frac{d^2w(\lambda)}{d\lambda^2} - \lambda w(\lambda) = 0 \tag{1}$$

The Airy equation has the same form as the time-independent Schrodinger equation in the neighborhood of a classical turning point.

The analysis program takes three inputs: (1) a complex integral with a parameter, (2) the contour for integration, and (3) a description of the parameter. Its output is a simple formula (in terms of elementary functions) representing the leading orders of the asymptotic behavior of the function.

The output of the analysis program is shown in Fig. 9.

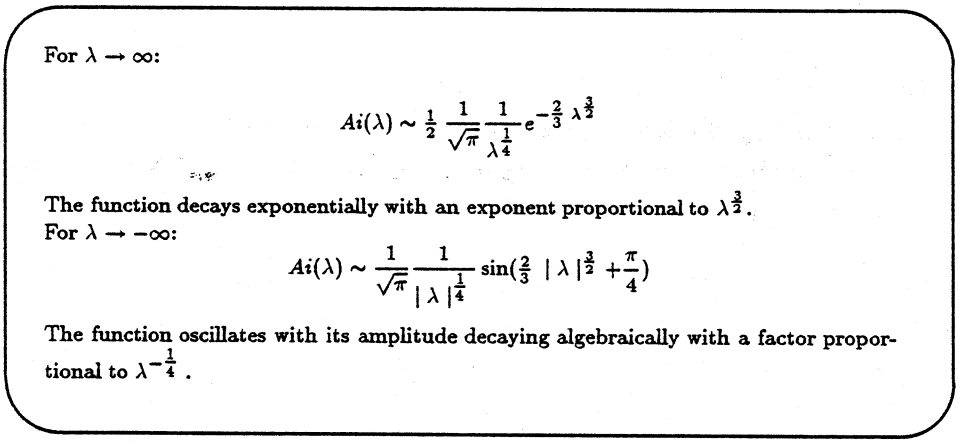


Figure 9: The program finds two qualitative distinct behaviors for $\lambda \to \infty$ and $\lambda \to -\infty$.

Programs can compare analytical and numerical predictions.

In the study of a physical problem, an investigator often makes many idealizations and approximations in order to arrive at some quantitative or semi-quantitative predictions. It is therefore important to compare any analytical predictions with either numerical results or experimental data or both.

In previous papers, we demonstrated how computer programs exploiting structural theorems in phase space can automate the numerical modeling of low-dimensional ODEs. We will show that such programs can be extended to automate much of the comparison work too. As an illustration, we consider the chaotic wave tank problem again. We will focus on the amplitude equations describing the coupling of the resonant longitudinal wave and cross wave. This case

is interesting because the resulting dynamics is quite complicated: it can exhibit chaos and multiple bifurcations of steady states.

The input to the program is the amplitude equations (case III of Fig. 6). The amplitude equations can be thought of as a Hamiltonian system with two degrees of freedom. A complete description of the system therefore requires a 4-dimensional phase space. Use of a Poincare section and the conservation of the hamiltonian reduce the 4-dimensional flow to a 2-dimensional discrete mapping, called the first return map.

Automatic numerical experiments with the amplitude equations reveal an interesting finding. At detuning frequency $\lambda = 0.1$ and low energy level $E = 1$, the phase portrait appears completely regular: an elliptic fixed point at the origin surrounded by a nested sequence of invariant curves (see Fig. 10a). When the energy is raised to $E = 3$, the outermost invariant curves are destroyed, and a large chaotic zone is seen (Fig. 10b). As the energy is further increased to $E = 6$, the total energy surface shrinks but the orbits appear completely regular again (Fig. 10c). I call this scenario the **banded energy phenomenon** because chaotic orbits seem to appear only for an interval (or a band) of energy values.

To explain the finding, the program applies the Resonance Overlap Criterion, a method developed by Chirikov to predict the onset of chaos. The program finds the resonance overlap zones, and reports the values of the hamiltonian at boundaries of the overlap zone (Fig. 11) . The value is consistent with that found from numerical experiments.

Intelligent workbench rests on symbolic techniques incorporating general methods and domain-specific knowledge.

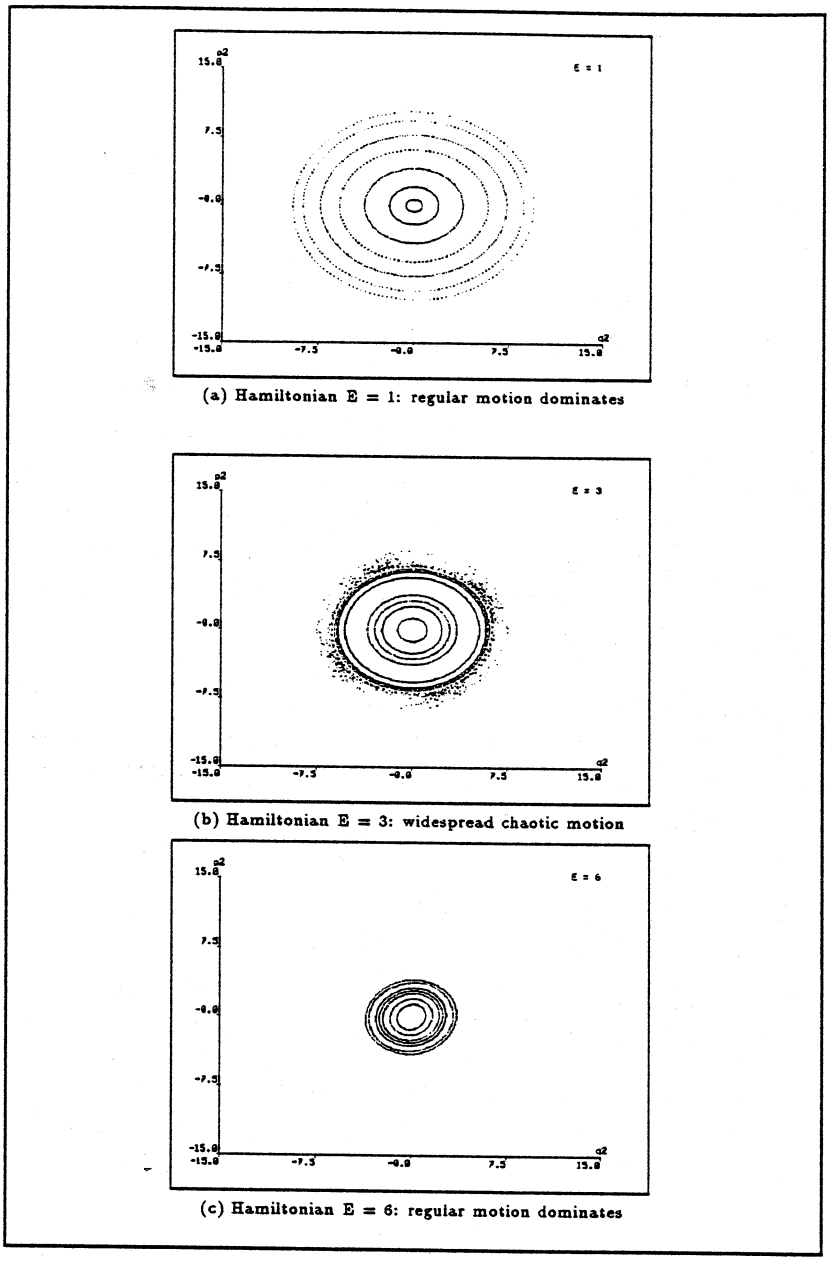
Viewed as abstract examples of symbolic techniques, these demonstration programs are hardly novel. However, the programs gain new power when powerful heuristic methods and specialized knowledge are embedded in them.

Balancing dominant terms guides the search for approximate models.

The basic idea in simplification is to identify small terms in an equation, drop these terms, solve the simplified equation, and check for consistency. But this does not always work. Consider the following simple polynomial:

$$3\epsilon^2 x^3 + x^2 - \epsilon x - 4 = 0$$

in the limit $\epsilon \to 0$. We might naively drop the cubic and the linear terms because their coefficients are small. But if we do that, we only get two roots $x = \pm 2$, losing the third root. Thus, the process of simplification leads to a loss of important information.



(a) Hamiltonian E = 1: regular motion dominates

(b) Hamiltonian E = 3: widespread chaotic motion

(c) Hamiltonian E = 6: regular motion dominates

Figure 10: $\lambda = 0.1$. Banded energy phenomenon: large chaotic zones appear only for an interval (or a band) of energy values.

What went wrong? The problem is that terms that appear small are not really small. The missing root depends inversely on ϵ in such a way that the cubic term is not negligible even its coefficient becomes small. To fix this problem, we introduce three concepts: an **undetermined gauge**, a **significant gauge**, and a **maximal set**. To begin, we will assume $x = O(\epsilon^n)$ where n is still undetermined – hence the name undetermined gauge. The order of each term is then:

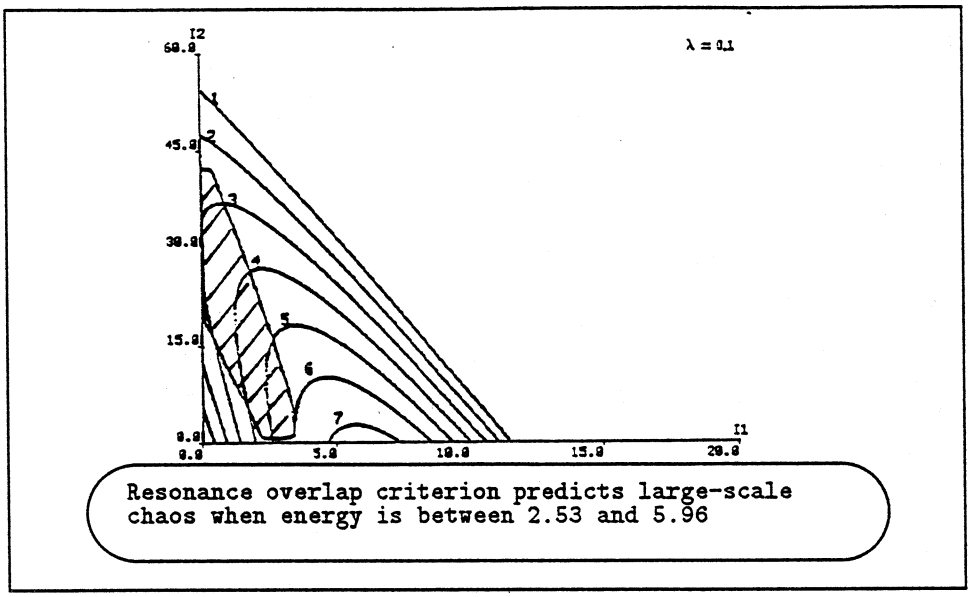


Figure 11: Predicting critical energy levels for large-scale chaos. The level curves of the hamiltonian first touch the overlap region when E is between 2 and 3; they leave the overlap region when E is about 6.

$$\underbrace{3\epsilon^2 x^3}_{O(\epsilon^{3n+2})} + \underbrace{x^2}_{O(\epsilon^{2n})} - \underbrace{\epsilon x}_{O(\epsilon^{n+1})} - \underbrace{4}_{O(1)} = 0$$

To determine the relative importance of terms, we use the heuristic that we only retain the smallest number of terms that will balance the equation. Since we must allow the situation where two or more terms may have the same asymptotic order, we group terms into equivalence classes by the relation $\sim$. A maximal set is any such class that is not smaller than any other classes. As an example, the cubic polynomial above has four maximal sets each containing one term. The heuristic can then be stated as follows:

Heuristic of minimal complication (or Method of Dominant Balance):
If the equation has two or more maximal sets, balance two of them; these two maximal sets are called *dominant.* Assume the remaining sets are negligible. Self-consistent choices of dominant maximal sets correspond to significant simplified equations.

Applying this heuristic to the polynomial, we get six cases to consider. For instance, one possibility is that the first two terms are dominant, i.e., $\epsilon^2 x^3 \sim x^2 \gg \epsilon x, 4$. Equating the two undetermined gauges, we get $3n + 2 = 2n$ and this implies $n = -2$. The remaining terms are $O(\epsilon^{-1})$ and $O(1)$, which is consistent with the assumption that the first two terms are dominant. So this possibility is included. On the other hand, if we assume $\epsilon^2 x^3 \sim \epsilon x \gg x^2, 4$, we get $n = -\frac{1}{2}$. But then $x^2 = O(\epsilon^{-1}) \gg O(\epsilon^{\frac{1}{2}})$, violating the assumption that it should be much smaller than the first term. This possibility must be excluded. A similar analysis shows that only one more possibility, when the second and fourth terms are dominant, i.e., $n = 0$, is

self-consistent. So the heuristic concludes that we should consider *two* simplified polynomials:

$$3\epsilon^2 x^3 + x^2 = 0 \Rightarrow x \sim \frac{1}{3\epsilon^2}$$

and

$$x^2 - 4 = 0 \Rightarrow x \sim \pm 2$$

The values of ϵ^n for which we get self-consistent dominant maximal sets are called **significant gauges.** The balancing of the dominant maximal sets produces simplified equations that correspond to qualitatively significant asymptotic behaviors.

In Fig. 12, we show how order of magnitude argument determines appropriate time scale for the wave tank problem.

$$\begin{aligned}\frac{\partial a}{\partial t} &= \epsilon + a + b + \\ &\quad (a\ a) + (b\ b) + (a\ b) + (\epsilon\ \epsilon) + (\epsilon\ a) + (\epsilon\ b) + \\ &\quad (a\ a\ a) + (a\ a\ b) + (a\ b\ b) + (b\ b\ b) + \cdots\end{aligned}$$

The harmonics of the terms on the right are respectively:

$$\begin{array}{l} 2\ 2\ 1 \\ (0\ \pm 4)\ (0\ \pm 2)\ (\pm 1\ \pm 3)\ (0\ \pm 4)\ (0\ \pm 4)\ (\pm 1\ \pm 3) \\ (\pm 2\ \pm 6)\ (\pm 1\ \pm 3\ \pm 5)\ (0\ \pm 2\ \pm 4)\ (\pm 1\ \pm 3)\ \cdots \end{array}$$

Initially, a is small and so forcing should be dominated by ϵ, the external drive. So $\frac{\partial a}{\partial t} \sim \epsilon$. As resonance develops, the longitudinal wave grows, the nonlinear term (a, a, a) becomes important and balances the effect of ϵ. So we have $a^3 \sim \epsilon$. From these two estimates, we can conclude that the long time scale at which the nonlinearity balances the drive must be $\tau = O(\epsilon^{\frac{2}{3}} t)$.

Figure 12: Order of magnitude argument to determine the long time scale for the longitudinal resonant wave. All possible influences on the modulus of the complex amplitude, a, are listed symbolically. Terms that do not contain the second harmonics are not resonance-forcing and therefore can be ignored.

To implement the dominant balance heuristic, we need a language to talk about **asymptotic behavior** of a function $f(\epsilon)$ as ϵ approaches some critical value ϵ_0. Without loss of generality, we can assume $\epsilon_0 = 0$, since translation $(\epsilon - \epsilon_0)$ and inversion $(\frac{1}{\epsilon})$ can be used to handle any non-zero finite and infinite limiting values.

There are several ways to describe the asymptotic behavior of a function with varying degrees of precision. For instance, we could describe the limiting value $f(\epsilon)$ as $\epsilon \to 0$ qualitatively, i.e., whether it is bounded, vanishing, or infinite. Or, we could describe the limiting value *quantitatively* by giving a numerical value for the bound. But it is most useful to describe the *shape of the function qualitatively* as a limit is approached. The description uses the order symbols O ("big oh"), o ("little oh"), and $\sim$ ("asymptotically equal") to express the relative magnitudes of two functions.

Definition 1 $f(\epsilon) = O(g(\epsilon)), \epsilon \to 0$ if $\lim_{\epsilon \to 0} \frac{f(\epsilon)}{g(\epsilon)} = K$ for some finite K.

Definition 2 $f(\epsilon) = o(g(\epsilon)), \epsilon \to 0$ if $\lim_{\epsilon \to 0} \frac{f(\epsilon)}{g(\epsilon)} = 0$

Definition 3 $f(\epsilon) \sim g(\epsilon), \epsilon \to 0$ if $\lim_{\epsilon \to 0} \frac{f(\epsilon)}{g(\epsilon)} = 1$

Typically, we will use a convenient set of simple functions inside an order symbol; they are called the **gauge functions** because they are used to describe the shape of an arbitrary function in the neighborhood of a critical point. Common gauge functions include the powers and inverse powers of ϵ. For example, $\sin(\epsilon) = O(\epsilon)$ as $\epsilon \to 0$. For more complicated problems, logarithms and exponentials of powers of ϵ may also be used.

The asymptotic order of magnitude must be distinguished from the numerical order of magnitude. If $f = 10^6 g$, then f and g differ by 6 numerical orders of magnitude, but they are still of the same asymptotic order. However, in a physical problem the variables are normally scaled in such as way that the proportionality constant K will be close to 1.

Below we list some useful rules of operation on order symbols:

1. $O(fg) = O(f)O(g)$
2. $O(f+g) = \max(O(f), O(g))$
3. $O(f) + o(f) = O(f)$
4. $o(fg) = O(f)o(g) = o(f)o(g)$
5. If $f = O(g)$, then $\int_0^\epsilon f(t)dt = \int_0^\epsilon \mid g(t) \mid dt$ as $\epsilon \to 0$.

Order relations cannot in general be differentiated. That is, if $f = O(g)$, then it is not generally true that $f' = O(g')$. However, using the definition of the total differential of a function $f(x,y)$, $df = \underbrace{\frac{\partial f}{\partial x}dx}_{df\text{-}x} + \underbrace{\frac{\partial f}{\partial y}dy}_{df\text{-}y}$ where df-x and df-y are the partial differentials, we can derive some useful rules involving partial derivatives:

1. $O(\frac{\partial f}{\partial x})O(dx) = O(df\text{-}x)$
2. $O(\frac{\partial f}{\partial y})O(dy) = O(df\text{-}y)$
3. $O(df) = \max(O(df\text{-}x), O(df\text{-}y))$

Rules for manipulating order of magnitude are implemented as constraints. Each equation in the input specification is expanded into a network of constraints. The purpose of the network is two-fold: (1) to deduce values for quantities, and (2) to assert equality among symbolic expressions when a particular value is deduced in more than one way. So for instance in balancing the inertia term $u\frac{du}{dx}$ and viscous term $\frac{1}{Re}\frac{d^2u}{dy^2}$ in the lower deck equation, the network notices a coincidence of values and asserts that:

$$Re^{1-2\beta+\alpha} = Re^{-\frac{1}{2}+\beta}$$

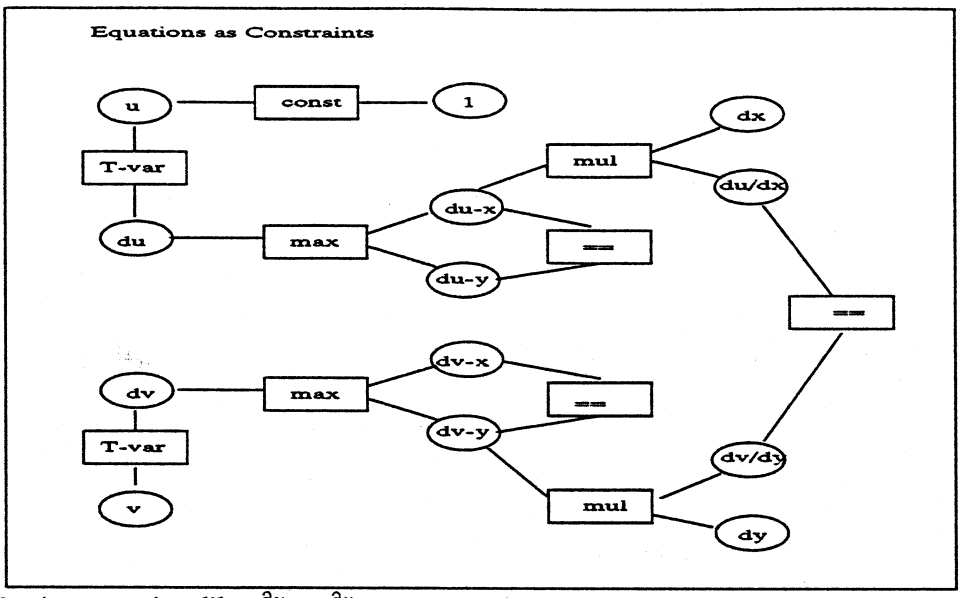


Figure 13: An equation like $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$ is implemented as a constraint network. The ovals stand for quantities, and the boxes for constraints. As an example of constraint, the T-var box compares the upper and lower bound of the quantity u in order to assign a value for the total differential du. Once a quantity is assigned a value, its effect will be propagated throughout the network. Given $u = O(1), x = O(1)$, and $y = O(\delta)$, the network deduces $v = O(\delta)$

Meta-rules simplify equation and inequalities solving.

Order of magnitude reasoning relies on an algebraic simplifier that can solve equations and inequalities involving magnitudes of terms. Many simple-looking equations, however, are beyond the capability of a commercial computer algebra system like Mathematica. For example, Mathematica 2.2 `Solve` procedure will not be able to solve x in term of a in a simple equation:

$$\log(x+1) + \log(x-1) = a$$

To determine maximal terms, the program frequently needs to test inequality like this:

$$\hat{\sigma} Re^{2\alpha} Re^{-1} < \hat{\sigma} Re^{\alpha}$$

where $0 < \sigma < 1, Re > 1$, and $0 < \alpha < 1/2$.

Our algebraic simplifier borrows two ideas from previous AI works: (1) Bundy and Wellham's meta-rules for equation solving [6], and (2) Bledsoe-Brooks-Sacks sup-inf method to determine the upper and lower bounds for algebraic expressions [17].

The meta-rule approach organizes algebraic rewrite rules into packets. Each packet performs a particular function on an expression. For instance, one has an `isolate` packet whose job is

to isolate an unknown from an equation containing a single occurrence of the unknown. A `collect` and an `attract` packet reduce the number of occurrences of unknowns to one so that the `isolate` packet can apply. Our simplifier has three rule sets: (1) meta-rules for equation and inequality solving, (2) algebraic rewrite rules for standard simplification, and (3) rules for implementing the sup-inf algorithm. A significant portion of the rules handle exponentiation, maximum/minimum, derivatives, and inequalities.

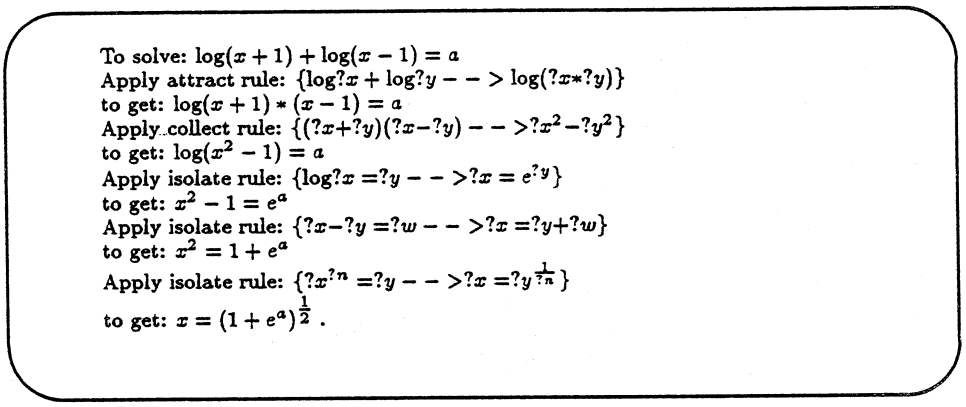


Figure 14: Repeatedly apply meta rules to the equation to solve for x.

Geometric interpretation of dynamics knowledge facilitates comparison with numerical data.

Chaotic dynamics is complicated. Even rough method to estimate the transition to chaos is extremely useful. Chirikov's Resonance Overlap Criterion is one such method: it postulates that the last invariant curve between two lowest-order resonances is destroyed when the sum of the half-widths of the two resonance zones just equals the distance between the resonance centers (Fig. 15).

The resonance overlap criterion can give us an analytical expression predicting when large-scale chaos occurs. However, it is much more illuminating to apply the method geometrically. A **resonance overlap diagram** is a graphical device for performing the computation geometrically.

A resonance overlap diagram has four parts: (1) the center and boundaries of the first resonance zone, (2) the center and boundaries of the second resonance zone, (3) the region where the two resonances overlap, and (4) level curves of the zero-order hamiltonian H_0. These four parts are illustrated in Fig. 16 for the case when the detuning frequency λ is 0.1.

Each resonance zone is represented by a trio of lines: the center line is the set of locations of the resonance center, and the two branches are the boundaries of the resonance zone. The

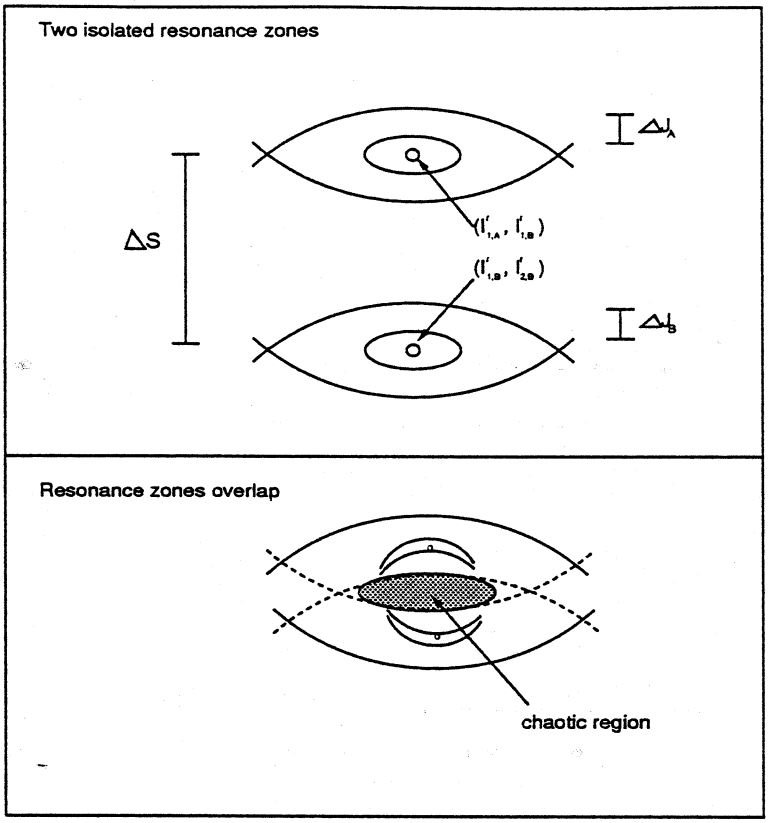


Figure 15: Graphical illustration of the resonance overlap criterion. When the separation ΔS between the two resonance centers $(I^r_{1,A}, I^r_{2,A})$ and $(I^r_{1,B}, I^r_{2,B})$ is less than the sum of the half resonance widths $\Delta J_A + \Delta J_B$, chaotic orbits appear in the overlap area (shown as the dotted region).

two resonance zones are displayed in Fig. 16a and 16b. Superposing these two zones, we obtain the overlap region as shown in Fig.16c. The level curves of the hamiltonian turn out to be parabolas, and they are presented in Fig. 16d. Putting all these parts together, we get the resonance overlap diagram (Fig. 16e).

The resonance overlap diagram contains important information about the behavior of the system; for example, the critical energy levels and the extent of chaotic regions can be read off from the diagram. Borrowing techniques from computational geometry [16], such as constructing convex hull, finding intersection of polygons, and determining lines intersecting a convex polygons, the program automatically constructs the resonance overlap diagrams and extracts information from them by using several well-known geometric algorithms. The main idea is to approximate each resonance zone by a convex polygon and represent the overlap region as the intersection of two convex polygons. From such representation, it is easy to deduce the critical

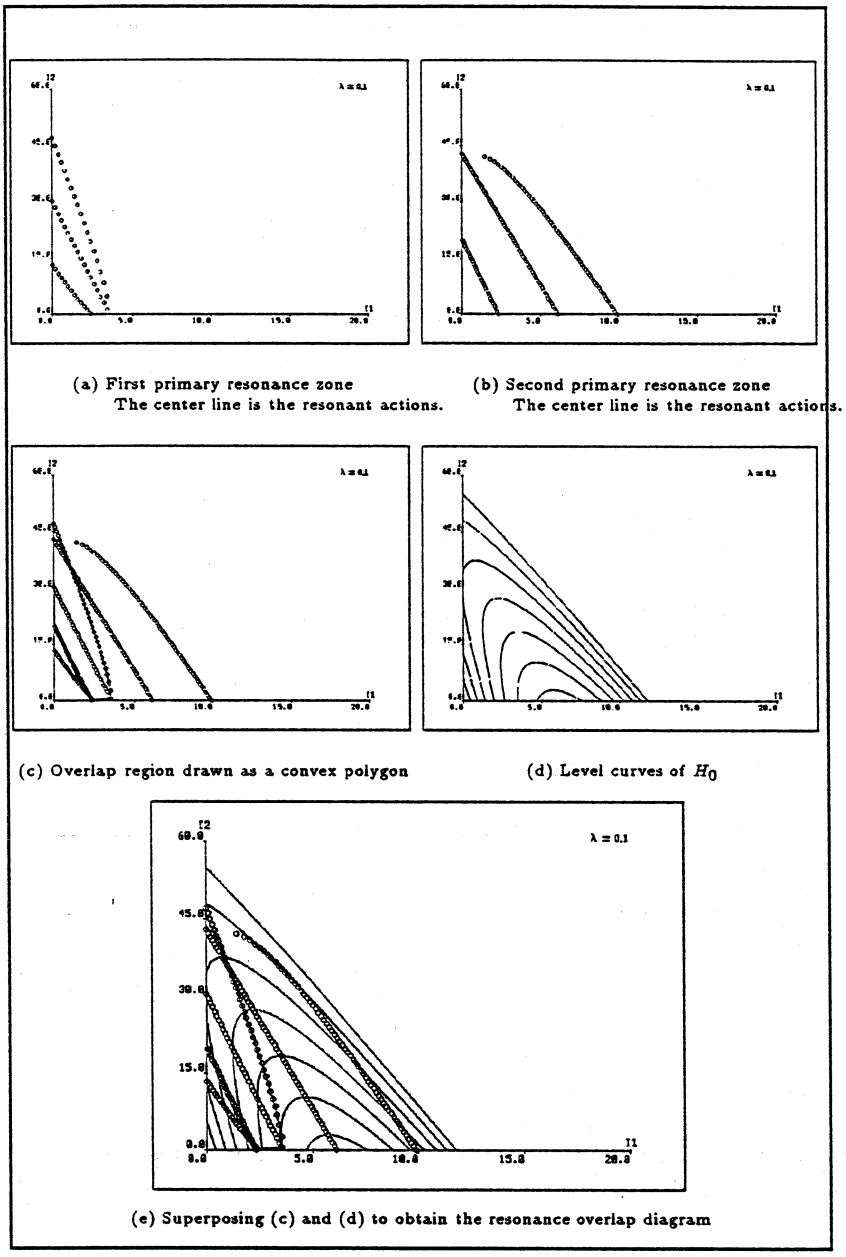


Figure 16: Resonance overlap diagram for detuning frequency $\lambda = 0.1$. The axes of the diagrams are the action (canonical) variables.

energy levels and the size of overlap region.

Let's see how the resonance overlap diagram explains the banded phenomenon observed in the numerical experiments. If we put the level curves of the hamiltonian (Fig. 16d) and the overlap region (Fig. 16c) together, something interesting happens (see Fig. 11). For small hamiltonian values (say, $E < 2$), we see that the level curves do NOT intersect the resonance

overlap zone. This suggests that isolated resonance zones dominate at low energies, and so we should see only regular motion. When E is approximately 2.5, the level curves just touch the edges of the overlap zone, indicating the onset of chaotic motion. As E is increased, the level curves sweep across the interior of the overlap zone. We therefore expect widespread chaos in the Poincaré section for these values of the hamiltonian. For larger values of E (say, $E > 5.9$), the level curves no longer intersect the overlap region. This predicts that regular motion becomes predominant again in the phase space in this regime of the hamiltonian. Comparing the predictions from the resonance overlap diagram with the numerical results in Fig. 10, we find that the resonance overlap criterion gives results that are in fairly accurate agreement with those obtained from the numerical experiments.

Geometric representation of analytical properties guides the search for integration contour.

The basic idea to estimate the order of magnitude of an integral is to find the region of principal contribution to the integral. If the region is localized, which will be the case if the integrand looks like a sharp gaussian $e^{-\lambda x^2}$, the integral can be approximated by the area of a local region around the peak. The approximation gets increasingly accurate as λ increases.

Consider the class of integral:

$$I(\lambda) = \int_C g(z) e^{\lambda h(z)} dz \tag{2}$$

where C is a contour in the complex plane, and $g(z)$, and $h(z)$ are analytic functions independent of λ.

In general the function $h(z)$ will not be sharply peaked along a given contour C. However, we can deform the contour in the complex plane, thanks to Cauchy Integral Theorem, without changing the value of the integral. Of all the possible contours, the one passing through a saddle point, say z_0, in the steepest descent direction of the function has a special property, namely, around z_0, the function would look like a sharp gaussian. This property follows from the Taylor expansion of $h(z)$:

$$h(z) = h(z_0) + h'(z_0)(z - z_0) + \tfrac{1}{2} h''(z_0)(z - z_0)^2 + \ldots$$

where $h'(z_0)$ vanishes at the saddle point z_0.

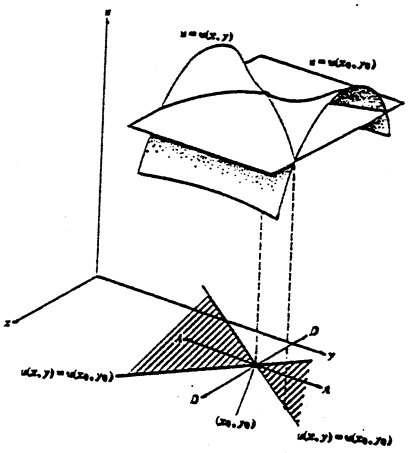


Figure 17: The surface of u(x,y) near a simple saddle point (x_0, y_0) The steepest descent paths are marked D, and the steepest ascent paths marked A. Note the alternating hills and valleys around the saddle point.

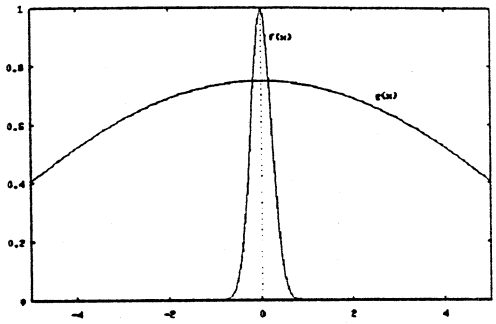


Figure 18: Laplace's idea of approximation. The function f(x) is sharply peaked, and g(x) can be well approximated by its value at the maximum of f(x).

The topography of an analytic function is exceedingly simple: it can't have peaks or pits, and around any saddle point there are alternating hills and valleys. Furthermore, the valleys of two distinct saddle points can't overlap partially (a consequence of the Maximum Modulus Theorem): either they are disjoint or one is completely included in the other. We will exploit these properties to come up with a simple graphical representation of the topography of $u(z)$, the real part of $h(z)$, and reduce the problem of contour selection to a shortest-path graph search.

Informally, to deform the contour we find the valley where C begins and run the path over a saddle point along the steepest paths into another valley. If C ends at this valley, then the process is done; otherwise, run the path over another saddle point into a third valley. We repeat the process until we find the valley where C ends.

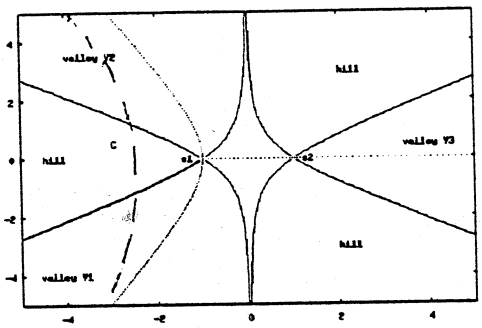
Since a valley may belong to more than one saddle point, we need a strategy to decide which saddle point to pick. Let's define some concepts. Given two simple saddle points s_1 and s_2, we say s_1 **dominates** s_2 if $s_2 \in V(s_1)$, where $V(s_1)$ is a valley of s_1. We say s_1 **immediately dominates** s_2 if there is no other saddle point s_3 such that s_1 dominates s_3 and s_3 dominates s_2. The saddle point s_1 **strictly dominates** s_2 if s_2 lies on a path of steepest descent emanating from s_1. The **admissible region** of $V(s_1)$ is defined to be:

$$AR(V(s_1)) = \begin{cases} V(s_1) & \text{if no other saddle point} \in V(s_1) \\ s_2 & \text{if } s_2 \in SDPATH(s_1, V(s_1)) \\ V(s_1) \cap AR(V(s_2)) & \text{if } SDPATH(s_1, V(s_1)) \cap V(s_2) \neq \phi \end{cases}$$

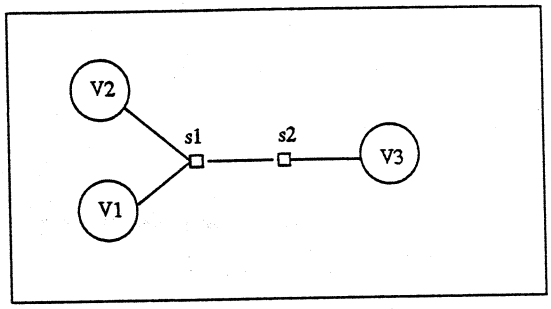
where $SDPATH(s_1, V(s_1))$ denotes the path of steepest descent from s_1 running down the valley $V(s_1)$.

Using these concepts, we can define a graphical representation for the topography of $u(z)$. The graph consists of two types of node and one type of edge. Each node represents an admissible region; it can be either a saddle point or a valley. Let's call the first type of node, S-node, and the second V-node. Each edge represents a steepest descent path from some saddle

point. Two S-nodes are connected if one is on the steepest descent path of another. A S-node is connected to a V-node if the steepest descent path lies in the valley represented by the V-node. See Fig. 19(ii). Now we can formulate our informal contour selection strategy more formally.



19(i): The topography of u(x,y) for $\lambda > 0$. The dotted lines are steepest descent paths through the saddle points s_1 and s_2. The original contour C marked by the dashed line is seen on the far left. The remaining solid lines are the boundaries of valleys and hills. Note that s_1 is strictly dominated by s_2.



19(ii): The topography graph corresponding to that in Fig. 19(i).

Contour Selection Heuristic: Given an integral $\int_C g(z)e^{\lambda h(z)}dz$ and a contour C with endpoints in valleys V_1 and V_2, deform it to a new contour represented by the shortest path connecting V_1 and V_2 in the topography graph of the real part of $h(z)$.

Let's see how this heuristic is used. For $\lambda > 0$, i.e., $phase(\lambda) = 0$, the shortest path between V_1 and V_2 is through the saddle point s_1. So s_1 is the only relevant saddle point. For $phase(\lambda) = \frac{\pi}{2}$ the shortest path still passes through only s_1 even the structure of the graph has changed. For $\lambda < 0$, the topography graph changes again, and the shortest path connecting V_1 and V_2 runs over both saddle points; so they both contribute to the integral.

To track the valley boundaries and the steepest descent paths of a saddle point, we use a numerical continuation algorithm. Let $h(z) = u(x,y) + iv(x,y)$. The valley boundaries are the constant u-lines and the steepest descent paths are the constant v-lines. Tracking a curve given by an implicit equation $u(x,y) = c$, where c is a constant, can be tricky because the curve can hit a turning point, or a bifurcation point, where the curve is split into multiple branches. The algorithm makes uses of analytical information about the saddle point to guide the numerical continuation (Fig. 20).

Diagrammatic representation of integrals short-circuits combinatorial arguments.

Consider acting the differential operator $\frac{1}{3!}(\frac{\partial}{\partial j})^3$ on the expression $e^{\frac{j^2}{2}}$:

$$\frac{1}{3!}(\frac{\partial}{\partial j})^3 e^{\frac{j^2}{2}} = \frac{1}{2}\, j e^{\frac{j^2}{2}} + \frac{1}{6}\, j^3 e^{\frac{j^2}{2}}$$

The procedure, `track-valley-boundaries`, computes four constant u-lines emanating from s. The main steps of the algorithm are:
1. Compute u(x,y), the real part of h(z), symbolically.
2. Evaluate u(x,y) at s to get the height d.
3. Compute the directions of steepest descent and ascent (by computing the second derivative of h(z) symbolically and evaluating it at s to get the phase).
4. Compute four starting points by taking a step of size $\frac{h}{2}$ in each of the four steepest directions.
5. Call *continuation* on each starting point.

The procedure, `continuation`, procedure returns a path on the constant u-line starting from (x_0, y_0). The algorithm has 4 steps:
1. Apply a corrector to the start point to get a more accurate initial point on the curve.
2. Apply a predictor to the current point to get a guess for the next point on the curve.
3. Apply a corrector to the predicted point.
4. Repeat steps 2 and 3 until the curve either exceeds the upper or lower bounds of x and y, or hits a special terminating point.

The predictor uses arc-length continuation to get past turning points [2]. It solves a set of two equations for two unknowns $\frac{dx}{dz}$ and $\frac{dy}{dz}$, where s is the arc length:

$$\frac{du}{ds} = \frac{\partial u}{\partial x}\frac{dx}{ds} + \frac{\partial u}{\partial y}\frac{dy}{ds}$$
$$(\frac{dx}{ds})^2 + (\frac{dy}{ds})^2 = 1$$

Once the numerical values of $\frac{dx}{ds}$ and $\frac{dy}{ds}$ are found, an explicit multi-step 4th-order Adams-Bashforth is used to solve for the increments x and y [10].

The corrector is Newton-Raphson with a small bound on the number of iterations allowed.

Figure 20: Details of numerical procedures to track valley boundaries and steepest descent paths.

There is a nice interpretation of the operator action in terms of diagrams: the differential operator is represented by a vertex with 3 legs. The two type of terms on the right hand side of the expression correspond to two topologically distinct arrangement of the 3-vertex. The 3-vertex with 3 hanging legs represents the term with the factor j^3 and the one with two legs tied in a loop (a tadpole as it is sometimes called) represents the term with the factor j. See Fig. 21. To account for the coefficient, we calculate the **symmetry factor** σ of a diagram, which is defined as the order of the automorphism group preserving the topological structure of the diagram. The coefficient of the term is then given by the formula $\frac{1}{\sigma}$. For the 3-vertex with 3 legs, the legs can be permuted in 3! ways without changing the topology. So $\sigma = 6$ and its coefficient is $\frac{1}{6}$. To try this out on the tadpole, we find there are only two ways to permute its legs (namely, the identity and the interchange of the two legs of the loop) without changing its topology. Therefore its symmetry factor is 2 and the corresponding coefficient is $\frac{1}{2}$.

Operators can be multiplied. In terms of diagrams, multiplication means tying legs of vertexes together in all possible ways. For example, the operator $\frac{1}{3!}(\frac{\partial}{\partial j})^3 \frac{1}{3!}(\frac{\partial}{\partial j})^3$ corresponds to

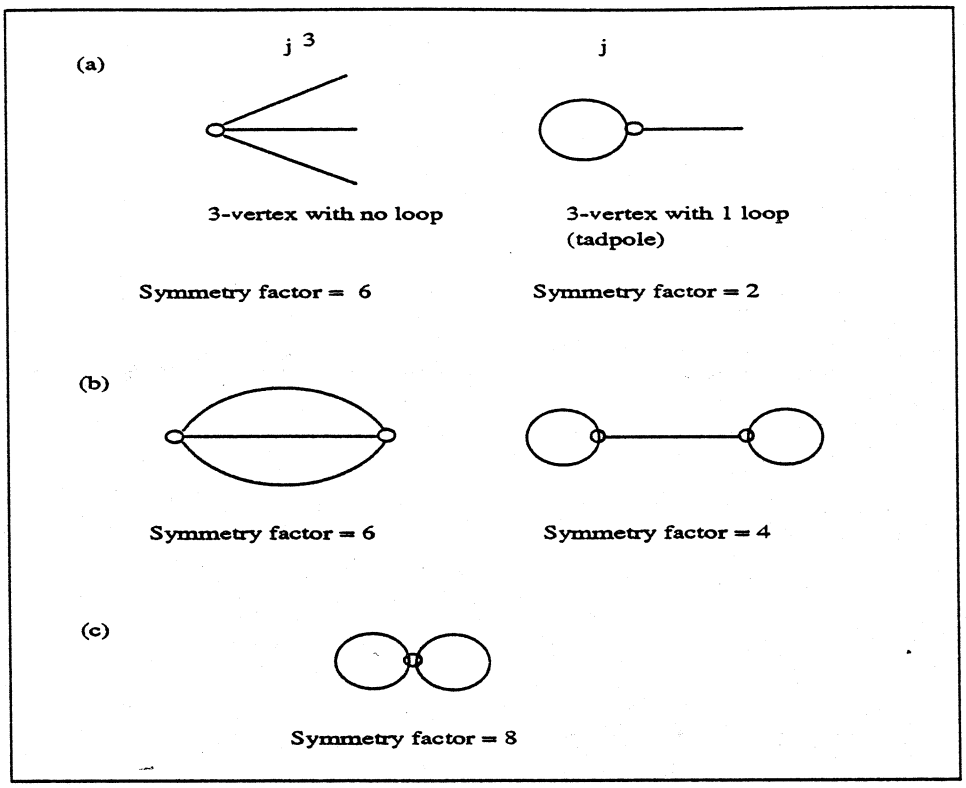


Figure 21: Diagrammatic representation of the action of differential operators. An n-vertex corresponds to $\frac{1}{n!}(\frac{\partial}{\partial j})^n$. Its action is represented by all topologically distinct n-vertex diagrams. Multiplication of operators corresponds to joining vertexes. Diagrams with hanging legs contribute zero when j is set to 0. In (b) and (c) we show the diagrams for $\frac{1}{3!}(\frac{\partial}{\partial j})^3 \frac{1}{3!}(\frac{\partial}{\partial j})^3$ and $\frac{1}{4!}(\frac{\partial}{\partial j})^4$ with non-zero contributions.

the set of topologically diagrams obtained by joining two 3-vertexes in all possible manner. It turns out that diagrams with no free hanging legs will be the ones that give non-zero contribution (because later we will set j to 0). Only two such diagrams are possible for the operator $\frac{1}{3!}(\frac{\partial}{\partial j})^3 \frac{1}{3!}(\frac{\partial}{\partial j})^3$.

One use of the diagrammatic representations is to calculate the higher order terms in the steepest descent approximation of an integral. The leading order approximation only retains the first two non-zero terms in the Taylor expansion of the exponent $h(z)$. To get higher order approximations, the whole Taylor expansion is kept. For simplicity, we will choose $g(z) = 1$. The diagrammatic representation can be extended easily to handle any analytic function $g(z)$. After a simple change of variable, the integral $I(\lambda)$ can be written as:

$$I(\lambda) = \int_{-\infty}^{\infty} \frac{dt}{2\pi} e^{-\frac{1}{2}t^2} e^{\sum_{n\geq 3} \frac{c_n}{n!} t^n}$$

where c_n is a function of λ and derivatives of $h(z)$. Using the identity:

$$\int_{-\infty}^{\infty} \frac{dt}{2\pi} e^{-\frac{1}{2}t^2+jt} = e^{\frac{j^2}{2}}$$

we express $I(\lambda)$ as an infinite series of differential operators acting on $e^{\frac{j^2}{2}}$. The details are given in Fig. 22.

$$\begin{aligned} I(\lambda) &= e^{\sum_{n\geq 3} \frac{c_n}{n!}(\frac{\partial}{\partial j})^n} \left(\int_{-\infty}^{\infty} \frac{dt}{2\pi} e^{-\frac{1}{2}t^2+jt}\right)|_{j=0} \\ &= e^{\sum_{n\geq 3} \frac{c_n}{n!}(\frac{\partial}{\partial j})^n} (e^{\frac{j^2}{2}})|_{j=0} \\ &= (1+(c_3\tfrac{1}{3!}(\tfrac{\partial}{\partial j})^3 + c_4\tfrac{1}{4!}(\tfrac{\partial}{\partial j})^4 + \cdots) + \frac{1}{2!}(c_3\tfrac{1}{3!}(\tfrac{\partial}{\partial j})^3 + c_4\tfrac{1}{4!}(\tfrac{\partial}{\partial j})^4 + \cdots)^2 + \cdots)(e^{\frac{j^2}{2}})|_{j=0} \end{aligned}$$

The $O(\frac{1}{\lambda})$ term is given by:

$$\tfrac{1}{2}c_3^2(\tfrac{1}{6}+\tfrac{1}{4}) + c_4(\tfrac{1}{8})$$

where the numerical factors inside the parentheses are obtained from the symmetry factors of the diagrams in Fig. 21b and 21c.

Figure 22: Calculation of higher order terms can be reduced to constructing diagrams and counting their symmetry factors.

Symmetry Rules to determine the contribution of a diagram.

1. For every loop around a vertex, there is a factor $\frac{1}{2}$.
2. For every pair vertexes joined by n links, there is a factor of $\frac{1}{n!}$.
3. If there are n different ways of permuting the links between two vertexes, keeping the topology unchanged and without cutting any links, there is a factor of $\frac{1}{n!}$.

Figure 23: The contribution of a diagram is equal to $\frac{1}{\sigma}$ where σ is the symmetry factor of the diagram.

To facilitate the computation of symmetry factors, we explicitly represent the number of loops and the multiplicities of links in the symbolic representation of vertexes and links. For example, the first diagram of Fig. 21b has one link triply connecting the two 3-vertexes; its symmetry factor is therefore 3!. The diagram on the right has one link singly connecting two tadpoles and each tadpole has one loop; its symmetry factor is $2 \times 2 = 4$.

Using combinatorial algorithms to compute partitions and k-compositions of an integer, the procedure for generating diagrams has three steps:

1. Compute the number of copies c of m-vertexes needed to make up an order n contribution.

2. Compute the possible vertex compositions making up the c copies.

3. Start with an empty diagram. Add one vertex at a time in all possible ways consistent with generating diagrams with no legs.

As an example, consider the constructing an order 1 diagram. There are two ways to make up an order 1 contribution: two copies of 3-vertex, and 1 copy of 4-vertex. Since there are two types of 3-vertex (3-vertex with no loop, 3-vertex with 1 loop), and three types of 4-vertex (4-vertex with no loop, 4-vertex with 1 loop, and 4-vertex wit 2 loops), six vertex compositions are possible of which three lead to diagrams with no legs hanging (Fig. 21c).

Topological properties of diagrams facilitate renormalization.

The calculation of the partition function Z (Fig. 7) is a fundamental problem in statistical mechanics. We'll see that a diagrammatic representation for the partition function is key to the renormalization process because we can exploit powerful theorems in graph theory and the topological properties of diagrams to simplify and organize computations.

According to the problem specification (Fig. 7), we need to calculate the configuration integral:

$$Q = \int e^{-\lambda \frac{\phi}{kT}} dV_1 \ldots dV_n$$

where the integration is taken over the configuration space, the 3N-dimensional of positions of the N particles. The small parameter λ indicates that the potential is a small deviation from the ideal gas case. Assuming the potential is made up of pairwise intermolecular forces,

$$\phi = \sum_{pairs\ i,j} u(r_{ij})$$

we expand the configuration integral as:

$$\begin{aligned} Q &= \frac{1}{V^N} \int \sum_{n=0}^{\infty} \frac{1}{n!} (-\frac{\lambda}{kT})^n (\sum_{pairs i,j} u(r_{ij}))^n dV_1 \ldots dV_n \\ &= \sum_{n=0}^{\infty} \frac{1}{n!} (-\frac{\lambda}{kT})^n Q_n \end{aligned}$$

where $Q_n = \int u(r_{ij}))^n dV_1 \ldots dV_n$.

There is a nice diagrammatic way to visualize this complicated infinite series of integrals. For order n = 1, there is only one type of integral of the form $\int u_{12}$. The actual indices do not matter because they are dummy integration variables. We associate the integral with a diagram with two vertexes and one link. For order 2, three types of integrals are possible: (1)

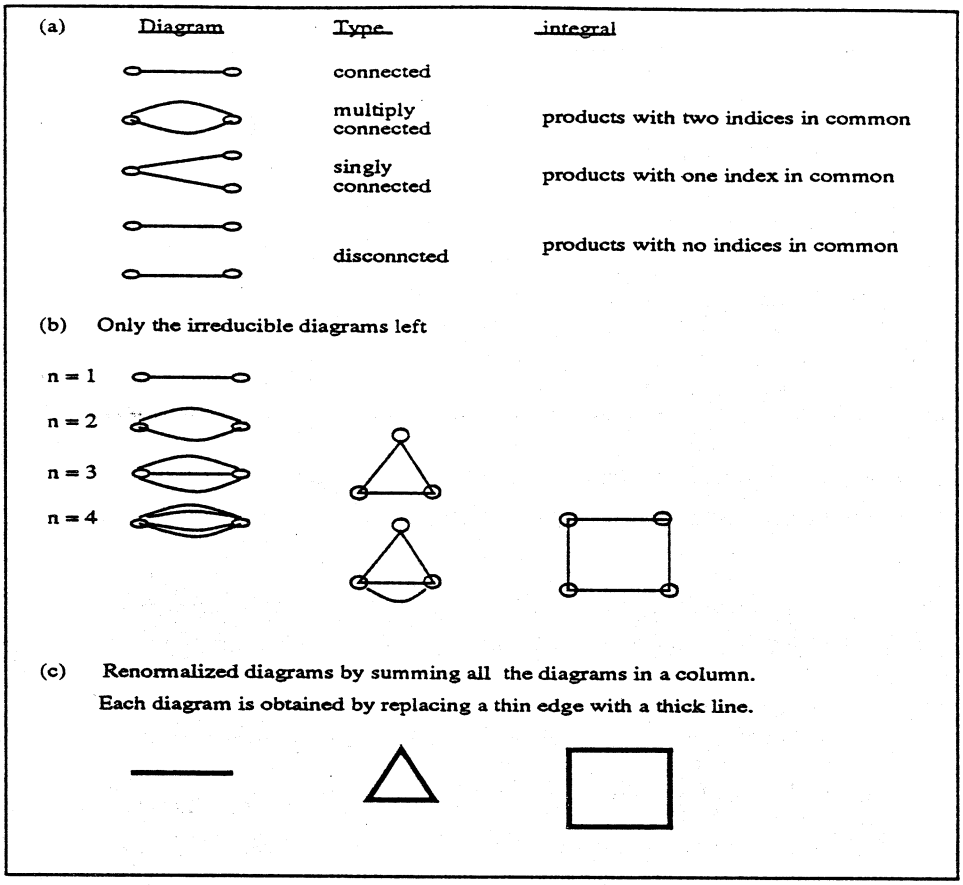


Figure 24: Diagrammatic representation of items in the expansion of configuration integral. (a) All possible diagrams up to order $n = 2$. (b) Only the irreducible diagrams have non-zero contribution. Diagrams up to order $n = 4$ are shown. (c) Renormalization amounts to partial summation of infinite series of diagrams. For example, all diagrams with two particles interacting correspond to one renormalized 2-particle diagram, one with a thick line. Renormalization reclassifies the diagrams according to columns.

$\int u_{12}^2$, (2) $\int u_{12}u_{23}$, and (3) $\int u_{12}u_{34}$. See Fig. 24a. The number of diagrams grows rapidly as n increases.

The advantage of a graphical representation is that now we can employ powerful counting theorems from graph theory. For instance, the exponentiation theorem (Fig. 25) allows us to ignore all the disconnected diagrams if we consider the logarithm of the configuration integral. In the thermodynamic limit, i.e., $N \to \infty$, an order of magnitude estimate allows us to further eliminate all the singly connected diagrams. The remaining multiply connected diagrams are called irreducible cluster integrals.

Fig. 24b shows all the irreducible diagrams up to order n = 4. These diagrams can be classified into diagrams containing k vertexes. Note that the classification effectively reorders

Exponentiation Theorem for linear graphs [22].

With each graph G_n, define a weight $w(G_n)$ satisfying the two properties:

1. $w(G_n)$ is independent of the labeling of the points.

2. For a disconnected graph, G_n consisting of connected parts C_p such that: $w(G_n) = \prod_{C_p} w(C_p)$.

Let $F_n = \sum_{G_n} w(G_n)$ over all graphs of n points and $f_n = \sum_{C_p} w(C_p)$ over all *connected* graphs of p points. Define the generating function for all graphs: $F(x) = \sum_{n=1}^{\infty} F_n \frac{x^n}{n!}$ and the generating function for all connected graphs: $f(x) = \sum_{n=1}^{\infty} f_n \frac{x^n}{n!}$.

THEN:

$$1 + F(x) = e^{f(x)}$$

Figure 25: The exponentiation theorem for linear graphs allows us to drop all the integrals corresponding to disconnected graphs in the calculation of the configuration integral.

the expansion series in a different parameter. Inspection of the infinite series of integrals corresponding to the simplest diagrams, clusters of two particles, shows that it can be summed exactly.

$$\int dV_1 dV_2 [-\frac{\lambda}{kT} u_{12} + \frac{1}{2!}(\frac{\lambda}{kT} u_{12})^2 + \cdots] = \int dV_1 dV_2 f_{12}$$

where $f_{12} = e^{-\frac{\lambda}{kT} u_{12}} - 1$. We may therefore replace each the infinite series of diagrams by a single diagram with a new meaning assigned to the link (Fig. 24c). This process of reinterpretation of a diagram, as a result of partial summation, is the pictorial way of representing the renormalization process.

It can be shown that the coefficient, $B(T)$, in the equation of state is equal to the integral of the renormalized irreducible 2-particle diagram. The conclusion that classes of summable diagrams can correspond to meaningful physical quantities is quite general. The prospect of using this correspondence as an automatic generator of rational approximate models remains to be explored.

Conclusion

The main conclusion I want to draw is that AI techniques, built on top of conventional symbolic and numerical computation platforms, are sufficient to implement programs embodying heuristic and qualitative methods to solve nontrivial physical problems. Several computer programs with similar intent already exist in different application areas [8]: automatic phase space analysis for dynamical systems [18, 24, 25, 15], nonlinear control [5], and kinetics [9], just to name a few.

Sophisticated physical reasoning depends on explicit, qualitative understanding of physics.

In observing how these programs work, we notice that they derive their power from clever use of knowledge and choice of representation. That knowledge is power is hardly a new theme – some AI experts have advocated this for years [11]. Expertise is not a mere accumulation of a vast amount of facts, theorems, and laws. Like professionals, our computer programs know how to use what they know: they use order of magnitude estimates to simplify equations, they use structure theorems to guide search in phase space, they use resonance criterion to explain numerical findings, they use analytical properties of complex plane to search for integration contours, and they use diagrams to shortcut combinatorial arguments and organize algebraic computations. In each case, the power lies not so much in sophisticated logical inference techniques, but in the appropriate choice of representation and constraints.

I expect the future of the development of intelligent programs for solving scientific problems will be influenced by two factors. First are the quality and variety of computational tools equipped with an explicit, qualitative understanding of the relevant physics and mathematical principles so that they know their own applicability, they interpret their output, and they control their own actions based on interpretation of partial results. Second is the ability to coordinate the use of a large number of tools, to perform "sanity" checks on their output, and to provide symbolic feedback to guide further experimentation of each individual tool.

Computers should learn to do what people are good at.

Much research in symbolic and numerical computation is motivated by the desire to automate calculations that either are too tedious for people to do by hand, but would be more accurately and quickly done by computer, or could not be done by people at all. What we are suggesting is that we may need to take the "bag of tricks" of professionals seriously. In order to give advice, plan and monitor computations, collect and interpret data, intelligent computer assistants to people would need broad knowledge of problem solving methods and specific domain knowledge. Identifying these methods and articulating them explicitly would go a long way towards understanding some of the core skills that make up a theoretical scientist.

References

[1] H. Abelson, M. Eisenberg, M. Halfant, M. Katzenelson, E. Sacks, G. Sussman, J. Wisdom, and K. Yip. Intelligence in scientific computing. *Communications of the ACM*, 32(5), 1989.

[2] Eugene Allgower and Kurt Georg. *Numerical Continuation Methods.* Springer-Verlag, 1990.

[3] Radu Balescu. *Equilibrium and nonequilibrium statistical mechanics.* Wiley, 1975.

[4] Carl Bender and Steven Orszag. *Advanced Mathematical Methods for Scientists and Engineers.* McGraw-Hill, 1978.

[5] E Bradley and F Zhao. Phase space control system design. *IEEE Control Systems Magazine*, 13, 1993.

[6] Alan Bundy. *The computer modelling of mathematical reasoning.* Academic Press, 1983.

[7] John Campbell, Per Olof Froman, and Erik Walles. Explicit series formulae for the evaluation of integrals by the method of steepest descent. *Studies in Applied Mathematics*, 77, 1987.

[8] Johan de Kleer and Brian Williams, editors. *Qualitative reasoning about physical systems II*, volume 51. Artificial Intelligence Journal, 1991.

[9] Michael Eisenberg. *A Kineticist's Workbench.* PhD thesis, MIT, 1989.

[10] David Kahaner, Cleve Moler, and Stephen Nash. *Numerical Methods and Software.* Prentice Hall, 1989.

[11] Douglas Lenat and Edward Feigenbaum. On the thresholds of knowledge. *Artificial Intelligence*, 47, 1991.

[12] J.D. Lin and L.N. Howard. Non-linear standing waves in rectangular tank due to forced oscillations. Hydrodynamics Laboratory Report 44, MIT, 1960.

[13] W.D. McComb. *Physics of Fluid Turbulence.* Oxford Univeristy Press., 1992. ISBN 0-19-856256-X (pbk.), $55.

[14] Chiang C. Mei. *The Applied Dynamics of Ocean Surface Waves.* World Scientific, 1989.

[15] Toyoaki Nishida, Kenji Mizutani, Atsushi Kubota, and Shuji Doshita. Automated phase portrait analysis by integrating qualitative and quantitative analysi. In *Proceedings AAAI-91*, 1991.

[16] Franco Preparata and Michael Shamos. *Computational Geometry.* Springer-Verlag, 1985.

[17] Elisha P. Sacks. Hierarchical reasoning about inequalities. In *Proceedings AAAI-87*, 1987.

[18] Elisha P. Sacks. Automatic analysis of one-parameter planar odes by intelligent numerical simulation. *Artificial Intelligence*, 48, 1991.

[19] F.T. Smith. On the high Reynolds number theory of laminar flows. *IMA Journal of Applied Mathematics*, 28, 1982.

[20] K Stewartson. Multistructured boundary layers on flat plates and relations bodies. *Advances in Applied Mechanics*, 14, 1974.

[21] Wu-ting Tsai, Dick Yue, and Kenneth Yip. Resonantly excited regular and chaotic motions in a rectangular wave tank. *Journal of Fluid Mechanics*, 216, July 1990.

[22] G.E. Uhlenbeck and G.W. Ford. The theory of linear graphs with applications to thetheory of the virial development of the properties of gases. In *Studies in Statistical Mechanics, Vol 2.* North-Holland, 1962.

[23] W.D. Wyld. Formulation of the theory of turbulence in an incompressible fluid. *Annals of Physics*, 14, 1961.

[24] Kenneth Yip. *KAM: A System for Intelligently Guiding Numerical Experimentation by Computer.* Artificial Intelligence Series. MIT Press, 1991.

[25] Feng Zhao. Extracting and representing qualitative behaviors of complex systems in phase spaces. In *Proceedings IJCAI-91.* International Joint Conference on Artificial Intelligence, 1991.