Journal of National Fisheries University 50 (1) 37−52 (2001)

気象観測データのWWW検索システムの構築について

楫取和明*1・瓜倉　茂*1・青木邦匡*1・川崎潤二*2

Constructing WWW Query Systems of Weather Observation Data

Kazuaki Kajitori*1, Shigeru Urikura*1, Kunimasa Aoki*1, and Junji Kawasaki*2

Two methods of constructing a WWW (World Wide Web) system that searches weather observation data are presented. One uses the Windows NT system and the other uses the Linux system. In these systems, the weather observation data are read via the campus LAN and stored in a database on the server. The HTML data based on the user's query and the established database are formed by the web scripts then sent to the browser.

1　まえがき

インターネット上を行き交うWWW(World Wide Web)情報は、一般に情報発信者が新たな情報で更新するまでは内容に変化がない、いわゆる静的なWWW情報が中心である。しかし、WWW技術の革新によりCGI(Common Gateway Interface)やASP(Active Server Pages)（たとえば、文献1),2)）などの技術を使って動的なWWWページが容易に作成できるようになってきた。たとえば、アクセス時刻に応じて表示する内容を変更したり、あるいはユーザとインタラクティブに情報をやり取りしながらWWWページを提供するなど、変化に富むWWWページの作成・提供も可能となってきている。

これら動的なWWWページの技術を支えるものの一つとして、データベースと連携させる利用法が確立してきたことがあり、多種多様な情報提供の基となっている。これはまた、ユーザ側では特にデータベース専用のクライアントソフトウェアを必要とせず、単にWWWブラウザによりデータベースの利用が出来る仕組みとなっている。

本報告では、本校に設置されている気象観測装置からのデータをWWW情報として提供するためのシステム構築法を紹介する。ユーザから要求される日付や時刻に対応する観測値のグラフやデータ表をインタラクティブに提供できるものである。システム構築は次の2つのOS環境、つまりWindows NT環境と、Linux環境とについて行われ、それぞれの特徴等を考察している。

本報告は、研究データ等の公開にWWWを利用する一つのケーススタディであり、実システムの構築を通して具体的な方法を提示し、WWWを使った公開システムの有効性を示すとともに、問題点を探ることを目的とするものである。

本報告の構成は次のようである。まず、2でシステム構成を紹介し、3でWindows NTサーバによるシステム構築を述べ、4でLinuxサーバによるシステム構築を述べる。5は考察で、6はあとがきである。

2　システム構成

今回考察の対象とした気象観測装置およびコンピュータネットワークとの接続形態は、Fig.1のようである。気象観測装置FWS-9800FD型（(株) 日本エレクトリックインスルメント）は、本校海洋生産管理学科舟艇管理棟に設置

2001年4月18日受付. Received Apr. 18, 2001.
＊1水産大学校水産情報経営学科(Department of Fisheries Information and Management, National Fisheries University).
＊2水産大学校海洋生産管理学科(Department of Fishery Science and Technology, National Fisheries University).

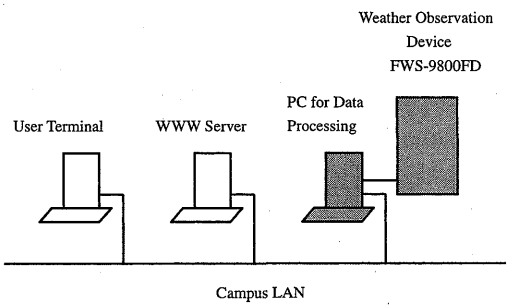

Fig. 1. Hardware topology of the WWW system and the weather observation system in the campus LAN.

されており、風力計、風向計、気温計、湿度計、気圧計で構成されている。これらの計器による観測値は装置に取り付けられた表示機に示されるが、同時にデータ処理パソコン（OS：Windows98）に電子化データとしてファイルで保存されている。保存される観測値は次の9項目、つまり平均風力、平均風向、瞬間風力、瞬間風向、気温、相対湿度、実効湿度、現地気圧、海面気圧である。

データ処理パソコン上での観測値の記録の仕方は次のようである。観測値は電子化データに変換され、1分間隔でファイルに追記され、1時間毎に1つのCSV形式のファイルとして保存される。これらのファイルは日毎のフォルダにまとめられ、日フォルダは月フォルダにまとめられ、月フォルダは西暦年フォルダにまとめられる。

データ処理パソコンは学内LANに接続されており、WWWサーバからネットワーク経由でディスクに保存された観測データにアクセスできるようになっている。

構築したWWW検索システムの概要を述べる。まず、定時処理で必要な時刻にネットワーク経由でデータ処理パソコンのファイルからデータを読み取り、WWWサーバ上のデータベースに蓄える。次に、ユーザからの検索要求に応じてこのデータベースからデータを読み出し、WWW形式のデータに処理・加工して提示する処理を行う。

3 Windows NT によるシステム構築

Windows NTによるWWW検索システム構築に当たっては、次のようなソフトウェアを利用した。

OS：Windows NT Server 4.0
Service Pack 5, Option Pack
WWW サーバ：Microsoft IIS(Internet Information Server) 4.0
スクリプト言語：VBScript 5.5
プログラミング言語：Microsoft Visual Basic 4.0
（以後、VBと略す）
データベース：Microsoft Access 97
スプレッドシート：Microsoft Excel 97
タスクスケジューラ：Multi Function Alarm 1.42[3]
通信プロトコル：NetBEUI（ファイルアクセス用）

気象観測値データのトップページは、Fig.2のようである。図中①の部分は定時処理によるグラフ表示、②の部分はユーザからの時刻指定によるグラフあるいは表の作成表示である。①および②のグラフは9つの項目の中から選ばれた観測値を対象とした折れ線表示である。また、表は9つの観測値項目の一覧表示となっている。

3.1 Windows NTにおける定時処理

WWWサーバ上で行われる定時処理は、次の2つから成る。

(1) データ処理パソコンのファイルにアクセスし、観測値データをサーバ上のデータベースに格納する処理（Table 1-1）。

(2) 最新データのグラフ作成（Table 1-2）。

これらの処理はいずれもVBにより作成したプログラムで行われ、タスクスケジューラにより5分毎に実行されるように設定されている。

Table 1-1は、データ処理パソコンのファイル読み込みから、WWWサーバ上のデータベースへの書き込みまでの処理を行うプログラムを抜き出したものである。データベース処理には、データベース操作用のライブラリの一つであるDAO(Data Access Object)ライブラリを利用している。関数firstcommaは、引数で与えられる文字列に含まれる最初のカンマの位置を求めるユーザ定義関数である。

次にTable 1-2は、最新のデータからグラフを作成するプログラムの部分である。前半はデータベースからデータを読み取りExcelの表を作成し、後半はExcelに内蔵されている描画ライブラリ(Microsoft Chart)を使って、グラフ（ここの例では平均風速）をGIF形式で作成している。

3.2 時刻指定によるグラフ作成

ここで紹介するのは、ユーザとインタラクティブに情報をやり取りする部分であり、ユーザが指定した日付、時刻等の情報に基づいて観測値のグラフを作成する仕組みであ

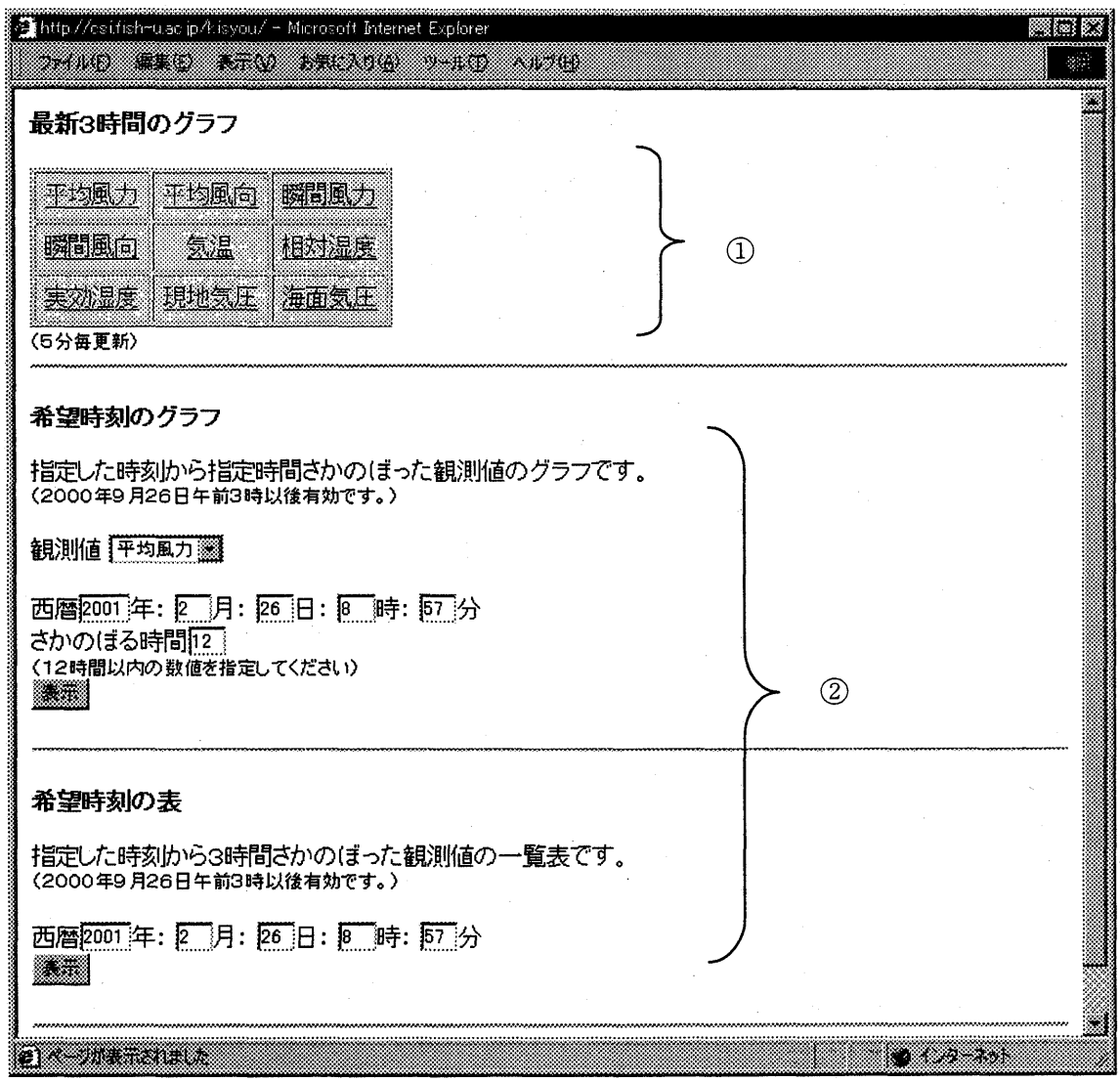

Fig. 2. The first WWW page of weather observation data under the Windows NT system.

る。要点は次の3つの処理からなる。

(1)ユーザによる指定時刻等の情報をファイルに保存する処理（Table 1-3)。

(2)グラフ作成プログラムを実行させる処理（Table 1-4)。

(3)時刻指定データのグラフ作成。

ユーザがWWWページ上で指定した日付や時刻等の情報は、ディスクに保存されたテキストファイルを経由して上記(3)のVBによるグラフ作成処理のプログラムに引き渡されるようになっている。

まず、Table 1-3はテキストファイルdatasave.txtを保存するスクリプトである。その内容は、観測値の種類、日付や時刻等である。このリストのように、ASPを使用する時には〈%と%〉で囲まれる部分にスクリプトが記述され、HTMLファイルの中に埋め込まれる。ASPはこのようなスクリプトを処理する技術のことである。特に指定がなければ、標準ではこれらのスクリプトはサーバ側で実行されることになっている。

次に、Table 1-4は、上記(3)のグラフ作成プログラムを起動させるスクリプトである。basp21[4]は多機能なDLL (Dynamic Link Library)ファイルであり、ここでは外部プログラムであるグラフ作成プログラムmake-graph.exeを実行させるために使われている。

最後に、VBによるグラフ作成プログラムmake-graphは、基本的なところはTable 1-2と同じであり、その主要部分をTable 1-5に示す。まず、テキストファイルdatasave.txtを読み込み、ユーザによる指定日付、時刻等の情報をもとにデータベースから必要なデータをExcelの表に移す。次に、Excelの描画ライブラリでグラフを作成し、GIF形式

のファイルとして保存する。

4 Linuxによるシステム構築

Linuxによる WWW 検索システム構築に当たっては、次のようなソフトウェアを利用した。これらはすべて無料で使えるものばかりから成っている。

OS：Linux2.2.16
WWW サーバ：Apache1.3.12
WWW スクリプト環境：mod_perl1.24
スクリプト言語：Perl5.005
グラフ描画ライブラリ：gnuplot3.7.1,GD1.8.3
データベース：MySQL3.22.32
タスクスケジューラ：cron
NETBUEI ライブラリ：Samba2.0.6

ここで、mod_perl は、Apache に Perl インタープリタを組み込んで使う環境を与えるモジュールであり、従来の CGI のように CGI リクエスト毎に Perl プロセスを立ち上げる無駄を回避できる。WWW サーバ Apache はこの mod_perl 付で構築したものである。

ソフトウェアの処理は大きく分けて２つの部分に分けられる。１つは定時にデータを取得してデータベースに格納すること、他の１つはユーザからの要求にしたがってグラフを作成し WWW ブラウザに送ることである。以下これらを順を追って説明する。

4．1 Linux における定時処理

Linux サーバ上に構築した気象観測値データベースのテーブルの定義は、次のようである。

データベース名：kisyou
データを格納するテーブル名：master
テーブル作成 SQL 文：

```
create table master（
date DATE not null, -- 年月日
hour int not null, -- 時
minute int not null, -- 分
winvel float, -- 瞬間風速
windir float, -- 瞬間風向
msinvel float, -- 平均風速
mwindir float, -- 平均風向
temp float, -- 気温
relhum float, -- 相対湿度
effhum float, -- 実効湿度
locatt float, -- 現地気圧
suratt float, -- 海面気圧
index（date),
unique（date,hour,minute）
)
```

データをデータ処理パソコンから読み取り、データベースに格納するのは、cron で毎時15分に getadd.pl という Perl スクリプトを動かすことによって行うようにした。たとえば、11時15分にこのスクリプトが動くと、10時台のデータが格納されるようになっている。

このスクリプトでは、データ処理パソコンからのデータ転送には smbclient という Samba のクライアントを使い、読み取ったデータファイルの１行ごとに個々のデータを取り出し、テーブル master に insert 文で挿入している。また、これらの作業が成功しているかどうかを見るためにログを残すようにしている。

getadd.pl のコードリストは省略するが、これを毎時15分に起動するには次の１行を crontab ファイルに登録すれば良い（コードリストについては本報告の最後に示した URL を参照されたい）。

```
15 * * * * /home/kisyou/getadd.pl〉〉
/home/kisyou/getadd.log 2〉 &1
```

4．2 時刻指定によるグラフ作成

Fig.3に示した WWW ブラウザ画面を通して送られてくる観測値の種類と年月日の情報に応じて、データベースを検索し、その日の午前０時からの観測値グラフを作成する。

まず、次の Table 2-1は観測値の種類と日時をキーとしてデータベースを検索する Perl スクリプトである。観測値の種類は、データベースのテーブルに登録した９項目に加えて平均風速と平均風向を組にしたもので、併せて10種類である。

グラフのタイプとしては、Fig.3の気温グラフのように横軸に時間、縦軸にデータ値をとって折れ線グラフで表すのが一般的と思われる。ただし、風向は１から16の整数で表されるので、Windows NT でグラフ化した際にそのまま折れ線グラフで表しても直観的に理解しがたいものであった。そこで、Linux による WWW 検索システムでは

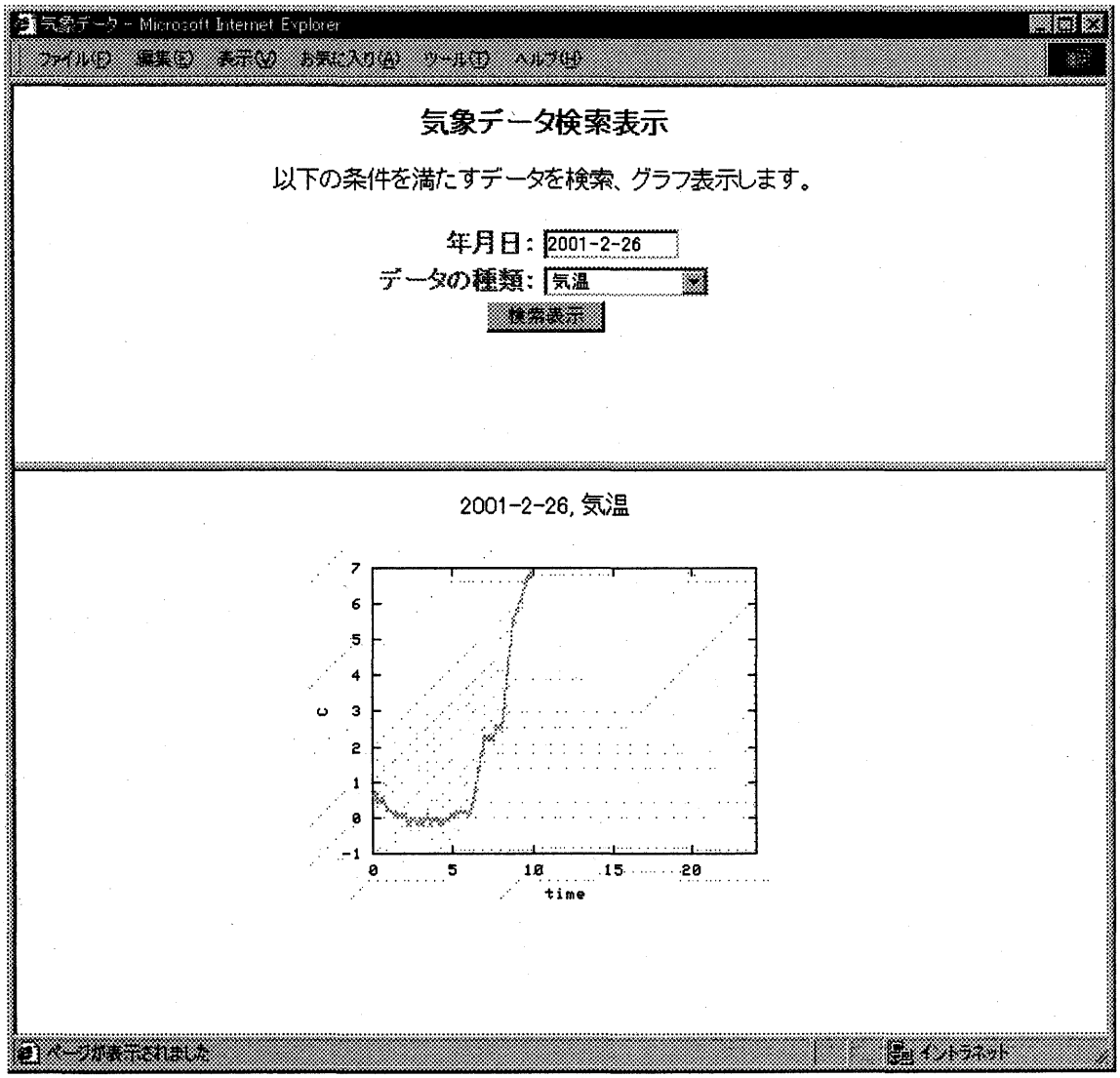

Fig. 3. The first WWW page of weather observation data under the Linux system.

平均風速と平均風向を対にした風のベクトルを極座標表示してプロットしてみた(Fig.4参照)。

次に、観測値グラフは、データベースから検索されたデータをグラフ作成スクリプトに渡し、グラフ描画ソフトgnuplotまたはグラフィックスライブラリGDを用いて生成されている。

グラフ作成スクリプトとして、GDの低レベル関数を使って平均風速と平均風向のグラフを生成する部分を含んだ、Perlスクリプトkisyou.plの概略をTable 2-2に挙げておく。ユーザは、まずkisyou.plにアクセスする。するとkisyou.plは検索キーを指定するためのページを返すから、ユーザは検索キーを指定して送り返す。すると、kisyou.plは今度は検索キーに応じたグラフを作成して返すようになっている。

5 考 察

5.1 Windows NTとLinuxにおけるシステム構築について

ここに示した、Windows NTにおけるシステム構築とLinuxにおけるそれとの違いは、一つにはスクリプトのスタイルにあって、前者では「HTMLの中にスクリプトを埋め込む」のに対し、後者では「スクリプトの中でHTML書き出しをしている」という違いがある。しかしこれはWindows NTとLinuxの本質的な違いとはいえない。例えば、現在ではJava servletを使えばどちらのスタイルもWindows NT、Linuxにおいて可能である。

Windows NTのASP環境においては、Table 1-4において使ったbasp21のようなActive Xコントロールと呼ばれ

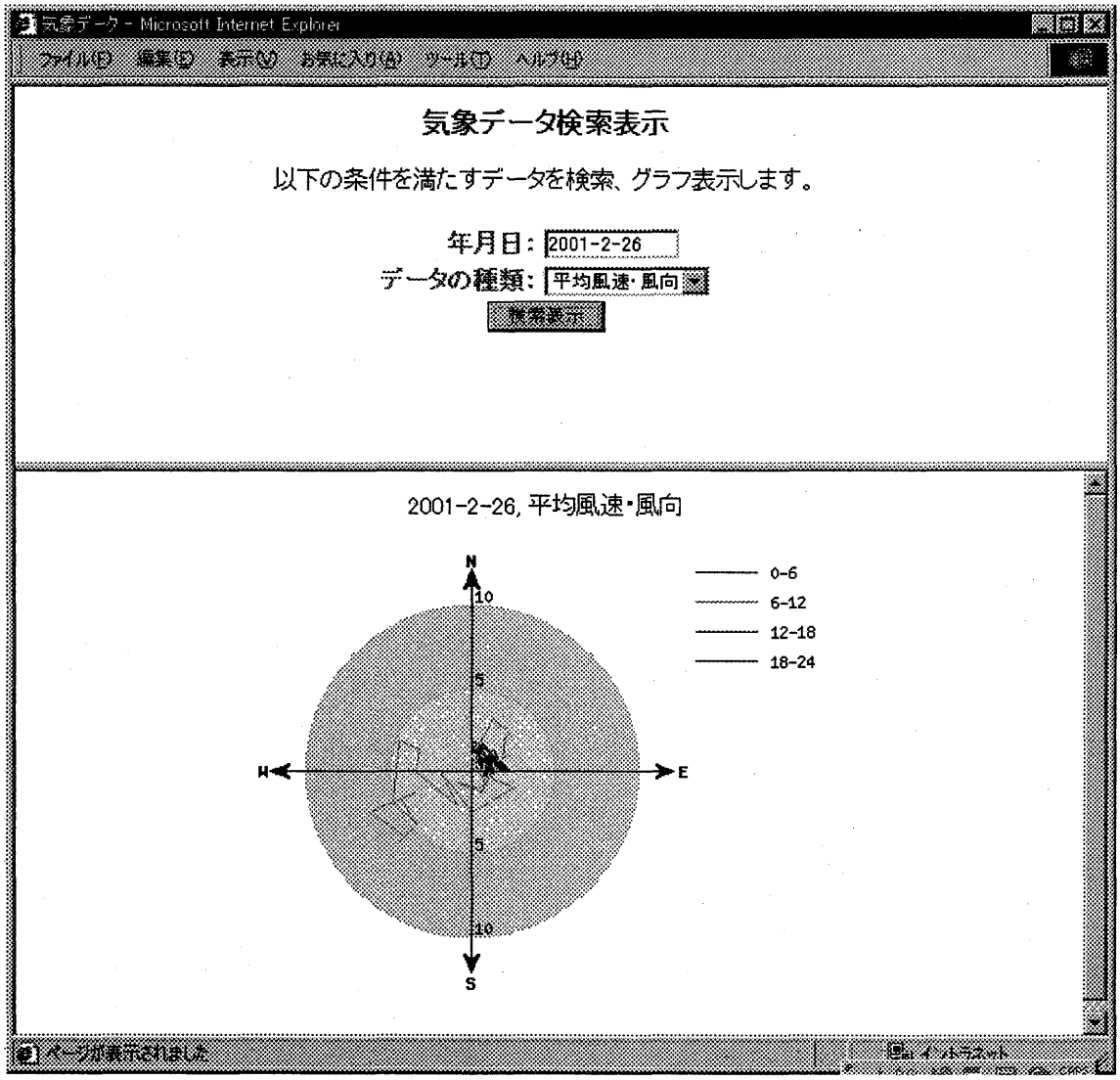

Fig. 4. Graph presentation of wind vectors with average wind velocity and average wind direction in polar coordinates.

るダイナミックリンクライブラリを使ったオブジェクト指向プログラミング環境が特徴である。グラフ生成を行うActive Xコントロールも商品として存在する。本論では身近なソフトで手軽に構築することを目指したため、これら商用のグラフ作成コントロールを用いた方法は示していない(ちなみにbasp21は無料である)。

WWWアプリケーションの一つの事例に過ぎない本論のシステム構築によって、Windows NTとLinuxにおけるWWW開発環境を一般的に比較するのは元々難しいが、Windows NTとLinuxにおけるWWW開発環境を比べるならWindows NTにおいては商用ソフトを十分考慮すべきであろう。

5．2　グラフの作成とWWWブラウザでの表示について

グラフ作成・表示の最も単純な方法は、グラフ描画ルーチンでグラフを一時ファイルに保存し、これをHTMLタグ〈IMG SRC=...〉でWWWブラウザに表示するやり方であろう。複数のリクエストがほぼ同時にきた場合に備えて一時ファイルの名前は疑似乱数を発生させるなどしてリクエストごとに変えるようにしておく必要がある。また、一時的にしか使わないとはいえファイルがそのまま残ってしまうので、ある時間後(WWWブラウザが読み終ったあと)自動的に削除することが望ましい。この方法は実現しやすいが、削除をスケジュールするのはプログラム全体のチェックをしにくくするのでできれば避けたい。

ちなみに、上記のHTMLタグ〈IMG SRC=...〉を使わ

ずに、HTMLヘッダにtext/htmlの代わりにimage/png等を指定して画像のバイナリを（ファイルに保存せずに）直接書き出す方法もある。しかし、この方法だとHTML文が書けないため画像以外の情報を書き送ることができなく、また画像のセンタリングなどのレイアウトもできないので不便である。よって、この方法は本論では用いていない。

LinuxでのGDライブラリを使うやり方では、Perlのスクリプトから呼び出したGDの低レベル関数を使って生成したグラフイメージを(ファイルに保存せずに)HTMLタグ〈IMG SRC=...〉を使って直接WWWブラウザに書き出すことができる（Table 2-2)。ただし汎用のグラフィックスライブラリの関数を使ってグラフを描くのは手軽とは言い兼ねる。

gnuplotは高機能なグラフ作成アプリケーションであるが、イメージを直接WWWブラウザに書き出す方法はプロセスを分岐するforkを使うなど複雑になるので、一時ファイルに保存してから変数に読み込んで書き出す方法をとった(これだと変数に読み込んだらすぐにファイルを消せるからである)。

上記Linuxの方法でファイルを送らないやり方では、WWWブラウザに送るHTMLタグ〈IMG SRC=...〉にトリックが必要である。なぜならファイルを指定するところは通常画像ファイルのURLであるが、ファイルは送らないからである。トリックとしてはTable 2-2のような方法がある。50行目の〈IMG SRC=...〉において、... の部分のパスがplotを含むのでprint_plotが実行され、その際print_plotに必要なデータが渡される。この方法は一応の目的は果たすが、分かりにくく、プログラムの構造に対する制限も大きい。

Windows NTのASPにおいては、HTMLの中にスクリプトが埋め込めることから、次のような書法が考えられる。

〈IMG SRC=〈%=Chart.Image%〉〉

ここにChartとはあるグラフ作成Active Xコントロールのオブジェクトであり、Imageはこのオブジェクトの生成するグラフイメージである。しかしこの方法はうまくいかない。〈%=%〉の書法はWriteメソッドの別記法であって、バイナリの書き出しにはBinaryWriteメソッドを使う必要があるからである。しかしBinaryWriteでイメージを表示するにはヘッダにtext/htmlの変わりにimage/gifなどの指定が必要となる。

そのほか、Java appletの活用が考えられるが、WWWブラウザのJava環境の違いが大きい、Java VMの起動に時間がかかるなど、克服すべき問題も多い。

ダイナミックに生成した画像イメージをWWWブラウザに表示するためのシンプルな機構が望まれる。

6 あとがき

本システムは当初観測値データ処理パソコンが不安定になることが多く、なかなか安定的な運用ができないでいたが、現在ではこのパソコンに専用のマシンをあて、OSをアップデートすることで、今のところ数ヵ月間比較的安定に運用できている。

本報告では、観測データのWWWによる提供システムが、データへのアクセスさえ確保すれば、あとは特殊なあるいは高価なハードウェアやソフトウェアを必要とせずに構築可能であることを示すことができた。また構築のノウハウについてもWindows NTやLinuxなどの主要なプラットフォームについては、WWW開発やデータベース関連の情報はインターネットや印刷物を通じてかなり豊富に流通している。また、本システムはいまだ実用には供していないが、普及しているWWWブラウザのインターフェイスにより、分かりやすいグラフイメージを提供できる点は広くデータを公開する際に十分意義あるものと思われる。

しかしながら、実装例を通じていくつかの課題も残った。

データ処理パソコンにおいて観測装置からのデータを取り込むのには観測装置と一緒に導入されたソフトウェアをそのまま使用したが、観測装置からのデータを取り込むためのドライバソフトを自分で作成すれば直接データベースに格納できる。そうすればファイルにいったん落す手間が省けるだけでなく、データベースの同時アクセス制御によって、データを観測装置から取り込む最中にも直近のデータが検索できるであろう。

WWWブラウザへのグラフ画像の送り方として理想的と思われる方法が見つけられなかった。いろいろな事情はあるが、ここではWWWのプログラミング環境が十分成熟していないこと、とくに研究用途に関しては未開拓なことを理由として指摘しておきたい。

Linuxの場合でいうと、gnuplotは高機能ではあるがWWWインターフェイスが整備されているとはいいがたい。PerlにおけるGDインターフェイス・パッケージであるGD.pmの低レベルな機能はWWWから使いやすいが、これを直接使ってグラフを描くのは手間がかかる。GD.pmを使ったグラフ描画パッケージもあるが、横軸方向は均等にしかプロットできないものがほとんどである。これらは

いわゆるビジネスチャートを描く目的のものであるからであり、科学分野で普通である横軸方向不均等なプロットや関数のグラフなどはサポートされていない。本論のWindows NTにおいても使われたExcelでは関数のグラフはサポートされていないのと同様の事情といえるだろう。広く科学分野で使えるWWW用のライブラリの充実が望まれる。またWWWオーサリング環境一般の一層の発展に待つところも少なくない。

研究用途へのサポートは商用ソフトウェアにおいてはなかなかなされないが、こういった開発整備はWWWサーバのユーザ（研究者）自身でもWWW用グラフ描画ライブラリの作成など、ある程度可能である点はWWW環境の優位性として強調されるべきであろう。

本論のシステム構築を始めた1998年ごろと違い、最近ではJava servletの環境がSun OS, Windows NT, Linuxその他の上で整備されつつある。JavaはVMさえあればOSを問わないシステムの構築を目指すもので、特にサーバアプリには適しているといわれる。本システムにJava servletを用いるとどうなるかを試してみる価値は十分にあるであろう。

結論として、現状ではWWWは研究データの活用・公表といった用途に対し十分実用的ではあるが、まだまだ整備の余地がある。

最後に、ここで紹介したWWW検索システムのURLは以下のようである。

・Windows NTによるWWW検索システムのURL

http://csi.fish-u.ac.jp/kisyou/

・LinuxによるWWW検索システムのURL

http://d165.fish-u.ac.jp/kk/perl/kisyou.pl

kisyou.plのソース：

http://d165.fish-u.ac.jp/kk/kisyou.pl

getadd.plのソース：

http://d165.fish-u.ac.jp/kk/getadd.pl

文献

1）升屋正人：Active Server Pages構築術, 初版, ソフトバンク, 東京, 1998.

2）生形洋一：ASP実践プログラミング, 初版, 技術評論社, 東京, 2000.

3）杉山利幸：Multi Function Alarm 1.42, http://member.nifty.ne.jp/T_sugiyama/

4）T. Baba：BASP21, http://www.hi-ho.ne.jp/babaq/

Table 1-1. Part of the VB source program, data-g-p.frm, which is to read the weather observation data and to insert them into the database.

```
'***file read***
 Open pass & fname For Input As #1
 For ic = 0 To 60              'レコード数指定
    Line Input #1, textline
    kiri = 1
    mojisu = Len(textline)
    fc = firstcomma(textline)
    data(ic, kiri) = Mid(textline, 1, fc - 1)
    textline = Right(textline, mojisu - fc)
    If Right(data(ic, kiri), 2) = "99" Then Exit For
    For kiri = 2 To 11
        mojisu = Len(textline)
        fc = firstcomma(textline)
        data(ic, kiri) = Mid(textline, 1, fc - 1)
        textline = Right(textline, mojisu - fc)
    Next kiri
 Next ic
 Close #1
 ic = ic - 1    '最新データのレコード番号
'***データベースへの書込み***
 nengetu = yrr & "/" & mon & "/" & dy
 Set Db = DBEngine.Workspaces(0).OpenDatabase(dbname)
 SQL = "INSERT INTO " & tbname & "(ID,年月日,時刻" _
 & ",平均風速,平均風向,瞬間風速,瞬間風向,気温," _
 & "相対湿度,実効湿度,現地気圧,海面気圧)" _
 & "VALUES(" & ip & "," & "'" & nengetu & _
 & "'" & "," & "'" & data(ic, 1) & "'" & "," _
 & data(ic, 3) & "," & data(ic, 4) & "," _
 & data(ic, 5) & "," & data(ic, 6) & "," _
 & data(ic, 7) & "," & data(ic, 8) & "," _
 & data(ic, 9) & "," & data(ic, 10) & "," _
 & data(ic, 11) & ")"
 Db.Execute SQL, dbFailOnError
 Db.Close
```

Table 1-2. Part of the VB source program, make-gif-graph.frm, which is to make the recent graphs.

```
'***Excel 表の作成***
'***ＤＢデータのレコード数を調べる***
 k = 0
 Do Until Rs.EOF
    k = k + 1
    Rs.MoveNext
 Loop
 Rs.MovePrevious
 ik = k - 1   '-1 の意味：最初のレコードはダミー
 mx = 36      '取得するレコード数（３時間分）
 keyid = ik - mx
 For m = 1 To mx
  keyid = keyid + 1
  SQL = "SELECT * FROM " & tbname & " WHERE ID=" & keyid & "ORDER BY ID";
'Rs に検索結果を格納
  Set Rs = Db.OpenRecordset(SQL, dbOpenDynaset)
'Excel 用データへの加工
  dat = ""
  Rl.Cells(m, 1).Value = Rs(0) 'ID
  dat = dat & Rs(0) & " : "
  Rl.Cells(m, 2).Value = Year(Rs(1)) & "/" & Month(Rs(1)) & "/" & Day(Rs(1))   '年月日
  dat = dat & Rs(1) & " : "
  Rl.Cells(m, 3).Value = Hour(Rs(2)) & ":" & Minute(Rs(2))   '時刻
  dat = dat & Rs(2) & " : "
   For i = 3 To Rs.Fields.Count - 1   'その他
     ii = i + 1
     Rl.Cells(m, ii).Value = Rs(i)
     dat = dat & Rs(i) & " : "
   Next i
 Next m
'***グラフ描画***
'①平均風速のグラフ作成&GIF 形式保存
 Robj.Activate
 Set chartobj = Robj.ChartObjects.Add(50, 40, 400, 300)
 With chartobj.Chart
    .ChartType = xlLineMarkers
    .SetSourceData Source:=Robj.Range("C1:D36"), PlotBy:=xlColumns
 End With
    chartobj.Chart.Location Where:= _
      xlLocationAsObject, Name:="Sheet1"
    chartobj.Chart.HasLegend = False
'グラフシートの GIF 形式保存
 chartobj.Chart.Export "c:¥InetPub¥Webroot¥kisyou" & "¥gif¥ave_wind_v.gif", "GIF"
```

Table 1-3. ASP script of saving the file of information including date, time and others specified by clients.

```
<%      fp=Server.MapPath("datasave.txt")
        Set fso=Server.CreateObject(
          "Scripting.FileSystemObject")
        Set tso=fso.OpenTextFile(fp,2)              'Writing
        fname=selfn & yr & mo & dy & hr & mn & sec
          & ".gif"
        tso.WriteLine yr & "," & mo   & "," & dy & ","
          & hr & "," & mn & "," & sel & "," & fname
          & "," & fn & "," & sa
        tso.close %>
```

Table 1-4. ASP script of executing the executable program.

```
<%      set bobj=Server.CreateObject("basp21")
        rc=bobj.Execute("c:¥InetPub¥wwwroot¥kisyou
          ¥make-graph.exe",1,stdout) %>
```

Table 1-5. Part of the VB source program, make-graph.frm, which is to make the graph as to the keys of weather observation, date, time and others specified by clients.

```
    Open "C:¥InetPub¥wwwroot¥kisyou¥datasave.txt" For Input As #1
    Input #1, yr, mo, dy, hr, mn, listname, fname, fdNum, sa
    Close #1
'日付設定
    sdate = DateSerial(yr, mo, dy)
    stime = TimeSerial(hr, mn, 0)
    If hr - sa < 0 Then
        dtime = TimeSerial(24 + hr - sa, mn, 0)
        ddate = sdate - 1
    Else
        dtime = TimeSerial(hr - sa, mn, 0)
        ddate = sdate
    End If
    sdate = "20" & sdate
    ddate = "20" & ddate
'NT 用時間表示変換 午前・午後表示－＞0～23 時表示
    sampm = Mid(stime, 1, 2)
    shr = Hour(stime)
    smn = Minute(stime)
    dampm = Mid(dtime, 1, 2)
    dhr = Hour(dtime)
    dmn = Minute(dtime)
    stime = CStr(shr) & ":" & CStr(smn)
    dtime = CStr(dhr) & ":" & CStr(dmn)
'sa 時間戻した時、日付が変わらない場合の処理は if の前半、日付が変わる場合は後半
    If sdate = ddate Then
'--------日付が変わらない場合------
        SQL = "SELECT * FROM [観測値マスタ] WHERE (年月日=#" & ddate & "# _
              and 時刻>=#" & dtime & "#) and " & "(年月日=#" & sdate & "# _
              and 時刻<=#" & stime & "#) ORDER BY ID;"
        Set Rs = Db.OpenRecordset(SQL, dbOpenDynaset)
        ira = 1
        Rs.MoveFirst
        Do While Not Rs.EOF
            Rl.Cells(ira, 3).Value = TimeValue(Rs.Fields(2))
            Rl.Cells(ira, 4) = Rs.Fields(fdNum)
            ira = ira + 1
            Rs.MoveNext
        Loop
        ir = ira - 1
    Else
'----前日分の処理-----
        SQL = "SELECT * FROM [観測値マスタ] WHERE (年月日=#" & ddate & "# _
              and 時刻>=#" & dtime & "#) and " & "(年月日=#" & ddate & "# _
              and 時刻<=#" & "23:59" & "#) ORDER BY ID;"
        Set Rs = Db.OpenRecordset(SQL, dbOpenDynaset)
        irb = 1
        Rs.MoveFirst
        Do While Not Rs.EOF
```

```
            Rl.Cells(irb, 3).Value = TimeValue(Rs.Fields(2))
            Rl.Cells(irb, 4) = Rs.Fields(fdNum)
            irb = irb + 1
            Rs.MoveNext
        Loop
'----- 翌日の処理-----
        SQL = "SELECT * FROM [観測値マスタ] WHERE (年月日=#" & sdate & "# _
                and 時刻>=#" & "00:00" & "#) and " & "(年月日=#" & sdate & "# _
                and 時刻<=#" & stime & "#) ORDER BY ID;"
        Set Rs = Db.OpenRecordset(SQL, dbOpenDynaset)

        Rs.MoveFirst
        Do While Not Rs.EOF
            Rl.Cells(irb, 3).Value = TimeValue(Rs.Fields(2))
            Rl.Cells(irb, 4) = Rs.Fields(fdNum)
            irb = irb + 1
            Rs.MoveNext
        Loop
        ir = irb - 1
    End If
'グラフ&ＧＩＦ作成
    Robj.Activate
    myrng = Robj.Cells(1, 3).CurrentRegion.Address
    Set chartobj = Robj.ChartObjects.Add(50, 40, 400, 300)
    With chartobj.Chart
        .ChartType = xlLineMarkers
        .SetSourceData Source:=Robj.Range(myrng), PlotBy:=xlColumns
    End With
        chartobj.Chart.Location Where:=xlLocationAsObject, Name:="Sheet1"
        chartobj.Chart.HasLegend = False
'グラフシートを GIF 形式で保存
    fname = "c:¥InetPub¥wwwroot¥kisyou¥gif¥" & fname
    chartobj.Chart.Export fname, "GIF"
```

Table 2-1. Perl script of searching the database by the keys of weather observation and date.

```
my $q = new CGI;
# ブラウザで指定されたデータの種類、年月日を受け取る
my $dataname = $q->param('dataname');
my $date = $q->param('date');

# 検索をする。
my $sql = "select hour,minute,$dataname from master where date=¥'$date¥'";
$sql .= " order by hour,minute";
my $dbh=DBI->connect('dbi:mysql:kisyou',"****") or die "Cannot connect";
# ****のところにはログイン情報がはいる。
my $sth=$dbh->prepare($sql);
$sth->execute or die "Cannot execute $sql";

# 検索結果を@rows に格納する。
my @rows=();
while (my @r=$sth->fetchrow) {
        push @rows,¥@r;
}
$sth->finish;
$dbh->disconnect;
```

Table 2-2. Perl script of making the graph of wind vectors with average wind velocity and average wind direction in polar coordinates under the Linux WWW system.

```
#!/usr/bin/perl -w
use Apache ();
use CGI;# Perl の CGI インターフェイス
use DBI;
use GD;# Perl の GD ライブラリインターフェイス
use vars qw/$query/;
use vars qw/$TITLE/;
local $query = new CGI;
local $TITLE = "気象データ";
# リクエストのパス情報によって対応を変える。
my $path_info = $query->path_info;
&print_frameset if !$path_info;
&print_query if $path_info=~/query/;
&print_response if $path_info=~/response/;
&print_plot if $path_info=~/plot/;

sub print_frameset {
# frame を記述する HTML をブラウザに書き出す。
        my $script_name = $query->script_name;
        print $query->header(-charset=>'x-euc');
        print <<EOF;
        <html><head><title>$TITLE</title></head>
        <frameset rows="40,60">
            <frame src="$script_name/query" name="query">
            <frame src="$script_name/response" name="response">
        </frameset>
        </html>
EOF
        ;
        Apache::exit(0);
}
sub print_query {
# 検索 FORM を記述する HTML をブラウザに書き出す。略。
}
sub print_response {
        unless ($query->param) {
                print $query->header(-charset=>'x-euc');
                print "<H3>Query Result</H3>¥n";
                print "<b>No query submitted yet.</b>";
                return;
        }
        my $script_name = $query->script_name;
        my $pdate=$query->param('date');
        my $dname=$query->param('dataname');
        print $query->header(-charset=>'x-euc');
        print "<center>";
        print $pdate,", ",&realname($pdataname),"<p>";
        my $ptime=time;# リクエスト毎に URL を変えるためのダミー情報
        my $key="dataname=$dname&date=$pdate&dummy=$ptime";
        print qq(<img src="$script_name/print_plot?$key><p>);
```

```
        # ここのパス情報により print_plot が実行される。

        print "</center>";
        print "</html>";
}
sub print_plot{

# FORM データにより気象データを検索する。上の Table 2-1 がここに入る。
# @rows に入っている気象データをグラフにする。
        &winplot(¥@rows);# これは風速-風向データの場合。その他の場合は略。
}
sub realname{
# カラムの日本語名を返す。略。
}
sub winplot {
        my ($data)=@_;

# GD のイメージベースを作る。
        my ($xspan,$yspan) = (400,300);
        my $im = new GD::Image($xspan,$yspan);

# 色の名前を定義する。略。
# $data の風速 max、風向 min を求める。略。

# 原点を定義する。GD の座標系では原点は左上、y は下方向に増える。
        my ($x0,$y0) = ($yspan/2,$yspan/2);

# 同心円を描き、ラベルを入れる。略。
# 方向軸を描き、ラベルを入れる。略。
# 凡例を描く。略。

# データを極座標プロットし、時間帯別に色分けした線で結ぶ。
        my $pi=4*atan2(1,1);
        my ($hr,$min,$r,$ang,$x,$y,$xp,$yp);
        $xp=0;
        for (@$data){
                ($hr,$min,$r,$ang)=@$_;
                if($hr<6){$color=$black;}
                elsif($hr<12){$color=$green;}
                elsif($hr<18){$color=$red;}
                else{$color=$blue;}
                $ang=($ang<5)?$ang+11:$ang-5;# 元データは、北=1 で右周り
                $x=$x0-$r*$runit*cos(-$pi*$ang/8);
                $y=$y0-$r*$runit*sin(-$pi*$ang/8);
                $im->line($xp,$yp,$x,$y,$color) if $xp;
                ($xp,$yp)=($x,$y);
        }
        print $im->png;
}
sub yajiri {
# 方向軸の矢印を描く。略。
}
```