原稿No.ＨＺ１１－０３

PCによる境界要素法と有限要素法に関する大容量計算

大分大学　平居　孝之＊
Takayuki Hirai

１．ＰＣと大型計算機

最近のＰＣの性能向上とその普及は目ざましく、事務処理から科学技術計算まで広範囲に利用され、ほとんどの事業所では手元にＰＣを置き自由に使えるようになっている。また必要にせまられれば直ちにＰＣを購入して利用に供することができ、そのための費用も安価になりつつある。このように一般に普及している汎用のＰＣを利用して、構造物や流体の解析ができれば大変便利であるが、ＰＣの計算速度が遅いことと演算容量が小さいという２つの制約から、なかなか思うようにいかない。大学や研究施設にある大型計算機あるいは大きな演算容量を持ち比較的計算速度の速い高価なミニコンを使っているのが現状である。

計算速度が遅いことは、利用者が休息を取る夜間に計算させることでカバーし、演算容量が小さいことはプログラミングで工夫する方法により、例えば１セットが50万円程度の価格の汎用のＰＣを使って大容量の演算領域が必要な計算をこなすことは、理屈では可能であるが、プログラミングが難解であるという問題があってなかなか実現しない。

ここでは弾性問題を境界要素法と有限要素法で解析する場合を例とし、演算容量の小さい汎用のＰＣにより数値計算する方法について述べる。なお計算方法の詳細は拙著[1)2)]を参照されたい。

２．解析方法の数値計算上の特徴

２－１　境界要素法

境界要素法は、境界積分方程式を解法の原理として発展してきたが、最近では古くから利用されている重ね合わせ法も境界要素法と呼ばれるようになった。積分方程式法には直接境界積分方程式法と間接境界積分方程式法の２種類があり、筆者は数値計算方法の観点からは、間接境界積分方程式法は重ね合わせ法と同一の数値計算を行っていると考えている。

２－１－１　直接境界積分方程式法

次式は直接境界積分方程式法の基本式である。対象としている問題と同じ領域における既知の平衡状態を用意し、これと問題の平衡状態との間に次式の積分方程式を導き、境界を分割した要素の上で積分計算を離散化して行い、境界上の未知の変位と表面力を未知数とする連立１次方程式を構成して解く方法である。

$$\int_{S_1} {}^*u\underline{T}ds - \int_{S_2} {}^*T\underline{u}ds = \int_{S_1} {}^*Tuds - \int_{S_2} {}^*uTds - \int_{v} {}^*uFdv$$

ただし

$\int\cdots ds$：境界上の積分　　$\int\cdots dv$：領域上の積分

S_1　：変位既知の境界　S_2　：表面力既知の境界

${}^*u$、${}^*T$　Ｔ：既知の平衡状態の変位と表面力

u、T　：問題の平衡状態の変位と表面力で既知の値

$\underline{u}$、$\underline{T}$　：問題の平衡状態の変位と表面力で未知の値

F　：問題の領域に作用する物体力

連立１次方程式を構成する個々の方程式中の境界上の変位と表面力を適当な関数で近似して表し、それを要素上で積分することになる。このため関数近似と積分計算の過程で生じる誤差が連立方程式の未知数に付

＊大分大学工学部教授

く係数と、右辺の定数項に含まれることになる。特に右辺の定数項は、全境界を分割した要素ごとの積分の代数和として計算されるので、有効数字の桁落ちに起因する誤差の影響が大きい。このように係数だけでなく右辺の定数項にも誤差を含んだ連立１次方程式を用いる直接境界積分方程式法は、重ね合わせ法や有限要素法に比べて解の精度の点で不利な場合が多い。

２－１－２　重ね合わせ法

問題と同じ材質の領域で、既知の平衡状態に重みを掛けて重ね合わせ、境界を分割して設定した要素上で境界条件が問題に与えられた値に近似するように解く方法である。重みの値ははじめ未知であり、境界条件を近似するという条件から重みを未知数とする連立１次方程式が構成され、これを解いて重みの値を求め、今度は既知となった重みを掛けた既知の平衡状態を重ね合わせて、解とする方法である。境界条件の近似における誤差の程度と、選んだ既知の平衡状態が適当であるか否かにより計算結果に含まれる誤差は左右されるが、満足な精度の計算結果を得られる場合が多い。さらに重ね合わせ法は理論解の分らない問題であっても、数値計算で得られた結果に含まれる誤差を検討できるという長所を持つ。

重ね合わせ法の場合の未知の重みを求めるための連立１次方程式は、個々の方程式において係数と右辺の定数項が表面力または変位のいずれかに統一された数値で構成され、また右辺の定数項には境界条件で与えられた値がそのまま入るので、解の精度の点で連立１次方程式の性質は良い。しかしブロックが接合されたような問題を除くと、係数マトリックスの成分がすべて０でなく、また対称マトリックスでもないので、係数マトリックスの実質的な容量を縮小して解く方法がない。このため未知数の数が少なくても数値計算においては、かなり大きな容量が必要である。

２－２　有限要素法

有限要素法は、問題の領域を小区域の要素に区切り、各々の区域ごとの変位または応力を簡単な関数で表し、隣接する要素および要素と外部は要素の端部に設けた節点で接合されているとして解く方法である。　有限要素法の数値計算において基本になる式は、節点の変位と外力の間に成り立つ剛性方程式であり、これは例えば要素の歪エネルギーと節点に作用する力による仕事の関係から求められる。これらの個々の要素ごとの方程式から、節点で接合された要素のそれぞれに作用している節点の力を合計し、それが外部から作用する外力に等しくなるという釣合方程式を作成する。この釣合方程式は、各々の節点で接続された要素の剛性方程式から構成されるので、その節点に接続されていない要素の剛性方程式は関係が無く、したがって項の数が少ない。釣合方程式は全体で連立１次方程式になり、これを解いて節点の変位と作用する力で未知の値を求める。

有限要素法で使う連立１次方程式は個々の方程式の項の数が少なく、また枢軸にする係数の絶対値が大きいため、適当な解法を使えば満足な精度の解が得られる。計算時間の短縮化と必要な演算容量の縮小化のためには、係数マトリックスが対称マトリックスであり、かつ０である成分がきわめて多いという特徴を生かした解法を選ぶ必要がある。

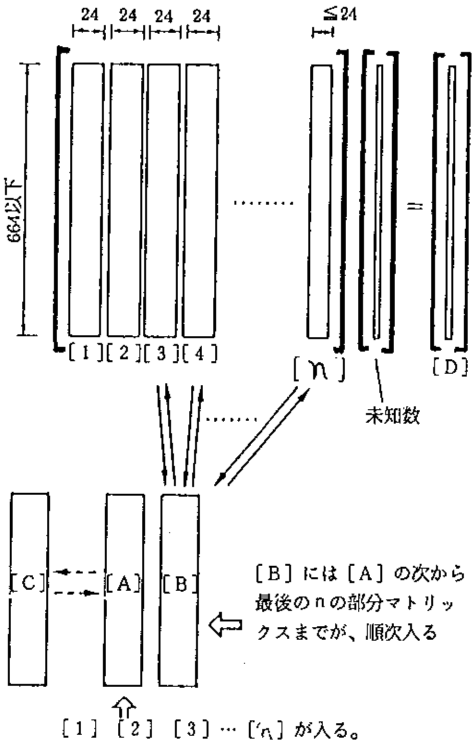
３．大容量の演算容量を必要とする連立１次方程式をＰＣで解くためのプログラミング

大型計算機の場合は、大きな係数マトリックスを１つの配列に収納できるので、一般に計算が簡単であり、また計算機に組み込みのＳＳＬ（サイエンテフィックサブルーチンライブラリイ）から適当なものを使える。ところがＰＣの場合は、未知数の数が数百を越え係数マトリックスの容量が大きい連立１次方程式を解くには、係数マトリックスで実質的に必要な部分だけを使って計算し、さらにＰＣ本体の演算容量だけで計算できない場合は、係数マトリックスを部分に分割してフロッピィディスクのファイルに記憶し、それらのファイルを読み書きしながら計算する必要がある。

３－１　境界要素法における枢軸変換の掃き出し法

重ね合わせ法による境界要素法では、連立方程式の係数マトリックスの成分のディメンジョンに大きな差がないように、係数の値を個々の方程式ごとに換算した後、係数マトリックスの各列で絶対値が最大になる成分を枢軸（ＰＩＶＯＴ）にする掃き出し法を使えば、一般に単精度の計算で十分な精度が得られる。

大容量の係数マトリックスで、その全体がＰＣの本体の容量に納まらない場合は、フロッピィディスクまたは本体に増設したディスクに、係数マトリックスを分割して記憶させて計算することになる。このとき、



第１図　部分マトリックスによる計算

枢軸の入れ換えをする場合は、数行ごとに区切るより数列ごとに区切る方がディスクへの読み書きの回数が減るが、それでもディスクとのファイルのやり取りに相当の計算時間を費やす。またハードの面でデータの取り扱いの基本単位が64ＫＢであるような、現在最も広く使われているＰＣの場合は、本体の演算処理部とディスクとのデータのやり取りを64ＫＢ単位で行う方が計算時間が短く、ソフト上でデータの取り扱い単位を64ＫＢより大きくできる場合に、大きな配列を作ってそれでディスクとやり取りを行うと、データの転送に費やす時間がより長くなる。

第１図は 664×664 以内の大きさで成分が全部必要な値で埋まった係数マトリックスを２枚の１ＭＢのディスクに記憶させて連立方程式を解く場合の例である。係数マトリックスを、24列ごとの部分に縦割りで分割する。最後の［ｎ］の部分は、列の数がちょうど24とは限らない。ここでは説明のためこれらの部分マトリックスに［１］、［２］、［３］……［ｎ］の番号を付ける。24列で 664行の配列Ａ、Ｂ、Ｃと、１列の配列Ｄを使い、次のような順序で計算を行う。

1）部分マトリックスの［１］、［２］、［３］……［ｎ］を、ディスクのそれぞれの名前を付けたファイルに書き込む。また配列Ｄに右辺の定数項を入れる。

2）配列Ｃに［１］の部分マトリックスをディスクのファイルから読み込んで入れる。

3）配列Ａを配列Ｃと同じ値にする。

4）配列Ｂに［２］の部分マトリックスをディスクのファイルから読み込んで入れる。

5）配列ＡとＢを合わせた48列のマトリックスで、配列Ａに含まれる24列の各列について、そこまでの枢軸に選ばれていない成分のうちから絶対値の最大のものを捜し出し、これを枢軸として列の他の成分が０になるように掃き出し計算を行う。

6）配列Ｂを［２］の部分マトリックスを記憶するディスクのファイルに書き込む。

7）［２］の代わりに［３］～［ｎ］の部分マトリックスを配列Ｂに入れて、それぞれ３）～６）を繰り返す。

8）ただし７）で［ｎ］の部分マトリックスについて計算するときは、配列ＡとＢを合わせたものの後にさらに配列Ｄを付けて配列ＡとＢとＤを合わせたマトリックスについて５）の計算を行い、配列Ａの成分についての掃き出し計算を配列Ｄにも行う。

9）配列Ｃに［２］から［ｎ－１］の部分マトリックスを順番にディスクのファイルから読み込んで、それぞれ３）～８）を繰り返す。このとき配列Ｂには配列Ｃに入れた部分マトリックスの次の部分マトリックスから最後の［ｎ］の部分マトリックスまでを入れる。

10）９）までが終了した時点で、配列Ｂに収納されている［ｎ］の部分マトリックスについてだけ掃き出し計算が行われていないので、配列Ｂと配列Ｄを合わせたマトリックスで、配列Ｂに含まれる列について、そこまでの枢軸に選ばれていない成分のうちから絶対値の最大のものを枢軸として、列の他の成分が０になるように掃き出し計算を行う。

11）枢軸の値を１にしたときの配列Ｄが連立１次方程式の解である。ただし、配列Ｄには枢軸の順番で解が並んでおり、元の未知数の順番と違っている。

３－２　有限要素法におけるスカイライン法

有限要素法の場合は、係数マトリックスが対称マトリックスでかつ実際の計算に必要でない成分がきわめ

$$
\begin{bmatrix}
u_{11} & u_{12} & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & u_{22} & u_{23} & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & u_{33} & u_{34} & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & u_{44} & u_{45} & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & u_{55} & u_{56} & u_{57} & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & u_{66} & u_{67} & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & u_{77} & u_{78} & \cdots & u_{7n} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & u_{88} & \cdots & u_{8n} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_{nn}
\end{bmatrix}
$$

$[U]$

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
l_{21} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & l_{32} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & l_{43} & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & l_{54} & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & l_{65} & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & l_{75} & l_{76} & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & l_{87} & 1 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & l_{n7} & l_{n8} & \cdots & 1
\end{bmatrix}
$$

$[L]$

第３図

$$
\begin{bmatrix}
g_{11} & g_{12} & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
g_{21} & g_{22} & g_{23} & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & g_{32} & g_{33} & g_{34} & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & g_{43} & g_{44} & g_{45} & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & g_{54} & g_{55} & g_{56} & g_{57} & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & g_{65} & g_{66} & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & g_{75} & 0 & g_{77} & g_{78} & \cdots & g_{7n} \\
0 & 0 & 0 & 0 & 0 & 0 & g_{87} & g_{88} & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & g_{n7} & 0 & \cdots & g_{nn}
\end{bmatrix}
$$

第２図　成分の並び換えをした係数マトリックス［G］

て多いという性質を利用できるスライカイン法が適している。この方法により、例えば本体の容量が 640KB程度の汎用のＰＣでも千を越える未知数の連立方程式を解ける。またディスクを記憶容量として使えば、さらに未知数の多い連立１次方程式を解ける。その場合はスカイライン法のほかここでは説明していないが共役傾斜法などの繰り返しによる近似計算などの利用も考えられる。

３－２－１　係数マトリックスのバンド幅の縮小化

係数マトリックスが対称であるという条件を満たしながら、０でない成分が対角線の付近に集まるように方程式の順番と未知数の順番を並べ換えることは、計算時間の短縮と必要な演算容量の縮小化に有効である。計算モデルの作成者が、０でない成分が対角線の付近に集まるように入力データを作ることも出来るが、この方法ではモデル設定が煩雑になるので、自動的に係数マトリックスの０でない成分が対角成分の付近に密集するようにデータの並べ換えを行うようプログラミングする必要がある。

３－２－２　係数マトリックスの三角分解

第２図に示すようにバンド幅が小さくなるように成分を並び換えた係数マトリックス［G］が与えられたとする。係数マトリックス［G］を三角行列に分解して、下三角マトリックス［L］と上三角マトリックス［U］の積に変形する。

［G］＝［L］［U］

三角マトリックスの［L］と［U］は第3図のようになり、その成分 u_{ij} と l_{ij} の計算は次のように行われる。

$$u_{1j}=g_{1j} \qquad j=1,2,3\cdots\cdots n$$

$$u_{ij}=g_{ij}-\sum_{r=1}^{i-1}\frac{u_{ri}u_{rj}}{u_{rr}} \qquad \begin{cases} i=2,3,4\cdots\cdots n \\ j=i,i+1\cdots\cdots n \end{cases}$$

$$l_{ij}=\frac{u_{ji}}{u_{jj}} \qquad \begin{cases} i=j,j+1\cdots\cdots n \\ j=1,2,3\cdots\cdots n \end{cases}$$

第２図と第３図では、三角分解される前のマトリックス［G］、三角分解されたマトリックス［U］と

第１表　ＰＣと計算可能容量の限度の例

	計算モデル			連立１次方程式		ＰＣの容量	
	対象領域	要素の数	節点の数	未知数の数	係数マトリックスの必要成分の数	本体	ディスク
境界要素法	２次元	166（２次要素）	—	661	440896	384ＫＢ	２ＭＢ
	３次元	221（１次要素）	—	664	440896	384ＫＢ	２ＭＢ
有限要素法	２次元	350（６節点三角形要素）	800	1600	96000	640ＫＢ	0
	３次元	200（20節点六面体要素）	1200	3600	223000	640ＫＢ	２ＭＢ

第２表　ＰＣによる計算時間の例

		計算モデルで使う連立１次方程式			計算過程で数値積分を使うか否か	計算時間 hour
		未知数の数	係数マトリックスの必要成分の数	解　法		
境界要素法	２次元問題	400	16000	枢軸変換の掃出し法	使う	2.5
	３次元問題	400	16000	枢軸変換の掃出し法	使う	2.5
有限要素法	２次元問題	1200	40000	スカイライン法	使う	7.0
	２次元問題	1200	40000	スカイライン法	使わない	1.5
	３次元問題	1200	76000	スカイライン法	使う	7.0
	３次元問題	1200	60000	共役傾斜法	使う	8.0

$$\begin{bmatrix} u_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & u_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & u_{33} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_{44} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & u_{55} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & u_{66} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & u_{77} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_{88} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & u_{nn} \end{bmatrix}$$

第４図　マトリックス〔M〕

〔L〕について、０でない成分の存在する部分を包含するように枠で囲んでいる。これらを比べると、〔L〕と〔U〕で枠で囲った部分は、〔G〕の枠で囲った部分にちょうど含まれている。これは、〔L〕と〔U〕が前述の式から計算されるので、必然的なことである。なお〔G〕において枠内に０の成分があるが、これと同じ位置にある〔L〕または〔U〕の成分は０とは限らない。第３図に示した〔U〕マトリックスの枠で、上方の折線のことをスカイラインと呼ぶ。次で述べるように〔U〕マトリックスのスカイラインから対角線までの部分だけで、連立方程式が解ける。

３－２－３　Ｕマトリックスを用いたスカイライン法による連立方程式の解法

第４図のように、対角成分が〔U〕の対角成分に等しく、その他の成分は０であるマトリックスを〔M〕とする。前述の〔L〕と〔U〕マトリックスを計算する式から〔L〕と〔U〕と〔M〕の間には〔U〕＝〔M〕〔L〕^{T} の関係があり、これから次式が得られる。

〔U〕^{T} ＝〔L〕〔M〕^{T} ＝〔L〕〔M〕であるから、

$$(\,[U]^{T}\,)^{-1} = (\,[L]\,[M]\,)^{-1} \qquad (1)$$

連立１次方程式を三角分解されたマトリックスを用いて解くが、プログラム上の数値計算では、以下のように連立１次方程式を変形し、〔U〕マトリックスと〔U〕の対角成分から成る〔M〕マトリックスだけを使い〔L〕マトリックスを使わないようにする。

連立１次方程式を三角マトリックスで表すと次式になる。ただし〔X〕は未知数の１列のマトリックスで、〔P〕は右辺の定数項の１列のマトリックスである。

$$[L]\,[U]\,[X] = [P]$$

右辺を変形して、

$$[L]\,[U]\,[X] = [L]\,[M]\,(\,[L]\,[M]\,)^{-1}\,[P]$$

(1)式を代入して、

［L］［U］［X］＝

［L］［M］（［U］^{T}）$^{-1}$［P］

両辺から［L］を消去出来るので、

［U］［X］＝［M］（［U］^{T}）$^{-1}$［P］　(2)

この(2)式が実際の数値計算で使う式であり、その計算は次のa)～c)の順序で行う。

a)［U］^{T}［W］＝［P］を解いて１行のマトリックス［W］を求める。このようにして(2)式の右辺の一部である（［U］$^{-T}$）$^{1\ -1}$［P］の値（＝［W］）が求まる。

b)［M］［W］を計算する。これが(2)式の右辺の値である。

c)［U］［X］＝［M］［W］を解いて、解［X］を求める。

以上のような［U］マトリックスだけを用いるスカイライン法によれば、係数マトリックスの必要な部分だけを記憶容量に納めて、未知数の数が多い連立１次方程式を解くことができる。

４．実行例

ＰＣの容量に対して計算できるモデルの大きさを第１表に示す。境界要素法で枢軸変換の掃き出し法を使う場合は、ＰＣの本体に384ＫＢの容量があり、１ＭＢのフロッピィディスクドライブを２つ接続しているシステムにより、成分がすべて埋まった係数マトリックスで未知数の数が664の連立１次方程式を使うモデルが解ける。この場合は、本体の記憶領域は384ＫＢであるが、外部記憶領域のディスクから1700ＫＢ程度を利用し、本体に２ＭＢの容量を持つ計算機と同じ計算をしている。有限要素法の場合は、係数マトリックスの必要な成分だけを記憶して解けるので、本体に640ＫＢの容量を持つＰＣを使って、未知数の数が1600の連立１次方程式を使うモデルが解ける。さらに外部記憶領域としてディスクで２ＭＢを使えば、未知数の数が3600の連立１次方程式を使うモデルが解ける。

第２表はＰＣによる計算時間の例である。境界要素法の場合は、計算過程で使う数値積分と、連立１次方程式を解く過程で係数マトリックスを分割してディスクに記憶するのに、ディスクへのデータの読み書きに時間がかかる。有限要素法の場合は、連立１次方程式を解くための計算時間は、未知数の数が多くても比較的短いが、要素の剛性方程式を作成する過程で数値積分をするための計算時間がかなり必要である。なお２次元で計算時間が短いのは、剛性方程式を数値積分でなく解析的に計算しているからである。

＜参考文献＞

(1) 平居孝之、有限要素と境界要素法、1988、共立出版

(2) 平居孝之、弾性解析プログラムとその使い方、1984、理工図書

筆　者　紹　介

ひらい　たかゆき
平　居　孝　之
現在、大分大学教授。昭和44年3月東京工業大学工学部建築学科卒業。昭和51年3月東京大学大学院博士課程修了。工学博士。主なる報文／「コンクリートの破壊靱性評価に関する研究」セメント技術年報39（1985）他。勤務先住所／〒870-11　大分市旦の原700、電話　0975-69-3311。