

Tesseral Quaternions for the Octtree

S. B. M. BELL AND D. C. MASON

NERC Unit for Thematic Information Systems, Geography Department, University of Reading, Whiteknights, PO Box 227, Reading RG6 2AB

The linear octtree of Gargantini¹ is discussed. Its addressing is extended to the whole of 3-dimensional Euclidean space. A place-system Tesseral arithmetic^{2,3} operating directly on the octtree addresses is described. It is based on the quaternions. They provide the geometrical operations of vector addition and subtraction, and rotate and scale.

Received March 1988, revised February 1989

1. INTRODUCTION

Hierarchical data structures have been reviewed by Samet,⁴ and recent papers on the octtree^{5, 6, 7, 8} show that this method of dividing and addressing 3-dimensional data has many advantages. The linear octtree is explored by Gargantini.¹ In this the 3-dimensional space of the image is addressed in an hierarchical manner, and the address plus the data relevant to that address are stored in the form of a sorted list. This method of storing the data is useful in computer graphics^{5, 7} and might be used in the field of Geographic Information Systems for digital elevation models in the same way as quadrees⁴ are already used. Processing such data usually involves some geometric transforms such as translation, rotation and scaling. If Cartesian geometry is used to provide the required geometric manipulation, the hierarchical address must be separated into Cartesian components before the geometrical transform, and possibly re-converted into an hierarchical address afterwards.¹ However, after extending the Gargantini address scheme¹ to the whole of 3-dimensional Euclidean space, it is possible to find a Tesseral² arithmetic which provides geometric transforms without conversion of the hierarchical address. This paper describes such an addressing and arithmetic for the octtree.

2. ADDRESSING THE OCTTREE

The image is taken to be a cube, with one vertex at the origin. The cube is split into 8 octants, and the octant nearest the origin is labelled with the symbol 0. Each parent octant is then split into eight further son octants each of which take a left-hand label symbol corresponding to the parent octant and a right-hand label symbol corresponding to the position of the son octant. For example, the son octant of the parent octant labelled 0 nearest the parent octant labelled *b* is labelled 0*b*. This process of subdividing octants is continued, the new label symbol always appended on the right, until the desired resolution of addressing is reached. Fig. 1 shows the situation after two subdivisions, and introduces the label symbols we shall use for the linear-octtree addressing: 0, *b*, *d*, *f*, *h*, *j*, *l* and *n*. The hidden octant is labelled *h*. In Gargantini's scheme¹ addresses covering a larger area than the minimum are indicated by appending another label symbol or label symbols, say *x* or *xx* etc., on the right-hand side of the address. *x* means that all 8 octants of the parent are addressed, so that *bx* means all of *b0*, *bb*, *bd*, *bf*, *bh*, *bj*, *bl*, *bn*. However, for our current purposes we want to address the octtree at the minimum

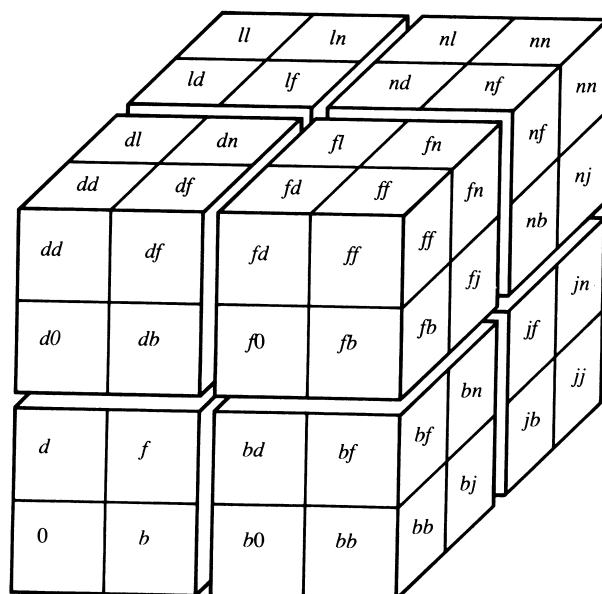


Fig. 1. The linear octtree subdivided twice.

resolution, and shall not be referring explicitly to the fact that the linear-octtree addressing lends itself to an hierarchical interpretation of the image-space.

3. ADDRESSING THE COMPLETE EUCLIDEAN SPACE

We see that the octtree addresses only cover one of the octants of the 3-dimensional space, which we shall call the positive octant. This is not sufficient if we want to provide a full geometry for the space, since the result of some spatial transform, for example a vector subtraction, or a rotate and scale, could lie outside the positive octant, and so we now extend the addressing scheme to cover the other 7 octants. The octant opposite the line 0 to *b* is given the prefix ...*bbbb*, by appending an infinite number of *bs* to the left-hand side of the address. This will be written symbolically as $\langle b$. All the other octants are similarly labelled: the one opposite the line 0 to *d* has all the addresses prefixed by $\langle d$ and so on. Suppose that the positive octant has been subdivided to some given resolution. Let *V* be the address of the pixel furthest from the origin in the direction of the line from 0 to *u*. To find the full address of a pixel in the octant prefixed by $\langle u$, where *u* is any Tesseral label symbol, pad the addresses in the positive octant with leading 0s until they are all the same length, then translate the positive octant along the

line u to 0 until V has passed the origin and is adjacent to it. Prefix all the addresses of all the translated pixels by $\langle u$.

4. MAPPING VECTOR ADDITION AND SUBTRACTION

We now provide an algorithm for doing vector addition directly on a pixel address label. We regard the address as a number in an arithmetic to which we give the generic name Tesseral², expressed in place-system form to a new 'areal' base. The digits are the individual symbols of the address, and we may map them on to Cartesian addresses as follows:

0	(0, 0, 0)
b	(1, 0, 0)
d	(0, 1, 0)
f	(1, 1, 0)
h	(0, 0, 1)
j	(1, 0, 1)
l	(0, 1, 1)
n	(1, 1, 1)

We first provide an answer for the addition of any two digits in addition table, Table 1. The method of forming

Table 1. Octtree digit addition

+	0	b	d	f	h	j	l	n
0	00	0b	0d	0f	0h	0j	0l	0n
b	0b	b0	0f	bd	0j	bh	0n	bl
d	0d	0f	d0	db	0l	0n	dh	dj
f	0f	bd	db	f0	0n	bl	dj	fh
h	0h	0j	0l	0n	h0	hb	hd	hf
j	0j	bh	0n	bl	hb	j0	hf	jd
l	0l	0n	dh	dj	hd	hf	l0	lb
n	0n	bl	dj	fh	hf	jd	lb	n0

the table is described in references 2 and 3. Multi-digit addition is performed by using a carry in a fashion exactly analogous to ordinary arithmetic with numbers expressed to any ordinary base. We provide a worked example.

$$\begin{array}{r}
 f \quad b \quad d \quad + \\
 n \quad j \quad d \\
 \hline
 f \quad j \quad l \quad 0 \quad \text{Answer} \\
 \hline
 b \quad d \quad \text{Carries}
 \end{array}$$

Subtraction is provided by an extension of the method used for P-Adic fields⁹ to a given base as described in reference 10 and forthwith described here. Suppose that the subtraction sum require is:

$$X = B - A$$

or, in the form we shall want to use it:

$$A + X = B \quad (1)$$

where A and B are known Tesseral numbers and X is unknown. We find the digits of X from right to left, starting with the least significant digit of X . Suppose that the last digit of A is d and the last digit of B is 0.

We require the last digit of X to be such that when we

add it to d we get the answer 0. A glance at the addition table, Table 1, shows that we must make the last digit of X equal to d in order for this to be true. The last digits of A , X , and B then cancel, and we can continue and find the penultimate digit of X via the equation:

$$\begin{aligned}
 A' + X' + d &= B' \quad \text{or} \\
 (A' + d) + X' &= B' \quad \text{or, if } A' + d \text{ is written } A'' \\
 A'' + X' &= B'
 \end{aligned} \quad (2)$$

It can be seen that equation (2) is now of the same form as equation (1), and we can proceed to solve it in the same way. The process ceases when we obtain an equation (3):

$$A''' + X''' = B''' \quad (3)$$

which is identical with some equation (4) previously found:

$$A''' + X''' = B''' \quad (4)$$

The sequence of digits found for X between equation (4) and equation (3), inclusive of (4) and exclusive of (3), will now repeat indefinitely, and the value of all digits of X has now been found.

To automate the process we provide a subtraction table, Table 2, in which the required digit and associated

Table 2. Octtree digit subtraction

−	0	b	d	f	h	j	l	n
0	0,0	0,b	0,d	0,f	0,h	0,j	0,l	0,n
b	b,b	0,0	b,f	0,d	b,j	0,h	b,n	0,l
d	d,d	d,f	0,0	0,b	d,l	d,n	0,h	0,j
f	f,f	d,d	b,b	0,0	f,n	d,l	b,j	0,h
h	h,h	h,j	h,l	h,n	0,0	0,b	0,d	0,f
j	j,j	h,h	j,n	h,l	b,b	0,0	b,f	0,d
l	l,l	l,n	h,h	h,j	d,d	d,f	0,0	0,b
n	n,n	l,l	j,j	h,h	f,f	d,d	b,b	0,0

carry to be added on to the remaining digits of A is found by table lookup. The carry is given first, then the answer digit itself. The row index of Table 2 is the A digit and the column index the B digit. A worked example for subtraction follows:

$$\begin{aligned}
 fbd + X &= fj0 \quad \text{Subtraction tables gives } d,d, \\
 &\quad X = \dots d \\
 fb + d + X' &= fj \quad \text{Subtraction table gives } b,j, \\
 ff + X' &= fj \quad \text{Subtraction table gives } b,j, \\
 &\quad X = \dots jd \\
 f + b + X'' &= fj \\
 bd + X'' &= fj \quad \text{Subtraction table gives } d,n, \\
 &\quad X = \dots njd \\
 b + d + X''' &= f \\
 f + X''' &= f \quad \text{Subtraction table gives } 0,0, \\
 &\quad X = \dots 0njd \\
 0 + X''' &= 0 \quad \text{Subtraction tables gives } 0,0 \\
 &\quad X = \dots 00njd
 \end{aligned}$$

The iterations of the equation are now identical, and the answer is:

$$\dots 00000njd = njd$$

Assuming that this algorithm is identical to vector subtraction, with pixel addresses defined as in the section on 'Addressing the complete Euclidean space', we are

assured that the algorithm provides a repeating sequence of no more than one digit for whole-pixel addresses. The identity of the algorithm with vector subtraction can be proved by separating each pixel address into its Cartesian components, as described by Gargantini,¹ and noting that the algorithm is then equivalent to two's complement subtraction for each Cartesian component.

5. QUATERNION ALGEBRA

We would like to extend our arithmetic so that it encompasses a multiplication also. The multiplication should have a simple geometric meaning that will enable us to provide Tesseral algorithms easily for any geometric transform we may want. In two dimensions this is done by making the multiplication correspond to that of the complex numbers, so that the geometric transform is rotate and scale. It can then be shown that, provided we can also perform the complex conjugate operation, quite general geometric transforms¹¹ can be programmed using algorithms which rest on the Tesseral addition, subtraction, multiplication, and division. Guided by this, we demand that the 3-dimensional multiplication should correspond to a 3-dimensional rotation and scaling. However, none exists, and we must go to 4 dimensions before we find the quaternions which do have this property. The quaternions have already had advocates who suggest their use in computer graphics.^{12,13} We first explore the algebra of the quaternions, and then turn to their implementation on the Tesseral hypercube, and its subset, the Tesserally extended linear octree.

A typical quaternion, Q , consists of four linearly independent components, r , s , u , and v , each of which are scalars. It may be written:

$$Q = r*t_0 + s*t_1 + u*t_2 + v*t_3$$

t_0 is a scalar and is equal to 1 as far as quaternion algebra is concerned. t_1 , t_2 , and t_3 are scalars, and we now explore their algebra. They obey algebraic rules that include those of the complex numbers as a subset. However, in the quaternion extension, multiplication is not commutative, i.e. t_x*t_y is not equal to t_y*t_x . $r*t_0$ is called the scalar part of Q ¹⁴ and Tesseral arithmetic with numbers that are linearly dependent on t_0 alone is identical to ordinary arithmetic to the appropriate base, which is 2 for the Tesseral arithmetic described in this paper. There is a general quaternion addition and subtraction. The Tesseral addition and subtraction explained above for the octree is a subgroup. There is a multiplication and a division. The multiplication rules for the three components t_1 , t_2 , t_3 are:

$$\begin{aligned} t_x*t_x &= -1 \\ t_x*t_y &= t_z \quad \text{for } x,y \text{ cyclic} \\ t_y*t_x &= -t_z \end{aligned}$$

It can be seen that each subset $\{t_0, t_x\}$ behaves like the complex numbers, with t_x playing the role of i . In the algebra presented below, $+$, $-$, $*$, and $/$ refer to the quaternion operations of add, subtract, multiply and divide. We consider the quaternion Q :

$$Q = r + s*t_1 + u*t_2 + v*t_3 \quad r, s, u, v \text{ scalar}$$

The scalar part of Q , $S(Q)$ is:

$$r$$

The rest of the quaternion is jointly called the vector part.¹⁴ The vector part of Q , $V(Q)$ is:

$$s*t_1 + u*t_2 + v*t_3$$

Its quaternion conjugate, $qc(Q)$ is:

$$S(Q) - V(Q)$$

Its magnitude, $\text{mag}(Q)$, a scalar number, is:

$$(qc(Q)*Q)^{\frac{1}{2}}$$

Its multiplicative inverse Q^{-1} (see appendix A.3) is

$$qc(Q) / \text{mag}(Q)^2$$

The four components t_0 , t_1 , t_2 and t_3 can be mapped onto unit vectors along four orthogonal axes. The Tesserally extended octree is only 3-dimensional, and the space it occupies is defined to be that spanned by the vector part of a quaternion. This is called imaginary space by extension of the term for the imaginary 1-dimensional space of the complex numbers. The quaternion addition and subtraction then map onto vector addition and subtraction. Proofs for the following assertions in this paragraph are to be found in Appendices A.5–A.8. The multiplication corresponds to the operation of rotate and scale. Each quaternion vector rotates every other by the same angle and scales it by the same amount. Q scales by $\text{mag}(Q)$ and rotates by an angle $\text{ang}(Q)$ such that:

$$\tan\{\text{ang}(Q)\} = (-V(Q)*V(Q))^{\frac{1}{2}} / S(Q)$$

It rotates by this angle in two different planes:

(i) in the plane (PV) at right-angles to the vector $V(Q)$ in imaginary space (see Fig. 2a, which illustrates $PV(Q)$ for the quaternion $r + s*t_1$);

(ii) in the plane (PS) defined by vector $V(Q)$ and vector t_0 (see Fig. 2b, which illustrates $PS(Q)$ for the quaternion $r + s*t_1$).

We introduce a second quaternion P . The properties

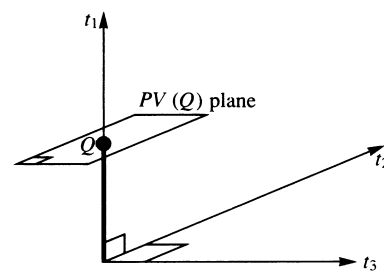


Fig. 2(a). The vector part of a quaternion, Q , lying along the t_1 -axis in imaginary space, and its plane $PV(Q)$ parallel to the plane defined by t_2 and t_3 .

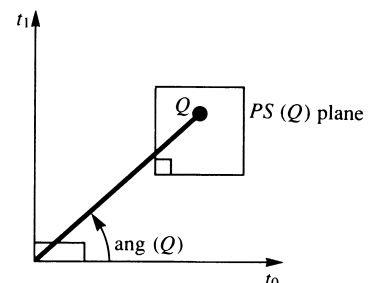


Fig. 2(b). The same quaternion, Q , shown in its plane $PS(Q)$ parallel to the plane defined by t_1 and t_0 .

we are about to remark on hold for any two general quaternions P and Q , but for the purpose of illustration suppose:

$$P = w + x*t_3$$

where w and x are scalars. The plane $PV(P)$ for P is defined by the vectors t_1 and t_2 illustrated in Fig. 3a and $PS(P)$ is defined by the vectors t_3 and t_0 illustrated in Fig. 3b. We shall define the arbitrary sign of the angle $\text{ang}(Q)$

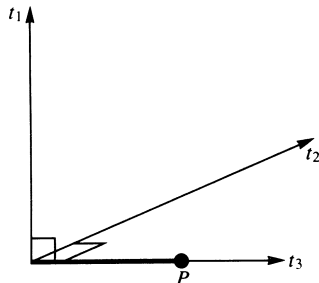


Fig. 3(a). Another quaternion, P , whose vector part lies along t_3 .

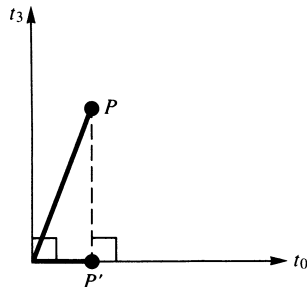


Fig. 3(b). The same quaternion, P , shown in its plane $PS(P)$, together with its projection, P' , into the plane $PS(Q)$ (i.e. that defined by t_1 and t_0).

to be such that we can say that the quaternion Q , when it pre-multiplies P , rotates P by an angle q in $PV(Q)$ and $PS(Q)$. For the particular P and Q that we have chosen as exemplars, this is illustrated in Fig. 4a for $PV(Q)$ and

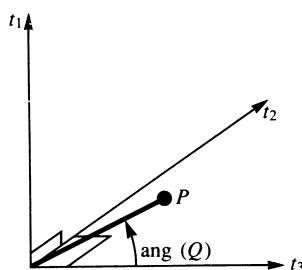


Fig. 4(a). The vector part of P is rotated by $\text{ang}(Q)$ in the plane $PV(Q)$, i.e. that defined by t_2 and t_3 .

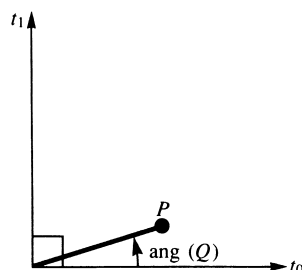


Fig. 4(b). The projection of P into $PS(Q)$ is rotated by $\text{ang}(Q)$ in $PS(Q)$, i.e. the plane defined by t_1 and t_0 .

4b for $PS(Q)$. However, the quaternion multiplication:

$$P*Q^{-1}$$

rotates P by an angle q in $PV(Q)$ and an angle $-q$ in $PS(Q)$.

So that:

$$Q*P*Q^{-1}$$

rotates P by an angle $2*q$ in $PV(Q)$ and zero in $PS(Q)$. Thus if $S(P)$ was zero to start with it will remain so.

We now consider Q and $Q^{\frac{1}{2}}$, and proofs for the following assertions in this paragraph are to be found in Appendix A.9. $V(Q)$ and $V(Q^{\frac{1}{2}})$ have an identical orientation in imaginary space, and thus so also does PV . If Q rotates by q then $Q^{\frac{1}{2}}$ rotates by $q/2$ in the same planes PV and PS . Also if Q scales by an amount s , then $Q^{\frac{1}{2}}$ scales by an amount $s^{\frac{1}{2}}$.

$qc(Q)$ shares PV with Q and Q^{-1} . $qc(Q)$ rotates by the same angle q with the same signs as Q^{-1} , but it scales by the same amount as Q , namely s .

For these reasons, we can see that if:

$$Q*P = B$$

then:

$$Q^{\frac{1}{2}}*P*qc(Q^{\frac{1}{2}}) = B$$

and, most importantly, the operator:

$$Q^{\frac{1}{2}}*[]*qc(Q^{\frac{1}{2}})$$

will not only scale all quaternions by the amount s , but will also rotate them by an angle q within imaginary space alone. This therefore is the operator we require to map every point in imaginary space to another point in imaginary space by means of a 3-dimensional rotation and scaling. Suppose, as may be the case in image processing, that we want to rotate and scale our image so that point, or equivalently, quaternion, P in imaginary space falls on point B in imaginary space. We calculate Q from its equality with $B*P^{-1}$, then we calculate $Q^{\frac{1}{2}}$ and $qc(Q^{\frac{1}{2}})$. We can then apply the operator above to all the image points, and the desired mapping is achieved. Should we want to rotate about a point O instead of the origin, the operator becomes:

$$(Q^{\frac{1}{2}})*([]-O)*qc(Q^{\frac{1}{2}})+O$$

It will be shown that it is possible to carry out all the required calculations using Tesseral techniques that work directly on the addresses of the octtree.

6. ADDRESSING THE HYPERCUBE, ADDITION AND SUBTRACTION

Table 3 relates the hypercube digits to Cartesian values and the algebraic symbols used in the previous section. It can be seen that the octtree tesseral addresses, in imaginary space, form a subset. We map vector addition onto Tesseral addition and produce Table 4, the addition table for Tesseral hypercube digits. Addition is performed in the same way as for the octtree, and a subtraction table, Table 5, is also provided. It is used for subtraction in an identical way to the octtree digit subtraction table, Table 2.

7. MULTIPLICATION

Table 6 is the multiplication table for digits. The row

index pre-multiplies the column index. We note that digit a is the multiplicative unity. Multi-digit multiplication is performed by using a carry in a fashion exactly analogous

Table 3. Hypercube addressing

Digit	Cartesian value	Component
0	(0, 0, 0, 0)	0
<i>a</i>	(1, 0, 0, 0)	t_0
<i>b</i>	(0, 1, 0, 0)	t_1
<i>c</i>	(1, 1, 0, 0)	—
<i>d</i>	(0, 0, 1, 0)	t_3
<i>e</i>	(1, 0, 1, 0)	—
<i>f</i>	(0, 1, 1, 0)	—
<i>g</i>	(1, 1, 1, 0)	—
<i>h</i>	(0, 0, 0, 1)	t_2
<i>i</i>	(1, 0, 0, 1)	—
<i>j</i>	(0, 1, 0, 1)	—
<i>k</i>	(1, 1, 0, 1)	—
<i>l</i>	(0, 0, 1, 1)	—
<i>m</i>	(1, 0, 1, 1)	—
<i>n</i>	(0, 1, 1, 1)	—
<i>o</i>	(1, 1, 1, 1)	—

to ordinary arithmetic to any ordinary base. We provide a worked example.

	f	b	d^*	
	n	j	d	
$\langle a$	a	a	a	a
$\langle h$	h	h	h	h
$\langle i$	i	i	i	i
$\langle b$	b	b	b	b
$\langle a$	a	a	a	a
$\langle c$	c	c	c	c
$\langle c$	c	c	c	c
$\langle i$	i	i	i	i
$\langle c$	c	c	c	c
$\langle h$	h	h	h	h
$\langle i$	i	i	i	i
$\langle b$	b	b	b	b
$\langle a$	a	a	a	a
$\langle c$	c	c	c	c
$\langle c$	c	c	c	c
$\langle i$	i	i	i	i
$\langle c$	c	c	c	c
$\langle h$	h	h	h	h
$\langle i$	i	i	i	i
$\langle b$	b	b	b	b
$\langle a$	a	a	a	a
$\langle c$	c	c	c	c
$\langle c$	c	c	c	c
$\langle i$	i	i	i	i
$\langle c$	c	c	c	c
$\langle h$	h	h	h	h
$\langle i$	i	i	i	i
$\langle b$	b	b	b	b
$\langle a$	a	a	a	a
$\langle c$	c	c	c	c
$\langle c$	c	c	c	c
$\langle i$	i	i	i	i
$\langle c$	c	c	c	c
$\langle h$	h	h	h	h
$\langle i$	i	i	i	i
$\langle b$	b	b	b	b
$\langle a$	a	a	a	a
$\langle c$	c	c	c	c
$\langle c$	c	c	c	c
$\langle i$	i	i	i	i
$\langle c$	c	c	c	c
$\langle h$	h	h	h	h
$\langle i$	i	i	i	i
$\langle b$	b	b	b	b
$\langle a$	a	a	a	a
$\langle c$	c	c	c	c
$\langle c$	c	c	c	c
$\langle i$	i	i	i	i
$\langle c$	c	c	c	c
$\langle h$	h	h	h	h
$\langle i$	i	i	i	i
$\langle b$	b	b	b	b
$\langle a$	a	a	a	a
$\langle c$	c	c	c	c
$\langle c$	c	c	c	c
$\langle i$	i	i	i	i
$\langle c$	c	c	c	c
$\langle h$	h	h	h	h
$\langle i$	i	i	i	i
$\langle b$	b	b	b	b
$\langle a$	a	a	a	a
$\langle c$	c	c	c	c
$\langle c$	c	c	c	c
$\langle i$	i	i	i	i
$\langle c$	c	c	c	c
$\langle h$	h	h	h	h
$\langle i$	i	i	i	i
$\langle b$				

Table 4. Hypercube digit addition

+	0	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0	00	0a	0b	0c	0d	0e	0f	0g	0h	0i	0j	0k	0l	0m	0n	0o
a	0a	a0	0c	ab	0e	ad	0g	af	0i	ah	0k	aj	0m	al	0o	an
b	0b	0c	b0	ba	0f	0g	bd	be	0j	0k	bh	bi	0n	0o	bl	bm
c	0c	ab	ba	c0	0g	af	be	cd	0k	aj	bi	ch	0o	an	bm	cl
d	0d	0e	0f	0g	d0	da	db	dc	0l	0m	0n	0o	dh	di	dj	dk
e	0e	ad	0g	af	da	e0	dc	eb	0m	al	0o	an	di	eh	dk	ej
f	0f	0g	bd	be	db	dc	f0	fa	0n	0o	bl	bm	dj	dk	fh	fi
g	0g	af	be	cd	dc	eb	fa	g0	0o	an	bm	cl	dk	ej	fi	gh
h	0h	0i	0j	0k	0l	0m	0n	0o	h0	ha	hb	hc	hd	he	hf	hg
i	0i	ah	0k	aj	0m	al	0o	an	ha	i0	hc	ib	he	id	hg	if
j	0j	0k	bh	bi	0n	0o	bl	bm	hb	hc	j0	ja	hf	hg	jd	je
k	0k	aj	bi	ch	0o	an	bm	cl	hc	ib	ja	k0	hg	if	je	kd
l	0l	0m	0n	0o	dh	di	dj	dk	hd	he	hf	hg	l0	la	lb	lc
m	0m	al	0o	an	di	eh	dk	ej	he	id	hg	if	la	m0	lc	mb
n	0n	0o	bl	bm	dj	dk	fh	fi	hf	hg	jd	je	lb	lc	n0	na
o	0o	an	bm	cl	dk	ej	fi	gh	hg	if	je	kd	lc	mb	na	o0

Table 5. Hypercube digit subtraction

+	0	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>	<i>i</i>	<i>j</i>	<i>k</i>	<i>l</i>	<i>m</i>	<i>n</i>	<i>o</i>
0	0,0	0, <i>a</i>	0, <i>b</i>	0, <i>c</i>	0, <i>d</i>	0, <i>e</i>	0, <i>f</i>	0, <i>g</i>	0, <i>h</i>	0, <i>i</i>	0, <i>j</i>	0, <i>k</i>	0, <i>l</i>	0, <i>m</i>	0, <i>n</i>	0, <i>o</i>
<i>a</i>	<i>a,a</i>	0,0	<i>a,c</i>	0, <i>b</i>	<i>a,e</i>	0, <i>d</i>	<i>a,g</i>	0, <i>f</i>	<i>a,i</i>	0, <i>h</i>	<i>a,k</i>	0, <i>j</i>	<i>a,m</i>	0, <i>l</i>	<i>a,o</i>	0, <i>n</i>
<i>b</i>	<i>b,b</i>	<i>b,c</i>	0,0	0, <i>a</i>	<i>b,f</i>	<i>b,g</i>	0, <i>d</i>	<i>b,e</i>	<i>b,j</i>	<i>b,k</i>	0, <i>h</i>	0, <i>i</i>	<i>b,n</i>	<i>b,o</i>	0, <i>l</i>	0, <i>m</i>
<i>c</i>	<i>c,c</i>	<i>b,b</i>	<i>a,c</i>	0,0	<i>c,g</i>	<i>b,f</i>	<i>a,e</i>	0, <i>d</i>	<i>c,k</i>	<i>b,j</i>	<i>a,i</i>	0, <i>h</i>	<i>c,o</i>	<i>b,n</i>	<i>a,m</i>	0, <i>l</i>
<i>d</i>	<i>d,d</i>	<i>d,e</i>	<i>d,f</i>	<i>d,g</i>	0,0	0, <i>a</i>	0, <i>b</i>	0, <i>c</i>	<i>d,l</i>	<i>d,m</i>	<i>d,n</i>	<i>d,o</i>	0, <i>h</i>	0, <i>i</i>	0, <i>j</i>	0, <i>k</i>
<i>e</i>	<i>e,e</i>	<i>d,d</i>	<i>e,g</i>	<i>d,f</i>	<i>a,a</i>	0,0	<i>a,c</i>	0, <i>b</i>	<i>e,m</i>	<i>d,l</i>	<i>e,o</i>	<i>d,n</i>	<i>a,i</i>	0, <i>h</i>	<i>a,k</i>	0, <i>j</i>
<i>f</i>	<i>f,f</i>	<i>f,g</i>	<i>d,d</i>	<i>d,e</i>	<i>b,b</i>	<i>b,c</i>	0,0	0, <i>a</i>	<i>f,n</i>	<i>f,o</i>	<i>d,l</i>	<i>d,m</i>	<i>b,j</i>	<i>b,k</i>	0, <i>h</i>	0, <i>i</i>
<i>g</i>	<i>g,g</i>	<i>f,f</i>	<i>e,e</i>	<i>d,d</i>	<i>c,c</i>	<i>b,b</i>	<i>a,a</i>	0,0	<i>g,o</i>	<i>f,n</i>	<i>e,m</i>	<i>d,l</i>	<i>c,k</i>	<i>b,j</i>	<i>a,i</i>	0, <i>h</i>
<i>h</i>	<i>h,h</i>	<i>h,i</i>	<i>h,j</i>	<i>h,k</i>	<i>h,l</i>	<i>h,m</i>	<i>h,n</i>	<i>h,o</i>	0,0	0, <i>a</i>	0, <i>b</i>	0, <i>c</i>	0, <i>d</i>	0, <i>e</i>	0, <i>f</i>	0, <i>g</i>
<i>i</i>	<i>i,i</i>	<i>h,h</i>	<i>i,k</i>	<i>h,j</i>	<i>i,m</i>	<i>h,l</i>	<i>i,o</i>	<i>h,n</i>	<i>a,a</i>	0,0	<i>a,c</i>	0, <i>b</i>	<i>a,e</i>	0, <i>d</i>	<i>a,g</i>	0, <i>f</i>
<i>j</i>	<i>j,j</i>	<i>j,k</i>	<i>h,h</i>	<i>h,i</i>	<i>j,n</i>	<i>j,o</i>	<i>h,l</i>	<i>h,m</i>	<i>b,b</i>	<i>b,c</i>	0,0	0, <i>a</i>	<i>b,f</i>	<i>b,g</i>	0, <i>d</i>	0, <i>e</i>
<i>k</i>	<i>k,k</i>	<i>j,j</i>	<i>i,i</i>	<i>h,h</i>	<i>k,o</i>	<i>j,n</i>	<i>i,m</i>	<i>h,l</i>	<i>c,c</i>	<i>b,b</i>	<i>a,a</i>	0,0	<i>c,g</i>	<i>b,f</i>	<i>a,e</i>	0, <i>o</i>
<i>l</i>	<i>l,l</i>	<i>l,m</i>	<i>l,n</i>	<i>l,o</i>	<i>h,h</i>	<i>h,i</i>	<i>h,j</i>	<i>h,k</i>	<i>d,d</i>	<i>d,e</i>	<i>d,f</i>	<i>d,g</i>	0,0	0, <i>a</i>	0, <i>b</i>	0, <i>c</i>
<i>m</i>	<i>m,m</i>	<i>l,l</i>	<i>m,o</i>	<i>l,n</i>	<i>i,i</i>	<i>h,h</i>	<i>i,k</i>	<i>h,j</i>	<i>e,e</i>	<i>d,d</i>	<i>e,g</i>	<i>d,f</i>	<i>a,a</i>	0,0	<i>a,c</i>	0, <i>b</i>
<i>n</i>	<i>n,n</i>	<i>n,o</i>	<i>l,l</i>	<i>l,m</i>	<i>j,j</i>	<i>j,k</i>	<i>h,h</i>	<i>h,i</i>	<i>f,f</i>	<i>f,g</i>	<i>d,d</i>	<i>d,e</i>	<i>b,b</i>	<i>b,c</i>	0,0	0, <i>a</i>
<i>o</i>	<i>o,o</i>	<i>n,n</i>	<i>m,m</i>	<i>l,l</i>	<i>k,k</i>	<i>j,j</i>	<i>i,i</i>	<i>h,h</i>	<i>g,g</i>	<i>f,f</i>	<i>e,e</i>	<i>d,d</i>	<i>c,c</i>	<i>b,b</i>	<i>a,a</i>	0,0

Table 6. Hypercube digit multiplication

*	0	a	b	c	d	e	f	g
a	<0000	<000a	<000b	<000c	<000d	<000e	<000f	<000g
b	<0000	<000b	<aaaa	<aaac	<hhhh	<hhhj	<iiii	<iiik
c	<0000	<000c	<aaac	<00b0	<hhhl	<hhho	<iiio	<hhjl
d	<0000	<000d	<000h	<ool	<aaaa	<aaae	<aaai	<aaam
e	<0000	<000e	<000j	<000o	<aaae	<00d0	<aaao	<00dj
f	<0000	<000f	<aaai	<aaao	<iiii	<iiio	<aaa0	<aaaf
g	<0000	<000g	<aaak	<00bl	<iiim	<hhlj	<aaaf	<aaga
h	<0000	<000h	<ddd	<ddd	<000b	<000j	<dddf	<dddn
i	<0000	<000i	<dddf	<dddo	<000f	<000o	<00bo	<00bi
j	<0000	<000j	<eeee	<eeeo	<hhhj	<00b0	<mmmo	<eege
k	<0000	<000k	<eeeg	<ddfl	<hhhn	<00be	<iiki	<0b00
l	<0000	<000l	<ddd	<00h0	<aaac	<aaao	<eeeo	<aaic
m	<0000	<000m	<dddn	<00hc	<aaag	<00dj	<aaci	<00jd
n	<0000	<000n	<eeem	<aaic	<iiik	<aace	<eeef	<aach
o	<0000	<000o	<eeeo	<00j0	<iiio	<00f0	<aac0	<acam
*	h	i	j	k	l	m	n	o
a	<000h	<000i	<000j	<000k	<000l	<000m	<000n	<000o
b	<000d	<000f	<aaae	<aaag	<hhhl	<hhhn	<iiim	<iiio
c	<000l	<000o	<aaao	<00bl	<00d0	<00dc	<aaec	<00fo
d	<bbbb	<bbbf	<bbbj	<bbbn	<cccc	<cccg	<ccck	<ccco
e	<bbbj	<bbbo	<00h00	<00he	<ccco	<bbfj	<aaie	<00l0
f	<bbbf	<00d0	<ccco	<aaei	<kkko	<iiim	<cccf	<aae0
g	<bbbn	<00di	<aaie	<00lb	<ccgc	<0d00	<aaeh	<aeak
h	<aaaa	<aaai	<eeee	<eeem	<aaac	<aaak	<eeeg	<eeeo
i	<aaai	<00h0	<eeeo	<ddlf	<aaao	<00hf	<aaci	<00j0
j	<aaae	<aaao	<aaao	<aaaj	<iiio	<aace	<iiij	<aac0
k	<aaam	<00hf	<aaaj	<aaka	<aaec	<00fh	<aacd	<acam
l	<cccc	<ccco	<gggo	<cckc	<aaa0	<aaal	<eeel	<aaio
m	<ccck	<bbjf	<eeem	<0h00	<aaal	<aama	<aaib	<aiaa
n	<cccg	<aaei	<cccj	<aaid	<iiil	<aaeb	<aa0a	<aa0o
o	<ccco	<00l0	<aaio	<aiaa	<aae0	<aeak	<aa0o	<aa0o

8. P-ADIC DIVISION

Division is necessary in the course of calculating, for example, how to rotate and scale an image such that point P falls on point B , as described in Section 5.

If J and K are known, X unknown, the division equation may be written:

$$\begin{aligned} X &= J^{-1} * K \quad \text{or} \quad J * X = K \\ X &= K * J^{-1} \quad \text{or} \quad X * J = K \end{aligned} \quad (5)$$

and we seek the solution of these. In a method similar to that used for subtraction, in an extension of the method for division of P-Adic numbers to some base,⁹ we find the digits of X from right to left, starting with the least significant digit of X , and using the second form of the equations above. Notionally, we use the multiplication table (Table 6) to find a last digit for X that, when operated on by the last digit of J , will produce a result (R) with the last digit equal to the last digit of K . The last digits of R and K are then additively cancelled, and the algorithm is iterated to produce the next least significant digit of X . We proceed in this manner until we find the same sequence of digits repeated indefinitely.

We should like to be able to produce a division table for digits analogous to the subtraction table, Table 5. This is not possible, however, because there are some digits that cannot be changed into a Tesseral number ending in some other arbitrary digit by pre- or post-multiplication by any digit whatever. For example, a number ending in 0 cannot be changed into a number

ending in any other digit using multiplication alone. There are other digits besides 0 which create similar difficulties. For example, study of the multiplication table, Table 6, shows us that we cannot turn a Tesseral number ending in l into one ending in a by pre- or post-multiplication using any other digit. Our strategy is to convert a number ending in 0 or l into another number which is more amenable.

We divide by $a0$ in the case of a number ending in 0, by shifting the radix point to the left by one place. If the number ends in several zeroes we repeat until the number ends in some other digit. If J in equation (5) ends in 0 we divide J by $a0$ and replace J by $J/a0$. We have naturally altered the value of X . We have multiplied the original X by $a0$. However, we may solve equation (5) for $X*a0$ and, having found $X*a0$ by the method already suggested, then divide the answer, $X*a0$, by $a0$ to regain X . This division is done by shifting the radix point one place to the left, and will thus in general result in the answer, X , having a non-zero digit to the right of the radix point.

We now consider our second example: if J ends in l we can convert J to a number ending in $a0$ by multiplying J by l . To preserve the equality of equation (5) we also multiply K by l . After this process J ends in $a0$, and we divide by $a0$ to obtain a number ending in a as previously described.

By contrast, we consider the case when J ends in digit a . Digit a can be changed into any other digit v by multiplication by v and thus we will always be able to proceed to find the next digit of X if J ends in a . Although

Table 7. Hypercube digit division (pre-multiplication)

/	<i>a</i>	<i>b</i>	<i>d</i>	<i>h</i>
0	0,0	0,0	0,0	0,0
<i>a</i>	0, <i>a</i>	<i>a</i> , <i>b</i>	<i>a</i> , <i>d</i>	<i>a</i> , <i>h</i>
<i>b</i>	0, <i>b</i>	0, <i>a</i>	0, <i>h</i>	<i>b</i> , <i>d</i>
<i>c</i>	0, <i>c</i>	<i>a</i> , <i>c</i>	<i>a</i> , <i>l</i>	<i>c</i> , <i>l</i>
<i>d</i>	0, <i>d</i>	<i>d</i> , <i>h</i>	0, <i>a</i>	0, <i>b</i>
<i>e</i>	0, <i>e</i>	<i>e</i> , <i>j</i>	<i>a</i> , <i>e</i>	<i>a</i> , <i>j</i>
<i>f</i>	0, <i>f</i>	<i>d</i> , <i>i</i>	0, <i>i</i>	<i>b</i> , <i>f</i>
<i>g</i>	0, <i>g</i>	<i>e</i> , <i>k</i>	<i>a</i> , <i>m</i>	<i>c</i> , <i>n</i>
<i>h</i>	0, <i>h</i>	0, <i>d</i>	<i>h</i> , <i>b</i>	0, <i>a</i>
<i>i</i>	0, <i>i</i>	<i>a</i> , <i>f</i>	<i>i</i> , <i>f</i>	<i>a</i> , <i>i</i>
<i>j</i>	0, <i>j</i>	0, <i>e</i>	<i>h</i> , <i>j</i>	<i>b</i> , <i>e</i>
<i>k</i>	0, <i>k</i>	<i>a</i> , <i>g</i>	<i>i</i> , <i>n</i>	<i>c</i> , <i>m</i>
<i>l</i>	0, <i>l</i>	<i>d</i> , <i>l</i>	<i>h</i> , <i>c</i>	0, <i>c</i>
<i>m</i>	0, <i>m</i>	<i>e</i> , <i>n</i>	<i>i</i> , <i>g</i>	<i>a</i> , <i>k</i>
<i>n</i>	0, <i>n</i>	<i>d</i> , <i>m</i>	<i>h</i> , <i>k</i>	<i>b</i> , <i>g</i>
<i>o</i>	0, <i>o</i>	<i>e</i> , <i>o</i>	<i>i</i> , <i>o</i>	<i>c</i> , <i>o</i>

the number occupying the notional place of *K* in equation (5) will change as we iterate, that occupying the place of *J* does not alter in value. If it starts by ending in *a*, it always ends in *a*, and we have therefore solved the problem for all iterations. The strategy is therefore to convert *J* so that it ends in a digit like *a* which can be converted to any other digit by multiplication by some other digit. The digits for which this is true are *a*, *b*, *d*, *h*. For completeness the next paragraph prescribes in a more formal fashion how to turn every other digit into one of these.

Our aim is to convert the last digit of *J* into one which allows us to write a simple division table, Tables 7 and 8, for any last digit of *K*. During the iterations of the division algorithm the value of *K* changes, but that of *J* does not. *J* is converted to the required forms as a one-off exercise at the start, and then each iteration of the algorithm starts with a simple table look-up involving no extra multiplications or radix-point shifts. For rows/columns *a*, *b*, *d*, *h* and *n*, in Table 6, the same condition as for addition holds. For any given digit *x*, we can always find a unique digit *y*, which, when multiplied by *x*, produces a number ending in any given digit *z*. We call the directed pair (*x*,*z*) soluble if and only if this is so. Any number, *N*, ending in digit *g*, *k*, *m*, or *n* can be converted

Table 8. Hypercube digit division (post-multiplication)

/	<i>a</i>	<i>b</i>	<i>d</i>	<i>h</i>
0	0,0	0,0	0,0	0,0
<i>a</i>	0, <i>a</i>	<i>a</i> , <i>b</i>	<i>a</i> , <i>d</i>	<i>a</i> , <i>h</i>
<i>b</i>	0, <i>b</i>	0, <i>a</i>	<i>b</i> , <i>h</i>	0, <i>d</i>
<i>c</i>	0, <i>c</i>	<i>a</i> , <i>c</i>	<i>c</i> , <i>l</i>	<i>a</i> , <i>l</i>
<i>d</i>	0, <i>d</i>	0, <i>h</i>	0, <i>a</i>	<i>d</i> , <i>b</i>
<i>e</i>	0, <i>e</i>	<i>a</i> , <i>j</i>	<i>a</i> , <i>e</i>	<i>e</i> , <i>j</i>
<i>f</i>	0, <i>f</i>	0, <i>i</i>	<i>b</i> , <i>i</i>	<i>d</i> , <i>f</i>
<i>g</i>	0, <i>g</i>	<i>a</i> , <i>k</i>	<i>c</i> , <i>m</i>	<i>e</i> , <i>n</i>
<i>h</i>	0, <i>h</i>	<i>h</i> , <i>d</i>	0, <i>b</i>	0, <i>a</i>
<i>i</i>	0, <i>i</i>	<i>i</i> , <i>f</i>	<i>a</i> , <i>f</i>	<i>a</i> , <i>i</i>
<i>j</i>	0, <i>j</i>	<i>h</i> , <i>e</i>	<i>b</i> , <i>j</i>	0, <i>e</i>
<i>k</i>	0, <i>k</i>	<i>i</i> , <i>g</i>	<i>c</i> , <i>n</i>	<i>a</i> , <i>m</i>
<i>l</i>	0, <i>l</i>	<i>h</i> , <i>l</i>	0, <i>c</i>	<i>d</i> , <i>c</i>
<i>m</i>	0, <i>m</i>	<i>i</i> , <i>n</i>	<i>a</i> , <i>g</i>	<i>e</i> , <i>k</i>
<i>n</i>	0, <i>n</i>	<i>h</i> , <i>m</i>	<i>b</i> , <i>k</i>	<i>d</i> , <i>g</i>
<i>o</i>	0, <i>o</i>	<i>i</i> , <i>o</i>	<i>c</i> , <i>o</i>	<i>e</i> , <i>o</i>

to some number, *M*, ending, respectively, in digit *h*, *d*, *b*, or *a* by pre- or post-multiplication with number *L*, $L = n$. Numbers, *N*, ending in *c*, *e*, *i*, *f*, *j*, *l*, can be converted to some number, *M*, ending, respectively, in digits *b0*, *d0*, *h0*, *a0*, *a0*, *a0* by pre- or post-multiplication with a number, *L*, consisting of one digit, the last digit of *N*. So *L* will be equal to digit *c*, *e*, *i*, *f*, *j*, *l*, respectively. The radix point in *M* is then shifted one digit to the left, and the answer, when found must have the radix point shifted one digit to the left also. If *N* ends in digit *o* it can be converted to some number, *M*, ending in digit *d* by post-multiplication by a number *L*, $L = m*n$. Finally, a number, *N*, whose last digit is 0, must always be dealt with by shifting the radix point. We can thus always convert our equation into one in which *J* has a last digit which forms a soluble pair with any last digit of *K* by multiplying both *J* and *K* by a suitable number, and then, if necessary, dividing *J* by some power of *a0* by shifting the radix point in *J* to the left. Our division table thus only needs to cater for the conversion of *a*, *b*, *d*, *h* to any digit.

Table 7, for digit division, answers the question: 'how do you convert *a*, *b*, *d*, *h* (columns) to any of the other digit (rows) by pre-multiplication?' Table 8, also for digit division, answers the question: 'how do you convert *a*, *b*, *d*, *h* (columns) to any of the other digits (rows) by post-multiplication?' The digit on the right of each column is the answer, and the digit to the left is the negative of the carry whose use is explained by the example below. It is the negative of the digit given in the multiplication table, Table 6.

We follow with an example of hypercube division. First we perform one iteration of the solution using only the multiplication table, Table 6, and writing out the steps in detail, then we will perform all the iterations necessary for a complete solution using the division table, Table 7.

$$\begin{aligned}
 X * njd &= \langle cf00dca && \text{Table 6 shows that } d*d = \langle a \\
 X' * njd &= \langle cf00dca && \text{Set } X = X'd \\
 X'0 * njd + d * njd &= \langle cf00dca \\
 X'0 * njd + d * njd + d * d &= \langle cf00dca \\
 X'0 * njd + d * njd + \langle a &= \langle cf00dca \\
 X'0 * njd &= \langle cf00dca - d * njd + a \\
 &= \langle cf00dc0 - d * njd + a + a \\
 &= \langle cf00dc0 - d * njd + a0
 \end{aligned}$$

Divide both sides by *a0*

$$\begin{aligned}
 X' * njd &= \langle cf00dc - d * nj + a \\
 X' * njd &= \langle cf00dc - \langle cij + a \\
 &= \langle knhkdh && \text{Using Table 6 for } d * nj \\
 &&& \text{Using Tables 4 and 5}
 \end{aligned}$$

We now perform the same iteration again using the division table, Table 7, and go on to find a complete solution. Table 7 rather than Table 8 is used because *X* pre-multiplies rather than post-multiplies *njd*.

$$\begin{aligned}
 X * njd &= \langle cf00dca && \text{Division table 7 gives } a,d \\
 &&& X = \dots d \\
 X' * njd &= \langle cf00dc - d * nj + a \\
 &= \langle knhkdh && \text{Division table 7 gives } h,b \\
 &&& X' = \dots b, X = \dots bd \\
 X'' * njd &= \langle knhkd - b * nj + h \\
 &= \langle cbefi && \text{Division table 7 gives } i,f \\
 &&& X'' = \dots f, X = \dots fbd \\
 X''' * njd &= \langle cbef - f * nj + i \\
 &= 0 && X''' = \dots 0, X = \langle 0fbd
 \end{aligned}$$

For completeness we mention that if the answer to a division sum is fractional it must be converted to a form that can be easily mapped to the 4-dimensional space of the hypercube. The algorithm for doing this is given in reference 10.

9. FINDING THE SQUARE ROOT

The algorithm for finding the square root of a hypercube quaternion is the same as that for the 2-dimensional Tesseral arithmetic based on the complex numbers¹⁵ and performs with similar efficiency. For the worst case it takes 13 iterations to find the square-root to within 1 per cent. The mathematical background to this algorithm is given in Appendix A.10 It is a variant of Newton's method of finding a square root.

Let T be the Tesseral number of which we wish to find the square root.

(1) Replace T with a normalised T between $0.0uU$ and $0.uU$ where u is a Tesseral digit and U a string of Tesseral digits, by an even number of shifts $2*v$ where v stands for an ordinary number. Steps (3) and (4) can still be done using integer arithmetic, but the position of the radix point must be remembered.

(2) If necessary replace T with a rotated T until it is close to the line 0 to a . The method is as follows. If T begins with $\langle 0, \langle b, \langle d, \langle f, \langle h, \langle j, \langle l$ or $\langle n$, do nothing. If T begins with $\langle a, \langle c, \langle e, \langle g, \langle i, \langle k, \langle m$ or $\langle 0$, multiply it by $b*b$, i.e. $\langle a$.

(3) Perform the Tesseral calculation:

$G = (T + a) / a0$ where G is a Tesseral number.

(4) Iterate, replacing G with Tesseral $G - (G * G - T) / a0$ at each iteration until $G * G$ is equal to T within the required error.

(5) Shift G back by v . We shift back by half the number of shifts we used under step (1) because G is the square root of T . For example, if we had divided by $a00$ on carrying out step (1) we must now multiply by the square root of $a00$, namely $a0$.

(6) If T was multiplied by $b*b$ under step (2), multiply G by $\langle b$. Multiplying by $\langle b$ is equivalent to dividing by b as we can see from the Cartesian equivalent of b given in Table 3 as t_1 , and the algebra of division given in section 5. As for step (5), we only divide by b once because G is the square root of T .

10. FINDING THE QUATERNION CONJUGATE

The quaternion conjugate of each digit may be looked up in the quaternion conjugate table, Table 9. The quaternion conjugates of each digit in the number are then added together, beginning each at the column position of the original digit. For example, the quaternion conjugate of the Tesseral number abc is given by:

$$qc(abc) = a00 + \langle b0 + \langle bc$$

11. EFFICIENCY CONSIDERATIONS AND SUMMARY

The labelling scheme for the linear octtree described here is an extension of that given by Gargantini,¹ where she

Table 9. Hypercube digit conjugation

0	0
a	a
b	$\langle b$
c	$\langle bc$
d	$\langle d$
e	$\langle de$
f	$\langle f$
g	$\langle fg$
h	$\langle h$
i	$\langle hi$
j	$\langle j$
k	$\langle jk$
l	$\langle l$
m	$\langle lm$
n	$\langle n$
o	$\langle no$

demonstrates that the scheme lends itself to efficient algorithms for non-geometric manipulations of 3-dimensional data such as finding the resultant octtree obtained by overlaying two octtree data-sets. It has been shown here that the address labels of the linear octtree can be regarded as quaternion numbers, and that quaternion addition, subtraction, multiplication, division, conjugation and extraction of the square root can be done, without translating the address labels into Cartesian form, with an arithmetic that uses the individual symbols in the address labels as digits. This is quaternion Tesseral arithmetic. The geometrical transforms thus provided are vector addition, subtraction, and rotation/scaling, reflection in the line 0 to a , and scaling by the square root/halving the angle of rotation, all in 4-dimensional space. These operations can be used to provide vector addition, subtraction and rotation/scaling in the 3-dimensional space of the octtree. It has been shown that if the translation and/or rotation/scaling required is defined as a linear mapping of two points in the 3-dimensional image onto two other points, then the required transform can be defined in terms of quaternion multiplications and additions (with the Tesseral numbers A , B , and C also found by Tesseral methods). The operator is:

$$A * (B + []) * C$$

which has 1 Tesseral addition as compared with 3 in the Cartesian case and 2 Tesseral multiplications as compared with 9 in the Cartesian case. So for this transform the method might be more efficient because (a) it is not necessary to translate the linear-octtree address into Cartesian form and back into linear-octtree form at the end of the calculation, and (b) the number of arithmetic operations is fewer.

It is conjectured that the geometric transforms of vector addition, subtraction and scaling/rotation in 3-dimensional space are sufficient to provide any linear geometric transform, and that, in line with the results for the Tesseral addressing of 2-dimensional space, based on the complex numbers,¹¹ fewer arithmetic operations per general geometric transform will be required. In that case the method might be more efficient generally since we may repeat: (a) it is not necessary to translate the linear-

octtree address into Cartesian form and back into linear-octtree form at the end of the calculation, and (b) the number of arithmetic operations per general geometric transform is less. Whether the method is more efficient depends on whether efficiency algorithms can be created using the tesseral arithmetic to provide geometric transforms of interest for image processing or geographic information systems, and the method of implementing the basic tesseral arithmetic operations of $+$, $-$, $*$ and $/$. For Cartesian arithmetic these operations are provided in hardware in the Arithmetic Logical Unit and a hardware or micro-coded implementation for the tesseral operations would clearly be faster.

Acknowledgements

Thanks are due to Dylan Morgan for the use of his manuscript on quaternions,¹⁷ and his assistance in interpreting it, to William Wingate who provided references 12 and 13 on the use of quaternions in 3 dimensions, and to Fred Holroyd for checking the mathematics of an earlier draft.

This work was partly supported under NERC Contract No. F60/G6/12.

REFERENCES

1. I. Gargantini, Linear octtrees for fast processing of three-dimensional objects. *Computer Graphics and Image Processing* **20**, 365–374 (1982).
2. S. B. M. Bell, B. M. Diaz, F. C. Holroyd and M. J. Jackson, Spatially referenced methods of storing and handling raster and vector data. *Image and Vision Computing* **1** (4), 211–220 (1983).
3. S. B. M. Bell, B. M. Diaz and F. C. Holroyd, Capturing image syntax using Tesseral addressing and arithmetic. In *Digital Image Processing in Remote Sensing*, edited J. P. Muller. Taylor and Francis, Basingstoke (1988).
4. H. Samet, The quadtree and related hierarchical data structures. *ACM Computing Surveys* **16** (2), 187–260 (1984).
5. J. Weng and N. Ahuja, Octtrees of objects in arbitrary motion: representation and efficiency. *Computer Vision, Graphics, and Image Processing* **39**, 167–185 (1987).
6. H. H. Atkinson, I. Gargantini and T. R. S. Walsh, Filling by Quadrants or Octants. *Computer Vision, Graphics, and Image Processing* **33**, 138–155 (1986).
7. I. Navazo, D. Ayala and P. Brunet, A Geometric Modeller Based on the Exact Octtree Representation of Polyhedra. *Computer Graphics Forum* **5**, 91–104 (1986).
8. N. Ahuja and C. Nash, Octree Representations of Moving Objects. *Computer Vision, Graphics and Image Processing* **26**, 207–216 (1984).
9. N. Koblitz, *p-Adic Numbers, p-adic Analysis, and Zeta-Functions*. Springer-Verlag, New York (1977).
10. S. B. M. Bell, B. M. Diaz and F. C. Holroyd, Constructive tesseral inverse. In B. M. Diaz and S. B. M. Bell (eds), *Spatial Data Processing using Tesseral Methods*. Natural Environment Research Council, Swindon (1986).
11. S. B. M. Bell and F. C. Holroyd, Tesseral Algorithms. In B. M. Diaz and S. B. M. Bell (eds), *Spatial Data Processing using Tesseral Methods*. Natural Environment Research Council, Swindon (1986).
12. R. R. Martin, Rotation by quaternions. *Mathematical Spectrum* **42–48** (March 1985).
13. J. Rooney, A survey of spatial rotations about a fixed point. *Environment and Planning B* (4), 185–210 (1977).
14. W. R. Hamilton, *Elements of Quaternions*. Longman, Green, London (1899).
15. S. B. M. Bell, B. M. Diaz and F. C. Holroyd, Tesseral square roots in 2d. In B. M. Diaz and S. B. M. Bell (eds), *Spatial Data Processing using Tesseral Methods*. Natural Environment Research Council, Swindon (1986).
16. S. B. M. Bell, B. M. Diaz and F. C. Holroyd, Tesseral quaternions for 3d. In B. M. Diaz and S. B. M. Bell (eds), *Spatial Data Processing using Tesseral Methods*. Natural Environment Research Council, Swindon (1986).
17. J. D. Morgan, Personal communication (1984).

APPENDIX: QUATERNION ALGEBRA

A.1 Alternative form of the quaternion:

$$r + s*t_1 + u*t_2 + v*t_3$$

Consider the quantity:

$$Q = x * [\cos(q) + b/q * \sin(q) * t_1 + c/q * \sin(q) * t_2 + d/q * \sin(q) * t_3]$$

$$\text{where } b^2 + c^2 + d^2 = q^2$$

It can be seen that Q is a quaternion. Set:

$$r = x * \cos(q) \quad (1)$$

$$s = x * b/q * \sin(q) \quad (2)$$

$$u = x * c/q * \sin(q) \quad (3)$$

$$v = x * d/q * \sin(q) \quad (4)$$

Then:

$$Q = r + s*t_1 + u*t_2 + v*t_3$$

We have:

$$\begin{aligned} \text{mag}(Q)^2 &= (r*r + s*s + u*u + v*v) \\ &= x*x * [\cos^2(q) + (b*b + c*c + d*d)/(q*q) * \sin^2(q)] \end{aligned}$$

from equations (1) to (4)

$$= x*x$$

So

$$x = \text{mag}(Q) \quad (5)$$

Also:

$$\{s*s + u*u + v*v\} / r*r = [x*x * (b*b + c*c + d*d)/q*q * \sin^2(q)] / \{x*x * \cos^2(q)\}$$

from equations (1) to (4). Therefore:

$$\{s*s + u*u + v*v\} / r*r = \tan^2(q) \quad (6)$$

From equations (5) and (6) we may proceed in the reverse direction and determine x and $\tan(q)$ given r, s, u, v . If we define q to be the angle less than $\pi/2$ radians which satisfies (6) we may then determine b, c, d from equations (2) to (4). Thus the two forms of writing the quaternion:

$$Q = r + s*t_1 + u*t_2 + v*t_3$$

and

$$Q = x * [\cos(q) + b/q * \sin(q) * t_1 + c/q * \sin(q) * t_2 + d/q * \sin(q) * t_3]$$

are interchangeable.

The value of $\text{ang}(Q)$ as defined in the body of the paper is:

$$\text{ang}(Q) = \tan^{-1} \{[-V(Q) * V(Q)]^{1/2} / S(Q)\}$$

Therefore:

$$\tan[\text{ang}(q)] = \sin(q) / \cos(q)$$

from equations (1) to (4) and the definition of quaternion multiplication. So:

$$\text{ang}(Q) = q$$

A.2 Behaviour under rotation of imaginary axes

The quaternion algebra is invariant under a rotation of the imaginary (3-dimensional) axes. x must be invariant under such a rotation since it is the length of the 4-dimensional vector (r, s, u, v) . The invariant length of the 3-dimensional vector in imaginary space, (s, u, v) , is $x \sin(q)$, and hence q must be invariant under rotation of the imaginary axes. Quaternion addition and subtraction are obviously invariant since these map to vector addition and subtraction. We prove that multiplication is also invariant.

Let the first quaternion be:

$$Q = r_q + s_q * t_1 + u_q * t_2 + v_q * t_3$$

Let the second quaternion be:

$$P = r_p + s_p * t_1 + u_p * t_2 + v_p * t_3$$

Then $V(Q)$ is the vector (s_q, u_q, v_q) and $V(P)$ is the vector (s_p, u_p, v_p) . Let the angle between $V(Q)$ and $V(P)$ be y , again invariant under rotation. The scalar product:¹

$$\begin{aligned} V(Q)V(P) &= s_q s_p + u_q u_p + v_q v_p \\ &= x_q x_p \sin(q) \sin(p) \cos(y) \end{aligned} \quad (7)$$

and is invariant under rotation, as may be checked by elementary 3-D trigonometry. The vector product:¹

$$\begin{aligned} V(Q) \times V(P) &= (u_q v_p - u_p v_q, v_q s_p - s_q v_p, s_q u_p - s_p u_q) \\ &= x_q x_p \sin(q) \sin(p) \sin(y) * e \end{aligned} \quad (8)$$

where e is the unit vector at right angles to the plane defined by $V(Q)$ and $V(P)$, and is also invariant under rotation, as may be checked by elementary trigonometry. The quaternion product:

$$\begin{aligned} Q * P &= \{r_q r_p\} - \{s_q s_p + u_q u_p + v_q v_p\} + \{r_q [s_p t_1 \\ &+ u_p t_2 + v_p t_3]\} + \{r_p [s_q t_1 + u_q t_2 + v_q t_3]\} \\ &+ \{(u_q v_p - u_p v_q) t_1 + (v_q s_p - s_q v_p) t_2 \\ &+ (s_q u_p - s_p u_q) t_3\} \end{aligned}$$

The first term in curly brackets is a scalar and hence invariant under rotation of the imaginary axes. The second term in curly brackets is the scalar product $V(Q)V(P)$, and from (7) invariant under rotation of the imaginary axes. The third and fourth terms in curly brackets are the vectors $V(Q)$ and $V(P)$ themselves, each multiplied by a scalar, and hence invariant under rotation of the imaginary axes. The fifth term in curly brackets is the vector product $V(Q) \times V(P)$ and from (8) invariant under rotation of the imaginary axes. Hence the quaternion product is invariant under rotation of the imaginary axes.

A.3 Quaternion division and inverse of a quaternion

Quaternion division may be defined in terms of quaternion multiplication and conjugation.

$$\begin{aligned} P/Q &= P * Q^{-1} \\ &= P * qc(Q) / [\text{mag}(Q)^2] \end{aligned}$$

as may be checked by multiplying Q by Q^{-1} and finding that the result is 1. Since quaternion conjugation is invariant under rotation of the imaginary axes, so is quaternion division.

A.4 Proof that $\text{mag}(Q * P)$ is $\text{mag}(Q) * \text{mag}(P)$

Since we have proved quaternions invariant under rotation of the imaginary axes, we can choose the orientation of these to simplify the algebra for the rest of the proofs. We choose the t_1 axis to be in the direction of $V(Q)$, so:

$$\begin{aligned} Q &= x_q * [\cos(q) + b_q / q * \sin(q) * t_1] \\ &= x_q * [\cos(q) + \sin(q) * t_1] \end{aligned} \quad (9)$$

We choose the t_2 axis to be in the direction of the component of $V(P)$ at right angles to $V(Q)$, so:

$$P = x_p * [\cos(p) + b_p * \sin(p) * t_1 + c_p / p * \sin(p) * t_2] \quad (10)$$

First we prove that Q scales P by x_q upon the transform $Q * P$. $\text{mag}(Q)$ is x_q , and $\text{mag}(P)$ is x_p by equation (5).

$$\begin{aligned} Q * P &= \cos(q) * \cos(p) - b_p / p * \sin(p) * \sin(q) \\ &+ [\cos(q) * b_p / p * \sin(p) + \cos(p) * \sin(q)] * t_1 \\ &+ [c_p / p * \sin(p) * \cos(q)] * t_2 \\ &+ [c_p / p * \sin(p) * \sin(q)] * t_3 \end{aligned} \quad (11)$$

from equations (1) to (4), equations (9) and (10), and the definition of quaternion multiplication. Therefore:

$$\begin{aligned} \text{mag}(Q * P)^2 &= (x_q x_p)^2 * \{[\cos(q) \cos(p) \\ &- b_p / p * \sin^2(p)]^2 + \{\cos(q) * b_p / p * \sin(p) \\ &+ \cos(p) * \sin(q)\}^2 + \{c_p / p * \sin(p) * \cos(q)\}^2 \\ &+ \{c_p / p * \sin(p) * \sin(q)\}^2\} \end{aligned}$$

The term in square brackets simplifies to yield 1, and hence $\text{mag}(Q * P)$ is $x_q * x_p$.

A.5 Result of pre-multiplying P by Q for the plane $PS(Q)$

We will now simplify the algebra yet further, by omitting explicit mention of x_p and x_q . Their re-insertion into the algebra is trivial.

Projection of $V(P)$ along $V(Q) = b_p / p * \sin(p)$

$$\text{from equations (9) and (10).} \quad (12)$$

The imaginary axes have been labelled (t_1, t_2, t_3) . Let the fourth axis along which the scalar parts $S(Q)$ and $S(P)$ lie be labelled t_0 .

Projection of P along t_0 , i.e. in the common direction

$$S(P), S(Q), = \cos(p). \quad (13)$$

From (12) and (13) the magnitude squared of the projection of P in $PS(Q)$, the plane defined by $V(Q)$ and $S(Q)$:

$$j_0 = b_p / p * b_p / p * \sin(p) * \sin(p) + \cos(p) * \cos(p) \quad (14)$$

and the angle made by the projection of P relative to t_0 :

$$k_{op} = \tan^{-1}\{b_p/p * \tan(p)\}. \quad (15)$$

The angle made by the projection of Q in the same plane relative to t_0 :

$$k_{oq} = q \quad (16)$$

After multiplication, and from equation (11), the projection of $V(Q*P)$ along

$$V(Q) = \cos(q) * b_p/p * \sin(p) + \cos(p) * \sin(q) \quad (17)$$

The projection of $Q*P$ along $t_0 = \cos(q) * \cos(p)$

$$-b_p/p * \sin(p) * \sin(q) \quad (18)$$

From (17) and (18) the magnitude squared of the projection of $(Q*P)$ in $PS(Q)$:

$$j_n = [\cos(q) * b_p/p * \sin(p) + \cos(p) * \sin(q)]^2 + [\cos(q) * \cos(p) - b_p/p * \sin(p) * \sin(q)]^2 \quad (19)$$

On explicitly carrying out the squaring of the terms in square brackets, two terms cancel, and factorising the remaining terms and using the fact that $\cos^2 + \sin^2 = 1$, equation (19) simplifies to yield $j_n = j_o$.

The angle made by the projection of $Q*P$ in $PS(Q)$, from equation (17) and (18), and relative to t_0 , is:

$$k_n = \tan^{-1} \{ [\cos(q) * b_p/p * \sin(p) + \cos(p) * \sin(q)] / [\cos(q) \cos(p) - b_p/p * \sin(p) * \sin(q)] \} \\ = \tan^{-1} \{ [b_p * \tan(p) + \tan(q)] / [1 - b_p/p * \tan(p) * \tan(q)] \} \\ = k_{op} + k_{oq} \quad (20)$$

A.6 Result of pre-multiplying P by Q for the imaginary plane $PV(Q)$

The magnitude squared of the projection of P in $PV(Q)$, i.e. at right angles to $V(Q)$ in imaginary space, is the component along t_2 :

$$l_o = \{c_p/p * \sin(p)\}^2 \quad (21)$$

The angle made by this projection relative to t_2 is zero:

$$m_o = 0 \quad (22)$$

After multiplication, from equation (11), the magnitude squared of the projection of $(Q*P)$ in $PV(Q)$:

$$l_n = \{c_p/p * \sin(p) \cos(q)\}^2 + \{c_p/p * \sin(q) * \sin(p)\}^2 \quad (23)$$

after extracting the common term $[c_p/p * \sin(p)]^2$, this expression readily yields $l_n = l_o$. The angle made by this projection relative to t_2 is:

$$m_n = \tan^{-1} \{ [c_p/p * \sin(p) * \sin(q)] / [c_p/p * \sin(p) * \cos(q)] \} \\ = q \\ = m_o + q \quad \text{from equation (22).} \quad (24)$$

A.7 Result of post-multiplying P by Q^{-1} for the plane $PS(Q)$

$$Q^{-1} = \cos(q) - b_q/q * \sin(q) * t_1 \quad (25)$$

from Section A.3. As before:

$$P = \cos(p) + b_p/p * \sin(p) * t_1 + c_p/p * \sin(p) * t_2 \quad (26)$$

Therefore:

$$P*Q^{-1} = \cos(q) * \cos(p) + b_p/p * \sin(p) * \sin(q) \\ + [\cos(q) * b_p/p * \sin(p) - \cos(p) * \sin(q)] * t_1 \\ + [c_p/p * \sin(p) * \cos(q)] * t_2 \\ + [c_p/p * \sin(q) * \sin(p)] * t_3 \quad (27)$$

Projection of $V(P)$ along $V(Q) = b_p/p * \sin(p)$

$$\text{from equations (9) and (26).} \quad (28)$$

Projection of P along t_0 , i.e. in the common direction

$$S(P), S(Q), = \cos(p). \quad (29)$$

From (28) and (29) the magnitude squared of the projection of P in $PS(Q)$, the plane defined by $V(Q)$ and $S(Q)$:

$$j_o = b_p/p * b_p/p * \sin(p) * \sin(p) + \cos(p) * \cos(p) \quad (30)$$

and the angle made by the projection of P relative to t_0 :

$$k_{op} = \tan^{-1}\{b_p/p * \tan(p)\} \quad (31)$$

The angle made by the projection of Q in the same plane relative to t_0 :

$$k_{oq} = q \quad (32)$$

After multiplication, and from equation (27), the projection of $V(P*Q^{-1})$ along

$$V(Q) = \cos(q) * b_p/p * \sin(p) - \cos(p) * \sin(q) \quad (33)$$

The projection of $P*Q^{-1}$ along

$$t_0 = \cos(q) * \cos(p) + b_p/p * \sin(p) * \sin(q) \quad (34)$$

From (33) and (34) the magnitude squared of the projection of $(P*Q^{-1})$:

$$j_{n'} = [\cos(q) * b_p/p * \sin(p) - \cos(p) * \sin(q)]^2 + [\cos(q) * \cos(p) + b_p/p * \sin(p) * \sin(q)]^2 \quad (35)$$

Equation (35) is identical to equation (19) apart from two signs swapped, and is solved in the same manner to yield $j_{n'} = j_o$.

The angle made by the projection of $P*Q^{-1}$ in $PS(Q)$, from equations (33) and (34), and relative to t_0 , is:

$$k_{n'} = \tan^{-1} \{ [\cos(q) * b_p/p * \sin(p) - \cos(p) * \sin(q)] / [\cos(q) \cos(p) + b_p/p * \sin(p) * \sin(q)] \} \\ = \tan^{-1} \{ [b_p * \tan(p) - \tan(q)] / [1 + b_p/p * \tan(p) * \tan(q)] \} \\ = k_{op} - k_{oq} \quad (36)$$

A.8 Results of post-multiplying P by Q^{-1} for the imaginary plane $PV(Q)$

The results of pre-multiplying P by Q and post-multiplying P by Q^{-1} are identical in the plane $PV(Q)$, as we now show.

The magnitude square of the projection of P in $PV(Q)$, i.e. at right angles to $V(Q)$ in imaginary space is the component along t_2 :

$$l_o = \{c_p/p * \sin(p)\}^2 \quad (37)$$

The angle made by this projection relative to t_2 is zero:

$$m_o = 0 \quad (38)$$

After multiplication, from equation (27), the magnitude square of the projection of $(P*Q^{-1})$ in $PV(Q)$:

$$l_n = \{c_p/p * \sin(p) * \cos(q)\}^2 + \{c_p/p * \sin(q) * \sin(p)\}^2 \quad (39)$$

which simplifies to yield $l_n = l_o$. The angle made by this projection relative to t_2 is:

$$m_n = \tan^{-1} \{ [c_p/p * \sin(p) * \sin(q)] / [c_p/p * \sin(p) * \cos(q)] \} = q = m_o + q \quad \text{from equation (38).} \quad (40)$$

A.9 Proofs for the behaviour of $Q^{\frac{1}{2}}$

Let $Q^{\frac{1}{2}}$ be:

$$R = \cos(r) + b_r/r * \sin(r) * t_1 + c_r/r * \sin(r) * t_2 + d_r/r * \sin(r) * t_3$$

Then, from the definition of quaternion multiplication:

$$\begin{aligned} Q &= R * R = \cos^2(r) - \sin^2(r) + 2 * \cos(r) * \sin(r) * [b_r/r * t_1 + c_r/r * t_2 + d_r/r * t_3] \\ &= \cos(2*r) - \sin(2*r) * [b_r/r * t_1 + c_r/r * t_2 + d_r/r * t_3] \end{aligned}$$

Thus $r = q/2$, and the orientation of $V(R)$ is the same as that of $V(Q)$.

A.10 Mathematical background for the iterative square root

That the function:

$$f(Q) = Q * Q - T \quad (41)$$

where Q and T are quaternions is differentiable with respect to Q may be proved via a generalisation of the Cauchy–Riemann relations following the method of Jeffreys and Jeffreys,² page 337. Taylor's Formula then applies, and in particular Newton's Method for the extraction of a square root on setting $f(Q) = 0$:

$$Q_{n+1} = Q_n - f(Q_n) / f'(Q_n)$$

Therefore, on substitution for $f(Q)$ and $f'(Q)$ from equation (41):

$$Q_{n+1} = Q_n - (Q_n * Q_n - T) / (2 * Q_n) \quad (42)$$

We start by proving that, if T is close to the multiplicative unity, 1, then a solution of $f(Q) = 0$ will also be close to 1.

$$\text{Let } T = 1 + d.$$

$$\text{Let } Q = 1 + e.$$

Then

$$(1 + e) - 1 - d = 0 \text{ and } e = -1 \pm (1 + d)^{\frac{1}{2}}$$

But T is close to 1, and so we may expand the square root in powers of d , neglecting all powers higher than 1.

$$e = -1 \pm (1 + d/2)$$

We now choose the positive sign, noting as we do that this will constrain our algorithm to find only the square root near to 1. We obtain:

$$e = d/2$$

and clearly Q is also close to 1.

We are thus justified in making our initial estimate for Q , Q_0 , 1, and equation (42) gives us:

$$Q_1 = \frac{1 + T}{2}$$

We immediately see that Q_1 is close to 1, and further iterations of the exact equation (42) will produce a value for every Q_n close to 1. We are thus justified in setting the Q_n in the denominator in powers of e_n , and neglecting all powers higher than 1. We obtain:

$$Q_{n+1} = Q_n - (Q_n * Q_n - T) / 2 - e_n * (Q_n * Q_n - T) / 2$$

But $(Q_n * Q_n - T)$ is itself of order e_n , and thus the final approximation is:

$$Q_{n+1} = Q_n - (Q_n * Q_n - T) / 2$$

REFERENCES FOR THE APPENDICES

1. B. Spain, *Vector Analysis*. Van Nostrand, London (1965).
2. H. Jeffreys and B. Jeffreys, *Methods of Mathematical Physics*. Cambridge University Press (1962).