ELSEVIER

Parallel Computing 23 (1997) 2115–2127

PARALLEL COMPUTING

Practical aspects and experiences

Data parallel evaluation of univariate polynomials by the Knuth–Eve algorithm

R.E. Overill [a,*], S. Wilson [b]

[a] *Algorithm Design Group, Department of Computer Science, King's College London, Strand, London WC2R 2LS, UK*

[b] *Rutherford Appleton Laboratory, Chilton, Oxfordshire OX11 0QX, UK*

Received 15 October 1995; revised 12 June 1997

Abstract

The performance of the Knuth–Eve algorithm for data parallel evaluation of univariate polynomials of degree 8, 16 and 32 has been systematically compared with that of the classical Newton–Horner algorithm using three vector processors and three processor arrays. Significant performance improvements have been observed beyond a critical problem size, as predicted by theory. The practical implications of this result for large-scale computations involving function approximation are outlined. © 1997 Elsevier Science B.V.

Keywords: Polynomial evaluation; Newton–Horner algorithm; Knuth–Eve algorithm; Data parallelism; Performance measurements; Vector processor; Processor array

1. Introduction

Univariate polynomials are widely used in the approximation of mathematically important functions by means of a Taylor series expansion about some point. For example, the Gamma function requires a polynomial approximation of degree 8 to attain single precision accuracy [1], and similarly approximations of degree 16 and 32 would be needed to obtain double and quadruple precision respectively.

Not uncommonly, an approximated function has to be evaluated at a large number of irregularly spaced points. For reasons of efficiency it is highly desirable that the function evaluations should be carried out in parallel. In these particular circumstances it is more

* Corresponding author. E-mail: reo@dcs.kcl.ac.uk.

PII S0167-8191(97)00096-3

natural to exploit the vast data parallelism present, rather than the inherent algorithm parallelism. That is, each element of the processor array or vector stream is concerned with the serial evaluation of the polynomial at a single point. Thus a suitable data parallel scheme must be formulated, including the selection of a fast general serial polynomial evaluation algorithm.

Conventionally, a data parallel formulation [2] of the classical Newton–Horner algorithm [3,4] would be employed. It has been shown [5] that the Newton–Horner algorithm is optimal for polynomials of degree not exceeding 4. However, the motivation behind the present study was to discover whether a data parallel formulation of the well-known but little-used Knuth–Eve algorithm [6–8] could significantly out-perform the conventional approach. It was our prior conjecture that this would indeed be the case beyond some *breakeven problem size*, due to the lower computational complexity of the Knuth–Eve evaluation step despite the existence of a non-trivial preconditioning step.

In a previous paper [9], we examined in detail the performance of a number of parallel algorithms for the evaluation of a polynomial at a single point, finding efficiencies of only 33% at best. This somewhat disappointing outcome was attributable largely to insufficient inherent parallelism, as confirmed by the improved efficiencies obtained in [2] with a data parallel approach.

In the present work, we turn our attention to the comparative performance of the Newton–Horner and Knuth–Eve data parallel schemes for evaluating a given polynomial $u(x)$ of degree $n = 8$, 16 or 32 at a large number of points N. In the next section, we describe the Knuth–Eve algorithm and analyze the performance of its data parallel implementation compared with that of the Newton–Horner scheme. The computational details of the present study are given in Section 3. Our results are presented, interpreted and discussed in Section 4, and our conclusions are drawn in Section 5.

2. The Knuth–Eve algorithm

Any polynomial $p(x)$ of degree n can trivially be written

$$p(x) = \sum_{i=0}^{n} a_i x^i = p_e(x^2) + x p_o(x^2) \tag{1}$$

where p_e and p_o are constructed from the even and odd indexed coefficients respectively of p.

From Euclid's remainder theorem for polynomials we have that if P, D, Q and R are polynomials, then P can be written

$$P = DQ + R \tag{2}$$

where D is the divisor, Q is the quotient, R is the remainder on dividing P by D, $\deg R < \deg Q$, and either $Q = 0$ or $\deg Q = \deg P - \deg D$.

It follows that for any real α we can write

$$p(x) = p_e(x^2) + x p_o(x^2) = (x^2 - \alpha)\big(q_e(x^2) + x q_o(x^2)\big) + \rho x + \beta \tag{3}$$

Separating out even and odd powers of x we have

$$p_e(x^2) = (x^2 - \alpha)q_e(x^2) + \beta \tag{4}$$

and

$$p_o(x^2) = (x^2 - \alpha)q_o(x^2) + \rho \tag{5}$$

In particular, if we can choose α such that $p_o(\alpha) = 0$, i.e. if p_o has at least one real root, then we will have $\rho = 0$, and for some appropriate q and β we can write

$$p(x) = (x^2 - \alpha)q(x) + \beta \tag{6}$$

If x and x^2 are both known the problem of evaluating p (of degree n) is reduced to that of evaluating q (of degree $n-2$) together with one further multiplication and two additions. We can repeat the above process to evaluate q provided that q_o has at least one real root, and so on.

Since

$$p_o(x^2) = (x^2 - \alpha)q_o(x^2) \tag{7}$$

the requirement that q_o has at least one real root is equivalent to the requirement that p_o has at least two real roots. For the process to continue to completion we require that p_o has all real roots. For n odd, the reduction terminates with a linear factor which requires one further multiplication and one addition, giving a total of $(n-1)/2 + 1$ multiplications and n additions. For n even, there remains a quadratic factor requiring two further multiplications and two additions, giving a total of $(n-2)/2 + 2 = n/2 + 1$ multiplications and n additions. In general therefore $\lfloor n/2 \rfloor + 1$ multiplications and n additions are required.

By Eve's theorem [7], if all the roots of $p(x)$ have non-positive real parts then the roots of p_o are all real (as indeed are the roots of p_e). Now the roots of $u(x)$ will not in general all have non-positive real parts, but those of $p(x) = u(x + c)$ will, for sufficiently large c. Thus in order to evaluate $u(x)$ we actually evaluate $p(x - c)$ for some suitably chosen c.

The preconditioning step required for this algorithm consists of (i) choosing a suitable value for c and performing the expansion $p(x) = u(x + c)$, and (ii) finding the $m = \lceil n/2 \rceil - 1$ roots α_i of p_o and the associated remainders β_i.

To perform the expansion of $u(x + c)$ serially by forming Pascal's triangle requires

$$\tfrac{5}{2}n^2 + \tfrac{7}{2}n + 2 \tag{8}$$

operations. Finding the roots α_i by the Newton–Raphson Birge–Vieta method [10]

$$x_{k+1} = x_k - f_{-1}/g_{-1} \tag{9}$$

where

$$f_{m-1} = h_m;\ f_{i-1} = f_i x_k + h_i, \quad i = m-1, 0 \tag{10}$$

$$g_{m-2} = f_{m-1};\ g_{i-1} = g_i x_k + f_i, \quad i = m-2, 0 \tag{11}$$

and h_i, $i = 0, m$ denote the coefficients of p_o, requires

$$2\bar{I}(\lceil n/2 \rceil + 1)(\lceil n/2 \rceil - 2) + 1 \tag{12}$$

operations, where $\bar{I}$ is the average number of iterations required for convergence.

Determining the remainders β_i by synthetic division requires

$$2(n - \lceil n/2 \rceil - 1)(\lceil n/2 \rceil - 1) \tag{13}$$

operations. We denote the total number of arithmetic operations in the preconditioning step for even n by

$$X(n) = \left(3 + \frac{\bar{I}}{2}\right)n^2 + \left(\frac{3}{2} - \bar{I}\right)n + 5 - 4\bar{I} \tag{14}$$

The Knuth–Eve algorithm therefore requires a total of

$$\left(\frac{3}{2}n + 3\right)N + X(n) \tag{15}$$

arithmetic operations to evaluate a polynomial $u(x)$ of degree n on a set of N data points x, since the preconditioning step needs to be performed only once. By contrast, the classical Newton–Horner algorithm requires nN multiplications and nN additions, giving a total of

$$2nN \tag{16}$$

arithmetic operations. Thus the breakeven problem size, the value of N above which the Knuth–Eve algorithm performs fewer arithmetic operations than the Newton–Horner is given by

$$\hat{N} = \frac{X(n)}{(n/2) - 3} \tag{17}$$

Using the empirically determined average value of $\bar{I} = 8$, the serial analysis above yields values of $\hat{N} = 369$, 333 and 534 for $n = 8$, 16 and 32 respectively. The minimum value of $\hat{N} = 300$ occurs at $n = 11$ and $\hat{N}$ only exists for $n > 6$.

We now consider data parallel computation on a p-processor machine using a simple SPMD model. The Newton–Horner data parallel scheme assigns the evaluation of the polynomial at a single data point to each processor. The Knuth–Eve data parallel scheme first performs the preconditioning step as a serial computation and then performs the evaluation of the polynomial at a single data point in each processor. For data parallel computation with p processors, the values of $\hat{N}$ are predicted to be precisely p times those for serial computation since p polynomial evaluations are performed in parallel. Thus for $p = 4096$ we find $\hat{N} = 1\,511\,425$, $1\,360\,692$ and $2\,184\,429$, for polynomials of degree 8, 16 and 32 respectively.More generally, by extending Hockney's method of Equal Performance Lines (EPLs) [11] to the case where at least one of the two algorithms has both serial and parallel phases, we obtain

$$\hat{N} = \frac{R_\infty s^{(\mathrm{K-E})}}{q^{(\mathrm{N-H})} - q^{(\mathrm{K-E})}} - n_{1/2} \tag{18}$$

for any SIMD machine architecture, including vector processors and processor arrays. Here, $n_{1/2}$ is the half-performance length and R_∞ is the ratio of the asymptotic performance of the vector or parallel processing mode to that of the scalar processing

Table 1
Parameters of the computing machines employed in the present study

Machine	Word	r_∞	R_∞	$n_{1/2}$	$\hat{N}(n=8)$	$\hat{N}(n=16)$	$\hat{N}(n=32)$
DAP-610C	32′	560	4096	2048	1509377	1358644	2182381
MP-1208	32′	600	15000	4096	5530905	4978905	7995520
CM-200	32′	8192	164	8192	52325	46289	79271
Y-MP8I	64′	333	14	50	5117	4601	7417
C3860	32′	240	13	25	4773	4294	6909
VPX240	32′	2500	21	100	7650	6877	11100

mode; q and s are respectively the number of vector or parallel operations and the number of scalar operations required by the indicated algorithm, obtained from the serial and parallel analyses given above. Substituting these operation counts yields

$$\hat{N} = \frac{R_\infty X(n)}{(n/2) - 3} - n_{1/2} \tag{19}$$

Taking $R_\infty = 4096$ and $n_{1/2} = 2048$ to characterise the AMT DAP-610C, we now find $\hat{N} = 1\,509\,377$, $1\,358\,644$ and $2\,182\,381$ for $n = 8$, 16 and 32 respectively. Similarly, with $R_\infty = 14$ and $n_{1/2} = 50$ for the Cray Y-MP8I, we predict $\hat{N} = 5117$, 4601 and 7417 for $n = 8$, 16 and 32 respectively. Values of $\hat{N}$ for all six machines are included in Table 1.

3. Computational details

We have followed closely the rigorous performance measurement framework set out by Hockney [11,12]. Great care has been taken in designing the implementations to ensure that they are as neutral as possible with respect to the performance of the six machine architectures.

Polynomials of degree 8, 16 and 32 were evaluated in data parallel mode on a 4K processor AMT DAP-610C/32 using Fortran-Plus 4.1S, on an 8K processor MasPar MP-1208 using MPF 2.2.9, on a 16K processor TMC CM-200 using CMF Slicewise 2.1.2, on a Fujitsu VPX240/10 using Fortran77 EX/VP V12L10, on one processor of a Convex C3860 using fc 8.0, and on one processor of a Cray Y-MP8I/8128 using CFT77 6.0.3.0. In each case the highest possible level of code optimisation was employed.

The polynomial coefficients u_i and the members of set of evaluation points x_i were both taken from the uniform distribution over [0, 1] of pseudo-random single-precision real numbers for each machine. In this particular case we have found empirically that it suffices to take the shift-of-origin parameter c equal to unity. A general analytic upper bound on the minimum value of c is however given by the inequality

$$c < \max\left(\gamma, |u_n|^{-1} \sum_{i=0}^{n-1} |u_i| \gamma^{i-n-1} \right) \tag{20}$$

for arbitrarily chosen positive γ.

A range of batch sizes N for data parallel evaluation was defined for each machine as follows: the lower limit was taken to be the (maximum) size of vector register for vector processors or the size of the array for processor arrays; the upper limit was set at 256K for vector processors and 2048K for processor arrays, reflecting the limiting 32 MByte memory size of the DAP-610C/32. Between the lower and upper limits the value of N was repeatedly doubled. This ensured that N was always an integer multiple of the size of the vector registers or the processor array, thereby minimising the effects of the exact problem size on the machine performance.

The mainly scalar computations of the Knuth–Eve preconditioning step were carried out within the DAP array, in the MasPar ACU and on the CM-200 front-end (a Sun 4/370). This permits a comparison of the different modes of scalar computation available, as well as hiding the different characteristics of the three front-end machines.

The results of this study are given in terms of average vector efficiencies

$$\eta = r/r_\infty \tag{21}$$

where r_∞ is the theoretical maximum or asymptotic performance of the machine and

$$r = 2nN/T \tag{22}$$

is its average vector performance [11]. For a p-processor array, η is formally equivalent to the classical efficiency E_p [13]. Here, $2n$ is the scalar operation count for the best serial algorithm [12], N is the batch size, and T is the measured overall CPU time. Each time T is in fact the minimum over 1000 such timings in order to minimise the effects of overheads due to task swapping on the results, after subtraction of the overhead for invoking the timer which is itself minimised over 1000 trials [11].

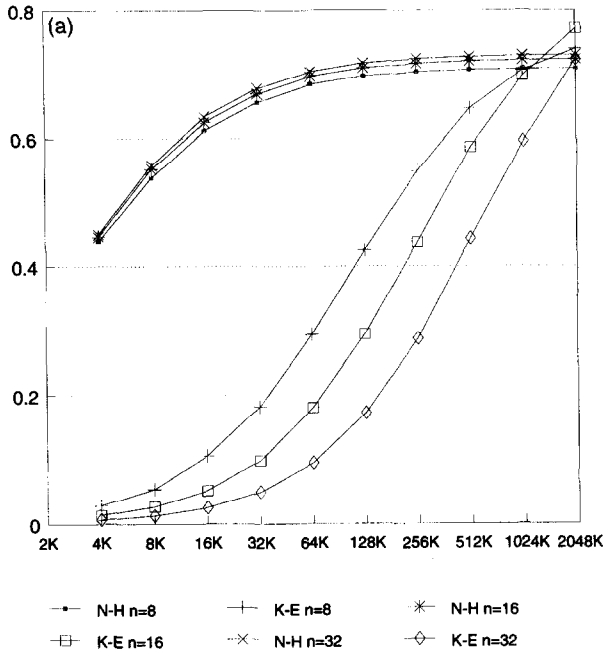
The parameters used for each machine are given in Table 1 together with the predicted values of $\hat{N}$.

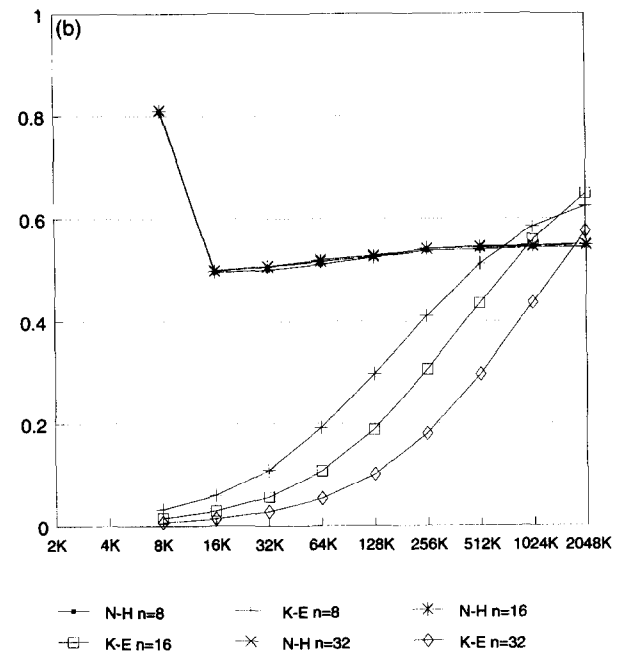
4. Results and discussion

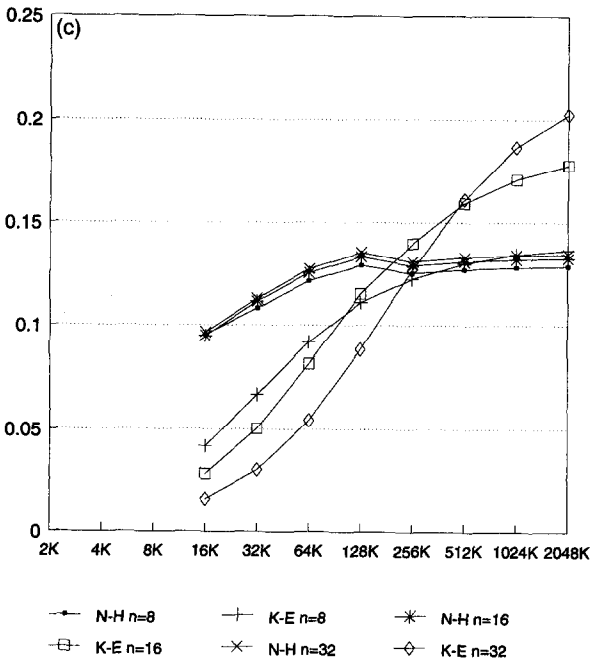
The measured vector efficiencies of the Newton–Horner and Knuth–Eve algorithms for data parallel evaluation of polynomials of degree 8, 16 and 32 at N data points are displayed graphically in Fig. 1 for the six respective machines.

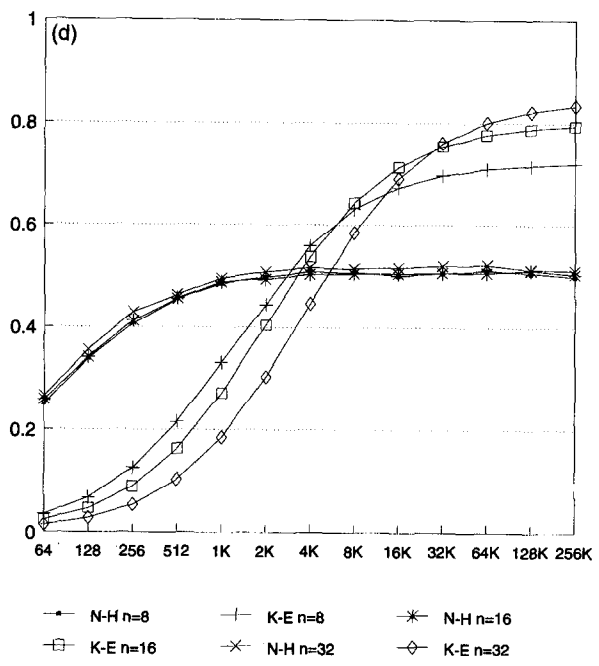
The principal features of our results are that in all cases the Newton–Horner algorithm is superior to the Knuth–Eve algorithm in absolute terms for smaller problem sizes N. However, the relative performance of the Knuth–Eve scheme with respect to the Newton–Horner scheme increases with increasing N. For all machines studied except the C3860 (see below), the Knuth–Eve algorithm outperforms the Newton–Horner algorithm beyond some breakeven problem size $\hat{N}$ within the range of problem sizes considered. For the processor arrays $\hat{N}$ is found to be relatively large (in the range 256K–2048K) in reasonable agreement with the theoretical predictions (Table 1), while for the vector processors $\hat{N}$ is 100–200 times smaller (in the range 2K–8K) in good agreement with theory.

An overall explanation of these general features is now given. The data parallel Newton–Horner scheme contains only vector/parallel operations which are generally


(a)
0.8
0.6
0.4
0.2
0
2K 4K 8K 16K 32K 64K 128K 256K 512K 1024K 2048K
N-H n=8
K-E n=8
N-H n=16
K-E n=16
N-H n=32
K-E n=32


(b)
1
0.8
0.6
0.4
0.2
0
2K 4K 8K 16K 32K 64K 128K 256K 512K 1024K 2048K
N-H n=8
K-E n=8
N-H n=16
K-E n=16
N-H n=32
K-E n=32


(c)
0.25
0.2
0.15
0.1
0.05
0
2K 4K 8K 16K 32K 64K 128K 256K 512K 1024K 2048K
N-H n=8
K-E n=8
N-H n=16
K-E n=16
N-H n=32
K-E n=32


(d)
1
0.8
0.6
0.4
0.2
0
64 128 256 512 1K 2K 4K 8K 16K 32K 64K 128K 256K
N-H n=8
K-E n=8
N-H n=16
K-E n=16
N-H n=32
K-E n=32

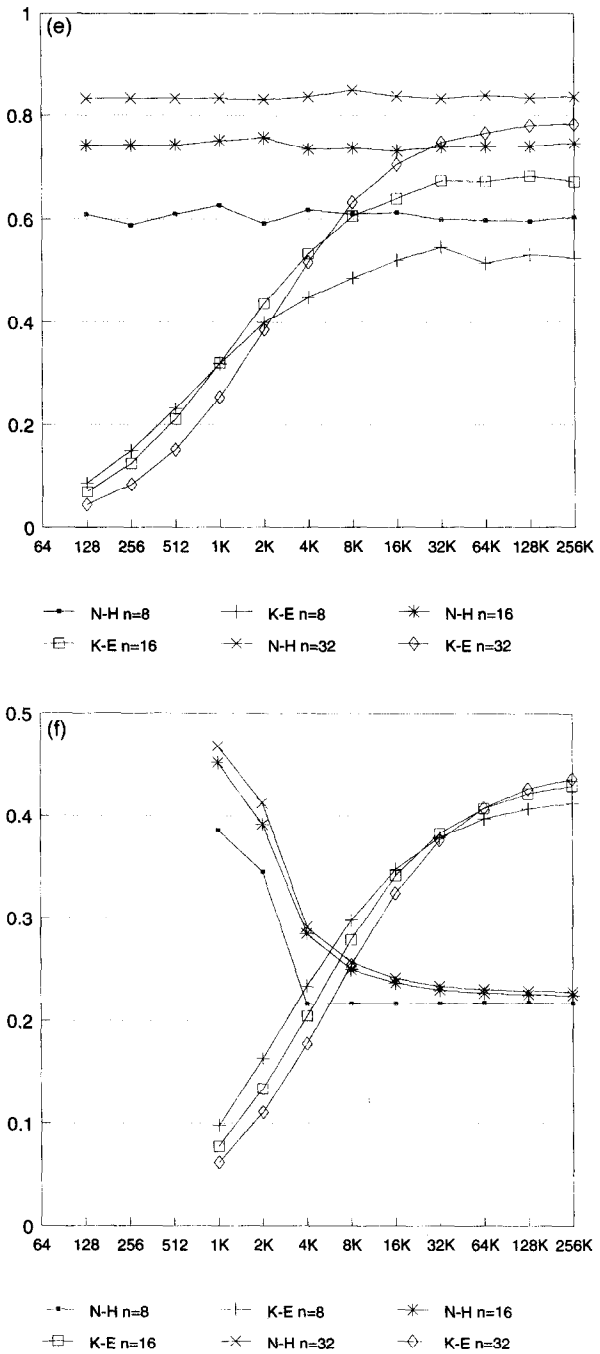


Fig. 1. Average vector efficiencies for Newton–Horner and Knuth–Eve algorithms for data parallel evaluation of N power series of degree n for (a) the DAP-610C computer, (b) the MP-1208 computer, (c) the CM-200 computer, (d) the CRAY Y-MP81 computer, (e) the C3860 computer, (f) the VPX240 computer.

performed more efficiently as the problem size N increases, reaching an asymptote as $N \to \infty$. The Knuth–Eve data parallel scheme, on the other hand, has an initial serial preconditioning step whose fixed cost $X(n)$ has to be amortised over the following evaluation step. Thus for small N the preconditioning cost is relatively high and the overall efficiency is correspondingly low, but as N increases the relative significance of preconditioning step decreases and the overall efficiency increases. For large N the cost of preconditioning becomes insignificant and the lower asymptotic complexity of the Knuth–Eve scheme causes it to outperform the Newton–Horner scheme beyond some breakeven value $\hat{N}$.

The relatively large values of $\hat{N}$ for the processor arrays reflects the relative slowness of serial (scalar mode) processing on these machines. Conversely the designers of modern vector processors have taken on board the implications of Amdahl's law [14,15] and equipped them with relatively fast scalar processing capabilities which are here manifested in much lower values of $\hat{N}$. Thus the data parallel Knuth–Eve scheme becomes viable at much smaller problem sizes on vector processors than on processor arrays.

Since $X(n) \sim O(n^2)$ from Eq. (14), we see that $\hat{N} \sim O(n)$ from Eq. (19). Thus a linear increase in the value of the breakeven problem size with increasing degree of polynomial is generally to be expected.

There is a significant difference in the memory/communications demands made by the two algorithms, as reflected in their respective computational intensity values f [16], the ratio of arithmetic operations to memory references. For the main step of the data parallel Newton–Horner scheme

$$v \leftarrow vx + p_j \quad 0 \le j \le n-1 \tag{23}$$

where v is a vector of N partial evaluations, we have $f = \frac{2}{3}$ without explicit loop unrolling. The principal step of the data parallel Knuth–Eve scheme

$$v \leftarrow v\left(x^2 - \alpha_j\right) + \beta_j \quad 1 \le j \le m$$

where v is again a vector of N partial evaluations, has $f = 1$ under the same conditions. The significance of this result is that formally the Knuth–Eve scheme requires fewer memory references to support each computational operation than the Newton–Horner scheme.

The global behaviour of the Knuth–Eve scheme is remarkably consistent across all six machines and all three degrees of polynomial studied. By contrast, the Newton–Horner scheme behaves in a more variable manner with examples of increasing (Fig. 1(a), (c), (d)), almost constant (Fig. 1(e)) and decreasing (Fig. 1(b), (f)) performance as N increases. These performance variations reflect particular features of the individual computer architectures or system software, to which we now turn our attention.

The AMT DAP-610C (Fig. 1(a)) follows the theoretical predictions closely. The efficiency of the Newton–Horner scheme displays only a slight dependence on the degree of polynomial n, but increases from about 45% at $N = 4$K to more than 70% at $N = 2048$K. For the Knuth–Eve scheme the efficiency rises from about 3% at $N = 4$K to over 70% at $N = 2048$K. The values of $\hat{N}$ are all greater than 1024K, and for $n = 32$ $\hat{N}$ exceeds 2048K, in good agreement with theory.

The MasPar MP-1208 (Fig. 1(b)) displays an anomalous performance discontinuity between $N = 8$K and $N = 16$K for the Newton–Horner scheme. This behaviour has been noted many times before with various applications [17], and is generally ascribed to the generation of inefficient code by the MPF compiler during automatic virtualization of oversize problems, i.e. when the 8K PE array has to be repeatedly refilled with data. After its initial drop from 81% at $N = 8$K the efficiency of the Newton–Horner scheme slowly increases to 55% at $N = 2048$K and, like the DAP-610C, is virtually independent of n. The Knuth–Eve scheme behaves as expected with efficiencies rising from under 2% at $N = 8$K to nearly 65% at $N = 2048$K for $n = 16$. As with the DAP-610C, the highest efficiencies are found for $n = 8$. The values of $\hat{N}$ are all in the range 512K–2048K, which is lower than predicted and is directly attributable to the poor performance of the Newton–Horner scheme discussed above.

The TMC CM-200 (Fig. 1(c)) behaves essentially as predicted. One small anomaly in the performance of the Newton–Horner scheme is the slightly higher efficiency found at $N = 128$K for each value of n. On a $16K$ processor array this corresponds to a virtual processor ratio of 8 which permits the CMF compiler to perform a slightly better optimisation. The efficiencies of the Newton–Horner scheme rise from below 10% at $N = 16$K to around 13% at $N = 2048$K, almost independently of n. This is directly comparable with an efficiency of 11.3% at $N = 512$K measured previously with the Newton–Horner scheme for large values of n [2]. With the Knuth–Eve scheme the efficiencies rise from under 2% at $N = 16$K to over 20% at $N = 2048$K for $n = 32$. It is worth noting that only by explicitly unrolling the outer (polynomial) loop was Butel [18] able to achieve an efficiency of 22% on a CM-2 with the Newton–Horner scheme, which suggests deficiencies in the scheduling of operations by the CMF compiler. The values of $\hat{N}$ all lie in the range 256K–512K, substantially lower than for the DAP-610C and MP-1208 due to the much faster scalar processing of the Knuth–Eve preconditioning step which is performed on the Sun 4/370 front-end.

The Cray Y-MP8I single processor (Fig. 1(d)) follows the theoretical predictions very closely indeed. For the Newton–Horner scheme the efficiencies rise from about 25% at $N = 64$ to over 50% at $N = 256$K and are almost independent of n. For the Knuth–Eve scheme they rise from under 2% at $N = 64$ to over 83% at $N = 256$K for $n = 32$, with somewhat lower peak efficiencies for $n = 16$ and $n = 8$. The values of $\hat{N}$ all lie close to $N = 4$K and are in increasing sequence with n as predicted in Table 1.

The Convex C3860 single processor (Fig. 1(e)) is the only machine in this study to exhibit approximately constant performance with the Newton–Horner scheme and no performance crossover between the two algorithms. An examination of the output from the Convex performance analysis tool *CXpa* suggests that the inability of the Knuth–Eve scheme to outperform the Newton–Horner scheme results from a higher than expected vector chime count for one section of the Knuth–Eve code and is due to the provision of only one vector load/store pipeline per processor. This memory bandwidth limitation is also responsible for the almost constant performance of the Newton–Horner scheme when N is an integer multiple of 128, the vector register size. As was also found previously [2], performance improvements can be obtained by increasing n, the degree of polynomial being evaluated, thus amortising the cost of vector memory references over a greater number of arithmetic operations. The Knuth–Eve scheme behaves much

as expected with the efficiency for $n = 32$ rising from just above 4% at $N = 128$ to over 78% at $N = 256$K, while somewhat lower peak efficiencies are found for $n = 16$ and $n = 8$.

The Fujitsu VPX240/10 (Fig. 1(f)) displays perhaps the most striking anomaly of the six machines in its performance on the Newton–Horner scheme. The efficiency drops from almost 47% at $N = 1024$, the maximum vector register size, to under 23% at $N = 256$K for $n = 32$. The behaviour for $n = 16$ and $n = 8$ is similar, and, unlike the case of the MP-1208, there is no indication of any recovery of efficiency as N increases. The cause of this behaviour appears to be inefficient strip-mining of vectors longer than 1024 elements by the F77 EX/VP compiler. The performance of the Knuth–Eve scheme is very much as predicted with an efficiency of over 6% at $N = 1024$ rising to over 43% at $N = 256$K for $n = 32$, and similar behaviour for $n = 16$ and $n = 8$. Due to the poor performance of the Newton–Horner scheme, the values of $\hat{N}$ are slightly smaller than predicted in Table 1, lying at or below $N = 8$K. They are however in the predicted order.

5. Summary and conclusions

The well-known but little-used Knuth–Eve algorithm has been shown to yield significant performance advantages over the classical Newton–Horner algorithm when employed in the data parallel evaluation of single, double and quadruple precision polynomial approximations at a sufficiently large number of points N. For vector processors with sufficient memory bandwidth, the breakeven problem size $\hat{N}$ is found to lie in the range 2K–8K points, in good agreement with theoretical predictions. For processor arrays, the measured values of $\hat{N}$ are 100–200 times larger, again in reasonably good agreement with theory.

Since there exist large-scale computational science problems where the evaluation of a large number of polynomial function approximations is required, this result offers the realistic prospect of substantial performance improvements for such problems, leading to consequentially accelerated developments in the application domain.

Acknowledgements

R.E.O. wishes to thank Mr A.L.J. Wells and Dr J.L. Martin for valuable insights into the Knuth–Eve algorithm. We are grateful to the following institutions for allowing access to the machines employed in the present work: University of London Computer Centre, Rutherford Appleton Laboratory, Manchester Computing Centre, London Parallel Applications Centre, Edinburgh Parallel Computing Centre, and Cambridge University Computing Laboratory. S.W. acknowledges the support of EPSRC.

References

[1] M. Abramowitz, I.A. Stegun (Eds.), A Handbook of Mathematical Functions, Dover, 1965, Sec. 6.1.36, p. 257; Sec. 7.1, pp. 297–299.

[2] R.E. Overill, Data parallel evaluation of power series, Proc. ESPRIT Workshop on Performance Evaluation of Parallel Systems (PEPS'93), 29–30 November 1993, University of Warwick, UK, pp. 219–224.
[3] I. Newton, De Analysi per Aequationes Infinitas, 1669, reprinted in D.T. Whiteside (Ed.), The Mathematical Papers of Isaac Newton, vol. 2, Cambridge Univ. Press, 1968, p. 222.
[4] W.G. Horner, A new method of solving numerical equations of all orders by continuous approximation, Philos. Trans. R. Soc. London 109 (1819) 308–335.
[5] A.M. Ostrowski, On two problems in abstract algebra connected with Horner's rule, in Studies in Mathematics and Mechanics presented to R. von Mises, Academic Press, 1954, pp. 40–48.
[6] D.E. Knuth, Evaluation of polynomials by computer, Commun. ACM 5 (1962) 595–599.
[7] J. Eve, The evaluation of polynomials, Numer. Math. 6 (1964) 17–21.
[8] D.E. Knuth, The Art of Computer Programming, vol. 2, 2nd. ed., Addison-Wesley, 1976, pp. 473–475.
[9] R.E. Overill, S. Wilson, Performance of parallel algorithms for the evaluation of power series, Parallel Comput. 20 (1994) 1205–1213.
[10] F.B. Hildebrand, Introduction to Numerical Analysis, McGraw-Hill, 1956, p. 453.
[11] R.W. Hockney, C.R. Jesshope, Parallel Computers 2, Adam Hilger, 1988, pp. 89, 95, 142, 443–445.
[12] R.W. Hockney, A framework for benchmark performance analysis, Supercomput. IX-2 (1992) 9–22.
[13] D.J. Kuck, The Structure of Computers and Computations, vol. 1, Wiley, 1978, p. 33.
[14] G.M. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, Proc. AFIPS Joint Spring Computer Conf., 1967, pp. 483–485.
[15] G.M. Amdahl, Limits of expectation, Int. J. Supercomput. Appl. 2 (1988) 88–94.
[16] R.W. Hockney, I.J. Curington, $f_{1/2}$: A parameter to characterize memory and communication bottlenecks, Parallel Comput. 10 (1989) 277–286.
[17] R.A. Fatoohi, Performance comparison of several SIMD machines, Proc. 5th SIAM Conf. on Parallel Processing for Scientific Computing, 25–27 March 1991, Houston, TX, pp. 419–424, Table 6.
[18] R. Butel, A Cray-2 versus CM-2 comparison using several polynomial benchmarks, Parallel Comput. 18 (1992) 931–945.