INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2006; **52**:321–354
Published online 10 February 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1184

Evolutionary optimization techniques as versatile solvers for hard-to-converge problems in computational fluid dynamics

Raed I. Bourisli[1,*,†] and Deborah A. Kaminski[2,‡]

[1]*Department of Mechanical Engineering, College of Engineering and Petroleum, Kuwait University, Kuwait*
[2]*Department of Mechanical, Aerospace and Nuclear Engineering, Rensselaer Polytechnic Institute, New York, U.S.A.*

SUMMARY

Evolutionary algorithms mimic the process of natural evolution governed by the 'survival of the fittest' principle. In this work, a genetic algorithm (GA) is successfully used to solve problems in potential flow in a gradual contraction, viscous flow over a backward facing step, and non-Newtonian flow using the power law model. Specifically, the GA heuristically searches the domain for potential solutions, precluding many convergence difficulties associated with the stiffness of a problem. The GA was able to solve problems that the gradient-based method could not mainly because of its relative indifference to regions of high gradients when performing the search and that systems of discretized equations are never actually solved. The GA exhibited excellent scalability properties for solving problems with a large number of nodes. These results show evolutionary techniques to be of great utility in solving stiff problems in fluid flow. Copyright © 2006 John Wiley & Sons, Ltd.

KEY WORDS: evolutionary optimization; genetic algorithms; solution techniques; potential flow; viscous flow; non-Newtonian fluids; convergence; backward facing step

1. INTRODUCTION

A computational fluid dynamics analysis starts with the choice/construction of mathematical models that describe the system of interest, (*flow model*), followed by the selection of the *discretization method*(*s*), and concludes with the solution of the resulting discretized algebraic equations by an appropriate solution technique. Common CFD methods, by and large, do succeed most of the time in arriving at 'a solution' to the modelled problem. The effectiveness of current numerical techniques is perhaps overstated, due to the fact that failure is rarely reported. In spite of that, it is well known that current solution techniques do face occasional difficulties when dealing with particular types of systems. Turbulence modelling,

*Correspondence to: Raed Bourisli, Department of Mechanical Engineering, College of Engineering and Petroleum, Kuwait University, P.O. Box 5969, Safat 13060, Kuwait.
†E-mail: raed@kuc01.kuniv.edu.kw
‡E-mail: kamind@rpi.edu

Received 3 August 2005
Revised 3 December 2005
Accepted 26 December 2005

reacting flows, conjugate heat transfer problems, fluid power systems and viscoelastic fluid flow simulations are but a few examples where the partial differential equations combined with the solution properties can lead to 'stiff' problems.

Stiffness is one of the main causes of divergence in many numerical schemes. Stiffness is a phenomenon exhibited by the system/method combination and is not a well-defined property of it. Here, it is loosely taken to mean that some components of the solution being sought are unstable or change much faster than others. This is equivalent to the eigenvalues of the Jacobian matrix of the system having two or more real parts that are large, positive and far apart, in a way that no scaling technique can resolve the problem. These depend on the system of equations and initial conditions, as well as the numerical method itself. It is known that some problems face difficulties converging using one solution technique but not another. If the issues of uniqueness, stability and consistency of a given discretization scheme are resolved, then one can safely assume that any resulting set of sound discretized equations will yield zero residuals (to within machine accuracy) if the *correct* solution is substituted in.

The main objective here is to apply genetic algorithms (GAs) to flow problems which do not converge easily or at all with conventional methods, such as the usual finite difference and finite volume methods. The idea is to detect divergence of the numerical scheme, switch to the evolutionary solver to get the solution, and go back to the main routine to carry on with further iterations or perform postprocessing. The key assumptions made at the outset is that the discretized equations of a particular numerical method 'know' the correct solution if they happen to see it, i.e. the calculated residuals of the discretized will be equal to zero. Furthermore, it is assumed that any appropriate norm of the residuals is a monotonic function of the *fitness* of the solution.

An attractive feature of evolutionary algorithms is the stochastic nature of their search for solutions. For common fluid dynamics problems, this means that the usual iterative procedure for solving steady problems might not be necessary since in evolutionary algorithms no system of equations is actually solved, and as a result, many of the stability and CFL conditions associated with iterative solvers cease to be a major concern. Most importantly, however, is that the search depends directly on the actual values of the unknowns and *not* on the gradients of unknowns in the flow field, a major cause of divergence for many CFD schemes.

The paper is organized as follows. Section 2 introduces GAs, their history as optimization tools and their various operators, with particular emphasis on how this GA is customized to be applied to fluid flow problems. Section 3 gives the results of applying the GA to potential flow through a gradual contraction channel. Section 4 gives the results of using the GA to solve problems of viscous fluid flows obeying the steady, incompressible Navier–Stokes equations. A brief overview of non-Newtonian fluids and the results of applying the GA to the flow of a power law non-Newtonian fluid are given in Section 5. Finally, Section 6 gives some final conclusions on the application of evolutionary techniques to problems in CFD as well as some suggested future directions for this work.

2. GENETIC ALGORITHMS

2.1. A brief overview

GAs are stochastic search and optimization techniques that mimic the evolution process of biological organisms in nature which is governed by *natural selection*. The idea that biological

evolution can be used as an optimization tool for problems in science and engineering was first addressed by computer scientists in the 1950s and 1960s. In 1965 Rechenberg [1] introduced the first evolution-based optimization tool. The method dealt with optimizing parameters for devices such as airfoils through letting a population of candidate solutions evolve according to rules inspired by biological systems to (hopefully) give the desired solution. In the 1960s, Holland [2] began studying the phenomenon of adaptation in natural systems with the aim of importing the mechanism into electronic computers. In general, the area of evolutionary algorithms has three mainstream categories: evolutionary strategies, developed by Rechenberg and Schwefel [3], genetic programming (GP), developed by Fogel [4, 5], and GAs, developed by Holland. Holland's GA loosely follows the Darwinian principle of natural selection best described by the phrase '*survival of the fittest*'. They stem from the notion of biological evolution where organisms mate and natural selection guarantees the passage of individuals with better qualities to the next generation.

The genetic operators *crossover* and *mutation* are the two main components of a GA simulation. Chromosomes represent potential solutions of a given problem, and genes are the parameters of the solution being sought. The crossover operator can be redefined more freely as cutting and recombining chromosomes at any place(s) in the two (or more) parents. The mutation operator is also redefined as changing one or more genes within the chromosome, flipped if binary genes are used, and perturbed if real-valued genes are used. GAs require a few more basic components to function: a way to assess fitness, and a parent selection scheme. An *objective function* is used to measure the fitness of chromosomes in each generation.

GAs have been used in many fields of optimization. Like most other 'meta-heuristic' methods, such as simulated annealing and tabu search, they have been mainly applied to difficult combinatorial optimization problems, the likes of the *travelling salesman problem* [6]. They were also applied to other areas of optimization like the design of radio antennas, fin profile designs, inverse initial-value boundary-value problems, and even areas as diverse as fashion design, and music composition [7–12]. Various problems in aerodynamics, ranging from wing shape optimization to active noise control, have also been tackled using GAs [13]. By and large, GAs have been used primarily for problems for which no established optimization techniques existed. In recent years, GAs have been applied to an increasing number of engineering problems in the areas of heat transfer and fluid mechanics, albeit not exactly as CFD solvers or meta-solvers. They have been applied to basic heat transfer problems, multiphase flow functions estimation, and pipeline flow optimization [14–16].

More recently, however, these evolutionary techniques have been applied to elementary fluid flow problems; Fan *et al.* [17] used a real-coded GA to solve a potential flow problem for a two-dimensional circular diffuser cascade with 40 nodal points. Pryor [18] used a binary GA to solve for a transient, 20-node 1-D flow through a circular pipe. Both efforts were greatly constrained by the number of genes that could be optimized and the high computational cost for the evolution, which put a limit on the size and complexity of the problems considered.

It has been reported that real-coded GAs outperform binary-coded ones in many types of design problems. However, with the above difficulties, even real-coded GAs lead to premature convergence (to local extrema) when applied to problems with a large number of design variables. Further, the nature of appropriate genetic operators have not been truly investigated; for a successful evolution the operators have to be designed in a way that takes into account the nature of the solution encoding, the tolerances of the problem, and the properties of the objective function used to evaluate the fitness. A minor added difficulty is the dependence of

an efficient GA on the quality of the initial guess for the domain of the global extremum. This limits the practicality of using a GA for meaningful engineering problems where the number of variables is often large.

Bourisli and Kaminski [19] introduced a new strategy for applying GAs to larger-scale fluid flow problems. A uniform, successive-refinement strategy was presented wherein the algorithm starts with a coarse solution and works its way through multi-levels of refinements to arbitrary solution resolution. At each level of refinement, a GA-window/sweep-through technique is used to obtain incremental solutions for various parts of the flow domain, thus reducing the number of GA optimized variables in each generation. A finite difference discretization was used as the objective function. The encouraging results of that application of GAs to potential flow problems motivated the extension of the technique to other applications in CFD with more involved flow models and discretization methods [20]. This is combined with the development of new genetic operators specifically designed to exploit the known nature of unknown parameters.

We discuss below in detail the basic components of the algorithm, the various genetic operators, and the general structure of the GA used in this study. We focus on new operators designed specifically to enable the GA to deal with fluid flow problems. The objective functions are particular to the flow problem considered and will be discussed in the respective sections.

2.2. *Population encoding*

Real-coded GAs, as opposed to binary-coded ones, give greater freedom in choosing different crossover and mutation techniques as well as inventing new ones. In this study, real-coded chromosomes are used in the form of matrices that correspond to possible solutions of the flow field variables. For example, the stream function ψ, the x- and y-direction velocities u and v, and the pressure p are encoded on multiple 2D arrays corresponding to the number of potential solution in the population; the 2D structure corresponds to the domain mesh. In general, populations are initialized by filling in the arrays of unknowns with pseudo-random numbers in intervals bounded by expected minimum and maximum values. More on the population initialization is said once numerical examples are discussed.

2.3. *Selection scheme*

Selection is the essence of the natural selection phenomenon on which the genetic evolution is based. Armed with information about the fitness of the individuals in a population, a selection scheme selects parents that will 'mate' and produce the next generation. Much research has gone into developing schemes that preserve the good qualities of fit individuals while still guaranteeing some presence of diversity possibly present in less fit individuals. A number of schemes exist for selecting candidate chromosomes (parents) that will undergo the genetic operators to produce the next generation (offspring.) Among the most popular selection schemes are the *roulette wheel selection* and *rank selection* schemes.

Roulette wheel selection is a stochastic sampling technique with replacement. Individuals are selected for mating with probabilities directly proportional to their fitness. While roulette wheel selection has zero bias, it has the disadvantage of not guaranteeing *minimum spread* (range of possible values for the number of offspring of an individual.) All selected parents could end up being the same individual (unlimited spread.) To overcome some of the drawbacks of

the plain roulette wheel selection, it is often combined with rank selection. In rank selection, individuals are ranked according to their objective values, lowest to highest, and a new fitness value is assigned to each individual depending on its position in the ranked list, often raised to a *rank power*. Rank selection is quite appealing in that it naturally overcomes stagnation, where the selective pressure (the probability of the best individual being selected compared to the average probability of selection of all individuals) is low, and premature convergence (when the search narrows down too quickly) is rare. Experiments have shown it to be superior to roulette wheel selection, and some researchers have gone so far as to label it as the best available selection scheme [21, 22]. Other popular selection schemes exist, such as, the *stochastic universal sampling* and *tournament selection* schemes [23]. All numerical results reported here, however, use a quadratic rank selection.

2.4. Crossover

Crossover is a close approximation to what happens biologically when the chromosomes of the parents are crossed over, i.e. cut and recombined, to form the chromosome of an offspring. In real-coded chromosomes, one or more crossover points may be chosen at random and alleles around them are exchanged between the parents.

The two-point crossover operator has been used widely in GA and has been claimed to outperform the single-point one [24]. In computational fluid dynamics it is often the case that potential solutions are cast in matrix form. Since we are at liberty in designing our own crossover operator(s) and using more than one crossover point, a number of possibilities arise as to how the offspring will look. A number of crossover possibilities at the phenotypic level are shown schematically in Figure 1. The optimum scheme might be problem-dependent and is best decided with full consideration of the particular solution structure. The crossover scheme of Figure 1(a) was determined to be a good choice because of the randomness it provides and thus was used in virtually all results reported in this study.

The number of crossover points N_c was made to depend on the total number of alleles N_{np} in a parent chromosome. The length of the solution building blocks is assumed to stay constant even if more refined solutions are considered. We suggest varying the number of crossover points in the following manner:

$$N_c = \left\lfloor \frac{\sqrt{N_{np}}}{1.2} \right\rfloor \tag{1}$$

where $\lfloor\ \rfloor$ is the greatest integer (or floor) function. This increases the number of crossover points with increasing chromosome length. The rationale for this slow increase is the expected increase in the size of the building blocks of the solution with increasing resolution.

With regard to the nature of the crossover at the gene level (genotypic crossover,) two types of crossover were used: the uniform crossover and the arithmetical crossover [23]. In a 0.5-uniform crossover, the first offspring acquires the *i*th gene from the first parent with probability 0.5, with the other offspring getting the other parent's gene. In the simple arithmetical crossover, instead of the alleles after a crossover point being exchanged between parents to form the offspring, a fraction α of one is added to the complement of the fraction of the other to form one offspring allele, with the opposite fractions forming the other offspring's allele. If the fraction α is set to 0.5, the resulting operator will simply average genes beyond

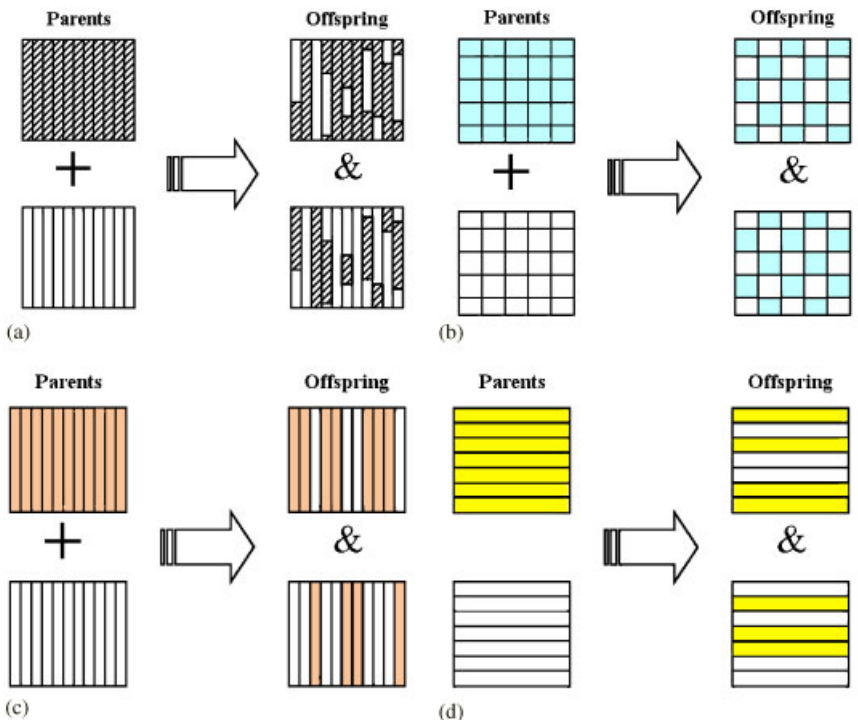

Figure 1. Schematic representations of various crossover operators.

the crossover point. In the current application of this operator the fraction α is chosen randomly from $[0.2, 0.8]$ at each application.

In general, either of the two genotypic crossover operators can be applied independently according to its own probability, with a maximum of one operator per crossover. In other words, for each set of two parents, either uniform, arithmetical or no crossover is applied. For example, suppose we chose to apply the uniform and arithmetical crossover operators with probabilities 0.1 and 0.5, respectively. For the set of two parents, a random number s from $[0, 1]$ is generated; if $s \leqslant 0.1$, the uniform crossover is applied, if $0.1 < s \leqslant (0.1 + 0.5)$, the arithmetical crossover is applied, and if $s > (0.1 + 0.5)$, no crossover takes place. In this case, the total crossover probability is 0.6, with the uniform crossover applied 17% of the time and the arithmetical crossover 83% of the time, within that 0.6 probability. This scheme provides more randomness and adds to the versatility of crossover.

2.5. *Mutation*

Mutation is the mechanism through which diversity is maintained in a population. By randomly changing values of alleles in a chromosome it also acts as the main safeguard against premature convergence. For real-coded GAs, however, mutation is often the main driving force in the population evolution, as reported by many researchers [19, 25, 26]. There are numerous possible mutation operators, such as creep mutation [27], coarse- and fine-grained

mutation [28], etc. In this study, four different mutation schemes are used and a probability of application is assigned to each; we discuss all four of them below.

2.5.1. Non-uniform mutation. A widely used mutation method for real-valued chromosomes is the *non-uniform mutation* [23, 29] which takes into account known limits on the value of a parameter. Given that a particular allele x_i can only assume values between an upper and a lower bound, x_U and x_L, then if the allele is mutated with probability P_m^u, the newly mutated allele is

$$x_i' = \begin{cases} x_i + (x_U - x_i)(1 - s_1^{(1-g/g_{\max})^\beta}) & \text{if } s_2 \leqslant 0.5 \\ x_i + (x_L - x_i)(1 - s_1^{(1-g/g_{\max})^\beta}) & \text{if } s_2 > 0.5 \end{cases} \tag{2}$$

where s_1 and s_2 are uniformly distributed random numbers from $[0, 1]$, g is the generation number, $g_{\max}$ is the maximal number of generations for the run, and β is a mutation parameter (a real number of order 10). This formula provides for aggressive exploration early in the evolution and concentrated exploitation in latter stages. But due to the nature of the convergence criteria, the maximum number of generations is not known in advance. More importantly, specifying meaningful upper and lower bounds on variables can be tricky, especially if the range must be narrow so that the operator is effective.

2.5.2. Fitness-guided mutation. This mutation operator is a slightly modified version of the non-uniform mutation discussed above but without many of its limitations. One possible way to overcome the limitation of having to specify meaningful lower and upper bounds on the value of the optimized variable is to use information calculated in the fitness evaluation phase. A linear transformation of a residual norm r from the calculated objective function is used as a random, yet targeted, perturbation of the alleles to be mutated. We call this scheme the *fitness-guided mutation*. To use the fitness information in the mutation operator, the general proposed formula is

$$x_i' = \begin{cases} x_i(1 + rs_1^{(1-f)}) & \text{if } s_2 \leqslant 0.5 \\ x_i(1 - rs_1^{(1-f)}) & \text{if } s_2 > 0.5 \end{cases} \tag{3}$$

where r is fitness information represented by a measure of the residual at the current node, f is the fitness of the fittest individual, and s_1 and s_2 are random numbers from $[0, 1]$. By using the fitness information, this mutation formula eliminates the need to specify bounds on the variable values. Furthermore, substituting the fitness of the fittest individual for the $g/g_{\max}$ ratio eliminates the need to set *a priori* limits on the number of generation the algorithm performs: when the fittest individual has a fitness close to 1 the GA is considered close to complete. More on the nature of the residuals and the fitness function is said once the objective functions of the different problems are discussed.

2.5.3. Random average mutation. A third mutation scheme proposed here is the *random average mutation* scheme, where the mutated allele is taken to be an average of a randomly selected set of four neighbouring nodes. One set consists of the south, east, north and west nodes, while the other set consists of the southwest, southeast, northeast and northwest nodes.

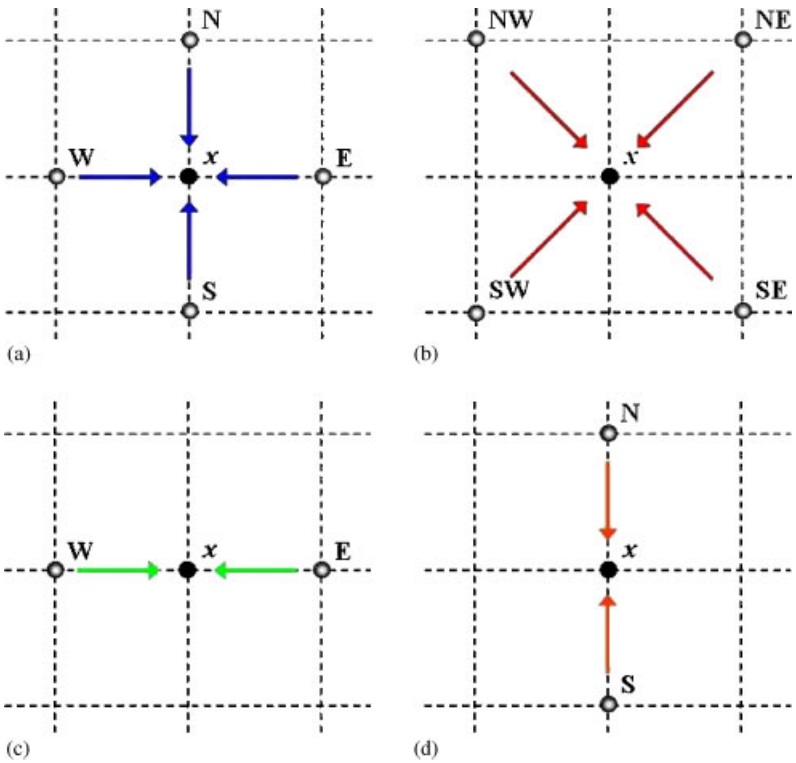

Figure 2. The four equally probable sets of nodes for the random average mutation operator.

The two equally possible sets of neighbouring nodes are shown in Figure 2. Obviously, this mutation scheme can only be applied to nodes that are interior to the flow domain since nodes on the boundaries do not have valid neighbours in all four directions.

2.5.4. Block mutation. Very rarely, the values of the variables optimized do not need random mutation that is applied selectively to random genes but a very small shift in the magnitude of the whole field (or GA window, as discussed below.) The fourth type of mutation is one where the values of all alleles are increased or decreased by a very small amount. If the field is slightly 'out of phase' with the optimum solution then this small change will benefit it directly. If, however, the field is not that uniformly out of phase, which is more likely, then this minute block change of the field will serve other chromosomes when the next crossover takes place. We call this new mutation operator the *block mutation*. This operator is most effective when large flow fields are combined with small GA windows; its probability should be at least one order of magnitude smaller than the probabilities of the three previous operators so that it will not interfere with the main search.

2.6. Elitism

To improve the process further, the elitist strategy is used: clones of a portion of the population, usually a clone of only the fittest individual in the generation, survives the genetic

operators and is automatically reinserted in the next generation, right before the next selection process takes place. This guarantees that a good solution is not lost by being crossed over or mutated away, resulting in offspring that are inferior to it, or, worse, not being selected for reproduction at all.

As a way of reducing the elitist reinsertion computational time, only two chromosomes are picked randomly, their fitness measured, and the elitist is reinstated in place of the inferior. This precludes the accidental writing of the elitist on top of a randomly selected better chromosome while still saves the time required to calculate the fitness of the whole population and replacing the overall worst chromosome. The elitist strategy has been used in all numerical results reported in this study.

2.7. Refinement

As alluded to previously, GAs can face difficulties when applied to problems where the number of optimized variables is very high, therefore it is unwise to tackle complex simulations with thousands and thousands of nodes directly, such as those found in fluid dynamics. Solving a fluid flow problem with even a few thousand nodes is quite challenging for a simple GA. Moreover, one hallmark of GA is its sensitivity to the initial guess; the quality and diversity of genes of the initial population do affect the time it takes the GA to converge, and whether it converges to a local or global extremum. This is a critical tradeoff: the algorithm search space consists of a large space of real numbers, so the smaller the region the GA is exposed to, the faster it can zero in on its extremum. On the other hand, if enough diversity is present in the population, the chances for the GA to converge to the absolute extremum are greatly enhanced.

To overcome these difficulties a multilevel refinement process that takes the solution from a small scale involving a handful of nodes to the desired level of refinement can be used. The process is similar to *successive* grid sequencing. The solution starts with a coarse grid of the flow domain, with enough nodes to describe the large-scale features of the flow field. Once the GA converges to an acceptable solution for this level, the refinement takes place by placing one or more nodes between existing ones, interpolating for new nodal values, and proceeding with the next GA refinement runs.

2.8. GA windows

Another technique proposed to dramatically reduce the number of optimized variables at a time is the GA windows sweep-through technique. One or more lines of nodes extending across the full width of the flowfield, perpendicular to the primary flow direction, is chosen as a subdomain for the GA solution. This *GA window* of nodes is solved, then moved downstream. The window width can be anywhere between one line to the whole domain (in which case its effect disappears.) The window must move across the whole domain, covering all interior nodes; this constitutes a *GA sweep*. Nodes not being optimized inside the window are held fixed, where an essential boundary condition is temporarily enforced for all unknowns. A schematic of the GA-windows used in the potential flow problem is shown in Figure 3.

In the potential flow problem, one-line GA windows were found to be most effective and were used to produce the results reported. We note that whenever large recirculation zones are expected, such as in Sections 4 and 5, these windows can take any suitable shape to

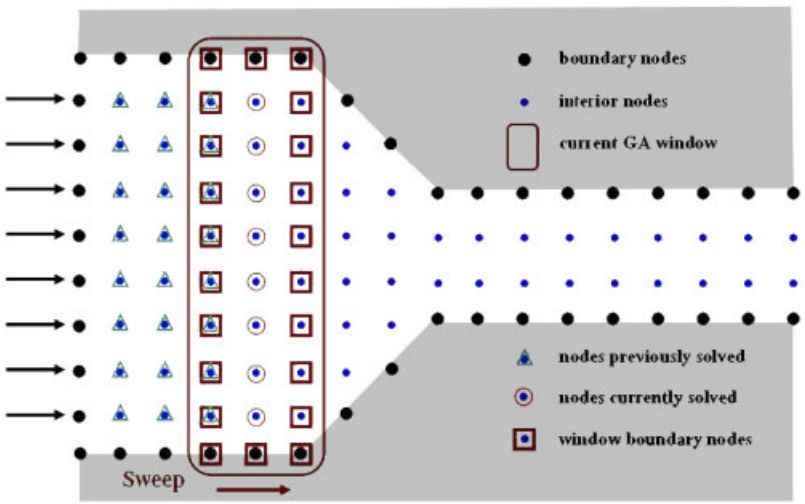

Figure 3. Flow through a contracting channel and an example of a single whole line GA window.

account for the flow behaviour, such as rectangular patches, for example. Results of Sections 4 and 5 were produced using a single GA window, thereby eliminating its effect. While this choice is not optimum, it provided consistent results for all flows involving recirculation and more than one unknown. Further research should go into optimizing the size, shape and path of these GA windows.

If the GA does not come adequately close to the preset fitness criterion, additional GA sweeps can be done for a single level of refinement. A second sweep of the GA over nodes already optimized can only improve the solution since the nodes on the GA window boundary will be more accurate having been optimized in the previous sweep-through.

2.9. Population shuffling

To guard against premature convergence and the occasional stagnation of the algorithm, a population shuffling mechanism is added. It can be activated at predetermined sweeps, e.g. at every *i*th sweep, or whenever no progress is detected from the previous sweep to the current. The *shuffle operator* takes the elitist and adds a fraction of the fitness information used in the fitness-guided mutation to *all* alleles, copying the newly formed chromosomes in place of older ones. In general, it was found that shuffling between 20 and 30% of the population every 4–5 GA sweeps raises the average fitness—enhancing the GA performance—and prevents premature convergence in many runs.

2.10. Gradient search and smoothing

There has been a noticeable move recently toward hybrid schemes that alternatively use both evolutionary and gradient search methods in a way that greatly improves the speed and precision of the search [30, 31]. In this algorithm, a gradient search is occasionally stepped into to find a niche that would concentrate the GA search and/or give new directions. The

search is usually applied every so often to a small fraction of the population and with what represents very small relaxation factors.

While the following minor addition to this GA is not quite a true hybridization of the algorithm, it was found that the final result of the GA can benefit from a little *smoothing*, mainly to smooth out rough edges in the fields and enhance the output figures. At the end of each refinement level the chromosomes are passed to a gradient search routine, e.g. the SIMPLER finite volume routine, with relaxation factors of 0.05 or less which are just enough to smooth out the variable fields while not disturbing the quality of the solution.

This operator can also be applied occasionally during the evolutionary search to perform a task that is the opposite of what the previous (shuffle) operator does. It was noticed that the delayed alternating application of these two operators greatly enhances the GA. A possible delayed alternating application may consist of applying the former operator at the third sweep, the latter operator at the sixth sweep, the former again at the ninth sweep, and so on.

2.11. Convergence criteria

As in most numerical simulations, a GA output 'solution' is defined as the best solution at the time of the voluntary stoppage of the simulation. For this GA, there are two levels of convergence: the convergence of the individual GA windows, and the convergence of the overall GA refinement level. For the former, instead of following the usual GA strategy of specifying *a priori* the number of generations until the algorithm is stopped, the GA is let to evolve until (i) the normalized fitness of the elitist reaches the maximum value of 1, or, (ii) the fittest individual does not improve for a certain number of GA window generations, e.g. on the order of 10 generations.

For any level of refinement, the greater sweeping-through process is stopped when the fitness does not change for a given number of GA sweeps (around 20,) or, preferably, when the maximum residual of the elitist reaches the preset limit. When this happens, the GA moves to the next refinement level, if any remain. On the last level of refinement, the algorithm was considered successful only when the maximum residual of the elitist was less than this preset limit. For most calculations, the preset convergence limit was set to 1.5×10^{-3}; this usually corresponds to about one hundredth of one percent of the value of the average property value.

2.12. The GA in pseudo-code

Program 2.1 is a simplified version of the pseudo-code that represents the GA described above and used in this work. The code operates on five levels; from inside out, they are: the local GA, GA windows, optimized variables, global sweeps and global refinements. In the next three sections the algorithm is tested using three different problems in fluid flow. The GA secondary capability to work a fluid flow problem from start to finish is demonstrated in the first example, while the main intended function of the GA to serve as an auxiliary solver when common techniques face difficulties is demonstrated in the second and third examples. The various objective functions used to measure the relative fitness of the individuals are discussed in detail. The interface known as the fitness function between the objective function and the GA is also discussed. All reported computations were performed on a 2.0 GHz Pentium 4 platform with 512 MB of RAM.

Program 2.1

```
The main genetic algorithm
  read parameters and initialize populations
  for refinement = 1 to maximum refinement do
    calculate global fitness
    repeat {global sweep generations}
      for variable = 1 to number of optimized variables do
        loop {over GA windows}
          calculate the GA window fitness
          repeat {local GA window generations}
            select parents
            crossover
            mutate
            reinstate elitist if option selected
            calculate GA window fitness
          until GA window convergence condition is met
        end loop
      end for
      calculate global fitness
      shuffle population if applicable
      gradient search if applicable
    until global converge condition is met
    refine mesh and interpolate
  end for
  smooth solution
  write results
```

3. GA FOR POTENTIAL FLOW THROUGH A GRADUAL CONTRACTION

3.1. Model problem and governing equations

As a first test of the GA, it is applied to the problem of a gradually contracting channel flow. The flow modelled is an incompressible, steady flow through a $2-1$, 45° gradual contraction channel, as depicted in Figure 3. For a number of flow situations, such as creeping Stokes flow and some non-Newtonian flows, recirculation zones are negligibly small or non-existent. These flows can be effectively modelled using the irrotational potential flow model. The stream function that describes the flow satisfies Laplace's equation,

$$\nabla^2 \psi = 0 \tag{4}$$

3.2. Objective function

The general features of the GA follow those described in Section 2 with regards to encoding, operators, convergence criteria, etc. The fitness calculation part is arguably the main part of any GA and this will be described in this and the next subsections. The fitness part of a GA determines the relative fitness of the chromosomes so that the selection scheme has the proper information to select parents for the next generation. It is noted about the terminology that

the *fitness function* is sometimes called the objective function, but it is the *objective function* that actually differentiates between chromosomes and it is the job of the fitness function to assign fitness values according to this differentiation.

For the objective function any valid method to differentiate between the 'goodness' of chromosomes as potential solutions can be used; finite difference-based residuals are used here. To obtain this the 2D domain is meshed with a uniformly-spaced Cartesian grid, and Equation (4) is discretized using second-order central differences,

$$\frac{\partial^2 \psi}{\partial x^2} \approx \frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{\delta x^2} \tag{5a}$$

$$\frac{\partial^2 \psi}{\partial y^2} \approx \frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{\delta y^2} \tag{5b}$$

Substituting these approximations into (4) and simplifying, we arrive at the nodal equation for the stream function at node (i,j),

$$\psi_{i,j} = \tfrac{1}{4}(\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1}) \tag{6}$$

giving the residual r at an interior node (i, j) as

$$r_{i,j} = \left|\tfrac{1}{4}(\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1}) - \psi_{i,j}\right| \tag{7}$$

3.3. Fitness function

There are several ways in which the fitness can be defined. The actual fitness of a chromosome can be calculated as the inverse of the maximum residual r_m, defined as

$$r_m = \max_{\substack{1 \leqslant i \leqslant N_x \\ 1 \leqslant j \leqslant N_y}} r_{i,j} \tag{8}$$

where N_x and N_y are the maximum number of nodes in the x- and y-directions, respectively. The inverse definition of the fitness function will give low fitness values for high residuals and very high ones for close to zero residuals. There are many advantages, however, to having the fitness values bounded in a known interval, for example, to compare solution quality across GA variables, parameters, geometries, refinements and runs. One way to accomplish this is to define the fitness as the exponential of the maximum residual among all interior nodes, which bounds the fitness to be in the interval [0,1]. In addition to the maximum residual r_{m}, which is the maximum residual among all nodes, two other fitness criteria can be used: a global residual r_{g}, which is the normalized sum of residuals of *all* optimized nodal points N_{np}, and a boundary residual r_{b}, which is the normalized sum of residuals of only exit boundary nodes N_{b}. These are defined as

$$r_{\mathrm{g}} = \frac{1}{N_{\mathrm{np}}} \sum_{\substack{1 \leqslant i \leqslant N_x \\ 1 \leqslant j \leqslant N_y}} r_{i,j} \tag{9}$$

$$r_{\mathrm{b}} = \frac{1}{N_{\mathrm{b}}} \sum_{\substack{1 \leqslant i \leqslant N_x \\ 1 \leqslant j \leqslant N_y}} \{r_{i,j} | (i,j) \in N_{\mathrm{b}}\} \tag{10}$$

Whereas r_{m} is a measure of the worst point in the solution, the global residual r_{g} represents the collective quality of the solution and the boundary residual r_{b} emphasizes the adherence of the flow to a zero longitudinal gradient at exit.

Incorporation of these additional indicators gives a useful balance that guards against underestimating a good solution that has relatively high residuals at a couple of nodes but near-zero residuals elsewhere, *and* against overestimating an average solution that has moderate residuals nearly everywhere. The boundary residual specifically measures the residuals of nodes at the exit, representing the quality of the exiting flow and whether it is parallel to the channel or not. The boundary residual factor is activated only when a GA-window boundary (cf. Section 2.8) coincides with global exit boundary nodes. Therefore, the fitness function can be written as

$$f = \frac{(w_{\mathrm{m}} \mathrm{e}^{-r_{\mathrm{m}}} + w_{\mathrm{g}} \mathrm{e}^{-r_{\mathrm{g}}} + w_{\mathrm{b}} \mathrm{e}^{-r_{\mathrm{b}}})}{w_{\mathrm{m}} + w_{\mathrm{g}} + w_{\mathrm{b}}} \tag{11}$$

where w_{m}, w_{g} and w_{b} are weights to control the emphasis placed on the three components of the fitness function; values $w_{\mathrm{m}} = w_{\mathrm{g}} = w_{\mathrm{b}} = 1$ were found to be adequate and were used throughout the calculations presented in this section. For all weighting combinations possible, the fitness should approach the maximum value of 1 as the residuals tend to 0.

3.4. Numerical results and discussion

The GA was run with population sizes 2, 4, 6, 10, 18, and 40. The population is initialized by filling in the matrix of the unknown stream function values with random numbers bounded by the lowest and the highest streamline values. Results reported here were made with a population size of 18 and crossover and mutation probabilities of 0.9. The fitness value for the fittest individual and the average fitness of the generation are plotted in Figure 4 against the GA sweeps performed. A comparable plot of the corresponding maximum and global residuals of the fittest individual is shown in Figure 5. It can be seen that the fitness goes from about 0.5 to near unity as the residuals drop four orders of magnitude to the preset convergence limit. At the beginning of each refinement level, we note a small dip in fitness; this is due to the interpolations needed to go from coarse solutions to the finer grids—the current level of grid sequencing.

As mentioned previously, one of the main motivations for using a GA for fluid mechanics problems was its apparent insensitivity to the stiffness many problems exhibit. An appropriate measure of this property might be the values of the residuals throughout the domain after a converged solution is reached. Inspection of the residual magnitudes of the converged solution reveals that in any region of the flow field the residual magnitudes are within 30% of each other. Specifically, the heuristic search exhibited the same level of ease (or difficulty) optimizing regions of uniform flow, such as the middle of the channel, as it did optimizing regions near the contracting walls. In addition, the magnitudes are more or less comparable over much of the domain–smaller residuals exist in the narrow section where solid boundaries are closer together on average and nodes are better optimized. This result shows that the GA was rather indifferent to the locations and magnitudes of gradients, a main cause of divergence in gradient-based numerical methods.

Barring complete elimination of diversity, a GA is expected, given enough time, to converge to the global maximum fitness. With smart introduction of new blood via mutation, random walk theory guarantees the long-term convergence to the desired solution quality. Therefore, a practical measure of the success of a GA simulation can be taken to be the time it takes to converge. For some combinations of population sizes and probabilities of mutation the GA

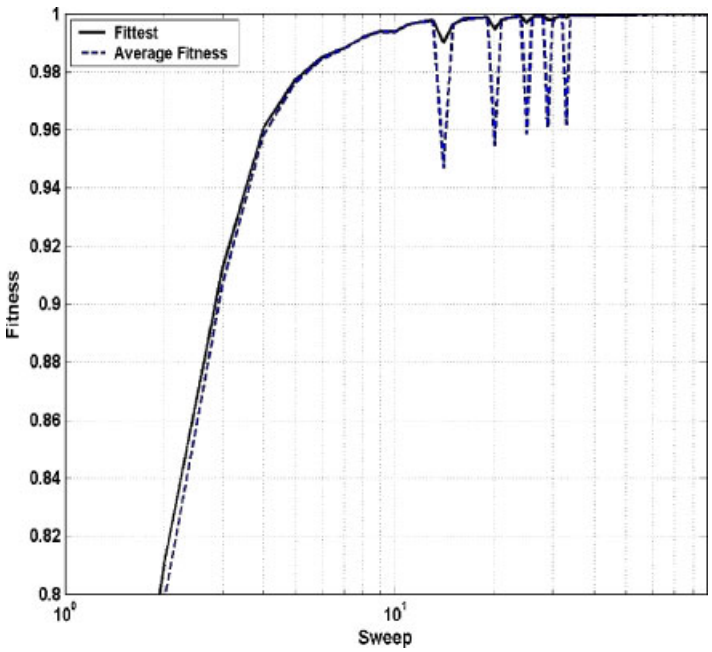

Figure 4. Fitness of the best chromosome.

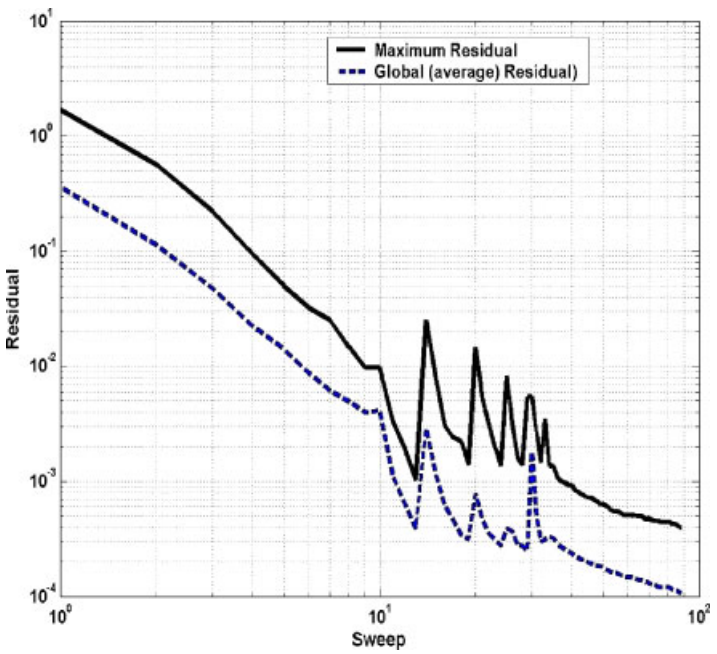

Figure 5. Maximum and global residuals.

did not converge, i.e. the maximum residual of the elitist did not fall below the convergence limit, *and*, no changes were noticed in fitness after more than about 20 GA sweeps. For the combinations that converged in less than 60 min, Figure 6 shows times to convergence versus the mutation rate for a number of population sizes.

It can be seen from Figure 6 that the best GA performance was for small to medium population sizes and high mutation probabilities in general. It is noticed that very large and very small population sizes were consistently outperformed by the moderate one. The reason is that for large populations much time goes into *unneeded* operations on chromosomes after the best possible result has already been determined. Conversely, for very small populations, not enough building blocks of the sought-after solution are present in the population. For these populations, low mutation rates were not adequate to consistently probe/scrutinize the global extremum region, resulting in the algorithm 'giving up' on the search. This is clearly evident in mutation rates below 0.3 for population size 4. It should be noted that for the extremely low population size of 2, the GA did not converge for any mutation rate; a population size 2 is best negotiated using *evolution strategies*, a close cousin to GAs designed to operate using one parent and one offspring. In general, for the current problem complexity, a pseudo-optimum mutation rate of around 0.7 can be deduced from the plot.

One of the notable differences between a regular GA and the refinement/line-by-line GA implemented here is the very large number of generations that can be simulated; a typical GA run performs between 500 and 1000 generations, whereas because of the greatly reduced number of variables optimized at a time, our GA performed on the order of 1 million total generations.

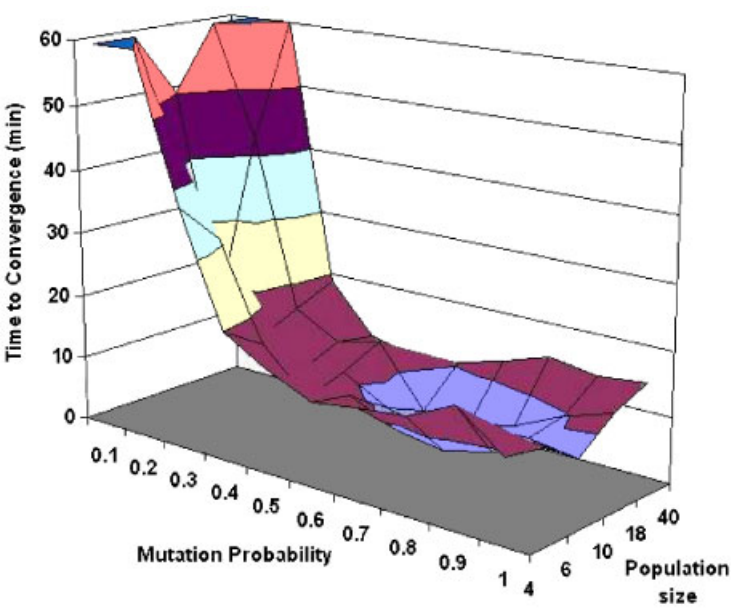

Figure 6. Time of convergence (minutes) vs probability of mutation.

Another major observation on the optimization of one variable only is that for real-coded chromosomes the crossover operator had very little effect on the convergence of the GA. For smaller populations (10 or fewer chromosomes), the crossover effect was practically negligible and mutation was the only mechanism through which the GA progresses. This is important since it was noticed that smaller population sizes performed better, i.e. converged solutions took less time to emerge. These findings confirm previous observations made by many researchers [19, 25, 26].

An instructive plot to look at is one that shows relative time spent by the GA on each of its main operators. Figure 7 shows that the bulk of the time is spent on mutation. This is warranted because mutation is the primary driver for convergence. Crossover also takes a substantial portion of the calculation time, and unless selection information is supplied directly to the mutation operator, stepping through the crossover routine will continue to be important because of its incorporation of the selection information, regardless of the effectiveness of the crossover process itself. For more elaborate fitness calculation schemes the fitness portion of the calculation time can increase significantly, (cf. Figure 21.) Figure 7 is a good place to start when increased efficiency of the GA is desired.

The uniform successive refinements combined with the moving GA-window strategy were proven effective in reducing the number of optimized quantities per generation, allowing for a considerable enhancement in performance in the larger sweeps. These results show that small to moderate population sizes combined with high mutation rates were most effective. Convergent solutions for a 65 000 node mesh were obtained in many cases in under 5 min. Previously reported work on GAs applied to fluid flow was limited to simple flows dealing with much smaller meshes [17, 32].

It is noted that this potential flow problem can be effectively solved using other numerical methods perhaps in less time than the GA required. The evolutionary solver is intended to be used primarily as an auxiliary solver that is activated when common methods of solution run into difficulties. Findings made for this simple problem are extended to areas of highly stiff

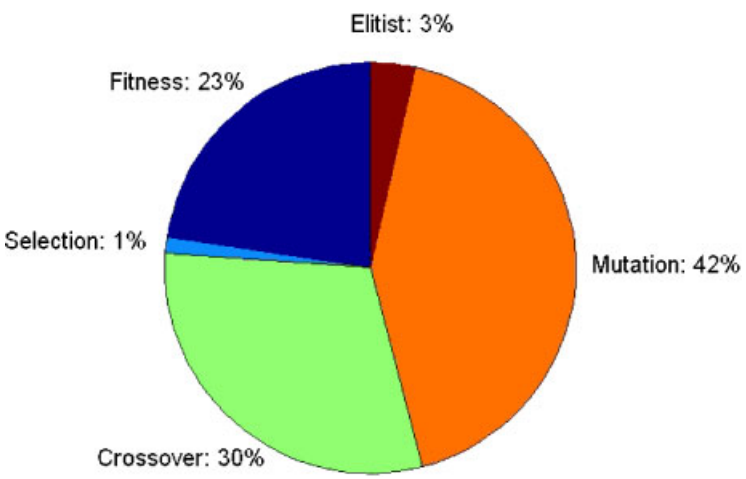

Figure 7. Percentages of calculation time for each GA component.

problems where conventional techniques suffer from divergence or slow convergence. These problems include a number of viscous flow simulations and many non-Newtonian polymeric liquid flows often encountered in industry; they are addressed in Section 5.

But next, we focus on the general way this evolution program is used to solve problems of viscous fluid flows with considerable recirculation zones. In practical flow problems, the effects of viscosity are always important and must be included in the analysis. As viscosity is added, recirculation patterns arise, presenting a new challenge to the GA; this will be addressed in the next section.

4. GA FOR VISCOUS FLUID FLOW

4.1. Model problem and governing equations

The flow modelled is assumed to be laminar and isothermal, obeying the steady, incompressible, Newtonian Navier–Stokes equations. The dimensional mass and momentum conservation equations in primitive forms are

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad \text{(continuity)} \tag{12}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right] \quad (x\text{-momentum}) \tag{13}$$

$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left[\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right] \quad (y\text{-momentum}) \tag{14}$$

One difficulty in solving the incompressible Navier–Stokes equations iteratively is the unknown pressure field. The pressure is indirectly specified via the continuity equation in that the *correct* pressure field ensures a divergence free continuity equation. To overcome this complication, in many CFD implementations the Poisson equation for pressure is substituted for the continuity equation. Another difficulty is that when the pressure gradient terms in the momentum equations are discretized using pressure differences the problem of the zigzagged pressure field arises. A common remedy for that is the use of staggered grids to calculate velocities. Both these modifications are implemented in the finite volume discretization below.

4.2. The discretized equations

The governing equations are discretized using the *finite volume method*. The calculation domain is subdivided into a number of non-overlapping control volumes over which the equations are integrated, resulting in a set of discretized equations in terms of the values of known and unknown quantities at grid points. The main grid points, located in the middle of the control volumes, are used for the pressure equation. The u and v velocities are calculated on meshes that are staggered in the x- and y-directions, respectively. Figure 8 shows part of the domain and a sample non-staggered grid is show in Figure 9. The discretized x- and

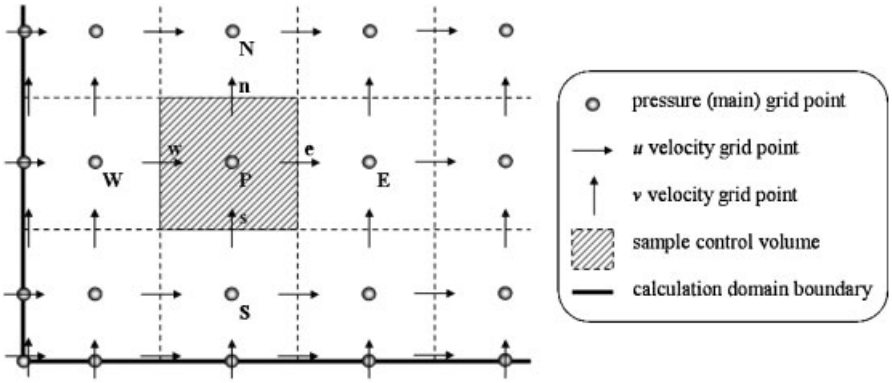

Figure 8. Staggered finite volume mesh.

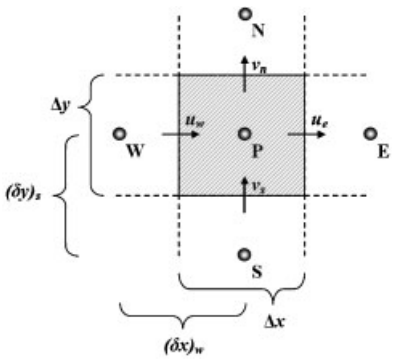

Figure 9. Sample control volume.

y-momentum equations are

$$a_p u_p = a_s u_s + a_e u_e + a_n u_n + a_w u_w + A_p(p_W - p_P)$$
$$= \Sigma a_{nb} u_{nb} + A_p(p_W - p_P) \tag{15}$$

$$a_p v_p = a_s v_s + a_e v_e + a_n v_n + a_w v_w + A_p(p_S - p_P)$$
$$= \Sigma a_{nb} v_{nb} + A_p(p_S - p_P) \tag{16}$$

and the equation for obtaining the pressure field is the Poisson's equation defined on the main grid,

$$a_P p_P = a_S p_S + a_E p_E + a_N p_N + a_W p_W + b \tag{17}$$

The coefficients a's, and b, and the areas A_p are defined in Reference [20], and the basic derivation of the discretized equations is given in Reference [33].

4.3. Objective and fitness functions

As in the last section the objective function is defined to be the residual of these discretized equations, only this time we have three different equations that could be used to optimize three different variables. Three residuals, r^p, r^u and r^v, corresponding to the three equations above are defined as

$$r^p = | - a_P p_P + a_S p_S + a_E p_E + a_N p_N + a_W p_W + b| \tag{18a}$$

$$r^u = | - a_p u_p + \Sigma a_{nb} u_{nb} + A_p(p_W - p_P)| \tag{18b}$$

$$r^v = | - a_p v_p + \Sigma a_{nb} v_{nb} + A_p(p_S - p_P)| \tag{18c}$$

The fitness function definition of the last section is used again here. There are a number of options for the variable(s) the GA can attempt to optimize in each sweep. The algorithm can solve the u velocity by itself, the v velocity by itself, pressure by itself, only velocities, or all variables simultaneously. The three residuals r^p, r^u and r^v are considered if p, u or v are being optimized, respectively. When more than one variable is being optimized, the maximum residual of all equations involved is considered the residual at that node, viz.,

$$r = \max(r^p, r^u, r^v) \tag{19}$$

The maximum, global and boundary residuals r_m, r_g and r_b, are defined in a similar fashion to Section 3.3, i.e. Equations (8)–(10). The fitness function is also defined as

$$f = \frac{w_\mathrm{m} \mathrm{e}^{-r_\mathrm{m}} + w_\mathrm{g} \mathrm{e}^{-r_\mathrm{g}} + w_\mathrm{b} \mathrm{e}^{-r_\mathrm{b}}}{(w_\mathrm{m} + w_\mathrm{g} + w_\mathrm{b})} \tag{20}$$

Results reported in this and the next section were obtained using simultaneous optimization of all three variables in a GA sweep.

4.4. Numerical results and discussion

The geometry of this problem is that of a 2-D channel containing a 2-1 expansion over a backward facing step with one inlet and one exit. The flow has a Reynolds number of 200. The mesh contains 3721 uniformly placed main grid points. A channel height $H = 3.6\,\mathrm{m}$ and a step height of $H/2 = 1.6\,\mathrm{m}$ were used. The fluid density $\rho = 0.1\ \mathrm{m}^3/\mathrm{kg}$ and the dynamic viscosity $\mu = 1.794 \times 10^{-3}\ \mathrm{kg/m\,s}$ were assumed constant. The inlet velocity profile is that of a fully developed channel flow and an exit more than $5H$ downstream was enough to recover a fully developed exit flow profile.

The flow entering the channel is assumed to be fully developed. For an initial guess for the GA a few iterations of a conventional gradient-based scheme are performed to get an initial picture of the signs and magnitudes of the various velocities and pressure in the field since it would take the GA a long time to weed out non-physical solutions and figure out the general features of the fields. It is stressed that this is the intended way the GA will be used; in other words, the GA will only be used when gradient-based schemes stagnate or diverge.

Table I. GA parameters for the Newtonian backward facing step.

Nodes	Popsize	P_c^a	P_c^u	P_m^r	P_m^u	P_m^a	P_m^b	shuffle
3721	20	0.2	0.2	0.2	0.01	0.001	0.001	20% every 4th

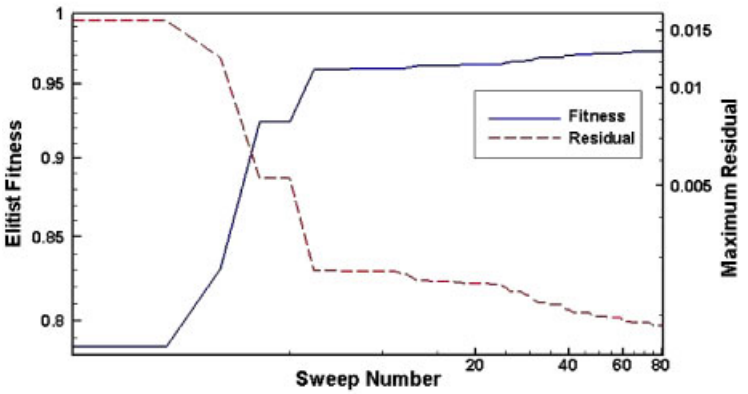

Figure 10. Progress of the elitist fitness and the maximum residual after each GA sweep.

To provide the necessary initial guess, and also to serve as a means of comparison and validation, the SIMPLER finite volume-based algorithm was used [33]. For this problem, the GA uses an initial guess obtained from running the SIMPLER algorithm for 30 iterations. (For the described problem the SIMPLER solution starts to be mass-conserving and begins to show physically possible features at the 27th iteration.) To simulate an even worse starting point, this SIMPLER output (velocities and pressure) is perturbed by 50% uniformly distributed random error and used to initialize the GA population.

The crossover operator is reinstated for this relatively more complex flow. The total probability that the chromosome is operated upon by a genetic operator (crossover and/or mutation) is kept at the same levels as the potential flow problems only with a redistribution of the operators probabilities, cf. Section 3.4. Additionally, because of the presence of three interdependent variables (u, v, and p) instead of one for the stream function, the previously deduced optimum population size was tripled. Crude parametric tests confirmed the appropriateness of these scalings. The parameters used to run the GA are listed in Table I.

A plot of the progress of the GA, represented by the history of the elitist fitness and the maximum residual, is shown in Figure 10. The evolution starts with a fitness of 0.78, which corresponds to a residual of about 0.016, and increases to well above 0.97. We note how the fitness of the elitist rises steadily as the maximum residual drops to around the preset convergence limit of 1.5×10^{-3}. The total GA sweeps performed were 81 and the total corrected number of generations (taking into account generations at the GA window level and generations of the optimized variables) was around 800. The GA ran for a total of approximately 17.6 minutes.

Contours of the velocity magnitude and pressure for the converged GA solution are shown in Figures 11 and 12, respectively. Scaled velocity vectors are also superimposed on these

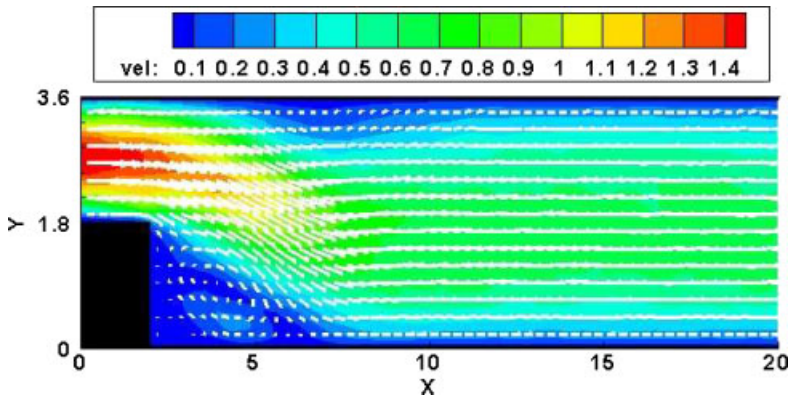

Figure 11. GA velocity magnitude contours with superimposed velocity vectors for a Newtonian flow over a backward facing step.

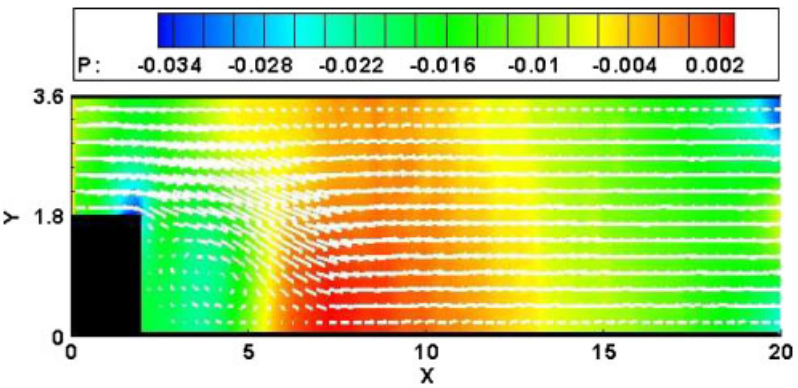

Figure 12. GA pressure contours with superimposed velocity vectors for a Newtonian flow over a backward facing step.

figures. Judging by these figures and the residual magnitudes, it can clearly be seen that the GA successfully arrived at a physically correct solution.

To assess the 'degree of convergence' of the GA we compare it with the fully converged solution obtain independently by SIMPLER. Choosing the u velocity as the comparison variable, we measure the difference in the nodal value of the velocity at each grid point and find out that the GA solution is converged to within 1.4% of the SIMPLER solution. Degrees of convergence of the other solution variables are expected to be of similar or lesser magnitudes.

Whether the converged solution supplied by SIMPLER is to be taken as an absolute correct solution is a point to be pondered. The SIMPLER solution for this Newtonian backward facing step problem was compared to benchmark published solutions and has shown very good overall agreement [34]. However, one thing we note about SIMPLER is that it is first order in its representations of gradients, just like the objective function used here. This somewhat

degrades the output solution when compared with higher order discretization. Additionally, because of the presence of circulation zones where the flow is oblique to the mesh lines, there is the issue of false diffusion, addressed in Reference [33] with regards to SIMPLER and more generally in Reference [35], and to which SIMPLER has no remedy.

Last but not least, we look at the question of SIMPLER's accuracy in regions of large gradients. This is the main area where we expect the GA to outperform, on a local level, gradient-based methods like the SIMPLER algorithm. In particular, it is interesting to find out whether a solution can be improved beyond the converged solution determined by SIMPLER (judged by the residuals magnitudes of the governing equations.) As a sample test problem, we consider a SIMPLER solution of the same backward facing step problem taken to full convergence, i.e. a pressure correction magnitude of around machine accuracy (10^{-14}). For this solution, the residuals of the three governing equations are calculated using the objective function described and are plotted in Figure 13. We note that the residuals near the tip of the step are clearly the highest.

Next, we turn to the GA and run it using this solution as an initial guess and check whether it is able to further reduce the maximum residual, despite the inability of SIMPLER to do so. The progress of the elitist fitness and maximum residual is shown in Figure 14 where we notice that the GA was indeed able to reduce the maximum residual considerably. The GA is able to get around difficulties associated with large gradients near the tip of the step and optimize those nodes in a way SIMPLER was not capable of, even though the two algorithms used virtually the same finite volume discretization method and governing equations. The only difference is that the GA calculated the absolute residuals of the equations and used them to measure fitness. More is said about this quality of the GA in Section 5.4.

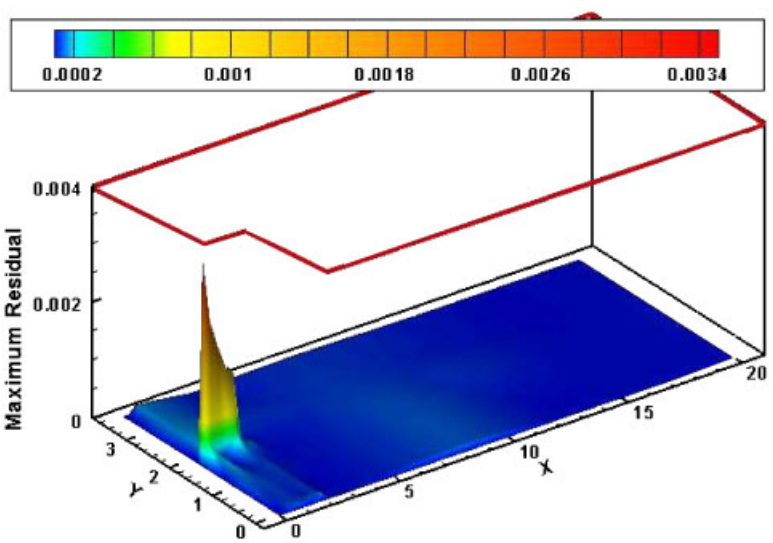

Figure 13. Residual map of the flow domain as given by a converged SIMPLER solution for the backward facing step problem.

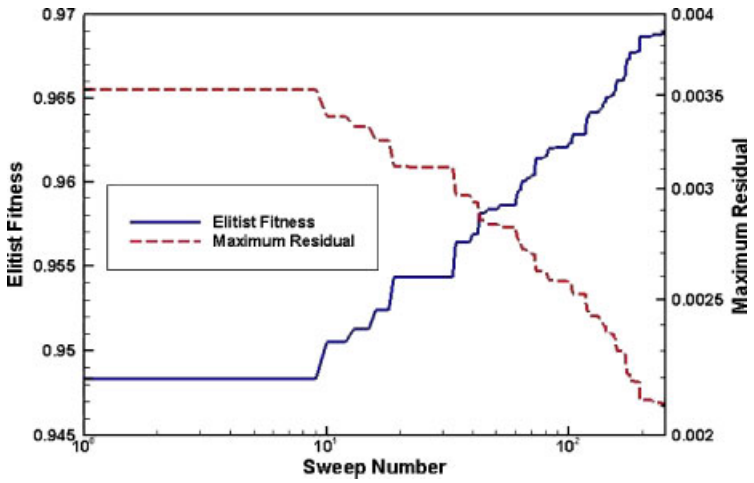

Figure 14. Progress of the elitist fitness and the maximum residual at each GA sweep starting from a fully converged SIMPLER solution.

4.5. *Convergence rate*

An important question regarding any numerical method is the rate of convergence it exhibits. The treatment of this issue is paramount for new methods intended to solve problems from start to finish, but is also important to address for auxiliary solvers intended to be activated when basic methods fail—such as the evolutionary solver at hand. A survey of the literature on evolutionary algorithms quickly shows the clear lagging of theory behind practice in this field. This lagging is most evident when one considers the scarcity of theoretical treatment of almost all evolutionary algorithms when it comes to studying the convergence characteristics (in the CFD sense) of GAs as a whole [36]. Some researchers have studied convergence models of various genetic operators, such as selection and mutation schemes, while others studied the local convergence using finite Markov Chains [37], which is limited in its scope.[§] Somewhat better treatment of the theory exists for *evolution strategies* (ES) [38], and much effort is being devoted lately toward the ever-growing area of multiobjective evolutionary algorithms [39].

With these limitations, we are forced to define our own convergence rate for this highly customized GA. We can call it, perhaps more appropriately, *computational complexity* (or scalability), and think of it as the relationship between computation time and the problem size, which is directly related to the number of optimized nodes. As a test case, the GA tackles the simple flow between two infinitely long parallel plates. The inlet boundary has the fully developed velocity profile. The problem was run using five different grids sizes: 21×21, 31×31, 41×41, 51×51 and 61×61. One GA window was used through all these runs in

[§]The finite Markov states are used to represent the GA generations, but they assume *infinite* populations, which is at odds with practice. Furthermore, they are limited to canonical GAs (GAs with no elitism) which are provably non-convergent to the global optimum.

Table II. GA Parameters for straight channels.

Popsize	P_c^a	P_c^u	P_m^r	P_m^u	P_m^a	P_m^b	shuffle
30	0.3	0.1	0.01	0.01	0.001	0.001	20% every 4th

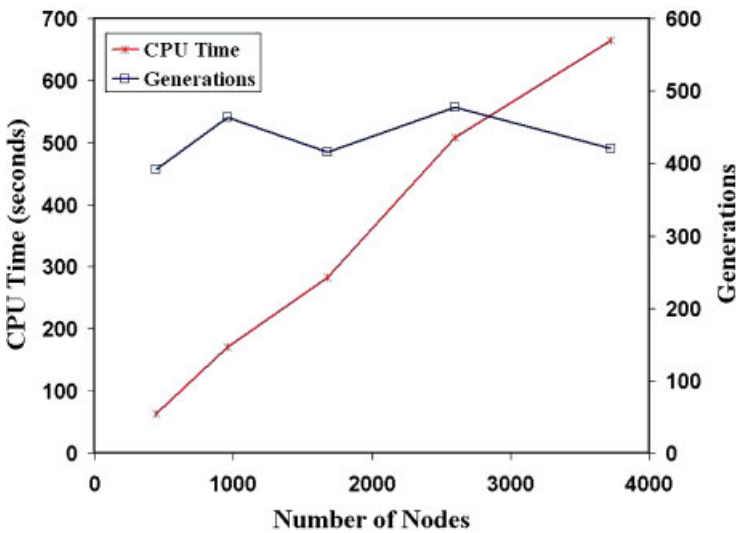

Figure 15. The GA rate of convergence for the straight channel.

order to be consistent across all runs, and make convergence time/generation solely dependent on the number of optimized nodes.

The GA is asked to start with random, yet realistic, guesses of the velocity and pressure fields and is expected to arrive at the fully developed velocity profiles with a consistent pressure drop. For this straight channel flow we expect the normal velocity to be small, thus the v velocities are initialized randomly from a uniform distribution between ± 0.15. We also expect the u velocities not to exceed the maximum inlet (non-dimensional) velocity $U_{\max} = 1.5$, and therefore the u's are initialized randomly from the interval [0, 2]-nodes closer to the walls have a higher probability of being assigned a lower value. In all cases the GA converged to the correct solution with the preset convergence limit. Relevant GA parameters are shown in Table II.

Figure 15 shows a plot of the computation time and the number of generation until convergence as functions of the problem size, taken from the GA solutions to the straight channel problem above. The relationship between the computation time and the number of nodes in the domain for this GA appears to be linear. The number of generations does not seem to be dependent on the problem size, which of course implies that for larger problems a generation (GA sweep) takes a longer time to meet its local convergence criterion.

4.6. Closing remarks

The problem of an incompressible flow over a backward facing step was treated in this section. It was shown that the GA is able to successfully arrive at a solution starting from a not-so-good initial guess using the various genetic operators and diversification techniques. The GA was even able to improve a solution past the point when the finite volume technique stopped completely. This problem, however, is still not considered a formidable problem for most gradient-based techniques. It was yet another demonstration that, given a rough guess about the general features of the flow field, the GA can take the problem to convergence.

Unlike GAs, the role of initial guesses for gradient-based techniques is not that paramount; there, relaxation factors play a much larger role for they govern how much of the gradient-based information is used to update the solution from one iteration to the next. Relaxation factors control the speed of the solution convergence but also are the source of divergence risks. On the other hand, good initial guesses go a long way toward shortening the time it takes an evolutionary algorithm to converge; but the magnitudes and behaviour of gradients of flow properties play a very small role in determining how they progress, as is evident from the problems considered. This fact will be of great consequences in the next section when we consider flows of non-Newtonian fluids and see how gradient-based methods face arduous difficulties dealing with fluids with low flow indices.

5. GA FOR A POWER LAW NON-NEWTONIAN FLUID FLOW

5.1. Overview

Fluids that have high molecular weights exhibit behaviours that cannot be explained/predicted by Newtonian models. By high here, we mean substances with molecular weights of more than about 1000 amu.[¶] The two demonstrative polymers used here, polyethylene and polyvinyl chloride, have average molecular weights in the ranges 3.1×10^6–5.9×10^6 and 6×10^4–14×10^4, respectively. They are used as examples because of their relative simple structures, long history, and common use in process industry.

Numerical simulations of non-Newtonian fluids, however, are in the class of the most difficult problems in engineering [40, 41]. The dependence of the shear stress on shear rate and a viscosity that, in turn, has some dependence on shear rate gives any numerical method a good challenge, no matter how simple this dependence is. Nevertheless, the mechanics of non-Newtonian fluids are not drastically different from those of their Newtonian counterparts. The difference in the analysis often comes down to the way the viscous stress tensor is represented. In this paper, we will focus on the power law model, which falls under the generalized Newtonian category and which is sufficient for describing basic non-Newtonian phenomena such as shear-thinning and thickening.

5.2. Model problem and governing equations

Channel flow over a backward facing step is modelled again. The flow considered is laminar, isothermal, steady and incompressible. The basic governing equations the flow obeys are

[¶]Compare the values of 1000+ amu for non-Newtonian fluids such as candle wax (3000) and Maleic anhydride-methyl vinyl ether (250 000) with hydrogen (2), water (18), air (28.9), CO_2 (44), and mercury (201).

the Navier–Stokes equations only with a power law model for viscosity used to describe the viscous stress term in the momentum equations. A famous, simple power-law model is that of *Ostwald and de Waele* (commonly called *the power law model*,) which has two parameters, the consistency index m, and the flow index n, and which for a simple unidirectional flow is written as

$$\tau_{yx} = -m \left| \frac{\mathrm{d}u}{\mathrm{d}y} \right|^{n-1} \frac{\mathrm{d}u}{\mathrm{d}y} \tag{21}$$

The consistency index m has units Pa s^n while the flow index n is dimensionless. Note that when $m = \mu$ and $n = 1$ the relation for a Newtonian fluid is recovered.

For the general 2-D flow of a *generalized Newtonian fluid*, the viscous stress tensor is given by

$$\boldsymbol{\tau} = -\eta\dot{\gamma} \tag{22}$$

where η is the apparent viscosity of the fluid. The power law model for the apparent viscosity involves the two parameters of the power law model as well as the magnitude of the strain rate tensor, defined as

$$\eta = m\dot{\gamma}^{n-1} \tag{23}$$

The magnitude $\dot{\gamma}$ of the strain rate tensor can be conveniently computed using the second invariant II (trace) of the tensor as

$$\dot{\gamma} = \sqrt{2II} = \sqrt{2\left[\left(2\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2 + \left(2\frac{\partial v}{\partial y}\right)^2\right]} \tag{24}$$

Computing the various apparent viscosities η requires appropriate interpolations of the u and v velocities around the main grid point and then substituting them into the power law formula below.

$$\eta_{pl} = m\,\dot{\gamma}^{n-1} = m\left(2\left[\left(2\frac{\partial u}{\partial x}\right)^2 + 2\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2 + \left(2\frac{\partial v}{\partial y}\right)^2\right]\right)^{(n-1)/2} \tag{25}$$

The apparent viscosity is determined outside the objective function based on the appropriate instantaneous velocity gradients, and used to update properties at each optimized node. This causes a few coefficients of the discretized momentum and pressure equations to change from the Newtonian case. Details of the coefficient are given in Reference [20].

Lastly, we note the limits on apparent viscosity for pseudo-plastic fluids:

$$\eta \longrightarrow 0 \quad \text{as } \dot{\gamma} \longrightarrow \infty \tag{26}$$

$$\eta \longrightarrow \infty \quad \text{as } \dot{\gamma} \longrightarrow 0 \tag{27}$$

While the shear-thinning is clearly manifest in pseudo-plastic polymeric liquids, the frictionless behaviour implied by the first limit seems unrealistic. More importantly, the infinite viscosity predicted at zero-shear-rate presents numerical difficulties for many algorithm and plagues the GA solver with the following problem: infinite viscosity leads to flatter velocity profiles

Table III. Power law parameters of some common polymers.

Polymer	m (Pa s^n)	n (dimensionless)
Polyvinyl chloride	1.7×10^4	0.25
Polyethylene	1.3×10^4	0.4
Polystyrene	2.8×10^4	0.28
Polypropylene	7.5×10^3	0.38
Polycarbonate	6.0×10^2	0.98

than experimentally observed and reported near the centers of channels where the strain-rates approach zero. We thus set two limits on the calculated apparent viscosity using the zero-shear viscosity μ_0 and infinite-shear viscosity μ_∞. These viscosities can often vary by factors of 100, 1000, or more for some liquids. The reported consistency index m is the zero-shear viscosity and is used as an upper bound on viscosity. 1/1000 of this value is used as the infinite-shear viscosity, thus allowing the apparent viscosity only to *thin out* as the shear rate increases. In other words, the bounded formula for the apparent viscosity of a pseudo-plastic fluid is

$$\eta_{\text{app}} = \min(m, \max(\eta_{\text{pl}}, m/1000)) \tag{28}$$

5.3. Objective and fitness functions

For these non-Newtonian cases, the modifications done to the discretized equations are not of essence; they only involve the details of computing the conductances at the control volume faces due to the change in the viscosity representation. Therefore, the same objective and fitness functions of Section 4, i.e. Equations (18) and (20), can be used.

5.4. Numerical results and discussion

A number of flow indices were tried with the SIMPLER finite volume-based algorithm to determine where it had the most trouble converging. The key parameter to investigate is the flow index n; for Newtonian fluids $n = 1$, and for pseudo-plastic non-Newtonian fluids n ranges from ~ 0.2 (rubber compounds) to 0.9+ (polycarbonate). The smaller n is, the more shear-thinning the fluid exhibits. Table III lists a few widely used industrial polymers and their consistency and flow indices. With the exception of n, problem description is the same as Section 4.4. Polyethylene and polyvinyl chloride will be specifically looked at; they have flow indices of 0.4 and 0.25, respectively.

The SIMPLER algorithm was run using these values of n with a number of relaxation factors ranging from 0.1 to 0.9, using increments of 0.1 at most. A solution was considered 'converged' when the maximum pressure correction coefficient reached 10^6 which corresponds to a maximum residual just below 10^{-2} in the equations of motion. As Figures 16 and 17 show, for $n = 0.4$ SIMPLER converged only for relaxation factors $\leqslant 0.22$, albeit very slowly, while for $n = 0.25$ it failed to converge for *any* constant relaxation factor above 0.1.

It should be noted that a SIMPLER solution can be obtained if one first obtains a converged Newtonian solution for the flow geometry then successively decreases the flow index using very small increments combined with very low relaxation factors, and taking each step to convergence, until the desired flow index is reached. The SIMPLER solution obtained by this

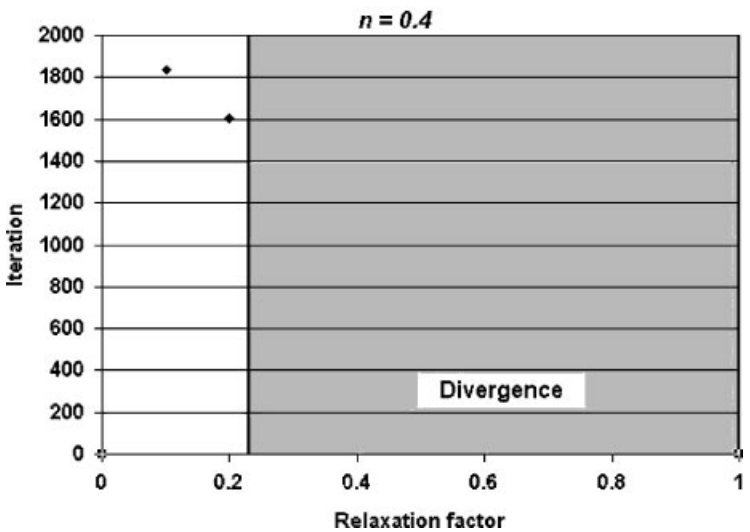

Figure 16. Minimum convergence iterations vs a number of relaxation factors for the SIMPLER algorithm with a 0.4 flow index power law parameter. Note: SIMPLER diverged for relaxation factors greater than 0.2.

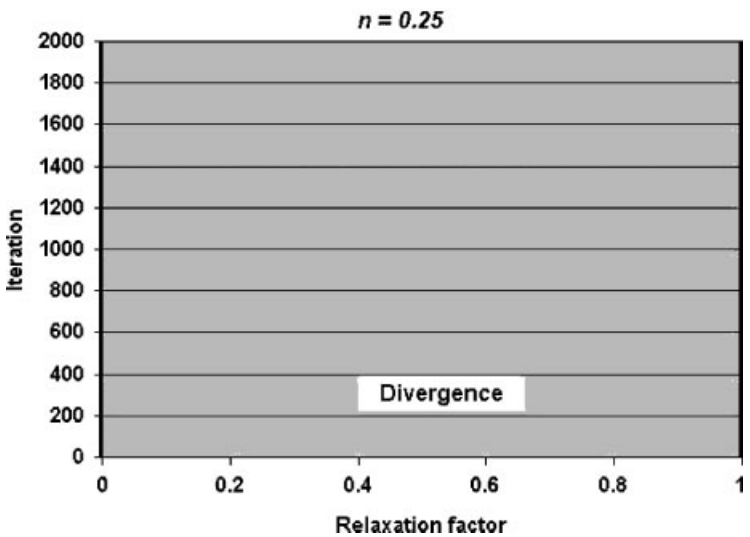

Figure 17. Minimum convergence iterations vs a number of relaxation factors for the SIMPLER algorithm with a 0.25 flow index power law parameter. Note: SIMPLER never converged for any relaxation factor above 0.1.

method, however, is very unstable: it diverges immediately upon restarting the solution with medium-to-large relaxation factors and/or suddenly changing the flow index by more than 0.05. This of course requires great experience and intuition on the part of the modeller to carry out the careful manipulations of these parameters.

Table IV. GA Parameters for the backward facing step with $n = 0.25$.

Nodes	Popsize	P_c^a	P_c^u	P_m^r	P_m^u	P_m^a	P_m^b	shuffle
1681	40	0.25	0.25	0.1	0.1	0.01	0.01	60% every 10th

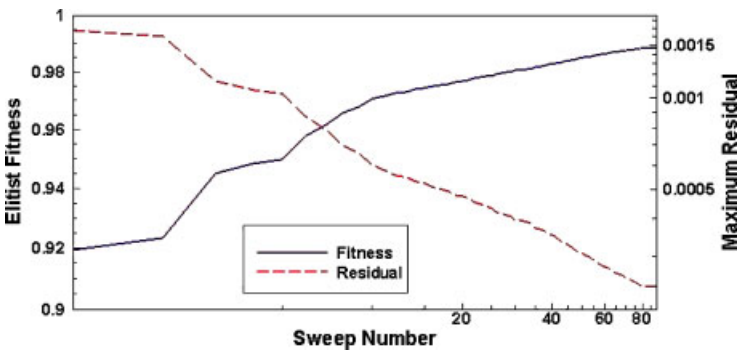

Figure 18. Progress of the GA represented by the fitness of the elitist and the maximum residual over the GA sweeps for a power law fluid with $n = 0.25$.

Because SIMPLER does not converge for any constant relaxation factor for this flow index, we are forced to look elsewhere for a good initial guess for the GA. A converged SIMPLER *Newtonian* solution was used as an initial guess for this run of the GA. While this guess is far from ideal (the velocities and pressure are very different and the location of the main eddy is not consistent with this pseudo-plastic fluid,) it does serve the purpose of giving the GA a place to start.

A grid of 41×41 was used for a total of 1681 nodal points to model the flow with Reynolds number 56. A complete list of the genetic parameters and options used is given in Table IV. The evolution took about 32 min, performing 89 GA sweeps for about 1900 corrected generations. The problem was run on the same 2.0 GHz Pentium 4 platform with 512 MB of RAM. Figure 18 shows the progress of the evolution represented by the fitness of the elitist and the magnitude of the maximum residual in the domain over the performed GA sweeps.

The resulting contours of velocity magnitudes and pressure with velocity vectors superimposed on them are shown in Figures 19 and 20, respectively. We see that the GA successfully arrived at a physical, converged solution for a case that SIMPLER could not do with any constant relaxation factor. The GA heuristically determined the fittest chromosome according to the supplied objective function disregarding the gradients that prevented SIMPLER from converging.

The GA was run with the option of *simultaneously* optimizing all three variables at once, which was found to give the fastest convergence. In simultaneous optimization, at each nodal point, all three residuals of the discretized equations are calculated and only the largest is considered for the fitness function. While this method superficially seems to optimize only one variable at each node, the selection of the largest residual emphasizes the worst of them in the fitness value. Since the three equations are heavily coupled, the actual residual might be

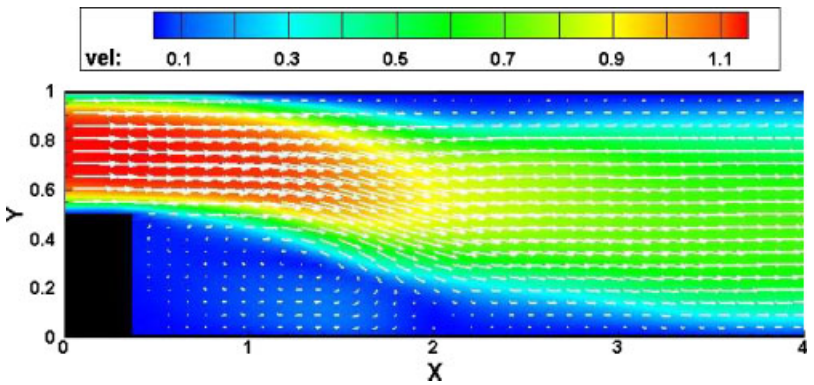

Figure 19. GA velocity magnitude contours with superimposed velocity vectors for flow over a backward facing step of a power law fluid with $n = 0.25$.

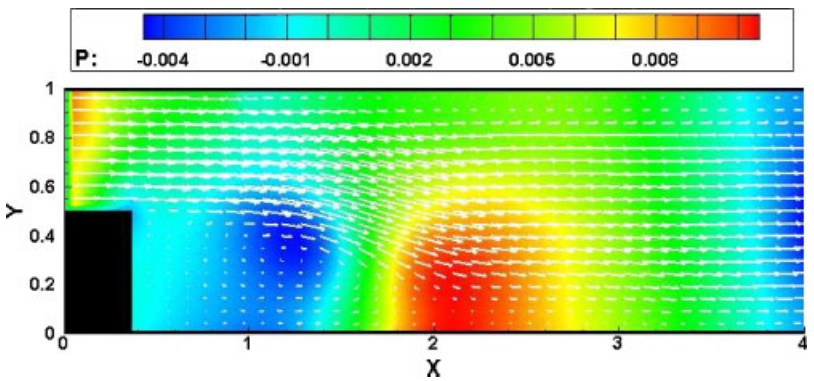

Figure 20. GA pressure contours with superimposed velocity vectors for flow over a backward facing step of a power law fluid with $n = 0.25$.

attributed to any of the variables u, v or p. Moreover, this method cuts down on the overhead computational time required to do other genetic operators such as the selection operator.

On the other hand, optimizing the unknowns *iteratively* means that the fitter chromosomes of the population are judged against the *presumed-constant* values of other unknowns. When the other variables are not optimal, a more fit variable might attain a lower fitness than a less fit one only because it matches the deviation(s) exhibited by the other 'constant' variables. For this reason, the two optimization methods are not quite similar in the GA as in other solution techniques. A simultaneous GA optimization of solution variables is *less simultaneous* and *more implicit*: it implicitly optimizes all variables treating them as one.

To assess the degree of convergence of the GA, we turn again to SIMPLER. But because SIMPLER will not converge for the given problem parameters and power law flow index, we follow the previously outlined strategy of successively reducing relaxation factors and flow indices from a converged Newtonian solution. Treating this solution as the correct solution,

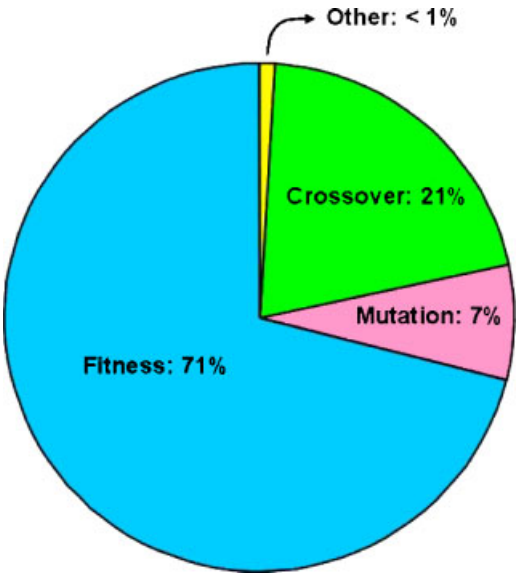

Figure 21. Breakdown of computation time of a GA run for a power law fluid with $n = 0.25$.

we determine that the GA has converged to within 1.6%. This is comparable to the degree of convergence of the Newtonian case (cf. Section 4.4). Furthermore, if we take into account that the non-Newtonian case has double the population size with roughly half the number of nodes, the difference in computational time can be attributed to the poor initial guess for the non-Newtonian case—which was a half-converged *Newtonian* solution. We note here that for this hard to converge problem, SIMPLER will not converge at all without the aforementioned tricks *even when the Newtonian solution is given as the initial guess*.

Finally, we examine the breakdown of computational time of the non-Newtonian case, shown in Figure 21. We notice that the fitness evaluations still take a large chunk of the CPU time. The calculation of the various equation coefficients and power law viscosity requires a large number of loops over nodal points, interpolations of velocities to calculate the viscosity, and calls for various functions and subroutines from within the fitness routine.

6. CONCLUSIONS

The main proposition of this study was not that GAs can completely replace conventional CFD methods as stand-alone solution techniques but rather be used as divergence-combating techniques to be called once divergence or stagnation is detected. The abilities of the GA as a fluid solver was clearly demonstrated with examples from potential, viscous and non-Newtonian fluid problems. Their capability of acting as auxiliary CFD solvers that are used once more traditional methods encounter difficulties was demonstrated with the power law problem where the finite volume method failed to converge. The heuristic nature of the search method permits it to overcome difficulties associated with large gradients in flow fields.

The newly introduced genetic operators greatly enhanced the performance of the GAs. The applicability of the GAs was expanded to include the previously untested area of computational fluid dynamics.

The problems considered in this study represent a small class of problems the method can be applied to. The GA has been applied successfully to many other problems outside the fluid dynamics arena during the course of this research—such as the problem of the combined conduction, convection and radiation heat transfer in addition to heat generation. In fluid dynamics, the problems considered were meant as a vehicle to demonstrate the abilities of GAs and explore the inner working of the process of artificial evolution as they relate to fluid flow problems.

Future research efforts in the area include experimenting with different types of chromosome coding, different objective functions, such as meshless methods, and other non-Newtonian constitutive equations. More importantly, in order for this evolution program to be truly beneficial, users cannot be expected to have the experience to come up with suitable evolution parameters. A fuzzy controller can best gauge the performance of the GA and make necessary changes to the different parameters and probabilities. In conclusion, we believe that these methods have the potential to be of great utility in CFD applications.

REFERENCES

1. Rechenberg I. Cybernetic solution path of an experimental problem. *Technical Report Library Translation No. 1122*, Royal Aircraft Establishment, Farnborough, Hants, U.K., 1965.
2. Holland JH. *Adaptation in Natural and Artificial Systems*. University of Michigan Press: Ann Arbor, 1975.
3. Schwefel HP. Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik. *Ph.D. Thesis*, Technische Universität Berlin, Berlin, Germany, 1965 (Diplomarbeit).
4. Fogel LJ. Toward inductive inference automata. *Proceedings of the International Federation for Information Processing Congress* 1962; 395–399.
5. Fogel LJ, Owens AJ. *Artificial Intelligence Through Simulated Evolution*. Wiley: New York, 1966.
6. Larrañaga P, Kuijpers C, Murga R, Inza I, Dizdarevic S. Genetic algorithms for the travelling salesman problem: a review of representations and operators. *Artificial Intelligence Review* 1999; **13**:129–170.
7. Jones EA, Joines WT. Genetic design of linear antenna arrays. *IEEE Antenna and Propagation Magazine* 2000; **42**(3):92–100.
8. Fabbri G. A genetic algorithm for fin profile optimization. *International Journal of Heat and Mass Transfer* 1997; **40**(9):2165–2172.
9. Sasikumar M, Balaji C. Optimization of convective fin systems: a holistic approach. *Heat and Mass Transfer* 2002; **39**(1):57–68.
10. Karr C, Yakushinb I, Nicolosi K. Solving inverse initial-value, boundary-value problems via genetic algorithm. *Engineering Applications of Artificial Intelligence* 2000; **13**:625–633.
11. Kim HS, Cho SB. Application of interactive genetic algorithm to fashion design. *Engineering Applications of Artificial Intelligence* 2000; **13**:635–644.
12. Haupt SE, Haupt RL. Optimizing complex systems. *Proceedings of the 1998 IEEE Aerospace Conference*. Snowmass, Colorado, 1998; 241–247.
13. Milano M, Koumoutsakos P. A clustering genetic algorithm for cylinder drag optimization. *Journal of Computational Physics* 2002; **175**:79–107.
14. Davalos RV, Rubinsky B. An evolutionary-genetic approach to heat transfer analysis. *Journal of Heat Transfer, Transactions of the ASME* 1996; **118**(3):528–532.
15. Akin S, Demiral B. Genetic algorithm for estimating multiphase flow functions from unsteady-state displacement experiments. *Computers and Geosciences* 1998; **24**(3):251–258.
16. Vuković S, Sopta L. Binary-coded and real-coded genetic algorithm in pipeline flow optimization. *Mathematical Communications* 1999; **4**:35–42.
17. Fan HY, Yam R, Dang C. A preliminary study of the use of genetic algorithms in flowfield numerical analysis. *Proceedings—Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, vol. 215, 2001; 203–212.
18. Pryor RJ, Cline DD. Use of a genetic algorithm to solve fluid flow problems on an ncube/2 multiprocessor computer. *NASA STI/Recon Technical Report N* 1992; **92**:1–42.

19. Bourisli R, Kaminski DA. Solving fluid flow problems using a real-coded genetic algorithm with uniform refinement. In *Advances in Fluid Mechanics*, Brebbia C, Mendes A, Rahman M (eds), vol. V. WIT Press: Southampton, U.K., 2004; 63–74.
20. Bourisli RI. Computationally intelligent CFD: solving potential, viscous and non-Newtonian fluid flow problems using real-coded genetic algorithms. *Ph.D. Thesis*, Rensselaer Polytechnic Institute, 110 8th Street, Troy, New York, May 2005.
21. Whitley D. The genitor algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In *Proceedings of the Third International Conference on Genetic Algorithms*, Schaffer J (ed.). Morgan Kaufmann: Los Altos, CA, 1989; 116–121.
22. Goldberg DE, Deb K. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, Rawlins GJE (ed.). Morgan Kaufmann: San Mateo, CA, 1991.
23. Michalewicz Z. *Genetic Algorithms + Data Structure = Evolution Programs* (3rd edn). Springer: Berlin, 1997.
24. Jong KAD. An analysis of the behavior of a class of genetic adaptive systems. *Ph.D. Thesis*, University of Michigan, Ann Arbor, 1975.
25. Tate DM, Smith AE. Expected allele coverage and the role of mutation in genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, Forrest S (ed.). Morgan Kaufmann: San Mateo, CA, 1993; 31–37.
26. Wright AH. Genetic algorithms for real parameter optimization. In *Foundations of Genetic Algorithms*, Rawlins GJE (ed.). Morgan Kaufmann: San Mateo, CA, 1991.
27. Davis L. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold: New York, 1991.
28. Park K. A coarse-grained parallel genetic algorithm for clustered document allocation in multiprocessor information retrieval systems. *Journal of Electrical Engineering and Information Science* 1999; **6**(4):641–649.
29. Neubauer A. A theoretical analysis of the non-uniform mutation operator for the modified genetic algorithm. *Proceedings of 1997 IEEE International Conference on Evolutionary Computation*. IEEE: Indianapolis, IN, U.S.A., 1997; 93–96.
30. Salomon R. Evolutionary algorithms and gradient search: similarities and differences. *IEEE Transactions on Evolutionary Computation* 1998; **2**(2):45–55.
31. Pan W, Zhun F, Shan F, Yun Z. Study on a novel hybrid adaptive genetic algorithm embedding conjugate gradient algorithm. *Proceedings of the 3rd World Conference on Intelligent Control and Automation*. IEEE: Hefei, P.R. China, 2000; 630–633.
32. Pryor RJ, Cline DD. Use of a genetic algorithm to solve fluid flow problems on an NCUBE/2 multiprocessor computer. *NASA STI/Recon Technical Report N*, vol. 92, April 1992; 1–42.
33. Patankar SV. *Numerical Heat Transfer and Fluid Flow*. Series in Computational Methods in Mechanics and Thermal Sciences. McGraw-Hill: New York, 1980.
34. Williams PT, Baker AJ. Numerical simulations of laminar flow over a 3d backward-facing step. *International Journal for Numerical Methods in Fluids* 1997; **24**:1159–1183.
35. Raithby GD. A critical evaluation of upstream differencing applied to problems involving fluid flow. *Computer Methods in Applied Mechanics and Engineering* 1976; **9**:75–103.
36. Rudolph G. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks* 1996; **5**(1):96–101.
37. Thierens D, Goldberg D. Convergence models of genetic algorithm selection schemes. *Proceedings of the Third Conference on Parallel Problem Solving from Nature*. Springer: Jerusalem, Israel, 1994; 119–129.
38. Beyer HG. *The Theory of Evolution Strategies*. Springer: Heidelberg, Germany, 2001.
39. Rudolph G. Convergence properties of some multi-objective evolutionary algorithms. *Proceedings of the 2000 Congress on Evolutionary Computation*, vol. 2. IEEE: La Jolla, CA, U.S.A., 2000; 1010–1016.
40. Crochet MJ, Davies AR, Walters K. *Numerical Simulation of Non-Newtonian Flow*, vol. 1, Rheology Series. Elsevier: Amsterdam, The Netherlands, 1984.
41. Bird RB, Armstrong RC, Hassager O. Dynamics of polymeric liquids. *Fluid Mechanics*, vol. 1 (2nd edn). Wiley: New York, 1987.

