School of Mechanical Aerospace and Civil Engineering

TPFE MSc CFD-1

Numerical Solution of Equations

T. J. Craft

George Begg Building, C41

Reading:
J. Ferziger, M. Peric, *Computational Methods for Fluid Dynamics*
H.K. Versteeg, W. Malalasekara, *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*
S.V. Patankar, *Numerical Heat Transfer and Fluid Flow*
Notes: http://cfd.mace.manchester.ac.uk/tmcfd
- People - T. Craft - Online Teaching Material

Introduction

- The Navier-Stokes equations are a set of coupled, non-linear, partial differential equations.
- Solving these numerically consists of two steps:
  - Approximation of the differential equations by algebraic ones.
  - Solution of the system of algebraic equations.
- Before considering how to approximate and solve such systems, it is useful to review briefly how we use numerical methods for approximate solutions of single non-linear algebraic equations.
- These are relevant, since the difference equations we obtain from discretizing the Navier-Stokes equations are non-linear.
- Since most solution methods for non-linear equations are iterative, this introduces a number of concepts and generic treatments that will also be met later when dealing with iterative solution methods for large sets of coupled equations.

- The general problem to be considered is that of solving the equation

$$f(x) = 0 \tag{1}$$

where f is some arbitrary (non-linear) function. For some methods we need to reformulate this as

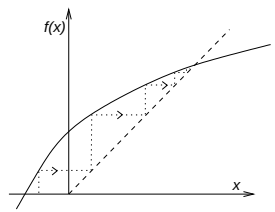
$$x = F(x) \tag{2}$$

- Many methods will probably have been met in earlier courses, but it is useful to review these, to understand their behaviour and examine some of their advantages and weaknesses.

Direct Iteration Method

- This is a fairly simple method, which requires the problem to be written in the form $x = f(x)$ for some function $f(x)$.
- We start with an initial guess to the solution, x_1, and then calculate a new estimate as

$$x_2 = f(x_1) \tag{3}$$



- This process is continued, at each step generating a new approximation

$$x_{n+1} = f(x_n) \tag{4}$$

- Iterations are stopped when the difference between successive estimates becomes less than some prescribed *convergence criterion* ε. ie. when $|x_{n+1} - x_n| < \varepsilon$.
- If the process is convergent, taking a smaller value for ε results in a more accurate solution, although more iterations will be needed.

- As an example, consider solving the equation

$$x^3 - 3x^2 - 3x - 4 = 0 \qquad (5)$$

(which has the exact solution $x = 4$).

- We first need to write the equation in the form $x = f(x)$, and there is more than one way of doing this.
- One way is to write the equation as

$$x = (3x^2 + 3x + 4)^{1/3} \quad \text{so} \quad f(x) = (3x^2 + 3x + 4)^{1/3} \qquad (6)$$

- Starting with an initial guess of $x = 2.5$, the above iterative method gives:

Iteration	1	2	3	4	5	6	7	8
x	2.5	3.12	3.49	3.71	3.83	3.91	3.95	3.97
$f(x)$	3.12	3.49	3.71	3.83	3.91	3.95	3.97	3.98

Iteration	9	10	11
x	3.98	3.99	3.99
$f(x)$	3.99	3.99	4.00

- The scheme converges, although not particularly rapidly.

- If, however, we rewrite equation (5) in the form

$$x = (x^3 - 3x^2 - 4)/3 \quad \text{so} \quad f(x) = (x^3 - 3x^2 - 4)/3 \qquad (7)$$

and again take $x_1 = 2.5$, then the iteration process leads to:

Iteration	1	2	3	4
x	2.5	-2.38	-11.44	-631.19
$f(x)$	-2.38	-11.44	-631.19	-8.4×10^7

- In this case the process is clearly *diverging*.
- Even if we had started from an initial guess much closer to the real solution, with this formulation the iterative process would have diverged.
- As can be seen from this example, the direct method as described is not guaranteed to converge. The particular formulation chosen for $f(x)$ can be highly influential in determining the convergence behaviour.
- To understand why this should be, we note that the difference between successive iterations will be

$$|x_{n+1} - x_n| = |f(x_n) - f(x_{n-1})| \qquad (8)$$

- Assuming that we are close to the true solution, x_r, we can approximate $f(x_n)$ and $f(x_{n-1})$ by Taylor series expansions about x_r:

$$f(x_n) \approx f(x_r) + (x_n - x_r)f'(x_r) + \ldots \qquad (9)$$

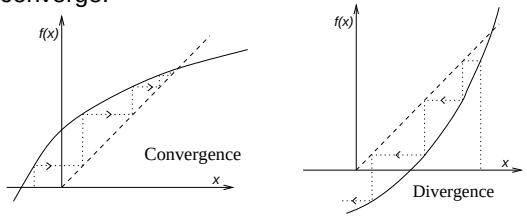
$$f(x_{n-1}) \approx f(x_r) + (x_{n-1} - x_r)f'(x_r) + \ldots \qquad (10)$$

where f' denotes the derivative df/dx.

- Substituting the expressions into equation (8) gives

$$|x_{n+1} - x_n| \approx |f'(x_r)||x_n - x_{n-1}| \qquad (11)$$

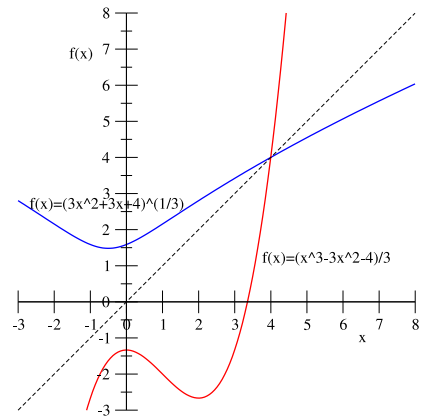
- Thus, if the gradient of f is such that $|f'(x_r)| < 1$, the difference between successive approximations decreases, and the scheme will converge. If $|f'(x_r)| > 1$ the difference between successive approximations increases and the method will not converge.
- This behaviour can also be inferred graphically.



- The graphs of the two particular forms of $f(x)$ examined in the example earlier are shown below.

$x_{n+1} = (3x_n^2 + 3x_n + 4)^{1/3}$ converges

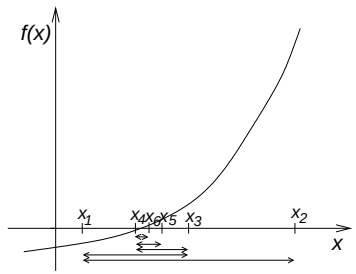
$x_{n+1} = (x_n^3 - 3x_n^2 - 4)/3$ diverges



- This highlights an important aspect of numerical solutions: one needs a good understanding of the problem to be solved and solution methods in order to select the most appropriate scheme.

The Bisection Method

- This is designed to solve a problem formulated as $f(x) = 0$.
- We start off with two points x_1 and x_2, chosen to lie on opposite sides of the solution. Hence $f(x_1)$ and $f(x_2)$ have opposite signs.
- We then bisect this interval, so take $x_3 = 0.5(x_1 + x_2)$, and evaluate $f(x_3)$.
- For the next iteration we retain x_3 and whichever of x_1 or x_2 gave the opposite sign of f to $f(x_3)$. The solution thus lies between the two points we retain.
- We continue bisecting the interval as above until it becomes sufficiently small, ie. $|x_n - x_{n-1}| < \varepsilon$ for some small convergence criteria ε.
- Clearly, as we reduce the convergence criteria ε we get a more accurate approximation to the solution, but have to perform more iterations.



- Solving the same example as earlier we write

$$f(x) = x^3 - 3x^2 - 3x - 4 = 0 \tag{12}$$

- Applying the bisection method iteration, with initial points $x_1 = 2.5$ and $x_2 = 5$ now gives:

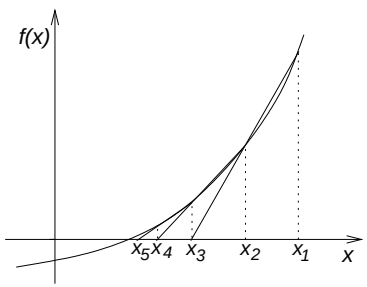
Iteration	1	2	3	4	5	6
(Δx)			(x_1,x_2)	(x_3,x_2)	(x_3,x_4)	(x_3,x_5)
x	2.5	5.0	3.75	4.375	4.0625	3.9062
$f(x)$	-14.63	69.0	-4.703	9.193	1.348	-1.891

Iteration	7	8	9	10	11	
(Δx)	(x_6,x_5)	(x_7,x_5)	(x_7,x_8)	(x_7,x_9)	(x_{10},x_9)	
x	3.9844	4.0235	4.0036	3.994	3.999	
$f(x)$	-0.325	0.499	0.077	-0.1257	0.021	

- The method will always converge, since the interval size always decreases.
- The method can be rather slow, since the interval size is only halved in each iteration.

The Secant Method

- This method solves the system $f(x) = 0$.
- It also requires two starting points, x_1 and x_2, but they need not be on opposite sides of the exact solution.
- We now fit a straight line through the two points $(x_1,f(x_1))$ and $(x_2,f(x_2))$, and take the next estimate as the point at which this line cuts the x axis.



- This iteration method can be formulated mathematically as

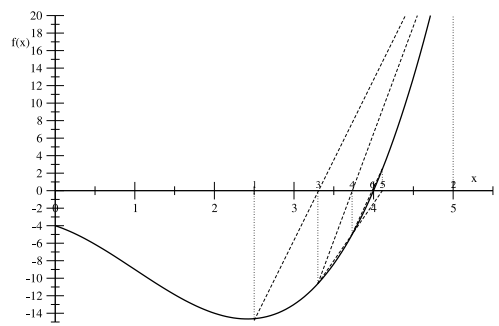
$$x_{n+1} = x_n - f(x_n)\left(\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}\right) \tag{13}$$

- The process is again repeated until the convergence criteria is reached.

- With the earlier example, solving $f(x) = x^3 - 3x^2 - 3x - 4 = 0$, and with the same starting points, we would get:

Iteration	1	2	3	4	5	6	7
x	2.5	5	3.3	3.73	4.11	3.99	4.00
$f(x)$	-14.63	31.0	-10.62	-4.96	2.51	-0.28	-0.01

- The process shows a fairly rapid convergence.

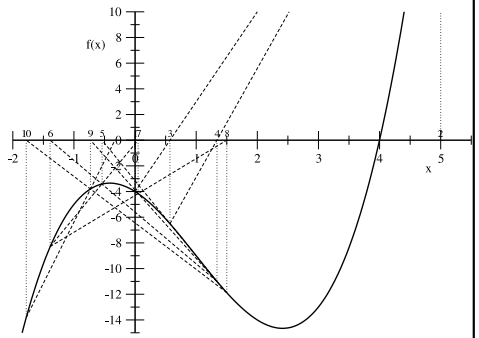


- However, if we instead start with the two points $x_1 = 0$ and $x_2 = 5$, we get the sequence:

Iteration	1	2	3	4	5	6	7	8
x	0.0	5	0.57	1.34	-0.54	-1.39	0.05	1.50
$f(x)$	-4.0	31.0	-6.51	-11.0	-3.41	-8.29	-4.16	-11.87

Iteration	9	10	
x	-0.73	-1.78	
$f(x)$	-3.8	-13.81	

- In this case the process does not appear to be converging.
- By plotting the sequence, we can see that the iterative process oscillates across the local maximum of f around $x = -0.41$.



Under-Relaxation

- Under-relaxation is commonly used in numerical methods to aid in obtaining stable solutions.
- Essentially it slows down the rate of advance of the solution process by linearly interpolating between the current iteration value, x_n and the value that would otherwise be taken at the next iteration level.
- This prevents the scheme from making such large changes in solution estimate from iteration to iteration.
- In the present example of using the Secant method, equation (13) could be modified to read

$$x_{n+1} = x_n - \omega f(x_n) \left(\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \right) \tag{14}$$

where the under-relaxation factor ω is taken between 0 and 1.

- If, for example, we take $\omega = 0.5$, the Secant method applied to the previous example gives:

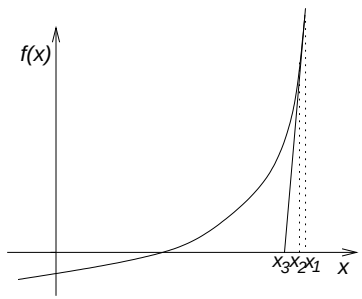
Iteration	1	2	3	4	5	6	7
x	0.0	5.0	2.79	3.13	4.23	4.06	4.03
$f(x)$	-4.0	31.0	-14.02	-12.11	5.20	1.31	0.71

Iteration	8	9	10
x	4.02	4.01	4.00
$f(x)$	0.36	0.18	0.09

- By reducing the jump between successive values of x in the early iterations the solution estimates stay to the right hand side of the local minimum in f, and the process now converges.
- Note that as the solution gets closer to converging, the under-relaxation factor could be increased to speed up convergence.

Convergence Criteria

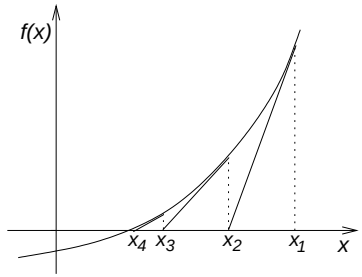
- In the examples presented so far we tested for convergence by checking on the difference between successive solution estimates.
- We thus took the solution as converged when $|x_{n+1} - x_n| < \varepsilon$ for some predefined convergence criteria ε.
- However, this condition should not be applied in a blind fashion.
- For example, if the gradient f' is very steep, the secant method might only give small changes in x_n between iterations, even if still far from the true solution, particularly if using heavy under-relaxation.



- A small value of $|x_{n+1} - x_n|$ could simply indicate very slow convergence.
- A safer way of checking for convergence is to also test whether the actual function value $f(x_n)$ is sufficiently small.

The Newton-Raphson Method

- This also solves the equation $f(x) = 0$.
- In this method we start with an initial guess x_1.
- We draw the tangent to the curve of f at x_1, and take our next estimate x_2 to be the point where this tangent cuts the x axis.



- Mathematically, we can write this iteration process as

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad (15)$$

where f' denotes the derivative df/dx.

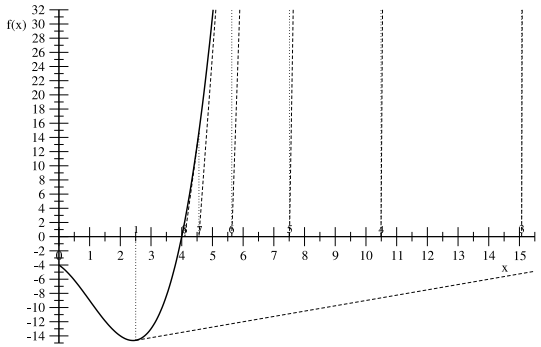
- We again continue the process until we satisfy some suitable convergence criteria.

- Using the same example as before, $x^3 - 3x^2 - 3x - 4 = 0$, with a starting point of $x_1 = 2.5$, we obtain the following sequence:

Iteration	1	2	3	4	5	6	7
x	2.5	22	15.07	10.49	7.51	5.63	4.56
$f(x)$	-14.63	9126	2692.31	789.11	227.27	62.27	14.66
$f'(x)$	0.75	1317	587.95	264.26	120.96	58.21	31.95

Iteration	8	9
x	4.10	4.00
$f(x)$	2.15	0.08
$f'(x)$	22.8	21.07

- Note the large jump in the first step, which occurs since $f'(x_1)$ is small. After this the convergence is quite rapid.

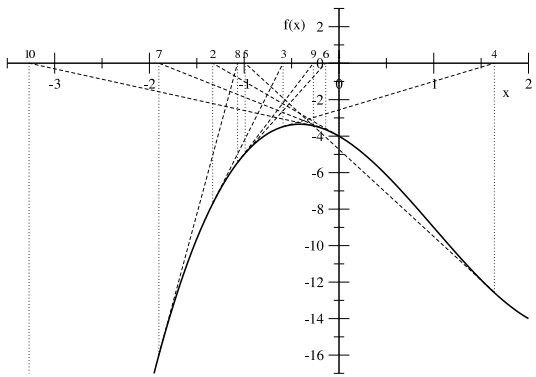


- If, however, we had started the iteration process from an initial value of $x_1 = 0$, we would have got:

Iteration	1	2	3	4	5	6	7	8
x	0.0	-1.33	-0.59	1.64	-0.99	-0.14	-1.90	-1.07
$f(x)$	-4	-7.70	-3.48	-12.56	-4.92	-3.63	-15.98	-5.44
$f'(x)$	-3	10.33	1.56	-4.79	5.84	-2.07	19.22	6.83

Iteration	9	10
x	-0.27	-3.27
$f(x)$	-3.43	-61.25
$f'(x)$	-1.14	48.71

- Because of the shape of f, the method spends a number of iterations oscillating around the local maximum at $x \approx -0.41$.



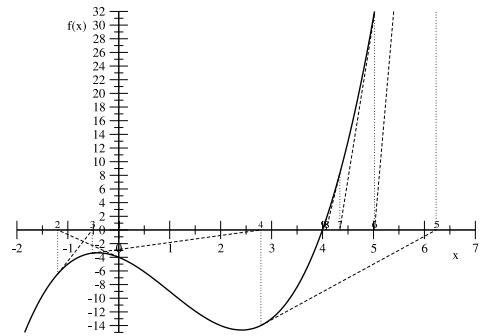
- In this case convergence can again be improved by introducing some under-relaxation. With an under-relaxation factor of 0.9, taking

$$x_{n+1} = x_n - 0.9 f(x_n)/f'(x_n) \qquad (16)$$

we get the iteration sequence:

It.	1	2	3	4	5	6	7	8	9
x	0.0	-1.20	-0.52	2.79	6.23	5.02	4.34	4.07	4.01
$f(x)$	-4	-6.45	-3.39	-13.99	102.69	31.64	8.30	1.55	0.19
$f'(x)$	-3	8.52	0.92	3.66	76.07	42.36	27.53	22.30	21.16

- The solution rapidly reaches a point near the maximum of f where f' is small enough and positive so that the next value of x lies to the right of the minimum at $x \approx 2.4$.
- Convergence could be further speeded up by using less under-relaxation after iteration 4.



Systems of Algebraic Equations

- Most of the methods outlined for solving a single equation can be extended to apply to a system of equations of the form:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ \vdots \qquad\qquad & \vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned}$$

- The functions $f_1 \ldots f_n$ could be linear or non-linear functions of $x_1 \ldots x_n$.
- As in the case of single equations, to solve the system efficiently (and sometimes to solve it at all) requires an understanding of the method adopted, and of the nature of the equations to be solved.

The Direct Iteration Method

- For the equivalent of the direct iteration method, we first have to rewrite the system of equations into the form

$$\begin{aligned} x_1 &= f_1(x_1, x_2, \ldots, x_n) \\ x_2 &= f_2(x_1, x_2, \ldots, x_n) \\ \vdots \quad & \vdots \qquad\qquad \vdots \\ x_n &= f_n(x_1, x_2, \ldots, x_n) \end{aligned}$$

- This can be written more compactly in vector form: $\underline{x} = \underline{f}(\underline{x})$
- One then proceeds, with a starting guess $\underline{x}_0$, constructing the sequence

$$\underline{x}_{n+1} = \underline{f}(\underline{x}_n) \tag{17}$$

stopping the iteration when $||\underline{x}_{n+1} - \underline{x}_n|| < \varepsilon$ for a suitably small convergence criterion ε.

- The norm $||\underline{x}_{n+1} - \underline{x}_n||$ is simply a scalar measure of the "size" of the vector $\underline{x}_{n+1} - \underline{x}_n$. For example, it could be the L_2 norm

$$||\underline{x}||_{L_2} = (\underline{x}.\underline{x})^{1/2} = \left(\sum x_i^2\right)^{1/2} \tag{18}$$

- As before the method is not always guaranteed to converge.
- In addition to convergence behaviour being dependent on the precise formulation used for f_i in the equation

$$x_i = f_i(x_1, x_2, \ldots, x_n) \tag{19}$$

the choice of which equation is used for updating which variable is now also important.

- It is possible to devise expressions for the conditions necessary to ensure convergence, in the same way as was done in the single equation case. However, the result is rather complex and will not be considered here.
- In general, a rule of thumb is to write the equations in a form where the right hand sides contain the variables raised to powers less than one.
- Under-relaxation is often also used, to aid convergence by avoiding large changes in the estimates for $\underline{x}_n$ from iteration to iteration. With this included, the iteration sequence can be written as

$$\underline{x}_{n+1} = (1-\omega)\underline{x}_n + \omega \underline{f}(\underline{x}_n) \tag{20}$$

where ω is the under-relaxation factor, with $0 < \omega < 1$.

The Newton-Raphson Method

- Other schemes can also be adapted to obtain solutions to a system of equations.
- To see how the Newton-Raphson method can be applied to a system of equations, we first notice that its form for a single equation can be devised by retaining the first term in a Taylor series expansion of f about the point x_n.
- If x_r is the actual root, then one can write

$$0 = f(x_r) \approx f(x_n) + (x_r - x_n) f'(x_n) + \ldots \tag{21}$$

- Neglecting the higher order terms and rearranging leads to the approximation

$$x_r \approx x_n - f(x_n)/f'(x_n) \tag{22}$$

- In the case of a system of equations a Taylor series expansion about the root, retaining only first order terms, gives

$$0 = f_1^{(r)} \approx f_1^{(i)} + (x_1^{(r)} - x_1^{(i)})\frac{\partial f_1^{(i)}}{\partial x_1} + (x_2^{(r)} - x_2^{(i)})\frac{\partial f_1^{(i)}}{\partial x_2} + \cdots + (x_n^{(r)} - x_n^{(i)})\frac{\partial f_1^{(i)}}{\partial x_n}$$

$$0 = f_2^{(r)} \approx f_2^{(i)} + (x_1^{(r)} - x_1^{(i)})\frac{\partial f_2^{(i)}}{\partial x_1} + (x_2^{(r)} - x_2^{(i)})\frac{\partial f_2^{(i)}}{\partial x_2} + \cdots + (x_n^{(r)} - x_n^{(i)})\frac{\partial f_2^{(i)}}{\partial x_n}$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$0 = f_n^{(r)} \approx f_n^{(i)} + (x_1^{(r)} - x_1^{(i)})\frac{\partial f_n^{(i)}}{\partial x_1} + (x_2^{(r)} - x_2^{(i)})\frac{\partial f_n^{(i)}}{\partial x_2} + \cdots + (x_n^{(r)} - x_n^{(i)})\frac{\partial f_n^{(i)}}{\partial x_n}$$

where $f_n^{(i)}$ denotes the value $f_n(\underline{x}_i)$ and superscripts r denote the actual root.

- Rearranging, and taking $\underline{x}^{(r)}$ as the estimate to the solution at iteration level $i+1$, leads to a system of linear equations for the elements of the vector $\underline{x}^{(i+1)} - \underline{x}^{(i)}$:

$$\begin{pmatrix} \frac{\partial f_1^{(i)}}{\partial x_1} & \frac{\partial f_1^{(i)}}{\partial x_2} & \cdots & \frac{\partial f_1^{(i)}}{\partial x_n} \\ \frac{\partial f_2^{(i)}}{\partial x_1} & \frac{\partial f_2^{(i)}}{\partial x_2} & \cdots & \frac{\partial f_2^{(i)}}{\partial x_n} \\ \vdots & & & \vdots \\ \frac{\partial f_n^{(i)}}{\partial x_1} & \cdots & \cdots & \frac{\partial f_n^{(i)}}{\partial x_n} \end{pmatrix} \begin{pmatrix} x_1^{(i+1)} - x_1^{(i)} \\ x_2^{(i+1)} - x_2^{(i)} \\ \vdots \\ x_n^{(i+1)} - x_n^{(i)} \end{pmatrix} = - \begin{pmatrix} f_1^{(i)} \\ f_2^{(i)} \\ \vdots \\ f_n^{(i)} \end{pmatrix} \tag{23}$$

which can be solved using a variety of methods, some of which will be discussed later.

- Under-relaxation can also be applied in this method.
- When the scheme converges it generally does so more rapidly than the simpler direct iteration method.

- However, more work has to be done per iteration, since the derivatives $\partial f_i/\partial x_j$ have to be computed, and the matrix system of equation (23) solved.
- In complex problems the gradients $\partial f_i/\partial x_j$ might also have to be computed numerically.
- In practical problems, the Jacobian matrix $[\partial f_i/\partial x_j]$ might not be recomputed every iteration, but perhaps only every few iterations.
- Methods for solving matrix systems such as equation (23) will be examined later in the course.

Summary

- We have examined a number of iterative numerical methods for solving single non-linear equations of the form $f(x) = 0$ and systems of the form $\underline{f}(\underline{x}) = 0$.
- Some schemes, such as the bisection method, will always converge, but may do so rather slowly.
- Others converge more rapidly for some cases, but are not guaranteed to always converge.
- Under-relaxation can be applied in some cases to aid convergence by reducing the change in solution estimates between iterations. The under-relaxation factor can often be increased as the true solution is approached.
- To select and apply the most appropriate method for a particular problem requires a good understanding of the characteristics of the method and of the problem being solved.