TABLE 3.4

Algorithm for Modified Newton's Method

Step 1: Given $\boldsymbol{x}_i$ (starting value of design variable)
ε_1 (tolerance of function value from previous iteration)
ε_2 (tolerance on gradient value)
$\Delta \boldsymbol{x}$ (required for gradient computation)
Step 2: Compute $f(\boldsymbol{x}_i)$, $\nabla f(\boldsymbol{x}_i)$, and $[\boldsymbol{H}]$ (function, gradient, and Hessian)
$\boldsymbol{S}_i = -[\boldsymbol{H}]^{-1}\nabla f(\boldsymbol{x}_i)$ (search direction)
Minimize $f(\boldsymbol{x}_{i+1})$ and determine α (use golden section method)
$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha \boldsymbol{S}_i$ (update the design vector)
If $\left|f(\boldsymbol{x}_{i+1}) - f(\boldsymbol{x}_i)\right| > \varepsilon_1$ or $\|\nabla f(\boldsymbol{x}_i)\| > \varepsilon_2$
then goto Step 2
else goto Step 3
Step 3: Converged. Print $\boldsymbol{x}^* = \boldsymbol{x}_{i+1}$, $f(\boldsymbol{x}^*) = f(\boldsymbol{x}_{i+1})$

3.4.3 Modified Newton's Method

The method is similar to Newton's method with a modification that a unidirectional search is performed in the search direction $\boldsymbol{S}_i$ of the Newton method. The algorithm for the modified Newton method is described in Table 3.4 and a MATLAB code of its implementation is given in *modified_newton.m.*

On executing the code with a starting value of $\boldsymbol{x}$ as (–3, 2), the following output is displayed in the command window for the test problem. For the same starting point, the modified Newton's method converges to the minimum point in just six iterations as compared to Newton's method, which converges in ten iterations.

```
      Initial function value = 1452.2619
No.       x-vector          f(x)          Deriv.
-------------------------------------------------
1     0.006      0.025     -1.010       1006.074
2     0.498      0.026     -8.227         36.392
3     0.496      0.121     -9.653         29.839
4     0.504      0.122     -9.656          0.873
5     0.504      0.122     -9.656          0.018
6     0.504      0.122     -9.656          0.003
-------------------------------------------------
```

3.4.4 Levenberg–Marquardt Method

The advantage of the steepest descent method is that it reaches closer to the minimum of the function in a few iterations even when the starting guess is far away from the optimum. However, the method shows sluggishness near the optimum point. On the contrary, Newton's method shows a faster convergence if the starting guess is close to the minimum point. Newton's method may not converge if the starting point is far away from the optimum point.

The Levenberg–Marquardt method is a kind of hybrid method that combines the strength of both the steepest descent and Newton's methods. The search direction in this method is given by

$$\boldsymbol{S}_i = -[\boldsymbol{H} + \lambda \boldsymbol{I}]^{-1} \nabla f(\boldsymbol{x}_i) \tag{3.18}$$

where $\boldsymbol{I}$ is an identity matrix and λ is a scalar that is set to a high value at the start of the algorithm. The value of λ is altered during every iteration depending on whether the function value is decreasing or not. If the function value decreases in the iteration, λ it decreases by a factor (less weightage on steepest descent direction). On the other hand, if the function value increases in the iteration, λ it increases by a factor (more weightage on steepest descent direction). The algorithm for the Levenberg–Marquardt method is described in Table 3.5 and a MATLAB code of its implementation is given in *levenbergmarquardt.m*.

On executing the code with a starting value of $\boldsymbol{x}$ as (–3, 2), following output is displayed at the command window for the test problem.

```
      Initial function value = 1452.2619
No.       x-vector          f(x)          Deriv.
-------------------------------------------------------
1     -2.384     1.604     815.738      1006.074
2     -1.680     1.139     325.925       733.709
3     -1.104     0.705     102.059       429.113
4     -0.740     0.327      28.673       201.554
5     -0.444     0.133       8.324        86.884
6     -0.164     0.105       1.186        34.005
7      0.546     0.091      -9.390        20.542
8      0.508     0.122      -9.655        11.361
9      0.505     0.122      -9.656         0.409
10     0.504     0.122      -9.656         0.016
-------------------------------------------------------
```

TABLE 3.5

Algorithm for the Levenberg–Marquardt Method

Step 1: Given $\boldsymbol{x}_i$ (starting value of design variable)
- ε_1 (tolerance of function value from previous iteration)
- ε_2 (tolerance on gradient value)
- $\Delta \boldsymbol{x}$ (required for gradient computation)

Step 2: Compute $f(\boldsymbol{x}_i)$, $\nabla f(\boldsymbol{x}_i)$, and $[\boldsymbol{H}]$ (function, gradient, and Hessian)
- $\boldsymbol{S}_i = -[\boldsymbol{H} + \lambda \boldsymbol{I}]^{-1} \nabla f(\boldsymbol{x}_i)$ (search direction)
- $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{S}_i$ (update the design vector)
- If $f(\boldsymbol{x}_{i+1}) < f(\boldsymbol{x}_i)$
  - then change the value of λ as $\lambda/2$
  - else change the value of λ as 2λ
- If $\left| f(\boldsymbol{x}_{i+1}) - f(\boldsymbol{x}_i) \right| > \varepsilon_1$ or $\| \nabla f(\boldsymbol{x}_i) \| > \varepsilon_2$
  - then goto Step 2
  - else goto Step 3

Step 3: Converged. Print $\boldsymbol{x}^* = \boldsymbol{x}_{i+1}$, $f(\boldsymbol{x}^*) = f(\boldsymbol{x}_{i+1})$