In the following function, we construct the residual needed for time integration of second-order system

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{F}_{ext}(t),$$

where we use the residual is defined as

$$\mathbf{r}(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q}) = \mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} - \mathbf{F}_{ext}(t).$$

A generic Residual function, whose *handle* is passed for performing Implicit Newmark and Generalized-α time integrations in this code has the following syntax for linear systems:

```
[r, drdqdd, drdqd, drdq] = Residual(q,qd,qdd,t);
```

where

$$\texttt{r} = \mathbf{r},$$
$$\texttt{drdqdd} = \frac{\partial \mathbf{r}}{\partial \ddot{\mathbf{q}}},$$
$$\texttt{drdqd} = \frac{\partial \mathbf{r}}{\partial \dot{\mathbf{q}}},$$
$$\texttt{drdq} = \frac{\partial \mathbf{r}}{\partial \mathbf{q}}$$

The extra arguments:

1. `Assembly`, which is an instance of Assembly class
2. `Fext`, which is a function handle for the external forcing,

are required for computing the residual in this case.

New residual functions that follow the above-mentioned syntax can be written according to user preference. This way, the same time integration class can be used to solve a variety of problems.

Please refer to the Mechanical directory in the examples folder to understand applications and usage.

```
function [ r, drdqdd, drdqd, drdq] = residual_linear( q, qd, qdd, t, Assembly, Fext)
```

In this function, it is assumed that the matrices $\mathbf{M}, \mathbf{C}, \mathbf{K}$ for the finite element mesh were precomputed and stored in the `DATA` property of the `Assembly` object to avoid unnecessary assembly during each time-step.

```
M = Assembly.DATA.M;
C = Assembly.DATA.C;
K = Assembly.DATA.K;
```

These matrices and the external forcing vector are appropriately constrained according to the boundary conditions:

```
M_red = Assembly.constrain_matrix(M);
C_red = Assembly.constrain_matrix(C);
K_red = Assembly.constrain_matrix(K);
```

```
F_red = Assembly.constrain_vector(Fext(t));
```

Residual is computed according to the formula above:

```
r = M_red * qdd + C_red * qd + K_red * q - F_red ;

drdqdd = M_red;
drdqd = C_red;
drdq = K_red;
end
```