3f. Inversion Point

- A particle is subject to an acceleration function of time,
$$a(t) = \frac{1 - e^{-t}}{\sqrt{1 + t^4}}$$
- Given the particle initial velocity $v(0) = -1$, use Newton's method with a tolerance $\varepsilon = 10^{-6}$ to ***determine when the inversion point is reached***, that is, find t^* such that
$$v(t^*) = v(0) + \int_0^{t^*} a(t)\, dt = 0$$
- Always use Gaussian quadrature with N_g=5 Gaussian points to evaluate the integral and print, each time you compute the velocity:
  i. the lower (`t_lo`) and upper (`t_hi`) bounds as well as the the number of intervals `n_int=ceil(|t_lo - t_hi|*2)` used to compute the integral;
  ii. the cumulative number of function ($a(t)$) evaluation calls (`nfv += Ng*n_int`).

- Of course `t_lo=0.0` initially but, you may modify it to make the algorithm more efficient .

- Upload your code with i) the output inserted in the comments at the beginning of the file, ii) the required library function at the end, e.g.

```
// Name: First Name, Last name
// Date: 14 Nov 2024
//
// Code output:
// ****************************************************
// tlo = ..; thi = ..; nint = ..; nfv = ..
// Inversion time (zero) = ..
// ****************************************************
#include ...
...
int main()
{
  // code here
}

double Velocity(double t){
  static int nfv = 0; // Cumulative number of function evaluations
  ...
  GaussianQuad(..);

  nfv += Ng*nint;
  cout << "tlo = "    << tlo  << "; thi = " << t << "; nint = " << nint
       << "; Func eval = " << nfv << endl;
  ...
}
double Acceleration(double t){}
...
void Gauss(..){}
void Newton(..){}
```