Using Bilinear Transform in Octave/MATLAB

Laplace domain transfer function ($say\ H(s)$) is applicable for continuous time-domain signals. However, in digital signal processing, both input and output are discrete time-domain signals. The Bilinear transform is a simple way for converting *H(s)* into *H(z),* which is widely used in Infinite Impulse Response (IIR) filter design. I'm not going into the deep details of mathematics as it's widely available online. I'll discuss how to use Octave / Matlab to convert *H(s)* into *H(z)* through bilinear transform to save time while designing a filter on AVR, ATSAM, ARM microcontrollers.

Say the transfer function of an analogue filter is defined as:

$$H(s) = \frac{2}{s^2 + 4s + 3}$$

Say the signal is sampled at an interval of 0.1 seconds. To apply bilinear transform on the transfer function, you need to use the bilinear() function of the Octave.

```
[ZB, ZA] = bilinear (SB, SA, T)
```

Here, 'SB' and 'SA' represent the coefficients of the numerator and denominator of *H(s),* respectively, and T stands for the sampling period. 'ZB' and 'ZA' represent the coefficients of the numerator and denominator of *H(z),* respectively. Type "help bilinear" (without the quotation marks) in the command window for more information.

Octave code:

```
%H(s) = 2 / (s^2 + 4s + 3)
pkg load signal;
T = 0.1;
[zn,zd]  = bilinear([2],[1 4 3], T);
[zn;zd]
```

Please write this code on the editor of Octave (or Matlab) and run it.

ans =

4.1408e-03	**8.2816e-03**	**4.1408e-03**
1.0000e+00	**-1.6439e+00**	**6.6874e-01**

From the matrix **[zn;zd],** *H(z)* can be written as:

$$H(z) = \frac{0.0041408 + 0.0082816z^{-1} + 0.0041408z^{-2}}{1 - 1.6439z^{-1} + 0.066874z^{-2}}$$

Simplifying further:

$$H(z) = \frac{0.0041408(1 + z^{-1})^2}{1 - 1.6439z^{-1} + 0.066874z^{-2}}$$