Ruby master - Bug #17032

BigDecimal's `to_d` behaves inconsistent compared to `to_f`

07/14/2020 07:56 PM - TiloS (Tilo S)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.5.5	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN

Description

I would expect to_f and to_d to behave identically. Specifically, nil.to_d should behave like nil.to_f.

```
require 'bigdecimal'
require 'bigdecimal/util'
nil.to_f # => 0.0
nil.to_d # >> NoMethodError (undefined method `to_d' for nil:NilClass)
```

Users should not have to resort to this:

```
nil.to_f.to_d #  => 0.0
```

History

#1 - 07/15/2020 02:31 AM - sawa (Tsuyoshi Sawada)

- Description updated

- Subject changed from BigDecimal .to_d is behaving inconsistent compared to .to_f to `BigDecimal#to_d` behaves inconsistent compared to `#to_f`

#2 - 07/15/2020 02:35 AM - sawa (Tsuyoshi Sawada)

- Subject changed from `BigDecimal#to_d` behaves inconsistent compared to `#to_f` to BigDecimal's `to_d` behaves inconsistent compared to `to_f`

#3 - 07/17/2020 09:44 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Closed

NilClass#to_d was added in Ruby 2.6. As it is not security-related (and seems more like a new feature than a bug fix), it will not be backported to Ruby 2.5, as that is in security maintenance mode.