Ruby trunk - Feature #13208

Vector.zero(n) and vector.zero?

02/11/2017 07:54 PM - qitar888 (Chia-sheng Chen)

Status:	Closed
Priority:	Normal
Assignee:	marcandre (Marc-Andre Lafortune)
Target version:	

Description

Found that I need this recently, and class Matrix has these two function while Vector not.
So I add two function based on Matrix counterpart and also add test.

Usage

```
require 'matrix'
v = Vector.zero(3)  # => Vector[0, 0, 0]
v.zero?  # => true
w = Vector[1, 0, 0]
w.zero?  # => false
```

Associated revisions

Revision 666df145 - 03/14/2017 08:09 PM - marcandre (Marc-Andre Lafortune)

- lib/matrix.rb: Add Vector.zero and Vector#zero? Patch by Chia-sheng Chen [#13208]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@57976 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 57976 - 03/14/2017 08:09 PM - marcandre (Marc-Andre Lafortune)

- lib/matrix.rb: Add Vector.zero and Vector#zero? Patch by Chia-sheng Chen [#13208]

Revision 57976 - 03/14/2017 08:09 PM - marcandre (Marc-Andre Lafortune)

- lib/matrix.rb: Add Vector.zero and Vector#zero? Patch by Chia-sheng Chen [#13208]

Revision 57976 - 03/14/2017 08:09 PM - marcandre (Marc-Andre Lafortune)

- lib/matrix.rb: Add Vector.zero and Vector#zero? Patch by Chia-sheng Chen [#13208]

History

#1 - 02/12/2017 08:15 PM - shevegen (Robert A. Heiler)

I think the method is fine but of course that is just my own opinion,
the core team and matz will decide that.

One hopefully small request though:

- Do you think you could make the description for the method a bit more explicit? Right now the documentation to it states only:

"Return a zero vector."

Perhaps this is obvious to others but I am not entirely sure
what a zero vector per se is, although of course the example is
explicit enough to infer from it (I assume a zero vector is one that
contains only 0 as its members but from the documentation alone,
without the example, I am not sure everyone may be able to infer
this completely correctly so.)

#2 - 02/12/2017 10:58 PM - marcandre (Marc-Andre Lafortune)

- Assignee set to marcandre (Marc-Andre Lafortune)

Looks good.

I think the short description is probably good enough. The example makes it clear if someone has doubts...

I'm travelling, will commit in a few days.

#3 - 03/14/2017 08:11 PM - marcandre (Marc-Andre Lafortune)

Thanks for the patch.
Committed.
PS: I allowed Vector.zero(0).

#4 - 03/14/2017 08:11 PM - marcandre (Marc-Andre Lafortune)

- Status changed from Open to Closed

#5 - 03/14/2017 08:21 PM - stomar (Marcus Stollsteimer)

Just curious:

```
Vector.zero(0).zero?  # => true or false?
```

Not sure what I would expect in this case.

#6 - 03/15/2017 03:07 AM - duerst (Martin Dürst)

stomar (Marcus Stollsteimer) wrote:

> Just curious:
>
> ```
> Vector.zero(0).zero?  # => true or false?
> ```
>
> Not sure what I would expect in this case.

Definitely true. More generally,

```
Vector.zero(x).zero?  # => true
```

for any value of x.

Colloquially, all the values in the vector are zero, even if there aren't that many values there.

That also matches how Mathematics works.

It also works best that way in programming. The simplest implementation of zero? is something like

```
class Vector
  def zero?
    all { |e| e.zero? }
  end
end
```

which will automatically give the right result.

#7 - 03/15/2017 09:42 PM - stomar (Marcus Stollsteimer)

Granted, in the case of #zero? returning true might suggest itself, but is "nothing" really the same as "zero"...? If there was a method Vector.one with Vector.one(3) => Vector[1, 1, 1], should Vector.one(0).one? be true? (All the values in the vector are 1, even if there aren't that many values there...). In that case, for Vector[] both #one? and #zero? would be true at the same time...

Generally speaking: I'm a bit sceptical about allowing zero size for these creator methods and about allowing these properties for zero size vectors/matrices, and I'm not sure what mathematics says about it.

It's easier to demonstrate the possible problems using the Matrix class (which has more methods to play around). The current behavior is a bit strange and contradictory:

```
size = 0

z = Matrix.zero(size)
z.zero? && z.orthogonal? && z.unitary?  # => true  (a zero matrix that's also orthogonal!)
z.det    # => 1  (should be 0 for *any* `size')

i = Matrix.identity(size)
i.zero?  # => true  (should be false for *any* `size')
```

```
# and of course:
i == z  # true  (!)
```

The question is whether the methods that generate those special matrices/vectors should allow size 0 (since the generated objects do not have all the expected properties), and/or whether the methods that query these properties should return a value or not (since the property might not be very meaningful).

#8 - 03/16/2017 06:46 PM - marcandre (Marc-Andre Lafortune)

Martin is exactly right. See https://en.wikipedia.org/wiki/Vacuous_truth

Files

0001-Add-Vector.zero-n-and-vector.zero.patch	2.06 KB	02/11/2017	qitar888 (Chia-sheng Chen)