Package ‘umbridge’

October 12, 2022

Title  Integration for the UM-Bridge Protocol
Version 1.0
Maintainer Linus Seelinger <mail@linusseelinger.de>
Description A convenient wrapper for the UM-Bridge protocol. UM-Bridge is a protocol designed for coupling uncertainty quantification (or statistical / optimization) software to numerical models. A model is represented as a mathematical function with optional support for derivatives via Jacobian actions etc.
License MIT + file LICENSE
Encoding UTF-8
RoxygenNote 7.2.1
BugReports https://github.com/um-bridge
Imports httr2, jsonlite, magrittr
Suggests testthat (>= 3.0.0)
Config/testthat/edition 3
NeedsCompilation no
Author Linus Seelinger [aut, cre] (<https://orcid.org/0000-0001-8632-8493>)
Repository CRAN
Date/Publication 2022-09-23 07:30:02 UTC

R topics documented:

<table>
<thead>
<tr>
<th>apply_hessian</th>
<th>apply_jacobian</th>
<th>evaluate</th>
<th>get_models</th>
<th>gradient</th>
<th>model_input_sizes</th>
<th>model_output_sizes</th>
<th>protocol_version_supported</th>
<th>supports_apply_hessian</th>
<th>supports_apply_jacobian</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**apply_hessian**

Evaluate Hessian of model.

**Usage**

```r
apply_hessian(
  url,  
  name,  
  out_wrt,  
  in_wrt1,  
  in_wrt2,  
  parameters,  
  sens,  
  vec,  
  config = jsonlite::fromJSON("{}")
)
```

**Arguments**

- **url** URL the model is running at.
- **name** Name of the desired model.
- **out_wrt** Output variable to take Hessian with respect to.
- **in_wrt1** First input variable to take Hessian with respect to.
- **in_wrt2** Second input variable to take Hessian with respect to.
- **parameters** Model input parameter (a list of vectors).
- **sens** Sensitivity with respect to output.
- **vec** Vector to multiply Hessian by.
- **config** Model-specific configuration options.

**Value**

Hessian with respect to given inputs and outputs, applied to given sensitivity and vector.
apply_jacobian

**Evaluate Jacobian of model.**

**Description**
Evaluate Jacobian of model.

**Usage**

```r
apply_jacobian(
  url, 
  name, 
  out_wrt, 
  in_wrt, 
  parameters, 
  vec, 
  config = jsonlite::fromJSON("{}")
)
```

**Arguments**

- **url** URL the model is running at.
- **name** Name of the desired model.
- **out_wrt** Output variable to take Jacobian with respect to.
- **in_wrt** Input variable to take Jacobian with respect to.
- **parameters** Model input parameter (a list of vectors).
- **vec** Vector to multiply Jacobian by.
- **config** Model-specific configuration options.

**Value**
Jacobian with respect to given input and output variables, applied to given vector.

---

evaluate

**Evaluate model.**

**Description**
Evaluate model.

**Usage**

```r
evaluate(url, name, parameters, config = jsonlite::fromJSON(""{}""))
```
Arguments

url  URL the model is running at.
name  Name of the desired model.
parameters  Model input parameter (a list of vectors).
config  Model-specific configuration options.

Value

The model output (a list of vectors).

---

**get_models**  Get models supported by server.

Description

Get models supported by server.

Usage

gerget_models(url)

Arguments

url  URL the model is running at.

Value

List of models supported by server.

---

**gradient**  Evaluate gradient of target functional depending on model.

Description

Evaluate gradient of target functional depending on model.

Usage

gergradient(
    url,
    name,
    out_wrt,
    in_wrt,
    parameters,
    sens,
    config = jsonlite::fromJSON("{}")
)
**model_input_sizes**

**Arguments**

- **url** URL the model is running at.
- **name** Name of the desired model.
- **out_wrt** Output variable to take gradient with respect to.
- **in_wrt** Input variable to take gradient with respect to.
- **parameters** Model input parameter (a list of vectors).
- **sens** Sensitivity of target functional with respect to model output.
- **config** Model-specific configuration options.

**Value**

Gradient of target functional.

---

```r
model_input_sizes(url, name, config = jsonlite::fromJSON("{}"))
```

**Description**

Retrieve model’s input dimensions.

**Usage**

Retrieve model’s input dimensions.

**Arguments**

- **url** URL the model is running at.
- **name** Name of the desired model.
- **config** Model-specific configuration options.

**Value**

List of input dimensions.
model_output_sizes

Retrieve model's output dimensions.

Description

Retrieve model's output dimensions.

Usage

model_output_sizes(url, name, config = jsonlite::fromJSON("{}"))

Arguments

url
Name of the desired model
config
Model-specific configuration options.

Value

List of output dimensions.

protocol_version_supported

Check if model’s protocol version is supported by this client.

Description

Check if model’s protocol version is supported by this client.

Usage

protocol_version_supported(url)

Arguments

url
URL the model is running at.

Value

TRUE if model’s protocol version is supported by this client, FALSE otherwise.
**supports_apply_hessian**

*Check if model supports Hessian action.*

**Description**

Check if model supports Hessian action.

**Usage**

```
supports_apply_hessian(url, name)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>URL the model is running at.</td>
</tr>
<tr>
<td>name</td>
<td>Name of the desired model.</td>
</tr>
</tbody>
</table>

**Value**

TRUE if model supports Hessian action, FALSE otherwise.

---

**supports_apply_jacobian**

*Check if model supports Jacobian action.*

**Description**

Check if model supports Jacobian action.

**Usage**

```
supports_apply_jacobian(url, name)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>URL the model is running at.</td>
</tr>
<tr>
<td>name</td>
<td>Name of the desired model.</td>
</tr>
</tbody>
</table>

**Value**

TRUE if model supports Jacobian action, FALSE otherwise.
supports_evaluate  
*Check if model supports evaluation.*

**Description**  
Check if model supports evaluation.

**Usage**  
supports_evaluate(url, name)

**Arguments**
- **url**  
  URL the model is running at.
- **name**  
  Name of the desired model.

**Value**  
TRUE if model supports evaluation, FALSE otherwise.

---

supports_gradient  
*Check if model supports gradient evaluation.*

**Description**  
Check if model supports gradient evaluation.

**Usage**  
supports_gradient(url, name)

**Arguments**
- **url**  
  URL the model is running at.
- **name**  
  Name of the desired model.

**Value**  
TRUE if model supports gradient evaluation, FALSE otherwise.
Index

apply_hessian, 2
apply_jacobian, 3

evaluate, 3

generate, 4
gradient, 4

model_input_sizes, 5
model_output_sizes, 6

protocol_version_supported, 6

supports_apply_hessian, 7
supports_apply_jacobian, 7
supports_evaluate, 8
supports_gradient, 8