TP – Introduction à Julia

1 Introduction

1.1 Qu'est-ce que Julia

De `julialang.org`

Julia is a high-level, high-performance dynamic programming language for numerical computing. It provides a sophisticated compiler, distributed parallel execution, numerical accuracy, and an extensive mathematical function library. Some of the main features are :

- MIT licensed : free and open source
- "just in time" (JIT) compiler
- multiple dispatch
- call C, Fortran or Python functions easily (no MEX files!!!)
- designed for parallelism and distributed computation

1.2 Installation

Pour installer Julia sur Windows, MacOS ou Linux, le plus simple est de se rendre dans `julialang.org` dans la section des téléchargements.

1.3 Quelques informations, remarques

Après l'installation, le programme Julia est ajouté au menu des programmes et peut être lancé. À défaut, la saisie de Julia dans un terminal doit provoquer l'apparition de

```
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.4.1 (2020-04-14)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia>
```

L'invite de commande est appelé dans la littérature anglaise JuliaREPL (read-eval-print loop) . La documentation complète est à :
`https://docs.julialang.org/en/v1/stdlib/REPL/index.html`

— Pour quitter Julia il faut taper `exit()`.

— Ce qui apparaît après le caractère # est un commentaire.

```
julia> # Ceci est un comentaire

julia> A = 1  # Un autre commentaire
1

julia>
```

```
julia> a = exp(1.)^5
148.41315910257657

julia> log(ans)    # ans est la variable answer
5.0

julia> M = [1 1 ; 1 0 ; 0 1]  # matrice d'entiers
3x2 Array{Int64,2}:
 1  1
 1  0
 0  1

julia> [M M ; M zero(M)] # definition d'une matrice par bloc
6x4 Array{Int64,2}:
 1  1  1  1
 1  0  1  0
 0  1  0  1
 1  1  0  0
 1  0  0  0
 0  1  0  0

julia> f(x) = x^2+1  # definition d'une fonction simple
f (generic function with 1 method)

julia> f(3)
10

julia>
```

1.4 Aide, commande shell

Pour accéder à l'aide en ligne il faut taper la caractère ? puis le mot pour lequel on veut l'aide

```
help?> for
search: for foreach foldr floor mapfoldr factorial EOFError OverflowError Ou
```

```
  for

  for loops repeatedly evaluate a block of statements while iterating over a

  Examples
  --------

  julia> for i in [1, 4, 0]
             println(i)
         end
  1
  4
  0

julia>
```

Qu'en à l'accès au shell, il se fait en tapant le caractère ;

```
shell> pwd
/Users/gergaud/ENS/INSA-IA/ode/TP/tp-intro-julia
```

1.5 Fichiers de programmes

Les fichiers de programmes en Julia sont des fichiers textes avec l'extension `.jl`. Si on a dans le répertoire courant par exemple le fichier `prog1.jl` qui contient

```
function helloworld()
    println("Hello, World!") # Bye bye, MATLAB!
end

helloworld()
```

Alors on peut exécuter ce fichier :

— depuis le terminal en tapant

```
(base) brigitte01:tp-intro-julia gergaud$ julia prog1.jl
Hello, World!
```

— À partir de Julia REPL

```
(base) brigitte01:tp-intro-julia gergaud$ julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _    |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |   |
  | | |_| | | | (_| |   |  Version 1.4.1 (2020-04-14)
```

```
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia> include("prog1.jl")
Hello, World!
```

— Si on ajoute à la première ligne du fichier `prog1.jl`

```
#!usr/local/bin/julia
```

et qu'ensuite on rende ce fichier exécutable via la commende système

```
chmod +x prog1.jl
```

alors si dans le terminal on tape `./prog1.jl` le programme est exécuté. On peut également dans ce cas passer des paramètres au programme. Si par exemple dans le fichier `cmd.jl` on a

```
println(ARGS)
```

alors en tapant dans le fenêtre de commande `julia cmd.jl param1 param2` on obtient

```
["param1", "param2"]
(base) brigitte01:tp-intro-julia gergaud$
```

Ceci fait de Julia un langage de script.

1.6 Paquets

[1] Comme nombre de langages modernes, Julia possède de très nombreuses bibliothèques de programmes. Parmi ces bibliothèques, certaines sont écrites en pur langage Julia, tandis que d'autres réalisent une interface entre Julia et une bibliothèque binaire ou écrite dans un autre langage comme `Python`. Les bibliothèques de programmes sont appelées des paquets (packages en anglais). L'installation de ces paquets se fait à l'aide du gestionnaire de paquets appelé `Pkg`. La manière la plus simple pour accéder à ce gestionnaire est sous Julia REPL de taper le caractère `]`. Pour quitter ce mode, il faut utiliser la touche retour arrière.

Une fois dans ce gestionnaire de paquets, si on veut installer par exemple le paquet `LinearAlgebra` il faut taper

```
(@v1.4) pkg> add LinearAlgebra
  Resolving package versions...
   Updating `~/.julia/environments/v1.4/Project.toml`
  [37e2e46d] + LinearAlgebra
   Updating `~/.julia/environments/v1.4/Manifest.toml`
 [no changes]
```

Une fois la bibliothèque installée et importer on peut utiliser les fonctions de celle-ci.

1. Package en anglais

```
julia> using LinearAlgebra

julia> M=[1 2 ; 3 4]
2x2 Array{Int64,2}:
 1  2
 3  4

julia> det(M)
-2.0
```

Voici les principales commandes de gestion de package

- `add Nom_du_Paquet` installe le paquet (et ses dépendances) ;
- `rm Nom_du_Paquet` supprime le paquet ;
- `update Nom_du_Paquet` met à jour le paquet s'il y a une version plus récente ;
- `status` liste les paquets installés ;
- `update` met à jour tous les package installés
- `gc` (garbage collector) fait le ménage en supprimant les paquests inutiles.

Remarque 1. On peut aussi gérer les paquets directement dans Julia REPL. Pour ajouter par exemple le paquet `Plots`, il faut taper

```
julia> using Pkg

julia> Pkg.add("Plots")
  Resolving package versions...
   Updating `~/.julia/environments/v1.4/Project.toml`
 [no changes]
   Updating `~/.julia/environments/v1.4/Manifest.toml`
 [no changes]
```

On trouvera la liste des packages à l'adresse `https://julialang.org/packages/`. Pour un paquet, on accède à la documentation en ligne. On peut aussi accéder au dépot GitHub (et donc aux sources).

Remarque 2. Pour les TP il faut installer les paquets suivants :

- `IJulia` pour pouvoir utiliser les notebook ;
- `Plots` pour les graphiques ;
- `LinearAlgebra` ;

1.7 Environnement

Un bon environnement de travail en Julia est l'IDE juno d'atom.

1.8 Quelques pointeurs

Comme pour `Python`, la documentation en ligne est très (trop ?) abondante. Voici quelques pointeurs pour apprendre ce langage :

- bien sur le site officiel de Julia `https://julialang.org` ;
- un résumé des principales commandes `https://juliadocs.github.io/Julia-Cheat-Sheet/` ;
- le site de l'ENSTA dédié à Julia `https://perso.ensta-paris.fr/~diam/julia/`

2 Introduction aux notebook

Jupyter est une application Web qui regroupe intimement deux fonctionnalités très différentes. Premièrement un outil qui permet de créer des documents multimédia intégrant du texte, des formules mathématiques, des graphiques, des images, voire des animations et des vidéos. Ensuite une interface qui permet d'exécuter du code informatique. Pour cela Jupyter s'appuie sur des programmes indépendants capables d'interpréter le langage dans lequel est écrit ce code. Dans la terminologie de Jupyter ces interpréteurs sont appelés des noyaux (kernel en anglais). Les documents Jupyter sont appelés des notebooks. Un fichier notebook est reconnaissable par son extension `.ipynb`. Avant de lancer le notebook il faut sous Julia installer le package `IJulia` (voir la section précédante). Pour lancer les notebook il suffit de taper dans la fenêtre de commande (à partir du répertoire qui contient les fichiers d'extension `.ipynb`) `jupyter notebook`.

3 Manipulation du notebook

À l'usage, il se révèle plus efficace d'utiliser les raccourcis clavier que de naviguer dans les menus avec la souris. Prenez-en l'habitude. Le notebook est constitué d'une succession de cellules. Les cellules peuvent être dans le mode commande ou le mode édition. Un liseré de couleur repère la cellule actuellement sélectionnée. Le liseré est bleu si la cellule est en mode commande et vert si elle est en mode édition. Le mode édition Pour entrer dans le mode édition de la cellule sélectionnée, il suffit de presser la touche [Enter] ou de cliquer à l'intérieur de la cellule. Quand une cellule est en édition vous pouvez saisir du texte comme dans un éditeur classique. Lorsque le curseur est en début de ligne ou lorsque vous avez sélectionné du texte, l'appui sur la touche [Tab] (respectivement [Shift-TAB]) indente (respectivement désindente) les lignes correspondantes. Voici d'autres raccourcis clavier :

- Ctrl-, : Commente ou décommente une ligne. Si du texte a été sélectionné, toutes les lignes correspondantes sont commentées ou décommentées.
- Ctrl-A : Sélectionne tout le texte de la cellule.
- Ctrl-Z : Annule les dernières saisies de texte.
- Ctrl-Enter : Exécute la cellule.
- Shift-Enter : Exécute la cellule et sélectionne la cellule suivante. L'appui répété de cette touche permet ainsi d'exécuter pas à pas toutes les cellules du notebook.
- Alt-Enter : Exécute la cellule et insére une nouvelle cellule juste en dessous.
- ESC : Passe dans le mode commande.

4 Le mode commande

Quand vous êtes dans le mode commande, vous pouvez ajouter ou supprimer des cellules mais vous ne pouvez pas saisir de texte dans une cellule. Notez bien que dans ce mode, certaines touches du clavier sont associées à des actions sur le notebook. Tenter de saisir du texte dans ce mode peut donc produire des effets inattendus. Voici les raccourcis principaux disponibles en mode commande :

- A et B : Insèrent une nouvelle cellule, respectivement au-dessus ou au-dessous de la cellule sélectionnée.
- M : Transforme la cellule en une cellule de type Markdown (Cf. ci-dessous).
- Y : Transforme la cellule en une cellule de type Code.
- C et V : Copie et colle des cellules.
- DD : Supprime la cellule sélectionnée.
- Z : Annule la dernière suppression de cellule.
- II : Interrompt l'exécution du code.
- 00 : Redémarre l'interpréteur. Il se retrouve alors dans son état initial.
- H : Affiche la liste de tous les raccourcis clavier.

5 Le langage Markdown

La touche M transforme la cellule sélectionnée en type Markdown. Vous pouvez alors rédiger du texte enrichi (titres, sous-titres, gras, italique, alinéas, tableaux, liens hypertexte, etc). Le texte de la cellule doit être rédigé en langage Markdown qui est un langage de balisage léger. La syntaxe markdown est facile à apprendre[1]. Le plus simple est d'ailleurs de regarder des exemples de documents tels que celui que vous lisez actuellement. Il est également possible d'insérer des morceaux de texte en langage LaTeX pour composer des expressions mathématiques.

6 Git

Git *est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2. En 2016, il s'agit du logiciel de gestion de versions le plus populaire qui est utilisé par plus de douze millions de personnes*[2].

Pour apprendre à utiliser Git et GitHub, vous pouvez par exemple voir [3]

Tous les packages de Julia sont sous GitHub, voici ci-après le dépot GitHub du package `https://github.com/SciML/DifferentialEquations.jl`

Références

[1] `https://daringfireball.net/projects/markdown/syntax`. 7

[2] https://fr.wikipedia.org/wiki/Git. 7

[3] https://openclassrooms.com/fr/courses/2342361-gerer-son-code-avec-git-et-github. 7