METAPOD - Template META-PrOgramming applied to Dynamics: CoP-CoM trajectories filtering.

M. Naveau[1], J. Carpentier[1], S. Barthelemy[2], O. Stasse[1], P. Soueres[1]

***Abstract*— In this contribution, Metapod, a novel C++ library computing efficiently dynamic algorithms is presented. It uses template-programming techniques together with code-generation. The achieved performances shows a clear advantage over the state-of-the art dynamic library RBDL. The advantage of this library is that it is open-source and does not rely on any external symbolic computational software. Additionnaly we show how it can help in current control problems for humanoid robots, and more specifically for dynamic filtering of walking gait trajectories.**

I. INTRODUCTION

Computing dynamic state estimate for a complex redundant robot such as humanoid robot is a crucial technique for controlling it. For instance, Inverse Dynamics control heavily relies on the computation of the inertia matrix [1], [2]. When considering balance, this matrix is of primary interest. Indeed most of the real-time approaches relies on the Linear Inverted Pendulum Model to generate a Center-Of-Mass reference trajectory which complies with the balance criteria provided by the Center-Of-Pressure and the contacts on the ground. Inverse Dynamics is necessary to counteract the inertial effect involved by the limbs: either by modifying the center-of-mass (CoM) reference [3], either by following the momentum reference trajectories. More advanced approaches are using Model Predictive Control (MPC) [4], [5] and need to compute the whole robot dynamic over a time horizon.

Since 2006, processor speed is limited to 3$\sim$4 GHz, because transistors have reached their physical limits mostly due to heat dissipation [6]. The Moore Law is still valid because as the chipsets integrate more cores, the transistor number continues to grow not because of their size, but because of their density. They are overheating when CPU clock is increased, and the focus is now more on concurrent software programming to take advantage of multiple cores. In the case of MPC, it mainly consists in having multiple instances starting with different initial solutions. Boosting the code running concurrently will also increase the performances. It is now necessary to implement code to compute dynamics efficiently while taking into account the underlying computing architecture.

*This work was supported by the European project KOROIBOT FP7-ICT-2013-10/611909, the French National project OSEO ROMEO-2 and by AIRBUS/Future of the Aircraft Factory.

[1]are with the CNRS, LAAS,7 av. du Colonel Roche, F-31400, Toulouse, France, Univ de Toulouse, LAAS, F-31400, Toulouse, France `mnaveau,jcarpent,ostasse,soueres@laas.fr`

[2] Sébastien Barthelemy is with Aldebaran Robotics `sbarthelemy@aldebaran.com`

A. Problem formulation

The rigid multibody dynamic model of the robot is assumed to be known and called $model$. According to [7], the dynamical equations can be written under the canonical form:

$$\mathbf{H}(model, \mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(model, \mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau} \tag{1}$$

where $\mathbf{q}$, $\dot{\mathbf{q}}$ et $\ddot{\mathbf{q}}$ are respectively the generalized vectors for positions, velocities and accelerations. $\mathbf{H}$ is the generalized inertia matrix, $\mathbf{C}$ is the matrix representing the centrifugal, gravitational and Coriolis forces.

We should then compute as fast as we can:

- $\boldsymbol{\tau}$ by using the Recursive Newton-Euler Algorithm (RNEA) with the following inputs $\mathbf{q}$, $\dot{\mathbf{q}}$ et $\ddot{\mathbf{q}}$.
- $\mathbf{H}(model, \mathbf{q})$ by using the Composite-Rigid-Body Algorithm (CRBA).

To represent position, velocities, acceleration, and forces, spatial algebra is nowadays one of the most used representation in robotics. Introduced by Roy Featherstone [7], it allows a correct mathematically algebra notation to represent the relationships between quantities of a rigid multibody system.

B. Applications

1) Walking: In the context of humanoid robotics, and following the method initiated by Nishiwaki et al. [3] for reactive walking, RNEA can be used to filter dynamically a Center-of-Mass trajectory. Typically, for a humanoid robot, the goal is to compute 160 iterations for 30 DOFs robot in less than 200 μs. This is correspond to a 1.6 s window for an optimal controller as the one used classically for the HRP-2 robot, with a 200 Hz sampling.

To reach this objective, a RNEA call should last no more than 1.25 μs. The result achieved by our library is 11.32 μs on a i7-2820QM CPU of 2.3 GHz. This imply to use a subsampling mechanism such as the one proposed by Nishiwaki et al. [3].

One may argue that sub-sampling this trajectory can decrease the overall complexity for walking (Nishiwaki uses a 20 ms time step). However we anticipate a demand on such information with the recent increase in work considering extreme motion involving strong angular moments (such as the Parkour [8]). In such situation linearization will be only correct at high frequency.

2) Whole-body control: Staying in the context of humanoid robotics, and by now considering a control architecture based on the inverse dynamics [9], [1], computing the inertia matrix at a very high speed may be necessary.

Following the previous remark on sub-sampling, this is probably not required for motion with a slow dynamic.

C. State of the art

A classical approach is to implement RNEA and CRBA using instances of classes and structures, and to iterate over each element of the kinematic tree. RBDL (Rigid Body Dynamic Library) and KDL (Kinematic Dynamic Library) are using this approach. RBDL is following specifically the directives suggested by Featherstone in his book [7] by implementing the spatial algebra through C++ operators overloading. A different approach is to use symbolic computation. One of the first software implementing this approach is Symoro+ by Khalil et al. [10]. The software is able to produce reduced models and to generate code where unnecessary computation are stripped out. SD/FAST is a commercially available toolbox used by Boston Dynamics and the MIT leg laboratory [11]. HuMAnS is a toolbox written by P.-B. Wieber [12] relying on Maple for manipulating the robot kinematic tree. ROBOTRAN is a JAVA application which performs symbolic manipulation and generates code integrated in MATLAB and the Simulink toolbox [13]. In each case, the robot model is written in a specific language to be simplified and finally generated in C++.

In Chapter 10 of his book, Featherstone indicates that a potential disadvantage of symbolic simplification is that it does not use vector-based arithmetic of CPUs, and that if the code is too large then it does not fit the computer's instruction cache. In this work the first point is addressed by using Eigen, a widely known linear algebra library carefully written to exploit vector-based floating-point units. The second point is solved by using template programming which allows each instanciation of the template to invoke common code while avoiding indirect calls.

More generally, in this work, we propose to use meta-programming to exploit directly the C++ capabilities to perform symbolic computation at compile time without any additional library or software other than the BOOST libraries. The current solution is completed with a C++ program which is parsing URDF files and generates appropriate structures to decouple model and instance of robots. The current limitation of the approach is the case where model modification is needed. A solution (not yet implemented in the current software solution) is an abstract layer switching from an optimized implementation to a generic implementation (RBDL or KDL). Another more complex solution is to compile the new model and link dynamically to this new library.

Our contribution is to have written, to our knowledge, one of the first meta-programming library aiming at computing inverse dynamic for the robots. The application targeted in this paper is humanoid robot walking. By putting together the power of functional programming with C++ speed, our library is able to achieve performance necessary for real-time control.

II. Template metaprogramming for Inverse Dynamics

This work takes its root from the fact that code generated by symbolic formal computation has in general better performances than generic code for complex robotic models. This comes for the inner structure of the problem itself. For instance, the robot inertia matrix is sparse because the sub-blocks matrices correspond to the kinematic branches. Then when no coupling exists between some branches, the related sub-blocks mainly is equal to zero. Formal computation detects such branches and suppresses the related computation. We are proposing to make the same kind of operations by exploiting C++ template meta-programming.

The result is a template library able to develop an optimized computational tree to evaluate the inverse dynamics of a robot class named **Robot** and to apply it onto the specific instance named **robot**, while considering the following complete state $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$. This is obtained by writing the following line:

```
rnea< Robot, true >::
  run(robot, q, dq, ddq);
```

It is template metaprogramming because the code function:

```
rnea<Robot,true>::run(robot,q,dq,ddq)
```

is adapted at compile time to the type **Robot**. During execution, the model specificities are used to update **robot** internal variables according to $(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$.

A. Simple example of functional programming using C++

A famous example of functional programming in C++ is the factorial computation:

```
template <int n>
struct factorial {
 enum {
   value = n * factorial<n - 1>::value
 };
};

template <>
struct factorial<0> {
enum { value = 1 };
};
```

At compile time, when the compiler finds

```
factorial<4>
```

it directly computes 24 by successive substitutions. The same type of mechanism is used to develop the RNEA algorithm from the model **Robot**. In order to achieve a decoupling between the model and the robot instances Metapod is using a template based container proposed by boost::fusion which allows an array like syntax with templates [1]. The

[1] A naive description of the approach is available here https://github.com/laas/metapod/wiki/Naive-implementation---First-approach

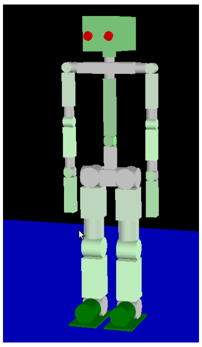

Fig. 1. Model used for benchmarking

kinematic structure is then embedded through indexes that are found classically in an array, but here declared through boost::fusion.

B. Spatial algebra and meta-programming

The spatial algebra can be easily implemented in meta-programming thanks to static polymorphism. This is allowing to keep a very generic code quite similar to Oriented Object programming but without the additional cost induced by virtual methods. We can also avoid errors by using strongly typed operators. It has also been possible to implement the optimization described in Appendix 2 of Featherstone's book [7]. This is especially useful to simplify computation when parts of the robot are planar. One of the unitary test used in the library available through Github is the planar arm described in [14].

C. Visitor design pattern to compute inverse dynamic

The depth first exploration of the robot structure is performed thanks to the visitor design pattern. This design pattern has been used in all algorithms where an iterative exploration of the tree structure is necessary. Finally, thanks to the design pattern coupled to boost::fusion which make possible to aggregate different types, it is possible to decouple the model from its instance.

D. Benchmarks

We compared the performances of Metapod with RBDL by computing the inertia matrix and the inverse dynamics on the CPUs listed in Table I. In each case, the library was recompiled using the GNU gcc compiler with the optimization options (-O3 -NDEBUG) trying to generate the code producing the best performances. The model used for benchmarking is the sample humanoid model provided by Tokyo University in OpenHRP. The model is displayed in Fig. 1. For this specific work we concentrated in two algorithms which are used in controlling a humanoid robot, the Composite Rigid Body Algorithm (CRBA) which computes the inertia matrix, and the Recursive Newton-Euler Algorithm (RNEA) which provides the torques necessary to counterbalance the gravity.

Table II provides a quick view on the performance difference between Metapod and RBDL on various CPUs. The next section will detail some aspects related to the CPUs which will try to explain that the values provided here are to be taken cautiously and are likely to highly vary according to the CPU, the context of its overload, the compiler and the operating system. For this reason, we provide in Table II a ratio according to the CPU between the time provided by Metapod and RBDL. Note that CPU-2 corresponds to the HRP-2 real chipset with its real-time operating system. The interested user should perform an identical comparison for its own robot to have accurate time measurement. Here we have put the most favorable time measurements for both algorithms. Despite this high variability, Metapod is almost always better in performances than RBDL.

We have started some comparisons with symbolic computation libraries, but they usually imply the use of a costly third-part software. A comparison with HuManS on a structure similar to the model used here gives $8.35\mu s$ for the CRBA algorithm on CPU-1. To achieve this result on Metapod with CPU-1 it was necessary to use the compiler options given in the remainder of the paper. They changed nothing for HuManS. As the later one does not use Eigen, this implies that, as mentionned by Featherstone, the use of vector-based arithmetic has a strong impact when computing the inertia matrix. Thanks to the authors of Symoro+, and if the paper is accepted, we hope to augment the comparison with this software.

E. CPUs

The Intel company divides its chipsets into 3 categories: Core, Xeon and Atom. With its low power consumption and low heat dissipation the Atom processor is aimed for embedded systems. The first and second families are used respectively for desktop applications and high power computation. The Xeon Phi category is targeted for many cores applications.

1) TurboBoost: To speed up the performance of the Core and Xeon families their coprocessors embed a technology that Intel names TurboBoost. It consists in increasing the CPU speed while respecting safety rules regarding heat dissipation and voltage. The maximum speed that can be reached is called the maximum turbo frequency. For instance regarding Table II CPU-1 and CPU-3 can reach respectively 3.4 GHz (from 2.3 GHz) and 3.8 Ghz (from 3.6 GHz). There is no guarantee that the maximum frequency is always reached as it depends on the current status on the chipsets. The Atom chipsets do not have the TurboBoost technology as well as the Core2 Duo E7500 (CPU-2) currently inside the HRP-2 humanoid robot. This is reflected by the numbers given in Table II, where the raw times are 10 times faster on Xeon chipsets than on Atom. Interestingly one would have expected that CPU-1 with the turbo-boost would be at best near 0.9 the speed of CPU-3, as it is the ratio of their respective Turbo-Boost frequency. This relationship is not verified between CPU-1 and CPU-3. In addition the description of the TurboBoost technology in the Sandy Bridge micro architecture [15] shows a complex pattern depending on the current overall status of the CPU. It is worth noting

Nickname	CPU name	Frequency	Core number	Cache size	Distribution
CPU-1	Intel(R) Core(TM) i7-2820QM CPU	2.3 GHz	8	8 Mb	Ubuntu 12.04 LTS
CPU-2	Intel(R) Core2(TM) Duo E7500	2.8 GHz	1	3 Mb	Ubuntu 10.04 LTS
CPU-3	Intel(R) Xeon(R) CPU E3-1240 v3	3.4 GHz	8	8 Mb	Ubuntu 12.04 LTS
CPU-4	Intel(R) Atom(TM) CPU N720	1.6 Ghz	1	512 Kb	Ubuntu 12.04 LTS

TABLE I

CPUS USED FOR THE BENCHMARKING. CPU-1 IS A LAPTOP, CPU-2 IS ONE USED IN HRP-2, CPU-3 IS A DESKTOP, AND CPU-4 IS A NETPC. CPU-2 IS USING ONLY ONE CORE DUE TO THE REAL-TIME OPERATING SYSTEM RECOGNIZING ONLY ONE CORE.

Library	Algorithms	CPU-1		CPU-2		CPU-3		CPU-4	
		μs	ratio	μs	ratio	μs	ratio	μs	ratio
Metapod	CRBA	7.89	0.77	12.97	0.69	5.08	0.76	61.06	0.76
	RNEA	9.28	0.97	13.85	0.75	4.93	0.82	60.98	0.90
RBDL	CRBA	10.16		18.59		6.60		79.68	
	RNEA	9.5		18.66		5.99		67.68	

TABLE II

PROCESSING TIME FOR THREE BASIC ALGORITHMS USED IN CONTROL OF HUMANOID ROBOT: INERTIA MATRIX COMPUTATION (CRBA), INVERSE DYNAMICS (RNEA). THE COMPUTATION HAVE BEEN TESTED ON 4 CPUS.

that the operating system can also request TurboBoost by sending a signal (P-state request) if it detects an overloading status.

2) Branch prediction: Performances depend on another key technology: branch prediction. The chipset maintains counters to predict the branches behavior. Depending on the statistics of the branch, the predictor tries to guess what will we be the most probable code to be executed by the CPU. The code and its data are then prefetched from the memory while the CPU is decoding the current operation. Branch predictors can also detect cycles and save in cache intermediate variables to avoid costly access memory. Branch predictors is also able to indirect branch where multiple address are possible targets. The interest of the symbolic approach is to avoid as much as possible to make branches in the code. It facilitates the work of the branch predictor as well as the memory pre-fetching mechanism. This may explain why in Table II Metapod provides a 25-30 % increase in efficiency on the CRBA algorithm. Unfortunately chipsets providers, in general, do not give a detailed description of their branch predictors. For this reason, users have usually to reverse engineer the behavior of the chipset in order to create branch predictor aware compilers [16], [17]. We used Valgrind framework to analyse cache access and branch prediction. The result is compiled in Table III and in Table IV. From Table III Metapod is doing more misprediction than RBDL for the Level 1 cache both for instruction and data, but less for the last level cache. As a last level misprediction implies a memory access, with the slower bus frequency it is the most costly. Therefore this result is in favor of Metapod. In addition Table IV shows clearly that branch misprediction is far lower for Metapod than for RBDL.

3) Compiler: The importance of the compiling options differs according to the library and the CPUs. It was particularly strong on the Atom chipset with a gain of $20\mu s$ for Metapod and RBDL. On RBDL, the impact of compiling option was null on CPU-1 and CPU-3 whereas it is improved by 10 % on Metapod. The compiler options used to exploit vector-based arithmetic and loop optimizations are [2]:

```
-msse -msse2 -msse3 -march=corei7 -mfpmath=sse
-fivopts -ftree-loop-im -fipa-pta
```

In addition during the linking phase it is possible to strip out useless code and perform whole-program optimization with the following optionx:

```
-fwhole-program
```

Again the static structure of the code generated through Metapod simplifies the work of optimization strategies applied by the compiler (here gcc).

III. DYNAMIC FILTERING FOR WALKING

To illustrate the use of Metapod on a humanoid robot, we consider an application where the robot has to evolve in a factory. AIRBUS/Future of the Aircraft factory is currently evaluating the potential of humanoid robots in such context. The use case considered here is the HRP-2 humanoid robot bringing an electric screw driver to an assembly line. On the other hand, in the context of the Koroibot project aiming at studying human walking to improve humanoid robot motion, we have created a stair climbing motion primitive. The behavior consisting in bringing a tool while climbing the stairs is then a combination of the two motion primitives. We show next that, thanks to Metapod, the dynamical consistency can be realized in real time.

A. Dynamic filtering on a sub-sampled walking pattern generator

This work is based upon the real time walking control system described by Nishiwaki in [3]. One key ingredient is

[2]We kindly invite the interested reader to follow the gcc manual for a more detailed explanation of the options

Library	Ir	I1mr	ILmr	Dr	D1mr	DLmr	Dw	D1mw	DLmw
RBDL	8,295,618,744	39,643	3,995	3,447,245,971	2,936,723	7,322	1,617,919,310	607,848	3,447
Metapod	4,529,574,180	69,802,287	2,722	2,154,169,130	9,709,886	5,040	1,034,216,699	27,501,390	1,136

TABLE III

Ir: NUMBER OF EXECUTED INSTRUCTIONS, **I1m1r**: INSTRUCTION READ MISSES FOR CACHE L1, **ILmr**: INSTRUCTION READ MISSES FOR LAST LEVEL CACHE, **Dr**: NUMBER OF CACHE READ, **D1mr**: DATA READ MISSES FOR CACHE L1, **DLmr**: DATA READ MISSES FOR LAST LEVEL CACHE, **Dw**: NUMBER OF CACHE WRITE, **D1mw**: DATA WRITE MISSES FOR CACHE L1, **DLmw**: DATA WRITE MISSES FOR LAST LEVEL CACHE. TESTS REALIZED ON CPU-1

Library	Bc	Bcm	Bi	Bim
RBDL	245,662,040	7,604,170	45,177,857	269,654
Metapod	98,901,411	977,079	14,105,063	726

TABLE IV

Bc: CONDITIONAL BRANCHES EXECUTED, **Bcm**: CONDITIONAL BRANCHES MISPREDICTED, **Bi**: INDIRECT BRANCHES EXECUTED **Bim**: INDIRECT BRANCHES MISPREDICTED. TESTS REALIZED ON CPU-1

to generate real-time dynamically stable CoM and Center-of-Pressure (CoP) trajectories using the cart-table model [18]. Then to correct the inertial effects induced by the legs movement Nishiwaki uses a subsampled dynamical filter. Indeed, the results tested on HRP-2 have shown that the CoP trajectory followed by the robot are not the desired one, the difference can be as big as 15 mm. To avoid this problem, the solutions consist in designing a controller regulating the linear and angular momentum [19] in addition to a dynamical filter aiming at correcting the CoM trajectory in order to take into account the dynamics of the robot. The first kind of solution is mostly used to reject instantaneous perturbations. However, it uses additional degrees of freedom that could have been used for manipulation tasks for example. For this reason, in our specific case, we have only implemented the dynamical filter as the commercially available stabilizer integrate already a perturbation rejection strategy. In the following the dynamic filter depicted by Nishiwaki and implemented in this paper is described in details:

- The walking pattern generator (WPG) defines a trajectory for the CoM, the feet and the CoP. The desired CoP trajectory is noted $\mathbf{CoP}^*$.
- The dynamic filter starts by computing the joint trajectories using the analytical inverse kinematics. We make the assumption that the CoM and the free flyer are rigidly connected.

$$(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) = IK(ponctual\_model, \mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}}, \mathbf{X}^f)$$

  with $\mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}}$ et $\mathbf{X}^f$ being respectively the position, the velocity, the acceleration of the CoM and the feet position.
- With the joint trajectory, it is possible to compute the inverse dynamics and to find the CoP matching up to the real robot motion. We call it the multi-body CoP noted $\mathbf{CoP}^{MB}$.

$$\begin{aligned}
(\mathbf{f}, \boldsymbol{\tau}) &= ID(model_complet, \mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \\
CoP_x^{MB} &= -\frac{\tau_y}{mg} \\
CoP_y^{MB} &= \frac{\tau_x}{mg} \\
CoP_z^{MB} &= 0 \\
\boldsymbol{\Delta}\mathbf{CoP} &= \mathbf{CoP}^* - \mathbf{CoP}^{MB}
\end{aligned}$$

- This provides an error between $\mathbf{CoP}^*$ and $\mathbf{CoP}^{MB}$. This error is computed over a time window and is injected in a preview control (PC) in the shape of an LQR described by Kajita and al. [18]. The result of this step is an error for the CoM in position, velocity and acceleration.

$$\boldsymbol{\Delta}\mathbf{CoM} = PC(\boldsymbol{\Delta}\mathbf{CoP})$$

- We can then sum this error on the reference CoM ($\mathbf{CoM}^*$) to correct the trajectory.

$$\mathbf{CoM} = \mathbf{CoM}^* + \boldsymbol{\Delta}\mathbf{CoM}$$

  With $\mathbf{CoM} = \begin{bmatrix}\mathbf{c} & \dot{\mathbf{c}} & \ddot{\mathbf{c}}\end{bmatrix}^T$
- Finally, the new CoM trajectory is used to compute the joint trajectory using the Inverse Kinematics.

The scheme depicted in Fig. 2 represents the above steps. This dynamic filter is based on the Newton-Raphson algorithm using the CoM trajectory as free variables. This method does not guarantee the convergence. However during tests on HRP-2 with other WPG, we observed that the CoP trajectory is corrected in one iteration. The goal is to implement this method on the WPG of Andrei Herdt [20] and Morisawa's [21].

B. Application on HRP-2

We propose to use this filter on the WPG proposed by Morisawa and al. in [21]. This algorithm uses an analytical form of the CoM and CoP trajectories assuming that the later is a third order polynomial. The Morisawa's WPG can either define off line the whole trajectory at once, or change online the foot steps given as reference.

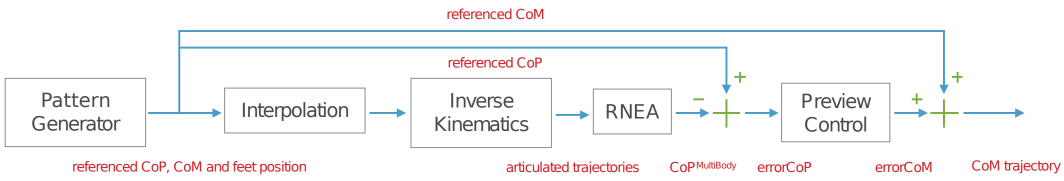

Fig. 2. Scheme of the walking pattern generator

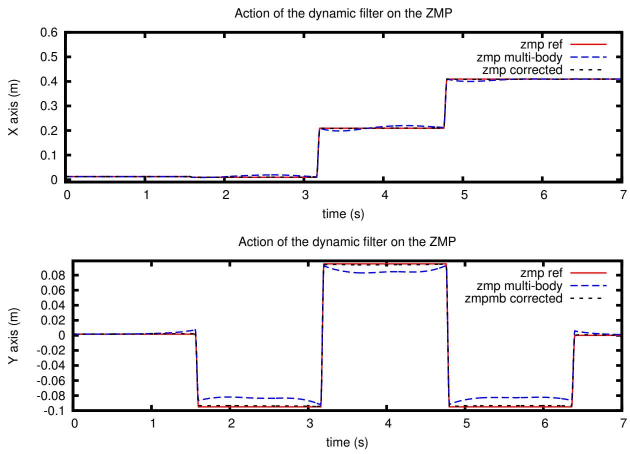

Fig. 3. Results obtained on the Morisawa offline WPG

To use the filter with its full potential on the online Morisawa's WPG we need to compute the trajectory over 1.6 s in advance, i.e. two steps forward. The computation of the trajectory is done every 5 ms including the control, so the remaining time for the WPG is 3 ms. Moreover, because the robot HRP-2 is controlled at 200 Hz, we need to compute 320 RNEA to use the whole preview. Our implementation, as fast as it is, does not allow this kind of computation, hence we used a strategy proposed by Nishiwaki, i.e. we sub sampled the preview window at a period of 0.05 s which leads to 32 RNEA every 0.1 s. We rather compute the whole dynamic filter for the offline WPG of Morisawa.

This study is based on the offline WPG. We computed 1.6 s more than the initial trajectory and we computed the multi-body CoP for each sampling time.

The results obtained are illustrated in the Fig. 3. The graphs here contains each :

- the trajectories of the reference CoP,
- the multi-body CoP computed with RNEA,
- and the corrected CoP obtained with the RNEA using the corrected CoM

The upper graph depicts the evolution on the axis X in function of the time and the lower graph shows the evolution of the Y axis in function of the time. This data are extracted from a straight walking using the offline WPG of Morisawa.

We can here easily see the influence of the filter, i.e. the corrected multi-body CoP is almost fused with the reference one. In terms of distance, the maximum error between the reference CoP and the multi-body CoP is 15.3 mm while the maximum distance after correction is 1.7 mm. In average the error before correction is 5.7 mm and after it is 0.7 mm.

In term of computation time, we executed the algorithm that creates a straight flat walking on the HRP-2 robot. The results are 15.0 us for the average time used to compute the RNEA during the whole trajectory and 342.0 us for the maximum duration of one iteration of RNEA. This peak corresponds to the first iterations of the algorithm, corresponding to memory allocation.

We have done a series of experiments using the dynamic filter including the straight walking and climbing stairs. For the climbing we had four scenarios: one simple walk, a walk while swinging the arm and holding a lightweight tool in one hand, another one holding the tool with 2 hands. In the last one the tool is hold with the hands lifted over the head. HRP-2 went through all the pattern has depicted in the companion video. We also succeded to make the robot going down the stairs with the lightweight in to hands, however we did not manage to make HRP-2 perform the rest of the scenario while going down. One of the raisons that explain this behaviour is that the dynamic filter places the CoM in a way such that the CoP is inside the support foot. To compensate for the arms being in front, the CoM is send slightly backward. The leg is then stretched and create a singularity. This shows the limit of this approach which calls for a more advanced development such as optimal control, whole-body predictive control or planning.

In this section we have shown that one iteration of a Newton-Raphson algorithm can make the CoP converge towards its reference and it could be applied to WPG focusing on the under-actuated part of the robot.

IV. Conclusion

We have presented a C++ library for efficient humanoid dynamic computation without relying on any particular symbolic computational tool. Tests have shown that we have in general better performances than RBDL. Some preliminary results show that we have a computational speed similar to symbolic computation. This is achieved through a mix of code generation and C++ meta-programming. Its use is established in the context of dynamic filtering for real-time walking gait generation, and more specifically applied to the humanoid robot HRP-2. The library is available under GPL-license at the following url: `https://github.com/laas/metapod`.

REFERENCES

[1] N. Mansard, "A dedicated solver for fast operational-space inverse dynamics," in *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 4943 –4949.

[2] A. D. Prete, F. Romano, L. Natale, G. Metta, G. Sandini, and F. Nori, "Prioritized optimal control," in *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[3] K. Nishiwaki and S. Kagami, "Online walking control system for humanoids with short cycle pattern generation," *The International Journal of Robotics Research*, vol. 28, no. 6, pp. 729–742, 2009.

[4] S. Feng, X. Xinjilefu, W. Huang, and C. G. Atkeson, "3d walking based on online optimization," in *IEEE/RAS Int. Conf. on Humanoid Robotics (ICHR)*, 2013, pp. 21–27.

[5] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 1168–1175.

[6] V. Zhirnoz, R. K. C. III, J. A. Hutchby, and G. I. Bourianoff, "Limits to binary logic switch scalinga gedanken model," *Proceedings of the IEEE*, vol. 91, pp. 1935–1939, 2003.

[7] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer Science, 2008.

[8] C. M. Dellin and S. S. Srinivasa, "A framework for extreme locomotion planning," in *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 989–996.

[9] L. Sentis and O. Khatib, "A whole-body control framework for humanoids operating in human environments," in *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, 2006.

[10] W. Khalil and D. Creusot, "Symoro+: A system for the symbolic modelling of robots," *Robotica*, vol. 15, no. 2, pp. 153–161, 1997.

[11] P. Inc., "Sd/fast." [Online]. Available: http://www.ptc.com/support/sdfast/index.html

[12] P.-B. Wieber, F. Billet, L. Boissieux, and R. Pissard-Gibollet, "The humans toolbox, a homogeneous framework for motion capture, analysis and simulation," in *Ninth ISB Symposium on 3D analysis of human movement*, 2006.

[13] *ROBOTRAN: Symbolic generator of multibody systems*. [Online]. Available: http://www.robotran.be/

[14] M. Spong, S. Hutchinson, and V. M., *Robot Modeling and Control*. John Wiley and Sons, 2006.

[15] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, "Power-management architecture of the intel microarchitecture code-named sandy bridge," *Micro, IEEE*, vol. 32, no. 2, pp. 20–27, March 2012.

[16] Z. Wang and D. A. Jiménez, "Program interferometry," in *International IEEE International Symposium on Workload Characterization (IISWC)*, 2011, pp. 172–183.

[17] V. Uzelac and A. Milenkovic, "Experiment flows and microbenchmarks for reverse engineering of branch predictor structures," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2009, pp. 207–217.

[18] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, September 2003, pp. 1620–1626.

[19] ——, "Resolved momentum control: Humanoid motion planning based on the linear and angular momentum," 2003.

[20] A. Herdt, H. Diedam, P. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement," *Advanced Robotics, 24*, vol. 5, no. 6, pp. 719–737, 2010.

[21] M. Morisawa, K. Harada, S. Kajita, S. Nakaoka, K. Fujiwara, F. Kanehiro, K. Kaneko, and H. Hirukawa, "Experimentation of Humanoid Walking Allowing Immediate Modification of Foot Place Based on Analytical Solution," in *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, 2007, pp. 3989–3994.

ACKNOWLEDGMENT

The Metapod library was made possible thanks to the work of Maxime REIS. Martin FELIS and Antonio EL-KHOURY took an active part in the improvement of Metapod. We warmly thank them for their help.