Pemodelan dan Simulasi Dinamika Lengan Robot 3-DOF Menggunakan Perangkat Lunak *Open Source*

Indar Sugiarto
Jurusan Teknik Elektro, Fakultas Teknologi Industri, Universitas Kisten Petra
Jl. Siwalankerto 121-131, Surabaya - 60236
Email: indi@petra.ac.id

ABSTRAK

Dalam paper ini dijelaskan salah satu metode alternatif untuk analisa dinamik sebuah robot lengan dengan menggunakan perangkat lunak *open source* yang disebut ODE (*open dynamic engine*). Dengan menggunakan ODE, sebuah robot lengan dapat dimodelkan dan disimulasikan dengan sederhana dan memberikan hasil yang akurat. Robot lengan yang dimodelkan dan disimulasikan dalam paper ini adalah Movemaster EX RV-M1 dari Mitsubishi. Simulasi dilakukan setelah model robot selesai dibuat dan simulasi dijalankan dengan membuat supaya semua link robot bergerak serempak membentuk garis lurus dan berputar dari sudut 0° hingga 100°. Dari hasil pemodelan dan simulasi didapatkan bahwa masing-masing *revolute joint* dari robot tersebut membutuhkan torsi yang berbeda-beda dan ditemukan pada kecepatan 1.26 rad/s, *joint* siku (*elbow*) membutuhkan torsi yang lebih besar untuk menggerakkan lengan depan (*forearm*) dari robot, yaitu sebesar 3101.79 $g.m^2.s^{-2}$. Torsi yang besar ini tidak hanya diakibatkan karena beban *link* yang harus diangkat, tetapi juga *constraint force* yang harus dilawan oleh *joint* siku (*elbow*) saat bergerak bersama-sama dengan *joint* yang lain. Hasil penelitian ini tidak menghasilkan program visualisasi tetapi dapat dikembangkan misalnya untuk melengkapi program-program simulator yang kebanyakan dikembangkan hanya berdasarkan simulasi dan pemodelan kinematika saja.

Kata kunci: pemodelan, simulasi, robot lengan 3DOF, ODE

ABSTRACT

This paper describes an alternative method for analyzing robot dynamics using open source software so-called ODE (open dynamic engine). Using ODE, an arm robot can be modeled and simulated in a simple but accurate fashion. The arm robot which is being modeled and simulated in this paper is the Movemaster EX RV-M1 from Mitsubishi. The simulation is performed after the robot has been completely modeled and the simulation was made so that all links within the robot will move simultaneously forming a straight line and rotating from 0° to 100°. The modeling and simulation result shows that each revolute joint of the robot requires distinct torque and it is found that at the speed of 1.26 rad/s, the elbow joint has the higher torque to move the body at the level of 3101.79 $g.m^2.s^{-2}$. This high torque was due to constraint force when the joint was moving altogether. This research does not produce any visualization effect but it can be extended to enrich any other simulator program which works according to kinematics modeling only.

***Keywords**: modeling, simulation, robot arm 3DOF, ODE*

PENDAHULUAN

Dalam penelitian sebelumnya tentang simulasi dan pemodelan sistem-sistem fisika dan mekanika, diketahui bahwa salah satu perangkat lunak *open source* yang kenal dengan nama ODE (*open dynamic engine*) merupakan salah satu *framework* yang kompleks dan dilengkapi dengan pustaka-pustaka yang berisi fungsi-fungsi untuk melakukan kalkulasi numerik guna pemodelan sistem dinamis [1].

ODE tersusun atas algoritma-algoritma untuk mensimulasikan interaksi dinamika antar benda-benda dalam sebuah kerangka ruang berdasarkan konsep dinamika benda tegar (*rigid body*). Namun ODE bukanlah program yang dapat menampilkan gambar / grafis, tetapi hanyalah pustaka yang berisi rutin-rutin untuk perhitungan gerakan sebuah benda berikut interaksinya dengan benda-benda lain dalam kerangka ruang yang sama berdasarkan hukum fisika, utamanya mekanika klasik (*Newtonian mechanics*) [2]. ODE sendiri tersusun atas dua komponen utama, yaitu komponen simulasi dinamika benda tegar (*rigid body dynamics simulation engine*) dan komponen pendeteksi tumbukan (*collision detection engine*). Hasil-hasil perhitungan oleh ODE selanjutnya dapat digunakan oleh program-program simulator grafis untuk menghasilkan efek visual ataupun digunakan langsung oleh sistem kontrol robot yang berbasis PC.

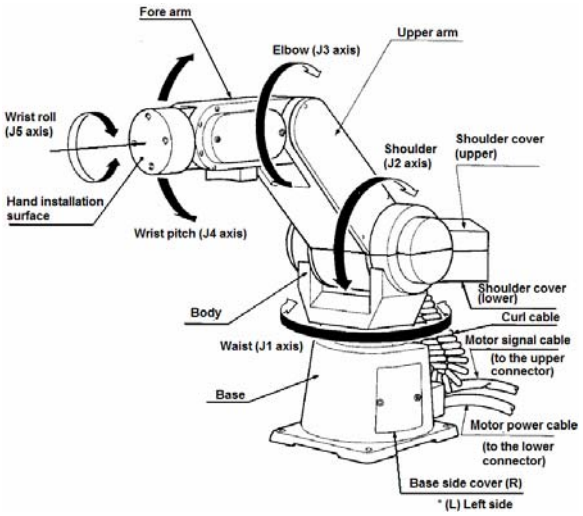
Catatan: Diskusi untuk makalah ini diterima sebelum tanggal 1 Desember 2008. Diskusi yang layak muat akan diterbitkan pada Jurnal Teknik Elektro volume 9, nomor 1, Maret 2009.

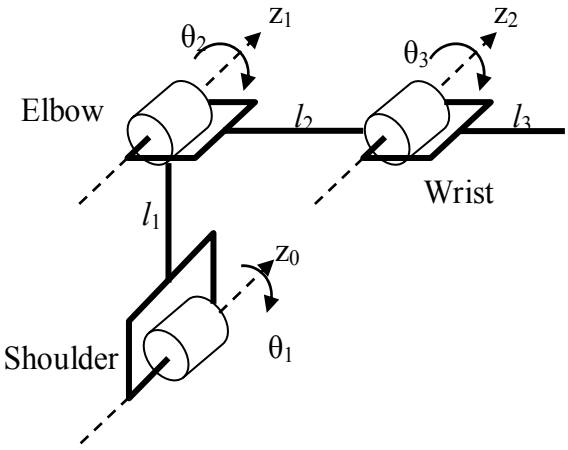
Salah satu kekuatan ODE yang menonjol adalah solusi terhadap permasalahan *Linear Complementary Problem* (LCP) menggunakan teknik *Lagrange Multiplier* yang banyak muncul pada sistem dinamis yang melibatkan fenomena gesekan dan tumbukan [3], [6]. Di dalam ODE dikenal beberapa obyek dasar yang parameter-parameternya melambangkan besaran fisik dari sebuah sistem dinamik dan sering digunakan dalam analisa kinematika maupun dinamika dari sebuah robot. Obyek-obyek dasar tersebut adalah: *world, body, geom, space,* dan *joint*. Sintaks yang digunakan oleh ODE sama dengan bahasa pemrograman C++ dan bisa berinteraksi dengan program-program yang dibuat dengan ANSI-C.

Pada penelitian ini, ODE versi 0.5 digunakan untuk mendapatkan model dinamika dari robot lengan 3-DOF yang presisi sekaligus melakukan simulasi terhadap model yang dibuat. Robot yang dimodelkan adalah Movemaster EX RV-M1 yang merupakan robot industri bertipe *articulated configuration robot* (RRR) [4]. Penelitian ini tidak menghasilkan program visualisasi tetapi dapat dikembangkan misalnya untuk melengkapi program-program yang dibuat untuk project-project semacam [5] yang dikembangkan hanya berdasarkan simulasi dan pemodelan kinematika saja. Untuk memudahkan pemodelan dan simulasi dalam paper ini, robot Movemaster EX RV-M1 dianggap memiliki *link-link* yang terhubung dengan *joint constraint* yang bersifat *holonomic*.

PEMODELAN UNTUK SIMULASI

Konstruksi robot lengan Movemaster EX RV-M1 dan representasi simbolisnya ditunjukkan pada gambar 1. Dari buku manual robot Movemaster EX RV-M1 [4], diperoleh data-data seperti pada table 1.





Gambar 1. Konstruksi robot lengan Movemaster EX RV-M1 dan representasi simbolisnya

Tabel 1. Spesifikasi robot Movemaster EX RV-M1

Dimensi lengan atas (upper arm)	250 x 106 x 94 mm
Panjang lengan depan (fore arm)	160 x 70 x 91 mm
Panjang pergelangan (wrist)	72 x 30 x 30 mm
Kecepatan max bahu (shoulder)	72°/s
Kecepatan max siku (elbow)	109°/s
Kecepatan max pergelangan (wrist)	100°/s
Massa lengan atas	0.255 kg
Massa lengan depan	0.306 kg
Massa pergelangan	0.102 kg

Seperti diketahui, dalam analisa dinamik sebuah robot biasanya digunakan salah satu dari formulasi *Newton-Euler* ataupun *Lagrangian*. Dalam paper ini, analisa dinamik dari robot lengan didasarkan pada formulasi *Lagrangian* karena formulasi ini secara *native* sudah terimplementasi di dalam ODE. Persamaan dasar dari fungsi *Lagrangian* L didapatkan dari selisih antara energi kinetik total *K* dan energi potensial *P* yang tersimpan dalam sebuah sistem dinamik, yang dalam koordinat Kartesius dapat dinyatakan sebagai:

$$L = K(\dot{\boldsymbol{x}}) - P(\boldsymbol{x}) \quad (1)$$

dengan $\boldsymbol{x} = [x_1, x_2, ..., x_n]^T$

Atau jika menggunakan konsep *generalized coordinates* $\boldsymbol{q} = [q_1, q_2, ..., q_s]^T$, dimana x_i (i = 1, 2, ... n) adalah fungsi dari $\boldsymbol{q}$ dan $\dot{\boldsymbol{x}}_i$ adalah fungsi dari $\boldsymbol{q}$ dan $\dot{\boldsymbol{q}}$, maka fungsi *Lagrangian* pada persamaan 1 di atas dapat ditulis menjadi:

$$L = K(q, \dot{q}) - P(q) \quad (2)$$

Untuk robot lengan dengan n-link, maka energi kinetik yang dimiliki untuk *link* ke-i adalah:

$$K_i = \frac{1}{2} m_i \bar{v}_i^T \bar{v}_i + \frac{1}{2} \bar{\omega}_i^T \mathbf{I}_i \bar{\omega}_i \quad (3)$$

Dengan demikian,

$$K = \sum_{i=1}^{n} K_i \quad (4)$$

Dengan memanfaatkan persamaan kinematik untuk lengan robot dengan n-*link*, didapatkan:

$$\dot{\boldsymbol{X}} = J\dot{\boldsymbol{q}} \quad (5)$$

dimana $\dot{\boldsymbol{X}} = [\boldsymbol{v}^T \ \boldsymbol{\omega}^T]$, dengan $\boldsymbol{v}$ adalah vektor kecepatan linier dan $\boldsymbol{\omega}$ adalah vektor kecepatan sudut serta J adalah matrix Jacobian dengan susunan sebagai berikut:

$$J = \begin{bmatrix} J_{l1}, & J_{l2}, & \cdots, & J_{ln} \\ J_{a1}, & J_{a2}, & \cdots, & J_{an} \end{bmatrix} \in R^{6xn} \quad (6)$$

Dengan demikian, vektor kecepatan linier $\boldsymbol{v}$ dan vektor kecepatan sudut $\boldsymbol{\omega}$ dapat ditulis sebagai:

$$\boldsymbol{v} = J_{l1}\dot{q}_1 + J_{l2}\dot{q}_2 +, \cdots, +J_{ln}\dot{q}_n \quad (7)$$
$$\boldsymbol{\omega} = J_{a1}\dot{q}_1 + J_{a2}\dot{q}_2 +, \cdots, +J_{an}\dot{q}_n \quad (8)$$

dimana J_{li} dan J_{ai} pada persamaan 6 adalah vektor-vektor yang berhubungan dengan kecepatan linier dan kecepatan sudut dari masing-masing persendian.

Dan jika diterapkan pada persamaan 3 akan menghasilkan:

$$K = \frac{1}{2} \dot{\boldsymbol{q}}^T D(\boldsymbol{q}) \dot{\boldsymbol{q}} \quad (9)$$

dimana $D(\boldsymbol{q})$ adalah matrix *inertia* atau tensor *inertia* dari sebuah robot lengan dengan n-*link*.
Rumusan untuk energi potensial dari robot lengan dengan n-*link* adalah:

$$P = \sum_{i=1}^{n} m_i \boldsymbol{g}^T \bar{\boldsymbol{p}}_{0i} \quad (10)$$

dimana $\boldsymbol{g}$ adalah vektor gravitasi, $\bar{\boldsymbol{p}}_{0i}$ adalah vektor posisi yang diukur dari basis robot ke pusat massa m_i.

Fungsi *Lagrangian* L kemudian dapat ditulis kembali menjadi:

$$L = K - P$$
$$= \frac{1}{2} \dot{\boldsymbol{q}}^T D(\boldsymbol{q}) \dot{\boldsymbol{q}} - \sum_{i=1}^{n} m_i \boldsymbol{g}^T \bar{\boldsymbol{p}}_{0i}$$
$$= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}(\boldsymbol{q}) \dot{\boldsymbol{q}}_i \dot{\boldsymbol{q}}_j - \sum_{i=1}^{n} m_i \boldsymbol{g}^T \bar{\boldsymbol{p}}_{0i} \quad (11)$$

Turunan parsial dari fungsi *Lagrangian* L terhadap $\dot{\boldsymbol{q}}_i$ adalah:

$$\frac{\partial L}{\partial \dot{q}_i} = \frac{\partial}{\partial \dot{q}_i} \left(\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}(\boldsymbol{q}) \dot{q}_i \dot{q}_j \right) = \sum_{j=1}^{n} d_{ij}(\boldsymbol{q}) \dot{q}_j \quad (12)$$

Dengan demikian,

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_i} = \sum_{j=1}^{n} d_{ij}(\boldsymbol{q}) \ddot{q}_j + \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial (d_{ij}(\boldsymbol{q}))}{\partial q_k} \dot{q}_k \dot{q}_j \quad (13)$$

Sedangkan

$$\frac{\partial L}{\partial q_i} = \frac{1}{2} \sum_{k=1}^{n} \sum_{j=1}^{n} \frac{\partial d_{kj}(\boldsymbol{q})}{\partial q_i} \dot{q}_k \dot{q}_j - \sum_{j=1}^{n} m_j \boldsymbol{g}^T \left(\frac{\partial \bar{\boldsymbol{p}}_{0j}}{\partial q_i} \right) \quad (14)$$

Seperti diketahui bahwa,

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = \tau_i \quad (15)$$

Maka jika tiga persamaan terakhir ini digabungkan, akan didapatkan persamaan diferensial untuk robot lengan dengan n-*link* sebagai berikut:

$$\sum_{j=1}^{n} d_{ij}(\boldsymbol{q}) \ddot{q}_j + \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial (d_{ij}(\boldsymbol{q}))}{\partial q_k} \dot{q}_k \dot{q}_j -$$
$$\frac{1}{2} \sum_{k=1}^{n} \sum_{j=1}^{n} \frac{\partial d_{kj}(\boldsymbol{q})}{\partial q_i} \dot{q}_k \dot{q}_j + \sum_{j=1}^{n} m_j \boldsymbol{g}^T \left(\frac{\partial \bar{\boldsymbol{p}}_{0j}}{\partial q_i} \right) = \tau_i \quad (16)$$

atau

$$\sum_{j=1}^{n} d_{ij}(\boldsymbol{q}) \ddot{q}_j + \sum_{j=1}^{n} \sum_{k=1}^{n} d_{ijk} \dot{q}_k \dot{q}_j + \phi_i = \tau_i \quad (17)$$

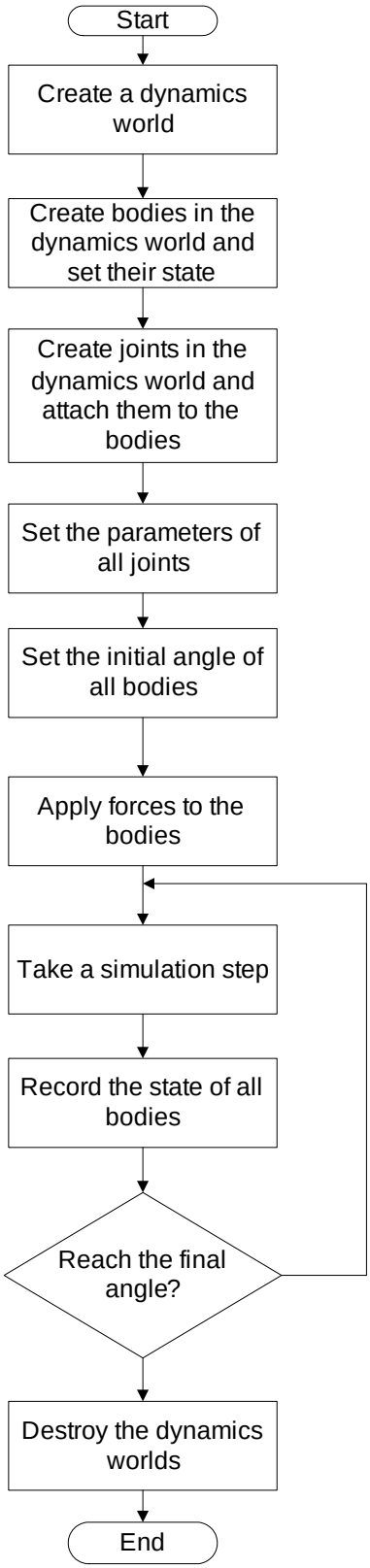
yang secara sederhana dapat ditulis sebagai:

$$D(\boldsymbol{q})\ddot{\boldsymbol{q}} + H(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \quad (18)$$

dimana $D(\boldsymbol{q})$ adalah matrix *inertia*, $H(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ adalah vektor kopling kecepatan, dan $\boldsymbol{G}(\boldsymbol{q})$ adalah vektor beban gravitasi.

Dengan menyelesaikan persamaan ini, akan dihasilkan nilai torsi yang diperlukan untuk memutar *link* ke-i dari lengan robot dengan kecepatan seperti ditentukan pada bagian $H(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$. Informasi torsi ini diperlukan saat proses disain sistem kontrol untuk lengan robot tersebut.

Untuk memodelkan dan mensimulasikan robot lengan dengan menggunakan ODE, diagram alir yang digunakan dalam paper ini seperti pada gambar 2.



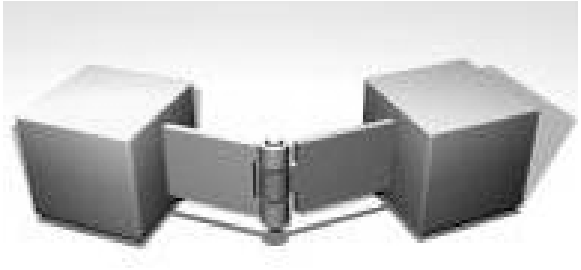
Gambar 2. Diagram alir pemodelan dan simulasi robot lengan dengan ODE

Dalam ODE, *body* pada dasarnya adalah sebuah obyek program untuk menyatakan sebuah benda tegar padat (*solid rigid body*). *Body* secara *default* tidak memiliki bentuk bangun tertentu. Tetapi untuk mempermudah pengguna, ODE menyediakan beberapa bentuk bangun dasar seperti kubus, prisma, silinder, bola, dan kapsul. Parameter-parameter sebuah *body* adalah sebagai berikut:

- Vektor posisi dari titik referensinya (biasanya pusat massanya), dinyatakan dalam sebuah vektor $\boldsymbol{p}^T = [p_x \;\; p_y \;\; p_z]$
- Kecepatan linier dari titik referensinya, dinyatakan dalam vektor $\boldsymbol{v}^T = [v_x \; v_y \; v_z] = [dp_x/dt \;\; dp_y/dt \;\; dp_z/dt]$
- Orientasi dari *body*, yang bisa dinyatakan dalam matrix rotasi R (3x3) atau vektor *quaternion* $\boldsymbol{q}^T = [q_s \; q_x \; q_y \; q_z]$
- Kecepatan sudut $\boldsymbol{\omega}^T = [\omega_x \;\; \omega_y \;\; \omega_z]$, yang menunjukkan perubahan orientasi *body* sepanjang waktu
- Massa
- Tensor inersia, berupa matrix 3x3 yang berisi informasi bagaimana massa dari sebuah *body* terdistribusi seputar pusat massa dari *body* tersebut.

World adalah sebuah kelas yang terdiri dari (dan mengikat) obyek-obyek *body* (secara dinamis maupun statis) bersama dengan variabel gaya dan waktu.

Pada dasarnya, simulasi dinamika menggunakan ODE bekerja dengan cara mengubah (memajukan) variabel waktu dari *world* dengan step tertentu dan kemudian meng-*update state* dari masing-masing *body* di dalam *world* yang bersangkutan berdasarkan hukum mekanika dan melibatkan gaya yang ada/didefinisikan dalam *world* tersebut. Berikut adalah cuplikan program untuk membuat *link* pertama beserta *joint*-nya dari robot lengan Movemaster EX dengan menggunakan obyek *body* berbentuk kubus. Dalam memodelkan lengan robot Movemaster EX, dipilih *joint* bertipe *hinge* karena pergerakan dari *link*-nya bersifat *revolute*.



Gambar 3. Struktur *joint hinge* yang digunakan untuk memodelkan persendian bertipe *revolute*

```
dWorldID _world = dWorldCreate();
dWorldSetGravity (_world,0,0,-9.81);

// Create a body
dBodyID UpperArm = dBodyCreate(_world);
dBodyID ForeArm = dBodyCreate(_world);

dMass mass;
  dMassSetBoxTotal(&mass,0.255f,0.25f,0.106f,0.094f);

dMatrix3 Theta1 = 0;
dQuaternion Q1;
dBodySetMass(UpperArm,&mass);
dBodySetPosition(UpperArm,0,0,0);
dBodySetRotation   (UpperArm, Theta1);
dBodySetQuaternion (UpperArm, Q1);
dBodySetAngularVel (UpperArm, 1.26f, 0, 0);
// equivalent with 72°/s

// Create joint
dJointGroupID ShoulderGroup = dJointGroupCreate (0);
JointID Shoulder = dJointCreateHinge (_world,
ShoulderGroup);
dJointAttach (Shoulder, UpperArm, ForeArm);
dJointSetHingeAnchor (Shoulder, 250.0f, 53.0f, 47.0f);
dJointSetHingeAxis (Shoulder, 250.0f, 106.0f, 47.0f);
```

Geom adalah parameter yang digunakan untuk merepresentasikan bentuk geometri dari sebuah obyek *body* untuk dipakai dalam algoritma pendeteksi tumbukan. Jadi setelah sebuah obyek *body* dibuat, obyek tersebut harus diberi parameter *geom* supaya bisa disimulasikan jika bertumbukan dengan obyek *body* lainnya. *Space* adalah sebuah obyek yang mengikat satu atau lebih *geom* menjadi satu kesatuan dan mengatur / mengendalikan algoritma pendeteksi tumbukan. Dengan kata lain, *geom* dan *space* adalah abstraksi dari *body* dan *world*. Berikut ini adalah kelanjutan dari skrip sebelumnya.

```
dSpaceID space = dHashSpaceCreate(0);

// Create a geom.
dGeomID geom = dCreateBox(space,1,1,1);
// Bind together our previously created body/link and the
geom
dGeomSetBody(geom,UpperArm);
```

Pada skrip di atas, obyek *body* diikat dengan sebuah *geom*. Jika sebuah *geom* tidak memiliki representasi *body*, maka *geom* tersebut bersifat statis. Cara seperti ini digunakan untuk menciptakan sebuah "*base*" untuk meletakkan model robot lengan yang dibuat. Sebuah obyek *body* bisa diikat dengan obyek *body* yang lain sehingga menjadi sebuah persendian, rantai, ataupun interaksi lainnya.

Beberapa *joint* bisa diikat dalam satu kesatuan yang disebut *joint group*. Tipe data *dJointGroup* digunakan untuk mengasosiasikan sebuah kelas dengan beberapa obyek *joint*. Kegunaan *joint group* adalah untuk memastikan bahwa hukum-hukum mekanika telah diaplikasikan pada masing-masing *joint* di dalam group tersebut dan mengurangi terjadi *joint error*. Namun dalam paper ini, *joint group* yang dibuat akan bernilai 1 karena hanya terdiri dari 1 buah *joint* saja (yaitu bertipe *hinge*).

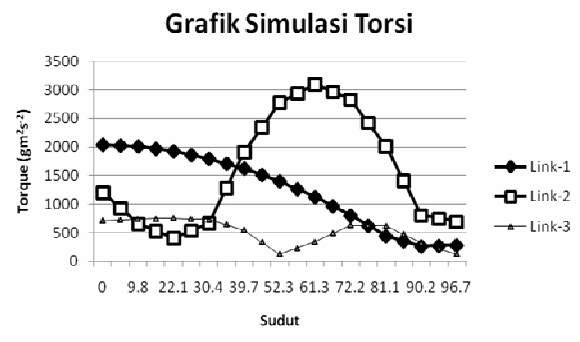
HASIL SIMULASI

Setelah parameter-parameter dari *body* dan *joint* dimasukkan, langkah berikutnya adalah melakukan simulasi sekuensial menggunakan fungsi simulasi yang disediakan ODE. Dalam paper ini dipilih fungsi *dWorldStep()* karena meskipun membutuhkan sumber daya memori komputer paling besar dan lebih lambat tetapi hasil simulasinya sangat akurat (dibandingkan fungsi *dWorldQuickStep()*). Dalam simulasi ini, lengan robot dibuat lurus dan dimulai dari sudut 0°. Motor pada ketiga link dari lengan robot tersebut harus membuat supaya lengan robot tetap dalam kondisi lurus dan torsi yang dibutuhkan oleh motor-motor tersebut akan disimpan di memori dalam bentuk *array*. Berikut adalah contoh skrip program yang dibuat (ditunjukkan hanya pada bagian link pertama saja, *link-link* yang lain memiliki format yang mirip).

```
//let's simulate
static float nbSecondsByStep = 0.001f;
// Find the time elapsed between last time
float nbSecsElapsed = _time.elapsed()/1000.0f;
// Find the corresponding number of steps that must be
taken
Int nbStepsToPerform = 0;
// Make these steps to advance world time
do
{
   // Step world
   dWorldStep(_world, nbSecondsByStep);
   // Remove all temporary collision joints after the stepping
   dJointGroupEmpty(ShoulderGroup);
   // Record body's state
   AngleDataUpperArm[nbStepsToPerform]                =
   dBodyGetRotation(UpperArm);
   TorqueDataUpperArm[nbStepsToPerform]               =
   dBodyGetTorque(UpperArm);
   nbStepsToPerform++;
//... and the other links as well!
} while dBodyGetRotation(UpperArm) >= (100*2*pi()/180);
// Restart the elapsed time counter
_time.restart();
```

Hasil dari simulasi di atas adalah 3 set *array* yang terdiri dari 2 buah data bertipe real, yaitu *AngleDataUpperArm**, *TorqueDataUpperArm**, *AngleDataForeArm**, *TorqueDataUpperArm**, *AngleDataWrist**, *TorqueDataWrist**. Keenam *array* tersebut kemudian disimpan ke dalam sebuah *file* bertipe ASCII untuk diolah lebih lanjut menggunakan Excel. Pada percobaan ini, kecepatan dari masing-

masing *link* mengikuti *update* dari kecepatan maksimum dari link pertama seperti ditunjukkan pada tabel 1.



Gambar 4. Hasil simulasi ODE yang menghasilkan perhitungan Torsi selama lengan robot bergerak dari 0° hingga 100°

Tujuannya adalah untuk mengetahui torsi maksimum yang diperlukan motor untuk menggerakkan masing-masing link dari robot tersebut. Berikut ini adalah hasil simulasi dari ODE dengan menggunakan data-data Robot Movemaster EX RV-M1.

Dari hasil simulasi oleh ODE seperti ditunjukkan pada gambar 4, terlihat bahwa masing-masing *joint* akan menghasilkan torsi yang berbeda untuk menjaga supaya lengan robot tetap berada pada kondisi lurus. Terlihat bahwa *link*-1 (lengan atas dari robot) membutuhkan torsi awal yang besar namun kemudian menurun landai seiring dengan pergerakan dari *joint-joint* yang lain. Sedangkan *link*-2 terlihat justru mengalami kenaikan torsi pada saat lengan robot berada pada posisi tengah. Ini menunjukkan bahwa *joint* yang terhubung dengan lengan depan (*forearm*) membutuhkan tenaga yang lebih untuk melawan gravitasi. Seperti disebutkan dalam [4], lengan depan memiliki sudut putaran awal sebesar 110° dan juga terhubung dengan *link*-3 (pergelangan) yang dapat bergerak bebas ±90°. Kondisi ini memaksa *link*-2 bekerja lebih berat dari sebelumnya. Sedangkan *link*-3 (pergelangan) memiliki distribusi torsi yang hampir merata dan lebih kecil dari *link-link* yang lain (sesuai dengan dimensi dan distribusi massa-nya). Berikut ini intisari dari torsi maksimum yang dibutuhkan oleh masing-masing *joint* untuk menggerakkan *link* dengan kecepatan 72°/s atau 1.26 rad/s.

Informasi tentang torsi maksimum seperti di atas sangat penting dalam disain robot sesungguhnya, khususnya dalam pemilihan tipe dan kapasitas motor yang digunakan serta perbandingan/*ratio* dari gigi jika menggunakan *gearbox*.

Tabel 2. Torsi maksimum masing-masing joint pada kecepatan 1.26 rad/s

Joint	**Torsi max ($g.m^2.s^{-2}$)**
Bahu (shoulder)	2038.811
Siku (elbow)	3101.79
Pergelangan (wrist)	752.539

KESIMPULAN

Sebuah metode alternatif untuk analisa dinamik sebuah robot lengan dengan menggunakan perangkat lunak *open source* telah dipaparkan. Dalam paper ini, perangkat lunak yang digunakan adalah ODE (*open dynamic engine*) yang banyak digunakan dalam aplikasi-aplikasi simulasi dan pemodelan sistem dinamika yang membutuhkan derajat akurasi dan presisi yang tinggi sementara proses kalkulasinya dilakukan hampir *real time*. Dengan menggunakan ODE, sebuah robot lengan dapat dimodelkan dan disimulasikan dengan sederhana dan menghasilkan hasil yang akurat. Robot lengan yang dimodelkan dan disimulasikan dalam paper ini adalah Movemaster EX RV-M1 dari Mitsubishi. Simulasi dilakukan setelah model robot selesai dibuat dan simulasi dijalankan dengan membuat supaya semua *link* robot bergerak serempak membentuk garis lurus dan berputar dari sudut 0° hingga 100°. Dari hasil pemodelan dan simulasi didapatkan bahwa masing-masing *revolute joint* dari robot tersebut membutuhkan torsi yang berbeda-beda dan ditemukan bahwa pada kecepatan 1.26 rad/s, *joint* siku (*elbow*) membutuhkan torsi yang lebih besar untuk menggerakkan lengan depan (*forearm*) dari robot, yaitu sebesar 3101.79 $g.m^2.s^{-2}$. Torsi yang besar ini tidak hanya diakibatkan karena beban *link* yang harus diangkat, tetapi juga *constraint force* yang harus dilawan oleh *joint* siku saat bergerak bersama-sama dengan *joint* yang lain. Untuk penelitian ke depan, akan lebih baik jika hasil simulasi juga bisa ditampilkan dengan menggunakan perangkat lunak *open source* lain seperti OGRE (*Object-Oriented Graphics Rendering Engine*) ataupun Irrlich.

DAFTAR PUSTAKA

[1] Indar Sugiarto, "Simulasi dan Pemodelan Fisika-Mekanika Menggunakan Perangkat Lunak Open Source", *Prosiding Seminar Nasional Basic Science 6*. Universitas Brawijaya, Malang. 2009.
[2] ODE website (http://www.ode.org)
[3] David Baraff, "Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies", *Computer Graphics*. 1989. Vol. 23, no. 3:223–232.

[4] Mitsubishi, "Industrial Micro-Robot System Model RV-M1", *Instruction Manual*. 1992.
[5] J. Świder, K. Foit, G. Wszołek, D. Mastrowski, "The Off-line Programming and Simulation Software for the Mitsubishi Movemaster RV-M1 Robot", *Journal of Achievement in Materials and Manufacturing Engineering*. Vol 20 (2):499-502. 2007.
[6] Man Zhihong, *Robotics for Computer Engineering Students*, Singapore: Prentice Hall–Pearson Education South Asia Pte Ltd. 2004.