PROCEEDINGS OF SCIENCE

Improving Polynomial-filtered Hybrid Monte Carlo With Hasenbusch

Taylor Haar*
CSSM, Department of Physics, The University of Adelaide, Adelaide, SA, Australia 5005
E-mail: taylor.haar@adelaide.edu.au

Waseem Kamleh
CSSM, Department of Physics, The University of Adelaide, Adelaide, SA, Australia 5005
E-mail: waseem.kamleh@adelaide.edu.au

James Zanotti
CSSM, Department of Physics, The University of Adelaide, Adelaide, SA, Australia 5005
E-mail: james.zanotti@adelaide.edu.au

Yoshifumi Nakamura
RIKEN Advanced Institute for Computational Science, Kobe, Hyogo 650-0047, Japan
E-mail: nakamura@riken.jp

The predominant method for generating Lattice QCD configurations is Hybrid Monte Carlo (HMC). In order to speed up this generation, a wide range of preconditioning techniques that modify the lattice action have been devised. This work compares the performance of the well-known Hasenbusch preconditioning technique with the polynomial filtering technique on a small $16^3 \times 32$ lattice with two flavours of Wilson fermions at a pion mass $M_\pi \sim 400$ MeV. We explore a novel method of combining polynomial and Hasenbusch filters, revealing a speedup when compared to the standard two Hasenbusch filters. This comes with the added advantage of simplified tuning.

The 26th International Nuclear Physics Conference
11-16 September, 2016
Adelaide, Australia

*Speaker.

PoS(INPC2016)319

1. Introduction and Motivation

Lattice QCD is the non-perturbative method of choice when dealing with strong interactions. In order to measure observables on the lattice, we numerically evaluate path integrals via an ensemble average over a large number of configurations, each of which are described by the state of the gauge field U and fermion field ψ. In order to generate these configurations, we most commonly use Hybrid Monte Carlo (HMC), which involves repeated inversions of the Dirac matrix M that describes the strong force between fermions at any two lattice sites. The large number of inversions required and the size of the matrix involved mean that HMC is very computational intensive, making it difficult to simulate near physical quark masses.

A large variety of optimization techniques have been applied to HMC. These techniques reduce the number of matrix inversions required or improve the condition number of the fermion matrix such that the overall cost is reduced. In our work [1], we examine mass preconditioning [2] and polynomial filtering [3] on a $16^3 \times 32$ lattice to compare their performance, before investigating a more novel technique where these two methods are combined. In section 2, we explain the formulation of both methods before the initial comparison in section 3. Then in section 4 we investigate the performance benefits of using polynomial filters on top of mass preconditioners.

2. Theory

The standard method for generating lattice configurations with dynamical fermions is Hybrid Monte Carlo (HMC). The starting point is the lattice action

$$S = S_G[U] + S_F[U, \psi, \overline{\psi}], \tag{2.1}$$

consisting of a gluonic part S_G that depends purely on the gauge fields U and a fermionic part S_F that also depends on the fermion fields $\psi, \overline{\psi}$. We wish to find configurations $(U, \psi, \overline{\psi})$ distributed according to $\exp(-S[U, \psi, \overline{\psi}])$. To do this, we first use Wick's theorem to convert the fermions field ψ to bosonic pseudofermion fields ϕ to make them computationally friendly, modifying the fermion action in the process. We then need to sample ϕ from the distribution $\exp[-S_F(U, \phi, \phi^\dagger)]$, which can be done in practise by relating to Gaussian noise vectors χ distributed according to $\exp[-\chi^\dagger \chi]$. To generate correctly distributed gauge fields U, we introduce a fictitious conjugate momentum field P distributed according to $\exp[-\sum \mathrm{tr}[P^2]]$, then produce configurations which preserve the Hamiltonian

$$H[U, \phi, \phi^\dagger] = \sum \mathrm{tr}[P^2] + S[U, \phi, \phi^\dagger] \tag{2.2}$$

by using Hamilton's equations, leading to integration steps

$$\hat{T}[\varepsilon]{:}(P, U) \to (P, e^{i\varepsilon P} U) \tag{2.3a}$$

$$\text{and} \quad \hat{S}[\varepsilon]{:}(P, U) \to (P - \varepsilon F, U), \tag{2.3b}$$

where $F = \frac{\partial S}{\partial U}$ is the force term. We combine a sequence of these steps into a trajectory to take an existing configuration (P, U) and produce a new state (P', U'), such that we get to a new candidate configuration (U', ϕ'). This candidate undergoes a Metropolis acceptance step to ensure detailed balance and hence that the process will converge to the target distribution $\exp[-S]$.

The main cost arises from the fermionic force term, which in the two-flavour degenerate case $S_F = \phi^\dagger (M^\dagger M)^{-1} \phi$ takes the form

$$\frac{\partial S_F}{\partial U} = \frac{\partial}{\partial U}[\phi^\dagger (M^\dagger M)^{-1} \phi] = -\phi^\dagger (M^\dagger M)^{-1} \frac{\partial M^\dagger M}{\partial U} (M^\dagger M)^{-1} \phi. \tag{2.4}$$

The issue here is the inversion of the fermion matrix $K \equiv M^\dagger M$. It is a very large, sparse matrix, so it takes iterative solvers (e.g. conjugate gradient) many matrix multiplications to invert. It can also give rise to large force terms, which then necessitates a reduction in the step-size to keep a good Metropolis acceptance rate and thus increases the number of required inversions. Therefore, modern lattice simulations use a variety of HMC improvements to reduce the force terms (thus reducing the number of required inversions) or to make the matrix K easier to invert.

A template for achieving a reduction in computational cost was proposed by [4], which considers splitting the action into two terms

$$S = S_{UV} + S_{IR}, \tag{2.5}$$

such that S_{UV} captures the high-energy modes ($\sim$ large forces) of the action whilst S_{IR} captures the low-energy modes ($\sim$ small forces), and F_{UV} is relatively cheap to calculate compared to F_{IR}. Such terms can then be placed on different time-scales using a multiple time-scale integrator, which allows the expensive F_{IR} to be evaluated less often and hence improve the cost.

One of the standard HMC improvement techniques is mass preconditioning (MP) [2], where we factorize the fermion action S_F into two terms as follows:

$$S_{MP} = \phi_1^\dagger J^{-1} \phi_1 + \phi_2^\dagger J K^{-1} \phi_2. \tag{2.6}$$

Here, J is a fermion matrix just like K, but with a modified mass parameter $\kappa' < \kappa$ giving rise to a 'heavier' quark mass. The first action term S_1 captures the high-energy modes but has a cheaper force term than S_2, so we can use multiple time-scales to reduce the overall cost.

We compare this technique to polynomial filtering (PF) [3], whereby the fermion action is filtered via

$$S_{PF} = \phi_1^\dagger P(K) \phi_1 + \phi_2^\dagger [P(K) K]^{-1}, \tag{2.7}$$

where $P(K)$ is a low-order polynomial approximating K^{-1}. In our work, we use Chebyshev polynomials such that the only parameter to tune is the polynomial order p. The polynomial filter term S_1 has a very cheap force term due to a lack of matrix inversions, and it captures the high-energy modes of the system by virtue of $P(K)$ approximating the inverse. Hence, we can put S_1 and S_2 on separate time-scales to achieve a cost reduction.

3. Comparison of filtering methods

3.1 Setup

Our initial study compares the computational cost of polynomial filtering against mass preconditioning. We use a $n_f = 2$, $16^3 \times 32$ lattice with a Wilson fermion action with hopping parameter $\kappa = 0.15825$, and even-odd preconditioning. This lattice has a pion mass of ~ 400 MeV and a lattice spacing of ~ 0.08 fm.

The metric we use for the cost is

$$C = \frac{N_{mat}}{P_{acc}} \tag{3.1}$$

where N_{mat} is the number of fermion matrix K multiplications per trajectory and P_{acc} is the Metropolis acceptance rate. We average this quantity over at least 2000 trajectories for each displayed data point in the graphs that follow.

As noted in equation (2.5), we split the different action terms onto different integration time-scales. By way of example, the single polynomial filter action (1PF) takes the form

$$S = S_G + \phi_1^\dagger P(K)\phi_1 + \phi_2^\dagger [P(K)K]^{-1}\phi_2, \tag{3.2}$$

and we set the step-sizes $h_0 = h_G < h_1 < h_2$ to reflect the relative cost and frequency scales. The corresponding number of steps n_i at each scale are given by the relation $\tau = h_i n_i$ where τ is the trajectory length; we have fixed $\tau = 1$ as is standard. We use a generalized integration scheme (see section 3.2) that gives a very flexible choice of step-sizes, and the 'balanced forces' method to tune these step-sizes. See the paper [1] for further details.

3.2 Generalized multi-scale integration

To create multiple time-scales in a HMC integration, one typically uses a nested integration scheme. Finer integration scales are built by substituting integration schemes for the time updates in the original integration scheme. For example, consider the space-time-space leapfrog integrator

$$I_{LPF}[\tau] = \left(\hat{S}\left[\tfrac{h}{2}\right]\hat{T}\left[h\right]\hat{S}\left[\tfrac{h}{2}\right]\right)^n, \tag{3.3}$$

where $n = \tau/h$ and $\hat{T}, \hat{S}$ are as defined in (2.3a), (2.3b). To add a nested integration scale to this scheme, we can replace the time updates $\hat{T}[h]$ with m leapfrog steps in the second term S_2:

$$I_2[h] = \left(\hat{S}_2\left[\tfrac{h}{2}\right]\hat{T}[h]\hat{S}_2\left[\tfrac{h}{2}\right]\right)^m, \tag{3.4}$$

$$\text{s.t.} \quad I_{\text{nested}}[\tau] = \left(\hat{S}_1\left[\tfrac{h_1}{2}\right] I_2[h_1]\hat{S}_1\left[\tfrac{h_1}{2}\right]\right)^n. \tag{3.5}$$

This ensures that S_1 is integrated with step-size $h_1 = \tau/n$ and S_2 is integrated with finer step-size $h_2 = h_1/m$. However, nested schemes force the step-sizes of each scale to evenly divide those on each coarser scale.

We have devised an generalized scheme for constructing multiple integration time-scales without any relative step-size restriction. The basic idea is to note that when we integrate some Hamiltonian $H = T + S = T + S_1 + S_2 + \ldots$, we always have just one kind of time step $\hat{T}[h]$ that is applied in a uniform direction $h > 0$. It thus makes sense to parametrize the progress of time-steps via a time parameter t that increases monotonically from 0 to τ. We can then combine integration schemes for Hamiltonians for each action term $H_i = T + S_i$ into a generalized scheme for the full Hamiltonian $H = T + S$ by integrating with time steps from 0 to τ, inserting the action step updates $\hat{S}_i[h]$ at their respective 'times' t in the composite integrators. For example, consider a two-step and a three-step leapfrog integrator for the two action terms:

$$I_1[\tau] = \hat{S}_1\left[\tfrac{\tau}{4}\right]\hat{T}\left[\tfrac{\tau}{2}\right]\hat{S}_1\left[\tfrac{\tau}{2}\right]\hat{T}\left[\tfrac{\tau}{2}\right]\hat{S}_1\left[\tfrac{\tau}{4}\right], \tag{3.6a}$$

$$\text{and} \quad I_2[\tau] = \hat{S}_2\left[\tfrac{\tau}{6}\right]\hat{T}\left[\tfrac{\tau}{3}\right]\hat{S}_2\left[\tfrac{\tau}{3}\right]\hat{T}\left[\tfrac{\tau}{3}\right]\hat{S}_2\left[\tfrac{\tau}{3}\right]\hat{T}\left[\tfrac{\tau}{3}\right]\hat{S}_2\left[\tfrac{\tau}{6}\right]. \tag{3.6b}$$

We can combine these two schemes via the generalized scheme by overlaying the space updates based on their position in 'time'. This produces the integrator

$$I[\tau] = \hat{S}_1\left[\tfrac{\tau}{4}\right]\hat{S}_2\left[\tfrac{\tau}{6}\right]\hat{T}\left[\tfrac{\tau}{3}\right]\hat{S}_2\left[\tfrac{\tau}{3}\right]\hat{T}\left[\tfrac{\tau}{6}\right]\hat{S}_1\left[\tfrac{\tau}{2}\right]\hat{T}\left[\tfrac{\tau}{6}\right]\hat{S}_2\left[\tfrac{\tau}{3}\right]\hat{T}\left[\tfrac{\tau}{3}\right]\hat{S}_2\left[\tfrac{\tau}{6}\right]\hat{S}_1\left[\tfrac{\tau}{4}\right] \tag{3.7}$$

Further details are given in [1], where we also prove that this method produces an integration scheme that meets the requirements for detailed balance, given that the composite integrators also do so.

3.3 Results

Figure 1 shows the cost function C (3.1) for a single polynomial filter and for a single mass preconditioner. As one can see, the mass preconditioned action performs much better: a cost of $C = 43,800 \pm 3,500$ at $\kappa' = 0.1545$ compared with $C = 87,500 \pm 7,400$ at $p = 10$. This suggests that a short order polynomial of order 10 can't capture as much of the dynamics as a mass preconditioner that requires 80 or more iterations to invert.

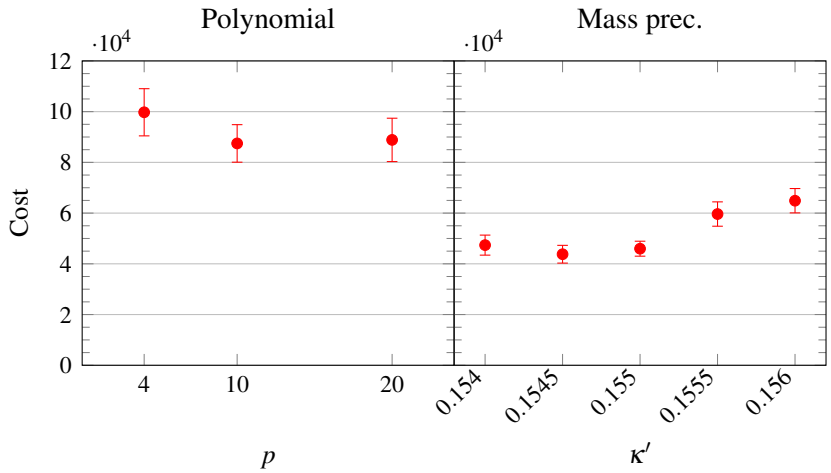

Figure 1: The simulation cost function (3.1) for (single filter) polynomial filtering and mass preconditioning.

We can improve the performance of both methods by using two filters instead of one. In the case of polynomial filtering, we take two polynomials $P_1(K)$ and $P_2(K)$ with orders $p_2 > p_1$ that factorize into a polynomial $Q(K) = P_2(K)/P_1(K)$, such that we can use action

$$S_{2PF} = \phi_1^\dagger P_1(K)\phi_1 + \phi_2^\dagger Q(K)\phi_2 + \phi_3^\dagger [P_2(K)K]^{-1}\phi_3 \tag{3.8}$$

and thus keep the cost of evaluating the intermediate force F_2 low. In the case of mass preconditioning, we use two modified mass parameters $\kappa_1 < \kappa_2 < \kappa$, and action

$$S_{2MF} = \phi_1^\dagger J_1^{-1}\phi_1 + \phi_2^\dagger J_1 J_2^{-1}\phi_2 + \phi_3^\dagger J_2 K^{-1}\phi_3. \tag{3.9}$$

To keep the parameter space manageable, we fixed the cheapest term S_1 in both cases for our analysis – namely, $p_1 = 4$ and $\kappa_1 = 0.145$.

Figure 2 shows the cost of these two methods side by side. In both cases we see improved results over the one-filter case. Two polynomial filters reduce the cost approximately as much as a single mass preconditioner: a cost of $C = 47,700 \pm 3,700$ at $p_2 = 54$. Two mass preconditioners reduce the cost further, to $C = 31,100 \pm 2,200$ at $\kappa_2 = 0.1555$.

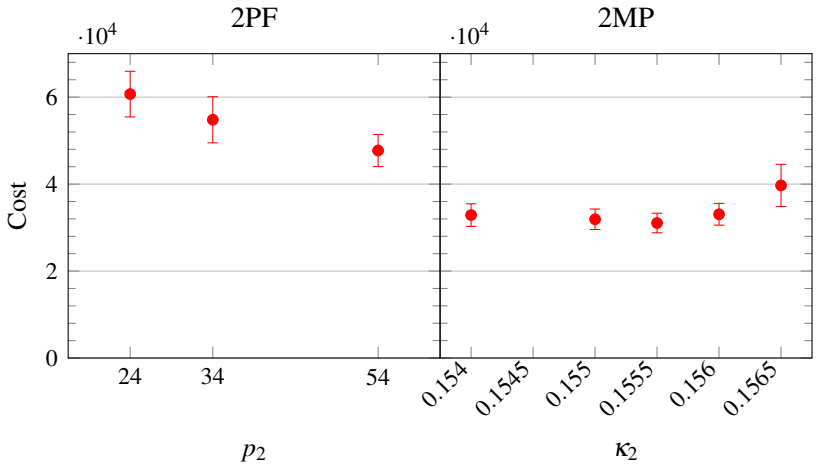

Figure 2: The cost function (3.1) for 2 filter actions, comparing polynomial filtering (2PF) and mass-preconditioning (2MP).

4. Combined filters

4.1 Combining polynomial filtering and mass-preconditioning

It is informative to compare the two techniques' relative efficiency. Polynomial filtering works better at capturing the high-frequency (energy) modes of the system, because we can then use a low polynomial order. To capture lower frequency modes, we have to increase the polynomial order and the cost to evaluate the force terms increases dramatically. On the other hand, Hasenbusch filtering is best suited to small changes $\Delta\kappa = \kappa' - \kappa$ in the quark mass, because this implies that $JK^{-1} \approx I$ and hence the correction term is easier to invert. However, the filter term $\phi_1^\dagger J^{-1}\phi_1$ would then have a more expensive force term due to the lighter mass.

To combine the benefits of both of these methods, we place a polynomial filter on top of a mass preconditioner, resulting in the action

$$S_{\text{PF-MP}} = \phi_1^\dagger P(J)\phi_1 + \phi_2^\dagger [P(J)J]^{-1}\phi_2 + \phi_3^\dagger JK^{-1}\phi_3 \tag{4.1}$$

where $J = K(\kappa')$, $\kappa' > \kappa$ and $P(J)$ is a polynomial. The idea here is that the high frequency modes are captured by the cheap polynomial $P(J)$, such that the relative mass of the Hasenbusch correction term $\Delta\kappa$ can be kept small.

We tested this on our lattice with the polynomial $P(K)$ fixed to order $p = 4$ and κ' varying. The cost function for various κ' is shown in Figure 3 in the second column, which can be compared to the 2MP action in the first column. As the graph shows, the cost for PF-MP is very similar to that of 2MP.

4.2 3-filter actions

Note that in the cost function for 2MP and PF-MP, the cost increases significantly depending on the choice of the intermediate term parameter κ_2/κ', so a degree of tuning is required to achieve a minimum. This is not a cheap procedure, as we need at least 500 trajectories to get a decent handle on the acceptance rate P_{acc} for each set of parameters and hence the cost. This is complicated by

the fact that the Hasenbusch parameter κ_2/κ' is a real parameter whose optimal value is heavily dependent on the quark mass κ, so we'd have to tune it again for different quark masses. This makes it difficult to optimize an action with three mass preconditioners effectively, because there are too many parameters $\kappa_1, \kappa_2, \kappa_3$ which require fine tuning.

On the other hand, polynomial filtering only depends on a single integral parameter p once a class of polynomials is chosen, and the same polynomial order should filter out similar proportions of the action regardless of κ. Hence, a polynomial filtering term requires very little to no tuning once a good set of polynomials are found. We can thus add more polynomial filters to our action without increasing the time it takes to tune.

We consider one polynomial filter on top of two mass preconditioners (1PF-2MP)

$$S_{1PF-2MP} = \phi_1^\dagger P(J_1)\phi_1 + \phi_2^\dagger [P(J_1)J_1]^{-1}\phi_2 + \phi_3^\dagger J_1 J_2^{-1}\phi_3 + \phi_4^\dagger J_2, K^{-1}\phi_4 \tag{4.2}$$

where we fix $p = 4$, $\kappa_1 = 0.145$; and two polynomial filters on top of a single mass preconditioner (2PF-1MP)

$$S_{2PF-1MP} = \phi_1^\dagger P(J)\phi_1 + +\phi_2^\dagger Q(J)\phi_2 + \phi_3^\dagger [P_2(J)J]^{-1}\phi_3 + \phi_4^\dagger JK^{-1}\phi_4, \tag{4.3}$$

where we fix $p = 4$, $q = 20$. The cost function for these two actions are shown in columns three and four respectively in Figure 3.

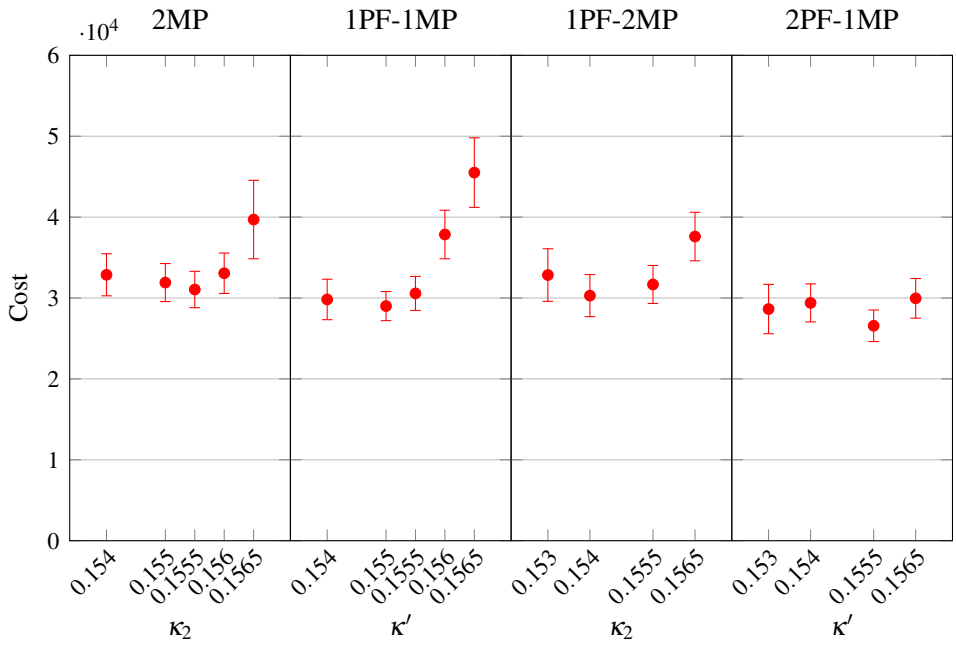

Figure 3: The cost function (3.1), comparing mass preconditioning (2MP) with polynomial filtered mass preconditioning (1PF-1MP, 2PF-1MP, 1PF-2MP)

As can be seen in Figure 3, the optimal cost for the two three-filter actions are very similar to those for PF-MP and 2MP. The extra polynomial filter in 1PF-2MP on top of 2MP has no effect on the performance. However, for 2PF-1MP the cost has a much lower dependence on the choice of the Hasenbusch parameter κ, as can be seen at the $\kappa' = 0.1565$ data point. Since the polynomial

filter orders (p,q) don't require much tuning, this indicates that the 2PF-1MP action doesn't require much tuning to reach close-to-optimum computational costs. In fact, it requires even less tuning than 2MP.

5. Conclusion and Outlook

We have compared the performance of polynomial filtering and mass preconditioning on a modest $16^3 \times 32$ lattice before combining the two techniques to try to achieve lower costs. Whilst we didn't observe any significant improvements in cost over the 'standard' 2MP algorithm, we did find that 2PF-1MP cost much less to tune due to the cost's low dependence on κ' and the fact that the polynomial filters don't require very much tuning. It remains to be seen if this behaviour still holds for larger lattices and for smaller quark masses.

References

[1] T. Haar, W. Kamleh, J. Zanotti and Y. Nakamura, *Applying polynomial filtering to mass preconditioned Hybrid Monte Carlo*, 1609.02652.

[2] M. Hasenbusch and K. Jansen, *Speeding up the HMC: QCD with clover improved Wilson fermions*, *Nucl.Phys.Proc.Suppl.* **119** (2003) 982–984, [hep-lat/0210036].

[3] W. Kamleh and M. Peardon, *Polynomial Filtered HMC: An Algorithm for lattice QCD with dynamical quarks*, *Comput.Phys.Commun.* **183** (2012) 1993–2000, [1106.5625].

[4] TrinLat Collaboration collaboration, M. J. Peardon and J. Sexton, *Multiple molecular dynamics time scales in hybrid Monte Carlo fermion simulations*, *Nucl.Phys.Proc.Suppl.* **119** (2003) 985–987, [hep-lat/0209037].

PoS(INPC2016)319